Ameba-ZII Peripheral Verification Note

This document provides information for usage of peripheral examples.

COPYRIGHT

DISCLAIMER

TRADEMARKS

USING THIS DOCUMENT

This document is intended for the software engineer's reference and provides detailed programming information.

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

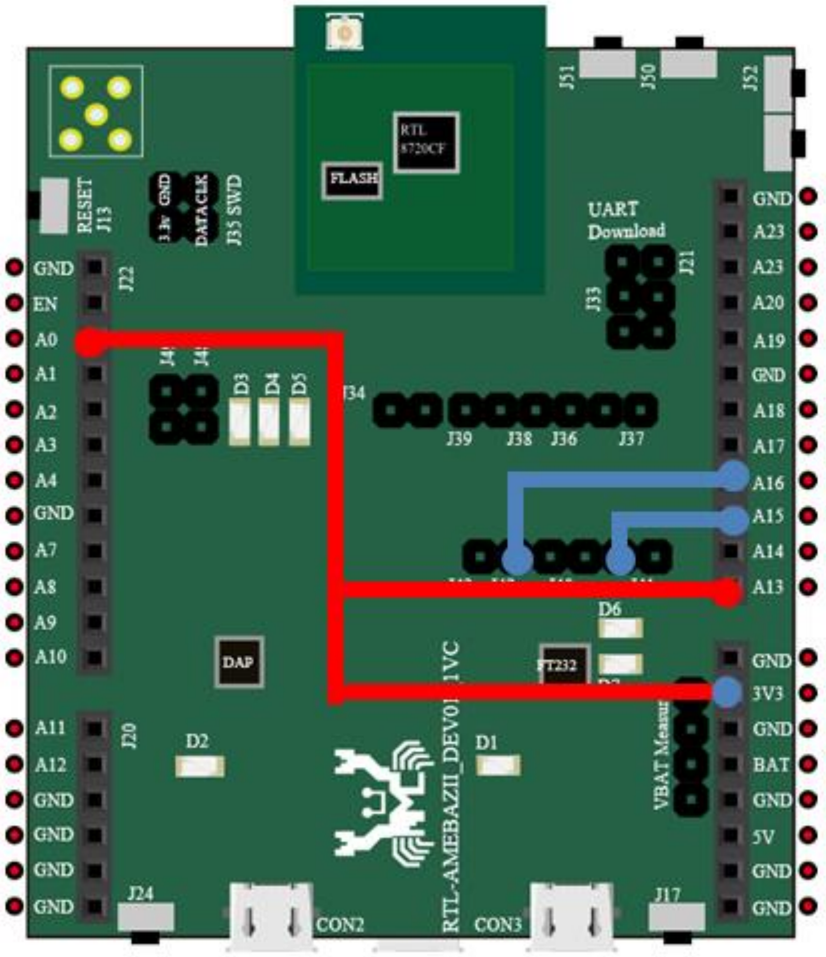
REVISION HISTORY

Revision	Release Date	Summary
0.1	2019/01/04	Initial draft
0.2	2019/01/21	Draft updates
0.3	2019/02/19	Add EVB V2.0 boards
0.4	2019/03/01	Add example “flash_erase_dword”, “pm_deepsleep”, “pm_sleepcg”, and “pm_standby”
0.5	2019/03/18	Add other wake up source of example “pm_sleepcg”, and “pm_standby”
0.6	2019/04/29	Update examples to avoid conflicts when enabled JTAG/SWD
0.7	2019/05/02	Update figures of UART examples
0.8	2019/06/28	Fix documents format errors

Table of Contents

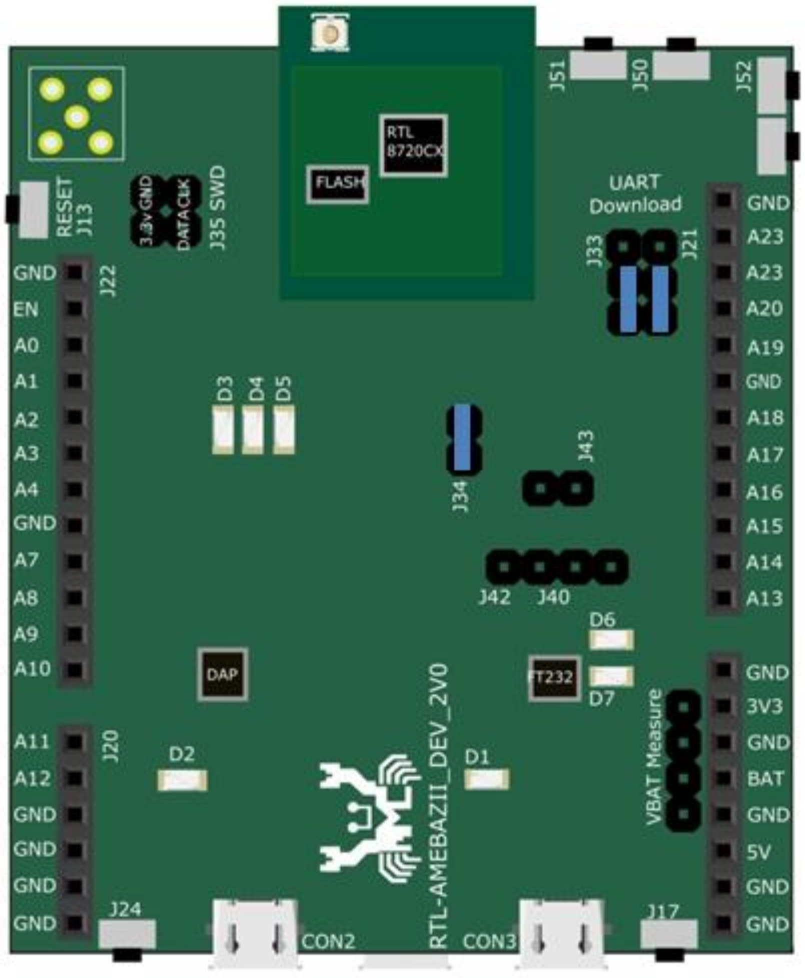
1 EVB boards initial connections

1.1 EVB V1.0



For EVB V1.0 board there are jumpers wires need to be added before starting use the board. For log UART usage, Pin A15 and Pin A16 need to connect with J41 Pin1 and J43 Pin1. For entering the Download Mode, there are two steps. The first step is connecting Pin A0, Pin A13 to 3.3V. The second step is pressing the RESET button then the system will enter the Download Mode. Please note that after entering the Download Mode Pin A0 and Pin A13 should NC to 3.3V, otherwise the next system reboot will fail and enter the Download Mode again.

1.2 EVB V2.0



For EVB V2.0 board there are jumpers need to be added before starting use the board. For log UART usage, J33 Pin3 and Pin5 should be connected by a jumper; J33 Pin6 and Pin4 should be connected by a jumper. For the Download Mode usage, J34 Pin1 and Pin2 should be connected by a jumper. Please follow the steps next to enter the Download Mode. Press and hold the UART Download button (J26), then press the RESET button (J13) and release both J26 and J13.

2 Examples

2.1 CRYPTO

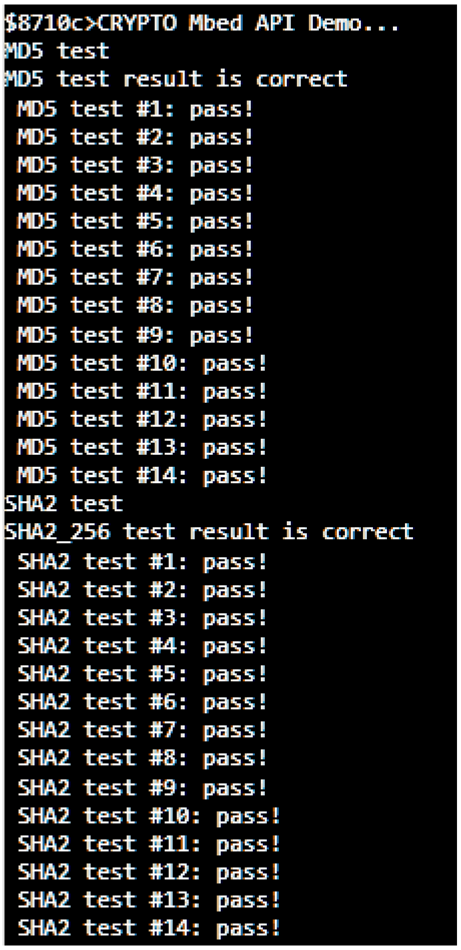
2.1.1 crypto

IMAGE NAME:

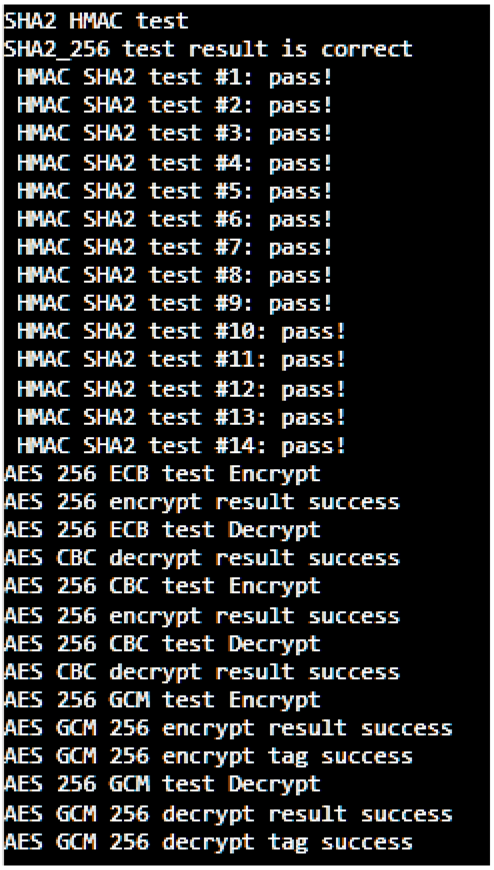
flash_is_crypto.bin

This example describes how to use CRYPTO function.

There is no need to connect wires. It verifies basically hardware encryption/decryption functions and show results on the LOG_OUT. (Using thread function so print arrangement may be different)



```
$8710c>CRYPTO Mbed API Demo...
MD5 test
MD5 test result is correct
 MD5 test #1: pass!
 MD5 test #2: pass!
 MD5 test #3: pass!
 MD5 test #4: pass!
 MD5 test #5: pass!
 MD5 test #6: pass!
 MD5 test #7: pass!
 MD5 test #8: pass!
 MD5 test #9: pass!
 MD5 test #10: pass!
 MD5 test #11: pass!
 MD5 test #12: pass!
 MD5 test #13: pass!
 MD5 test #14: pass!
SHA2 test
SHA2_256 test result is correct
 SHA2 test #1: pass!
 SHA2 test #2: pass!
 SHA2 test #3: pass!
 SHA2 test #4: pass!
 SHA2 test #5: pass!
 SHA2 test #6: pass!
 SHA2 test #7: pass!
 SHA2 test #8: pass!
 SHA2 test #9: pass!
 SHA2 test #10: pass!
 SHA2 test #11: pass!
 SHA2 test #12: pass!
 SHA2 test #13: pass!
 SHA2 test #14: pass!
```



```
SHA2 HMAC test
SHA2_256 test result is correct
 HMAC SHA2 test #1: pass!
 HMAC SHA2 test #2: pass!
 HMAC SHA2 test #3: pass!
 HMAC SHA2 test #4: pass!
 HMAC SHA2 test #5: pass!
 HMAC SHA2 test #6: pass!
 HMAC SHA2 test #7: pass!
 HMAC SHA2 test #8: pass!
 HMAC SHA2 test #9: pass!
 HMAC SHA2 test #10: pass!
 HMAC SHA2 test #11: pass!
 HMAC SHA2 test #12: pass!
 HMAC SHA2 test #13: pass!
 HMAC SHA2 test #14: pass!
AES 256 ECB test Encrypt
AES 256 encrypt result success
AES 256 ECB test Decrypt
AES CBC decrypt result success
AES 256 CBC test Encrypt
AES 256 encrypt result success
AES 256 CBC test Decrypt
AES CBC decrypt result success
AES 256 GCM test Encrypt
AES GCM 256 encrypt result success
AES GCM 256 encrypt tag success
AES 256 GCM test Decrypt
AES GCM 256 decrypt result success
AES GCM 256 decrypt tag success
```

2.2 FLASH

2.2.1 flash

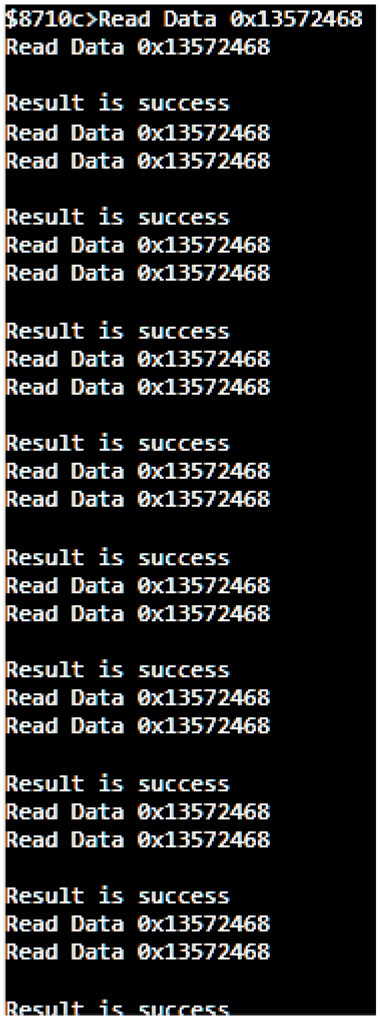
IMAGE NAME:

flash_is_flash.bin

This example describes how to use flash api.

We don’t need connect wires. It verifies hardware DMA functions and show results on the LOG_OUT.

Here is the success log:



```
$8710c>Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
Read Data 0x13572468
Read Data 0x13572468

Result is success
```

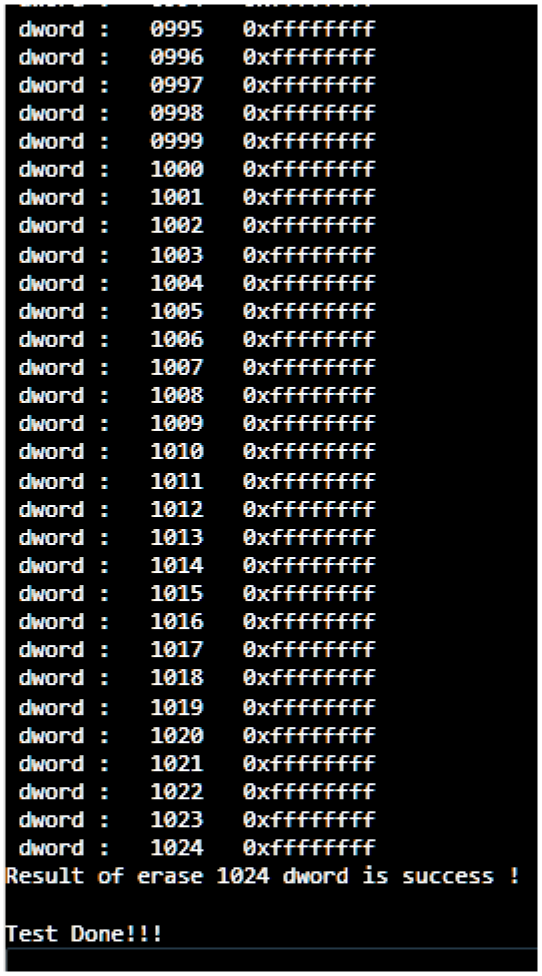
2.2.2 flash_erase_dword

IMAGE NAME:

flash_erase_dword.bin

This example writes specific values in flash and applies the flash_erase_dword function to erase any numbers of dwords that written before.

Here is the success log:



```
 dword :   0995   0xffffffff
 dword :   0996   0xffffffff
 dword :   0997   0xffffffff
 dword :   0998   0xffffffff
 dword :   0999   0xffffffff
 dword :   1000   0xffffffff
 dword :   1001   0xffffffff
 dword :   1002   0xffffffff
 dword :   1003   0xffffffff
 dword :   1004   0xffffffff
 dword :   1005   0xffffffff
 dword :   1006   0xffffffff
 dword :   1007   0xffffffff
 dword :   1008   0xffffffff
 dword :   1009   0xffffffff
 dword :   1010   0xffffffff
 dword :   1011   0xffffffff
 dword :   1012   0xffffffff
 dword :   1013   0xffffffff
 dword :   1014   0xffffffff
 dword :   1015   0xffffffff
 dword :   1016   0xffffffff
 dword :   1017   0xffffffff
 dword :   1018   0xffffffff
 dword :   1019   0xffffffff
 dword :   1020   0xffffffff
 dword :   1021   0xffffffff
 dword :   1022   0xffffffff
 dword :   1023   0xffffffff
 dword :   1024   0xffffffff
Result of erase 1024 dword is success !

Test Done!!!
```

2.2.3 flash_micron_block_protect

Skipped.

2.2.4 flash_setstatus

Skipped.

2.3 GDMA

2.3.1 gdma

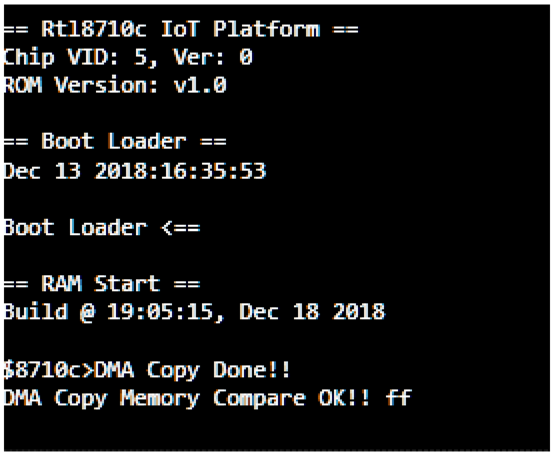
IMAGE NAME:

flash_is_gdma.bin

This example describes how to use GDMA (General Direct Memory Access).

We don’t need connect wires. It verifies hardware DMA functions and show results on the LOG_OUT.

Here is the success log:



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:05:15, Dec 18 2018

$8710c>DMA Copy Done!!
DMA Copy Memory Compare OK!! ff
```

2.4 GPIO

Please note that for all GPIO related examples,

- The supported GPIO pins, PA_0, PA_1, PA_2, PA_3, PA_4, PA_7, PA_8, PA_9, PA_10, PA_11, PA_12, PA_13, PA_14, PA_15, PA_16, PA_17, PA_18, PA_19, PA_20, and PA_23.
- Please off JTAG/SWD, when using PA_0, PA_1, PA_2, PA_3, and PA_4.
- Please off log UART, when using PA_15, and PA_16.
- PA_7, PA_8, PA_9, PA_10, PA_11, and PA_12 only available on RTL8720CF.

2.4.1 gpio

IMAGE NAME:

flash_is_gpio.bin

This example describes how to use GPIO read/write by mbed api.

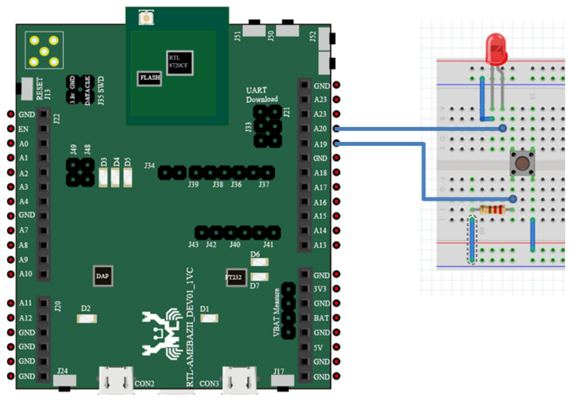
Requirement Components:

1. a LED
2. a push button
3. a resister

Pin name PA_19 and PA_20 map to GPIOA_19 and GPIOA_20:

- PA_19 as input with internal pull-high, connect a resistor to ground in series. PA_19 also connect to a push button and the other side of the push button connected to 3.3V.
- PA_20 as output, connect a LED to GND in series.

In this example, the LED is on/off when the push button is pressed.



2.4.2 gpio_dht_temp_humidity

IMAGE NAME:

flash_is_gpio_dht_temp_humidity.bin

This example describes how to use DHT11/DHT22/DHT21 temperature and humidity sensor.

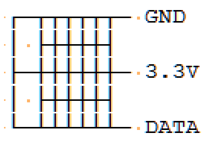
Since DHT require microseconds level GPIO operation, we use GPIO register to perform GPIO read.

Requirement Components:

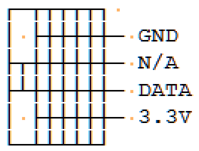
1. DHT11/DHT22/DHT21

DHT series may have 3 pin or 4 pin product.

3 pin DHT has pin layout:



4 pin DHT has pin layout:

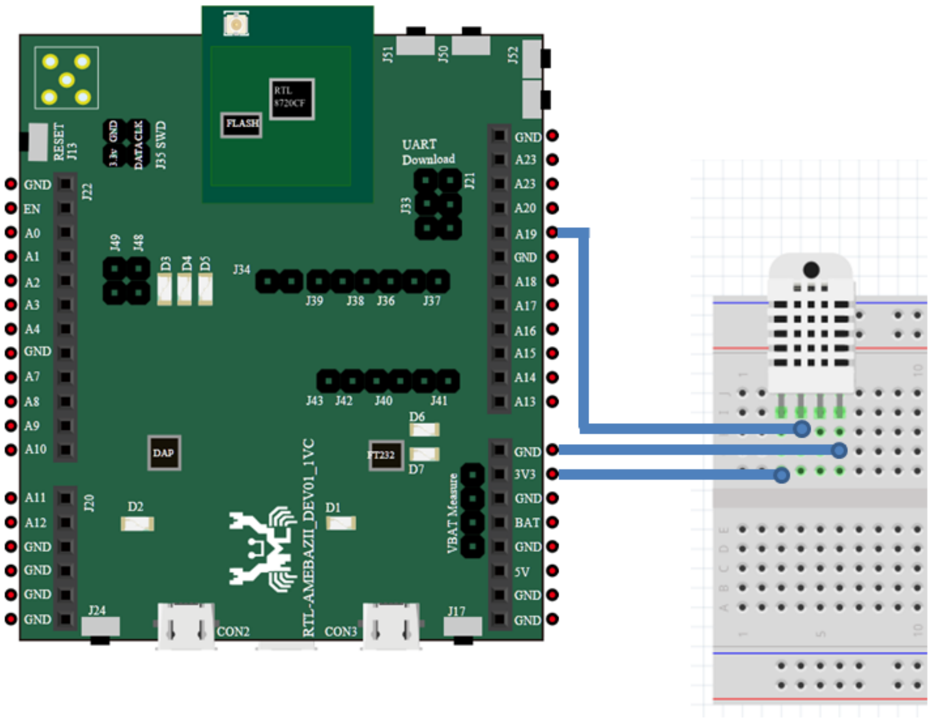


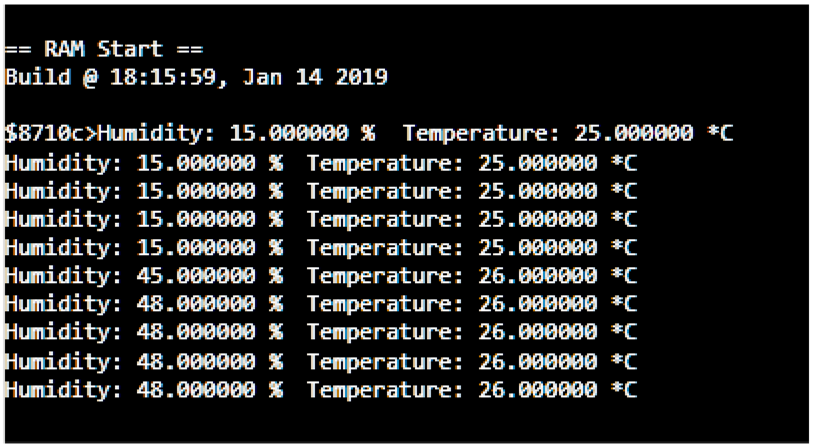
All we need is 3.3V, GND, and DATA (connect to PA_19). DATA has default level high.

To get data, it has 3 stage:

1. Turn DHT from power saving to high speed mode:
   - Ameba toggle low on DATA pin
2. Wait DHT ready:
   - DHT toggle low on DATA pin
3. Repeatly get 40 bits of data:
   - If level high has shorter length than level low, then it's bit 0.
   - If level high has longer length than level high, then it's bit 1.

