INTERNAL USE

Introduction to SDK 1.2.0 for MT7687/7697

Maxx Chen
ACS/ACS7/AE5
2016/03/03

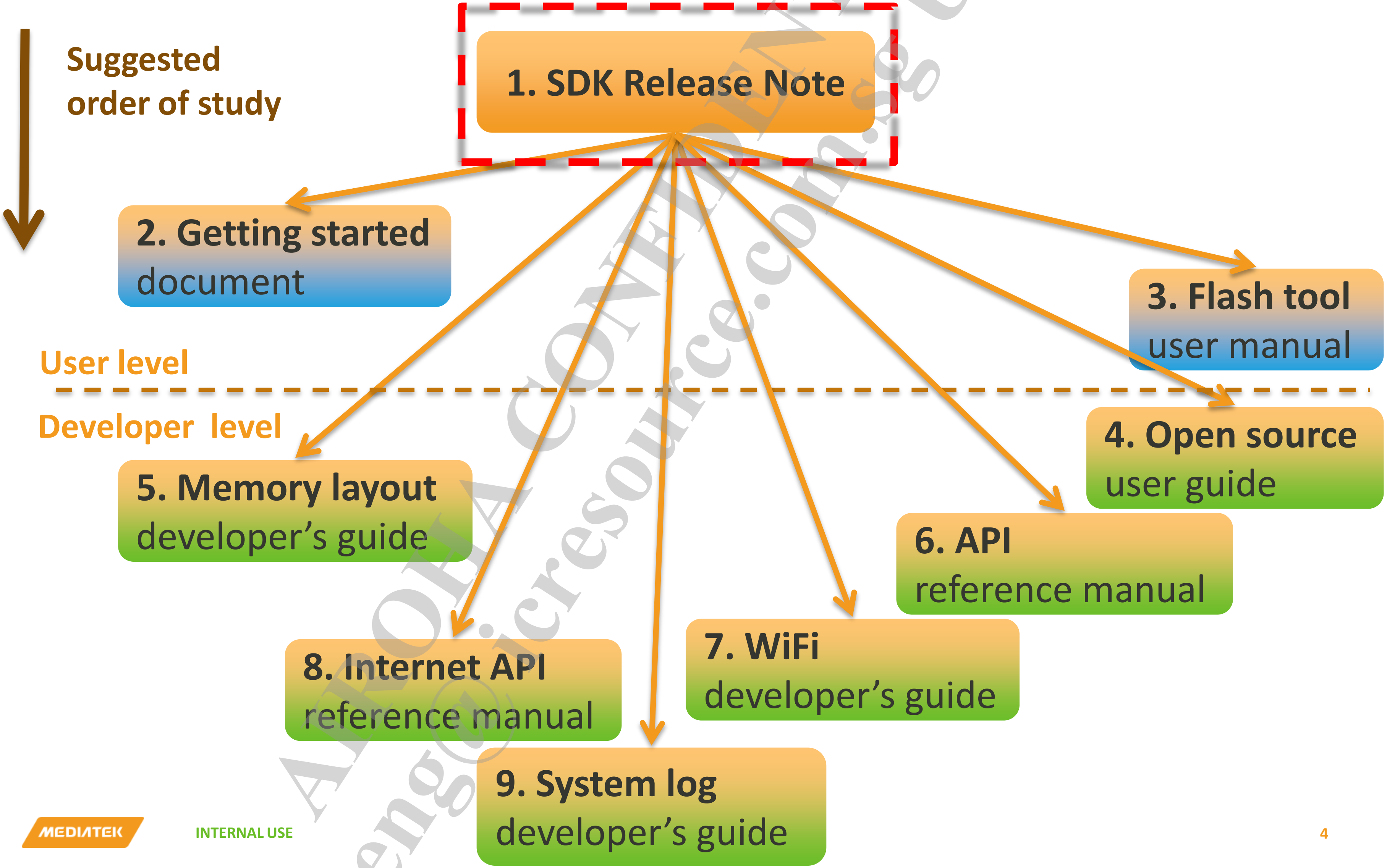
Outline

- MT7687/7697 Family
- A map to SDK
- Getting started with SDK
- Application demo - Smart connection
- Create your own project
- Add a module into Middleware

High-Level Feature – MT7687/MT7697/MT7697D

	MT7687F	MT7697	MT7697D
	Pin to Pin & SW Compatible		
Process	40nm RFCMOS technology	40nm RFCMOS technology	40nm RFCMOS technology
Package	QFN68(8x8)	QFN68(8x8)	QFN68(8x8)
Apps CPU	ARM Cortex-M4 MCU with FPU up to 192MHz clock speed	ARM Cortex-M4 MCU with FPU up to 192MHz clock speed	ARM Cortex-M4 MCU with FPU up to 192MHz clock speed
Memory	Embedded 352KB SRAM 64KB boot ROM	Embedded 352KB SRAM 64KB boot ROM	Embedded 352KB SRAM 64KB boot ROM
Flash	Embedded 2MB Supports eXecute In Place (XIP) on flash	External SPI Flash support (QPI mode) Supports eXecute In Place (XIP) on flash	External SPI Flash support (QPI mode) Supports eXecute In Place (XIP) on flash
Lower power RTC	Low power RTC mode with 32KHz crystal support	Low power RTC mode with 32KHz crystal support	Low power RTC mode with 32KHz crystal support
Security Boot & Crypto Engine	Security Boot supported Hardware crypto engines (AES, DES/3DES and SHA2)	Security Boot supported Hardware crypto engines (AES, DES/3DES and SHA2)	Security Boot supported Hardware crypto engines (AES, DES/3DES and SHA2)
RF	1x1n 2.4G	1x1n 2.4G BLE	1x1n 2.4G/5G BLE
Connectivity	SPI, UART, I2C, I2S, PWM, ADC, IrDA, GPIO	SPI, UART, I2C, I2S, PWM, ADC, IrDA, GPIO	SPI, UART, I2C, I2S, PWM, ADC, IrDA, GPIO

A map to SDK



Suggested order of study

1. SDK Release Note

2. Getting started document

3. Flash tool user manual

User level

Developer level

4. Open source user guide

5. Memory layout developer's guide

6. API reference manual

7. WiFi developer's guide

8. Internet API reference manual

9. System log developer's guide

Getting started with SDK

Getting started with SDK

- **Getting started document**
  - **“Getting_Started_with_SDK_vX.X_on_MTXXXX.pdf”**
  - this document includes the feature lists and step by step to setup the environment for the SDK.

Main features-WiFi

Table 1 Wi-Fi station features

Item	Features
Standard	802.11 b/g/n station
Channel	Channel 1~13
Personal Security	Open, WEP-Open, WPA, WPA2
Enterprise Security	N/A
WPS	Enrollee (PBC / PIN)
Advanced	AMPDU, Rx-Filter, DTIM

Table 2 Wi-Fi AP features

Item	Features
Standard	802.11 b/g/n Soft AP
Channel	Channel 1~13
Personal Security	Open, WEP-Open, WPA, WPA2
Support Clients	9 STAs(AP only mode)
WPS	Registrar (PBC/PIN) , Enrollee (PIN)
Enterprise Security	N/A

Main features-Network

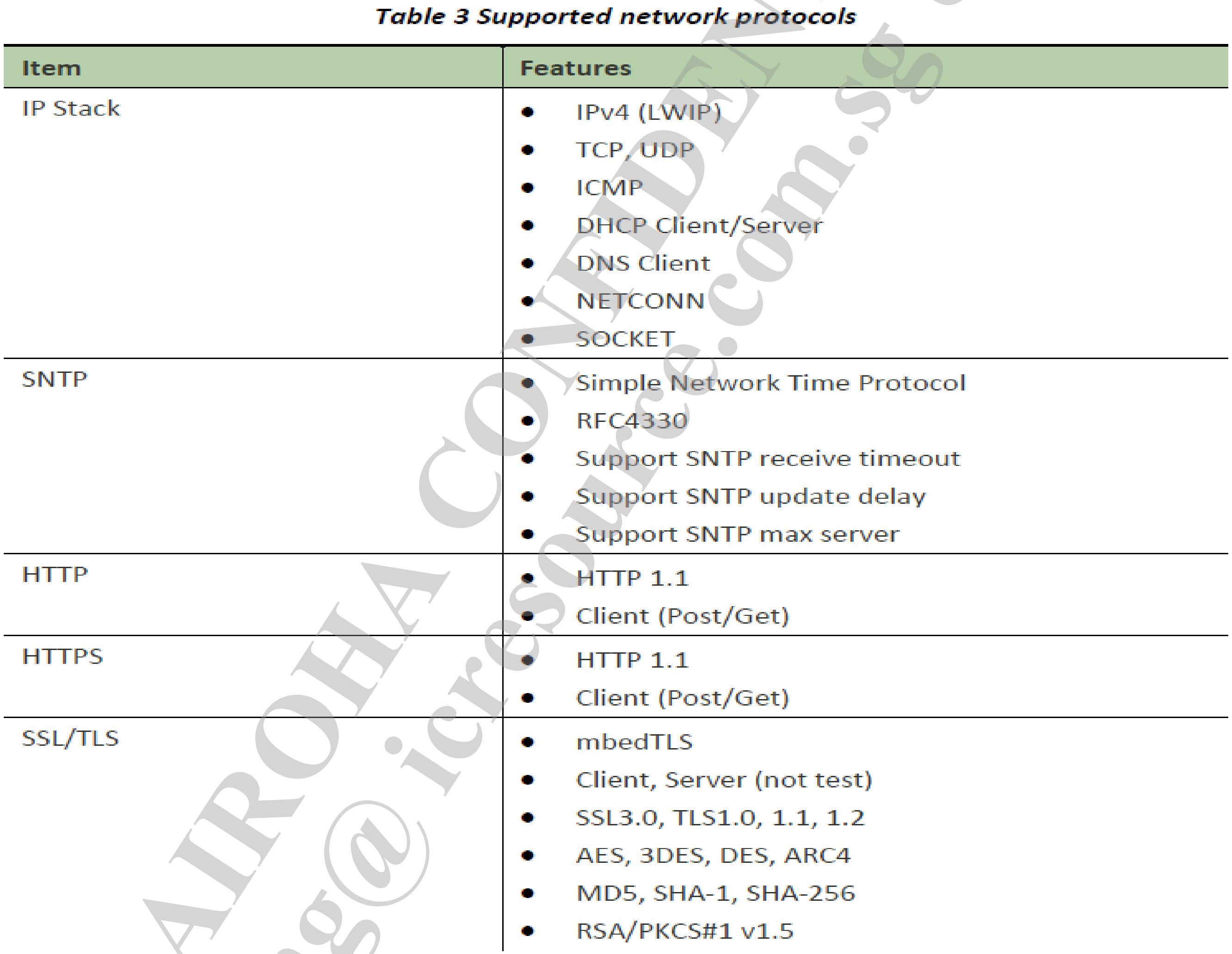
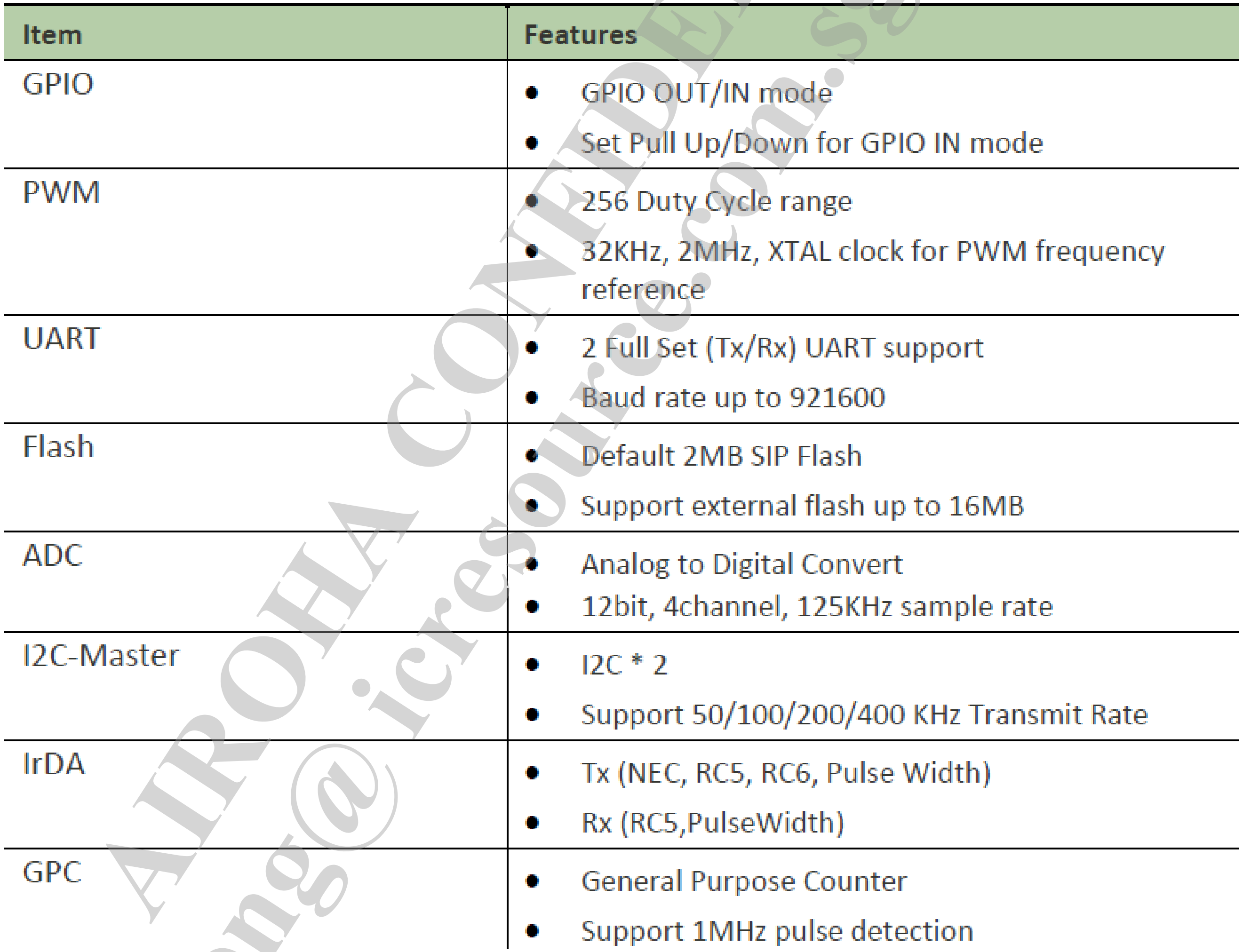


Table 3 Supported network protocols

Item	Features
IP Stack	• IPv4 (LWIP) • TCP, UDP • ICMP • DHCP Client/Server • DNS Client • NETCONN • SOCKET
SNTP	• Simple Network Time Protocol • RFC4330 • Support SNTP receive timeout • Support SNTP update delay • Support SNTP max server
HTTP	• HTTP 1.1 • Client (Post/Get)
HTTPS	• HTTP 1.1 • Client (Post/Get)
SSL/TLS	• mbedTLS • Client, Server (not test) • SSL3.0, TLS1.0, 1.1, 1.2 • AES, 3DES, DES, ARC4 • MD5, SHA-1, SHA-256 • RSA/PKCS#1 v1.5

Main features-Peripheral

Table 4 Supported peripheral drivers



Item	Features
GPIO	• GPIO OUT/IN mode • Set Pull Up/Down for GPIO IN mode
PWM	• 256 Duty Cycle range • 32KHz, 2MHz, XTAL clock for PWM frequency reference
UART	• 2 Full Set (Tx/Rx) UART support • Baud rate up to 921600
Flash	• Default 2MB SIP Flash • Support external flash up to 16MB
ADC	• Analog to Digital Convert • 12bit, 4channel, 125KHz sample rate
I2C-Master	• I2C * 2 • Support 50/100/200/400 KHz Transmit Rate
IrDA	• Tx (NEC, RC5, RC6, Pulse Width) • Rx (RC5,PulseWidth)
GPC	• General Purpose Counter • Support 1MHz pulse detection

Main features-Peripheral

Item	Features
WDT	• Support H/W, S/W watchdog • Support whole chip reset
I2S-Slave	• Support sample rate: 8k/12k/16k/24k/32k/48k • Support mono and stereo mode
SPI-Master	• Serial Peripheral Interface
RTC	• Real Time Clock
GDMA	• General Purpose DMA
Security	• SHA1, SHA2 (256, 384, 512), MD5, AES, 3DES
TRNG	• Truly Random Number Generator • Generate 32bit random number

Advanced features

Table 5 Advanced feature list

Item	Features
XML	● Mini-xml ● Support Entity ● Support Get/Set ● Support Index ● Support Search
JSON	● cJSON ● JSON string parser
Smart Connection	● MTK smart connection
CLI command	● CLI command parser

SDK on MT7687 architecture layout

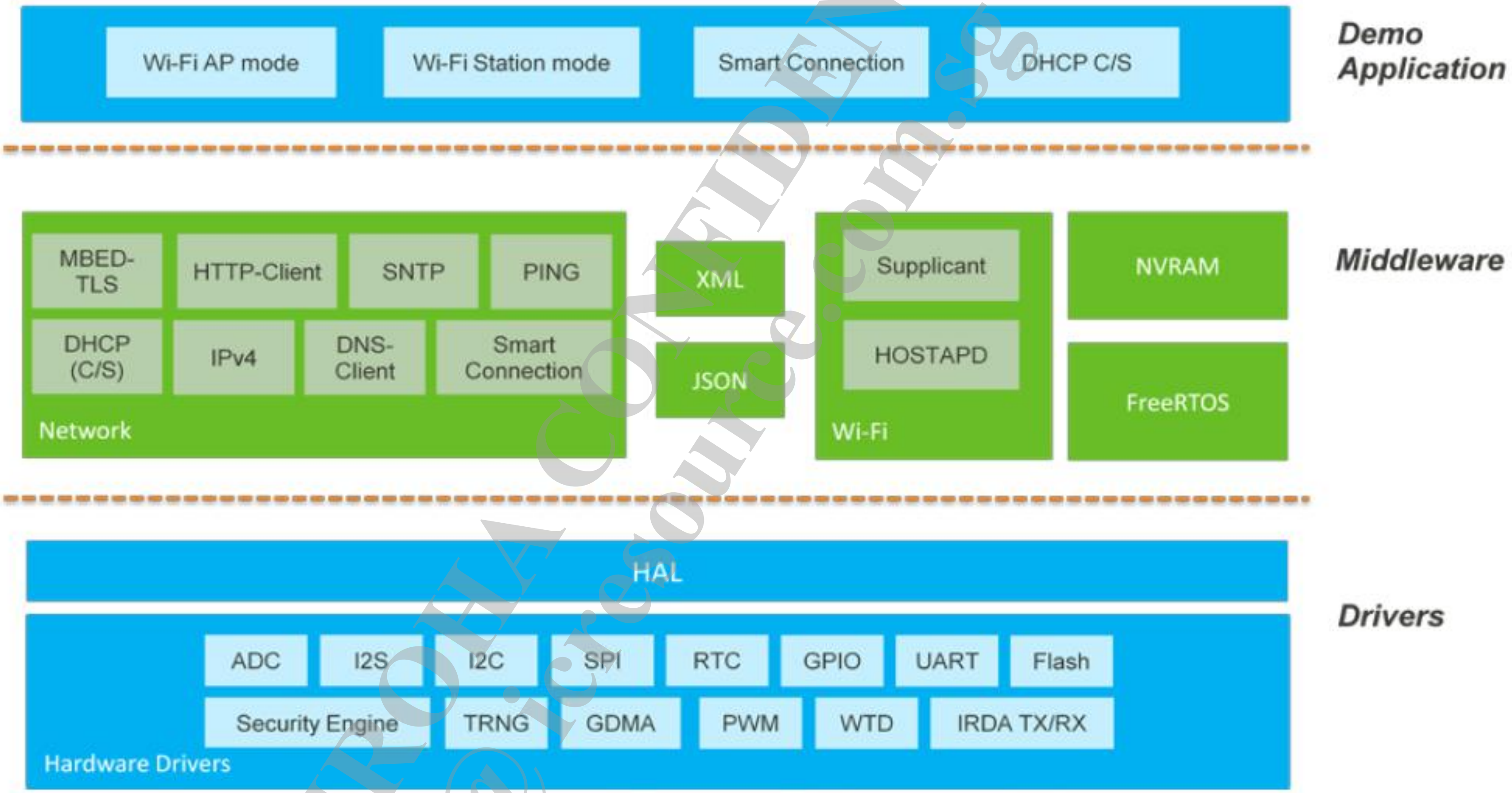


Figure 1 SDK on MT7687 architecture

System requirement

- Building the project on Linux OS.
- Downloading the project on Windows OS
- The default GCC toolchain is supported on the following versions of the Linux 32/64 bits hosts
  - Ubuntu 8.x or later (tarball).
  - Ubuntu LTS 10.04 or later (PPA).
  - RHEL 4/5/6 (tarball).

