Introduction to Ameba-Z SDK

Compatible with Ameba-1

Content

- **Ameba-Z**
- SDK overview
- API of Components
- IDE Tool Demo
- MP Related

Ameba-Z Startup Document

- HW
  - UM0115 Realtek Ameba-Z Introduction.pdf
- Compare to Ameba-1
  - UM0116 Realtek Ameba-Z SDK change.pdf
- Datasheet
  - UM0114 Realtek Ameba-Z Data Sheet.pdf
- DEV board
  - UM0113 Realtek Ameba-Z DEV 1v0 User Manual.pdf
- IAR setup
  - UM0110 Realtek Ameba-Z build environment setup - iar.pdf
- Memory Layout
  - UM0111 Realtek Ameba-Z memory layout.pdf
- OTA
  - AN0110 Realtek Ameba-Z over the air firmware update.pdf

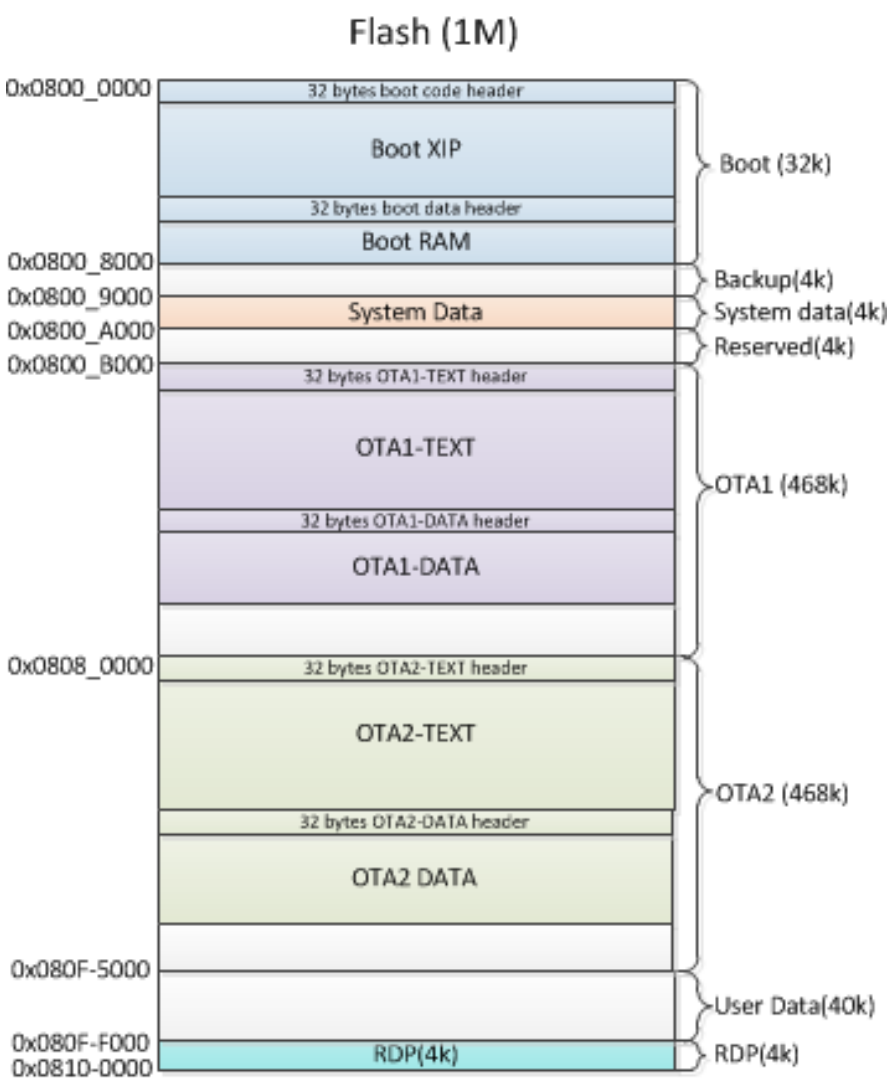
Ameba-Z New Software Feature

- XIP
  - About 140K SRAM free and 200K flash free
  - RF Calibration data in EFUSE
  - OTA mechanism
- Security
  - Hidden EFUSE key + RDP
  - Polar SSL 1.3.8 in ROM (Most code)
- High-speed Log UART
  - For normal debug download or UART upgrade
- Peripheral API
  - Both Light weight and Mbed API

Flash Layout (Ref: UM0111)

- Flash Layout
  - Bootloader: Image1
  - System data
  - OTAx image: Image2
- Image Header (32Bytes)
  - Signature
    - Flash calibration data for image1
    - String “81958711” for image2
  - Address
    - Code executes address after boot
    - ‘BOOT RAM’, ‘OTA1 DATA’ ‘OTA2 DATA’ is target RAM address
    - ‘BOOT XIP’. ‘OTA1 TEXT’, ‘OTA2 TEXT’ is Flash XIP address

	0x00	0x04	0x08	0x0C
0x00	Signature		Size	Address
0x10	reserved			
0x20	Firmware Code			



System Data (Ref: UM0111)

- Flash Offset: 0x9000
- OTA2 Flash Address
  - Consistent with Image2 Flash offset selected at compile time
- RDP Flash Address
  - Any address from OTA1 end
  - Length less than 4K
- Valid IMG2
  - Used for OTA1 and OTA2 switch

	0x00	0x04	0x08	0x0C
0x00	OTA2 Flash Address	Valid IMG2	Forth OTA1 GPIO	
0x10	RDP Flash Address	RDP Len (no checksum 4B)		
0x20	WORD1: SPI Speed WORD0: SPI Mode	WORD1: Flash Size WORD0: Flash ID		
	reserved			
0x100 ~ 0x147	USB Parameter			
	reserved			
0x200	ADC Prameter			

Validate Image2 (32 Bits) ☒ 0 ☐ 1

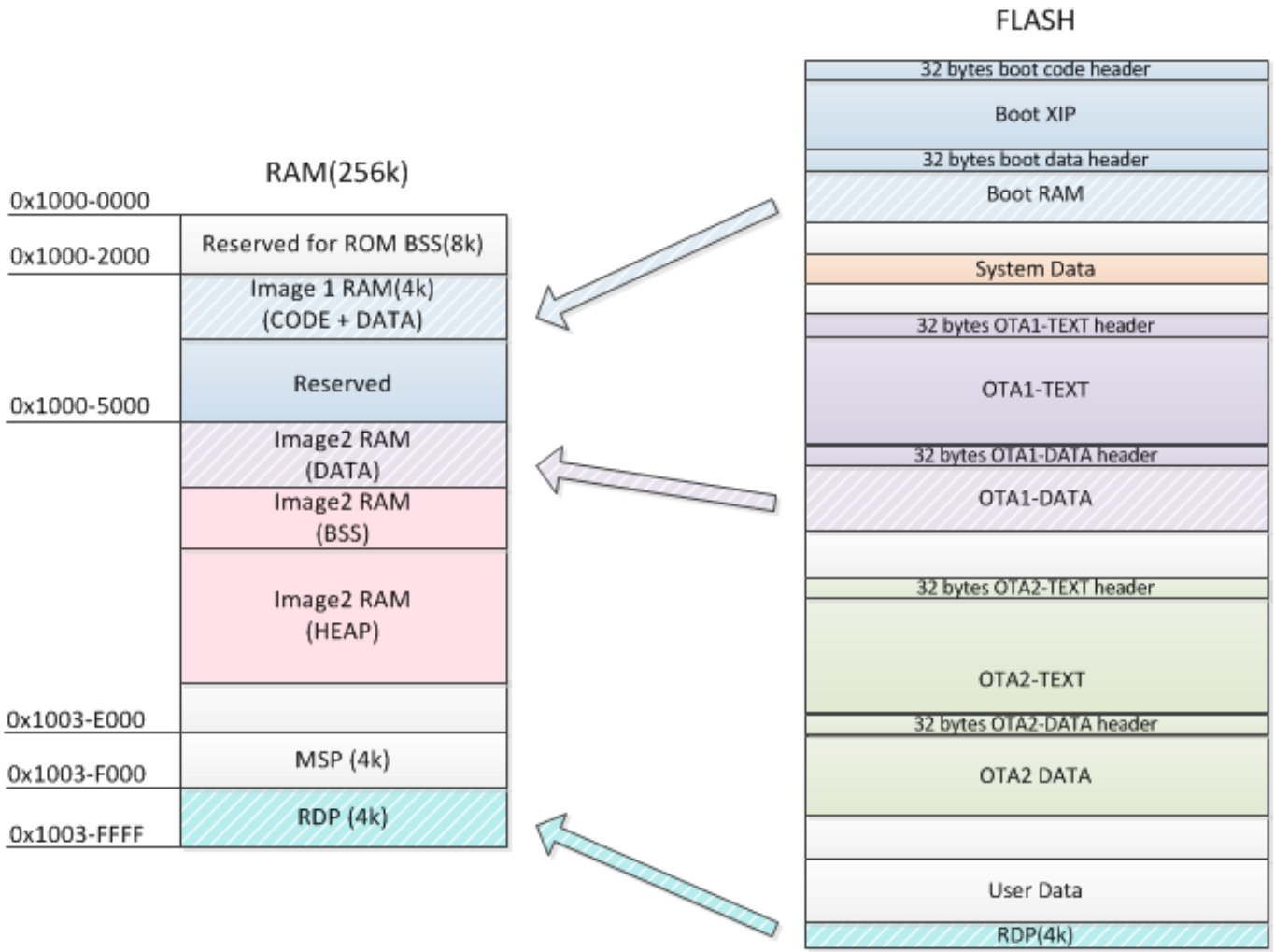
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0xFFFFFFFF Boot OTA1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0xFFFFFFFE Boot OTA2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0xFFFFFFFC Boot OTA1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0xFFFFFFF8 Boot OTA2
• • • • • •																																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00000000 Boot OTA1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0xFFFFFFFE Boot OTA2

System data

- Flash speed
  - 0xFFFF: 100MHz
  - 0x7FFF: 83MHz
  - 0x3FFF: 71MHz
  - 0x1FFF: 62MHz
  - 0x0FFF: 55MHz
  - 0x07FF: 50MHz
  - 0x03FF: 45MHz
- Flash size
  - 0xFFFF: 2MB
  - 0x7FFF: 32M
  - 0x3FFF: 16M
  - 0x1FFF: 8MB
  - 0x0FFF: 4MB
  - 0x07FF: 2MB
  - 0x03FF: 1MB
- Flash SPI mode
  - 0xFFFF: Read quad IO, Address & Data 4 bits mode
  - 0x7FFF: Read quad O, Just data 4 bits mode
  - 0x3FFF: Read dual IO, Address & Data 2 bits mode
  - 0x1FFF: Read dual O, Just data 2 bits mode
  - 0x0FFF: 1 bit mode

RAM Layout (Ref: UM0111)

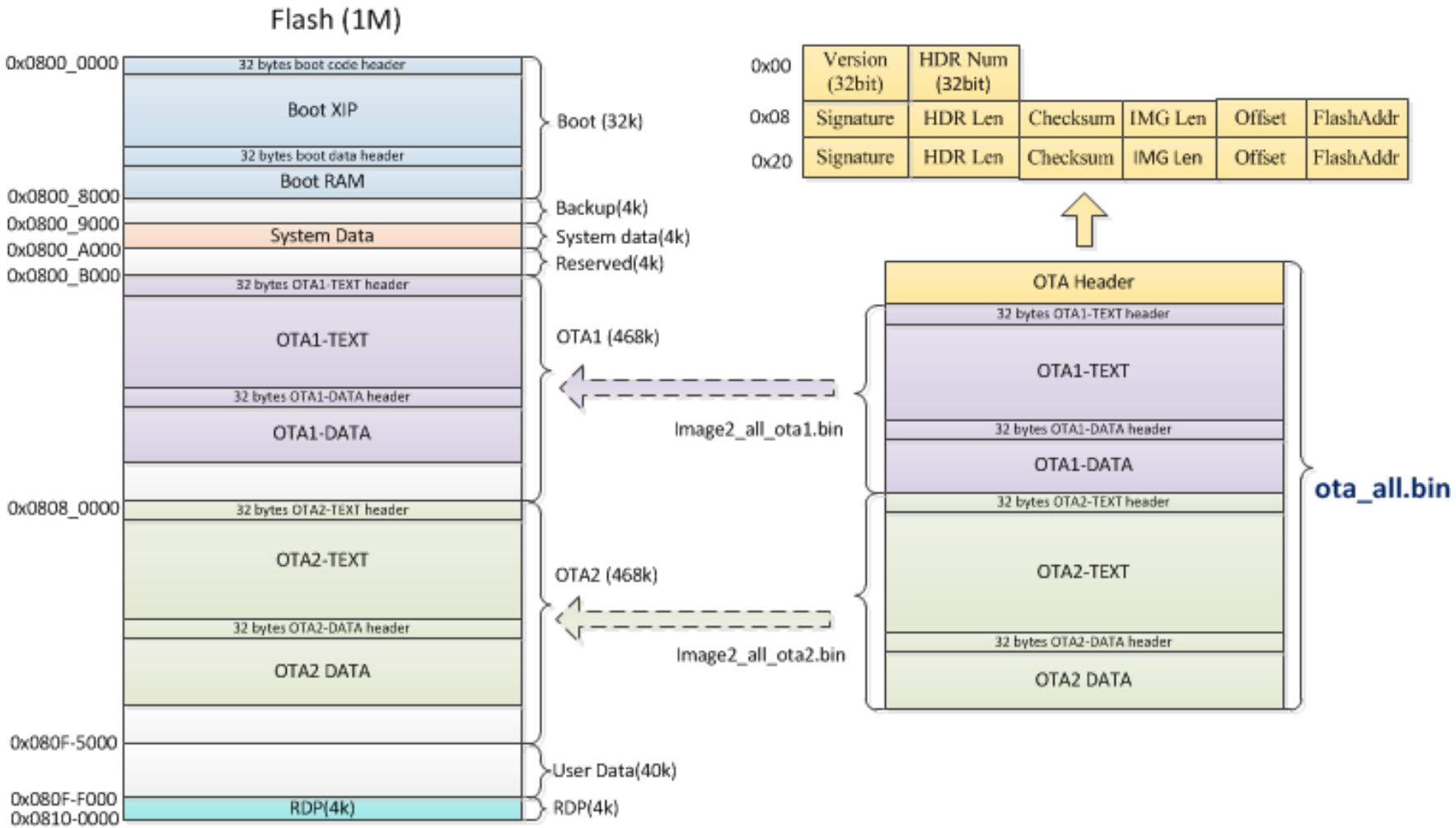
- SRAM layout
  - ROM BSS
  - Image 1 used SRAM
  - Image 2 used SRAM
  - MSP
  - RDP: Fixed location



Output Binary

- Boot loader
  - boot_all.bin
    - Boot loader code and initial value of data
- Application Image2
  - image2_all_ota1.bin
    - Application code and initial value of data
    - Run in flash 0x0800B000
  - image2_all_ota2.bin
    - Application code and initial value of data
    - Default run in flash 0x08080000
- OTA image
  - ota_all.bin
    - Integrated OTA header, image2_all_ota1.bin and image2_all_ota2.bin

OTA Update (Ref: AN0110)



OTA Header

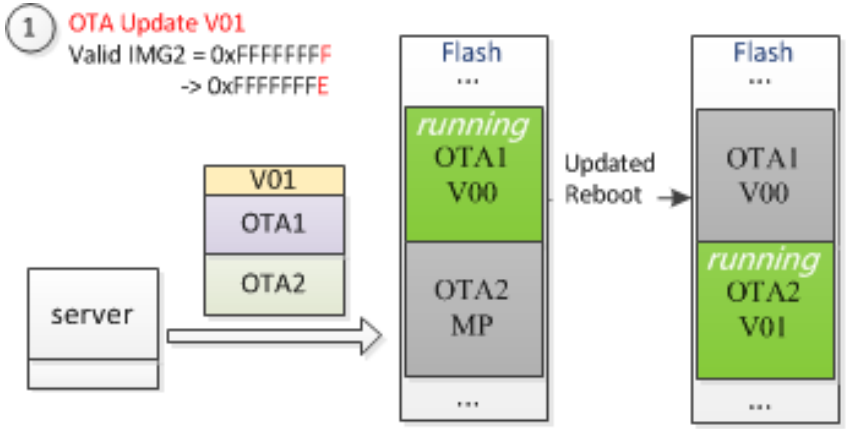
- Version
  - The version of OTA image
- Header Number
  - The number of OTA Entry Header
- OTA Entry Header
  - Signature
    - “OTA1” for OTA1, and “OTA2 ” for OTA2
  - Header Length
    - The length of OTAx header
  - Checksum
    - The checksum of OTAx image
  - Image Length
    - The size of OTAx image
  - Offset
    - The start position of OTAx in current image
  - Flash Address
    - Address in flash where OTAx will be programmed