```
         ______           ________________
________/      \_________/                \__________
         bit 0            bit 1
```





```
== RAM Start ==
Build @ 18:15:59, Jan 14 2019

$8710c>Humidity: 15.000000 %  Temperature: 25.000000 *C
Humidity: 15.000000 %  Temperature: 25.000000 *C
Humidity: 15.000000 %  Temperature: 25.000000 *C
Humidity: 15.000000 %  Temperature: 25.000000 *C
Humidity: 15.000000 %  Temperature: 25.000000 *C
Humidity: 45.000000 %  Temperature: 26.000000 *C
Humidity: 48.000000 %  Temperature: 26.000000 *C
Humidity: 48.000000 %  Temperature: 26.000000 *C
Humidity: 48.000000 %  Temperature: 26.000000 *C
Humidity: 48.000000 %  Temperature: 26.000000 *C
```

2.4.3 gpio_HC_SR04_ultrasonic

IMAGE NAME:

flash_is_gpio_HC_SR04_ultrasonic.bin

This example describes how to use HC-SR04 ultrasonic.

Requirement Components:

1. a HC-SR04 ultrasonic

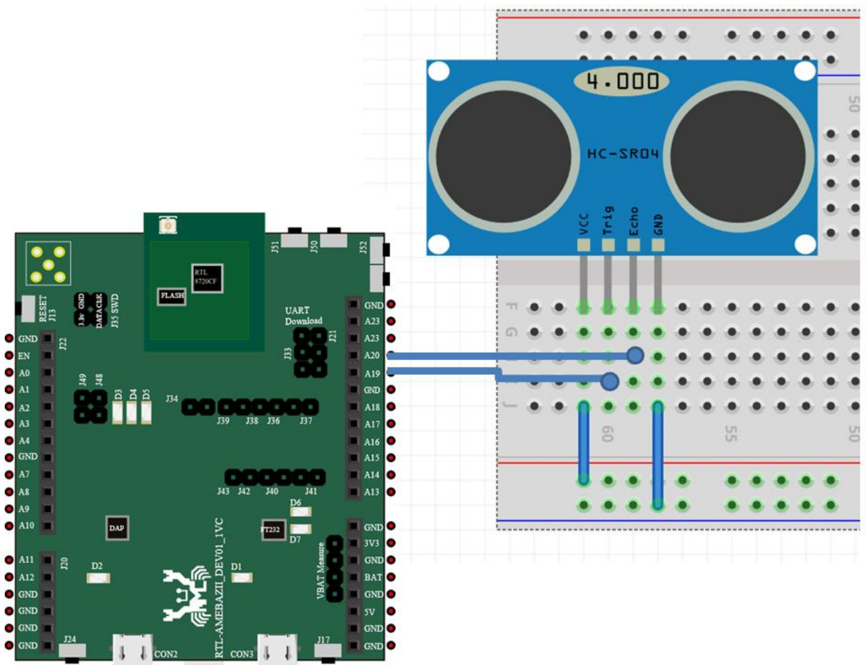
HC-SR04 4 pin connections:

VCC	<---------->	5V
TRIG	<---------->	PA_19
ECHO	<---------->	PA_20 (with level converter from 5V to 3.3V)
GND	<---------->	GND

HC-SR04 use ultrasonic to raging distance. We send a pulse HIGH on TRIG pin for more than 10us, then HC-SR04 return a pulse HIGH on ECHO pin which correspond distance. The speed of sound wave is 340 m/s, which means it takes 29us for 1cm.

Thus the distance of result is:

distance (in cm) = time (in us) / (29 * 2)



2.4.4 gpio_irq

IMAGE NAME:

flash_is_gpio_irq.bin

This example describes how to use GPIO read/write by mbed api.

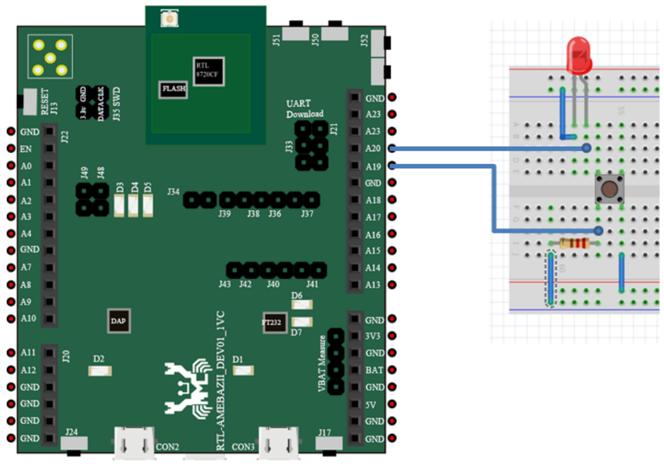
Requirement Components:

1. a LED
2. a push button
3. a resister

Pin name PA_19 and PA_20 map to GPIOA_19 and GPIOA_20:

- PA_19 as input with internal pull-high, connect a resistor to ground in series. PA_19 also connect to a push button and the other side of the push button connected to VCC.
- PA_20 as output, connect a LED to ground in series.

In this example, push the button to trigger interrupt to turn on/off the LED.



2.4.5 gpio_level_irq

IMAGE NAME:

flash_is_gpio_level_irq.bin

This example describes how to implement high/low level trigger on 1 gpio pin.

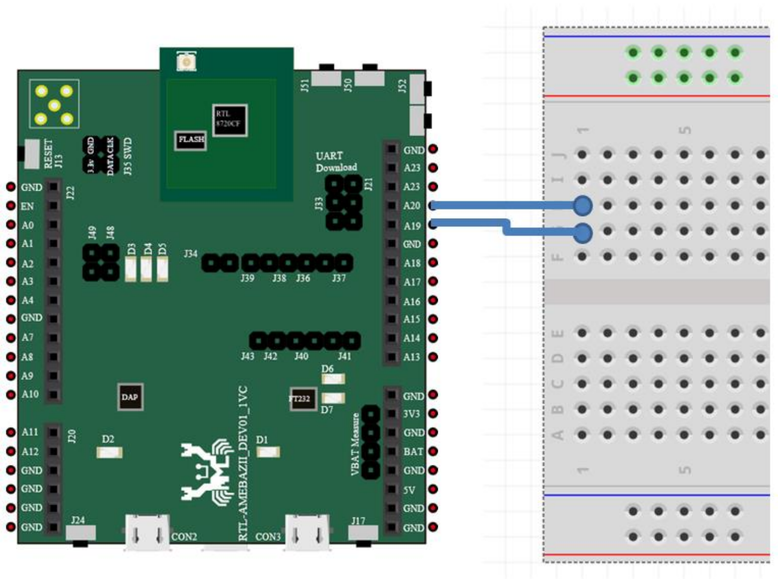
Pin name PA_19 and PA_20 map to GPIOA_19 and GPIOA_20:

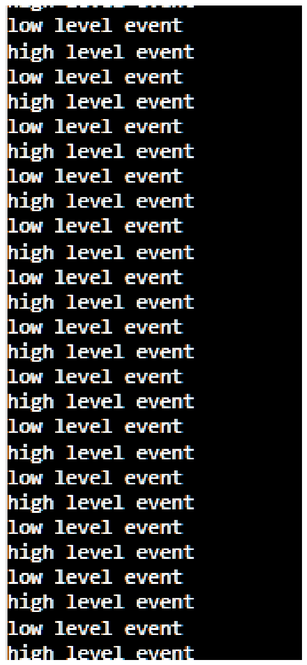
Connect PA_19 and PA_20

- PA_19 as gpio input high/low level trigger.
- PA_20 as gpio output

PA_20 is signal source that change level to high and low periodically. PA_19 setup to listen low-level events in initial. When PA_19 catch low-level events, it disables the irq to avoid receiving duplicate events. (NOTE: the level events will keep invoked if level keeps in same level) Then PA_19 is configured to listen high-level events and enable irq. As PA_19 catches high-level events, it changes back to listen low-level events. Thus, PA_19 can handle both high/low level events.

In this example, you will see log that prints high/low level event periodically.





```
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
low level event
high level event
```

2.4.6 gpio_light_weight

IMAGE NAME:

flash_is_gpio_light_weight.bin

This example describes how to use GPIO read/write in a light weight way.

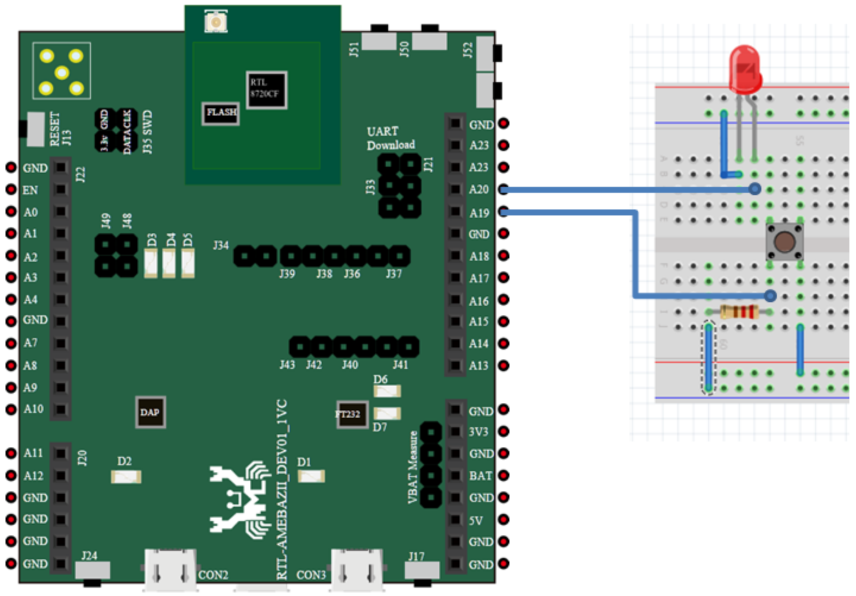
Requirement Components:

1. a LED
2. a push button
3. a resistor

Pin name PA_19 and PA_20 map to GPIOA_19 and GPIOA_20:

- PA_19 as input with internal pull-high, connect a resistor to ground in series. PA_19 also connect to a push button and the other side of the push button connected to VCC.
- PA_20 as output, connect a LED to ground in series.

In this example, the LED is on/off when the push button is pressed.



2.4.7 gpio_port

IMAGE NAME:

flash_is_gpio_port_output.bin;

flash_is_gpio_port_input.bin

This example describes how to use GPIO Port read/write by mbed api.

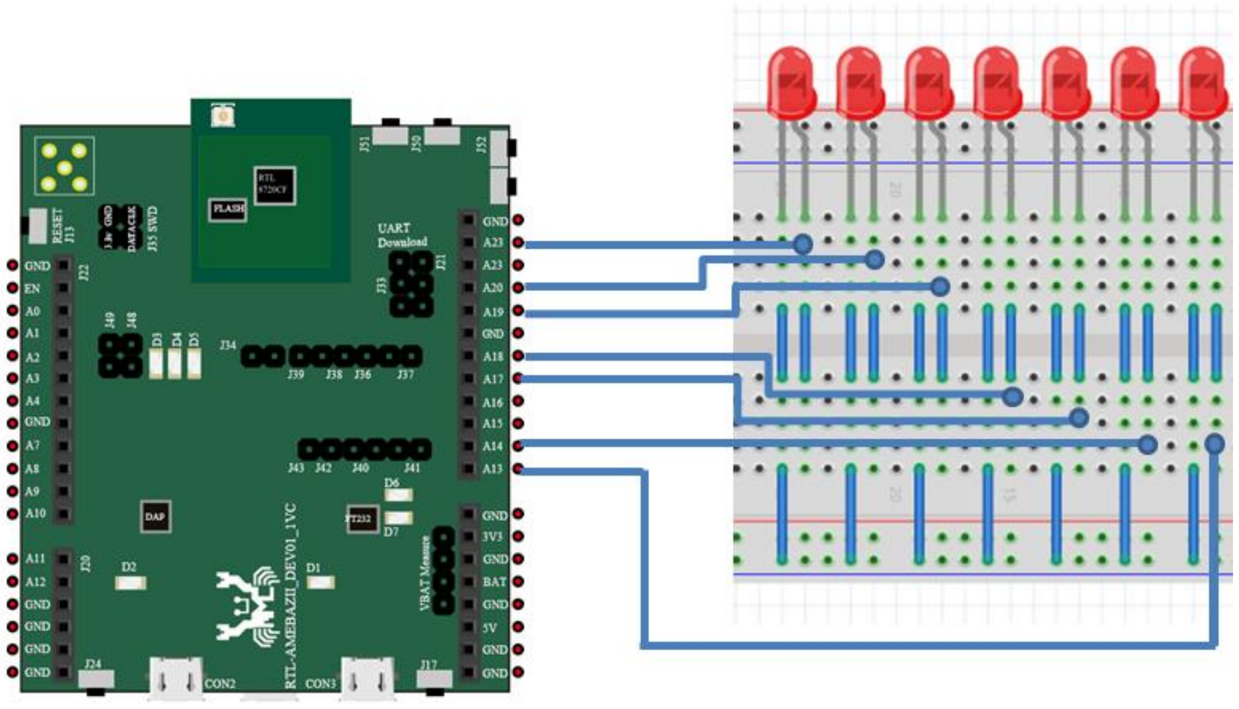
Requirement Components:

1. 7 LEDs
2. 2 boards

Connections:

Output board	<---------->	Input board
PA_23	<---------->	PA_23
PA_20	<---------->	PA_20
PA_19	<---------->	PA_19
PA_18	<---------->	PA_18
PA_17	<---------->	PA_17
PA_14	<---------->	PA_14
PA_13	<---------->	PA_13

Please connect input and output boards together. In addition, each pins of output board connects a LED to ground in series. Define "PORT_OUTPUT_TEST" value to set the board to be input or output. 1: output test, 0: input test. The default port is PortA. "pin_mask" value enables the LED pins. Each bit map to 1 pin: 0: pin disable, 1: pin enable. For example, "pin_mask = 0x9E6000;", 0x9E6000 is binary 1001,1110,0110,0000,0000,0000, represents PA_23, PA_20, PA_19, PA_18, PA_17, PA_14, PA_13. The LEDs will show the "led_pattern" and the log of input board will show every "pin_mask" of each "led_pattern".


GND
EN
A0
A1
A2
A3
A4
GND
A7
A8
A9
A10
A11
A12
GND
GND
GND
GND
A23
A23
A20
A19
GND
A18
A17
A16
A15
A14
A13
GND
3V3
GND
BAT
GND
5V
GND
GND
UART Download
RTL-AMEBAZII_DEV01_1VC
VBAT Measure
DAP
FT232
FLASH
CON2
CON3

2.4.8 gpio_pulse_measure

IMAGE NAME:

flash_is_gpio_pulse_measure.bin

This example describes how to use GPIO read/write mbed api to generate a pulse and to measure the pulse width.

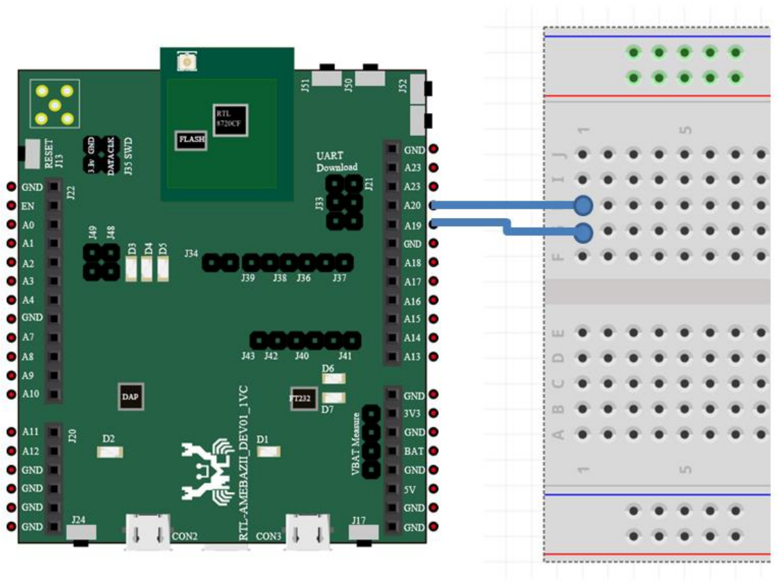
Requirement Components:

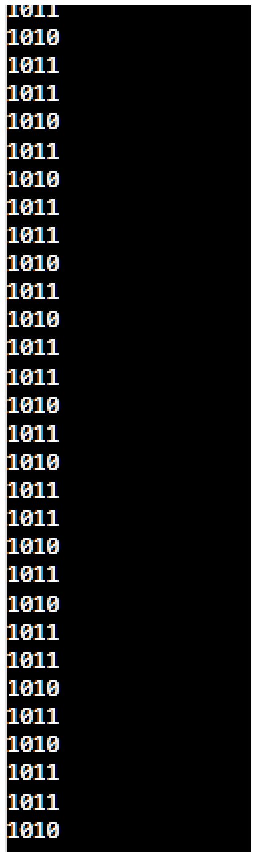
1. a wire

Pin name PA_19 and PA_20 map to GPIOA_19 and GPIOA_20:

- PA_19 as the interrupt GPIO pin with no pull (High-Z).
- PA_20 as output to generate a pulse.
- Use a wire to connect PA_19 and PA_20

In this example, the UART console will print out the measured width (in us) of the pulse which generated by the interrupt GPIO pin.





```
1011
1010
1011
1011
1010
1011
1010
1011
1011
1010
1011
1010
1011
1011
1010
1011
1010
1011
1011
1010
1011
1010
1011
1011
1010
1011
1010
1011
1011
1010
```

2.5 GTIMER

2.5.1 Gtimer

IMAGE NAME:

flash_is_gtimer.bin

This example describes how to use general timer.

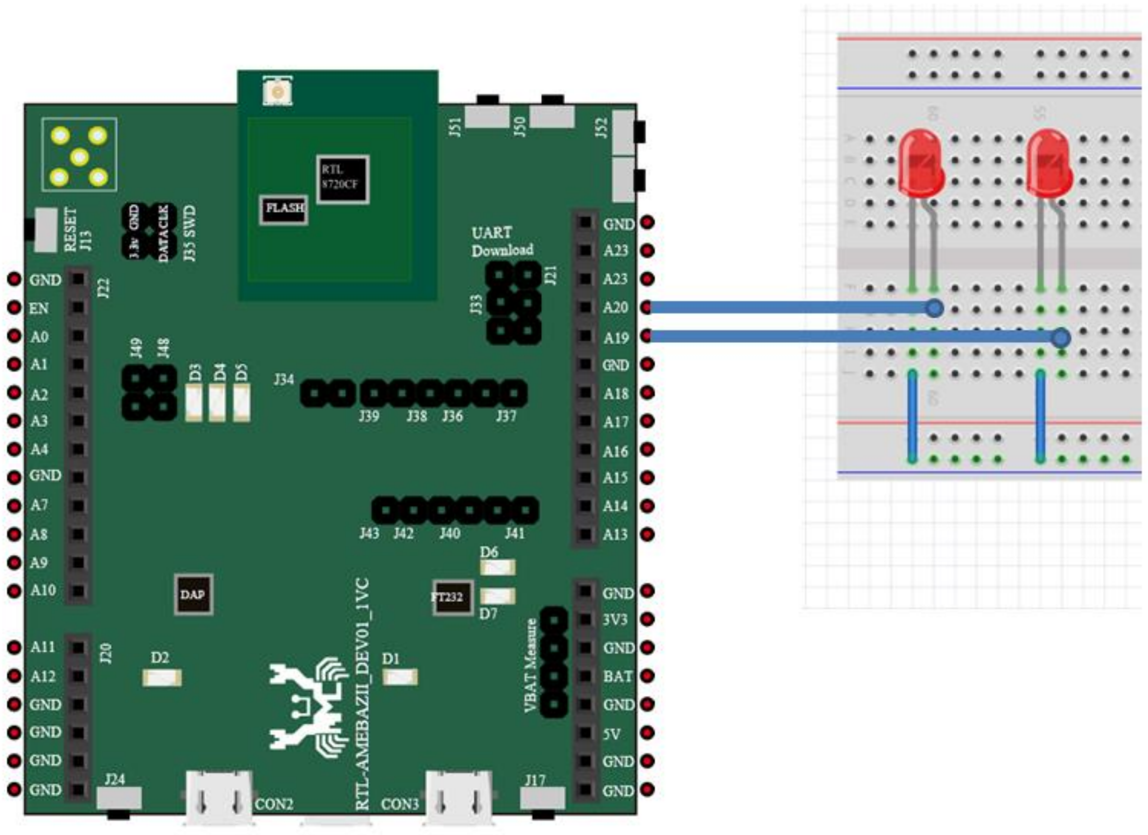
Requirement Components:

1. 2 LED

Connect the two LED to port PA_19 and PA_20 respectively.

The two LED will blink at different frequency. The two timers are initialized in this example.

- Periodic timer
- One shut timer



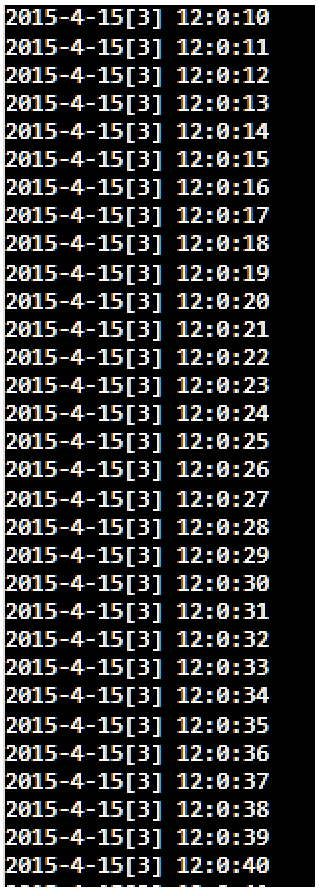
2.5.2 Gtimer_RTC

IMAGE NAME:

flash_is_gtimer_rtc.bin

This example describes how to use general timer API to implement a software RTC.

This example will print the time message to the log UART every 1 sec.