Source Folder

- Directories
  - **doc/** : documents
  - **driver/** :source code of drivers
  - **kernel/** : source code of RTOS and system services
  - **middleware/** : source code of middleware
  - **project/** : user projects
    - <board> /apps/<project>/GCC
    - <board> /apps/<project>/GCC/Makefile
    - <board> /apps/<project>/GCC/feature.mk
  - **config/** : config files of chips, boards, and projects
    - chip/<ic_name>/chip.mk
    - board/<board_name>/board.mk
    - project/<board>/<project script>
- Files
  - **build.sh** : build command see the next page

Project makefile, the main file that trigger other makefile listed in chip.mk to generate libs and form the final bin file

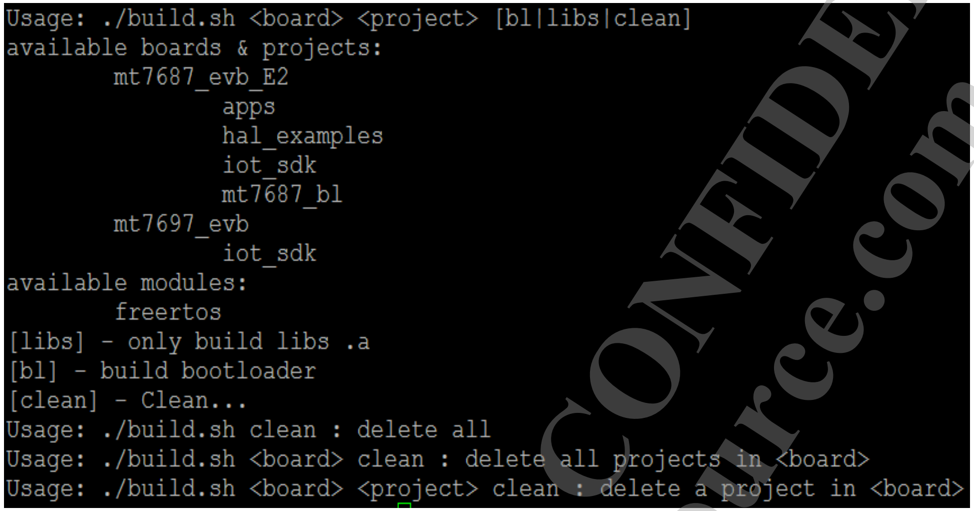
Project's feature option are defined in this file

Compiler, CFLAGS, Middleware Module Path are defined in this file

Extra CFLAGS used for each board are defined in this file

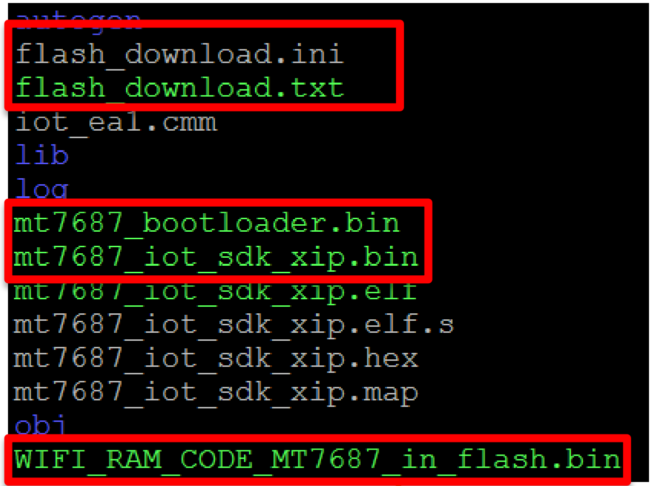
Build Image

- ./build.sh would list all available <board> <project> can be built



```
Usage: ./build.sh <board> <project> [bl|libs|clean]
available boards & projects:
        mt7687_evb_E2
                apps
                hal_examples
                iot_sdk
                mt7687_bl
        mt7697_evb
                iot_sdk
available modules:
        freertos
[libs] - only build libs .a
[bl] - build bootloader
[clean] - Clean...
Usage: ./build.sh clean : delete all
Usage: ./build.sh <board> clean : delete all projects in <board>
Usage: ./build.sh <board> <project> clean : delete a project in <board>
```

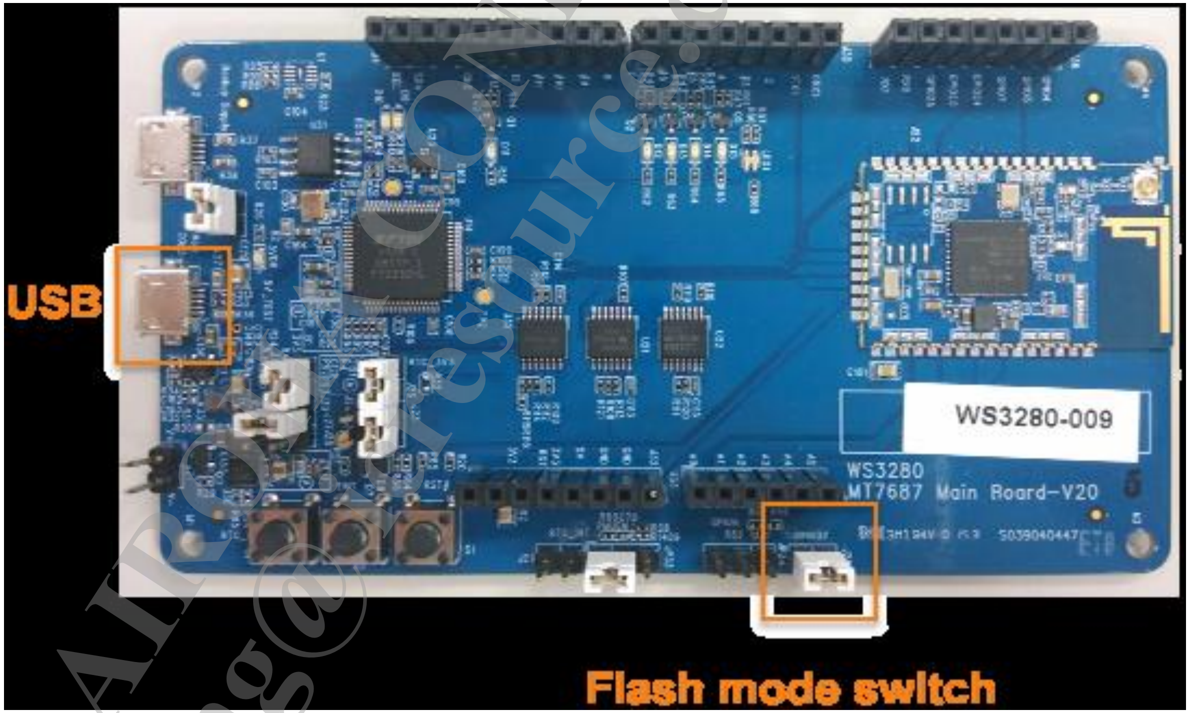
- ./build.sh <board> <project>
  - **./build.sh mt7687_evb_E2 iot_sdk**
  - 1. run the script config/project/mt7687_evb_E2/iot_sdk
  - 2. build in project/mt7687_evb_E2/apps/iot_sdk/GCC
    - -> to make libs .a in Makefile
    - -> build image
  - 3. Result is in out/mt7687_evb_E2/iot_sdk/



```
autogen
flash_download.ini
flash_download.txt
iot_eal.cmm
lib
log
mt7687_bootloader.bin
mt7687_iot_sdk_xip.bin
mt7687_iot_sdk_xip.elf
mt7687_iot_sdk_xip.elf.s
mt7687_iot_sdk_xip.hex
mt7687_iot_sdk_xip.map
obj
WIFI_RAM_CODE_MT7687_in_flash.bin
```

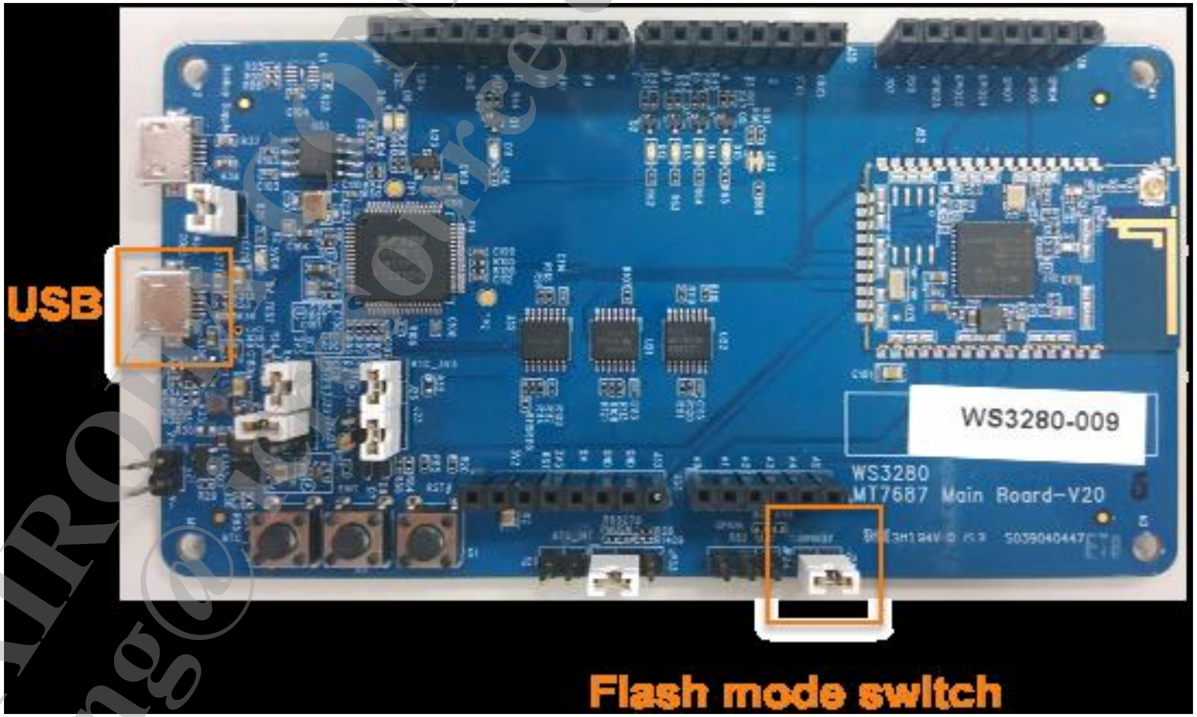
Firmware upgrade (Step 1/5)

- Set the jumper **J25 on to FLASH Recovery mode**



Firmware upgrade (Step 2/5)

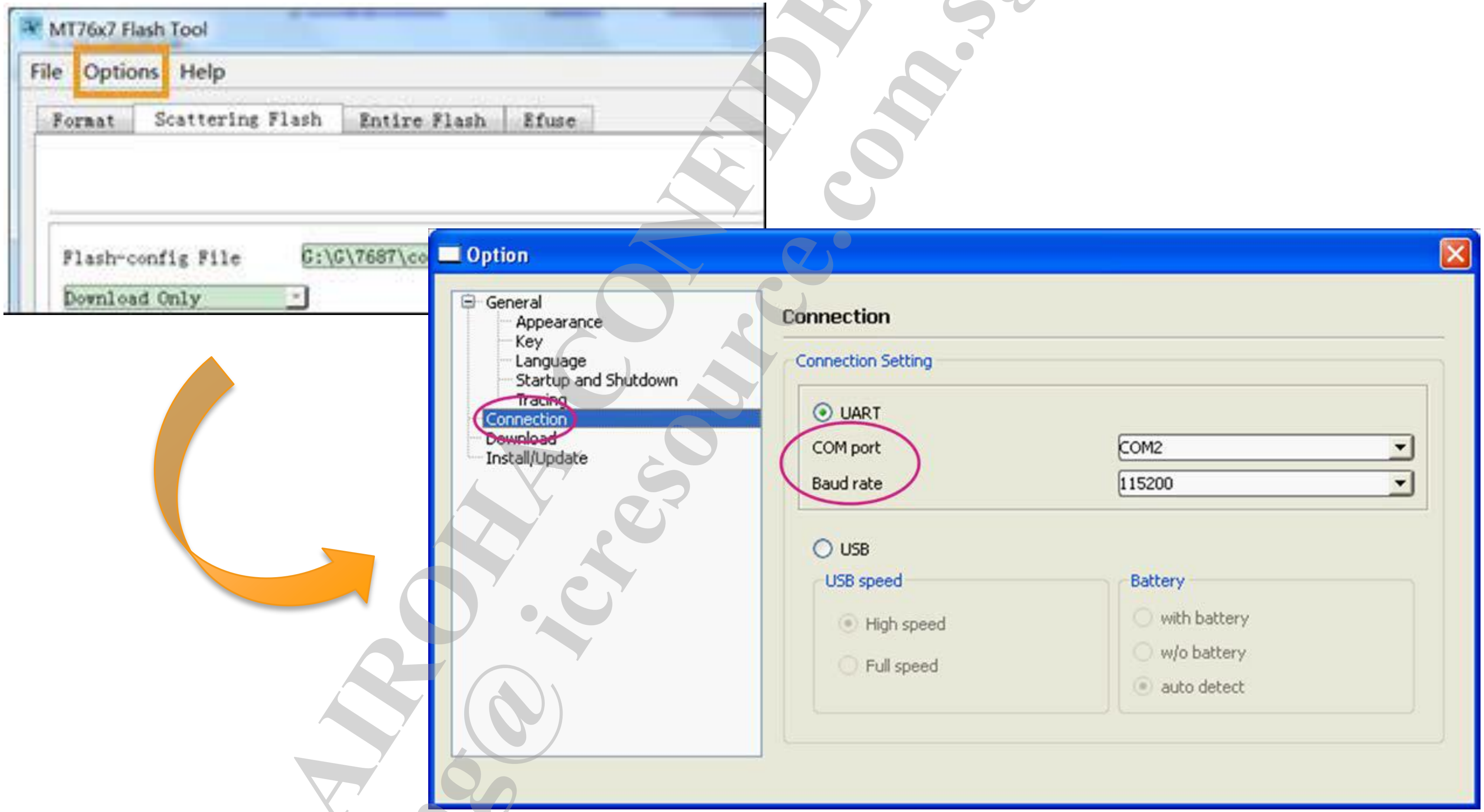
- Connect MT7687 reference board to PC using micro-USB cable.