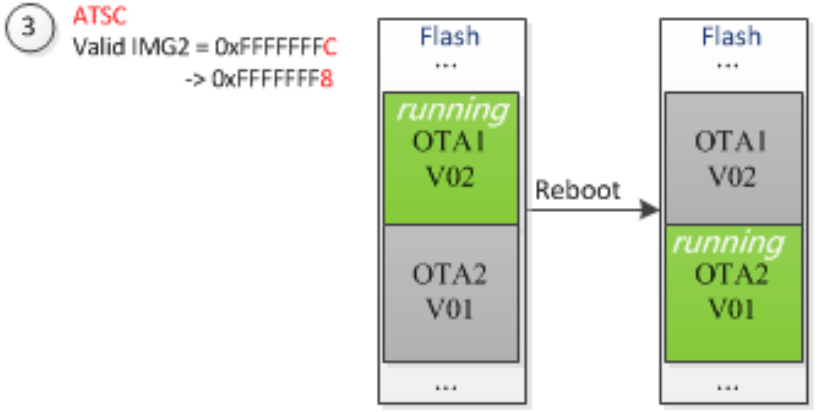
0x00	Version (32bit)	HDR Num (32bit)				
0x08	Signature	HDR Len	Checksum	IMG Len	Offset	FlashAddr
0x20	Signature	HDR Len	Checksum	IMG Len	Offset	FlashAddr

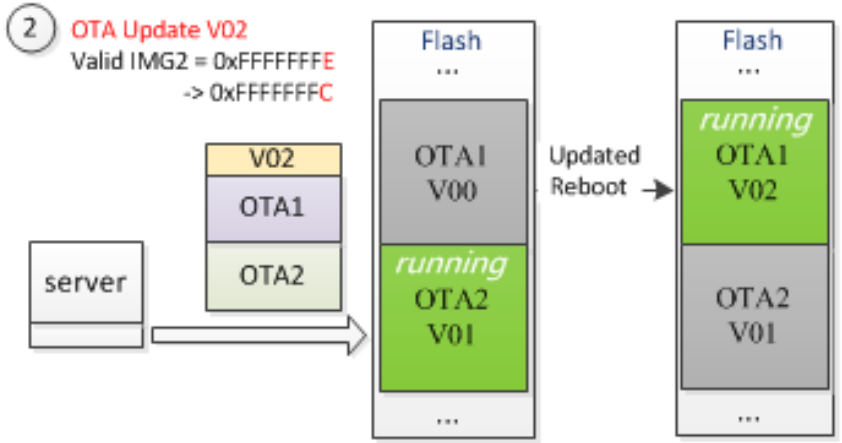
OTA Steps

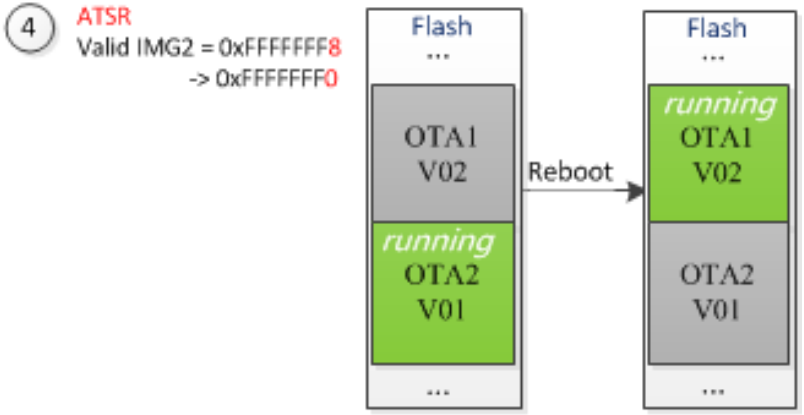
1. Received upgrade command from the cloud server.
2. Find the target OTA area in flash according to “Valid IMG2” and erase.
3. Start receiving ota_all.bin from the server.
4. Find the correct OTA Entry header in OTA header.
5. Find the correct otax.bin start offset written in OTA Entry header .
6. Save otax.bin to the target flash area.
7. Calculate checksum of otax.bin in flash and compare to the checksum in OTA Entry header. If not equal, OTA fail.
8. If target is OTA2, verify OTA2 address in 0x9000 (flash system data) is the same as the flash offset in OTA Entry header. If not equal, OTA fail.
9. Update Validate IMG2 in 0x9004 (flash system data).
10. Reboot.

OTA Example







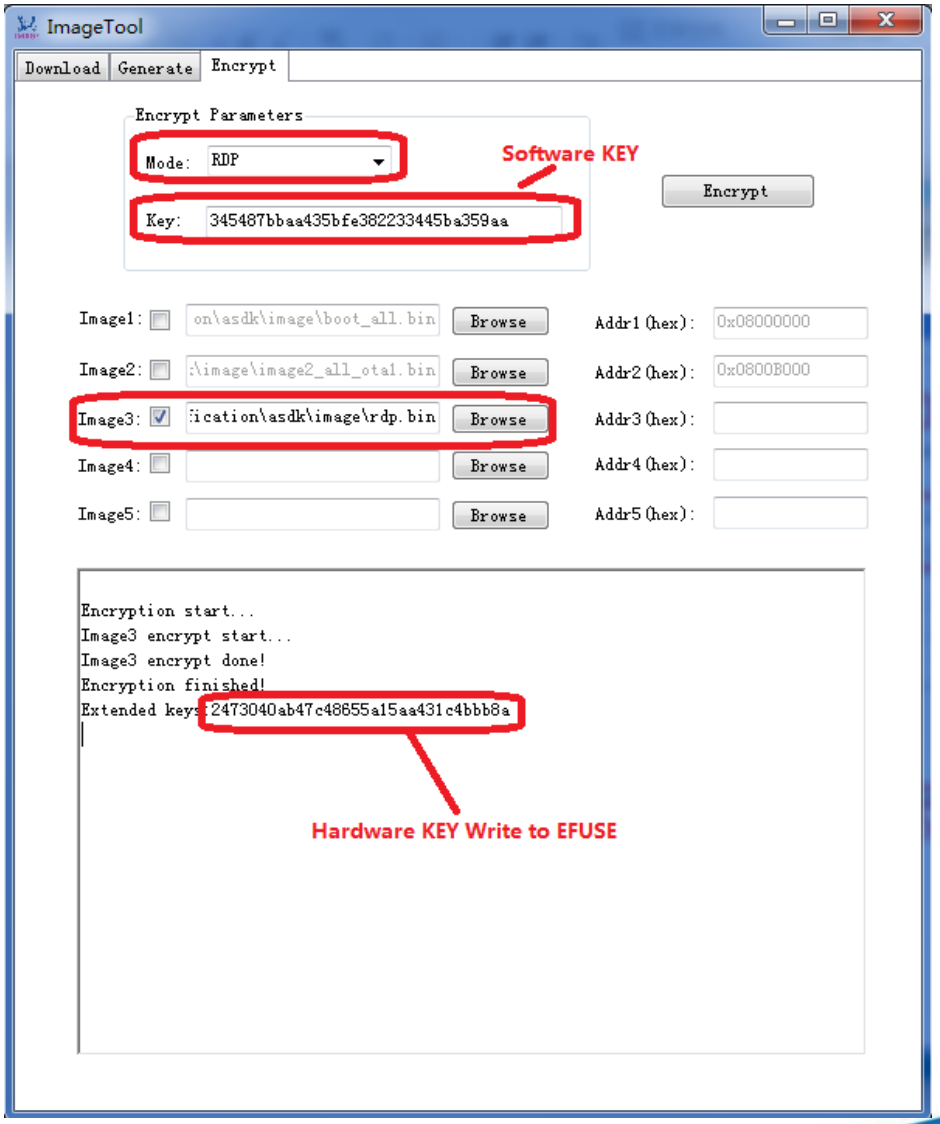


FW protection – Read Protection

- RDP
  - Top 4k RAM can not be read.
  - RDP Interrupt will happen when invalid access happen.
  - RDP image should be encrypted use RDP KEY
  - RDP image can only be decrypted and load to RDP RAM use IPSEC.
- KEY
  - 16B RDP key should be written to EFUSE RDP key area
    - Hidden EFUSE 0xB0~0xBF
  - Can not read back again. (HW protect)
  - Auto-load to IPSEC when boot.
- Enable
  - Hidden EFUSE 0xC0[0].
  - Can not be closed after open.

How to use RDP

- Prepare rdp.bin
  - Set CONFIG_ENABLE_RDP to 1 in Platform_opts.h
  - Add secret data or codes use specified section name:
    - RDP_DATA_SECTION
    - RDP_TEXT_SECTION
  - Image2.icf will auto link RDP section to 0x1003F000
  - Output file: rdp.bin
- Enable RDP protection
  - efuse_rdp_enable
- Input RDP Software key and rdp.bin to Image Tool Encrypt page, click Encrypt, then generate rdp-en.bin
- Set RDP Hardware Key to Efuse
  - efuse_rdp_keyset
- Finally write rdp-en.bin to Flash



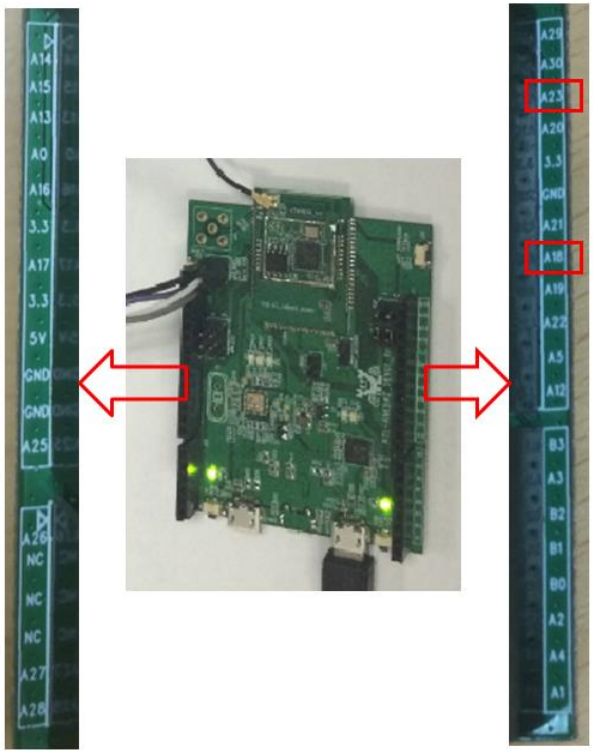
PIN Multiplex Table

QFN68	QFN48	QFN32	GPIO	UART	SPI Master	SPI Slave	SPI Flash	I2C	SDIO	PWM/TIMER	EXT32K	I2S	Others
✓	✓	✓	PA_14							PWM0	SWD_CLK		
✓	✓	✓	PA_15							PWM1	SWD_DATA		
✓			PA_13							PWM4			
✓	✓	✓	PA_0							PWM2	ext_32K		
✓	✓		PA_16	UART2_log_RXD						PWM1	RTC_OUT		
✓	✓		PA_17	UART2_log_TXD						PWM2			
✓	✓		PA_25	UART1_RXD									
✓	✓		PA_26	UART1_TXD									
✓			PA_28					I2C1_SCL					
✓			PA_27					I2C1_SDA					
✓		✓	PA_12							PWM3			
✓	✓		PA_4	UART0_TXD	SPI1_MOSI	SPI0_MOSI		I2C0_SDA					
✓	✓		PA_1	UART0_RXD	SPI1_CLK	SPI0_SCK		I2C0_SCL					
✓	✓		PA_2	UART0_CTS	SPI1_CS	SPI0_CS		I2C1_SDA					
✓	✓		PA_3	UART0_RTS	SPI1_MISO	SPI0_MISO		I2C1_SCL					
✓	✓	✓	PA_6				SPIC_CS		SD_D2				
✓	✓	✓	PA_7				SPIC_DATA1		SD_D3				
✓	✓	✓	PA_8				SPIC_DATA2		SD_CMD				
✓	✓	✓	PA_9				SPIC_DATA0		SD_CLK				
✓	✓	✓	PA_10				SPIC_CLK		SD_D0				
✓	✓	✓	PA_11				SPIC_DATA3		SD_D1				
✓	✓	✓	PA_5						SDIO_SIDEBAND_INT	PWM4			WAKEUP_1
✓	✓	✓	PA_18	UART0_RXD	SPI1_CLK	SPI0_SCK		I2C1_SCL	SD_D2	TIMER4_TRIG		I2S_MCK	WAKEUP_0
✓	✓	✓	PA_19	UART0_CTS	SPI1_CS	SPI0_CS		I2C0_SDA	SD_D3	TIMER5_TRIG		I2S_SD_TX	ADC1
✓	✓		PA_20						SD_CMD			I2S_SD_RX	ADC3
✓	✓		PA_21						SD_CLK	PWM3		I2S_CLK	
✓	✓	✓	PA_22	UART0_RTS	SPI1_MISO	SPI0_MISO		I2C0_SCL	SD_D0	PWM5		I2S_WS	WAKEUP_2
✓	✓	✓	PA_23	UART0_TXD	SPI1_MOSI	SPI0_MOSI		I2C1_SDA	SD_D1	PWM0			WAKEUP_3
✓			PB_1		SPI1_CLK	SPI0_SCK							
✓			PB_0		SPI1_CS	SPI0_CS							
✓			PB_2		SPI1_MISO	SPI0_MISO							
✓			PB_3		SPI1_MOSI	SPI0_MOSI							
✓			PB_4								SWD_CLK	I2S_MCK	
✓			PB_5								SWD_DATA	I2S_SD_TX	
✓			PA_24									I2S_SD_RX	
✓			PA_31									I2S_CLK	
✓			PB_6									I2S_WS	
✓	✓	✓	PA_30	UART2_log_TXD				I2C0_SDA		PWM3	RTC_OUT		
✓	✓	✓	PA_29	UART2_log_RXD				I2C0_SCL		PWM4			

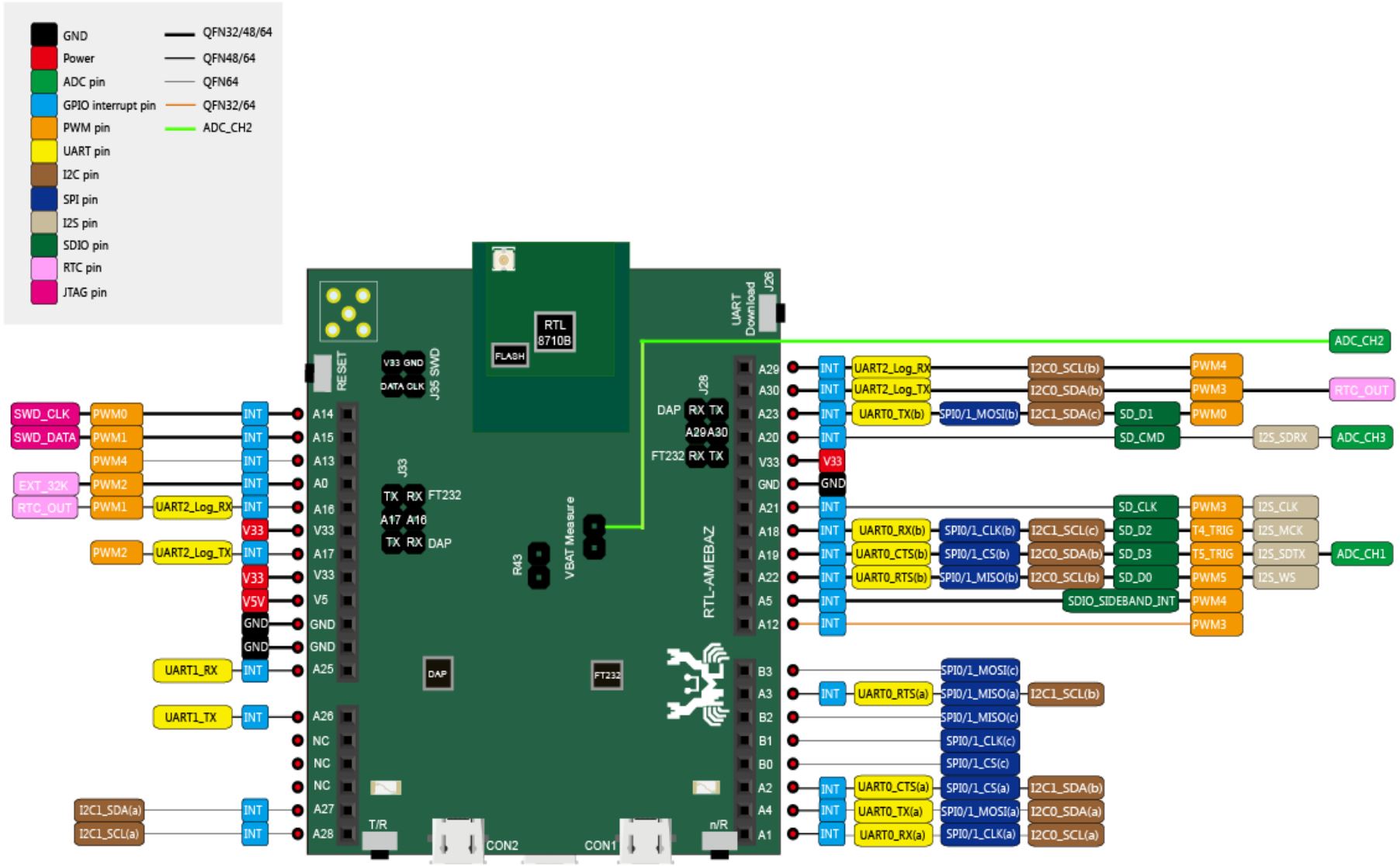
How to find UART0

- Firstly, find UART0 groups in Pin mux table defined in UM0113 Chapter 3 and shown in Blue and Red box in the lower left corner.
- Then pick the pin group which is enabled for your package. For example, PA_4/PA_1 is not available for QFN32.
- Finally, find the pin name in the right side of the dev board shown as below.