```
2015-4-15[3] 12:0:10
2015-4-15[3] 12:0:11
2015-4-15[3] 12:0:12
2015-4-15[3] 12:0:13
2015-4-15[3] 12:0:14
2015-4-15[3] 12:0:15
2015-4-15[3] 12:0:16
2015-4-15[3] 12:0:17
2015-4-15[3] 12:0:18
2015-4-15[3] 12:0:19
2015-4-15[3] 12:0:20
2015-4-15[3] 12:0:21
2015-4-15[3] 12:0:22
2015-4-15[3] 12:0:23
2015-4-15[3] 12:0:24
2015-4-15[3] 12:0:25
2015-4-15[3] 12:0:26
2015-4-15[3] 12:0:27
2015-4-15[3] 12:0:28
2015-4-15[3] 12:0:29
2015-4-15[3] 12:0:30
2015-4-15[3] 12:0:31
2015-4-15[3] 12:0:32
2015-4-15[3] 12:0:33
2015-4-15[3] 12:0:34
2015-4-15[3] 12:0:35
2015-4-15[3] 12:0:36
2015-4-15[3] 12:0:37
2015-4-15[3] 12:0:38
2015-4-15[3] 12:0:39
2015-4-15[3] 12:0:40
```

2.6 I2C

Please note that for all I2C related examples,

- The supported I2C_SCL & I2C_SDA pins, PA_2 & PA_3, PA_11 & PA_12, PA_15 & PA_16, and PA_19 & PA_20.
- If defined PA_15 & PA_16 as I2C_SCL & I2C_SDA, please off log UART and configure PA_13 and PA_14 to be new log UART. Please use FTDI USB cable to connect the new log UART to the terminal, in order to see the log.
- If defined PA_2 & PA_3 as I2C_SCL & I2C_SDA, please off JTAG/SWD.
- PA_11 & PA_12 as I2C_SCL & I2C_SDA only available on RTL8720CF.

2.6.1 i2c_dual (master & slave)

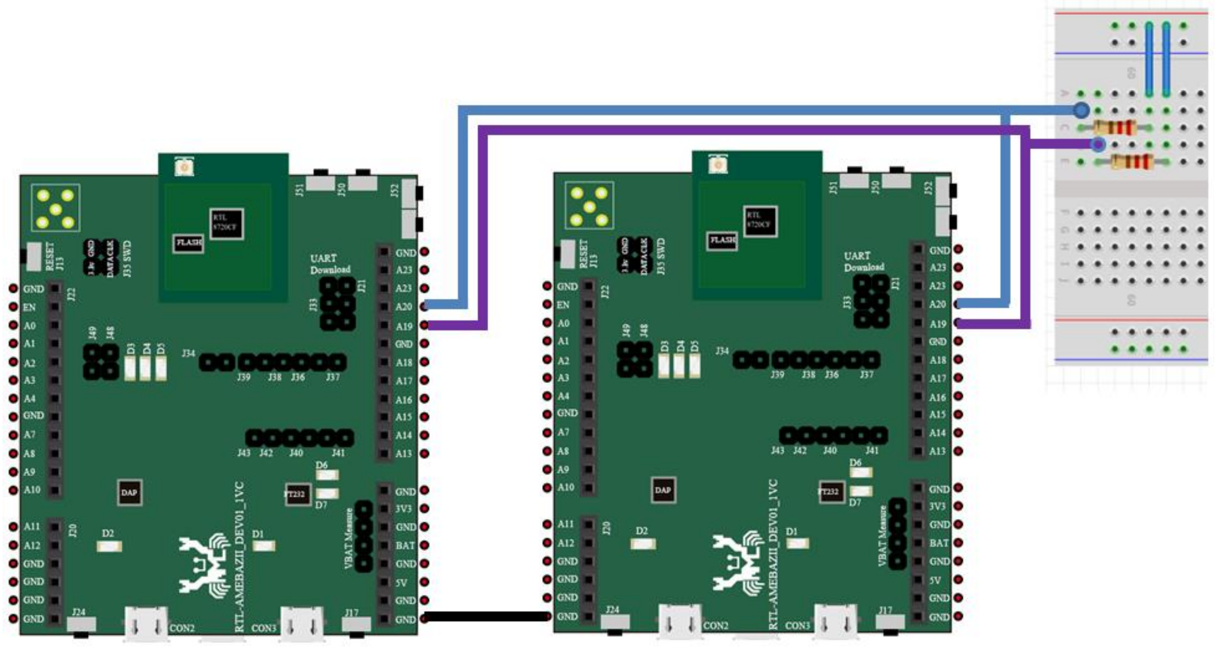
IMAGE NAME:

flash_is_i2c_dual_master.bin;

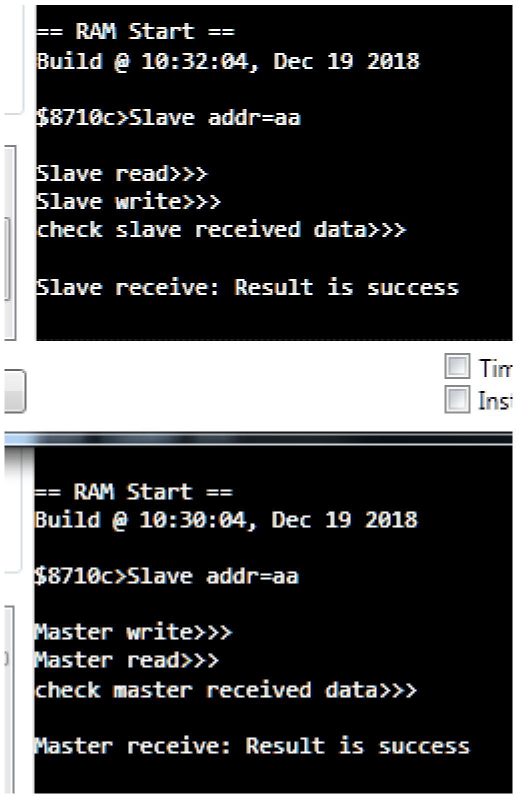
flash_is_i2c_dual_slave.bin

This example describes how to use i2c by using mbed api

1. This test needs two demo board, one as master and the other as slave
2. The codes of Master and Slave are distinguished by the define of macro 'I2C_MASTER_DEVICE' in the code
3. Connect LOG-UART connector to PC
4. Connections
    - Master board I2C0 SDA (PA_20) to Slave board I2C0 SDA (PA_20) pin, and connect an external pull high register;
    - Master board I2C0 SCL (PA_19) to Slave board I2C0 SCL (PA_19) pin, and connect an external pull high register;
    - Master board and Slave board should have same GND.
5. First run Slave and then Master.
6. Get the Master and Slave Data.



Just check the result - the red-circled text. If the result is success, that is mean passing this test.



```
== RAM Start ==
Build @ 10:32:04, Dec 19 2018

$8710c>Slave addr=aa

Slave read>>>
Slave write>>>
check slave received data>>>

Slave receive: Result is success
```

```
== RAM Start ==
Build @ 10:30:04, Dec 19 2018

$8710c>Slave addr=aa

Master write>>>
Master read>>>
check master received data>>>

Master receive: Result is success
```

2.6.2 i2c_epl2197_heartrate

IMAGE NAME:

flash_is_i2c_epl2197_heartrate.bin

This example is used to measure heart rate of human.

Requirement Components:

1. extend board

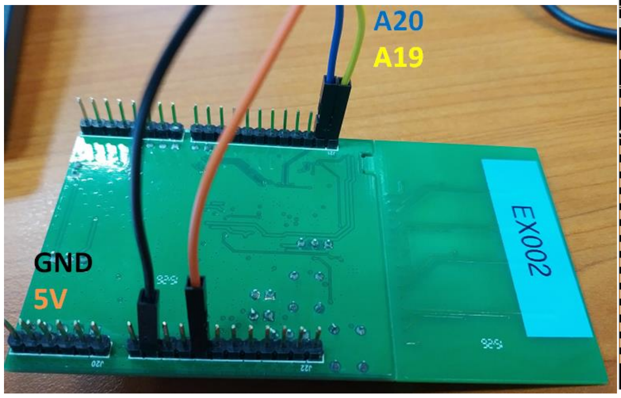
Work with arduino extended board, which has heart rate sensor during the measurement, user has to lie his pulp on the sensor and do not rock the sensor the test code will return back the heart rate.

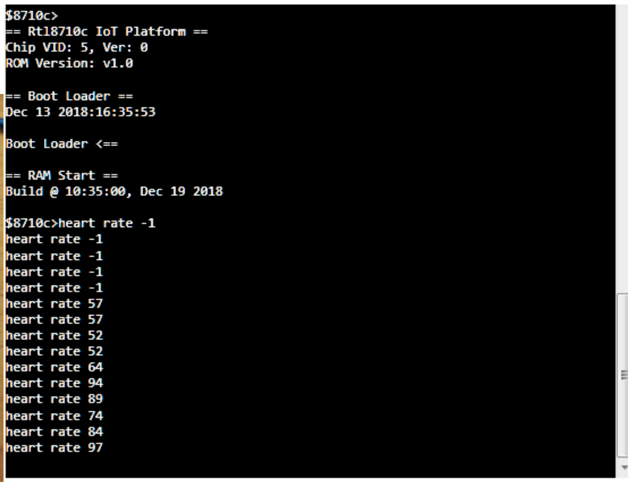
Connections

- I2C0 SDA (PA_20) to extended board's SDA
- I2C0 SCL (PA_19) to extended board's SCL

Build code

- Please be sure to copy inc\heart_interface.h, inc\HRM_2197.h
- Include hr_library.a in project. Add hr_library.a into folder "lib" in project.





```
$8710c>
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 10:35:00, Dec 19 2018

$8710c>heart rate -1
heart rate -1
heart rate -1
heart rate -1
heart rate -1
heart rate 57
heart rate 57
heart rate 52
heart rate 52
heart rate 64
heart rate 94
heart rate 89
heart rate 74
heart rate 84
heart rate 97
```

2.6.3 i2c_epl2590_light

IMAGE NAME:

flash_is_i2c_epl2590_light.bin

This example describes how to use proximity sensor to detect lightness.

Requirement Components:

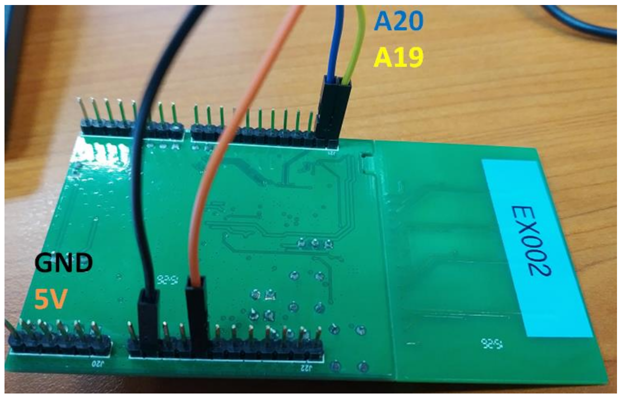
1. extend board

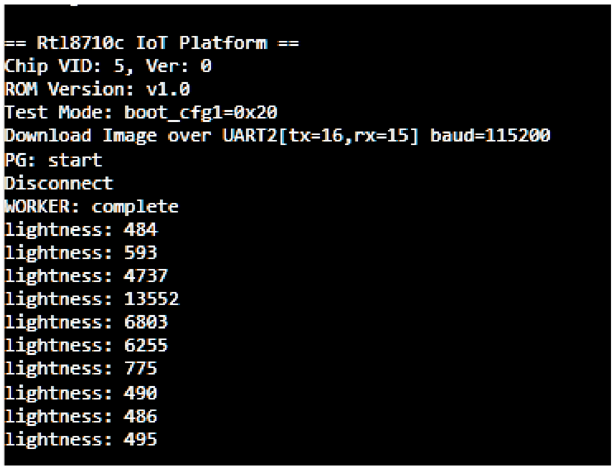
Work with arduino extended board, which has proximity sensor.

When the proximity sensor is in ALS mode (detect lightness), it will keep polling lightness output.

Connect

- I2C0 SDA (PA_20) to extended board's SDA
- I2C0 SCL (PA_19) to extended board's SCL





```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x20
Download Image over UART2[tx=16,rx=15] baud=115200
PG: start
Disconnect
WORKER: complete
lightness: 484
lightness: 593
lightness: 4737
lightness: 13552
lightness: 6803
lightness: 6255
lightness: 775
lightness: 490
lightness: 486
lightness: 495
```

2.6.4 i2c_epl2590_proximity

IMAGE NAME:

flash_is_i2c_epl2590_proximity.bin

This example describes how to use proximity sensor to detect distance

Requirement Components:

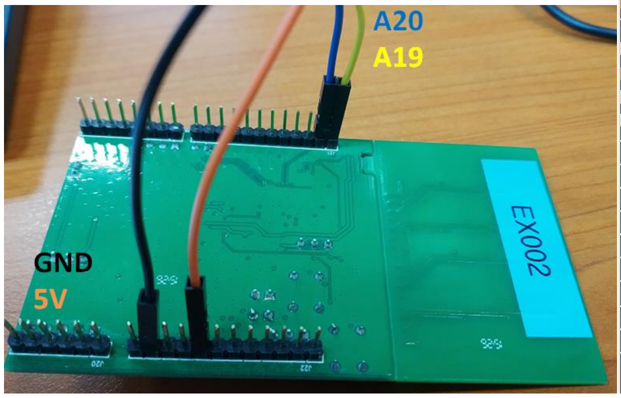
1. extend board

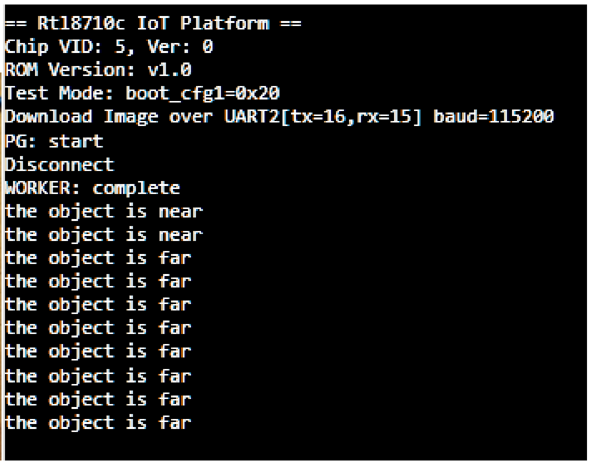
Work with arduino extended board, which has proximity sensor

When the proximity sensor is in PS mode (detect distance), if the object is close to the sensor, a near message will print out. Otherwise, a far message will print out.

Connect

- I2C0 SDA (PA_20) to extended board's SDA
- I2C0 SCL (PA_19) to extended board's SCL





```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x20
Download Image over UART2[tx=16,rx=15] baud=115200
PG: start
Disconnect
WORKER: complete
the object is near
the object is near
the object is far
the object is far
the object is far
the object is far
the object is far
the object is far
the object is far
the object is far
```

2.6.5 i2c_LPS25HB_pressure

IMAGE NAME:

flash_is_i2c_LPS25HB_pressure.bin

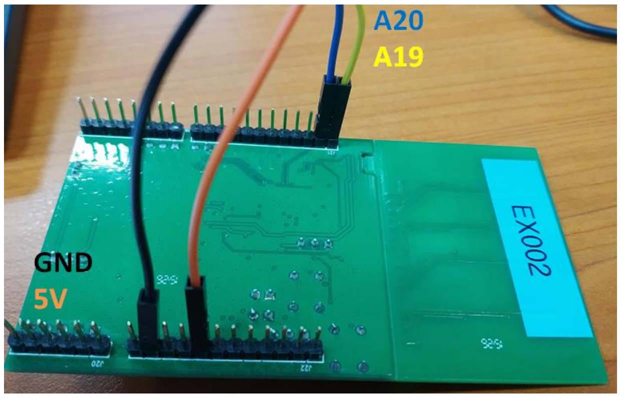
This example is used to measure atmos

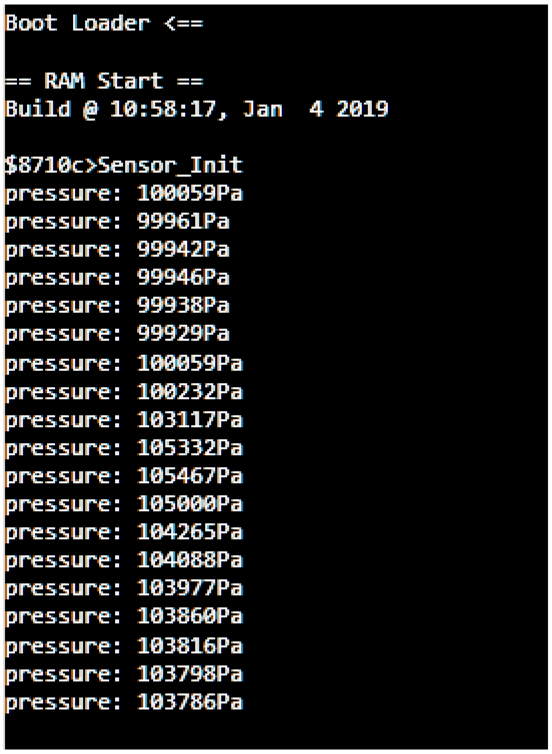
Work with arduino extended board, which has pressure sensor

The terminal will feedback real pressure value, which is represented in Pa

Connect

- I2C0 SDA (PA_20) to extended board's SDA
- I2C0 SCL (PA_19) to extended board's SCL, PA_19 may need a pull high resister if there is no data flow.





```
Boot Loader <==

== RAM Start ==
Build @ 10:58:17, Jan  4 2019

$8710c>Sensor_Init
pressure: 100059Pa
pressure: 99961Pa
pressure: 99942Pa
pressure: 99946Pa
pressure: 99938Pa
pressure: 99929Pa
pressure: 100059Pa
pressure: 100232Pa
pressure: 103117Pa
pressure: 105332Pa
pressure: 105467Pa
pressure: 105000Pa
pressure: 104265Pa
pressure: 104088Pa
pressure: 103977Pa
pressure: 103860Pa
pressure: 103816Pa
pressure: 103798Pa
pressure: 103786Pa
```

2.6.6 i2c-shtc1

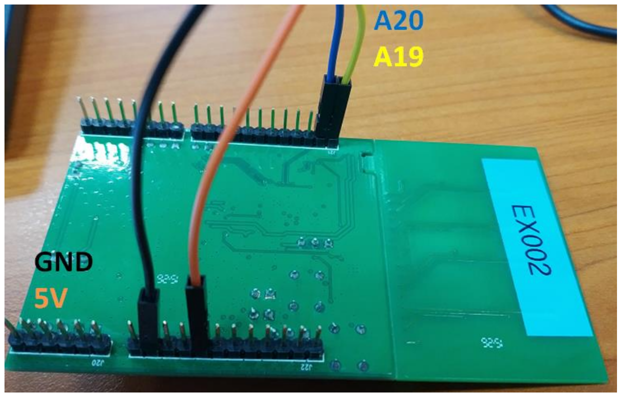
IMAGE NAME:

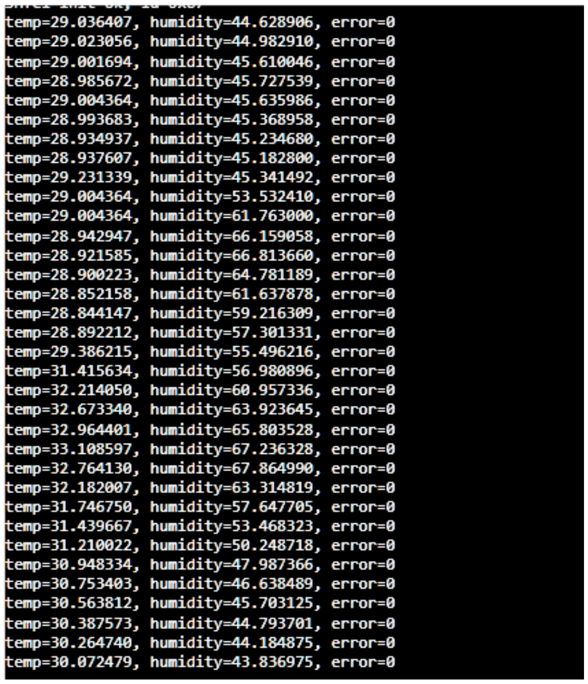
flash_is_i2c-shtc1.bin

This example describes how to use i2c by using mbed api work with arduino extended board, which has SHTC1 temperature and humidity sensor.

Connect

- I2C0 SDA (PA_20) to extended board's SDA
- I2C0 SCL (PA_19) to extended board's SCL