USB

Flash mode switch

Firmware upgrade (Step 3/5)



Connection

Connection Setting

● UART

COM port: COM2

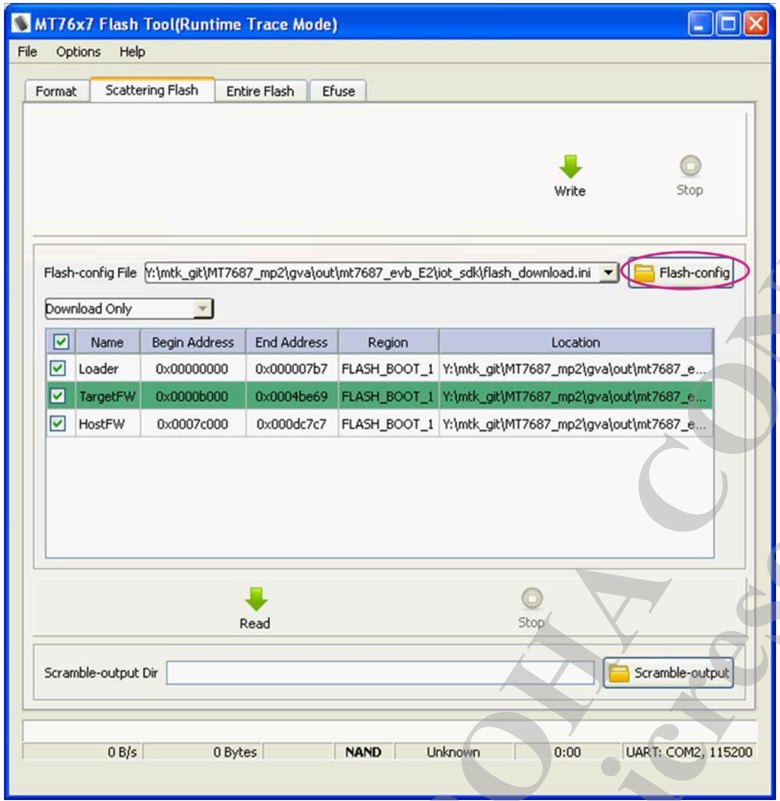
Baud rate: 115200

○ USB

USB speed: ● High speed ○ Full speed

Battery: ○ with battery ○ w/o battery ● auto detect

Firmware upgrade (Step 4/5)



MT76x7 Flash Tool(Runtime Trace Mode)

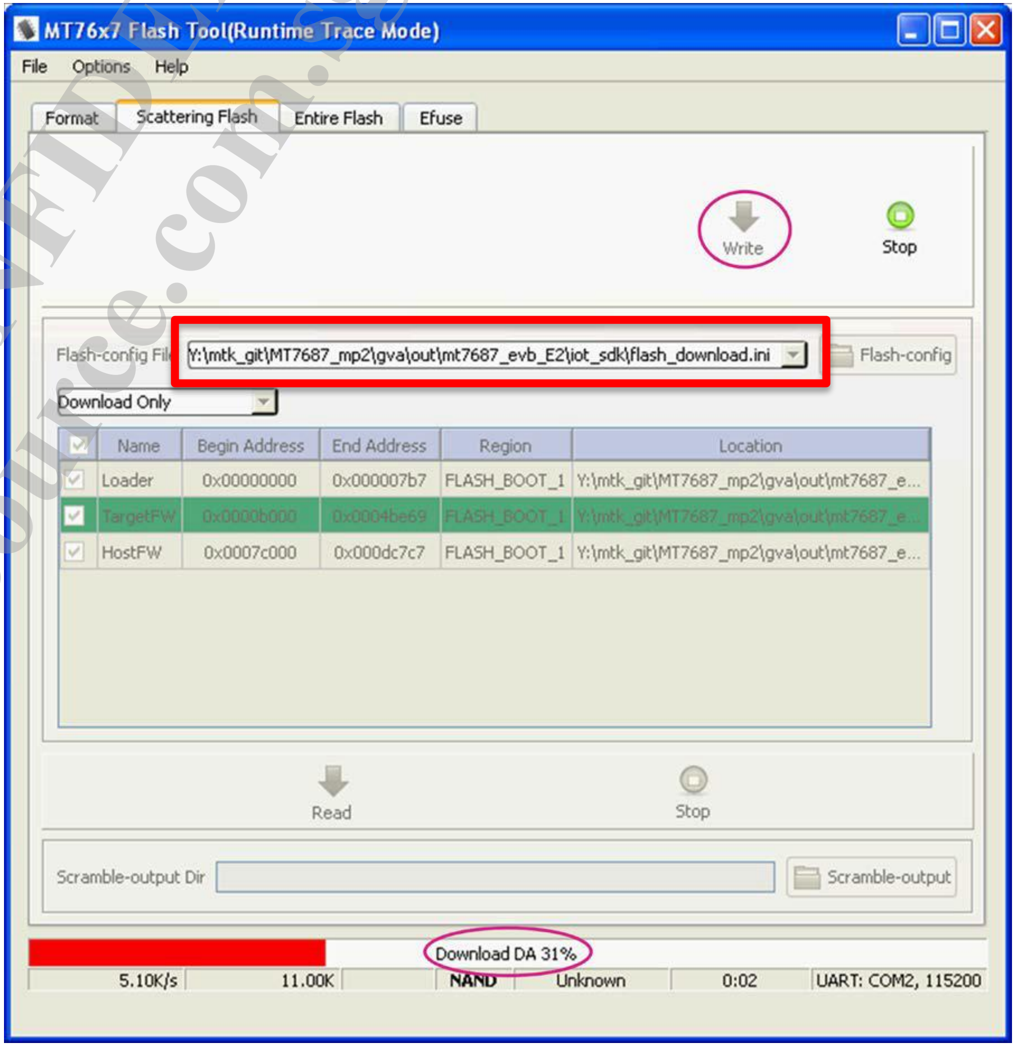
Flash-config File Y:\mtk_git\MT7687_mp2\gva\out\mt7687_evb_E2\iot_sdk\flash_download.ini

Download Only

☑	Name	Begin Address	End Address	Region	Location
☑	Loader	0x00000000	0x000007b7	FLASH_BOOT_1	Y:\mtk_git\MT7687_mp2\gva\out\mt7687_e...
☑	TargetFW	0x0000b000	0x0004be69	FLASH_BOOT_1	Y:\mtk_git\MT7687_mp2\gva\out\mt7687_e...
☑	HostFW	0x0007c000	0x000dc7c7	FLASH_BOOT_1	Y:\mtk_git\MT7687_mp2\gva\out\mt7687_e...

Scramble-output Dir

```
0 B/s | 0 Bytes | NAND | Unknown | 0:00 | UART: COM2, 115200
```



MT76x7 Flash Tool(Runtime Trace Mode)

Flash-config File Y:\mtk_git\MT7687_mp2\gva\out\mt7687_evb_E2\iot_sdk\flash_download.ini

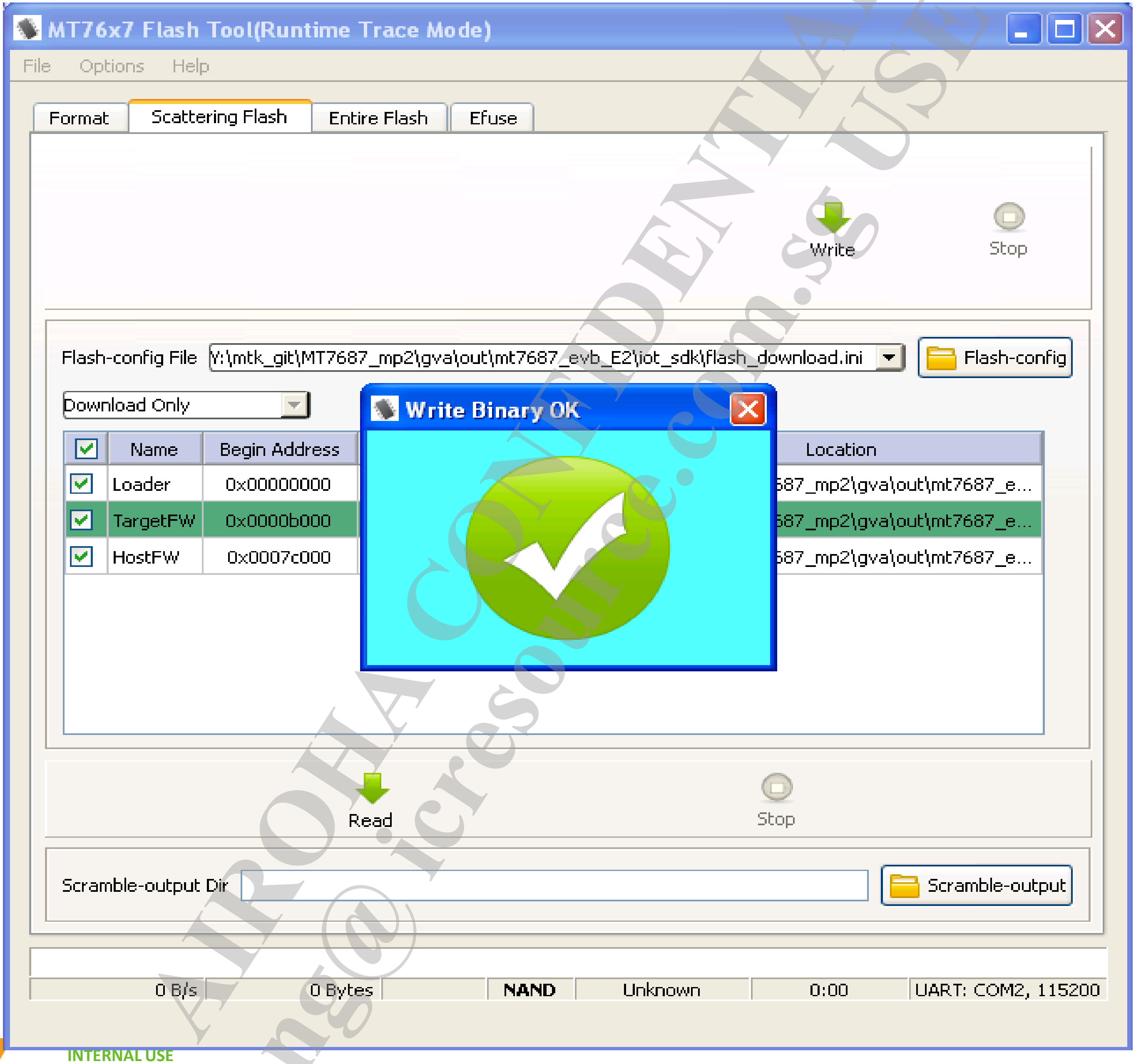
Download Only

☑	Name	Begin Address	End Address	Region	Location
☑	Loader	0x00000000	0x000007b7	FLASH_BOOT_1	Y:\mtk_git\MT7687_mp2\gva\out\mt7687_e...
☑	TargetFW	0x0000b000	0x0004be69	FLASH_BOOT_1	Y:\mtk_git\MT7687_mp2\gva\out\mt7687_e...
☑	HostFW	0x0007c000	0x000dc7c7	FLASH_BOOT_1	Y:\mtk_git\MT7687_mp2\gva\out\mt7687_e...

Scramble-output Dir

Download DA 31%

```
5.10K/s | 11.00K | NAND | Unknown | 0:02 | UART: COM2, 115200
```



Flash-config File Y:\mtk_git\MT7687_mp2\gva\out\mt7687_evb_E2\iot_sdk\flash_download.ini

Download Only

☑	Name	Begin Address	Location
☑	Loader	0x00000000	687_mp2\gva\out\mt7687_e...
☑	TargetFW	0x0000b000	687_mp2\gva\out\mt7687_e...
☑	HostFW	0x0007c000	687_mp2\gva\out\mt7687_e...

Write Binary OK

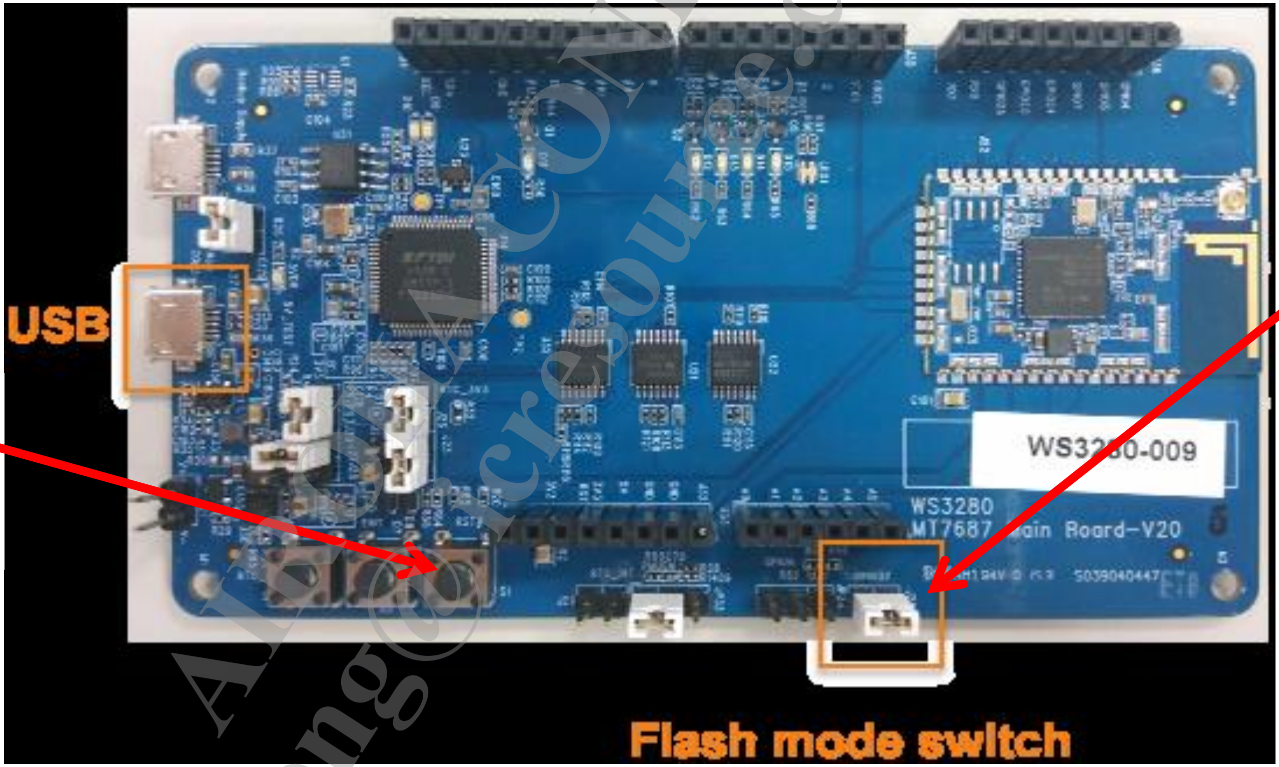
Scramble-output Dir

```
0 B/s    0 Bytes    NAND    Unknown    0:00    UART: COM2, 115200
```

Firmware upgrade (Step 5/5)

- Set the MT7687 reference board to **FLASH Normal mode** by removing J25

Step 5.2
Press Reset
Button

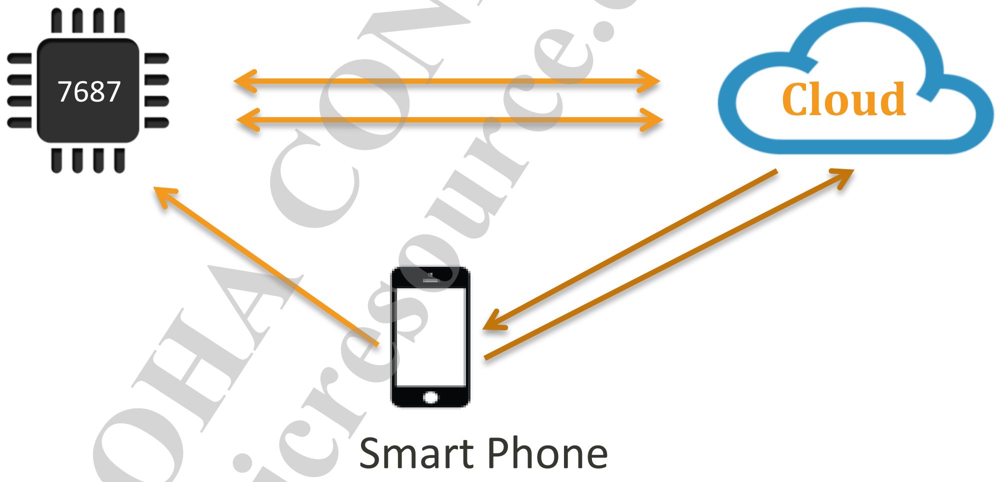


Step 5.1
Remove it !

Application demo - Smart connection

Background

- IoT device will need to connect to Wi-Fi network
- IoT device doesn't have any input interface

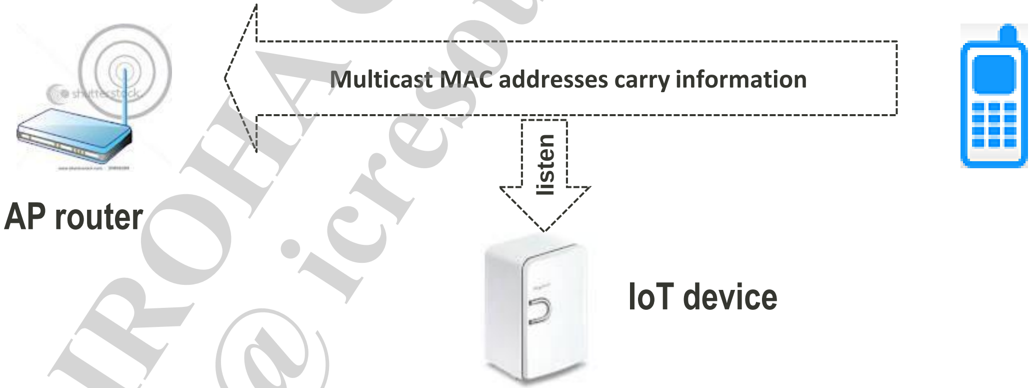


MTK smart connection protocol

- IoT device Wi-Fi set as sniffer mode, capture the packets in the air.
- In sniffer mode, IoT device doesn’t have the AP’s password and it can’t decrypt the data.
- Solution: encode data in 802.11 packet MAC address

-- 802.11 packet header is not encrypted.

-- IPv4 multicast address low-order 23 bits same as MAC address low-order 23 bits.