QFN68	QFN48	QFN32	GPIO	UART	SPI Master	SPI Slave
✓	✓	✓	PA_14			
✓	✓	✓	PA_15			
✓			PA_13			
✓	✓	✓	PA_0			
✓	✓		PA_16	UART2_log_RXD		
✓	✓		PA_17	UART2_log_TXD		
✓	✓		PA_25	UART1_RXD		
✓	✓		PA_26	UART1_TXD		
✓			PA_28			
✓			PA_27			
✓		✓	PA_12			
✓	✓		PA_4	UART0_TXD	SPI1_MOSI	SPI0_MOSI
✓	✓		PA_1	UART0_RXD	SPI1_CLK	SPI0_SCK
✓	✓		PA_2	UART0_CTS	SPI1_CS	SPI0_CS
✓	✓		PA_3	UART0_RTS	SPI1_MISO	SPI0_MISO
✓	✓	✓	PA_6			
✓	✓	✓	PA_7			
✓	✓	✓	PA_8			
✓	✓	✓	PA_9			
✓	✓	✓	PA_10			
✓	✓	✓	PA_11			
✓	✓	✓	PA_5			
✓	✓	✓	PA_18	UART0_RXD	SPI1_CLK	SPI0_SCK
✓	✓	✓	PA_19	UART0_CTS	SPI1_CS	SPI0_CS
✓	✓		PA_20			
✓	✓		PA_21			
✓	✓	✓	PA_22	UART0_RTS	SPI1_MISO	SPI0_MISO
✓	✓	✓	PA_23	UART0_TXD	SPI1_MOSI	SPI0_MOSI
✓			PB_1		SPI1_CLK	SPI0_SCK



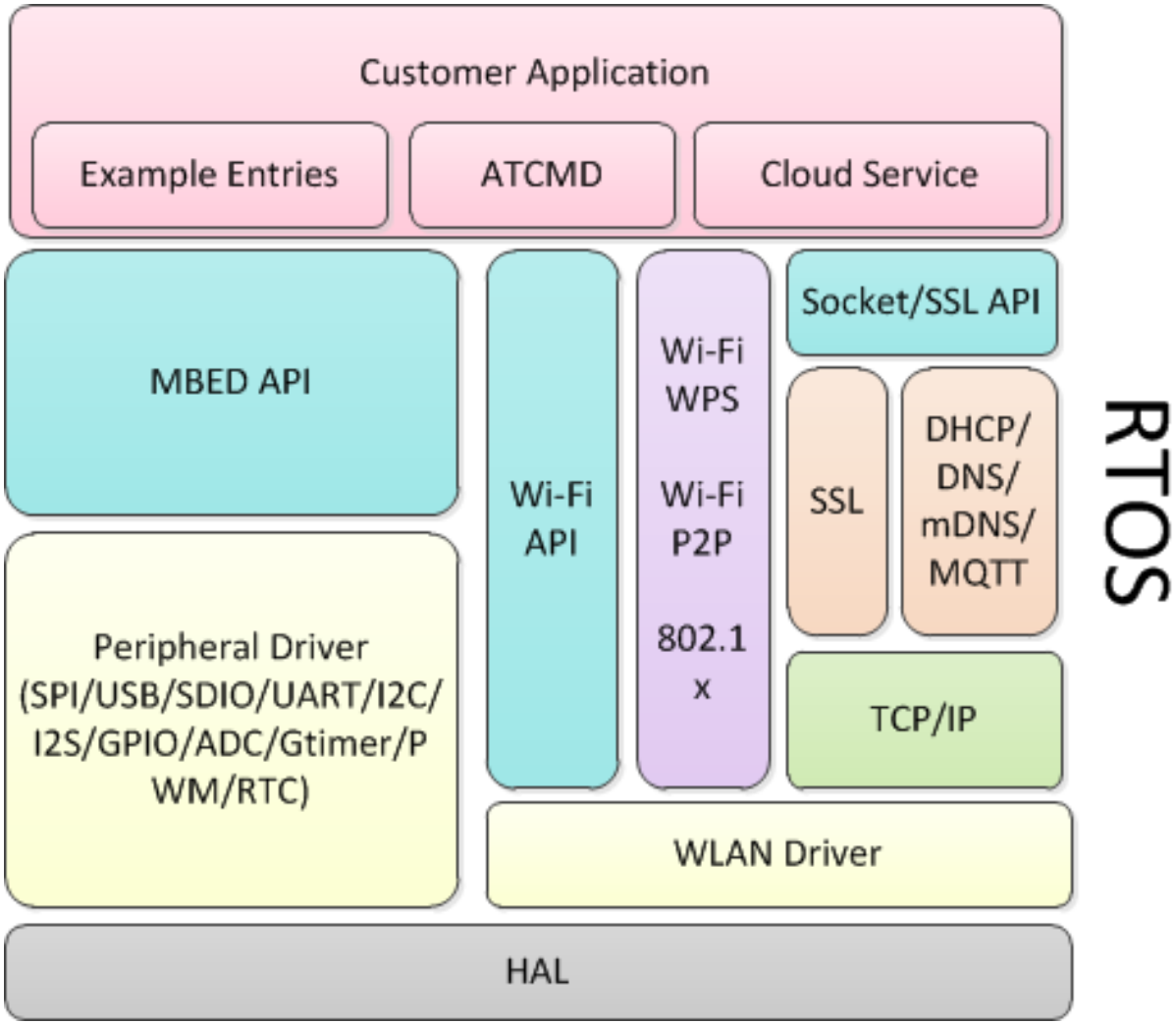
DEV Board PIN Multiplex



Content

- Introduction to Ameba-Z
- **SDK overview**
- API of Components
- IDE Tool Demo
- MP Related

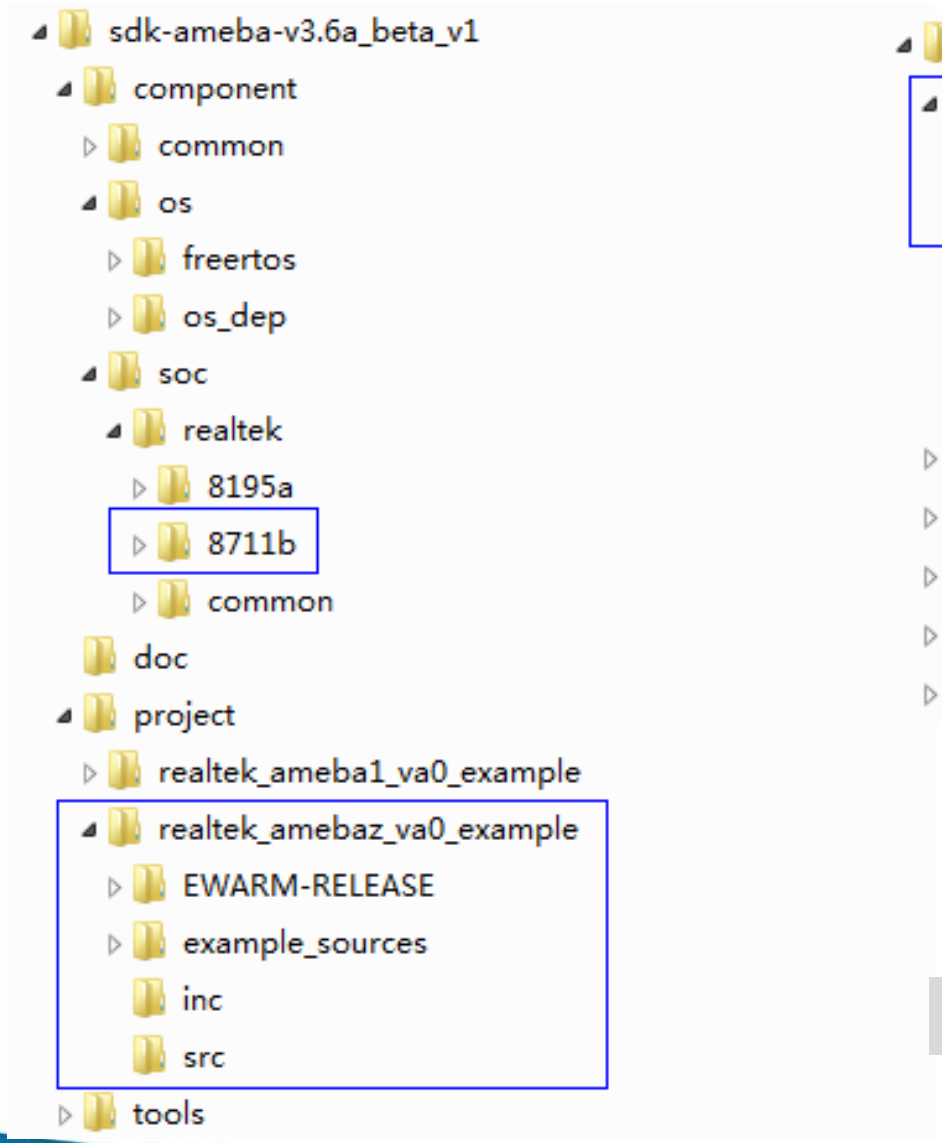
Software Stack

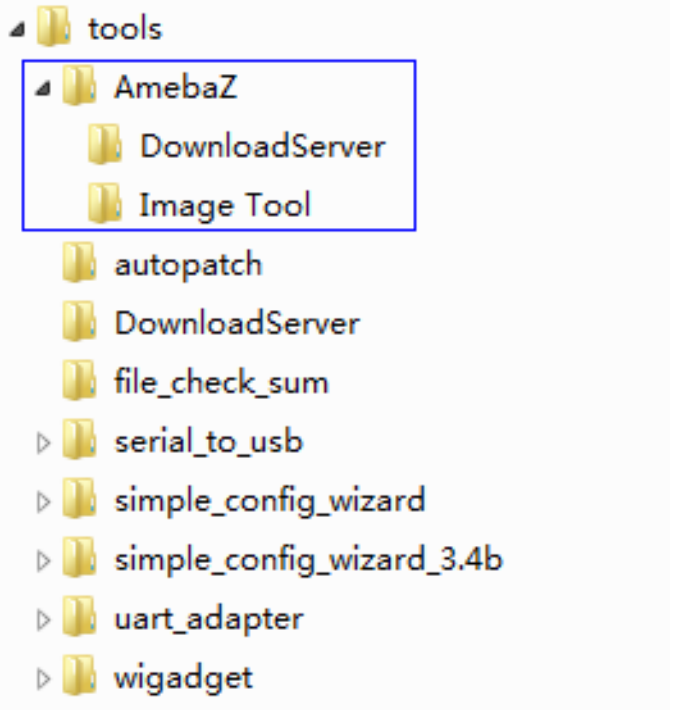


Software Features

- Operation System
  - FreeRTOS
- Network Stack
  - LW/IP
- Wlan Security
  - Open/WEP/TKIP/AES PSK
- Architecture
  - STA mode
  - AP mode
  - STA+AP mode
  - Promiscuous mode
  - P2P mode
- Device Simple Config
  - SoftAP mode config
  - WPS
  - Realtek simple config
  - Customizable Promiscuous Mode
- Secure Sockets Layer
  - Polar SSL (Ref: AN0012)
  - Mbed TLS 2.4.0
- Instant messaging protocol
  - MQTT
  - Web Socket
- Application examples
- Peripheral operation examples
- Update Firmware
  - OTA update
  - UART upgrade
- Cloud
  - Homekit (Ref: AN0035)
  - Google Nest(Ref: AN0038)
  - Gitwits (Ref: UM0062)
  - Joylink (Ref: AN0052)
  - Wechat/Airkiss (Ref: AN0054)
  - QQLink (Ref: UM0074)
  - Ali Alink (Ref: UM0098)
  - Amazon
  - Hilink (Ref: UM0095)
- Application
  - Wi-Fi RS 232 (Ref: AN0046)