```
temp=29.036407, humidity=44.628906, error=0
temp=29.023056, humidity=44.982910, error=0
temp=29.001694, humidity=45.610046, error=0
temp=28.985672, humidity=45.727539, error=0
temp=29.004364, humidity=45.635986, error=0
temp=28.993683, humidity=45.368958, error=0
temp=28.934937, humidity=45.234680, error=0
temp=28.937607, humidity=45.182800, error=0
temp=29.231339, humidity=45.341492, error=0
temp=29.004364, humidity=53.532410, error=0
temp=29.004364, humidity=61.763000, error=0
temp=28.942947, humidity=66.159058, error=0
temp=28.921585, humidity=66.813660, error=0
temp=28.900223, humidity=64.781189, error=0
temp=28.852158, humidity=61.637878, error=0
temp=28.844147, humidity=59.216309, error=0
temp=28.892212, humidity=57.301331, error=0
temp=29.386215, humidity=55.496216, error=0
temp=31.415634, humidity=56.980896, error=0
temp=32.214050, humidity=60.957336, error=0
temp=32.673340, humidity=63.923645, error=0
temp=32.964401, humidity=65.803528, error=0
temp=33.108597, humidity=67.236328, error=0
temp=32.764130, humidity=67.864990, error=0
temp=32.182007, humidity=63.314819, error=0
temp=31.746750, humidity=57.647705, error=0
temp=31.439667, humidity=53.468323, error=0
temp=31.210022, humidity=50.248718, error=0
temp=30.948334, humidity=47.987366, error=0
temp=30.753403, humidity=46.638489, error=0
temp=30.563812, humidity=45.703125, error=0
temp=30.387573, humidity=44.793701, error=0
temp=30.264740, humidity=44.184875, error=0
temp=30.072479, humidity=43.836975, error=0
```

2.7 POWER MODE

Please note that for all POWER MODE related examples,

- The supported GPIO pins, PA_0, PA_1, PA_2, PA_3, PA_4, PA_7, PA_8, PA_9, PA_10, PA_11, PA_12, PA_13, PA_14, PA_15, PA_16, PA_17, PA_18, PA_19, PA_20, and PA_23.
- Please off JTAG/SWD when using PA_0, PA_1, PA_2, PA_3, and PA_4.
- Please off log UART, when using PA_15, and PA_16.
- PA_7, PA_8, PA_9, PA_10, PA_11 and PA_12 only available on RTL8720CF.
- When using PA_23 as the wake up source, please set the interrupt as "IRQ_RISE". PA_23 connect to GND. PA_17 also connect to a push button then series to a resister and 3.3V.

2.7.1 pm_deepsleep

IMAGE NAME:

flash_is_pm_deepsleep_Stimer_Clock-250k.bin;

flash_is_pm_deepsleep_Stimer_Clock-4M.bin;

flash_is_pm_deepsleep_GPIO_Clock-250k.bin;

flash_is_pm_deepsleep_GPIO_Clock-4M.bin

This example describes how to use Power Mode API for DeepSleep.

Requirement Components:

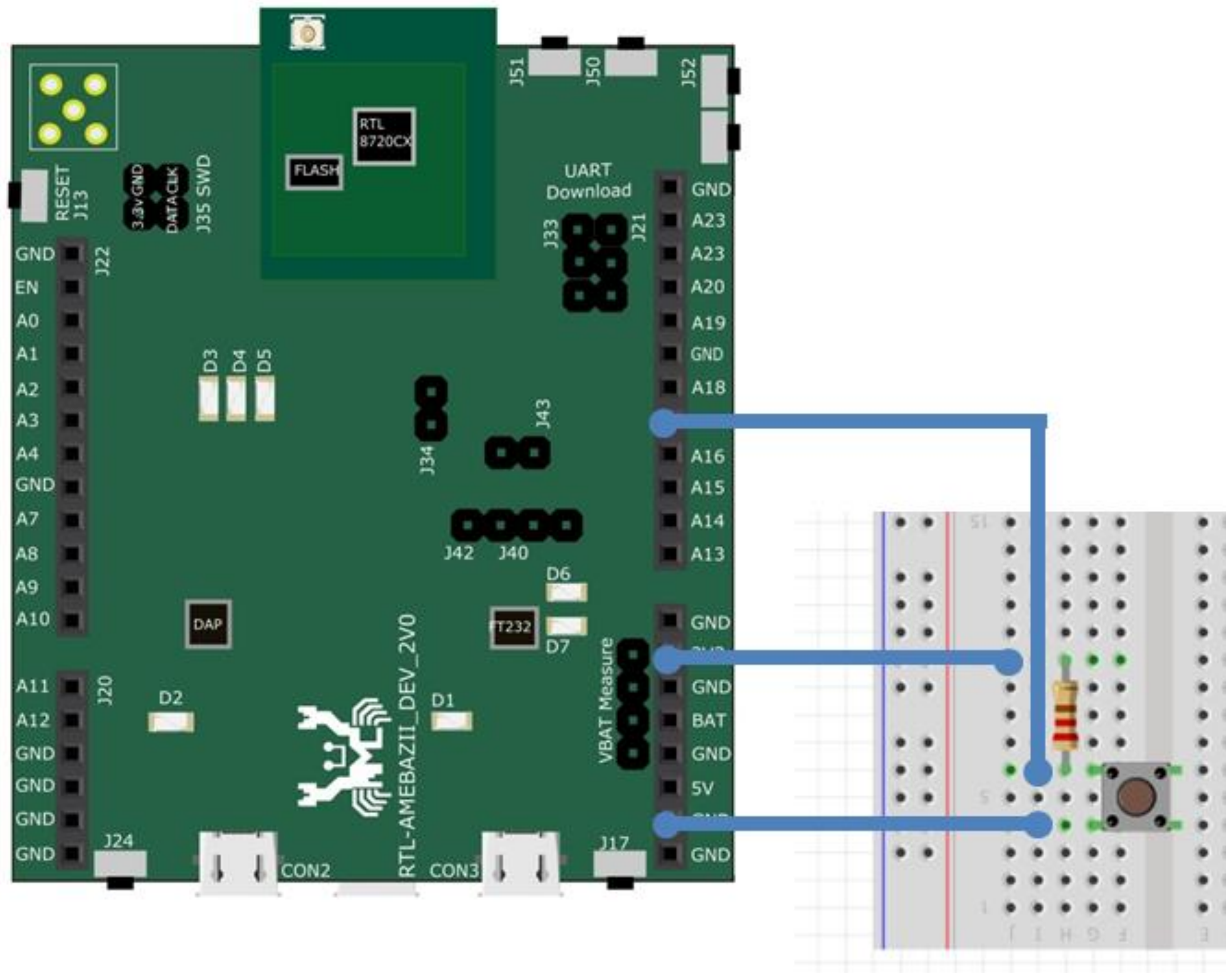
- wake up by Stimer
  1. NONE
- wake up by GPIO
  1. a push button
  2. a resister

In this example,

- When wake up by Sitmer,
  1. The system will enter DeepSleep mode by 5s and then reboot the system.
- When wake up by GPIO,
  i) if there is configured GPIO interrupt.
     1. The system will enter DeepSleep mode by 5s.
     2. if the GPIO interrupt has been activated the system will be rebooted.

ii) if there is no configured GPIO interrupt.

1. PA_17 connect to a resistor series to 3.3V. PA_17 also connect to a push button then series to GND.
2. The system will enter DeepSleep mode by 5s.
3. Please set PA_17 as GPIO interrupt for waking up the system. When the push button is pressed, the system will be rebooted.



2.7.2 pm_sleepcg

IMAGE NAME:

flash_is_pm_sleepcg_Stimer_Clock-250k.bin;

flash_is_pm_sleepcg_Stimer_Clock-4M.bin;

flash_is_pm_sleepcg_GPIO_Clock-250k.bin;

flash_is_pm_sleepcg_GPIO_Clock-4M.bin;

flash_is_pm_ sleepcg _UART_Clock-250k.bin;

flash_is_pm_ sleepcg _UART_Clock-4M.bin;

flash_is_pm_ sleepcg _Gtimer_Clock-250k.bin;

flash_is_pm_ sleepcg _Gtimer_Clock-4M.bin;

flash_is_pm_ sleepcg _PWM_Clock-250k.bin;

flash_is_pm_ sleepcg _PWM_Clock-4M.bin

This example describes how to use Power Mode API for SleepCG.

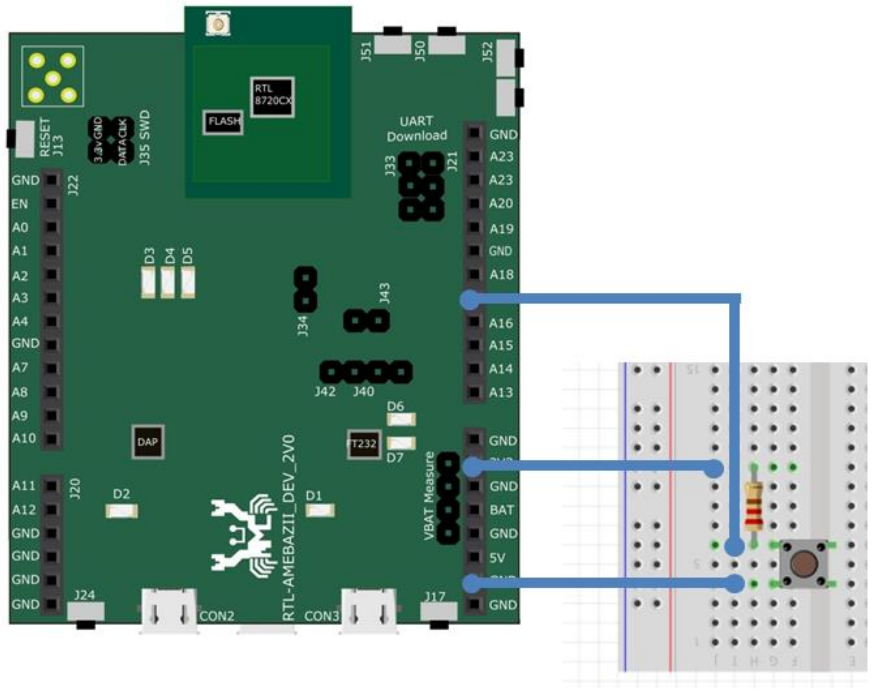
Requirement Components:

- wake up by Stimer
  1. NONE
- wake up by GPIO
  1. a push button
  2. a resister
- wake up by UART
  1. a USB FTDI cable
- wake up by Gtimer
  1. NONE
- wake up by PWM
  1. NONE

In this example,

- When wake up by Sitmer,
  1. The system will enter SleepCG mode by 5s and then reboot the system.

- When wake up by GPIO,
  - i) if there is configured GPIO interrupt.
    1. The system will enter SleepCG mode by 5s.
    2. if the GPIO interrupt has been activated the system will be rebooted.
  - ii) if there is no configured GPIO interrupt.
    1. PA_17 connect to a resistor series to 3.3V. PA_17 also connect to a push button then series to GND.
    2. The system will enter SleepCG mode by 5s.
    3. Please set PA_17 as GPIO interrupt for waking up the system. When the push button is pressed, the system will be rebooted.



- When wake up by UART,
  1. Connecting PA_14, PA_13 and GND(UART_0) to USB FTDI cable. Then connect to a terminal.
  2. The system will enter SleepCG mode by 5s.
  3. Please enter inputs at the UART_0 terminal for waking up the system.
- When wake up by Gtimer,
  1. The system will enter SleepCG mode by 5s and then resume the system.
- When wake up by PWM,
  1. The system will enter SleepCG mode by 5s and then resume the system.

2.7.3 pm_standby

IMAGE NAME:

flash_is_pm_ standby_Stimer_Clock-250k.bin;

flash_is_pm_ standby_Stimer_Clock-4M.bin;

flash_is_pm_ standby_GPIO_Clock-250k.bin;

flash_is_pm_ standby_GPIO_Clock-4M.bin;

flash_is_pm_ standby_UART_Clock-250k.bin;

flash_is_pm_ standby_UART_Clock-4M.bin;

flash_is_pm_ standby_Gtimer_Clock-250k.bin;

flash_is_pm_ standby_Gtimer_Clock-4M.bin;

flash_is_pm_ standby_PWM_Clock-250k.bin;

flash_is_pm_ standby_PWM_Clock-4M.bin

This example describes how to use Power Mode API for Standby.

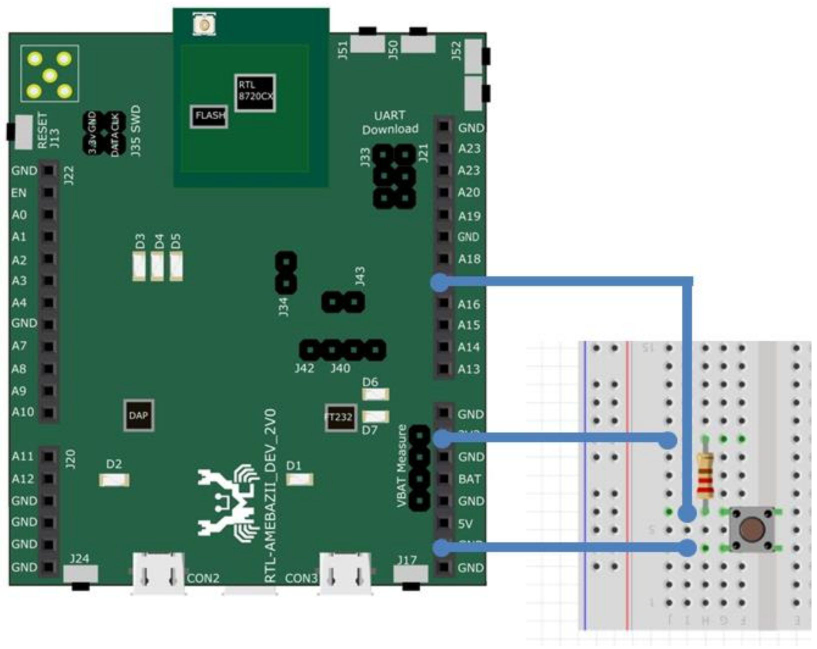
Requirement Components:

- wake up by Stimer
  1. NONE
- wake up by GPIO
  1. a push button
  2. a resister
- wake up by UART
  1. USB FTDI cable
- wake up by Gtimer
  1. NONE
- wake up by PWM
  1. NONE

In this example,

- When wake up by Sitmer,
  1. The system will enter Standby mode by 5s and then reboot the system.

- When wake up by GPIO,
  i) if there is configured GPIO interrupt.
     1. The system will enter Standby mode by 5s.
     2. if the GPIO interrupt has been activated the system will be rebooted.
  ii) if there is no configured GPIO interrupt.
     1. PA_17 connect to a resistor series to 3.3V. PA_17 also connect to a push button then series to GND.
     2. The system will enter Standby mode by 5s.
     3. Please set PA_17 as GPIO interrupt for waking up the system. When the push button is pressed, the system will be rebooted.



- When wake up by UART,
  1. Connecting PA_14, PA_13 and GND(UART_0) to USB FTDI cable. Then connect to a terminal.
  2. The system will enter Standby mode by 5s.
  3. Please enter inputs at the UART_0 terminal for waking up the system.
- When wake up by Gtimer,
  1. The system will enter Standby mode by 5s and then resume the system.
- When wake up by PWM,
  1. The system will enter Standby mode by 5s and then resume the system.

2.8 PWM

Please note that for all PWM related examples,

- The supported PWM, PWM_0 to PWM_7.
    - PWM_0, PA_0, PA_11, PA_20
    - PWM_1, PA_1, PA_12
    - PWM_2, PA_2, PA_14
    - PWM_3, PA_3, PA_15
    - PWM_4, PA_4, PA_16
    - PWM_5, PA_17
    - PWM_6, PA_18
    - PWM_7, PA_13, PA_19, PA_23
- Please off JTAG/SWD, when using PA_0, PA_1, PA_2, PA_3, and PA_4.
- Please off log UART, when using PA_15, and PA_16.
- PA_11 and PA_12 only available on RTL8720CF.

2.8.1 pwm

IMAGE NAME:

flash_is_pwm.bin

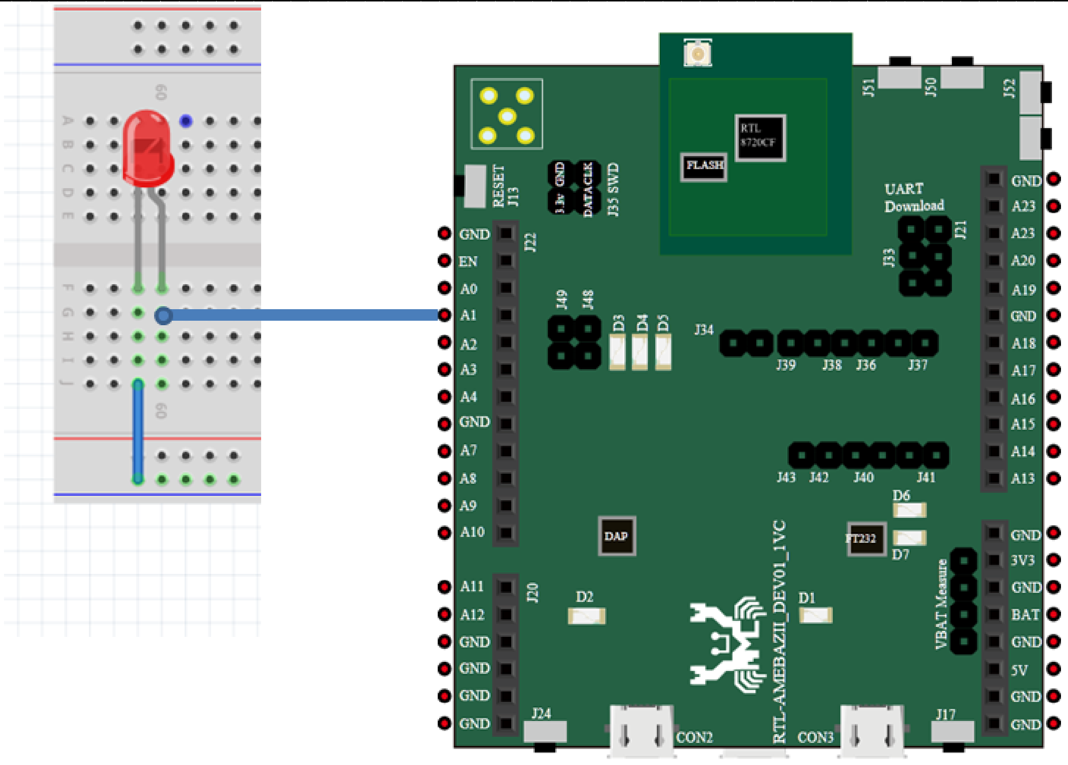
This example describes how to use pwm

Requirement Components:

1. 1~4 LED

Connect LED to below PWM pins and ground, then the LED would gradually become brighter and then darker with different speed.

- connect PA_17 to a LED and ground in series
- connect PA_18 to a LED and ground in series
- connect PA_13 to a LED and ground in series
- connect PA_14 to a LED and ground in series


RESET
J13
GND
3.3v
DATACLK
J35 SWD
RTL
8720CF
FLASH
J51
J50
J52
UART
Download
J21
J33
J22
GND
EN
A0
A1
A2
A3
A4
GND
A7
A8
A9
A10
A11
A12
GND
GND
GND
GND
J49
J48
D3
D4
D5
J34
J39
J38
J36
J37
J43
J42
J40
J41
D6
D7
DAP
FT232
RTL-AMEBAZII_DEV01_1VC
D1
D2
J20
VBAT Measure
J24
CON2
CON3
J17
GND
A23
A23
A20
A19
GND
A18
A17
A16
A15
A14
A13
GND
3V3
GND
BAT
GND
5V
GND
GND

2.8.2 pwm-buzzer

IMAGE NAME:

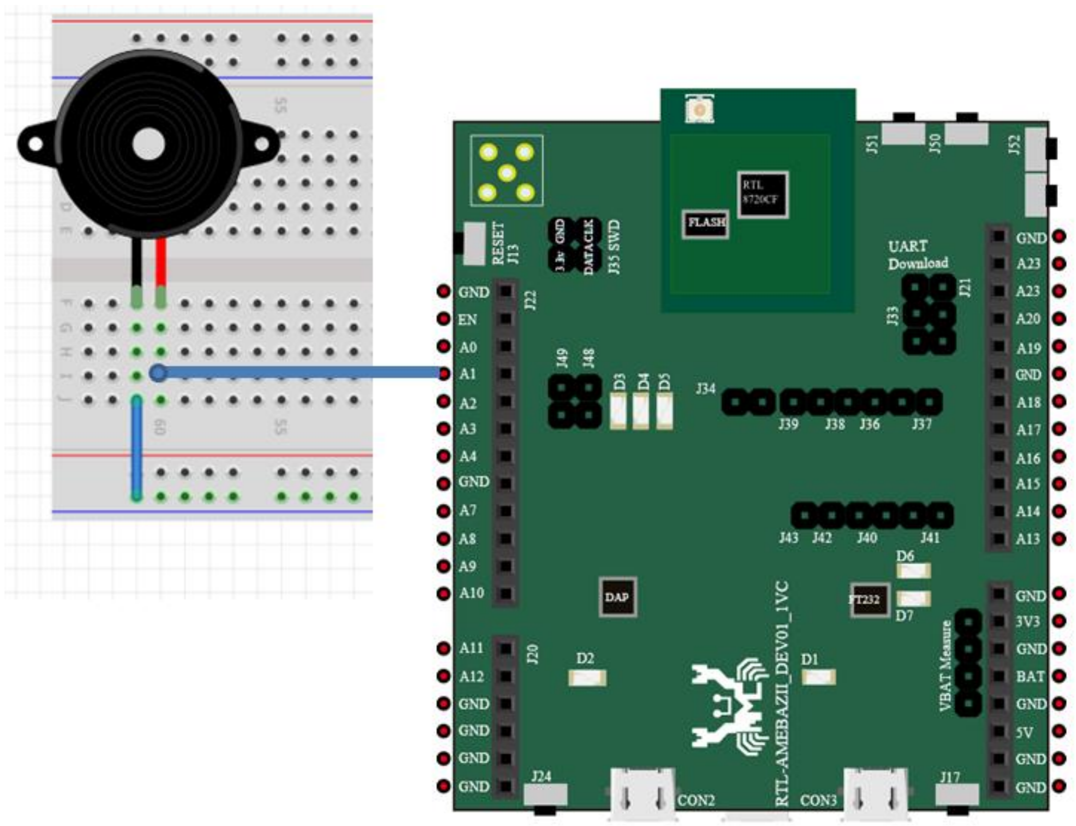
flash_is_pwm-buzzer.bin

This example describes how to use pwm buzzer

Requirement Components:

1. buzzer

Connect buzzer positive pin to PA_17 and buzzer negative pin to ground pin, then the buzzer would play sound from “Do” to higher “Do”.



2.9 SPI

Please note that for all SPI related examples,

- The supported SPI_CS, SPI_SCL, SPI_MOSI and SPI_MISO
    - SPI_CS, PA_2, PA_7, PA_15
    - SPI_SCL, PA_3, PA_8, PA_16
    - SPI_MOSI, PA_4, PA_9, PA_19
    - SPI_MISO, PA_10, PA_20
- Please off JTAG/SWD, when using PA_2, PA_3, and PA_4.
- Please off log UART, when using PA_15, and PA_16. Please off log UART and configure PA_13 and PA_14 to be the new log UART. Please use FTDI USB cable to connect the new log UART to the terminal, in order to see the log.
- PA_7, PA_8, PA_9, and PA_10 only available on RTL8720CF.

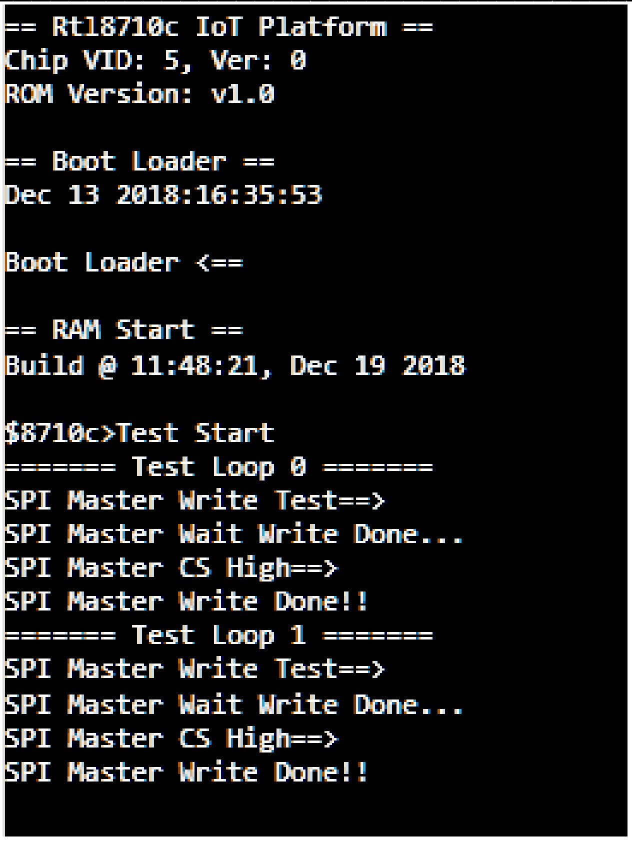
2.9.1 spi_gpio_chipselect

IMAGE NAME:

flash_is_spi_gpio_chipselect.bin

This example describes how to use another GPIO to replace CS line of SPI master.

The SPI Interface provides a "Serial Peripheral Interface" Master. This interface can be used for communication with SPI slave devices, such as FLASH memory, LCD screens and other modules or integrated circuits. This example shows how users can use GPIO to replace SPI's default slave select pin for SPI master. Users are allowed to select another available GPIO pin as a slave select line by defining the value of SPI_GPIO_CS Before any master operation that requires the slave select to pull up, users should call the function gpio_write(&spi_cs, 1); as demonstrated in example code. Slave select line then pulls low by calling the function gpio_write(&spi_cs, 0); in the interrupt function master_cs_tr_done_callback to indicate the operation is done.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 11:48:21, Dec 19 2018

$8710c>Test Start
======= Test Loop 0 =======
SPI Master Write Test==>
SPI Master Wait Write Done...
SPI Master CS High==>
SPI Master Write Done!!
======= Test Loop 1 =======
SPI Master Write Test==>
SPI Master Wait Write Done...
SPI Master CS High==>
SPI Master Write Done!!
```

2.9.2 spi_stream_dma_twoboard

IMAGE NAME:

flash_is_spi_stream_dma_twoboard_master.bin;

flash_is_spi_stream_dma_twoboard_slave.bin

This example describes how to use SPI read/write dma mode by mbed api.

The SPI Interface provides a "Serial Peripheral Interface" Master. This interface can be used for communication with SPI slave devices, such as FLASH memory, LCD screens and other modules or integrated circuits.

In this example, we use config SPI_IS_AS_MASTER to decide if device is master or slave.

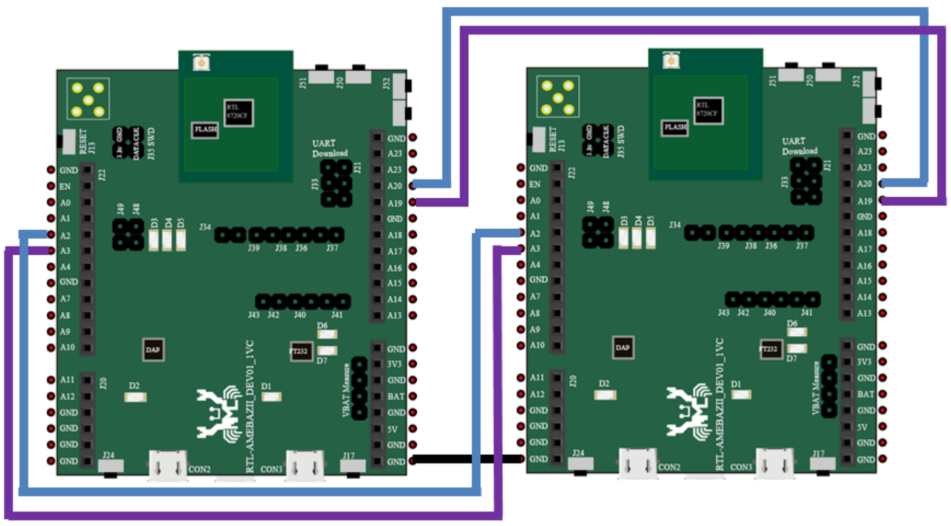
- If SPI_IS_AS_MASTER is 1, then device is master.
- If SPI_IS_AS_MASTER is 0, then device is slave.