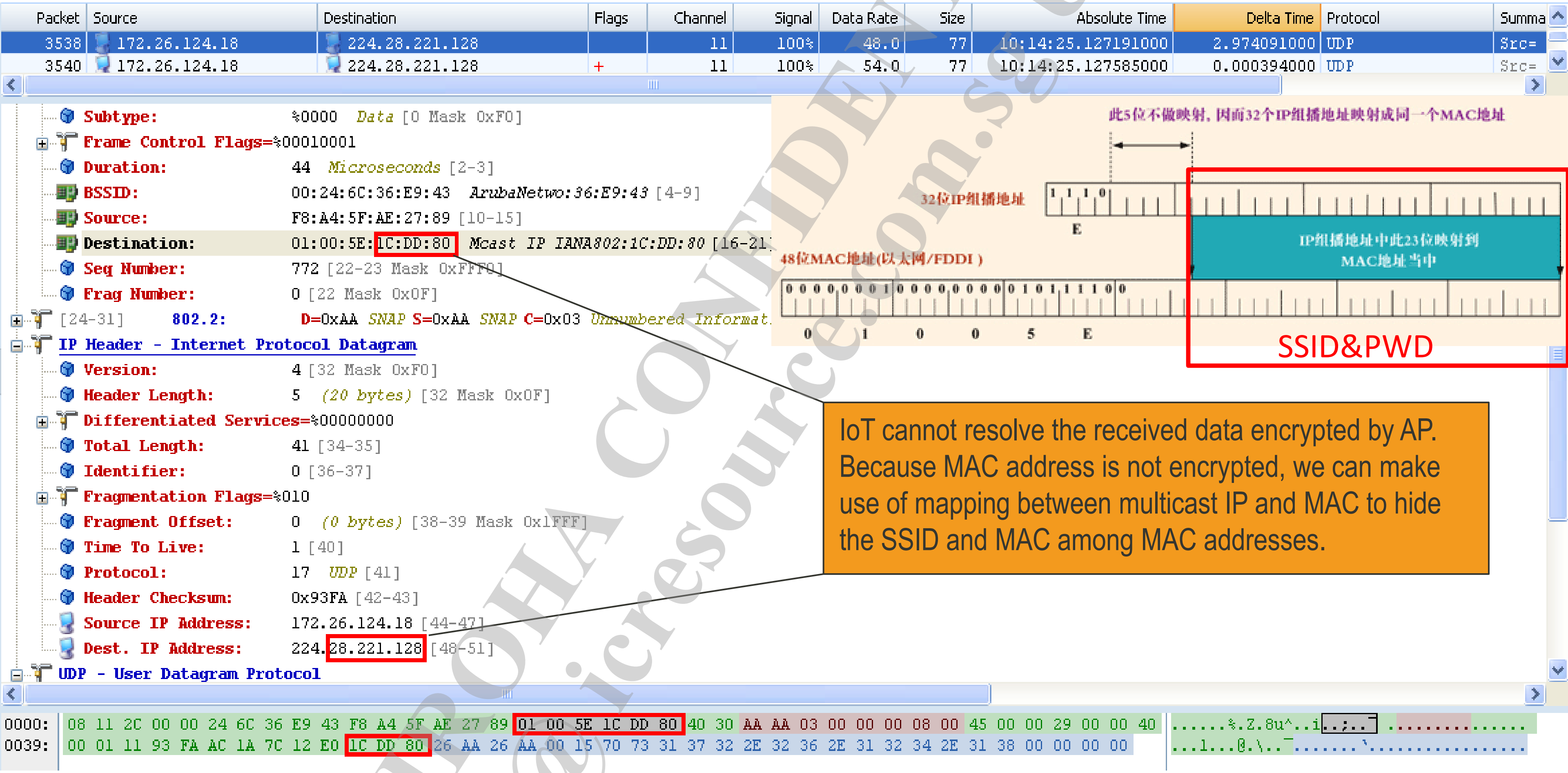
MTK smart connection protocol

IPv4 multicast address and MAC address mapping

		4bits	5bits	23bits
IPv4		1110	The 5 bits not used	Multicast group ID
MAC	00000001 00000000 01011110 0			Multicast group ID
	25bits			Low-order 23 bits of multicast group ID are copied to Ethernet address

- IPv4 multicast address mapping MAC address range：
  01:00:5e:00:00:00 ~ 01:00:5e:7f:ff:ff
- IPv4 multicast address low-order 23 bits are copied to Ethernet MAC address low-order 23 bits.
- encode data in IPv4 multicast address low-order 23 bits (SSID, PWD…).

MTK smart connection protocol



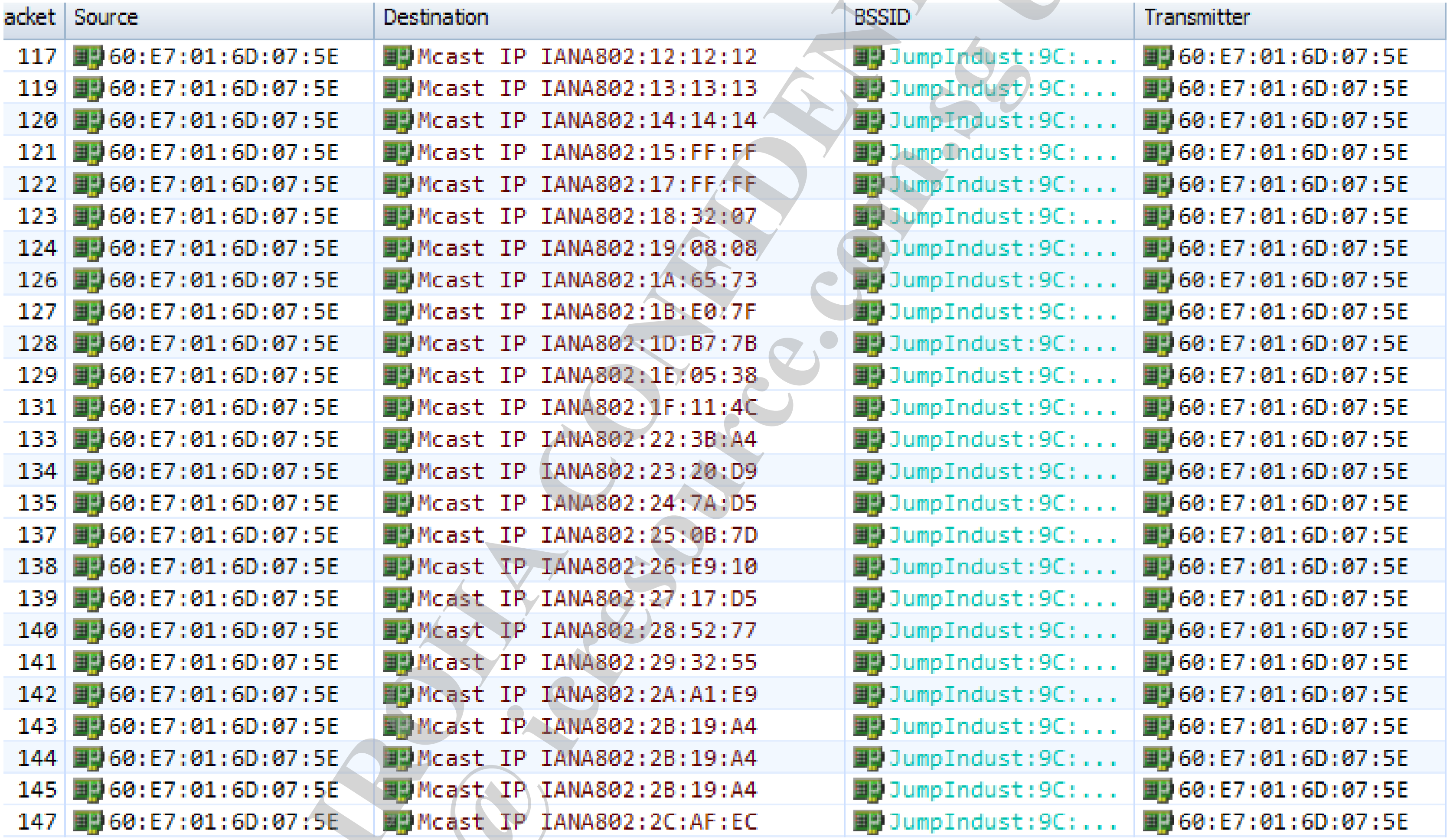
Packet	Source	Destination	Flags	Channel	Signal	Data Rate	Size	Absolute Time	Delta Time	Protocol	Summa
3538	172.26.124.18	224.28.221.128		11	100%	48.0	77	10:14:25.127191000	2.974091000	UDP	Src=
3540	172.26.124.18	224.28.221.128	+	11	100%	54.0	77	10:14:25.127585000	0.000394000	UDP	Src=

```
    Subtype:                  %0000  Data [0 Mask 0xF0]
 ⊞ Frame Control Flags=%00010001
    Duration:                 44  Microseconds [2-3]
    BSSID:                    00:24:6C:36:E9:43  ArubaNetwo:36:E9:43 [4-9]
    Source:                   F8:A4:5F:AE:27:89 [10-15]
    Destination:              01:00:5E:1C:DD:80  Mcast IP IANA802:1C:DD:80 [16-21]
    Seq Number:               772 [22-23 Mask 0xFFF0]
    Frag Number:              0 [22 Mask 0x0F]
⊞ [24-31]      802.2:     D=0xAA SNAP S=0xAA SNAP C=0x03 Unnumbered Informat.
⊟ IP Header - Internet Protocol Datagram
    Version:                  4 [32 Mask 0xF0]
    Header Length:            5  (20 bytes) [32 Mask 0x0F]
 ⊞ Differentiated Services=%00000000
    Total Length:             41 [34-35]
    Identifier:               0 [36-37]
 ⊞ Fragmentation Flags=%010
    Fragment Offset:          0  (0 bytes) [38-39 Mask 0x1FFF]
    Time To Live:             1 [40]
    Protocol:                 17  UDP [41]
    Header Checksum:          0x93FA [42-43]
    Source IP Address:        172.26.124.18 [44-47]
    Dest. IP Address:         224.28.221.128 [48-51]
⊟ UDP - User Datagram Protocol
```

```
0000: 08 11 2C 00 00 24 6C 36 E9 43 F8 A4 5F AE 27 89 01 00 5E 1C DD 80 40 30 AA AA 03 00 00 00 08 00 45 00 00 29 00 00 40   ......%.Z.8u^..i..;.. .............
0039: 00 01 11 93 FA AC 1A 7C 12 E0 1C DD 80 26 AA 26 AA 00 15 70 73 31 37 32 2E 32 36 2E 31 32 34 2E 31 38 00 00 00 00   ...l...@.\..¯.......`...............
```

IoT cannot resolve the received data encrypted by AP. Because MAC address is not encrypted, we can make use of mapping between multicast IP and MAC to hide the SSID and MAC among MAC addresses.

MTK smart connection protocol



acket	Source	Destination	BSSID	Transmitter
117	60:E7:01:6D:07:5E	Mcast IP IANA802:12:12:12	JumpIndust:9C:...	60:E7:01:6D:07:5E
119	60:E7:01:6D:07:5E	Mcast IP IANA802:13:13:13	JumpIndust:9C:...	60:E7:01:6D:07:5E
120	60:E7:01:6D:07:5E	Mcast IP IANA802:14:14:14	JumpIndust:9C:...	60:E7:01:6D:07:5E
121	60:E7:01:6D:07:5E	Mcast IP IANA802:15:FF:FF	JumpIndust:9C:...	60:E7:01:6D:07:5E
122	60:E7:01:6D:07:5E	Mcast IP IANA802:17:FF:FF	JumpIndust:9C:...	60:E7:01:6D:07:5E
123	60:E7:01:6D:07:5E	Mcast IP IANA802:18:32:07	JumpIndust:9C:...	60:E7:01:6D:07:5E
124	60:E7:01:6D:07:5E	Mcast IP IANA802:19:08:08	JumpIndust:9C:...	60:E7:01:6D:07:5E
126	60:E7:01:6D:07:5E	Mcast IP IANA802:1A:65:73	JumpIndust:9C:...	60:E7:01:6D:07:5E
127	60:E7:01:6D:07:5E	Mcast IP IANA802:1B:E0:7F	JumpIndust:9C:...	60:E7:01:6D:07:5E
128	60:E7:01:6D:07:5E	Mcast IP IANA802:1D:B7:7B	JumpIndust:9C:...	60:E7:01:6D:07:5E
129	60:E7:01:6D:07:5E	Mcast IP IANA802:1E:05:38	JumpIndust:9C:...	60:E7:01:6D:07:5E
131	60:E7:01:6D:07:5E	Mcast IP IANA802:1F:11:4C	JumpIndust:9C:...	60:E7:01:6D:07:5E
133	60:E7:01:6D:07:5E	Mcast IP IANA802:22:3B:A4	JumpIndust:9C:...	60:E7:01:6D:07:5E
134	60:E7:01:6D:07:5E	Mcast IP IANA802:23:20:D9	JumpIndust:9C:...	60:E7:01:6D:07:5E
135	60:E7:01:6D:07:5E	Mcast IP IANA802:24:7A:D5	JumpIndust:9C:...	60:E7:01:6D:07:5E
137	60:E7:01:6D:07:5E	Mcast IP IANA802:25:0B:7D	JumpIndust:9C:...	60:E7:01:6D:07:5E
138	60:E7:01:6D:07:5E	Mcast IP IANA802:26:E9:10	JumpIndust:9C:...	60:E7:01:6D:07:5E
139	60:E7:01:6D:07:5E	Mcast IP IANA802:27:17:D5	JumpIndust:9C:...	60:E7:01:6D:07:5E
140	60:E7:01:6D:07:5E	Mcast IP IANA802:28:52:77	JumpIndust:9C:...	60:E7:01:6D:07:5E
141	60:E7:01:6D:07:5E	Mcast IP IANA802:29:32:55	JumpIndust:9C:...	60:E7:01:6D:07:5E
142	60:E7:01:6D:07:5E	Mcast IP IANA802:2A:A1:E9	JumpIndust:9C:...	60:E7:01:6D:07:5E
143	60:E7:01:6D:07:5E	Mcast IP IANA802:2B:19:A4	JumpIndust:9C:...	60:E7:01:6D:07:5E
144	60:E7:01:6D:07:5E	Mcast IP IANA802:2B:19:A4	JumpIndust:9C:...	60:E7:01:6D:07:5E
145	60:E7:01:6D:07:5E	Mcast IP IANA802:2B:19:A4	JumpIndust:9C:...	60:E7:01:6D:07:5E
147	60:E7:01:6D:07:5E	Mcast IP IANA802:2C:AF:EC	JumpIndust:9C:...	60:E7:01:6D:07:5E

MTK smart connection protocol

IPv4 multicast address and MAC address mapping

7bits	8bits	8bits	Payload
idx	C0	C1	payload

Encode format:

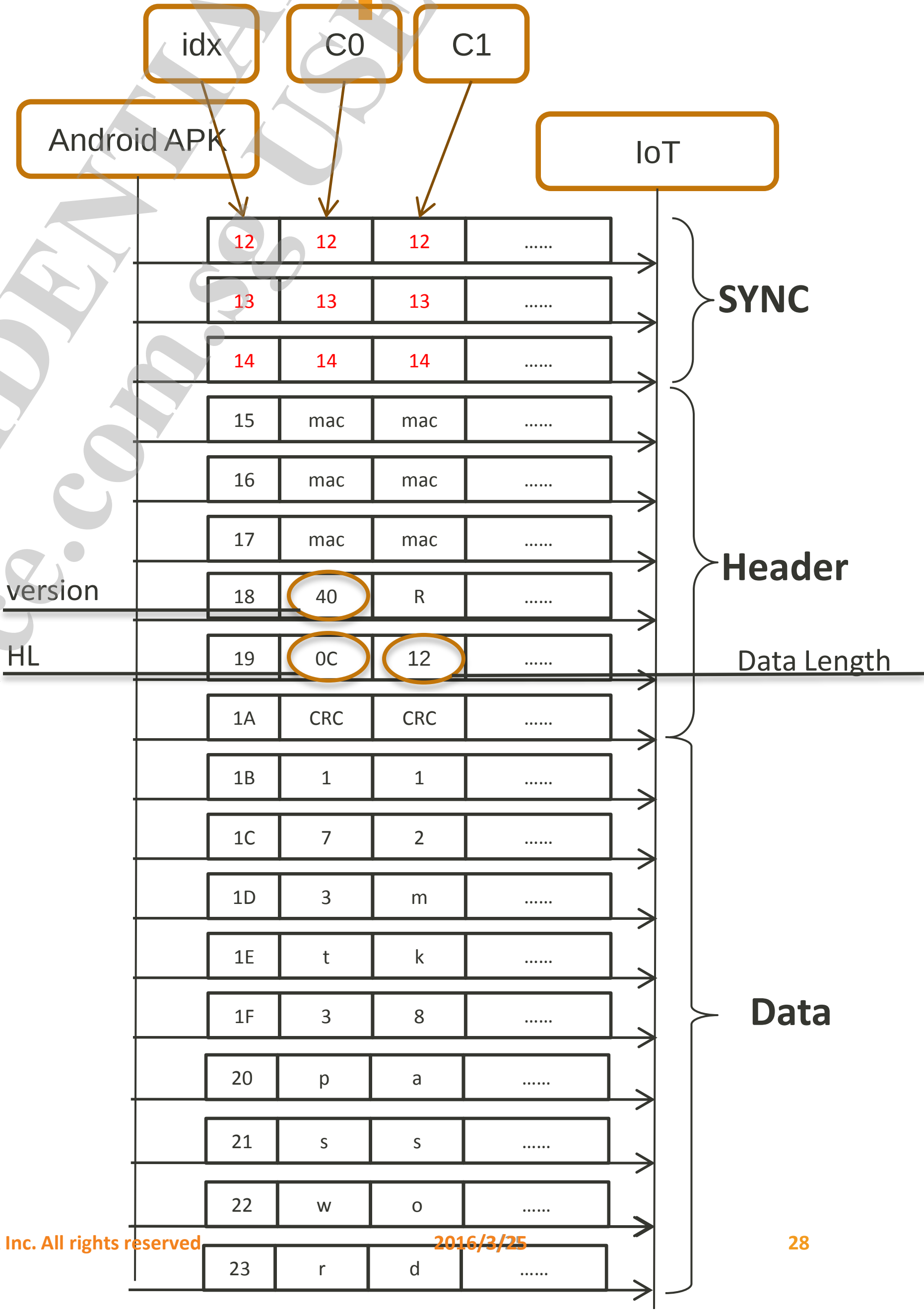
- Bits[0~15] : carry information
- Bit[16~23} : index

Example

- ssid = “mtk”
- pwd = “password”

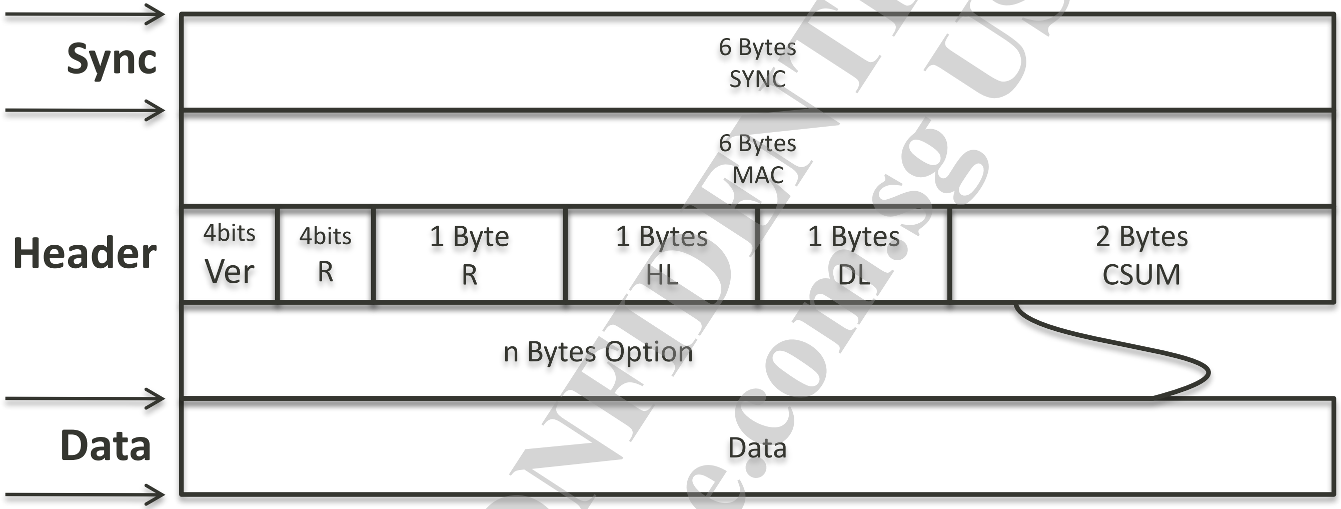
Notes:

1. SYNC and MAC ADDR repeat 3 times.
2. Length of data field is fixed, used as a filter.
3. Idx start from 0x15.
4. All above digits are hexadecimal.
5. mac field used to configure which device to receive the packets, set to FF:FF:FF:FF:FF:FF means configuring all IoT devices.



idx	C0	C1	
12	12	12	......
13	13	13	......
14	14	14	......
15	mac	mac	......
16	mac	mac	......
17	mac	mac	......
18	40	R	......
19	0C	12	......
1A	CRC	CRC	......
1B	1	1	......
1C	7	2	......
1D	3	m	......
1E	t	k	......
1F	3	8	......
20	p	a	......
21	s	s	......
22	w	o	......
23	r	d	......