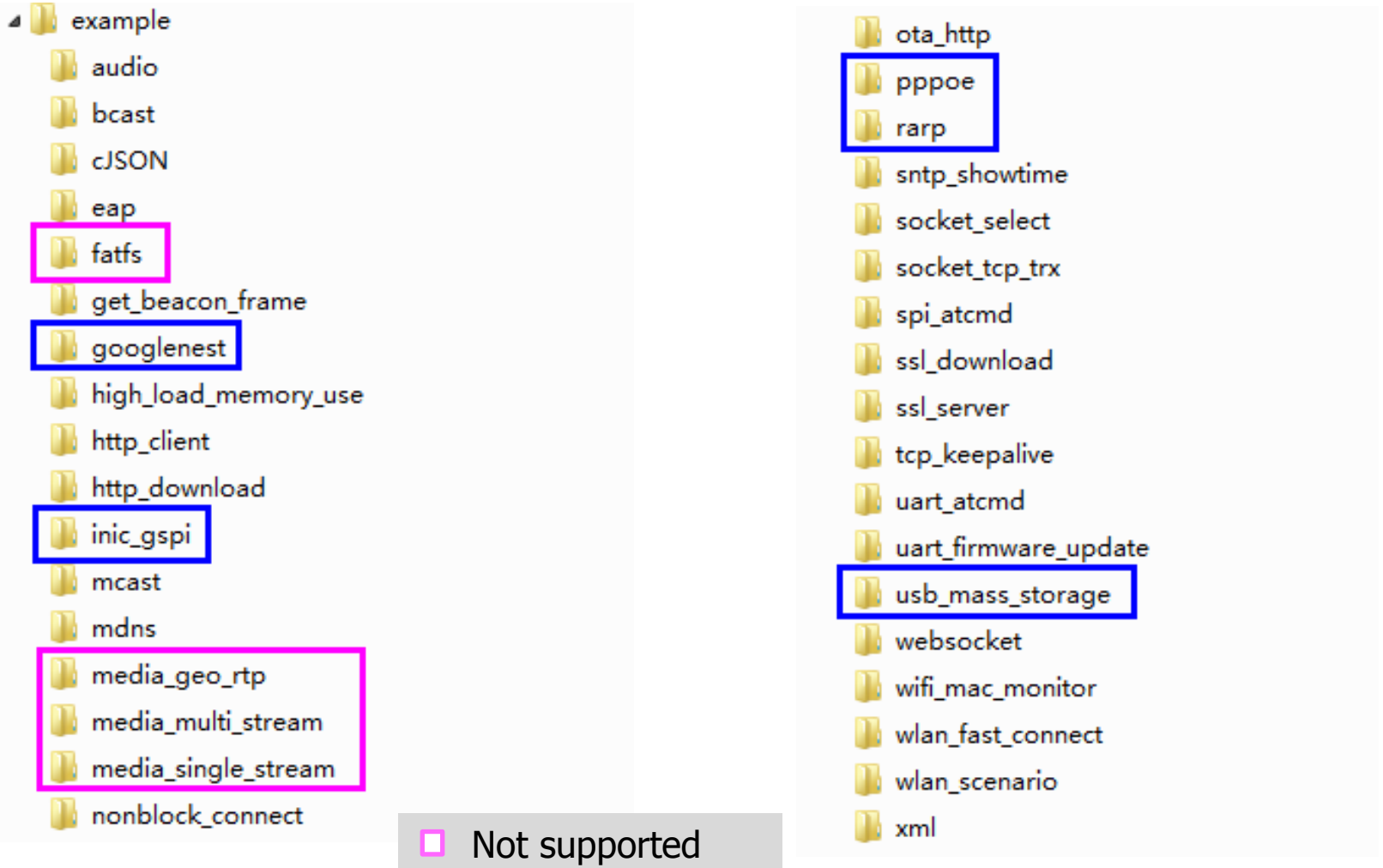
Directory Structure – Overview





New element

Directory Structure – Common Example



Directory Structure – Peripheral example

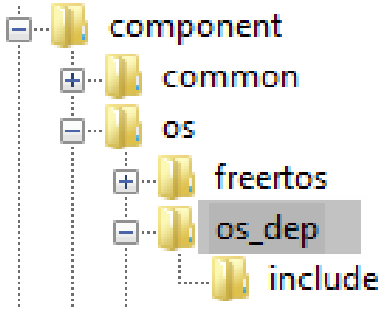
- example_sources
  - analogin
  - analogin_voltage
  - crypto
  - crypto-customize
  - efuse_mtp
  - efuse_otp
  - flash
  - flash_micron_block_protect
  - flash_setstatus
  - gdma
  - gpio
  - gpio_dht_temp_humidity
  - gpio_HC_SR04_ultrasonic
  - gpio_irq
  - gpio_jtag
  - gpio_level_irq
  - gpio_light_weight
  - gpio_port
  - gpio_pulse_measure
  - gspi
  - gspi_fw_downloader
  - gspi_fw_loader
  - gspi_mp_downloader
  - gtimer
  - i2c
  - i2c_dual
  - i2c_epl2197_heartrate
  - i2c_epl2590_light
  - i2c_epl2590_proximity
  - i2c_LPS25HB_pressure
  - i2c-shtc1
  - i2s
  - i2s_bypass
  - i2s_tx_and_rx_only
  - log_uart_char_loopback
  - log_uart_loopback
  - log_uart_stream_loopback
  - pm_deepsleep
  - pm_deepstandby
  - pm_sleep
  - pm_tickless
  - pwm
  - pwm-buzzer
  - rtc
  - sdio_device
  - spi
  - spi_gpio_chipselect
  - spi_master_write_read_one_byte
  - spi_multislave
  - spi_pl7223
  - spi_stream_twoboard
  - spi_stream_twoboard_concurrent
  - spi_twoboard
  - uart
  - uart_auto_flow_ctrl
  - uart_clock
  - uart_irq
  - uart_stream_2_threads
  - uart_stream_4_threads
  - uart_stream_dma
  - uart_stream_irq
  - uart_stream_rx_timeout
  - uart_stream_rx_timeout_by_GTimer
  - uart_stream_rx_timeout_by_semaphore_iar
  - uart_stream_rx_timeout_by_SoftTimer
  - uart_stream_rx_timeout_by_WaitSemaphore
  - uart_stream_tx_rx_concurrent_iar
  - usb_uvc
  - watchdog
  - wlan
  - wlan_inic

Freertos

- Minimal RAM and processing overhead
  - Typically an RTOS kernel binary image will be in the region of 4K to 9K bytes.
  - The core of the FreeRTOS kernel is contained in only 4 C files.
    - Tasks.c
    - Queue.c
    - Heap_5.c
    - Timer.c
- Real time, reliable and scalable

OS API

- Usage
  - To use OS related API, the osdep_service.h in folder $sdk\component\os\osdep\include provides all APIs that may be used.



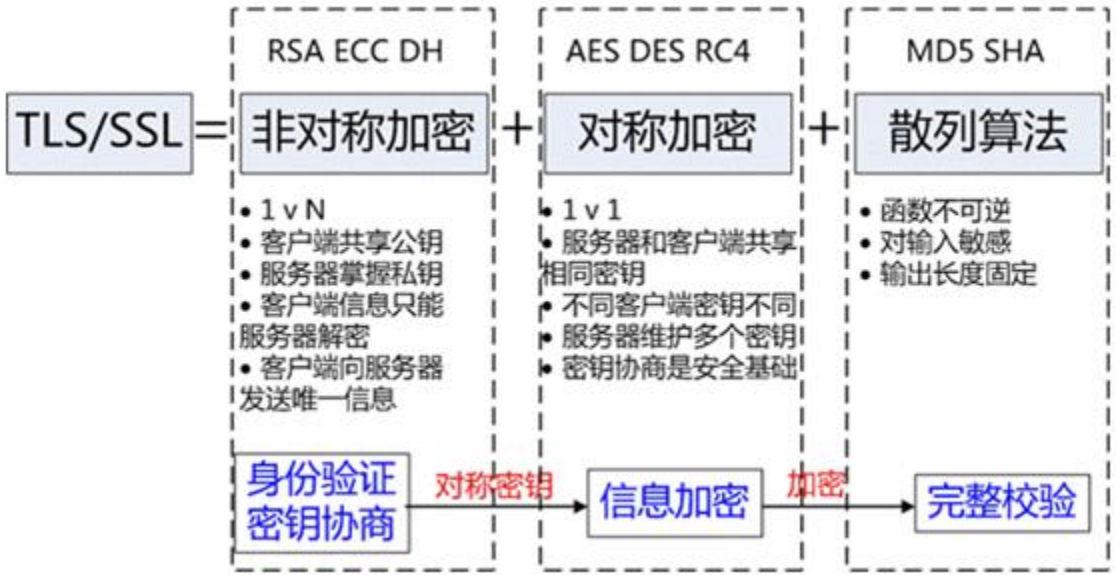
- Reference
  - The details of each function are declared in osdep_service.h

LWIP

- Lightweight and open source TCP/IP stack
- Provide basic features of TCP Protocol with decreased system occupation
- Fit for small embedded applications , requires only 20K RAM and 40K ROM
- Support protocols
  - IP protocol
  - ARP protocol
  - ICMP protocol
  - UDP protocol
  - TCP protocol including Congestion Control, RTT Estimation and Fast Recovery/Fast Retransmit

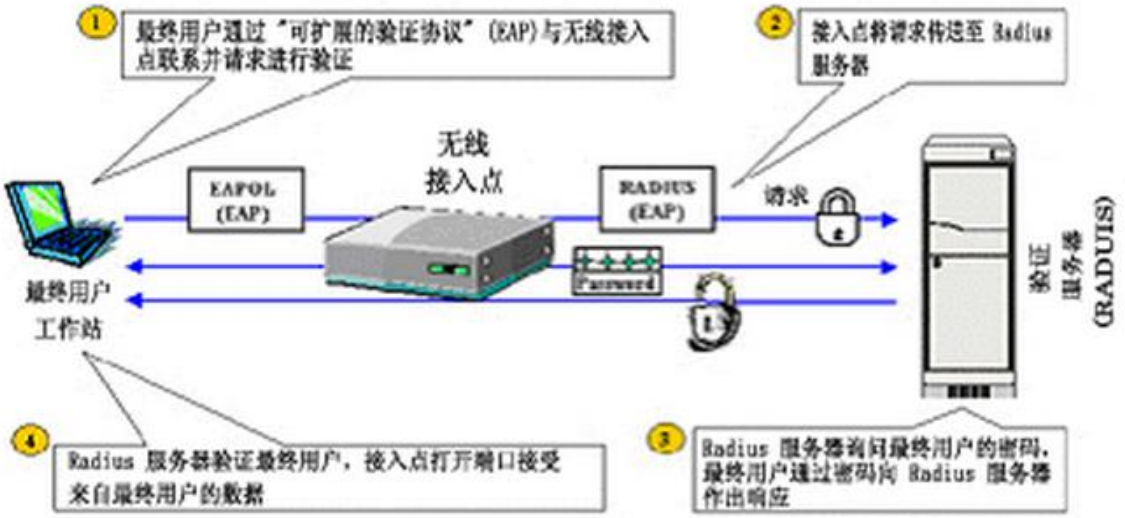
Security - SSL

- Three kinds of basic algorithms:
  - Asymmetric encryption algorithm
    - Identity Authentication
    - Key Negotiation
  - Symmetric encryption algorithms
    - Encrypted data using the negotiated key
  - Hash Functions
    - Verify the integrity of information

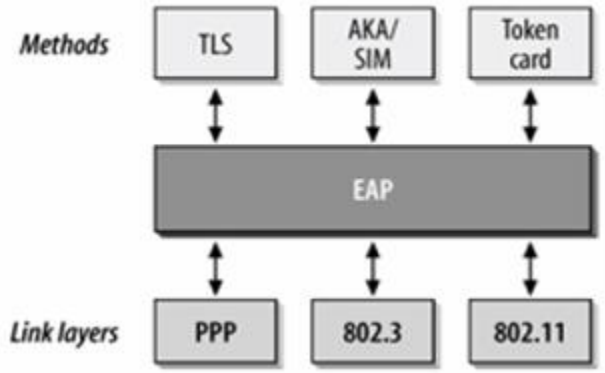


Security – 802.1x EAP

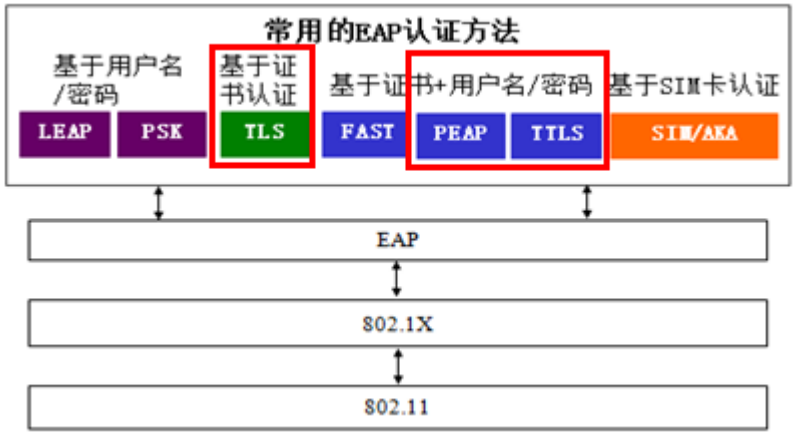
- 802.1x Authentication
  - Requester
  - Verifier
  - Authentication Server



- Extensible Authentication Protocol (EAP)

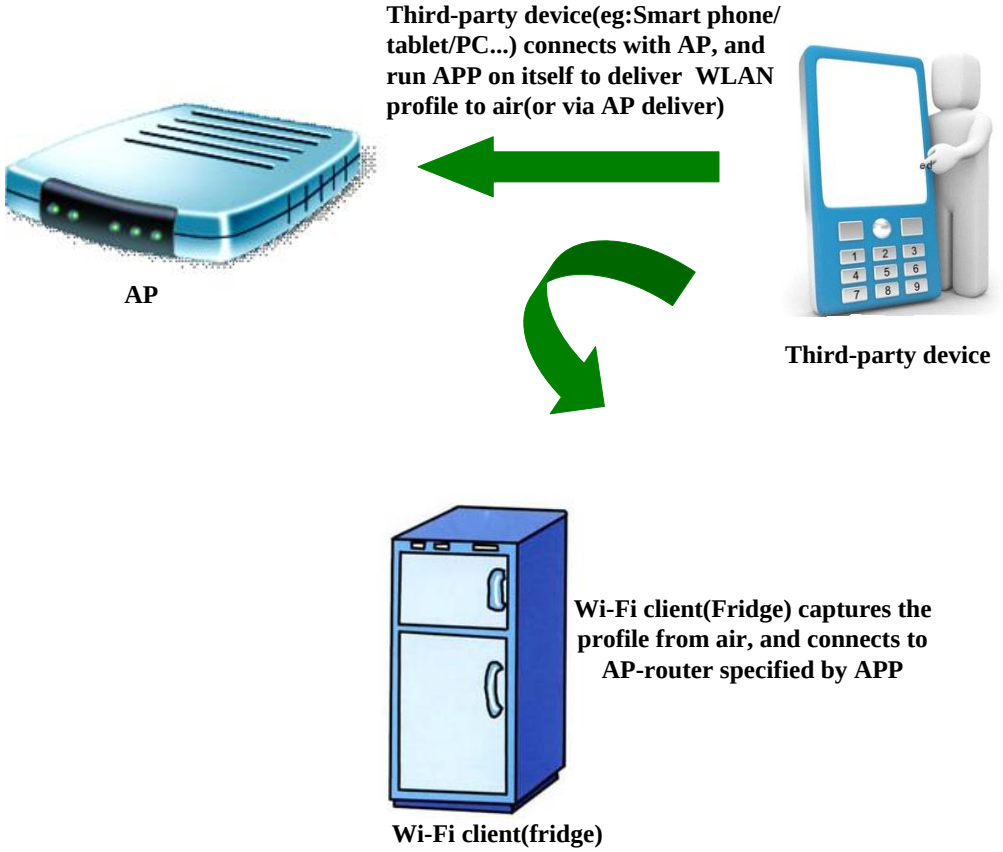


- EAP Method
  - Support PEAP, TLS and TTLS



Simple Config R3 (Ref: AN0011)

- SoftAP + Promiscuous mode
- As fast as old version
- High success rate
- Cover both Android and iOS
- Cost Flash 14K and SRAM 0.5K



Cloud Service

Cloud Service	Code	Data + BSS	Status
Homekit	-	-	Ready
Google	-	-	Ready
Amazon	-	-	Ready
Alink 1.1	60K+50K	68K	Ready
Joylink 1.3.3	61K	36K	Ready
QQlink 1.1.101	70K	44K	Ready
Hilink 0.5.4	73K	17K	Ready
Gagent + Airkiss	77K	22K	Ready
Wechat 3.1.0	89K	35K	Ready

Content

- Introduction to Ameba-Z
- SDK overview
- **API of Components**
- IDE Tool Demo
- MP Related

AT Command for Wi-Fi (Ref: AN0025)

- ‘AT??’ Print Log History
- ‘AT--’ Exit Log Service
- ‘ATW0’ Wlan Set Network SSID
- ‘ATW1’ Wlan set Network Passphrase
- ‘ATW2’ Wlan Set Key ID
- ‘ATWC’ Wlan Join a Network
- ‘ATWD’ Wlan Disconnect from Network
- ‘ATW3’ Wlan Set Access Point SSID
- ‘ATW4’ Wlan Set Access Point Security Key
- ‘ATW5’ Wlan Set Access Point Channel
- ‘ATWA’ Wlan Activate Access Point
- ‘ATWB’ Wlan Activate Access Point mode and Station mode

$sdk\component\common\api\at_cmd\atcmd_wifi.c

- ‘ATW?’ Wlan Show WiFi information
- ‘ATWS’ Wlan Scan for Network Access Point
- ‘ATWR’ Wlan Get RSSI of Associated Network Access Point
- ‘ATWM’ Wlan Wi-Fi promisc
- ‘ATWE’ Wlan Start Web Server
- ‘ATWQ’ Wlan Wi-Fi Simple Config
- ‘ATWP’ Wlan Power on/off wifi module
- ‘ATWI’ Wlan ping test
- ‘ATWO’ Wlan OTA update
- ‘ATWT’ Wlan TCP throughput test
- ‘ATWU’ Wlan UDP test
- ‘ATWL’ Wlan SSL client
- ‘ATWW’ Wlan Wi-Fi Protected Setup
- ‘ATWZ’ Wlan IWPRIV

AT Command for system

- ‘ATSD’ Dump register
- ‘ATSE’ Edit register
- ‘ATSC’ Clear OTA signature
- ‘ATSR’ Recover OTA signature
- ‘ATSA’ MP ADC test
- ‘ATSG’ MP GPIO test
- ‘ATSP’ MP Power related test
- ‘ATSB’ OUT PIN setup
- ‘ATSS’ Show CPU stats
- ‘ATSM’ Apple CP test
- ‘ATSJ’ Turn off JTAG
- ‘ATS@’ Debug message setting
- ‘ATS!’ Debug configure setting
- ‘ATS#’ Test command
- ‘ATS?’ Help

WiFi common API (Ref:UM0006)