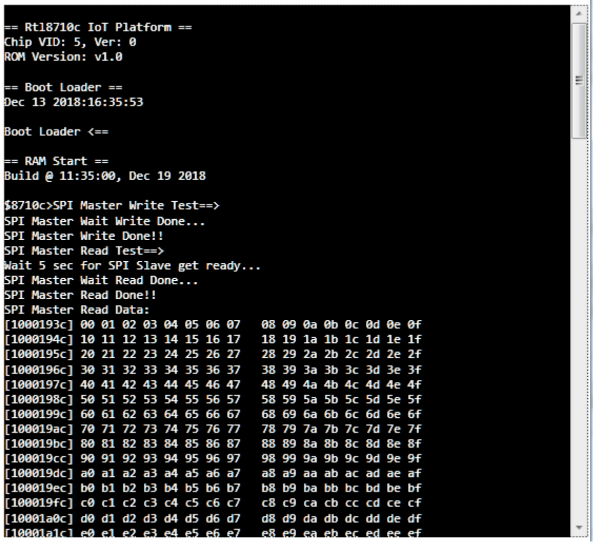
Connections:

Master board	<---------->	Slave board
master's MOSI (PA_19)	<---------->	slave's MOSI (PA_19)
master's MISO (PA_20)	<---------->	slave's MISO (PA_20)
master's SCLK (PA_3)	<---------->	slave's SCLK (PA_3)
master's CS (PA_2)	<---------->	slave's CS (PA_2)

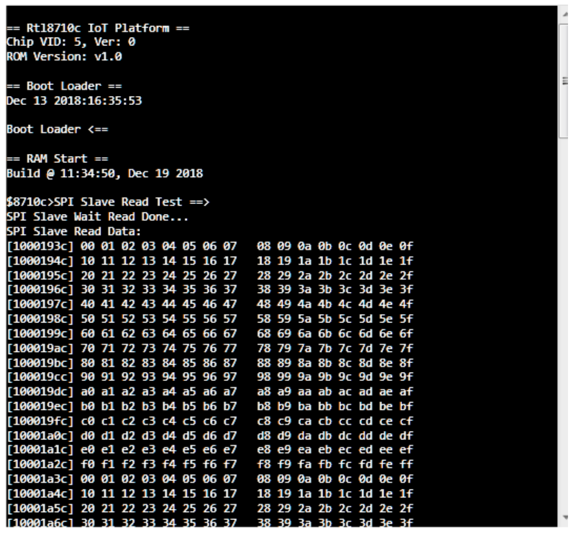
This example shows master sends data to slave in dma mode.

We bootup slave first, and then bootup master. Then log will presents that master sending data to slave.

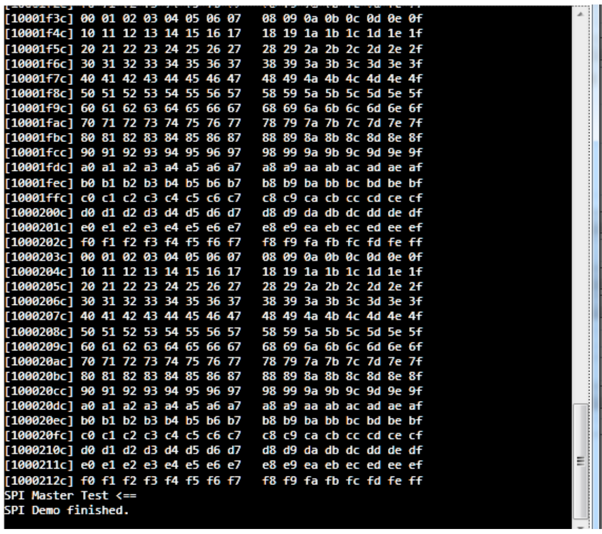




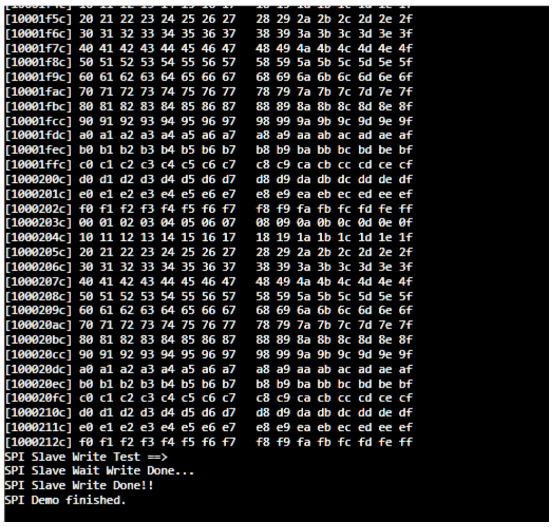
```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 11:35:00, Dec 19 2018

$8710c>SPI Master Write Test==>
SPI Master Wait Write Done...
SPI Master Write Done!!
SPI Master Read Test==>
Wait 5 sec for SPI Slave get ready...
SPI Master Wait Read Done...
SPI Master Read Done!!
SPI Master Read Data:
[1000193c] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[1000194c] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[1000195c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[1000196c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[1000197c] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[1000198c] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[1000199c] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100019ac] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100019bc] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100019cc] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100019dc] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[100019ec] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[100019fc] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10001a0c] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10001a1c] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
```



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 11:34:50, Dec 19 2018

$8710c>SPI Slave Read Test ==>
SPI Slave Wait Read Done...
SPI Slave Read Data:
[1000193c] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[1000194c] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[1000195c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[1000196c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[1000197c] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[1000198c] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[1000199c] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100019ac] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100019bc] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100019cc] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100019dc] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[100019ec] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[100019fc] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10001a0c] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10001a1c] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10001a2c] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10001a3c] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001a4c] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001a5c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001a6c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
```



```
[10001f3c] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001f4c] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001f5c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001f6c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001f7c] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001f8c] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001f9c] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001fac] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001fbc] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001fcc] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001fdc] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001fec] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10001ffc] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[1000200c] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[1000201c] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[1000202c] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[1000203c] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[1000204c] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[1000205c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[1000206c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[1000207c] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[1000208c] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[1000209c] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100020ac] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100020bc] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100020cc] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100020dc] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[100020ec] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[100020fc] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[1000210c] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[1000211c] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[1000212c] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
SPI Master Test <==
SPI Demo finished.
```



```
[10001f5c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001f6c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001f7c] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001f8c] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001f9c] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001fac] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001fbc] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001fcc] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001fdc] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001fec] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10001ffc] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[1000200c] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[1000201c] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[1000202c] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[1000203c] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[1000204c] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[1000205c] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[1000206c] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[1000207c] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[1000208c] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[1000209c] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100020ac] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100020bc] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100020cc] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100020dc] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[100020ec] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[100020fc] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[1000210c] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[1000211c] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[1000212c] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
SPI Slave Write Test ==>
SPI Slave Wait Write Done...
SPI Slave Write Done!!
SPI Demo finished.
```

2.9.3 spi_stream_read_unfix_size

IMAGE NAME:

flash_is_spi_stream_read_unfix_size_master.bin;

flash_is_spi_stream_read_unfix_size_slave.bin

This example describes how to use SPI read dma mode by mbed api.

The SPI Interface provides a "Serial Peripheral Interface" Master.

This interface can be used for communication with SPI slave devices, such as FLASH memory, LCD screens and other modules or integrated circuits.

In this example, we use config SPI_IS_AS_MASTER to decide if device is master or slave.

- If SPI_IS_AS_MASTER is 1, then device is master.
- If SPI_IS_AS_MASTER is 0, then device is slave.

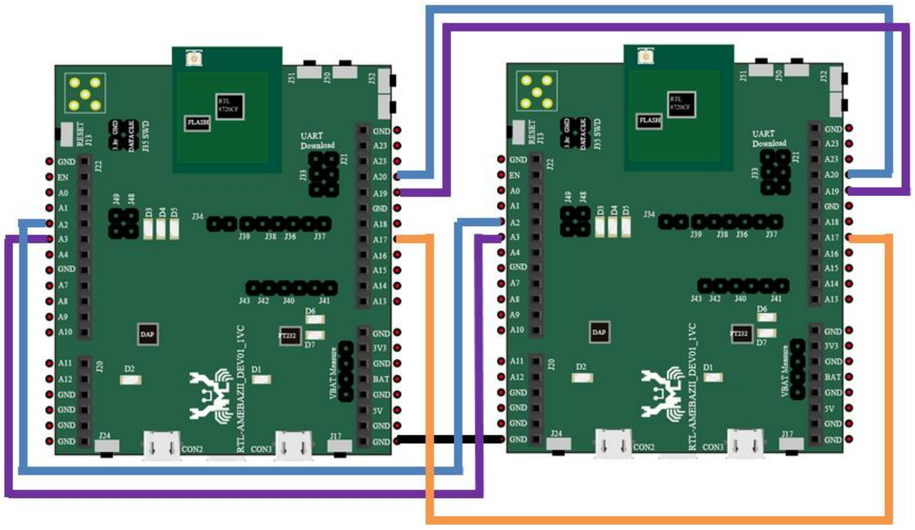
Connections:

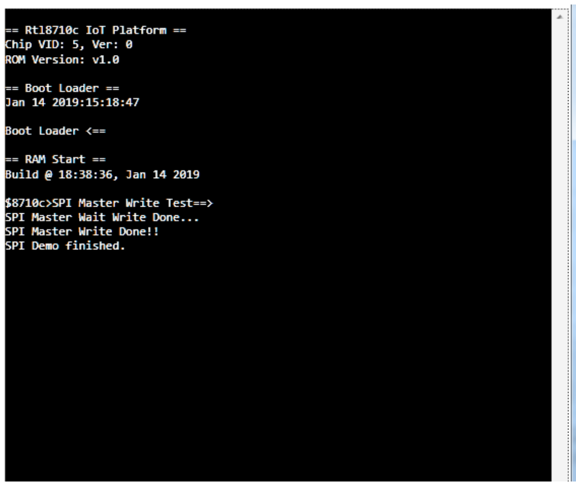
Master board	<---------->	Slave board
master's MOSI (PA_19)	<---------->	slave's MOSI (PA_19)
master's MISO (PA_20)	<---------->	slave's MISO (PA_20)
master's SCLK (PA_3)	<---------->	slave's SCLK (PA_3)
master's CS (PA_2)	<---------->	slave's CS (PA_2)
master's gpio (PA_17)	<---------->	slave's gpio (PA_17)

This example shows master sends data to slave in by using function "spi_slave_read_stream_unfix_size()". "read_stream_unfix_size" means the slave board is able to read data with unfixed size and then call back.

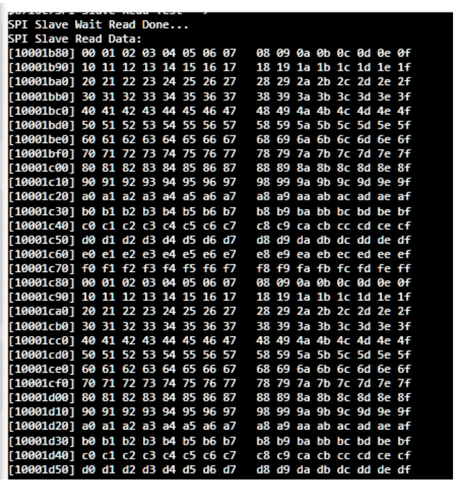
We bootup slave first, and then bootup master.

Then log will presents that master sending data to slave.

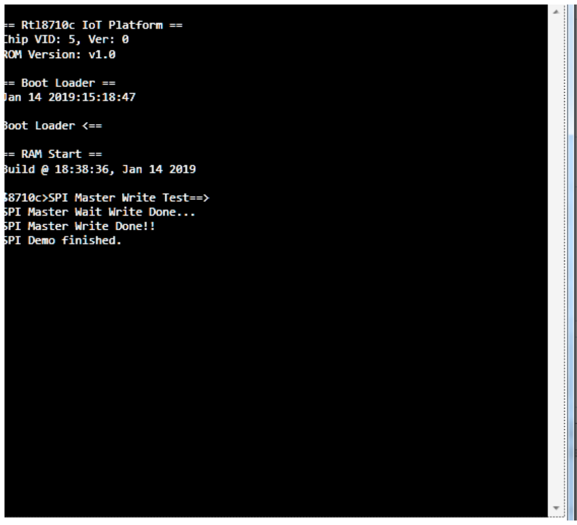




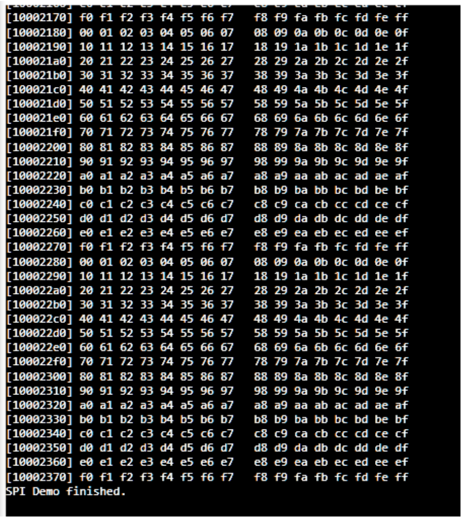
```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Jan 14 2019:15:18:47

Boot Loader <==

== RAM Start ==
Build @ 18:38:36, Jan 14 2019

$8710c>SPI Master Write Test==>
SPI Master Wait Write Done...
SPI Master Write Done!!
SPI Demo finished.
```



```
SPI Slave Wait Read Done...
SPI Slave Read Data:
[10001b80] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001b90] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001ba0] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001bb0] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001bc0] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001bd0] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001be0] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001bf0] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001c00] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001c10] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001c20] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001c30] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10001c40] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10001c50] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10001c60] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10001c70] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10001c80] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001c90] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001ca0] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001cb0] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001cc0] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001cd0] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001ce0] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001cf0] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001d00] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001d10] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001d20] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001d30] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10001d40] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10001d50] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
```



```
= Rtl8710c IoT Platform ==
hip VID: 5, Ver: 0
OM Version: v1.0

= Boot Loader ==
an 14 2019:15:18:47

oot Loader <==

= RAM Start ==
uild @ 18:38:36, Jan 14 2019

8710c>SPI Master Write Test==>
PI Master Wait Write Done...
PI Master Write Done!!
PI Demo finished.
```



```
[10002170] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10002180] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10002190] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[100021a0] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[100021b0] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[100021c0] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[100021d0] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[100021e0] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100021f0] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10002200] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10002210] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10002220] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10002230] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10002240] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10002250] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10002260] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10002270] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10002280] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10002290] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[100022a0] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[100022b0] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[100022c0] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[100022d0] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[100022e0] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100022f0] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10002300] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10002310] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10002320] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10002330] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10002340] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10002350] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10002360] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10002370] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
SPI Demo finished.
```

2.9.4 spi_stream_twoboard

IMAGE NAME:

flash_is_spi_stream_twoboard_master.bin;

flash_is_spi_stream_twoboard_slave.bin

This example describes how to use SPI stream read/write by mbed api.

The SPI Interface provides a "Serial Peripheral Interface" Master.

This interface can be used for communication with SPI slave devices, such as FLASH memory, LCD screens and other modules or integrated circuits.

In this example, we use config SPI_IS_AS_MASTER to decide if device is master or slave.

- If SPI_IS_AS_MASTER is 1, then device is master.
- If SPI_IS_AS_MASTER is 0, then device is slave.

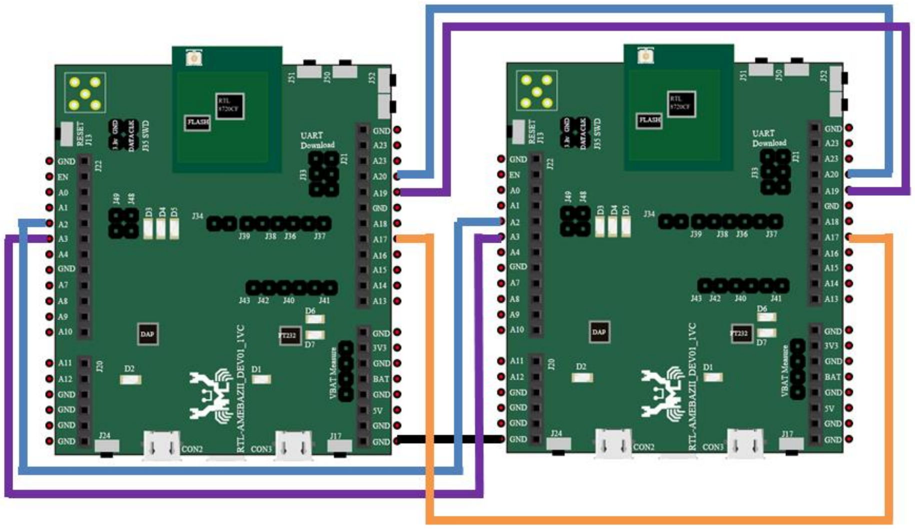
Connections:

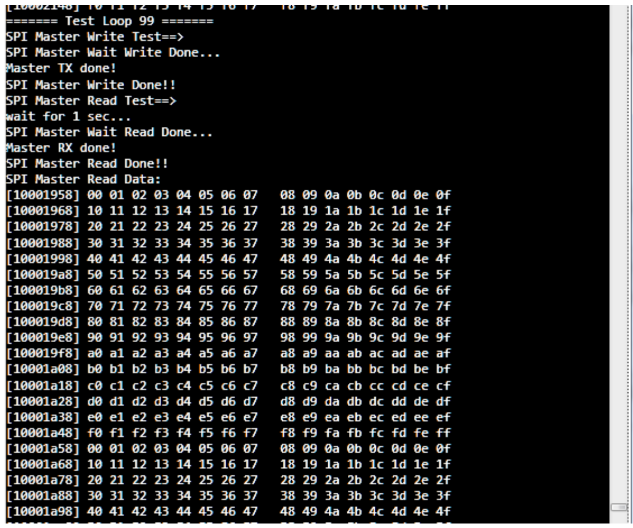
Master board	<---------->	Slave board
master's MOSI (PA_19)	<---------->	slave's MOSI (PA_19)
master's MISO (PA_20)	<---------->	slave's MISO (PA_20)
master's SCLK (PA_3)	<---------->	slave's SCLK (PA_3)
master's CS (PA_2)	<---------->	slave's CS (PA_2)
master's gpio (PA_17)	<---------->	slave's gpio (PA_17)

This example shows master sends data to slave.

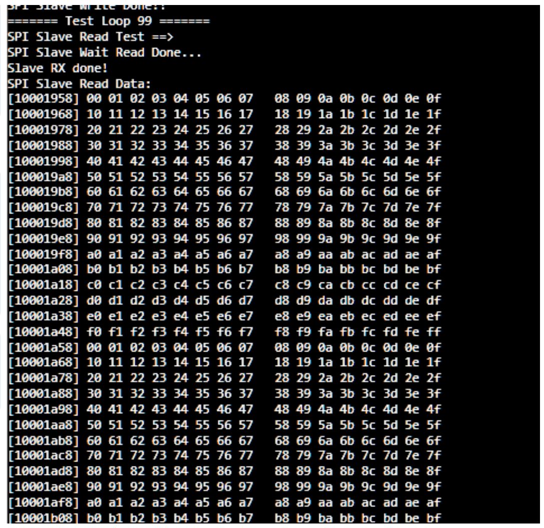
We bootup slave first, and then bootup master.

Then log will presents that master sending data to slave. To ensure the order is correct, we use a GPIO pin to notify the master that the slave device is ready to write or read data.

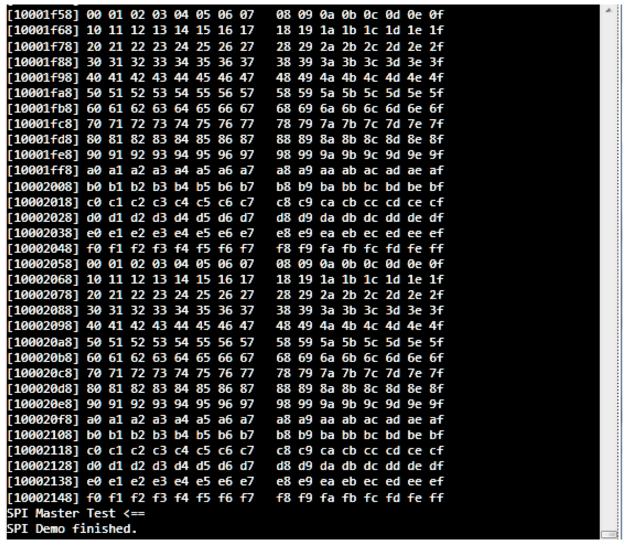




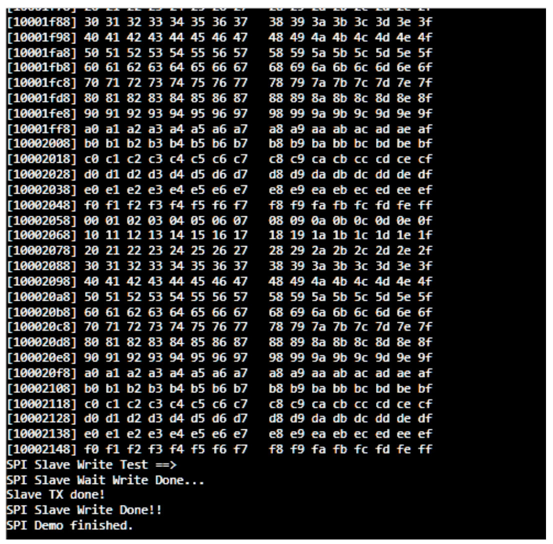
```
======= Test Loop 99 =======
SPI Master Write Test==>
SPI Master Wait Write Done...
Master TX done!
SPI Master Write Done!!
SPI Master Read Test==>
wait for 1 sec...
SPI Master Wait Read Done...
Master RX done!
SPI Master Read Done!!
SPI Master Read Data:
[10001958] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001968] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001978] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001988] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001998] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[100019a8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[100019b8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100019c8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100019d8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100019e8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100019f8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001a08] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10001a18] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10001a28] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10001a38] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10001a48] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10001a58] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001a68] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001a78] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001a88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001a98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
```



```
======= Test Loop 99 =======
SPI Slave Read Test ==>
SPI Slave Wait Read Done...
Slave RX done!
SPI Slave Read Data:
[10001958] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001968] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001978] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001988] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001998] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[100019a8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[100019b8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100019c8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100019d8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100019e8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100019f8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001a08] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10001a18] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10001a28] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10001a38] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10001a48] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10001a58] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001a68] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001a78] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001a88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001a98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001aa8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001ab8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001ac8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001ad8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001ae8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001af8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10001b08] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
```



```
[10001f58] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001f68] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001f78] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001f88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001f98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001fa8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001fb8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001fc8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001fd8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001fe8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001ff8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10002008] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10002018] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10002028] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10002038] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10002048] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10002058] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10002068] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10002078] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10002088] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10002098] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[100020a8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[100020b8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100020c8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100020d8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100020e8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100020f8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10002108] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10002118] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10002128] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10002138] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10002148] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
SPI Master Test <==
SPI Demo finished.
```