MTK smart connection protocol



Sync

6 Bytes
SYNC

Header

6 Bytes
MAC

4bits Ver	4bits R	1 Byte R	1 Bytes HL	1 Bytes DL	2 Bytes CSUM

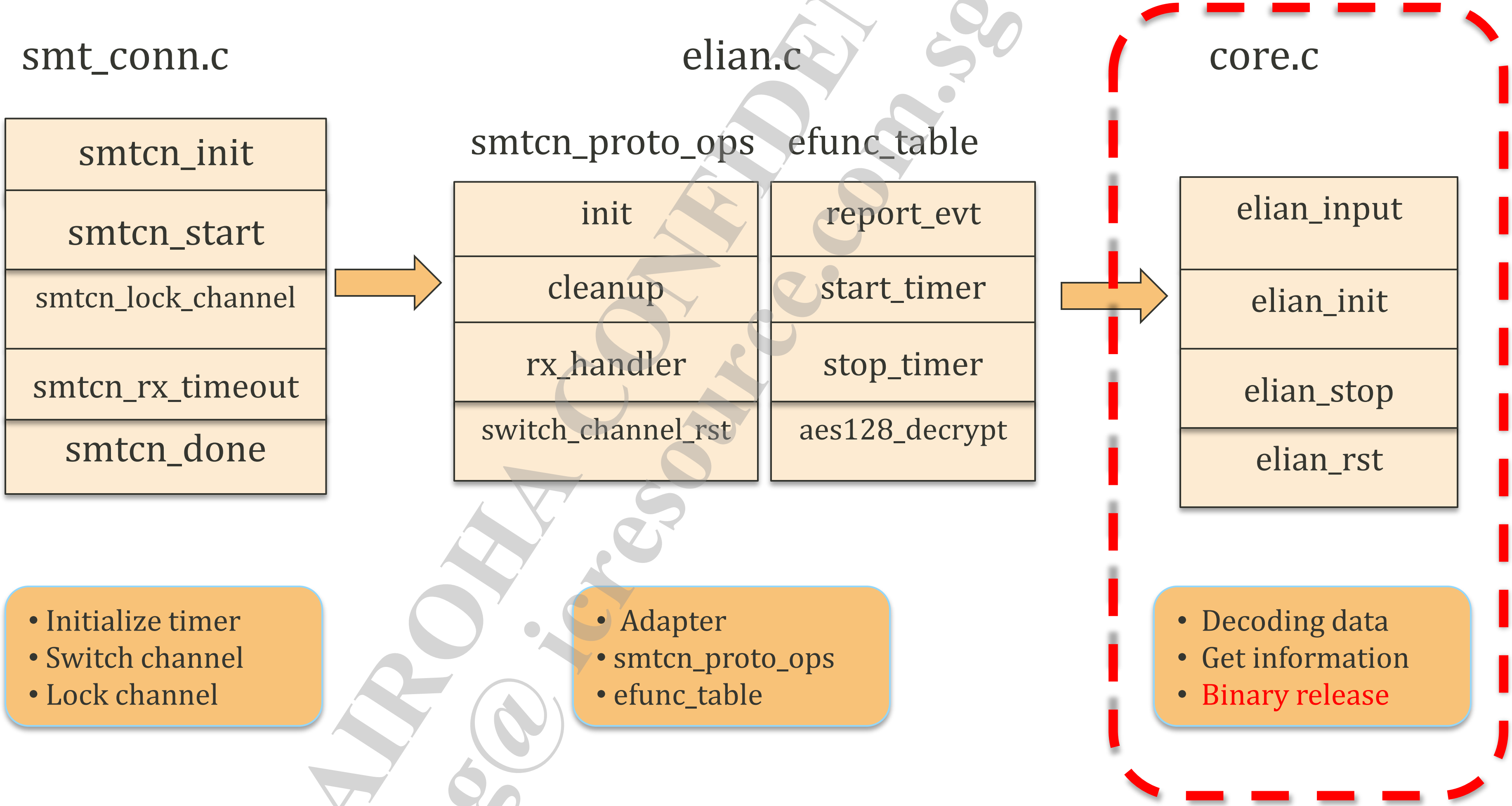
n Bytes Option

Data

Data

Sync field: 0x12, 0x12, 0x13, 0x13, 0x14, 0x14
MAC field: Target device MAC address
Ver: version code
R: reserved, default set to 0
HL: Header Length
DL: Data Length
CRC: CRC16 for Header & Data
Option: For protocol extension if needed
Data: APP Layer Data.

Source Files



smt_conn.c

- smtcn_init
- smtcn_start
- smtcn_lock_channel
- smtcn_rx_timeout
- smtcn_done

elian.c

smtcn_proto_ops	efunc_table
init	report_evt
cleanup	start_timer
rx_handler	stop_timer
switch_channel_rst	aes128_decrypt

core.c

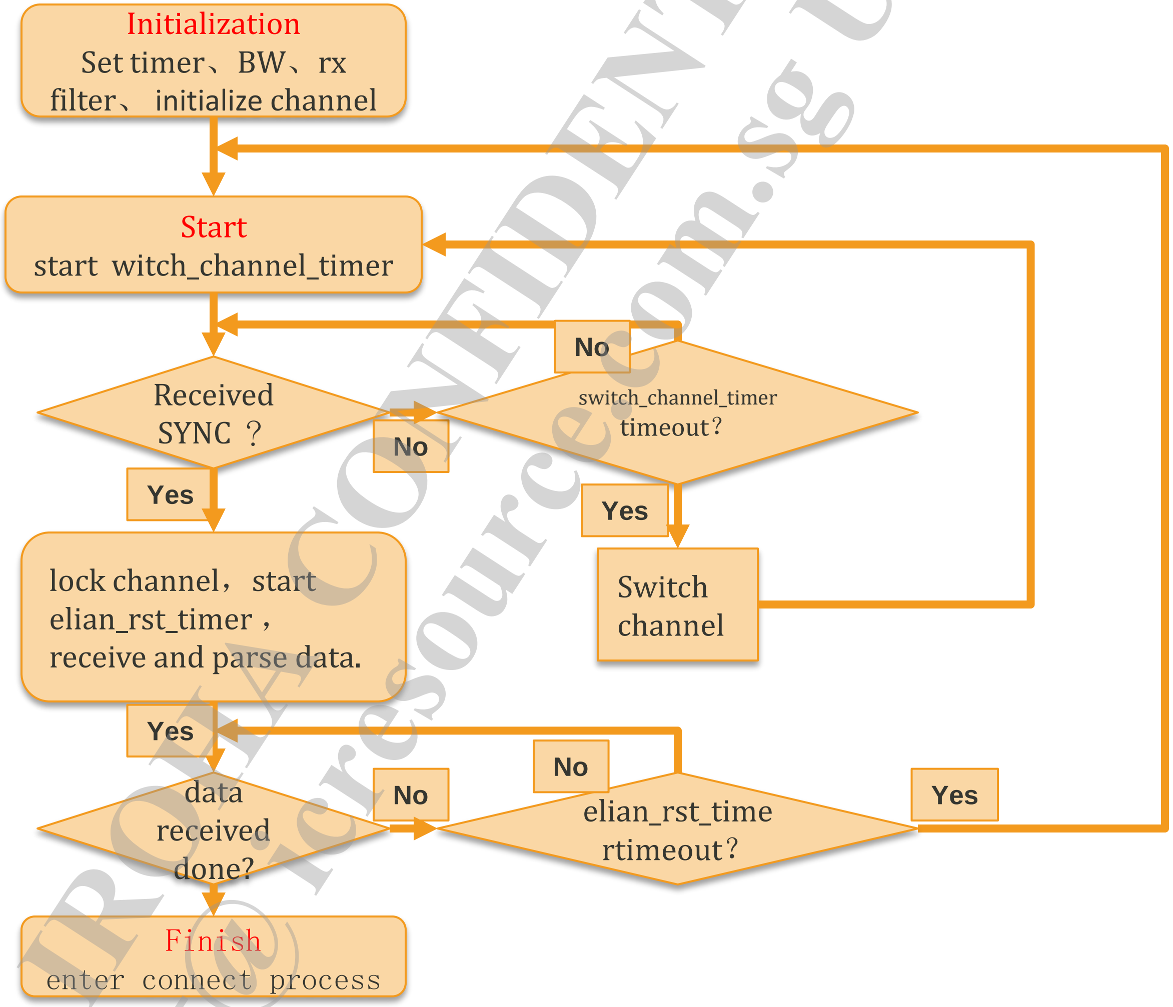
- elian_input
- elian_init
- elian_stop
- elian_rst

- Initialize timer
- Switch channel
- Lock channel

- Adapter
- smtcn_proto_ops
- efunc_table

- Decoding data
- Get information
- Binary release

Code Flow



Initialization
Set timer、BW、rx filter、 initialize channel

Start
start witch_channel_timer

Received SYNC ？

Yes

No

switch_channel_timer timeout？

No

Yes

Switch channel

lock channel， start elian_rst_timer， receive and parse data.

Yes

data received done?

No

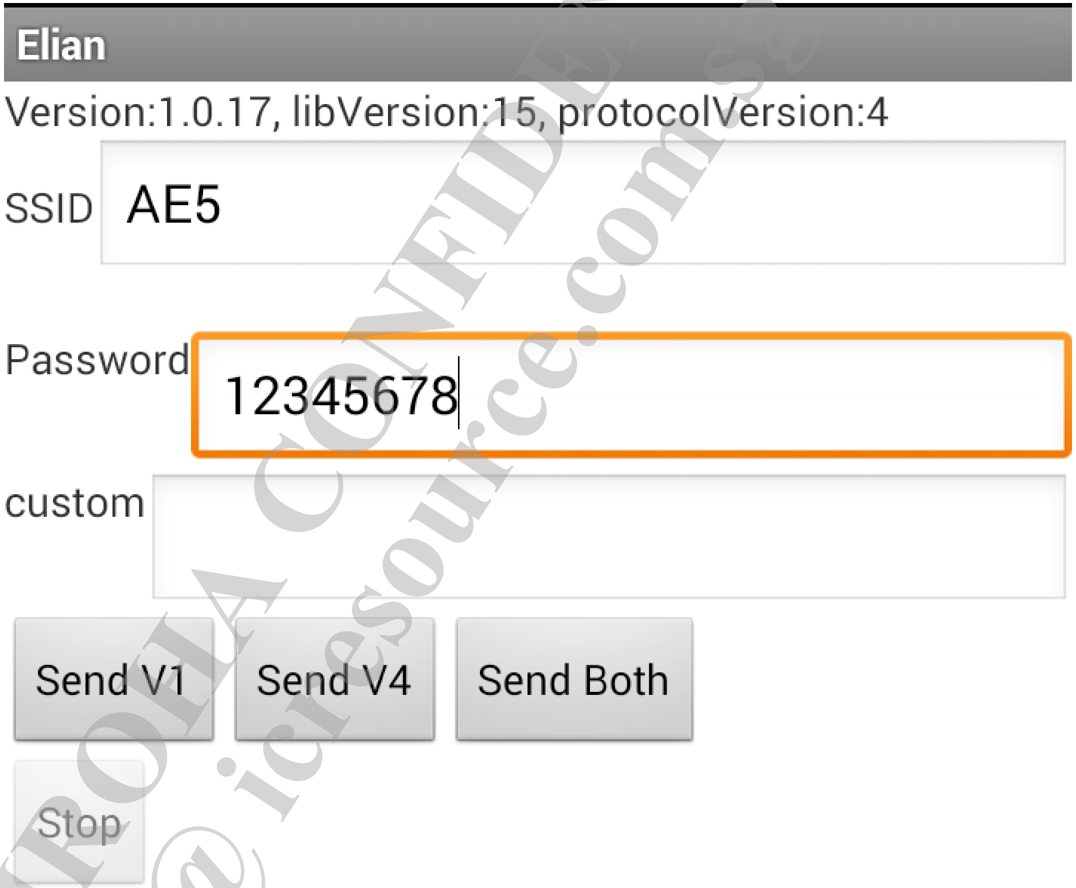
elian_rst_time rtimeout？

No

Yes

Finish
enter connect process

Demo – phone with Elian


Elian

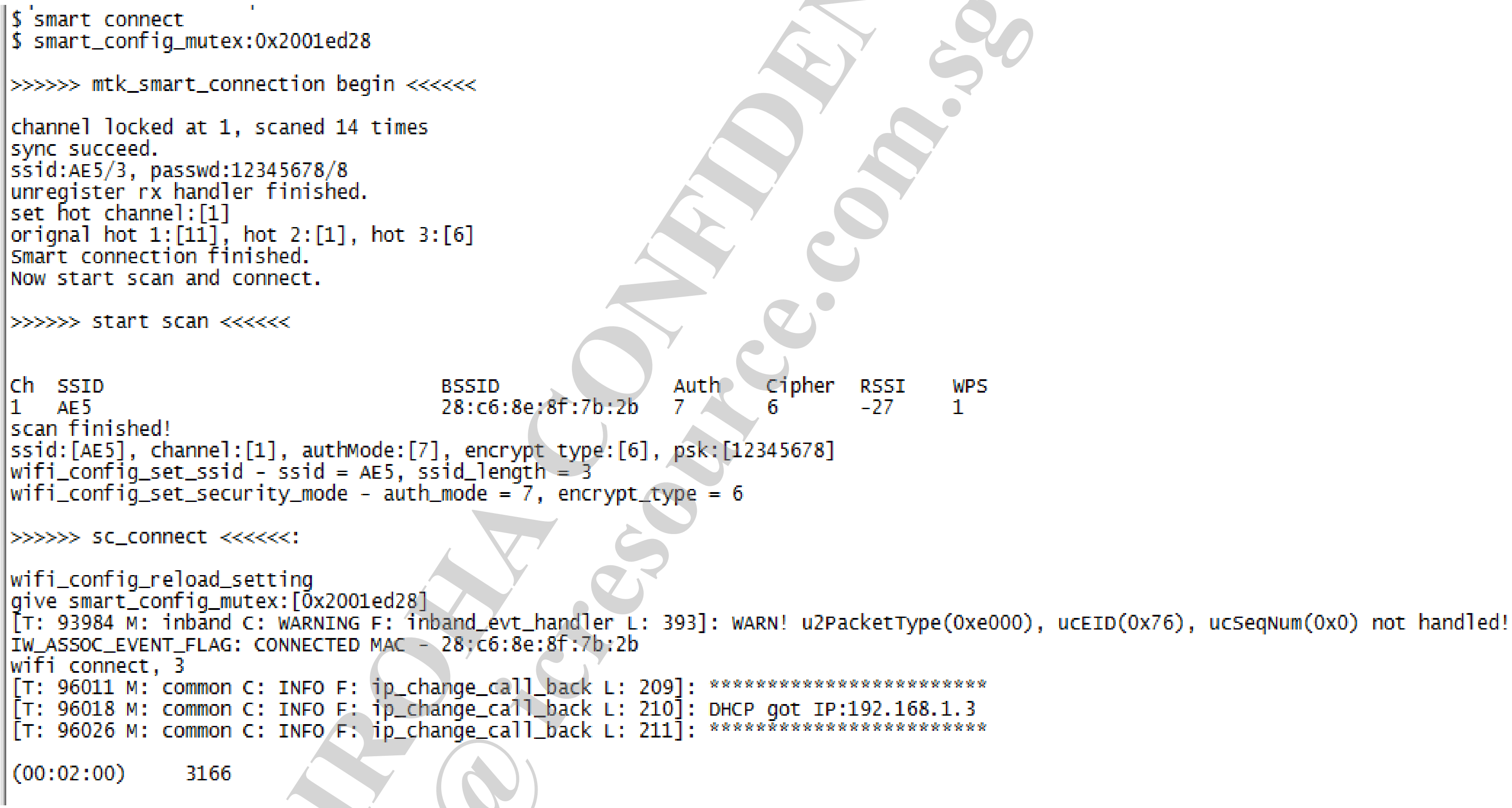
Version:1.0.17, libVersion:15, protocolVersion:4