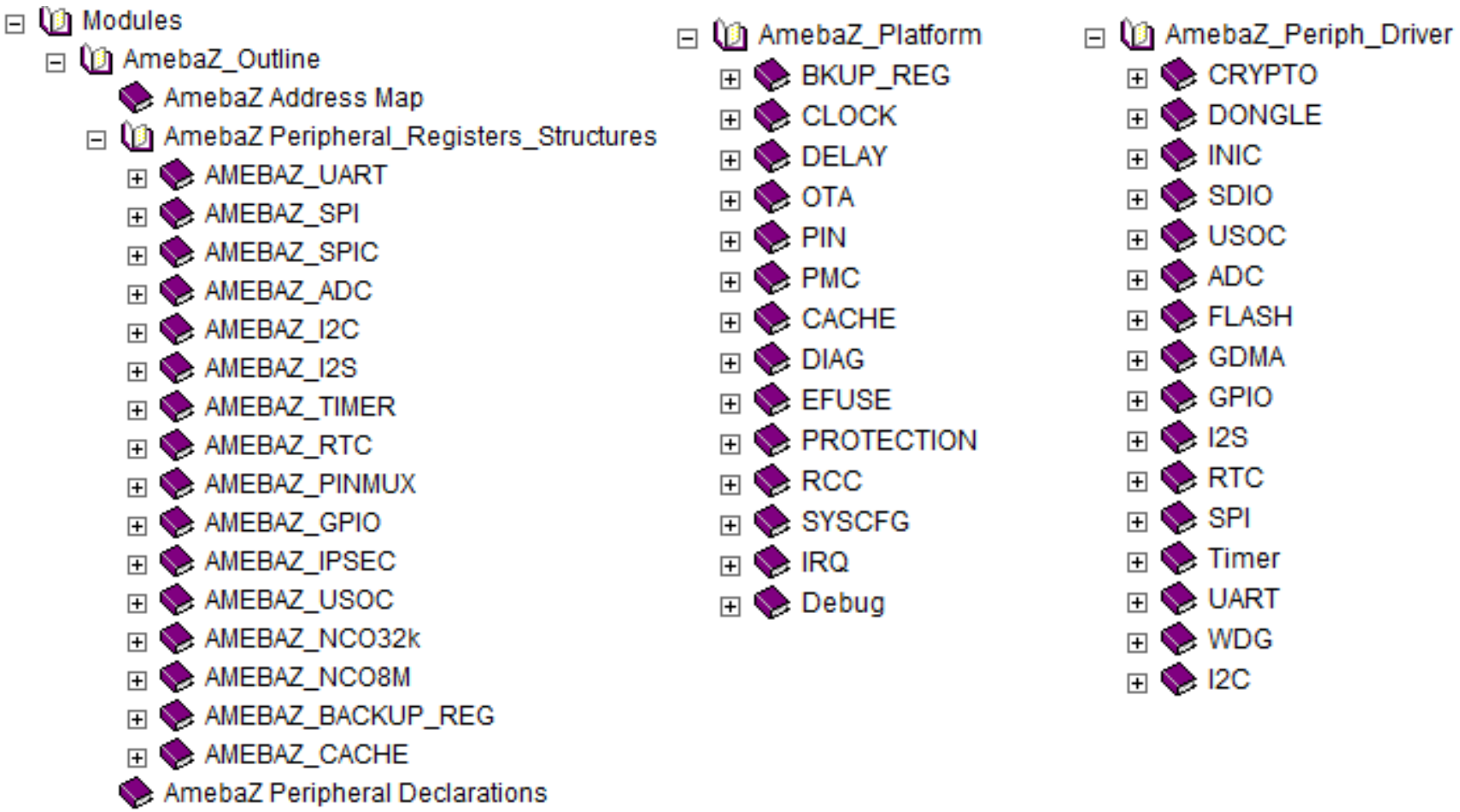
- Wifi enable/disable
  - wifi_on
  - wifi_off
  - wifi_is_up
  - wifi_is_ready_to_transceive
- Station Mode Connection
  - wifi_connect
  - wifi_disconnect
- AP Mode Startup
  - wifi_start_ap
  - wifi_restart_ap
  - wifi_get_ap_info
  - wifi_get_associated_client_list
- AP+STA Concurrent Mode
  - wifi_start_ap
  - wifi_connect
- Wifi Scan
  - wifi_scan_networks
  - wifi_set_pscan_chan
- Wlan Driver Indication
  - wifi_indication
- Wifi Promiscuous Mode
  - wifi_enter_promisc_mode
  - wifi_set_promisc
  - wifi_init_packet_filter
  - wifi_add_packet_filter
  - wifi_enable_packet_filter
  - wifi_disable_packet_filter
  - wifi_remove_packet_filter
- Wifi Setting Information
  - wifi_get_setting
  - wifi_show_setting

WiFi common API

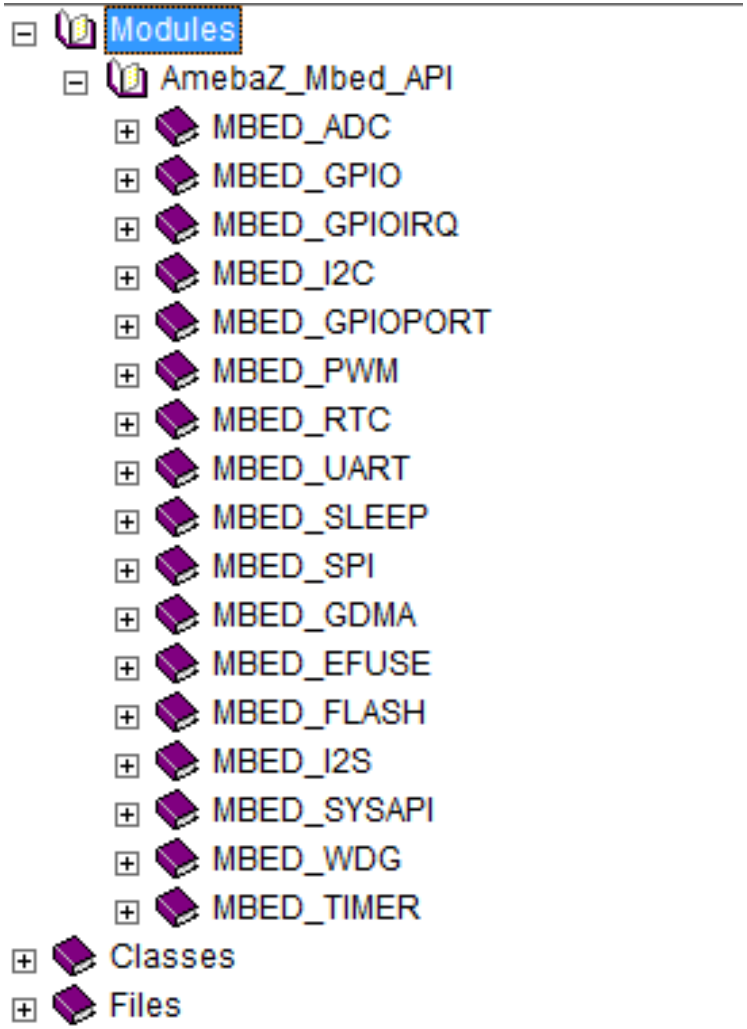
- Wifi Mac Address
  - wifi_set_mac_address
  - wifi_get_mac_address
- Wifi Power save
  - wifi_enable_powersave
  - wifi_disable_powersave
- Wifi Tx Power
  - wifi_set_txpower
  - wifi_get_txpower
- Wifi Channel
  - wifi_set_channel
  - wifi_get_channel
- Wifi Multicast Address
  - wifi_register_multicast_address
  - wifi_register_multicast_address
- Wifi RF Control
  - wifi_rf_on
  - wifi_rf_off
- Wifi Auto Reconnection
  - wifi_set_autoreconnect
  - wifi_get_autoreconnect
- Wifi Custom IE
  - wifi_add_custom_ie
  - wifi_update_custom_ie
  - wifi_del_custom_ie
- Wifi RSSI Information
  - wifi_get_rssi
- Country Code Setup
  - wifi_set_country
- Network Mode Setup
  - wifi_set_network_mode

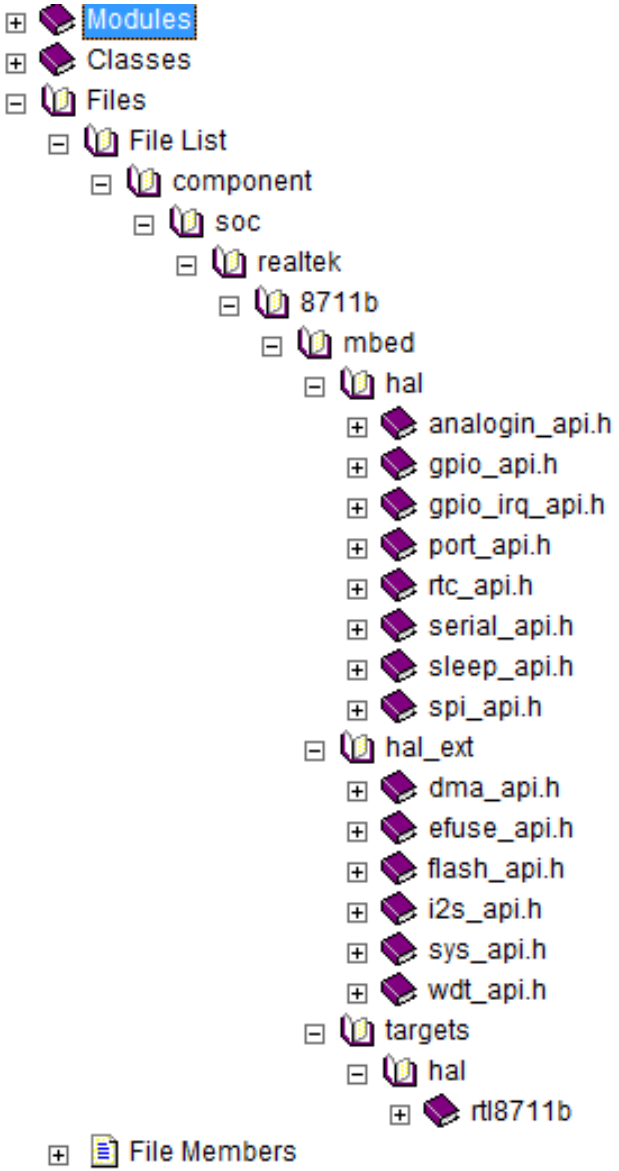
$sdk\component\common\api\wifi\wifi_conf.c

Peripheral API (Ref: UM0117)



Mbed API (Ref: UM0118)





LWIP API

- Socket
- Shutdown
- Bind
- Listen
- Accept
- Connect
- Recv
- Recvfrom
- Send
- Sendto
- Select
- Ioctlsocket
- Read
- Write
- Close

$sdk\component\common\network\lwip\lwip_v1.4.1\src\api\sockets.c

- tcp_new
- tcp_accept
- tcp_recv
- tcp_sent
- tcp_poll
- tcp_recved
- tcp_bind
- tcp_connect
- tcp_listen
- tcp_abort
- tcp_close
- tcp_write
- udp_new
- udp_remove
- udp_bind
- udp_connect
- udp_recv
- udp_send

Freertos API

- Task
  - rtw_create_task
  - rtw_delete_task
  - rtw_wakeup_task
  - rtw_get_scheduler_state
- Queue
  - rtw_init_xqueue
  - rtw_push_to_xqueue
  - rtw_pop_from_xqueue
  - rtw_deinit_xqueue
- Semaphore
  - rtw_init_sema
  - rtw_free_sema
  - rtw_up_sema
  - rtw_up_sema_from_isr
  - rtw_down_timeout_sema
- Delay
  - rtw_mdelay_os
- Timer
  - rtw_timerCreate
  - rtw_timerDelete
  - rtw_timerIsTimerActive
  - rtw_timerStop
  - rtw_timerChangePeriod
- Memory
  - rtw_malloc
  - rtw_zmalloc
  - rtw_mfree
  - rtw_getFreeHeapSize
- Wake Lock
  - rtw_acquire_wakelock
  - rtw_release_wakelock
  - rtw_wakelock_timeout
- Time
  - rtw_get_current_time
- Random
  - rtw_get_random_bytes

$sdk\component\os\os_dep\osdep_service.c

Power Saving Related API

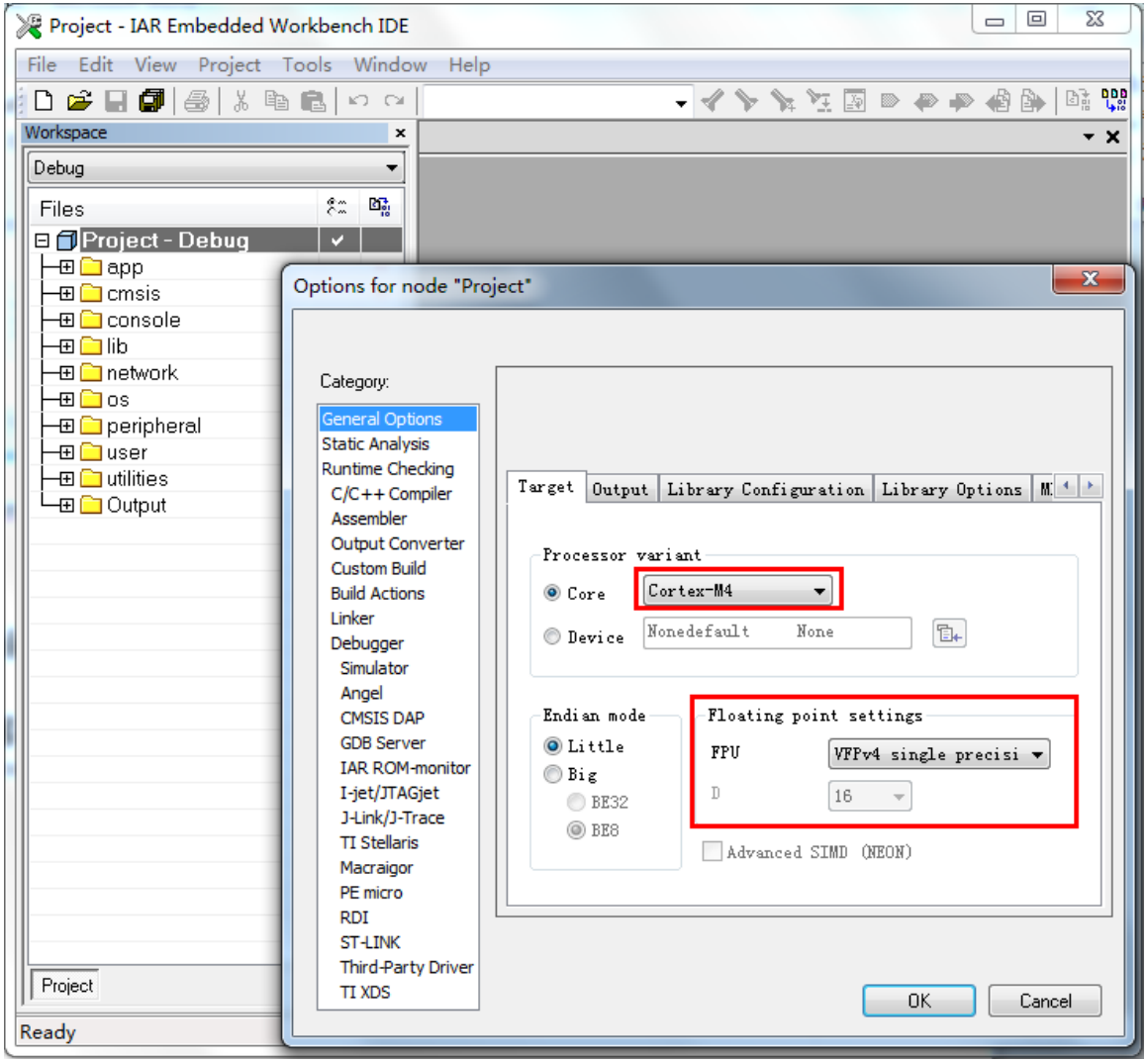
- CM4 power mode
  - sleep_ex
  - deepstandby_ex
  - deepsleep_ex
- Wi-Fi Power Save
  - wifi_enable_powersave
  - wifi_disable_powersave
  - wifi_set_power_mode
  - wifi_set_lps_dtim
  - wifi_get_lps_dtim
- Wakeup Event
  - DSLEEP_WAKEUP_BY_TIMER/GPIO
  - STANDBY_WAKEUP_BY_STIMER/GPIO/RTC
  - SLEEP_WAKEUP_BY_STIMER/GTIMER/GPIO_INT/WLANSDIO/USB/GPIO/UART/I2C/RTC
- PM tickles
  - Suspend and Resume callback
    - pmu_register_sleep_callback
    - pmu_unregister_sleep_callback
  - Wake Lock
    - pmu_acquire_wakelock
    - pmu_release_wakelock
    - pmu_get_wakelock_status
  - Wakeup Event
    - add_wakeup_event
    - del_wakeup_event
  - PMU_DEVICE: nDeviceId
    - PMU_OS/USER_BASE/WLAN/LOGUART/CONSOL/SDIO/UART0/UART1/RTC/I2C0/I2C1/ADC/USOC/DONGLE

Content

- Introduction to Ameba-Z
- SDK overview
- API of Components
- **IDE Tool Demo**
- MP Related