```
[10001f88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001f98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001fa8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001fb8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001fc8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001fd8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[10001fe8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[10001ff8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10002008] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10002018] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10002028] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10002038] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10002048] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
[10002058] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10002068] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10002078] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10002088] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10002098] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[100020a8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[100020b8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[100020c8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[100020d8] 80 81 82 83 84 85 86 87   88 89 8a 8b 8c 8d 8e 8f
[100020e8] 90 91 92 93 94 95 96 97   98 99 9a 9b 9c 9d 9e 9f
[100020f8] a0 a1 a2 a3 a4 a5 a6 a7   a8 a9 aa ab ac ad ae af
[10002108] b0 b1 b2 b3 b4 b5 b6 b7   b8 b9 ba bb bc bd be bf
[10002118] c0 c1 c2 c3 c4 c5 c6 c7   c8 c9 ca cb cc cd ce cf
[10002128] d0 d1 d2 d3 d4 d5 d6 d7   d8 d9 da db dc dd de df
[10002138] e0 e1 e2 e3 e4 e5 e6 e7   e8 e9 ea eb ec ed ee ef
[10002148] f0 f1 f2 f3 f4 f5 f6 f7   f8 f9 fa fb fc fd fe ff
SPI Slave Write Test ==>
SPI Slave Wait Write Done...
Slave TX done!
SPI Slave Write Done!!
SPI Demo finished.
```

2.9.5 spi_stream_twoboard_concurrent

IMAGE NAME:

flash_is_spi_stream_twoboard_concurrent_master.bin;

flash_is_spi_stream_twoboard_concurrent_slave.bin

This example describes how to use SPI master to transmit & receive data concurrently by mbed api.

The SPI Interface provides a "Serial Peripheral Interface" Master.

This interface can be used for communication with SPI slave devices, such as FLASH memory, LCD screens and other modules or integrated circuits.

In this example, we use config SPI_IS_AS_MASTER to decide if device is master or slave.

- If SPI_IS_AS_MASTER is 1, then device is master.
- If SPI_IS_AS_MASTER is 0, then device is slave.

The option SPI_DMA_DEMO provides a demonstration in the SPI DMA mode.

- If SPI_DMA_DEMO is 1, then the device operates in DMA mode.
- If SPI_DMA_DEMO is 0, then the device operates in interrupt mode.

Connections:

Master board	<---------->	Slave board
master's MOSI (PA_19)	<---------->	slave's MOSI (PA_19)
master's MISO (PA_20)	<---------->	slave's MISO (PA_20)
master's SCLK (PA_3)	<---------->	slave's SCLK (PA_3)
master's CS (PA_2)	<---------->	slave's CS (PA_2)
master's gpio (PA_17)	<---------->	slave's gpio (PA_17)

SPI_IS_AS_MASTER is 1

The device operates in the master mode.

The SPI master transmits 512 bytes data in increasing order to the slave.

In the meantime, we expect the master to receive data sent by the SPI slave via spi_master_write_read_stream(spi_master_write_read_stream_dma).

SPI_IS_AS_MASTER is 0

The device operates in the slave mode.

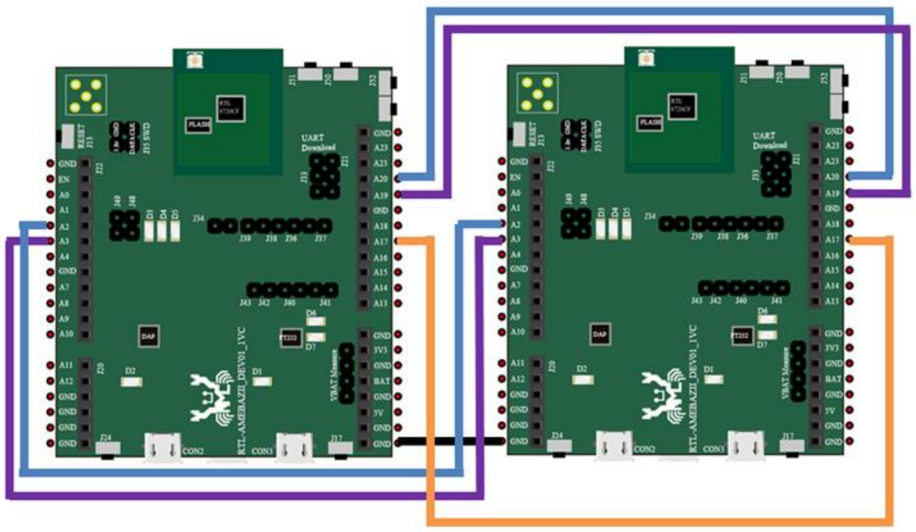
To differentiate data from the master or the slave, we assign data sent from the slave in decreasing order.

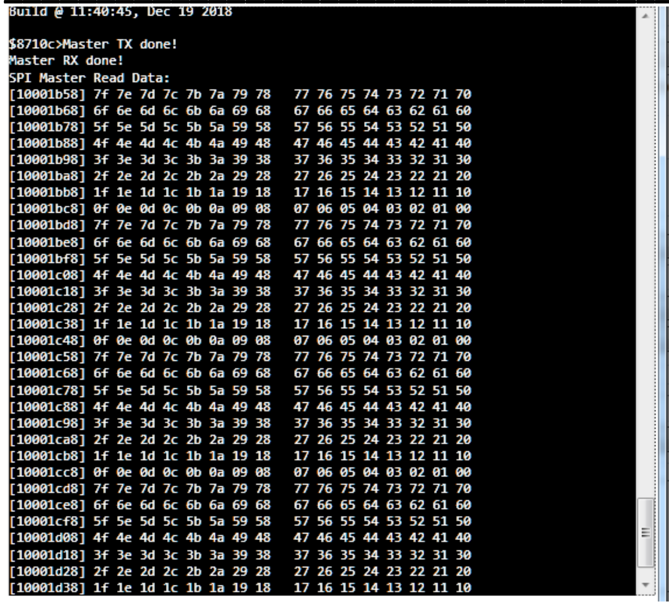
We call the spi_slave_read_stream (or spi_slave_read_stream_dma) first to get the slave device ready to receive, then call the spi_slave_write_stream_dma that the slave would push data into its fifo.

The master device should be enabled first prior to the slave device.

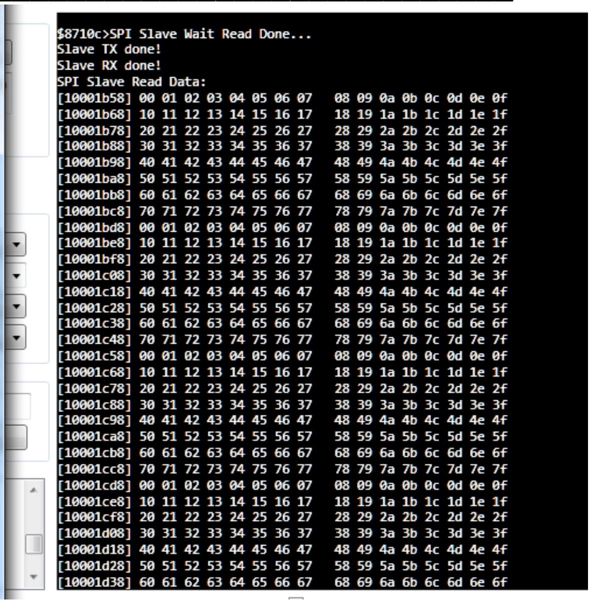
As soon as the master sends the clock to the slave, the slave then transmits the data from its fifo to the master and receive data from the master simultaneously.

To make sure the order is right, we use a GPIO pin to signal the master that the slave device is ready.

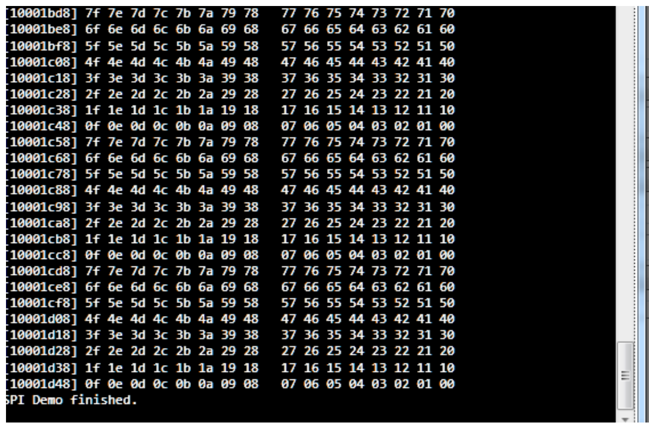




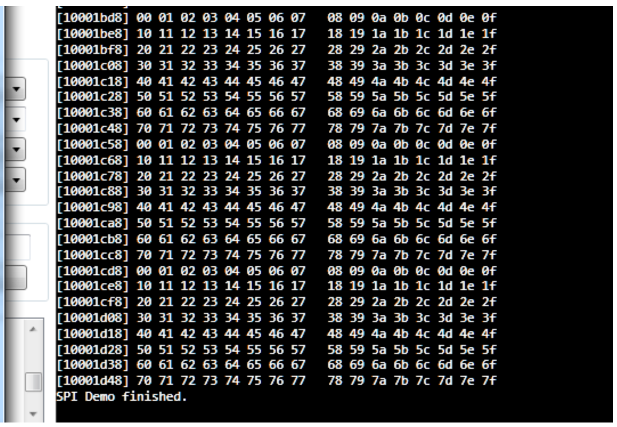
```
Build @ 11:40:45, Dec 19 2018

$8710c>Master TX done!
Master RX done!
SPI Master Read Data:
[10001b58] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001b68] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001b78] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001b88] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001b98] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001ba8] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001bb8] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
[10001bc8] 0f 0e 0d 0c 0b 0a 09 08   07 06 05 04 03 02 01 00
[10001bd8] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001be8] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001bf8] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001c08] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001c18] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001c28] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001c38] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
[10001c48] 0f 0e 0d 0c 0b 0a 09 08   07 06 05 04 03 02 01 00
[10001c58] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001c68] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001c78] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001c88] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001c98] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001ca8] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001cb8] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
[10001cc8] 0f 0e 0d 0c 0b 0a 09 08   07 06 05 04 03 02 01 00
[10001cd8] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001ce8] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001cf8] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001d08] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001d18] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001d28] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001d38] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
```



```
$8710c>SPI Slave Wait Read Done...
Slave TX done!
Slave RX done!
SPI Slave Read Data:
[10001b58] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001b68] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001b78] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001b88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001b98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001ba8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001bb8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001bc8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001bd8] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001be8] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001bf8] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001c08] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001c18] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001c28] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001c38] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001c48] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001c58] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001c68] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001c78] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001c88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001c98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001ca8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001cb8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001cc8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001cd8] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001ce8] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001cf8] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001d08] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001d18] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001d28] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001d38] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
```



```
[10001bd8] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001be8] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001bf8] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001c08] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001c18] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001c28] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001c38] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
[10001c48] 0f 0e 0d 0c 0b 0a 09 08   07 06 05 04 03 02 01 00
[10001c58] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001c68] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001c78] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001c88] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001c98] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001ca8] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001cb8] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
[10001cc8] 0f 0e 0d 0c 0b 0a 09 08   07 06 05 04 03 02 01 00
[10001cd8] 7f 7e 7d 7c 7b 7a 79 78   77 76 75 74 73 72 71 70
[10001ce8] 6f 6e 6d 6c 6b 6a 69 68   67 66 65 64 63 62 61 60
[10001cf8] 5f 5e 5d 5c 5b 5a 59 58   57 56 55 54 53 52 51 50
[10001d08] 4f 4e 4d 4c 4b 4a 49 48   47 46 45 44 43 42 41 40
[10001d18] 3f 3e 3d 3c 3b 3a 39 38   37 36 35 34 33 32 31 30
[10001d28] 2f 2e 2d 2c 2b 2a 29 28   27 26 25 24 23 22 21 20
[10001d38] 1f 1e 1d 1c 1b 1a 19 18   17 16 15 14 13 12 11 10
[10001d48] 0f 0e 0d 0c 0b 0a 09 08   07 06 05 04 03 02 01 00
SPI Demo finished.
```



```
[10001bd8] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001be8] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001bf8] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001c08] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001c18] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001c28] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001c38] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001c48] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001c58] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001c68] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001c78] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001c88] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001c98] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001ca8] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001cb8] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001cc8] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
[10001cd8] 00 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
[10001ce8] 10 11 12 13 14 15 16 17   18 19 1a 1b 1c 1d 1e 1f
[10001cf8] 20 21 22 23 24 25 26 27   28 29 2a 2b 2c 2d 2e 2f
[10001d08] 30 31 32 33 34 35 36 37   38 39 3a 3b 3c 3d 3e 3f
[10001d18] 40 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
[10001d28] 50 51 52 53 54 55 56 57   58 59 5a 5b 5c 5d 5e 5f
[10001d38] 60 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
[10001d48] 70 71 72 73 74 75 76 77   78 79 7a 7b 7c 7d 7e 7f
SPI Demo finished.
```

2.9.6 spi_twoboard

IMAGE NAME:

flash_is_spi_twoboard_master.bin;

flash_is_spi_twoboard_slave.bin

This example describes how to use SPI read/write by mbed api.

The SPI Interface provides a "Serial Peripheral Interface" Master.

This interface can be used for communication with SPI slave devices, such as FLASH memory, LCD screens and other modules or integrated circuits.

In this example, we use config SPI_IS_AS_MASTER to decide if device is master or slave.

- If SPI_IS_AS_MASTER is 1, then device is master.
- If SPI_IS_AS_MASTER is 0, then device is slave.

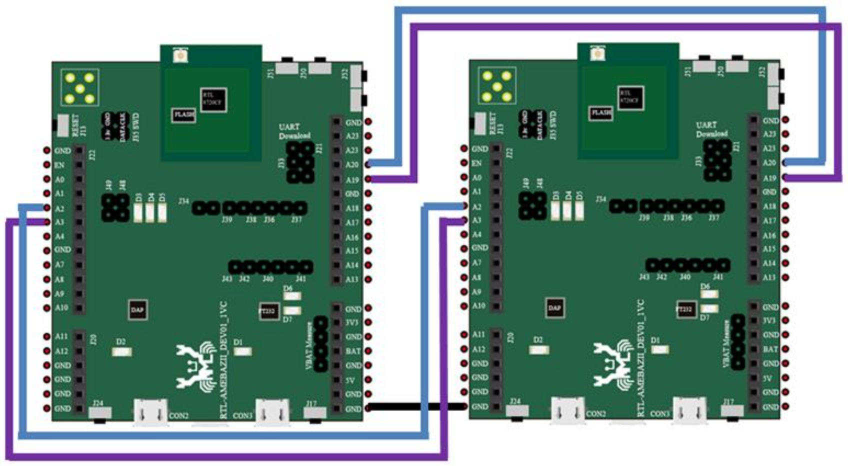
Connections:

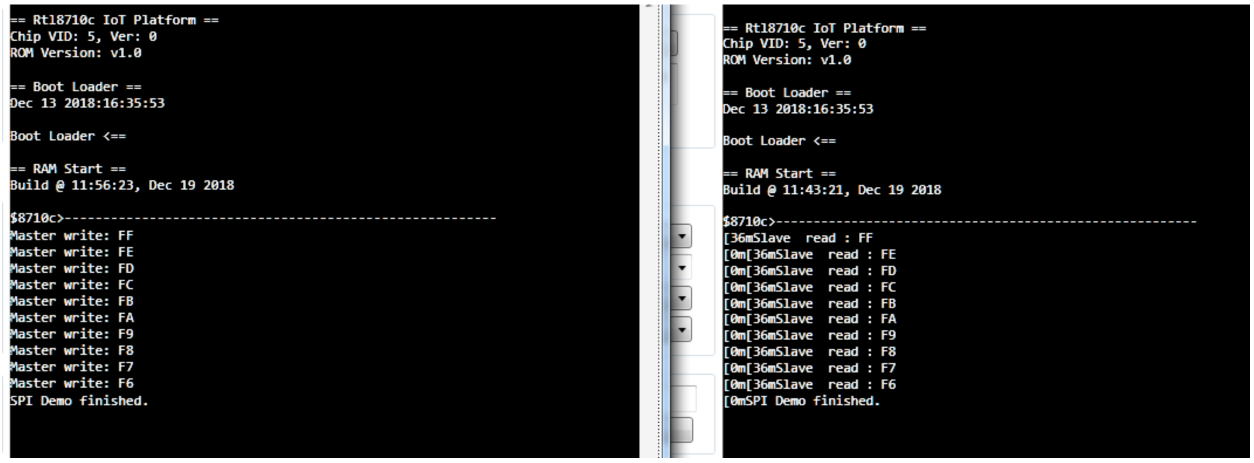
Master board	<---------->	Slave board
master's MOSI (PA_19)	<---------->	slave's MOSI (PA_19)
master's MISO (PA_20)	<---------->	slave's MISO (PA_20)
master's SCLK (PA_3)	<---------->	slave's SCLK (PA_3)
master's CS (PA_2)	<---------->	slave's CS (PA_2)

This example shows master sends data to slave.

We bootup slave first, and then bootup master.

Then log will presents that master sending data to slave.





```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 11:56:23, Dec 19 2018

$8710c>------------------------------------------------------
Master write: FF
Master write: FE
Master write: FD
Master write: FC
Master write: FB
Master write: FA
Master write: F9
Master write: F8
Master write: F7
Master write: F6
SPI Demo finished.
```

```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 11:43:21, Dec 19 2018

$8710c>------------------------------------------------------
[36mSlave  read : FF
[0m[36mSlave  read : FE
[0m[36mSlave  read : FD
[0m[36mSlave  read : FC
[0m[36mSlave  read : FB
[0m[36mSlave  read : FA
[0m[36mSlave  read : F9
[0m[36mSlave  read : F8
[0m[36mSlave  read : F7
[0m[36mSlave  read : F6
[0mSPI Demo finished.
```

2.10UART

Please note that for all UART related examples,

- The supported UART, UART0 to UART2
  - UART0_RX, PA_12, PA_13
  - UART0_TX, PA_11, PA_14
  - UART0_CTS, PA_10
  - UART0_RTS, PA_9
  - UART1_RX, PA_0, PA_2
  - UART1_TX, PA_1, PA_3
  - UART1_CTS, PA_4
  - UART1_RTS, NA
  - UART2_RX, PA_15
  - UART2_TX, PA_16
  - UART2_CTS, PA_19
  - UART2_RTS, PA_20
- Please off JTAG/SWD, when using PA_0, PA_1, PA_2, PA_3, and PA_4.
- PA_9, PA_10, PA_11, and PA_12 only available on RTL8720CF.

2.10.1 uart

IMAGE NAME:

flash_is_uart.bin

This example describes how to use UART to communicate with PC.

Required Components:

1. USBtoTTL adapter

Connect to PC

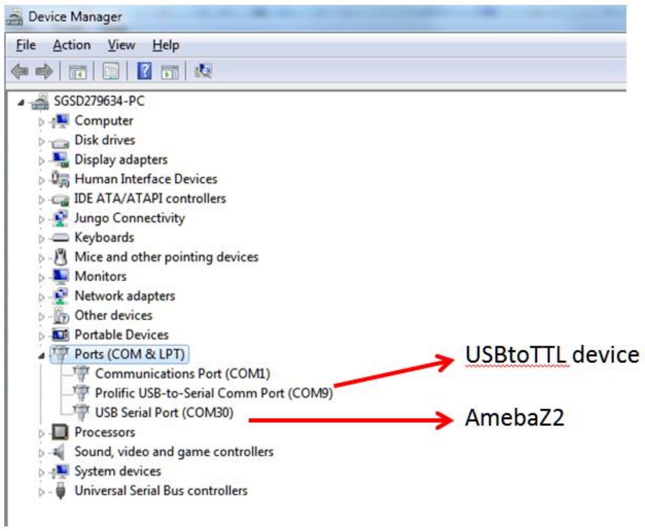
- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

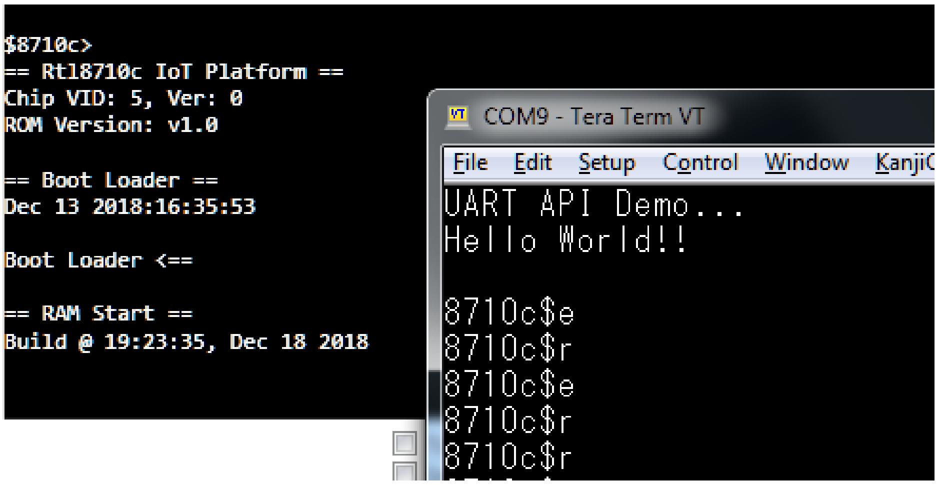
This example shows:

User input will be received by demo board, and demo board will loopback the received character with a prompt string "8710c$".

There will be two serial ports we concerned, one is our device, and the other is USBtoTTL adapter.



Open serial terminal on serial port of USBtoTTL:


```
$8710c>
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:23:35, Dec 18 2018
```

```
UART API Demo...
Hello World!!