SSID AE5

Password 12345678

custom

Demo – IOT device

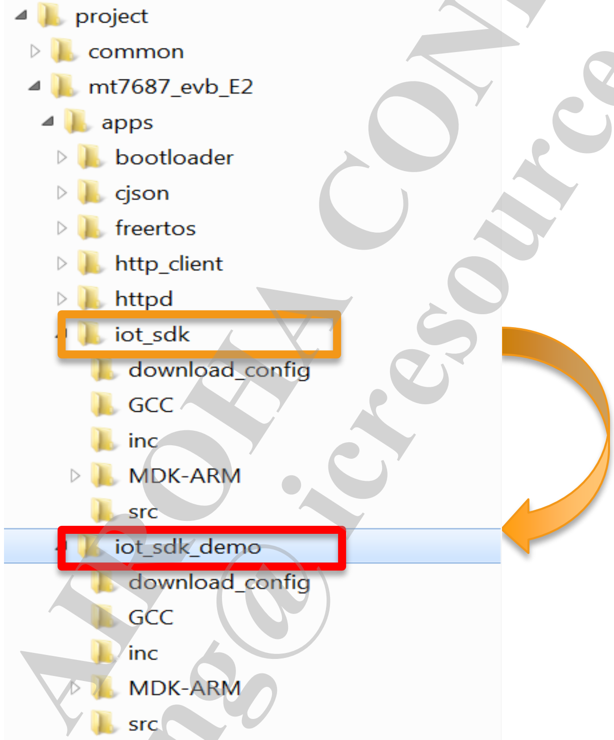


```
$ smart connect
$ smart_config_mutex:0x2001ed28

>>>>>> mtk_smart_connection begin <<<<<<

channel locked at 1, scaned 14 times
sync succeed.
ssid:AE5/3, passwd:12345678/8
unregister rx handler finished.
set hot channel:[1]
orignal hot 1:[11], hot 2:[1], hot 3:[6]
Smart connection finished.
Now start scan and connect.

>>>>>> start scan <<<<<<


Ch  SSID                                BSSID               Auth    Cipher  RSSI    WPS
1   AE5                                 28:c6:8e:8f:7b:2b   7       6       -27     1
scan finished!
ssid:[AE5], channel:[1], authMode:[7], encrypt type:[6], psk:[12345678]
wifi_config_set_ssid - ssid = AE5, ssid_length = 3
wifi_config_set_security_mode - auth_mode = 7, encrypt_type = 6

>>>>>> sc_connect <<<<<<:

wifi_config_reload_setting
give smart_config_mutex:[0x2001ed28]
[T: 93984 M: inband C: WARNING F: inband_evt_handler L: 393]: WARN! u2PacketType(0xe000), ucEID(0x76), ucSeqNum(0x0) not handled!
IW_ASSOC_EVENT_FLAG: CONNECTED MAC - 28:c6:8e:8f:7b:2b
wifi connect, 3
[T: 96011 M: common C: INFO F: ip_change_call_back L: 209]: ************************
[T: 96018 M: common C: INFO F: ip_change_call_back L: 210]: DHCP got IP:192.168.1.3
[T: 96026 M: common C: INFO F: ip_change_call_back L: 211]: ************************

(00:02:00)     3166
```

Create your own project

How to create a project (1/5)

- Here we showing how to create a project named iot_sdk_demo on board mt7687_evb_E2
- Create a folder under project/mt7687_evb_E2/apps/ named iot_sdk_demo
- Copy download_config, GCC, inc, MDK-ARM, and src folder under project/mt7687_evb_E2/apps/iot_sdk to your project folder

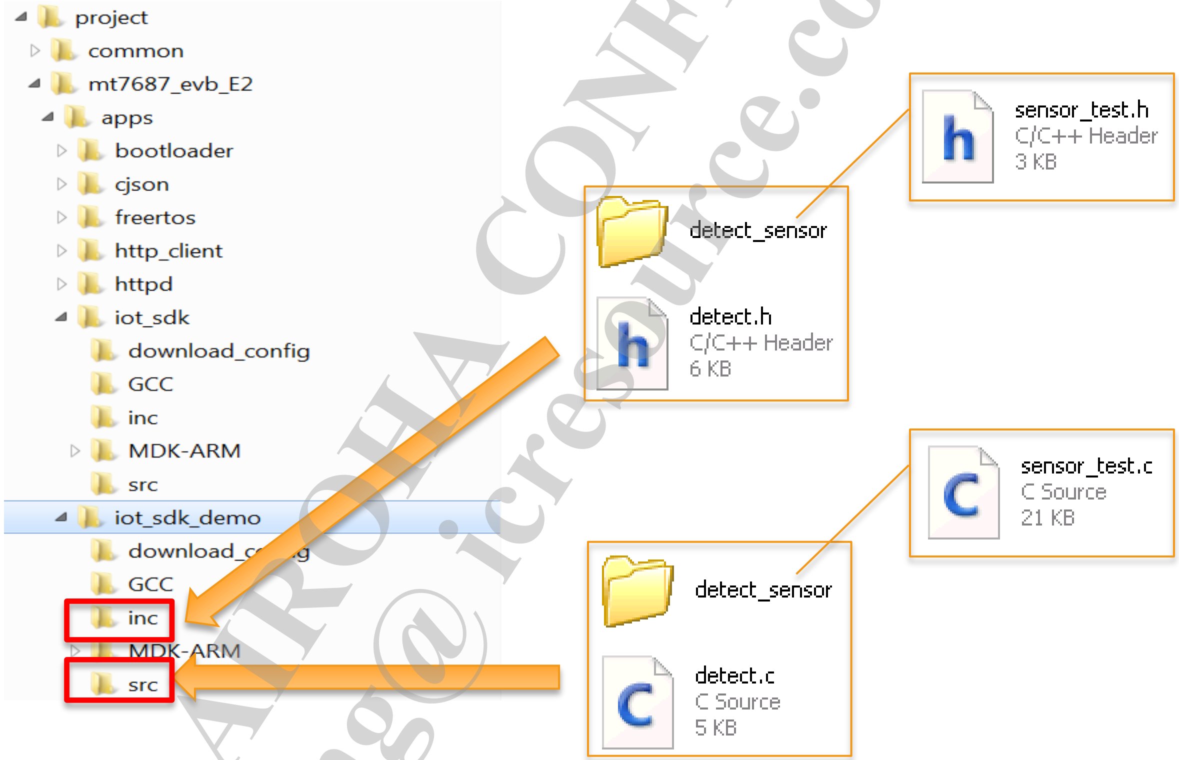


```
◢ project
  ▷ common
  ◢ mt7687_evb_E2
    ◢ apps
      ▷ bootloader
      ▷ cjson
      ▷ freertos
      ▷ http_client
      ▷ httpd
      ◢ iot_sdk
          download_config
          GCC
          inc
        ▷ MDK-ARM
          src
      ◢ iot_sdk_demo
          download_config
          GCC
          inc
        ▷ MDK-ARM
          src
```

How to create a project (2/5)

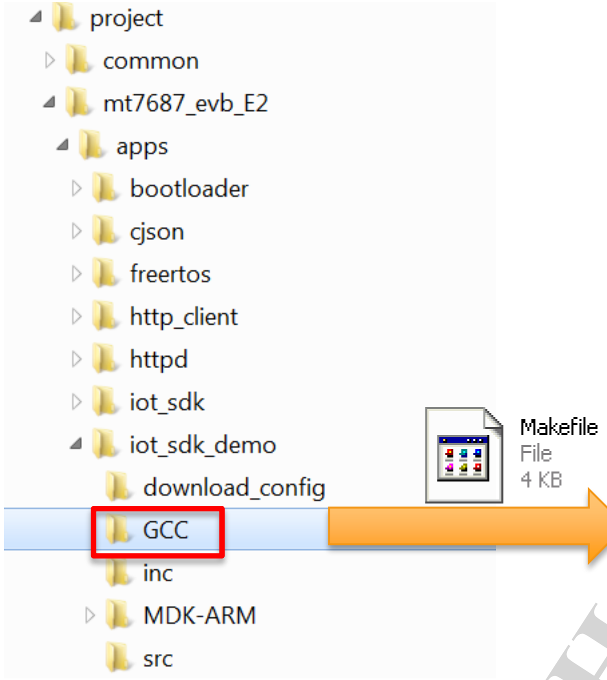
- Add your project files
  - inc folder: header files
  - src folder: source files

P.S. you can create your own folder under inc & src if needed



How to create a project (3/5)

- Modify the red colored part in Makefile to your own



gva/project/mt7687_evb_E2/apps/iot_sdk_demo/Makefile

```
....
SOURCE_DIR = ../../../../..
BINPATH = $(SOURCE_DIR)/tools/gcc/gcc-arm-none-eabi/bin
....
BUILD_DIR = $(PWD)/Build
# Project name
PROJ_NAME = iot_sdk_demo
PROJ_PATH = $(PWD)
....
# Main APP files
APP_PATH = project/mt7687_evb_E2/apps/iot_sdk_demo
APP_PATH_SRC = $(APP_PATH)/src

APP_FILES = $(APP_PATH_SRC)/main.c \
                        $(APP_PATH_SRC)/cli_def.c \
                        $(APP_PATH_SRC)/io_def.c \
                        $(APP_PATH_SRC)/task_def.c \
                        $(APP_PATH_SRC)/net_init.c \
                        $(APP_PATH)/GCC/syscalls.c
                        $(APP_PATH_SRC)/detect.c \
                        $(APP_PATH_SRC)/detect_sensor/sensor_test.c
....
##################################################
# include path
CFLAGS += -I$(SOURCE_DIR)/$(APP_PATH)/inc
CFLAGS += -I$(SOURCE_DIR)/driver/chip/mt7687/inc
CFLAGS += -I$(SOURCE_DIR)/driver/chip/inc
CFLAGS += -I$(SOURCE_DIR)/middleware/lwip/src/include
CFLAGS += -I$(SOURCE_DIR)/middleware/lwip/ports/include
CFLAGS += -I$(SOURCE_DIR)/$(APP_PATH)/inc/detect_sensor
##################################################
```

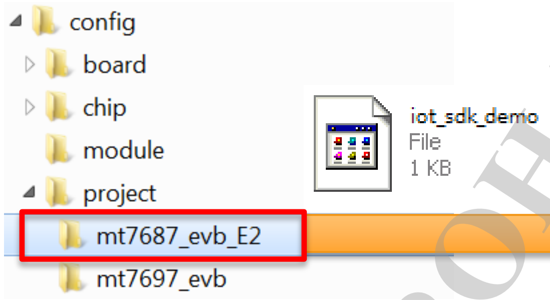
Root directory (gva)

Add file path of your project source files here

Add include path of your project header files here

How to create a project (4/5)

- Add a project config file named iot_sdk_demo to config/project/mt7687_evb_E2 folder
- Copy the example below & paste into iot_sdk_demo
- modify the red colored part to your own

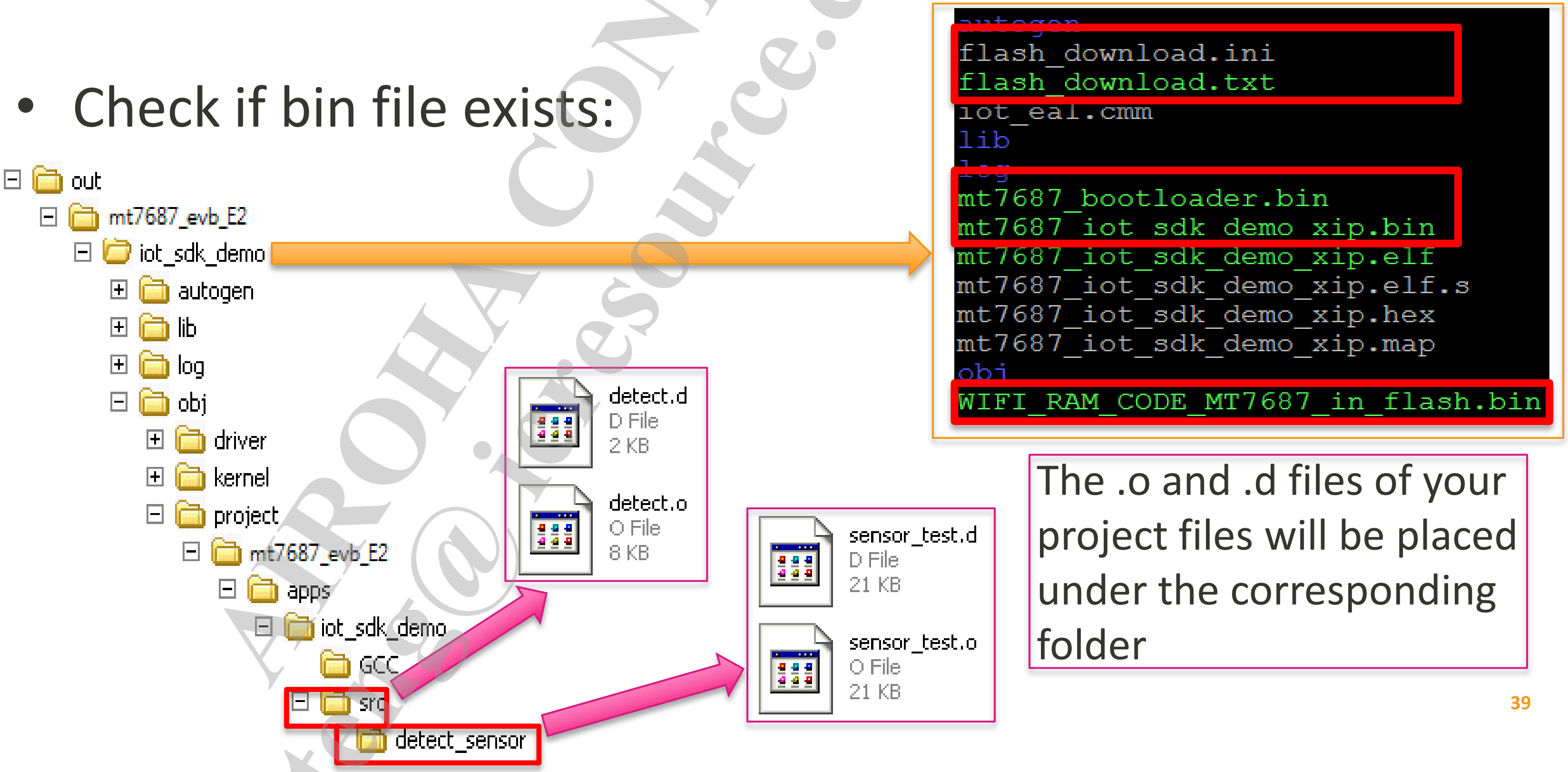


gva/config/project/mt7687_evb_E2/iot_sdk_demo

```
#!/bin/bash

echo "Config...iot_sdk_demo"
export TARGET_PATH=project/mt7687_evb_E2/apps/iot_sdk_demo/GCC
export FREERTOS_CONFIG_PATH=$PWD/project/mt7687_evb_E2/apps/iot_sdk_demo/inc
```

How to create a project (5/5)

- Now you can build your project and see if the bin file of your project generated successfully
  - Build project:
    - ./build.sh mt7687_evb_E2 iot_sdk_demo
  - Check if bin file exists:



```
autogen
flash_download.ini
flash_download.txt
iot_eal.cmm
lib
log
mt7687_bootloader.bin
mt7687_iot_sdk_demo_xip.bin
mt7687_iot_sdk_demo_xip.elf
mt7687_iot_sdk_demo_xip.elf.s
mt7687_iot_sdk_demo_xip.hex
mt7687_iot_sdk_demo_xip.map
obj
WIFI_RAM_CODE_MT7687_in_flash.bin
```

The .o and .d files of your project files will be placed under the corresponding folder

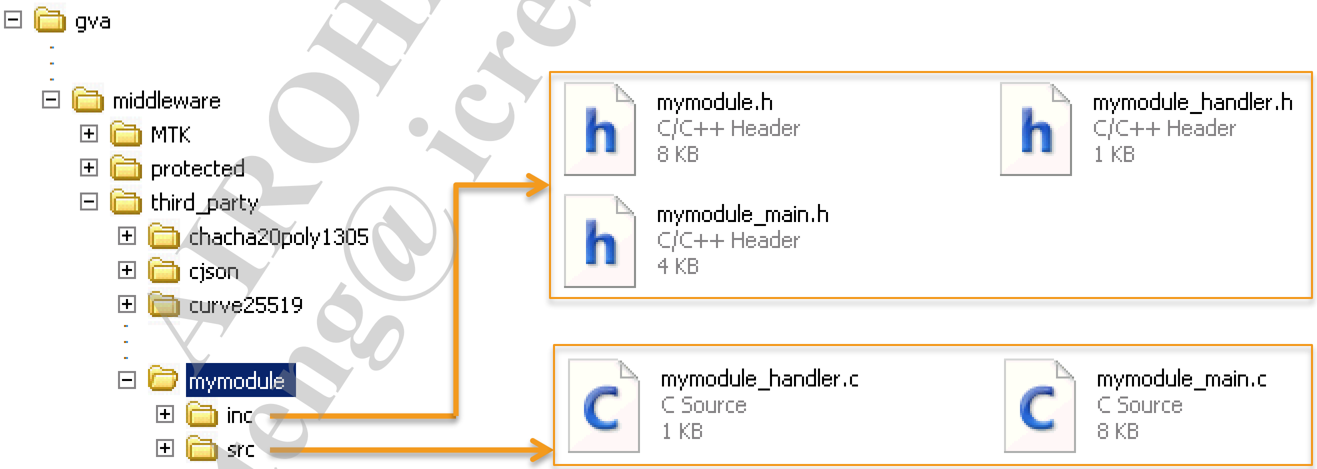
Add a module into Middleware

How to add a module into middleware (1/5)

- Here we showing how to add a module named mymodule into project iot_sdk_demo on board mt7687_evb_E2

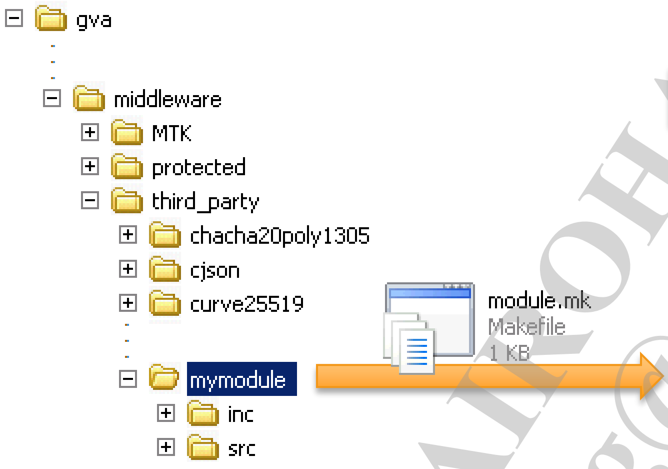
- Create a folder under middleware/third_party named mymodule

- Add your module files
  - inc folder: header files
  - src folder: source files



How to add a module into middleware (2/5)

- Create a file named module.mk under module folder
  - Copy the example below & paste into module.mk
  - modify the red colored part to your own

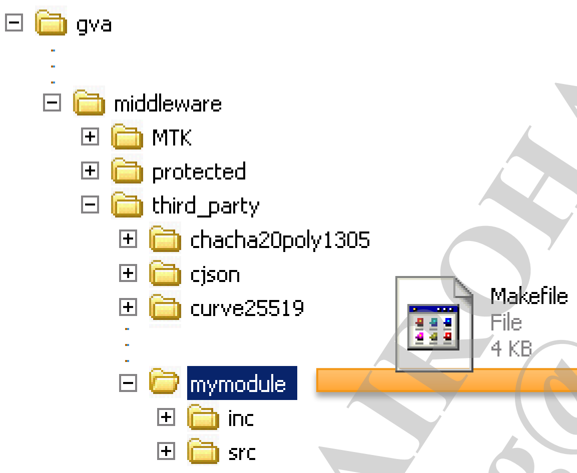


gva/middleware/third_party/mymodule/module.mk

```
#module path
MYMODULE_SRC = middleware/third_party/mymodule

#source file
C_FILES  +=  $(MYMODULE_SRC)/src/mymodule_main.c \
             $(MYMODULE_SRC)/src/mymodule_handler.c
#include path
CFLAGS   += -I$(SOURCE_DIR)/middleware/third_party/mymodule/inc
```