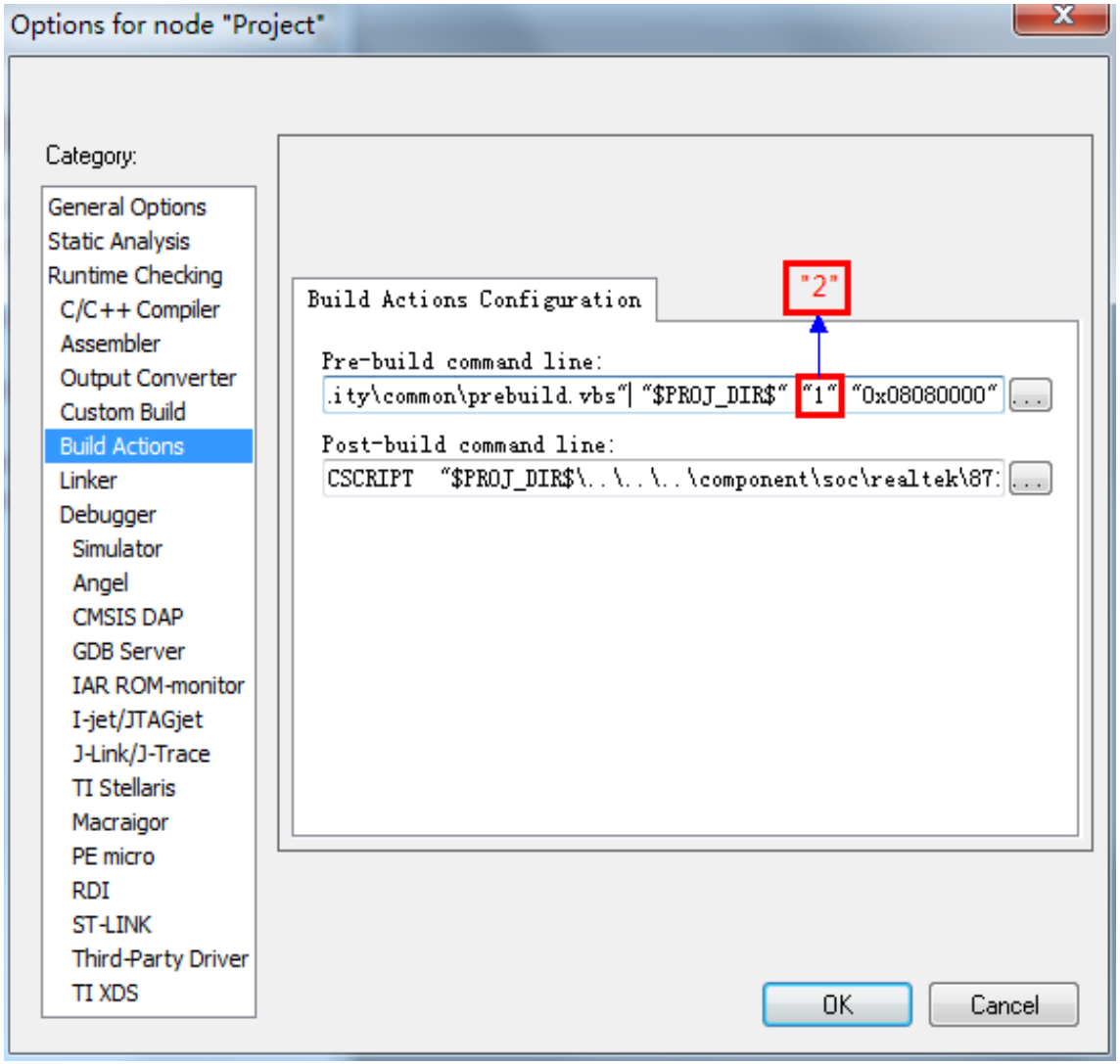
IAR build environment (Ref: UM0110)

- IAR build code
  - Open released IAR workspace
  - Choose Cortex-M4F or Cortex-M4 at least
  - Click Project -> **Rebuild All**
- image2.icf
  - Determines the location of sections on the Flash or SRAM
  - Include external symbol of functions in ROM and Boot loader



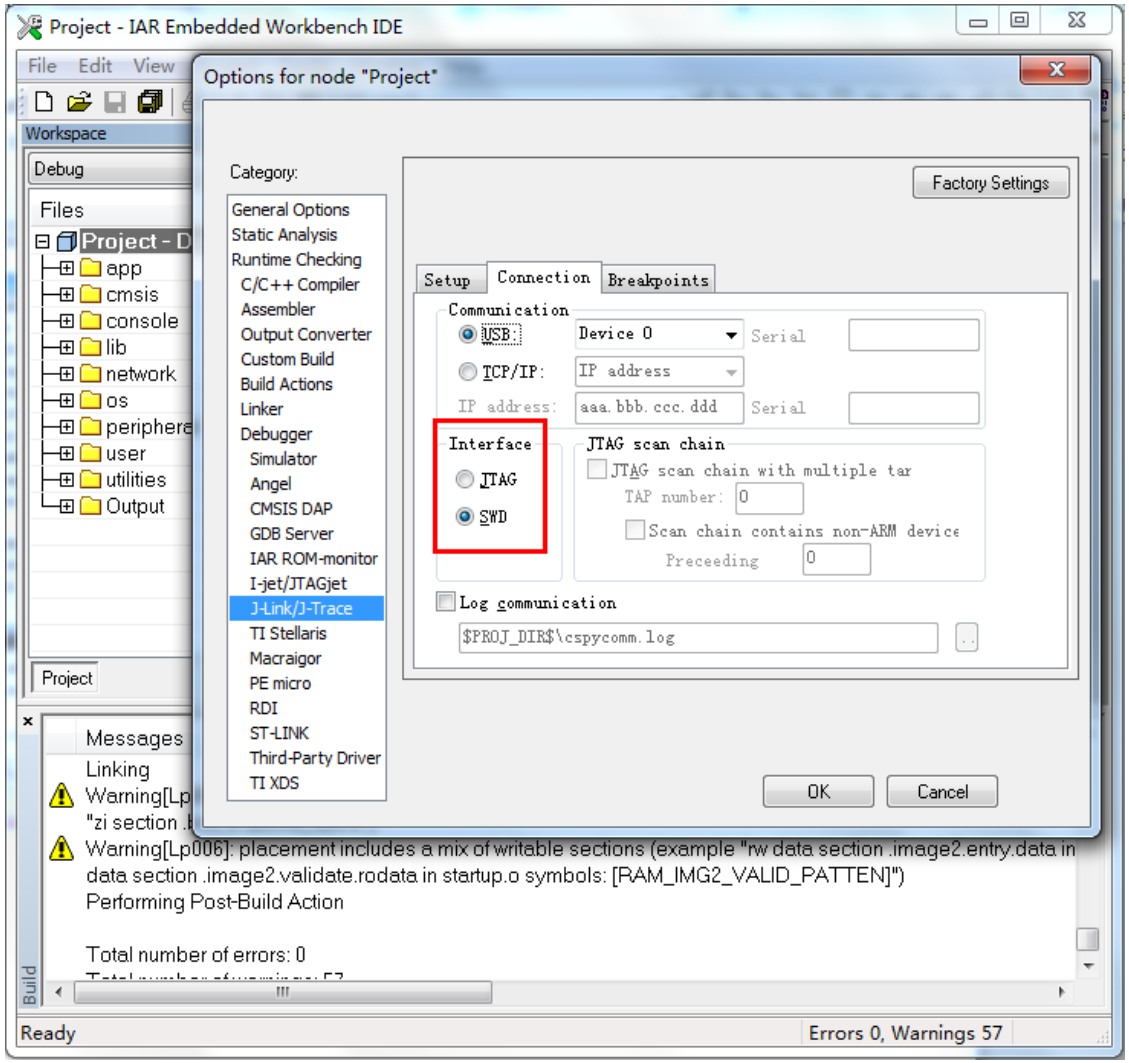
IAR build

- OTA2 binary
  - Change the secondary input parameter of prebuild VB script from “1” to “2” to build **image2_all_ota2.bin** and generate **ota_all.bin**
  - **Please Rebuild All**
- OTA2 flash offset
  - The third input parameter of prebuild VB script is OTA2 flash offset, default set to **0x08080000**



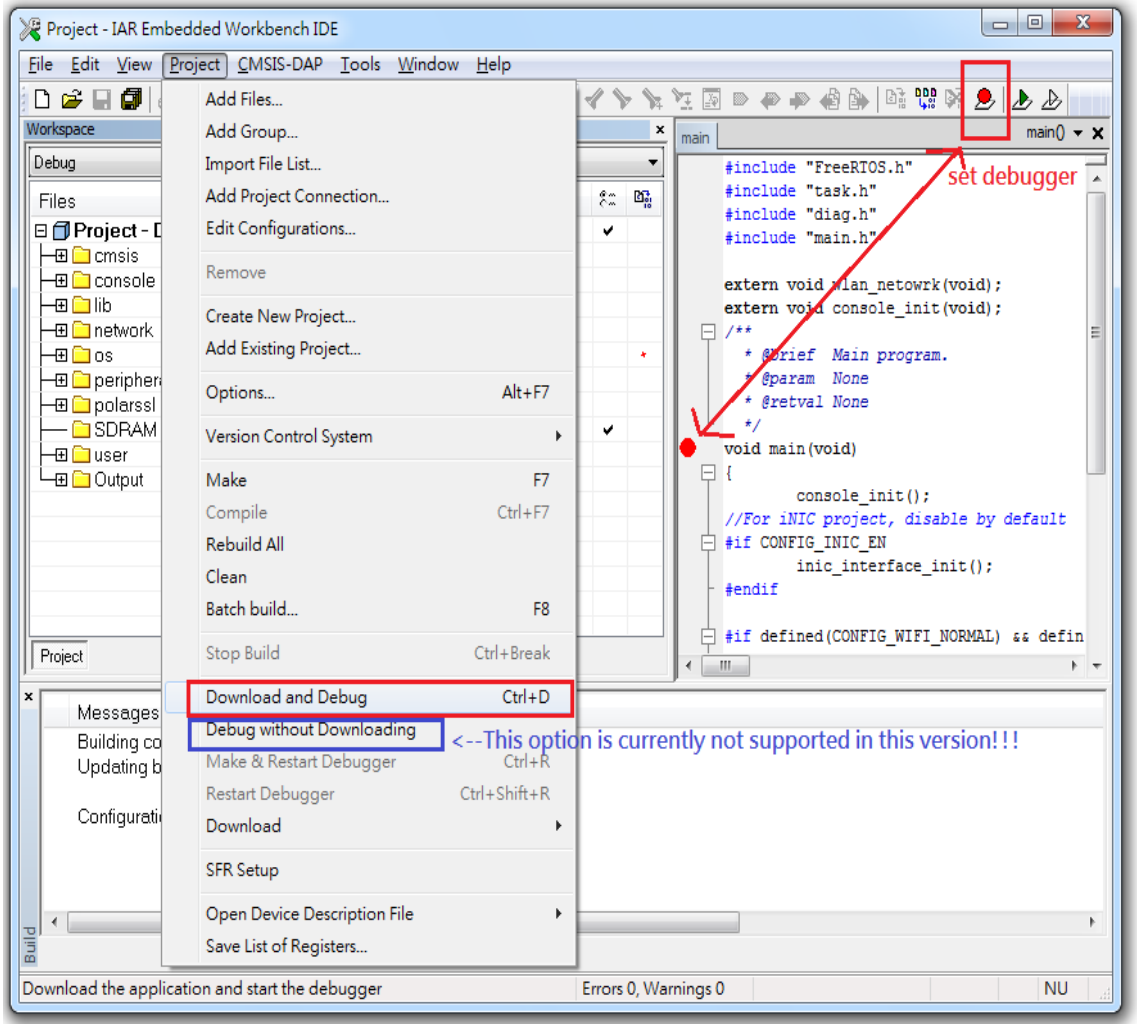
IAR Download

- Debugger
  - J-Link/J Trace
- J-Link Interface
  - SWD
- .board file
  - Decide what to write in output file and write to what position in Flash
  - tmp.board
  - Dynamically generated by postbuild VB script



IAR Debug

- Click Project->Download and Debug
- Set breakpoints
- View
  - Registers
  - Watch
  - Stack
  - Memory



Trouble shooting

- Project build fail
  - Check Core is set to CM4F or CM4.
  - Check if Flash and SRAM is enough.
- Download fail
  - Check SWD physical connection
  - Check the interface of J-Link settings is SWD
- UART log fail
  - Check jumper of J28
  - Check baud rate
- WLAN connect fail
  - Check log for connection status
  - Check security correctness
  - Check sniffer log

Trouble shooting – Hardfault

1. Record register PC and LR
2. Open application.asm and find out PC and LR in which function
3. The system crash caused by the instructions near PC
4. Check the PC's function and LR's function
5. If PC is not a valid memory address, may be memory overflow or catch flash instructions error
6. If PC is pointed to ROM or boot loader address, check the .icf symbol file to find the function.

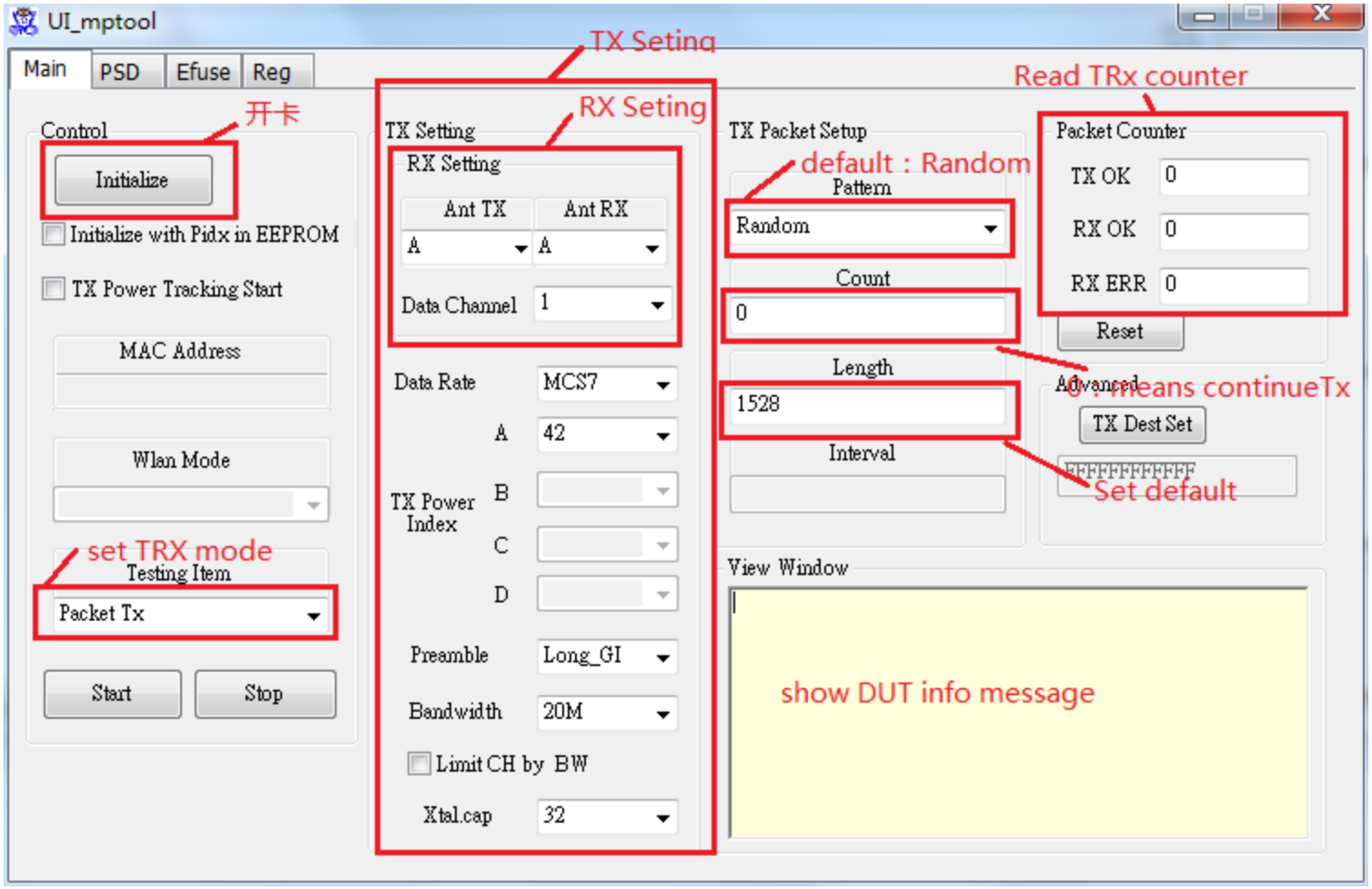
```
RTL8195A[HAL]: R0 = 0x0
RTL8195A[HAL]: R1 = 0x1005a37a
RTL8195A[HAL]: R2 = 0x6
RTL8195A[HAL]: R3 = 0x0
RTL8195A[HAL]: R12 = 0x1005f4a5
RTL8195A[HAL]: LR = 0x1004d0bf
RTL8195A[HAL]: PC = 0x1001ce00
RTL8195A[HAL]: PSR = 0x60000000
RTL8195A[HAL]: BFAR = 0x8
RTL8195A[HAL]: CFSR = 0x20000
RTL8195A[HAL]: HFSR = 0x40000000
RTL8195A[HAL]: DFSR = 0x0
RTL8195A[HAL]: AFSR = 0x0
RTL8195A[HAL]: PriMask 0x0
RTL8195A[HAL]: BasePri 0x0
RTL8195A[HAL]: SVC priority: 0x00
RTL8195A[HAL]: PendSVC priority: 0xf0
```