8710c$e
8710c$r
8710c$e
8710c$r
8710c$r
```

2.10.2 uart_auto_flow_ctrl

IMAGE NAME:

flash_is_uart_auto_flow_ctrl.bin

This example demo the function of Auto Flow control.

Please connect 2 boards to run this example.

Required Components:

1. 2 EV boards

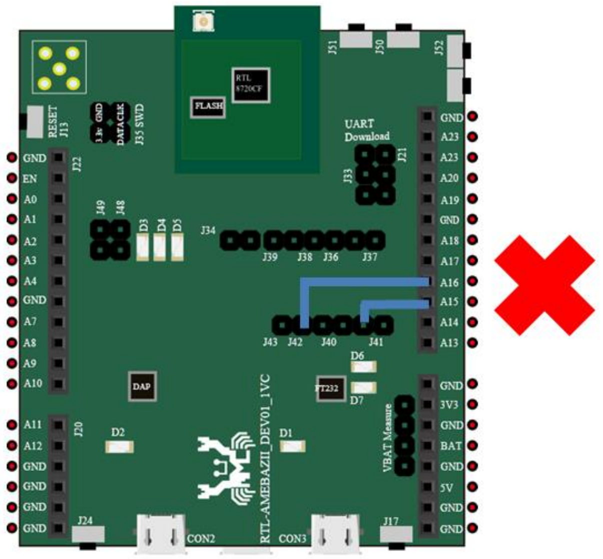
The UART2 is default Log UART.

In this example, the log UART is shifted to UART1.

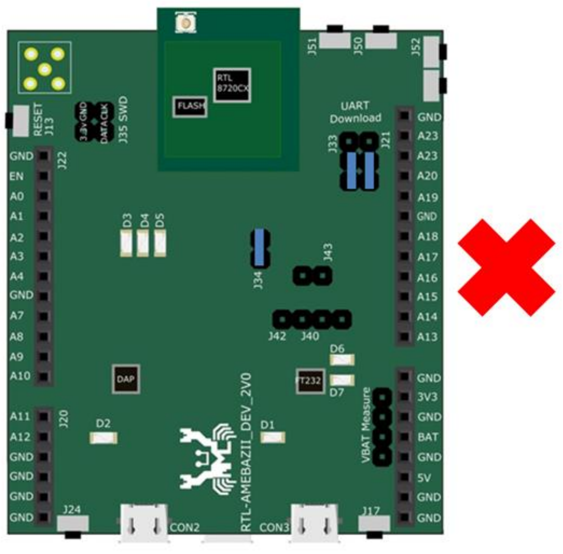
UART2 is tested for the example demo (auto Flow control).

Disconnection for both boards

PA15	<---------->	log_uart_rx
PA16	<---------->	log_uart_tx



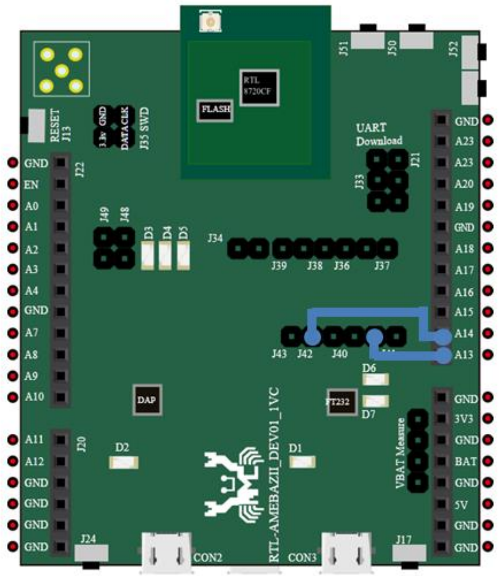
EVB V1.0



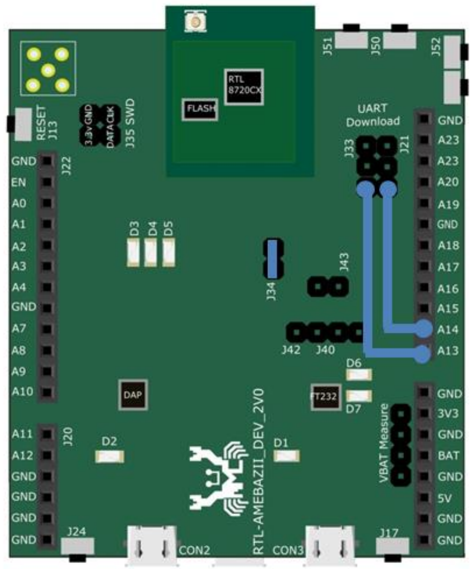
EVB V2.0

Connection for both boards

PA13	<---------->	log_uart_rx
PA14	<---------->	log_uart_tx



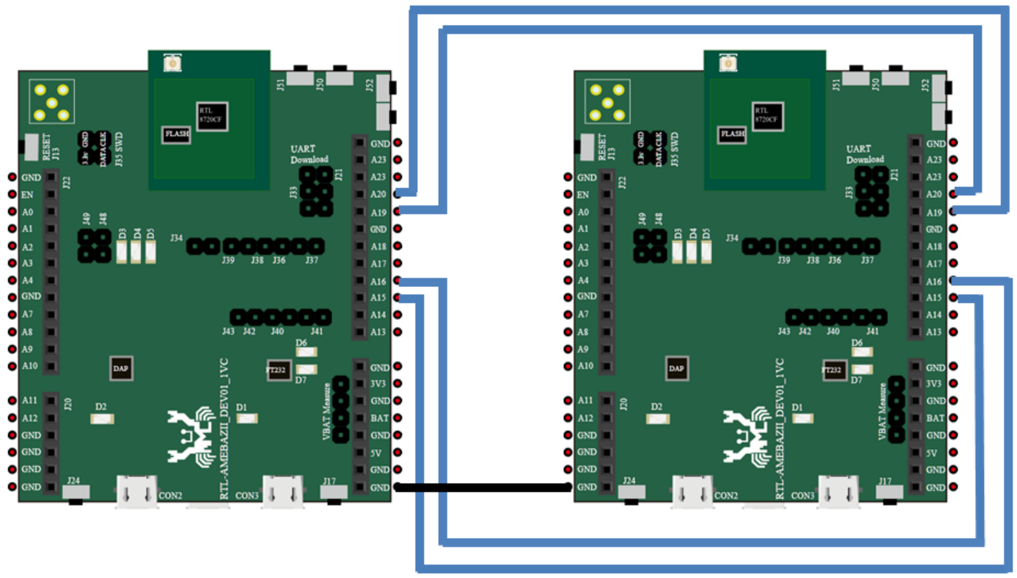
EVB V1.0



EVB V2.0

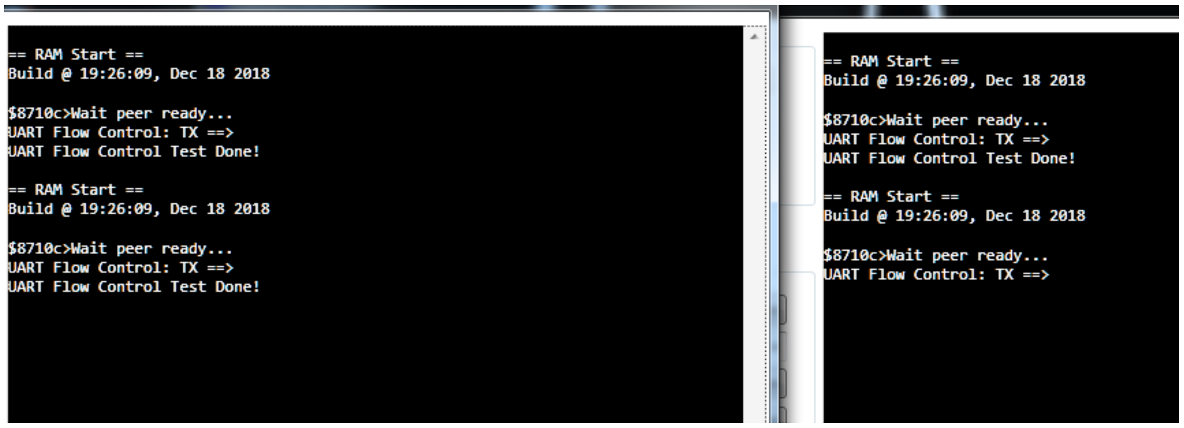
Connect to 2 boards

Board1	<---------->	Board2
PA19	<---------->	PA20
PA20	<---------->	PA19
PA15	<---------->	PA16
PA16	<---------->	PA15
GND	<---------->	GND



This example shows:

- The first powered board will be the TX side, the other one will be the RX side.
- The RX side will make some delay every 16-bytes received, by this way we can trigger the flow control mechanism.



```
== RAM Start ==
Build @ 19:26:09, Dec 18 2018

$8710c>Wait peer ready...
UART Flow Control: TX ==>
UART Flow Control Test Done!

== RAM Start ==
Build @ 19:26:09, Dec 18 2018

$8710c>Wait peer ready...
UART Flow Control: TX ==>
UART Flow Control Test Done!
```

```
== RAM Start ==
Build @ 19:26:09, Dec 18 2018

$8710c>Wait peer ready...
UART Flow Control: TX ==>
UART Flow Control Test Done!

== RAM Start ==
Build @ 19:26:09, Dec 18 2018

$8710c>Wait peer ready...
UART Flow Control: TX ==>
```

2.10.3 uart_clock

IMAGE NAME:

flash_is_uart_clock.bin

This example describes how to use UART TX to simulate clock source

Required Components:

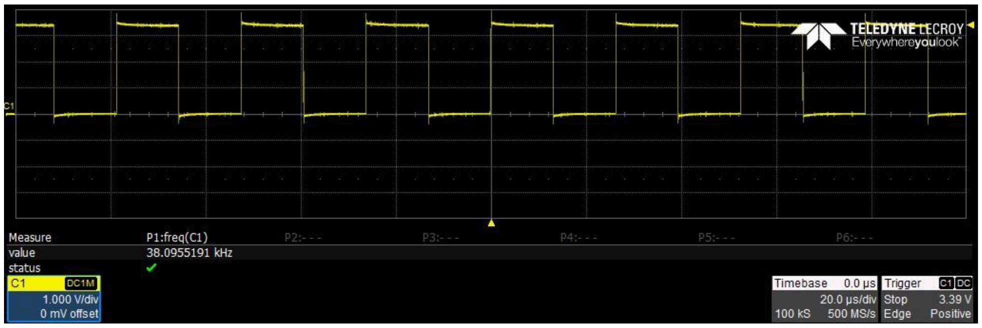
1. Oscilloscope

Connect to PC

- Connect Ground: connect to GND of oscilloscope
- Use UART0
  - GPIOA_13 as UART0_RX connect NOTHING
  - GPIOA_14 as UART0_TX connect to probe of oscilloscope

This example shows:

- Clock signal output from UART1_TX to oscilloscope



2.10.4 uart_DMA_rx_continuous

IMAGE NAME:

flash_is_uart_DMA_rx_continuous.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

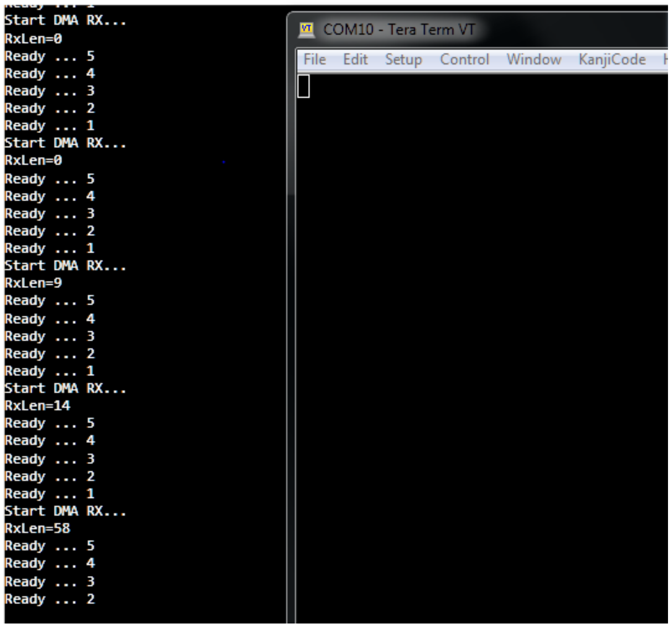
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
    - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
    - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
Start DMA RX...
RxLen=0
Ready ... 5
Ready ... 4
Ready ... 3
Ready ... 2
Ready ... 1
Start DMA RX...
RxLen=0
Ready ... 5
Ready ... 4
Ready ... 3
Ready ... 2
Ready ... 1
Start DMA RX...
RxLen=9
Ready ... 5
Ready ... 4
Ready ... 3
Ready ... 2
Ready ... 1
Start DMA RX...
RxLen=14
Ready ... 5
Ready ... 4
Ready ... 3
Ready ... 2
Ready ... 1
Start DMA RX...
RxLen=58
Ready ... 5
Ready ... 4
Ready ... 3
Ready ... 2
```

2.10.5 uart_irq

IMAGE NAME:

flash_is_uart_irq.bin

This example describes how to use UART to communicate with PC.

Required Components:

1. USBtoTTL adapter

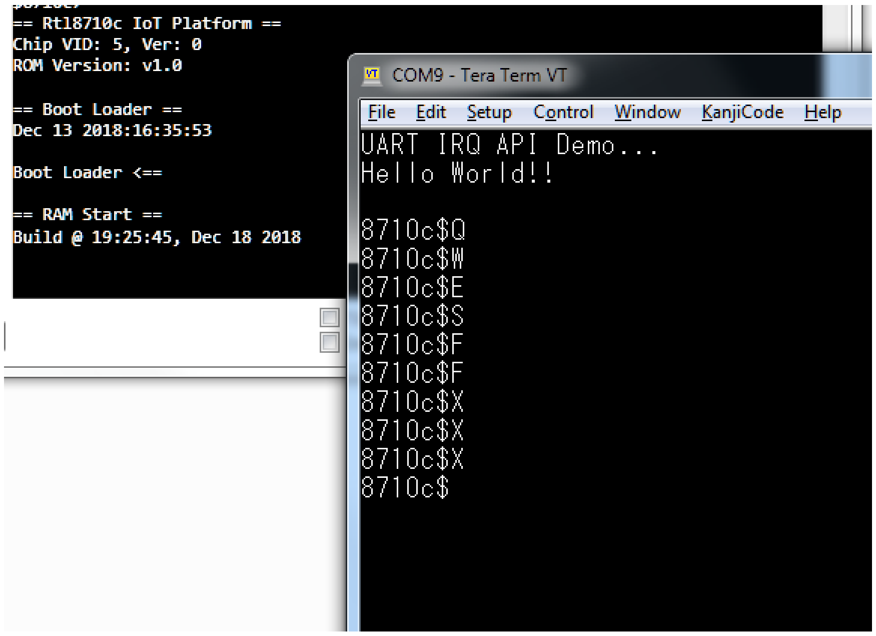
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
    - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
    - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send a prompt string "8710c$" to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:25:45, Dec 18 2018
```

```
UART IRQ API Demo...
Hello World!!

8710c$Q
8710c$W
8710c$E
8710c$S
8710c$F
8710c$F
8710c$X
8710c$X
8710c$X
8710c$
```

2.10.6 uart_stream_2_threads

IMAGE NAME:

flash_is_uart_stream_2_threads.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

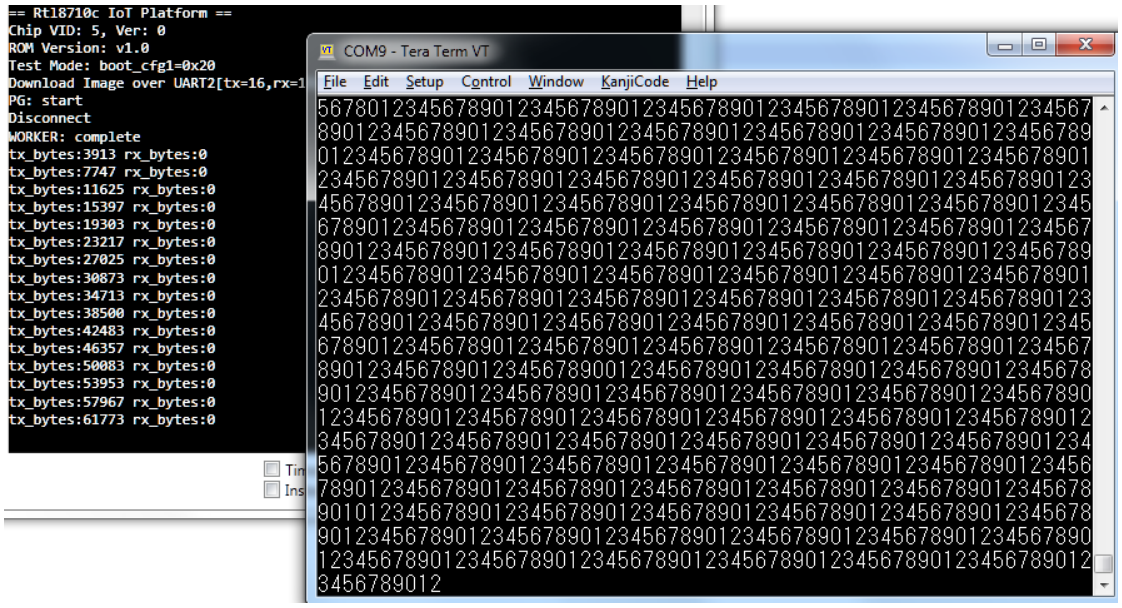
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x20
Download Image over UART2[tx=16,rx=1
PG: start
Disconnect
WORKER: complete
tx_bytes:3913 rx_bytes:0
tx_bytes:7747 rx_bytes:0
tx_bytes:11625 rx_bytes:0
tx_bytes:15397 rx_bytes:0
tx_bytes:19303 rx_bytes:0
tx_bytes:23217 rx_bytes:0
tx_bytes:27025 rx_bytes:0
tx_bytes:30873 rx_bytes:0
tx_bytes:34713 rx_bytes:0
tx_bytes:38500 rx_bytes:0
tx_bytes:42483 rx_bytes:0
tx_bytes:46357 rx_bytes:0
tx_bytes:50083 rx_bytes:0
tx_bytes:53953 rx_bytes:0
tx_bytes:57967 rx_bytes:0
tx_bytes:61773 rx_bytes:0
```

```
56780123456789012345678901234567890123456789012345678901234567
89012345678901234567890123456789012345678901234567890123456789
01234567890123456789012345678901234567890123456789012345678901
23456789012345678901234567890123456789012345678901234567890123
45678901234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890123456789012345678901234567
89012345678901234567890123456789012345678901234567890123456789
01234567890123456789012345678901234567890123456789012345678901
23456789012345678901234567890123456789012345678901234567890123
45678901234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890123456789012345678901234567
89012345678901234567890012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234
56789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678
90101234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012
3456789012
```

2.10.7 uart_stream_4_threads

IMAGE NAME:

flash_is_uart_stream_4_threads.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

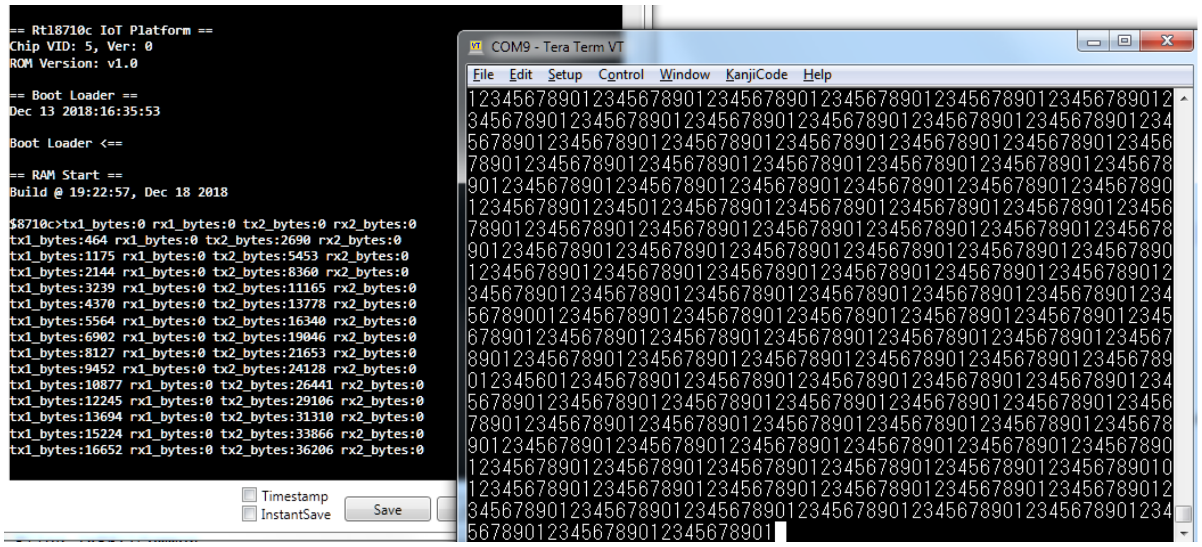
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:22:57, Dec 18 2018

$8710c>tx1_bytes:0 rx1_bytes:0 tx2_bytes:0 rx2_bytes:0
tx1_bytes:464 rx1_bytes:0 tx2_bytes:2690 rx2_bytes:0
tx1_bytes:1175 rx1_bytes:0 tx2_bytes:5453 rx2_bytes:0
tx1_bytes:2144 rx1_bytes:0 tx2_bytes:8360 rx2_bytes:0
tx1_bytes:3239 rx1_bytes:0 tx2_bytes:11165 rx2_bytes:0
tx1_bytes:4370 rx1_bytes:0 tx2_bytes:13778 rx2_bytes:0
tx1_bytes:5564 rx1_bytes:0 tx2_bytes:16340 rx2_bytes:0
tx1_bytes:6902 rx1_bytes:0 tx2_bytes:19046 rx2_bytes:0
tx1_bytes:8127 rx1_bytes:0 tx2_bytes:21653 rx2_bytes:0
tx1_bytes:9452 rx1_bytes:0 tx2_bytes:24128 rx2_bytes:0
tx1_bytes:10877 rx1_bytes:0 tx2_bytes:26441 rx2_bytes:0
tx1_bytes:12245 rx1_bytes:0 tx2_bytes:29106 rx2_bytes:0
tx1_bytes:13694 rx1_bytes:0 tx2_bytes:31310 rx2_bytes:0
tx1_bytes:15224 rx1_bytes:0 tx2_bytes:33866 rx2_bytes:0
tx1_bytes:16652 rx1_bytes:0 tx2_bytes:36206 rx2_bytes:0
```

```
12345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234
56789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890
12345678901234501234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234
56789001234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890123456789012345678901234567
89012345678901234567890123456789012345678901234567890123456789
01234560123456789012345678901234567890123456789012345678901234
56789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789010
12345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234
567890123456789012345678901
```

2.10.8 uart_stream_dma

IMAGE NAME:

flash_is_uart_stream_dma.bin

This example describes how to use UART to communicate with PC.

Required Components:

1. USBtoTTL adapter

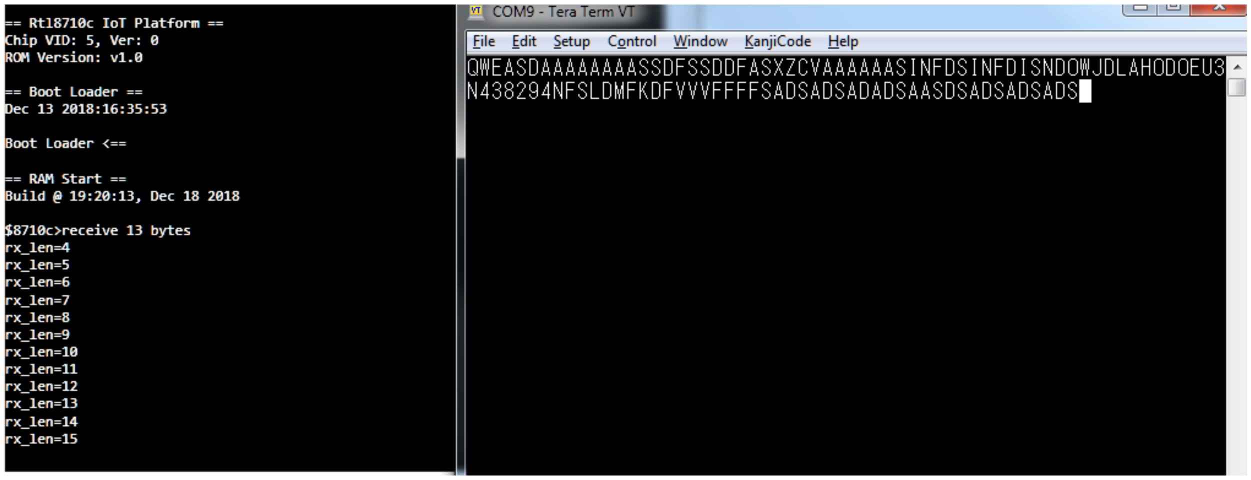
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
    - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
    - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The UART Rev DMA mode is used to receive characters from the PC, and then loopback them to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:20:13, Dec 18 2018

$8710c>receive 13 bytes
rx_len=4
rx_len=5
rx_len=6
rx_len=7
rx_len=8
rx_len=9
rx_len=10
rx_len=11
rx_len=12
rx_len=13
rx_len=14
rx_len=15
```

```
QWEASDAAAAAAAASSDFSSDDFASXZCVAAAAAASINFDSINFDISNDOWJDLAHODOEU3
N438294NFSLDMFKDFVVVFFFFSADSADSADADSAASDSADSADSADS
```

2.10.9 uart_stream_dma_rx_timeout

IMAGE NAME:

flash_is_uart_stream_dma_rx_timeout.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