How to add a module into middleware (3/5)

- Create a file named Makefile under module folder
  - Copy the example below & paste into Makefile
  - modify the red colored part to your own



gva/middleware/third_party/mymodule/Makefile

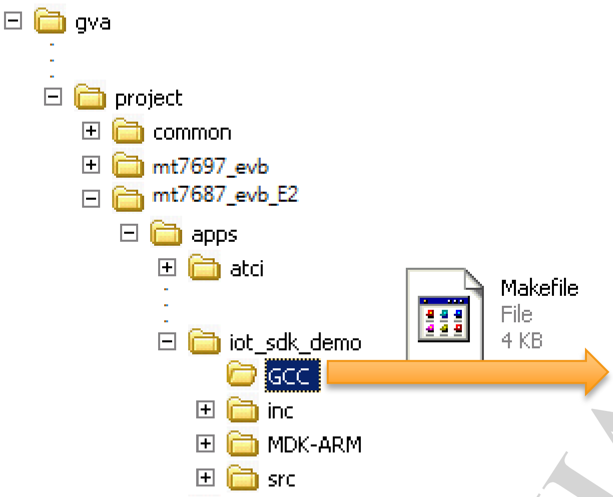
```
SOURCE_DIR = ../../..
BINPATH    = ~/gcc-arm-none-eabi/bin
PROJ_PATH = ../../../project/mt7687_evb_E2/apps/iot_sdk_demo/GCC
CONFIG_PATH ?= .

CFLAGS += -I$(PROJ_PATH)/../inc
CFLAGS += -I$(SOURCE_DIR)/$(CONFIG_PATH)

FEATURE   ?= feature.mk
include $(PROJ_PATH)/$(FEATURE)

# Gloabl Config
-include $(SOURCE_DIR)/.config
# IC Config
-include $(SOURCE_DIR)/config/chip/$(IC_CONFIG)/chip.mk
# Board Config
-include $(SOURCE_DIR)/config/board/$(BOARD_CONFIG)/board.mk

# Project name
TARGET_LIB=libmymodule

BUILD_DIR = Build
OUTPATH = Build


# Sources
include module.mk

C_OBJS = $(C_FILES:%.c=$(BUILD_DIR)/%.o)


.PHONY: $(TARGET_LIB).a

all: $(TARGET_LIB).a
	@echo Build $< Done

include $(SOURCE_DIR)/.rule.mk

clean:
	rm -rf $(OUTPATH)/$(TARGET_LIB).a
	rm -rf $(BUILD_DIR)
```

the path to your project folder that contains project Makefile

the lib name for the added module

How to add a module into middleware (4/5)

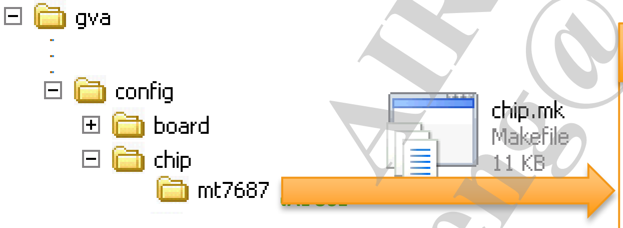
- Add module to linking LIBS & make command in project Makefile



gva/project/mt7687_evb_E2/apps/iot_sdk_demo/Makefile

```
....
# HAL driver files
LIBS += $(OUTPATH)/libbsp.a
LIBS += $(OUTPATH)/libhal.a

LIBS += $(OUTPATH)/libmymodule.a
....
MOD_EXTRA = BUILD_DIR=$(BUILD_DIR) OUTPATH=$(OUTPATH) PROJ_PATH=$(PROJ_PATH)

libs:
        make -C $(DRV_CHIP_PATH) $(MOD_EXTRA)
        make -C $(DRV_BSP_PATH) $(MOD_EXTRA)
        make -C $(MID_MY_MODULE_PATH) $(MOD_EXTRA)

proj: $(OUTPATH)/$(PROJ_NAME).elf
....
```

- Add $(MID_MY_MODULE_PATH) definition to project related chip.mk

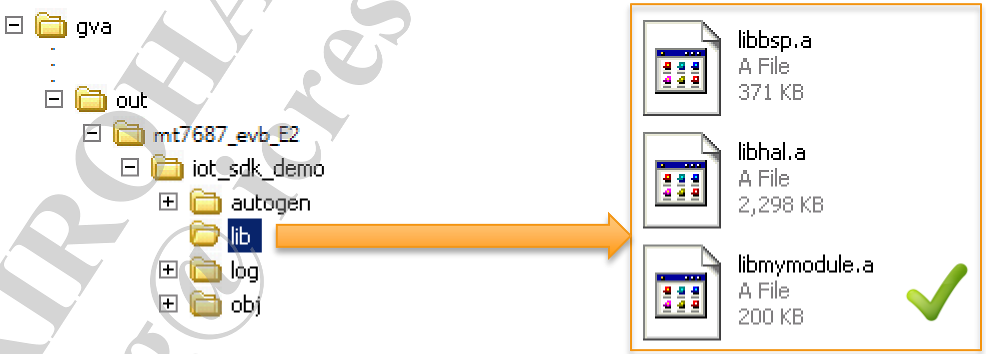


gva/config/chip/mt7687/chip.mk

```
.....
#Middleware Module Path
DRV_BSP_PATH                  = $(SOURCE_DIR)/driver/board/mt76x7_evb
KRL_OS_PATH                   = $(SOURCE_DIR)/kernel/rtos/FreeRTOS
KRL_SRV_PATH                  = $(SOURCE_DIR)/kernel/service
MID_MY_MODULE_PATH            = $(SOURCE_DIR)/middleware/third_party/mymodule
```

How to add a module into middleware (5/5)

- Now you can build your project and see if the lib of the added module generate successfully
  - Build project:
    - ./build.sh mt7687_evb_E2 iot_sdk_demo
  - Check if the lib exists



gva
out
mt7687_evb_E2
iot_sdk_demo
autogen
lib
log
obj

libbsp.a
A File
371 KB

libhal.a
A File
2,298 KB

libmymodule.a
A File
200 KB
✓

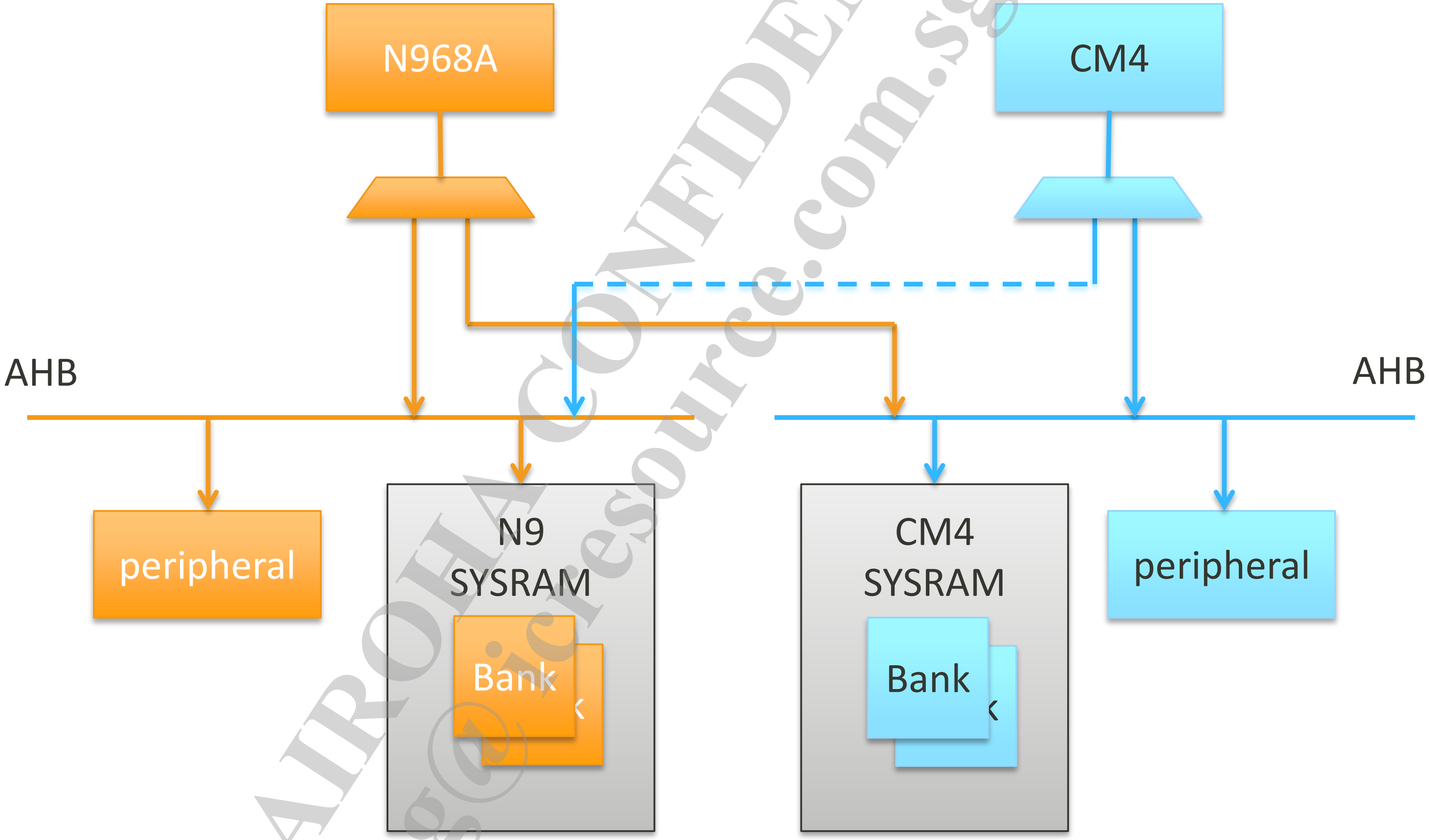
MEDIATEK

everyday genius

Appendix 1

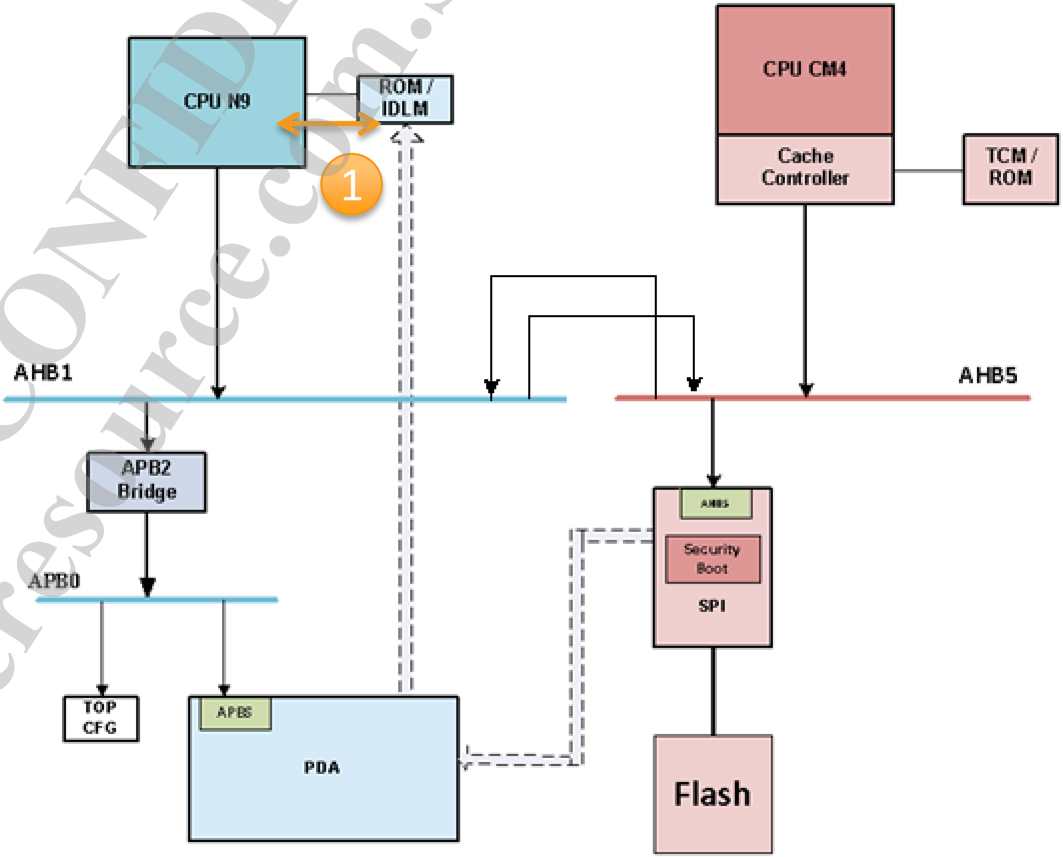
Boot up sequence of MT7687

H/W Architecture (dual core)



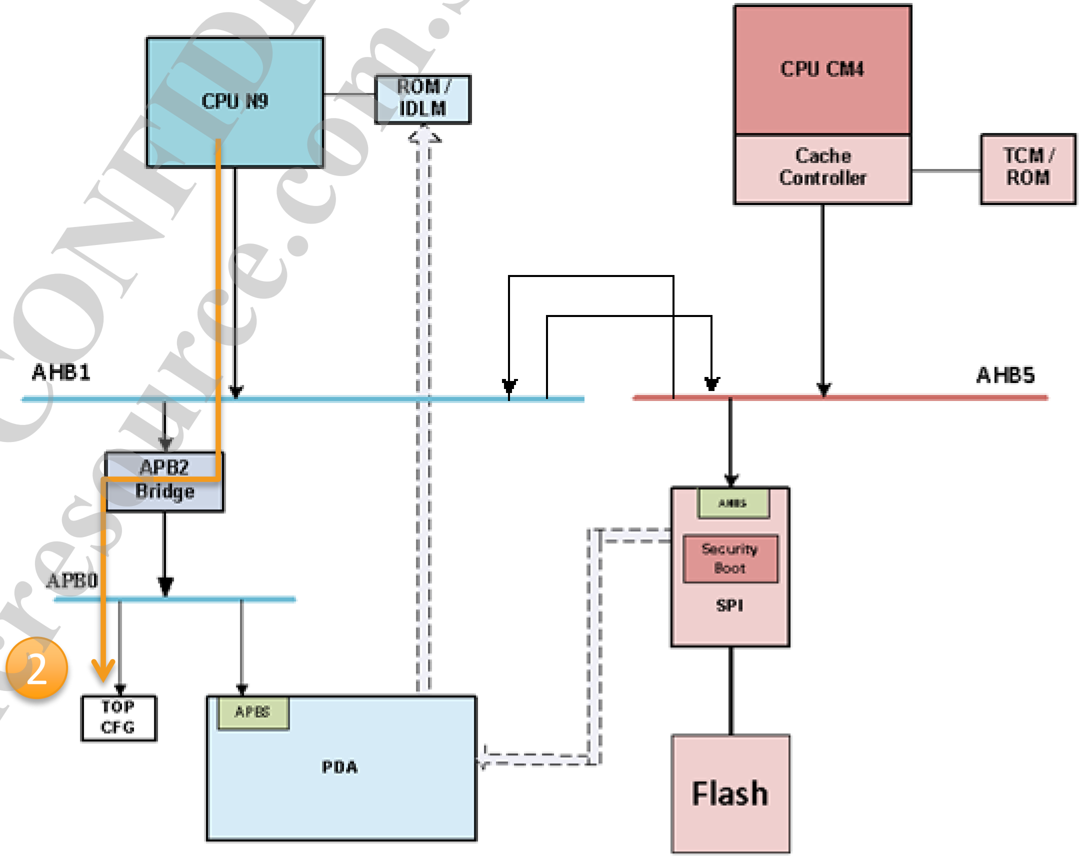
Dual-core platform power sequence

- Step 1:
  - Chip power on
  - N9 :
    - boot from N9-ROM
  - CM4 : reset asserted



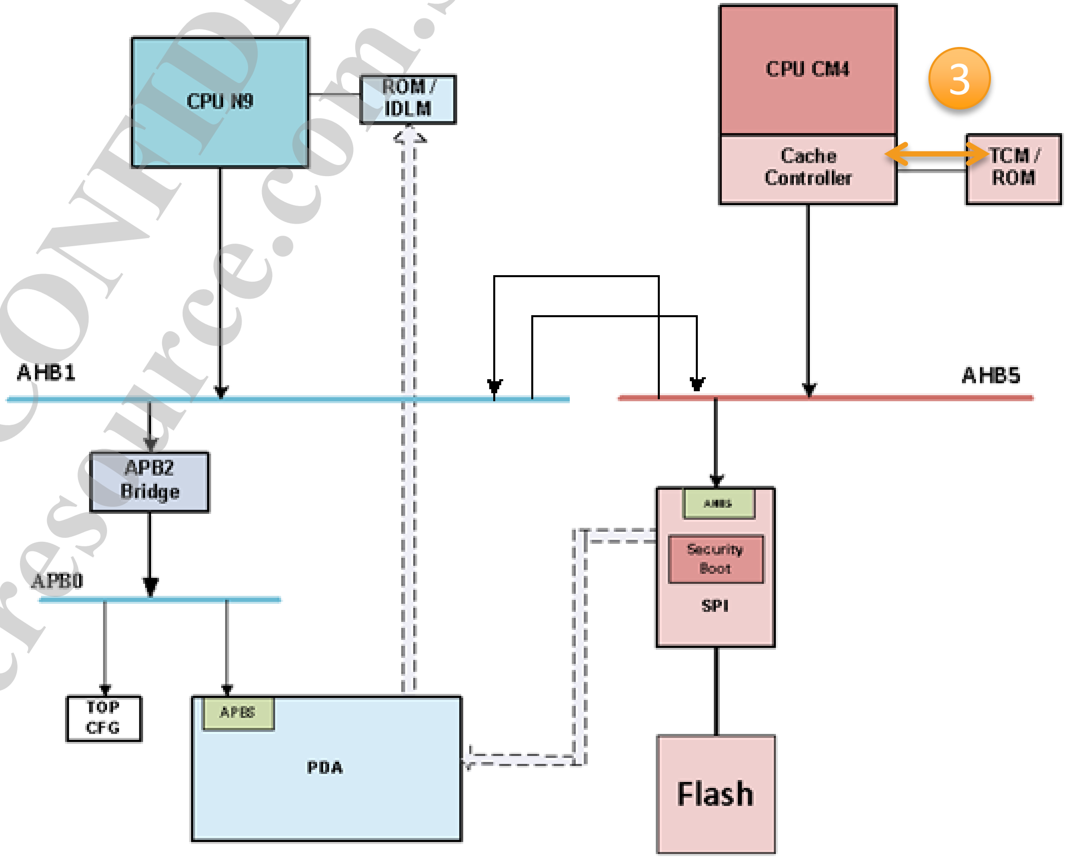
Dual-core platform power sequence

- Step 2:
  - N9 :
    - Setting TOP cfg. registers (PLL setting, etc.)
    - de-assert reset signal of CM4