Content

- Introduction to Ameba-Z
- SDK overview
- API of Components
- IDE Tool Demo
- **MP Related**

MP tool – RF calibration (Ref: UM0119)

- Realtek Ameba MP Flow (Ref: UM0059)



MP tool – EFUSE (Ref: UM0119)

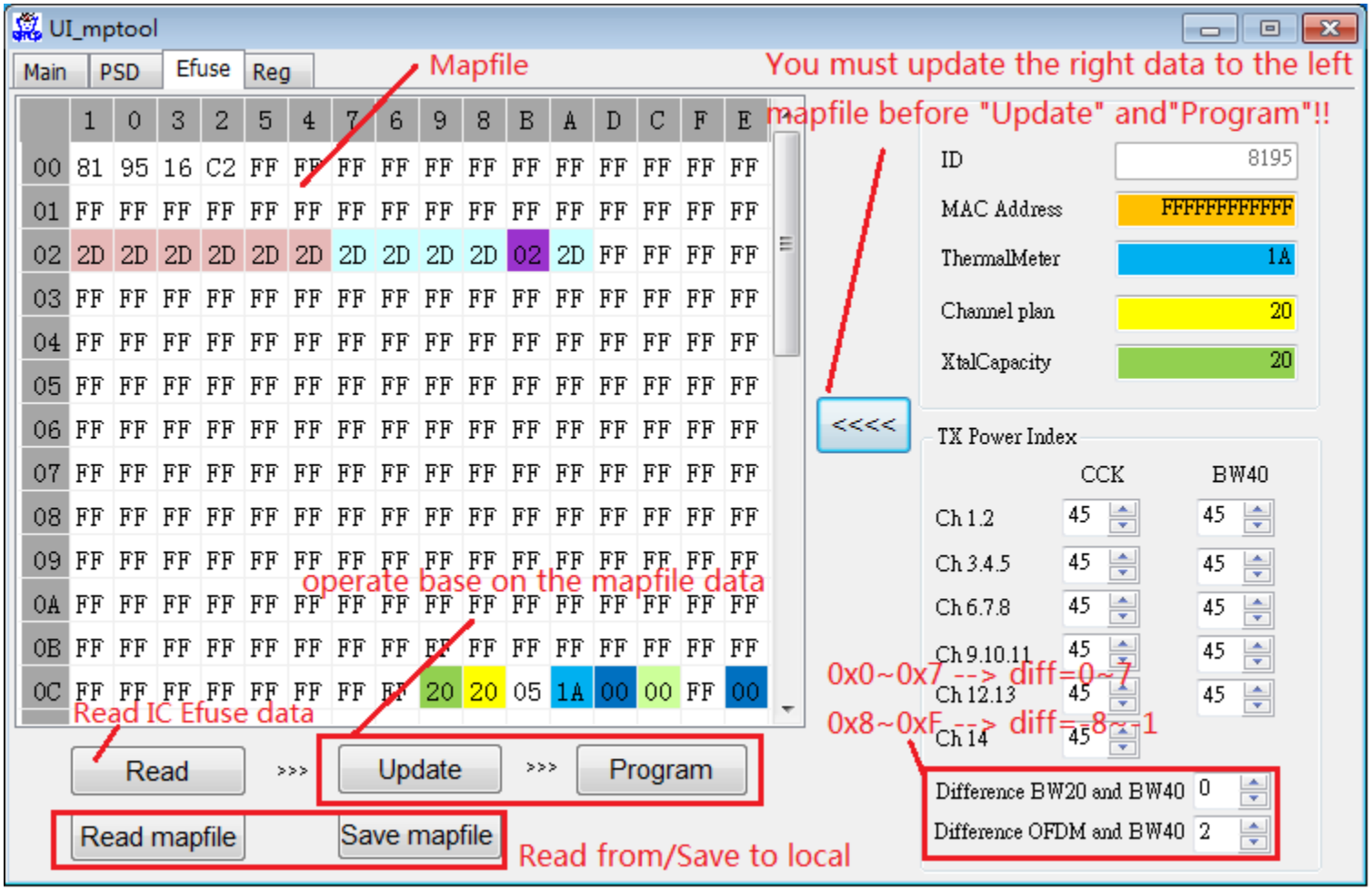
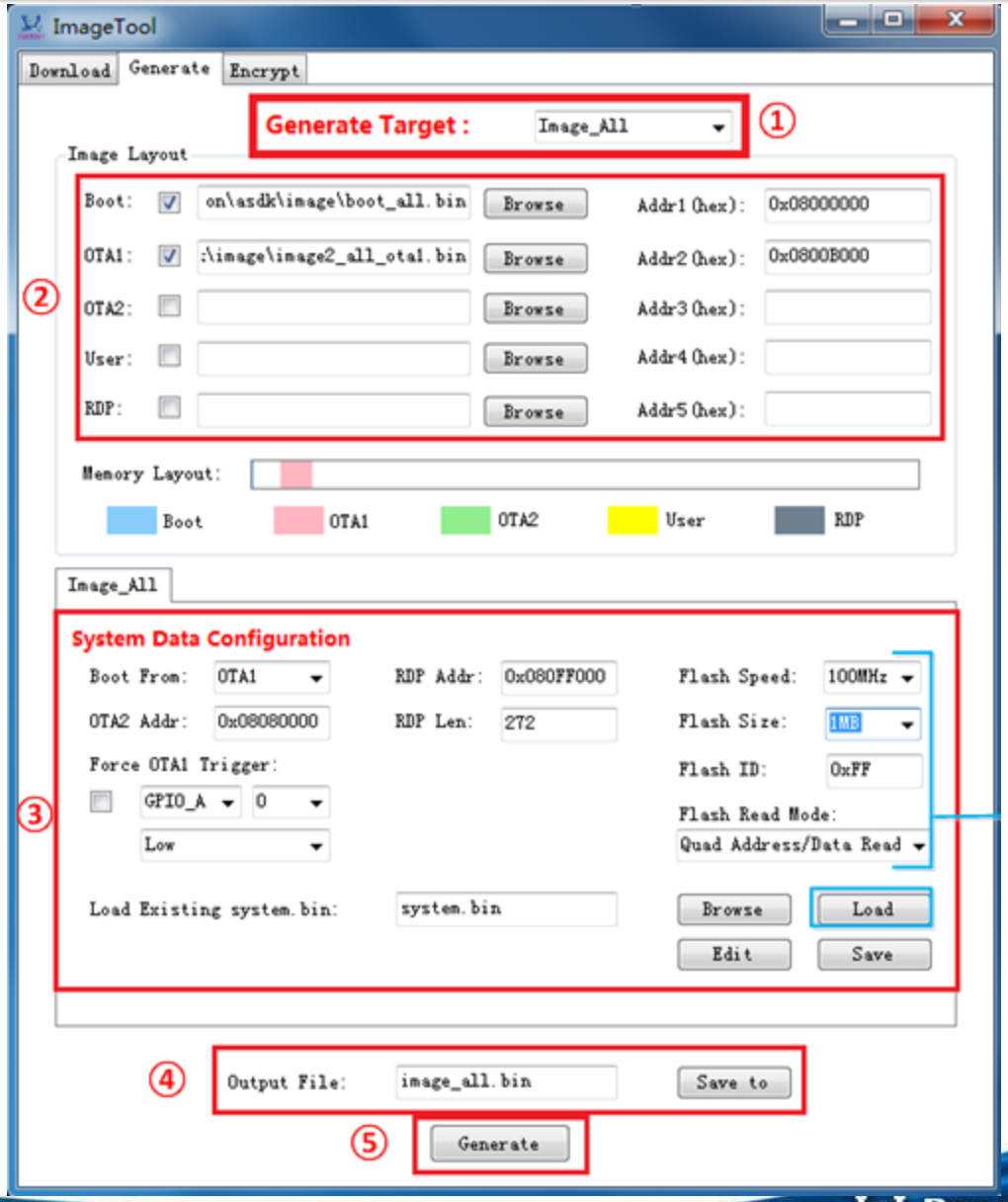


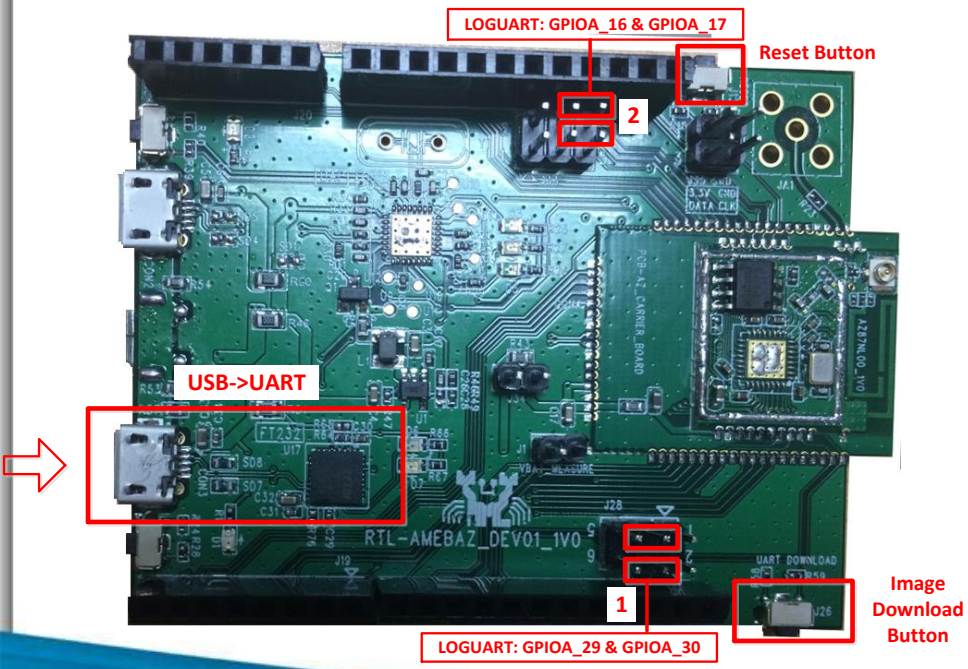
Image Generation (Ref: AN0112)

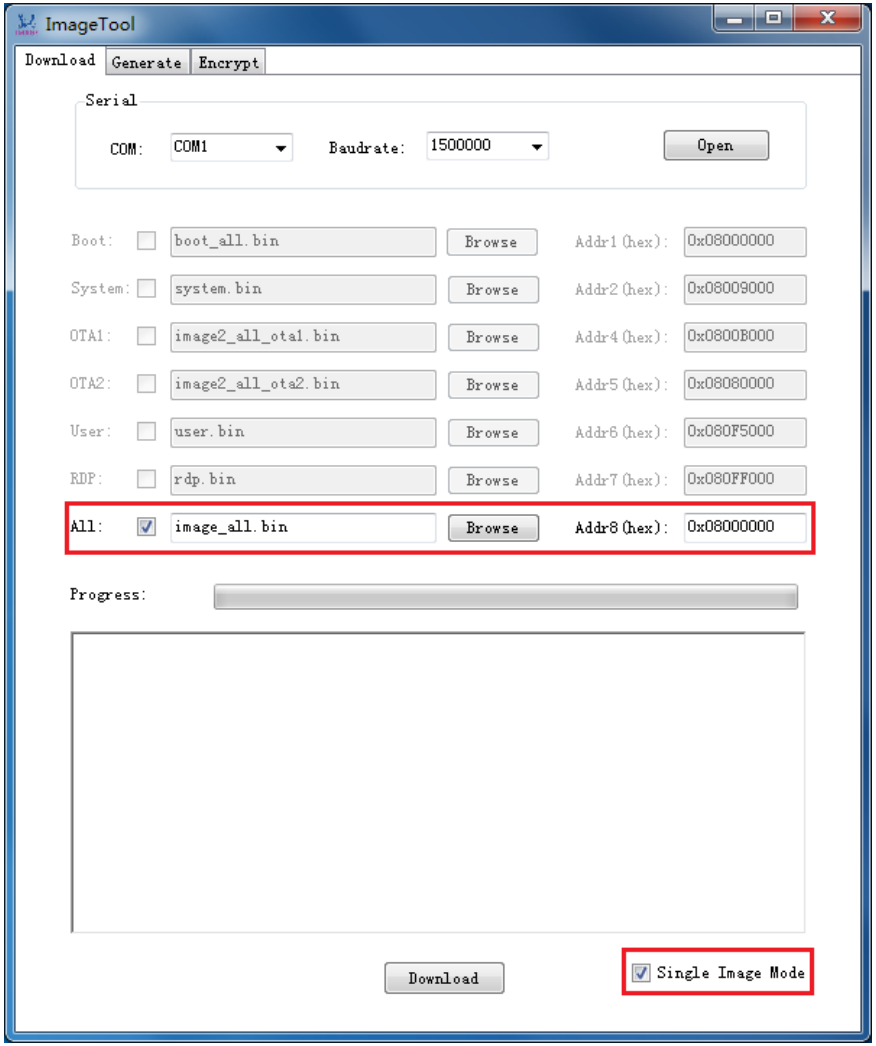
- Prepare binary files
  - boot_all.bin
  - image2_all_ota1.bin
  - image2_all_ota2.bin
- Config System Data
  - Boot From OTA1 or OTA2
  - OTA2 Address must be the same as Addr3 in Image Layout
  - Force OTA1 Trigger
  - RDP Addr and Len
  - Flash Parameters
  - Configure or Load system.bin
- Output file
  - image_all.bin



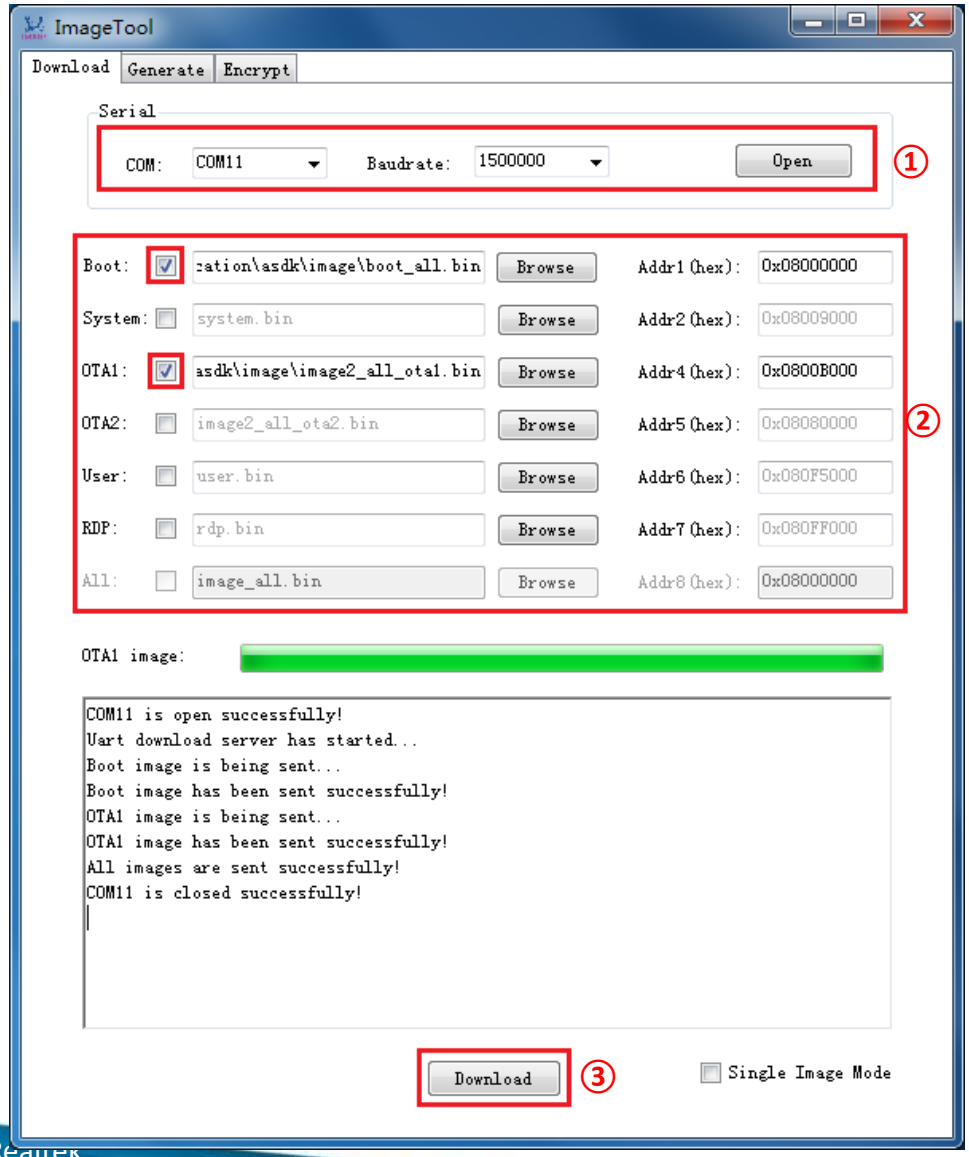
Flash Downloader for Factory (Ref: AN0112)