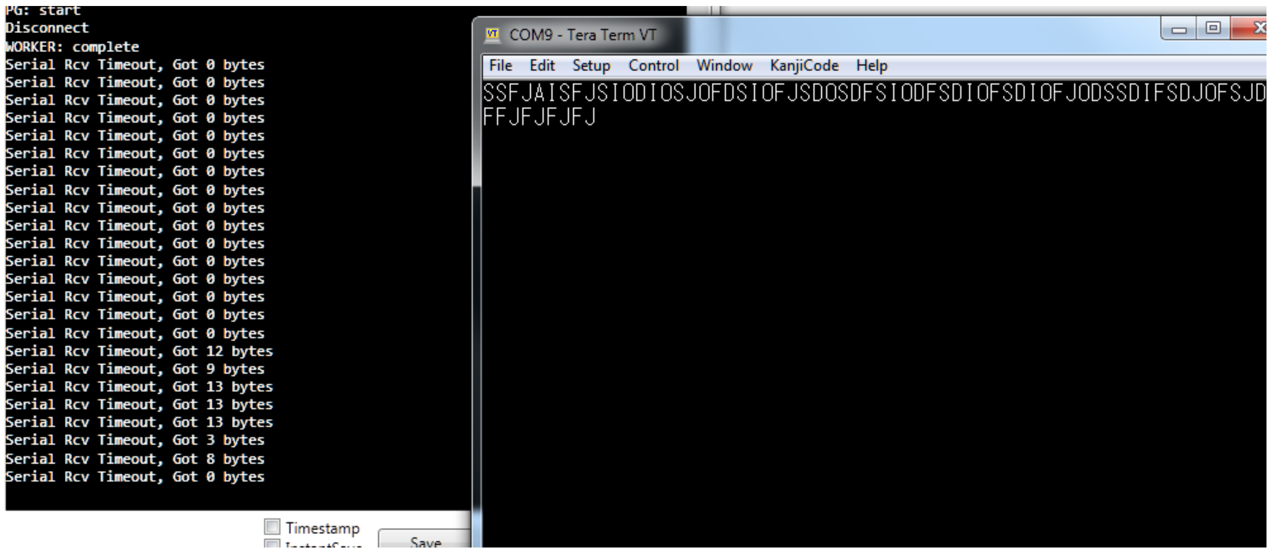
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
PG: start
Disconnect
WORKER: complete
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 12 bytes
Serial Rcv Timeout, Got 9 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 3 bytes
Serial Rcv Timeout, Got 8 bytes
Serial Rcv Timeout, Got 0 bytes
```

```
SSFJAISFJSIODIOSJOFDSIOFJSDOSDFSIODFSDIOFSDIOFJODSSDIFSDJOFSJD
FFJFJFJFJ
```

2.10.10 uart_stream_irq

IMAGE NAME:

flash_is_uart_stream_irq.bin

This example describes how to use UART to communicate with PC.

Required Components:

1. USBtoTTL adapter

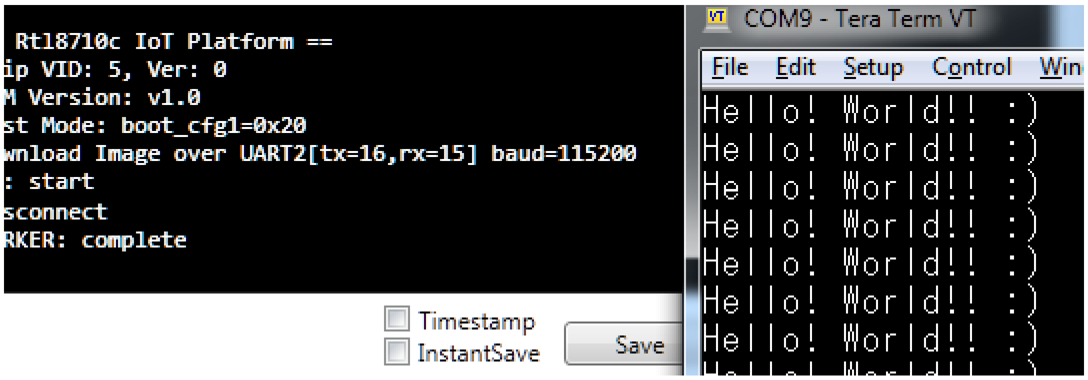
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive 8 characters from the PC, and then loopback them to the PC.



```
 Rtl8710c IoT Platform ==
ip VID: 5, Ver: 0
M Version: v1.0
st Mode: boot_cfg1=0x20
wnload Image over UART2[tx=16,rx=15] baud=115200
: start
sconnect
RKER: complete
```

```
Hello! World!! :)
Hello! World!! :)
Hello! World!! :)
Hello! World!! :)
Hello! World!! :)
Hello! World!! :)
Hello! World!! :)
```

2.10.11 uart_stream_rx_timeout

IMAGE NAME:

flash_is_uart_stream_rx_timeout.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

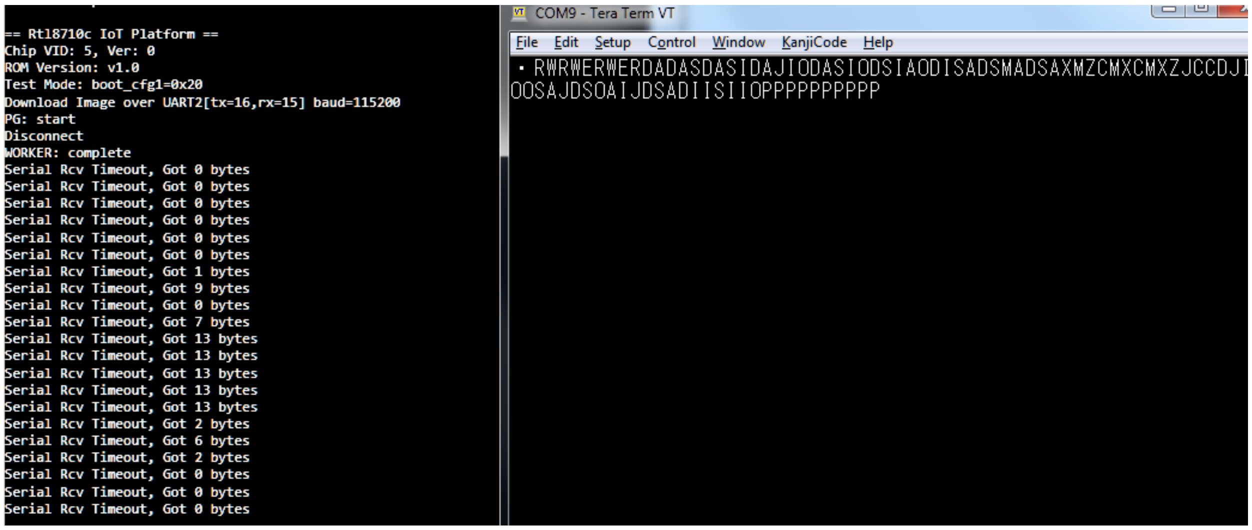
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 38400, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x20
Download Image over UART2[tx=16,rx=15] baud=115200
PG: start
Disconnect
WORKER: complete
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 1 bytes
Serial Rcv Timeout, Got 9 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 7 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 13 bytes
Serial Rcv Timeout, Got 2 bytes
Serial Rcv Timeout, Got 6 bytes
Serial Rcv Timeout, Got 2 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
Serial Rcv Timeout, Got 0 bytes
```

```
・RWRWERWERDADASDASIDAJIODASIODSIAODISADSMADSAXMZCMXCMXZJCCDJI
OOSAJDSOAIJDSADIISIIOPPPPPPPPPP
```

2.10.12 uart_stream_rx_timeout_by_GTimer

IMAGE NAME:

flash_is_uart_stream_rx_timeout_by_GTimer.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

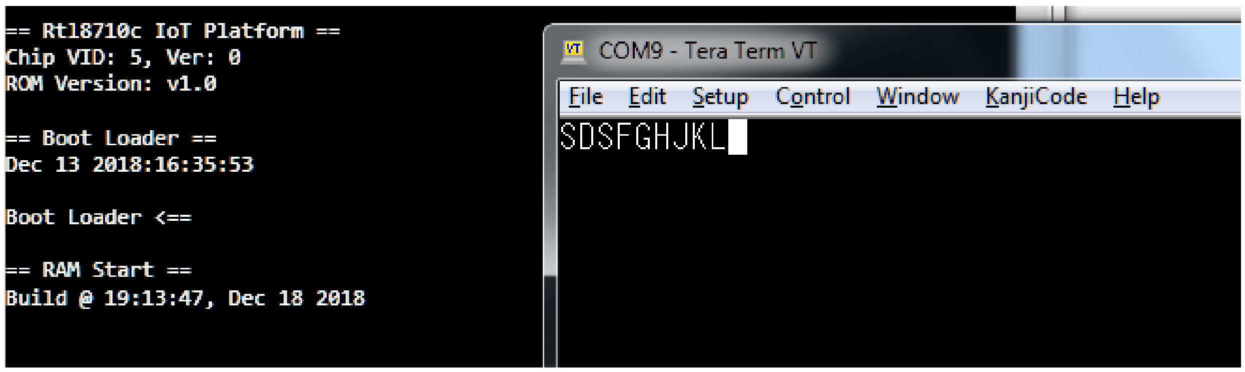
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
    - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
    - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 115200, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- Gtimer is used for UART RX timeout.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:13:47, Dec 18 2018
```

```
SDSFGHJKL
```

2.10.13 uart_stream_rx_timeout_by_semaphore_iar

IMAGE NAME:

flash_is_uart_stream_rx_timeout_by_semaphore_iar.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

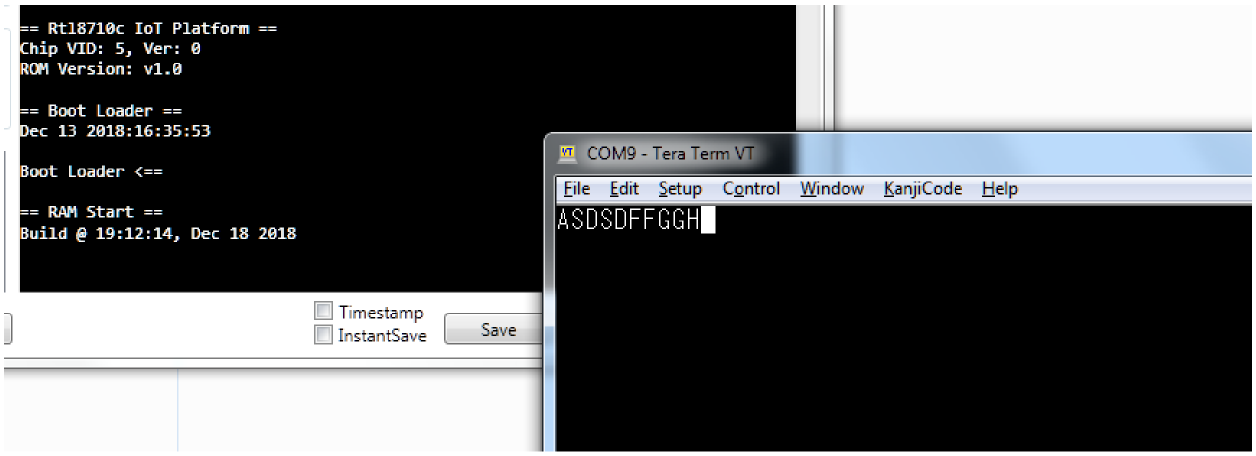
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
    - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
    - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 115200, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:12:14, Dec 18 2018
```

```
ASDSDFFGGH
```

2.10.14 uart_stream_rx_timeout_by_SoftTimer

IMAGE NAME:

flash_is_uart_stream_rx_timeout_by_SoftTimer.bin

This example describes how to use UART RX API with timeout.

Required Components:

USBtoTTL adapter

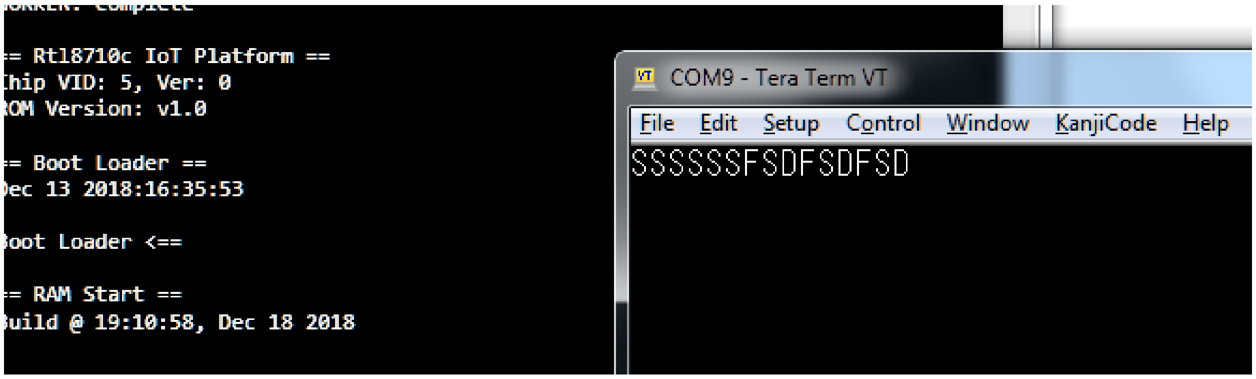
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 115200, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- soft timer in freertos is used for uart rx timeout mechanism



```
== Rtl8710c IoT Platform ==
hip VID: 5, Ver: 0
OM Version: v1.0

== Boot Loader ==
ec 13 2018:16:35:53

oot Loader <==

== RAM Start ==
uild @ 19:10:58, Dec 18 2018
```

```
SSSSSSFSDFSDFSD
```

2.10.15 uart_stream_rx_timeout_by_WaitSemaphore

IMAGE NAME:

flash_is_uart_stream_rx_timeout_by_WaitSemaphore.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

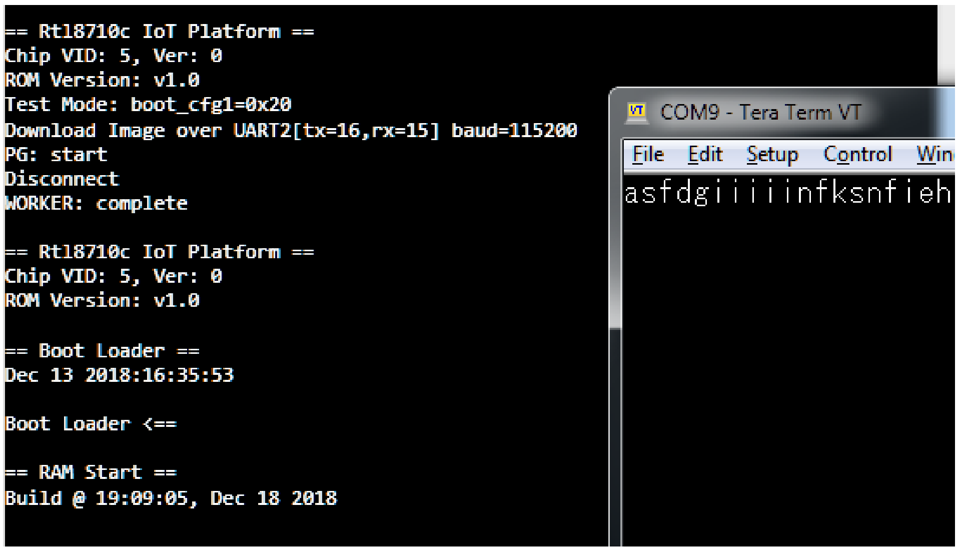
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 115200, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC, and then loopback them to the PC.
- semaphore in freertos is used for uart rx dma mechanism.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x20
Download Image over UART2[tx=16,rx=15] baud=115200
PG: start
Disconnect
WORKER: complete

== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:09:05, Dec 18 2018
```

```
asfdgiiiiinfksnfieh
```

2.10.16 uart_stream_tx_rx_concurrent_iar

IMAGE NAME:

flash_is_uart_stream_tx_rx_concurrent_iar.bin

This example describes how to use UART RX API with timeout.

Required Components:

1. USBtoTTL adapter

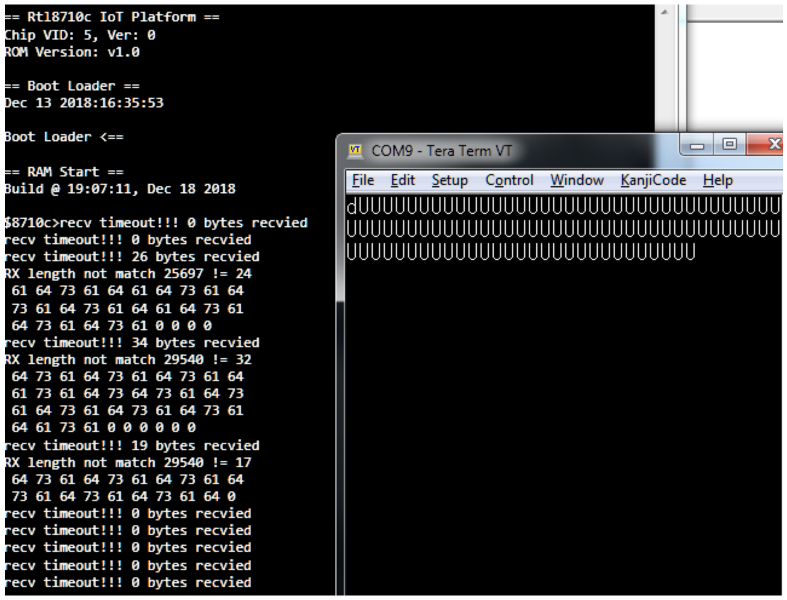
Connect to PC

- Connect Ground: connect to GND pin via USBtoTTL adapter
- Use UART0
  - GPIOA_13 as UART0_RX connect to TX of USBtoTTL adapter
  - GPIOA_14 as UART0_TX connect to RX of USBtoTTL adapter

Open Super terminal or Teraterm and set baud rate to 115200, 1 stopbit, no parity, no flow control.

This example shows:

- The RX data ready interrupt service routine is used to receive characters from the PC and then loopback them to the PC.
- The TX done interrupt service routine will send the received string to the PC.



```
== Rtl8710c IoT Platform ==
Chip VID: 5, Ver: 0
ROM Version: v1.0

== Boot Loader ==
Dec 13 2018:16:35:53

Boot Loader <==

== RAM Start ==
Build @ 19:07:11, Dec 18 2018

$8710c>recv timeout!!! 0 bytes recvied
recv timeout!!! 0 bytes recvied
recv timeout!!! 26 bytes recvied
RX length not match 25697 != 24
 61 64 73 61 64 61 64 73 61 64
 73 61 64 73 61 64 61 64 73 61
 64 73 61 64 73 61 0 0 0 0
recv timeout!!! 34 bytes recvied
RX length not match 29540 != 32
 64 73 61 64 73 61 64 73 61 64
 61 73 61 64 73 64 73 61 64 73
 61 64 73 61 64 73 61 64 73 61
 64 61 73 61 0 0 0 0 0 0
recv timeout!!! 19 bytes recvied
RX length not match 29540 != 17
 64 73 61 64 73 61 64 73 61 64
 73 61 64 73 61 64 73 61 64 0
recv timeout!!! 0 bytes recvied
recv timeout!!! 0 bytes recvied
recv timeout!!! 0 bytes recvied
recv timeout!!! 0 bytes recvied
recv timeout!!! 0 bytes recvied
```

```
dUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUU
```

2.11WATCHDOG

2.11.1 WatchDog

IMAGE NAME:

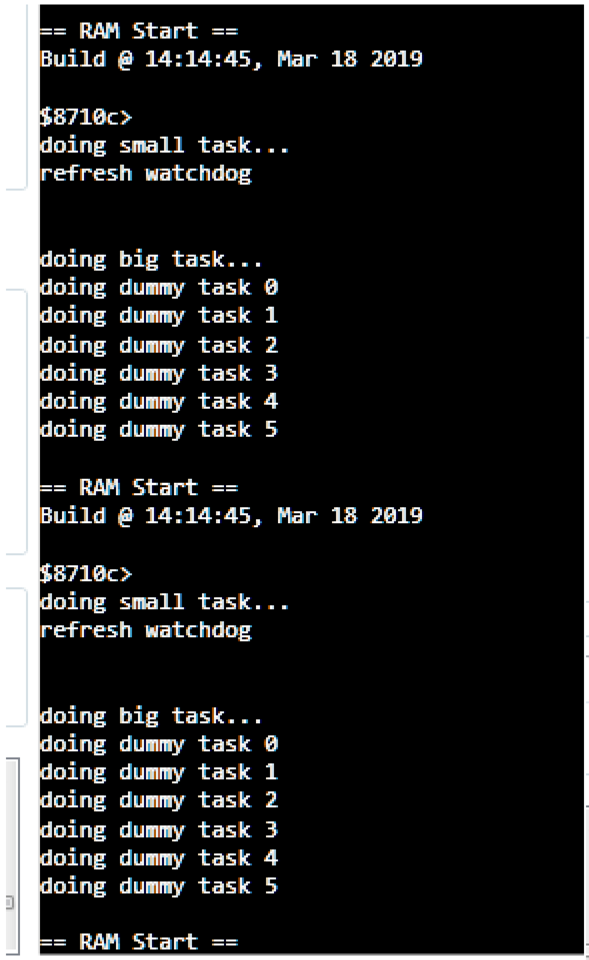
flash_is_watchdog_Reset.bin;

flash_is_watchdog_IRQ

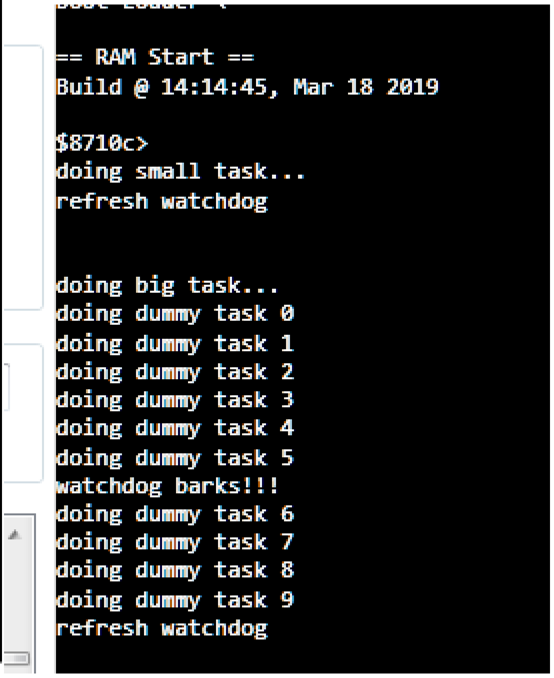
This example describes how to use watchdog api.

In this example, watchdog is setup to 5s timeout. Watchdog will not bark if we refresh it before timeout. The timer will be reloaded after refresh. Otherwise, it will restart system in default or call callback function if registered. It will not refresh the watchdog in time, watchdog will timeout and restart the system.

Note, EVB V1.0 has bug of WatchDog example. Test of EVB V1.0 can be skipped.



```
== RAM Start ==
Build @ 14:14:45, Mar 18 2019

$8710c>
doing small task...
refresh watchdog


doing big task...
doing dummy task 0
doing dummy task 1
doing dummy task 2
doing dummy task 3
doing dummy task 4
doing dummy task 5

== RAM Start ==
Build @ 14:14:45, Mar 18 2019

$8710c>
doing small task...
refresh watchdog


doing big task...
doing dummy task 0
doing dummy task 1
doing dummy task 2
doing dummy task 3
doing dummy task 4
doing dummy task 5

== RAM Start ==
```



```
== RAM Start ==
Build @ 14:14:45, Mar 18 2019

$8710c>
doing small task...
refresh watchdog


doing big task...
doing dummy task 0
doing dummy task 1
doing dummy task 2
doing dummy task 3
doing dummy task 4
doing dummy task 5
watchdog barks!!!
doing dummy task 6
doing dummy task 7
doing dummy task 8
doing dummy task 9
refresh watchdog
```