- Demo board: Enter download mode
  - Push the Image Download Button and keep it pressed
  - Power on the board or press the Reset Button
  - Release the Image Download Button
- Image tool
  - Choose COM and click Open
  - Choose image_all.bin
  - Click Download



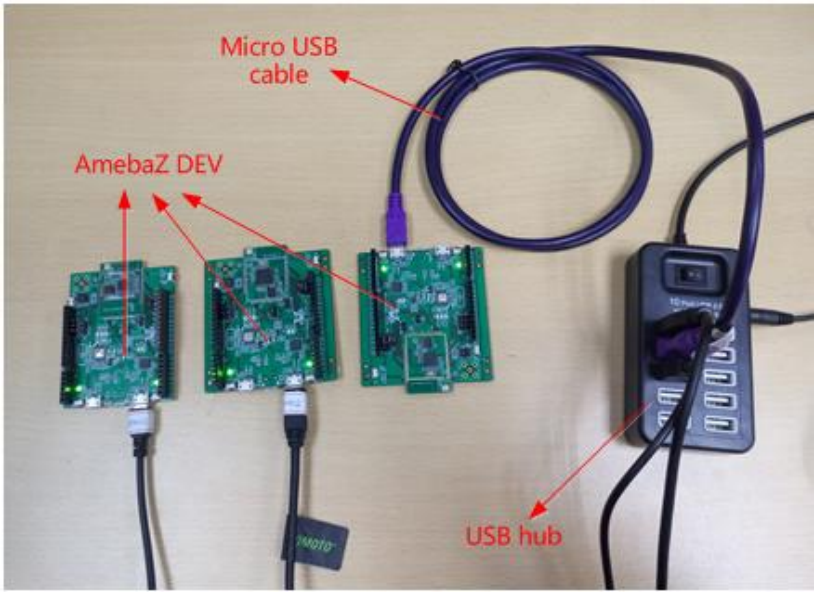


Flash Downloader for RD (Ref: AN0112)

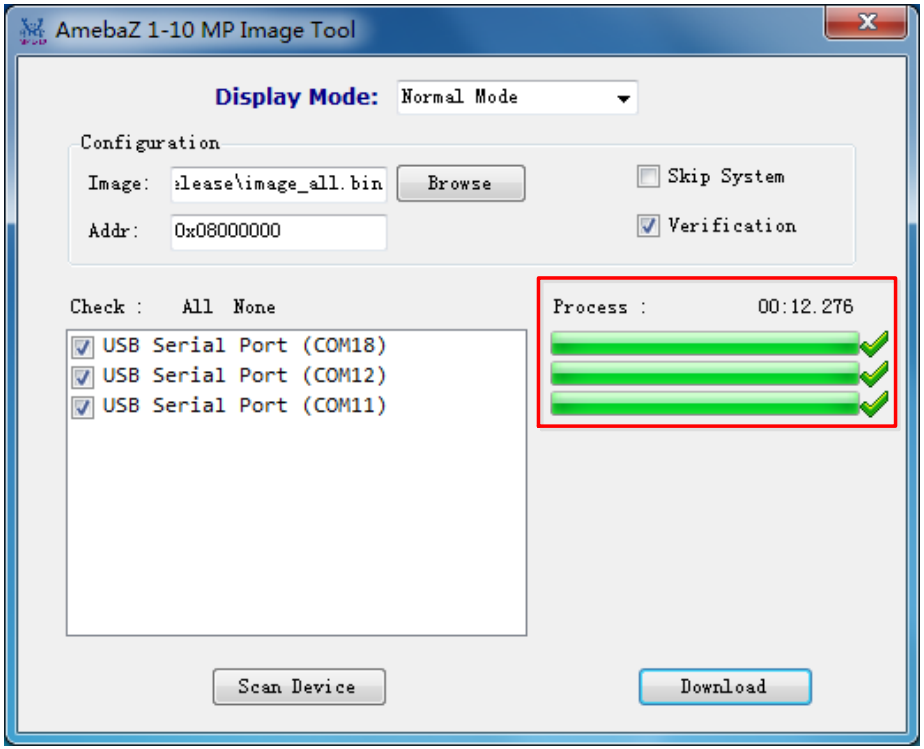


Ameba-Z 1-10 MP Image Tool

- Physical connection

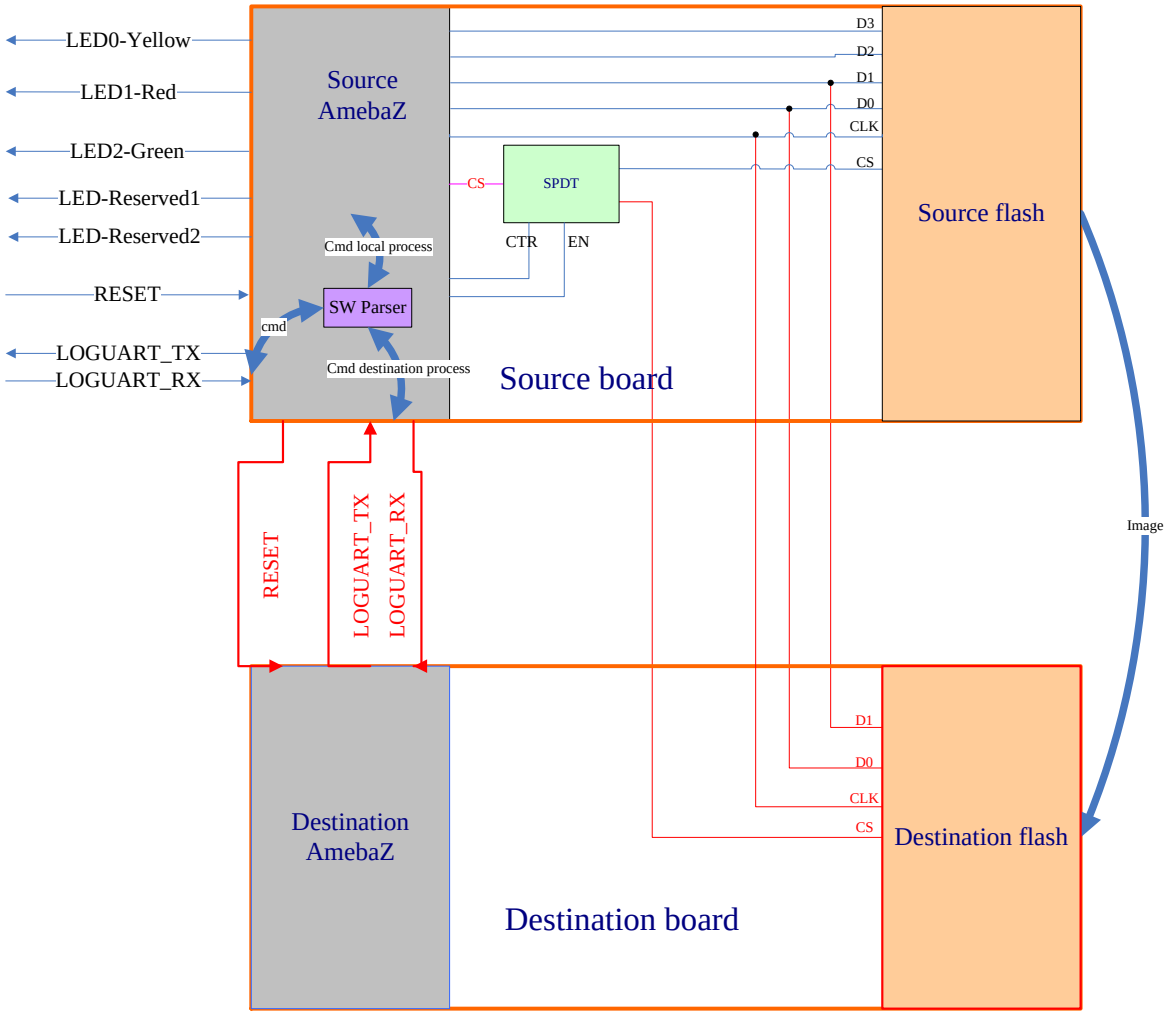


- 1-10 MP Image Tool



Flash To Flash Downloader (Ref: AN0115)

- Source board
  - On-board-flash download with DEST CPU OFF
  - Control SPDT to select SRC or DEST flash use GPIO
  - Control reset pin of DEST CPU, low for downloader, high for MP test through log UART of DEST Ameba-Z
  - Flash loader + SW Parser + DEST flash image
- LED
  - Yellow: Power on when flash programing
  - Red: Power on when flash program fail
  - Green: Power on when flash program success

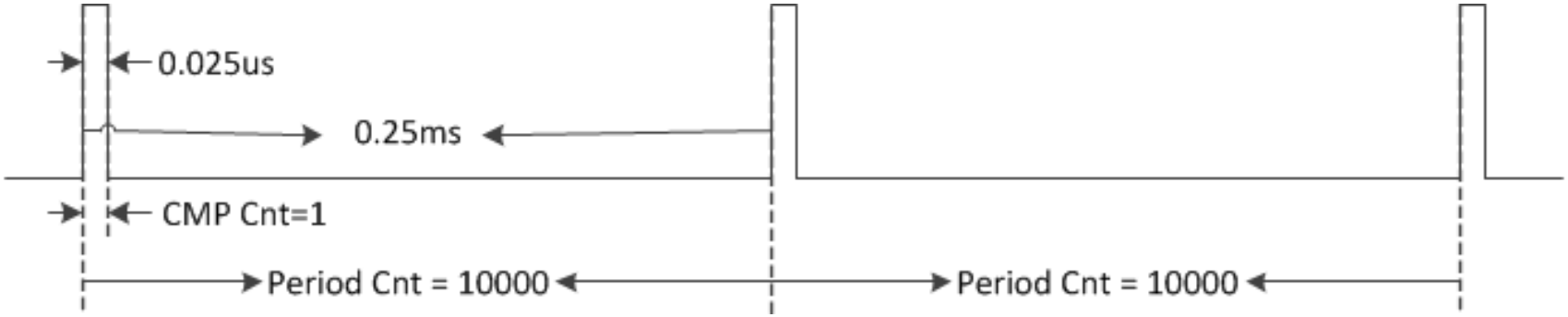


WiFi Performance

AP	Chipset	ch	Security		20M TP	40M TP
TPLINK TL-WDR4310	Atheros	1	open	TX	11.9	11.7
				RX	12.2	11.9
			WPA/WPA2AES	TX	11.4	11.5
				RX	11.9	12.8
ASUS RT-AC87U	Quantenna	1	open	TX	17	17
				RX	13.8	13.8
			WPA/WPA2AES	TX	16.7	16.8
				RX	13.4	13.1
Tenda FH456	Broadcom	1	open	TX	12.8	
				RX	8.78	
			WPA/WPA2AES	TX	12.5	
				RX	8.66	
Netgear R7000	Broadcom	1	open	TX	15.1	
				RX	13.7	
			WPA/WPA2AES	TX	14.6	
				RX	12.9	
DLINK Dir-880L	Broadcom	1	open	TX	15.8	
				RX	13.6	
			WPA/WPA2AES	TX	14.9	
				RX	13.1	

PWM

- Ameba Z PWM Parameters:
  - Clock source = 40MHz
  - Prescaler: 8Bits (Max 255)
  - Counter: 16Bits (Max 65535)
- Which means **Maximum Frequency** is **4K** if **Steps** equal to **10000**.
- PWM waveform example:
  - Prescaler = 0, then Counter clock frequency = 40M/(0+1) = 40M
  - Steps = 10000, Duty ratio = 1/10000, Frequency = 4K



MAX Steps = 65536

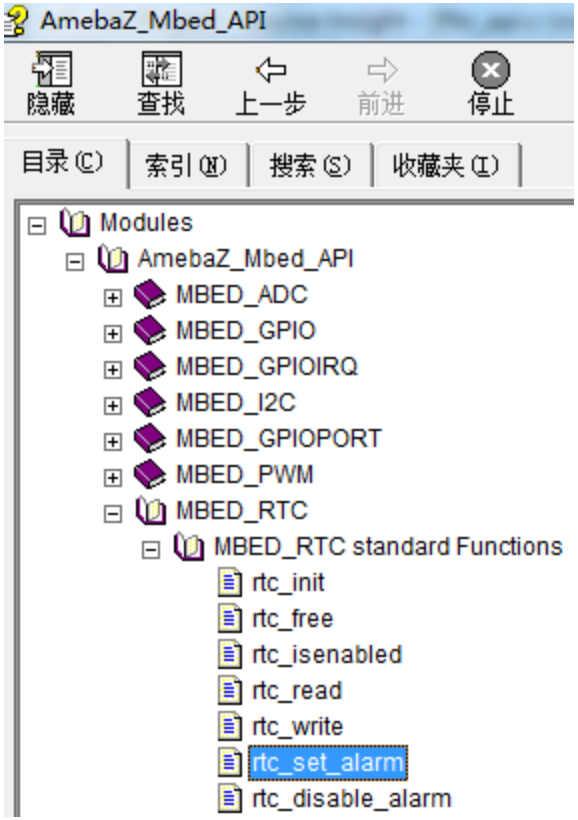
MIN Counter Clock = 40M/256 = 156K

MAX Period = 1/156K * 65536 = 0.42s

Example - RTC

```
int main()
{
  time_t seconds;
  struct tm *timeinfo;

  rtc_init();
  rtc_write(1256729737);  // Set RTC time to Wed, 28 Oct 2009 11:35:37

  while(1) {
    seconds = rtc_read();
    timeinfo = localtime(&seconds);

    DBG_8195A("Time as seconds since January 1, 1970 = %d\n", seconds);

    DBG_8195A("Time as a basic string = %s", ctime(&seconds));

    DBG_8195A("Time as a custom formatted string = %d-%d-%d %d:%d:%d\n",
      timeinfo->tm_year, timeinfo->tm_mon, timeinfo->tm_mday, timeinfo->tm_hour,
      timeinfo->tm_min,timeinfo->tm_sec);

    wait(1.0);
  }
} ? end main ?
```

```
Time as seconds since January 1, 1970 = 1256729737
Time as a basic string = Wed Oct 28 11:35:37 2009
Time as a custom formatted string = 109-9-28 11:35:37
Time as seconds since January 1, 1970 = 1256729738
Time as a basic string = Wed Oct 28 11:35:38 2009
Time as a custom formatted string = 109-9-28 11:35:38
Time as seconds since January 1, 1970 = 1256729739
Time as a basic string = Wed Oct 28 11:35:39 2009
Time as a custom formatted string = 109-9-28 11:35:39
Time as seconds since January 1, 1970 = 1256729740
Time as a basic string = Wed Oct 28 11:35:40 2009
Time as a custom formatted string = 109-9-28 11:35:40
```



Example – UART2 & GPIO Multiplex

- Assumption:
  - LOG UART TX PIN (PA_30) used to show log
  - LOG UART RX PIN (PA_29) used as normal GPIO
  - LOG UART function has been enabled

```
/* Disable LOG UART RX*/
UART_INTConfig(UART2_DEV, RUART_IER_ERBI | RUART_IER_ELSI, DISABLE);
UART_ClearRxFifo(UART2_DEV);
UART_RxCmd(UART2_DEV, DISABLE);

/* Enable GPIO function for PA29 */
Pinmux_Config(_PA_29, PINMUX_FUNCTION_GPIO);

/* Enable the GPIO interface clock */
RCC_PeriphClockCmd(APBPeriph_GPIO, APBPeriph_GPIO_CLOCK, ENABLE);

/* Configure GPIO */
GPIO_InitStruct.GPIO_Pin = _PA_29;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(&GPIO_InitStruct);

/* Output 1 */
GPIO_WriteBit(_PA_29, 1);
```

Thank you!