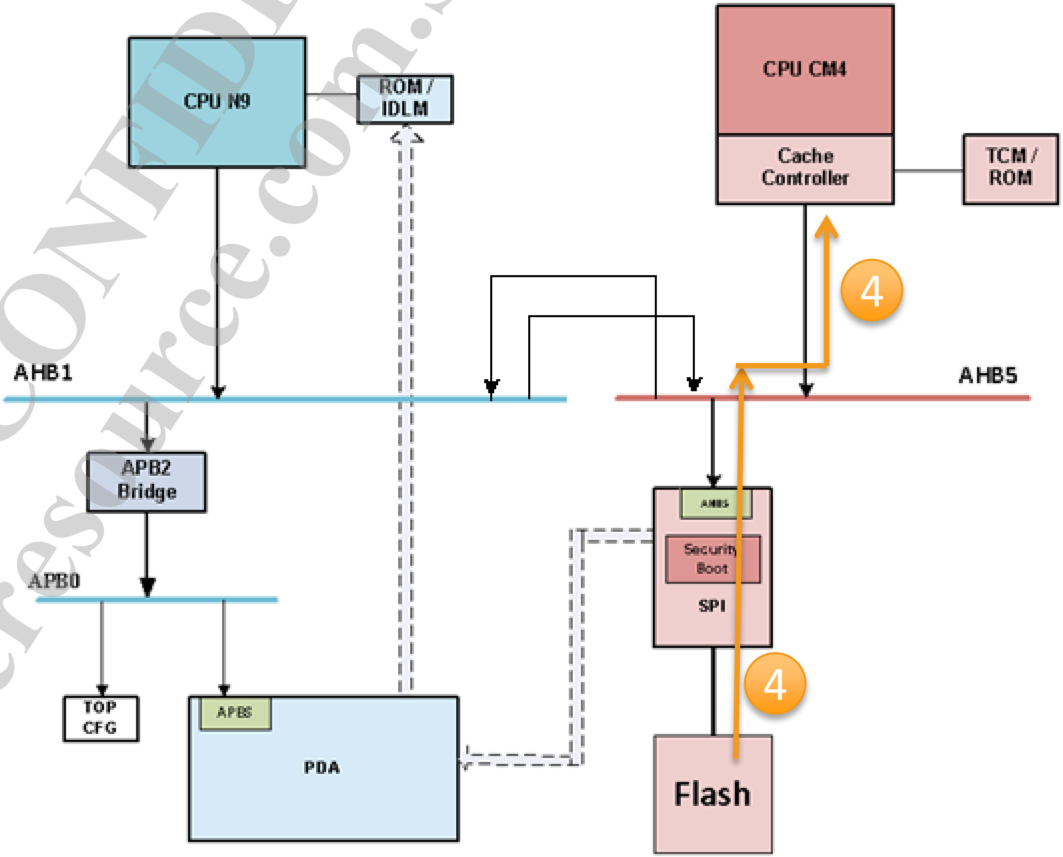
Dual-core platform power sequence

- Step 3:
  - CM4 :
    - boot from CM4-ROM



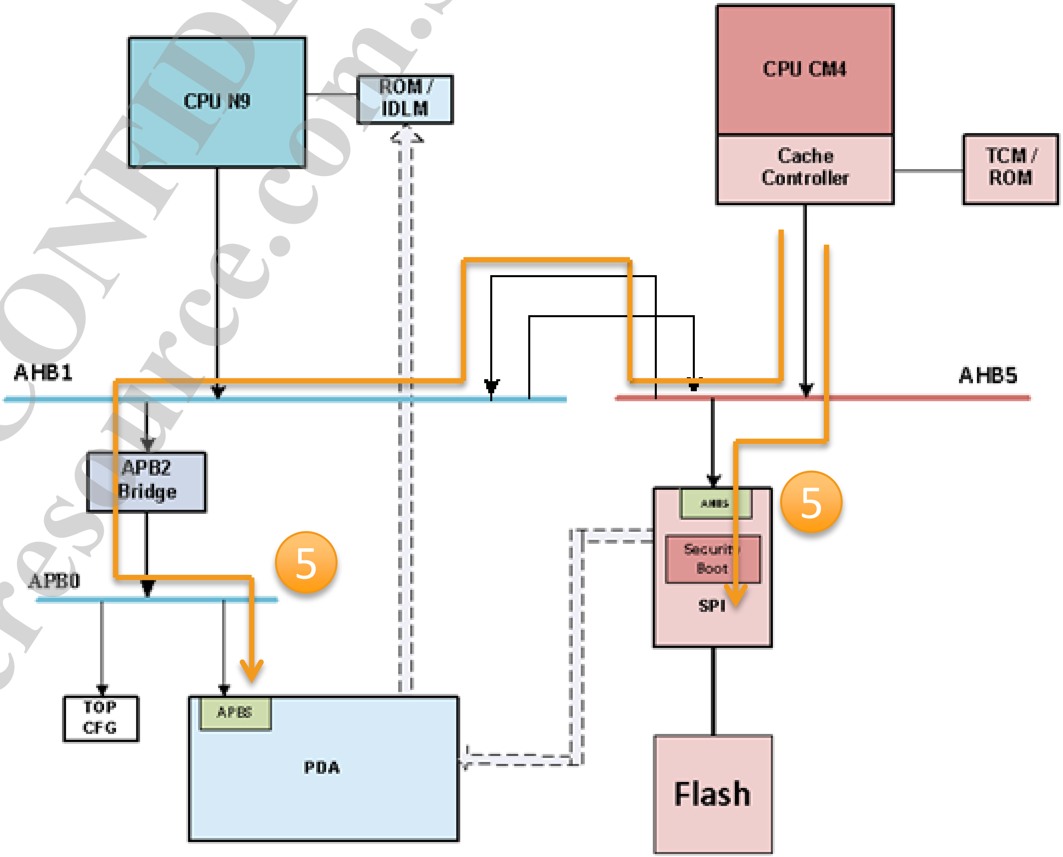
Dual-core platform power sequence

- Step 4:
  - CM4 :
    - get F/W download information of N9 from external Flash



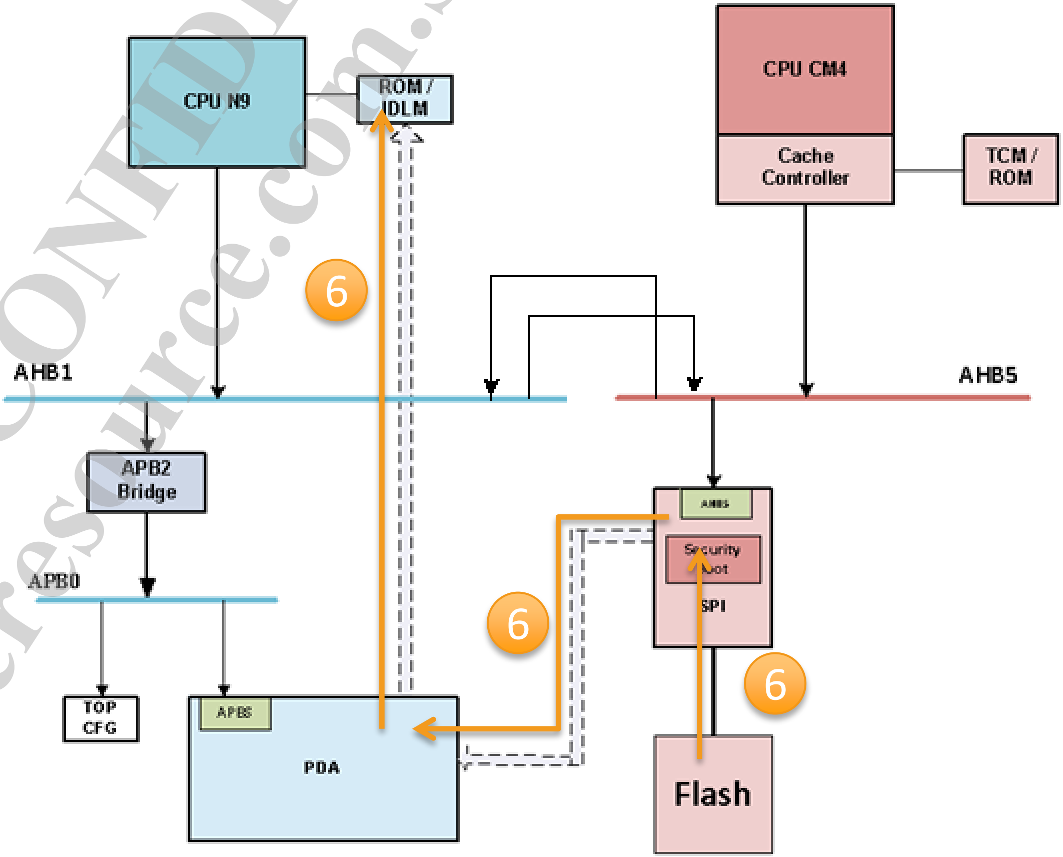
Dual-core platform power sequence

- Step 5:
  - CM4 :
    - set PDA (Patch Decryption Accelerator) cfg. registers for prepare to F/W download
    - set SPI cfg. registers for prepare to F/W download



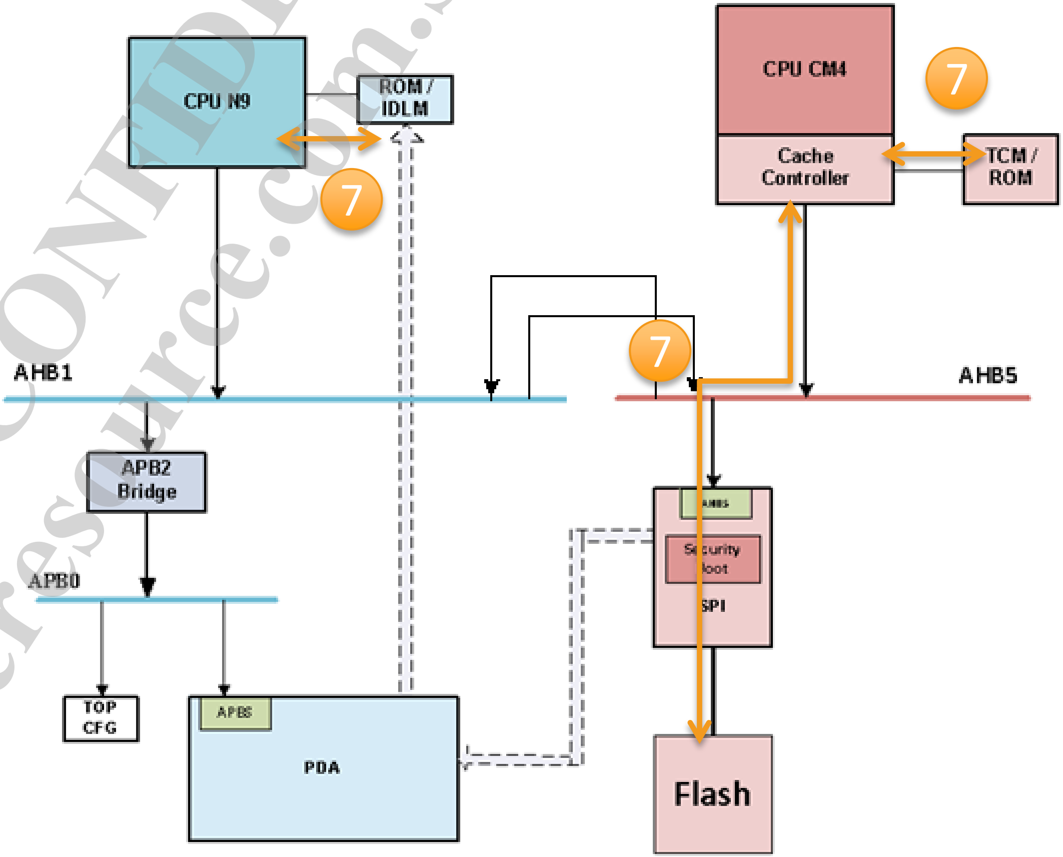
Dual-core platform power sequence

- Step 6:
  - N9 :
    - PDA F/W download, from external flash memory to N9-IDLM.

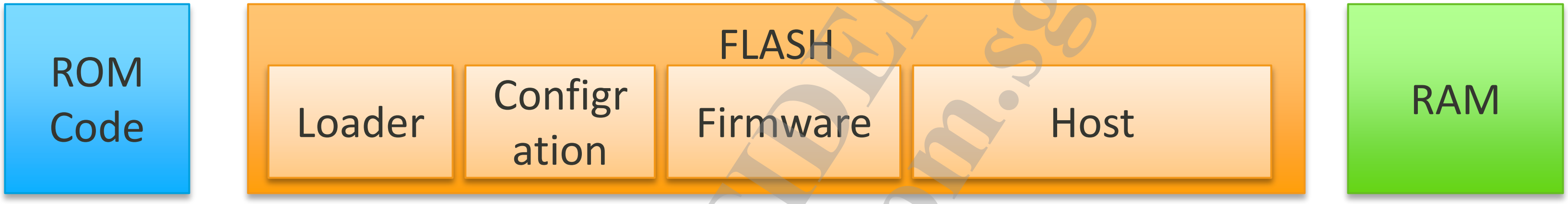


Dual-core platform power sequence

- Step 7:
  - N9 :
    - execute instructions from IDLM
  - CM4 :
    - execute instructions from cache or external flash memory



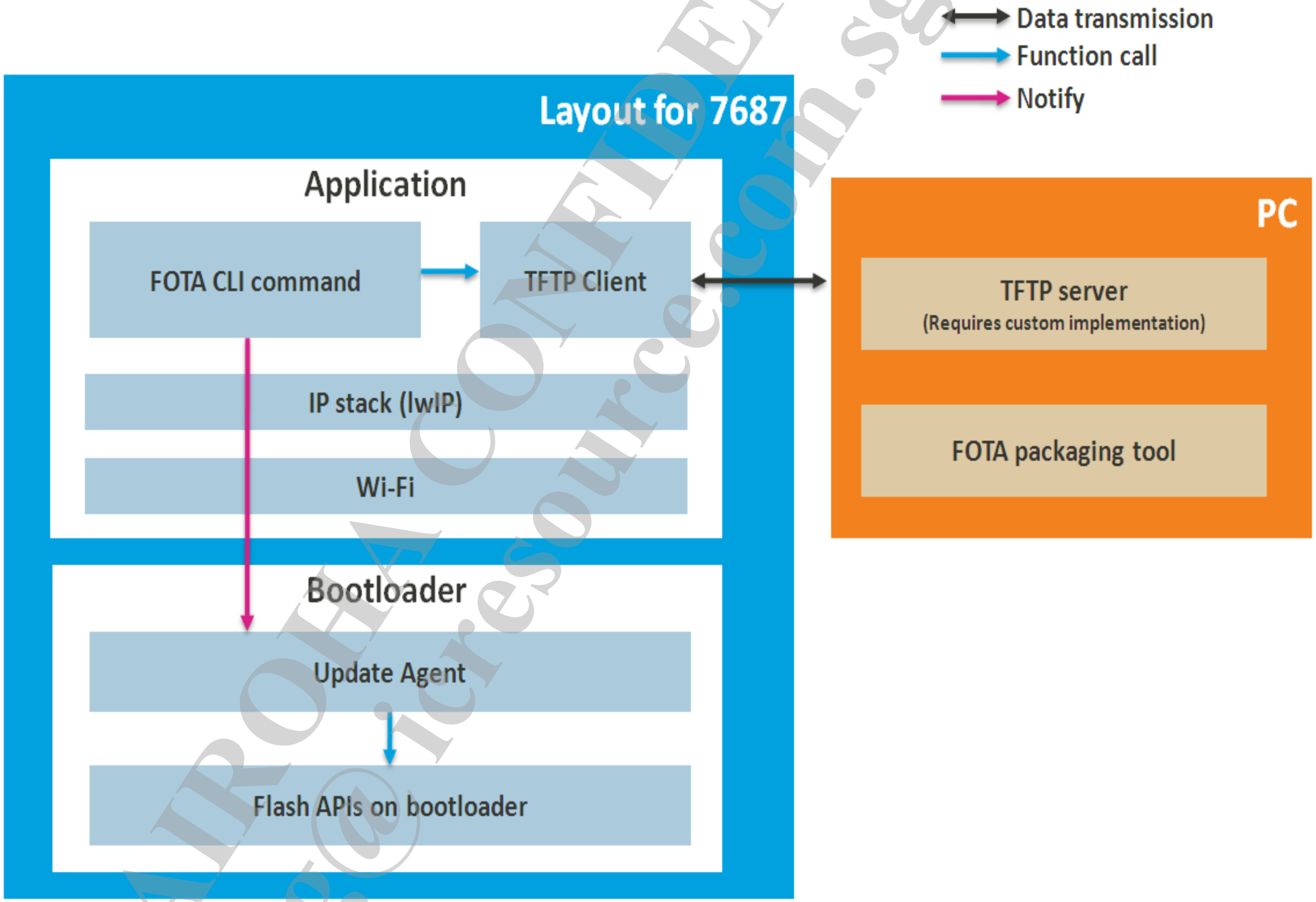
MT7687 Boot-Up Flow



- Step1: ROM Code--- bootstrap check
- Step2: ROM Code--- load loader Image from Flash to RAM
- Step3: Loader --- start run XIP Host driver
- Step4: Host --- load Wi-Fi Firmware to RAM
- Step5: Firmware --- initial Wi-Fi Register start Wi-Fi State Machine
- Step6: Host --- Run Host Task

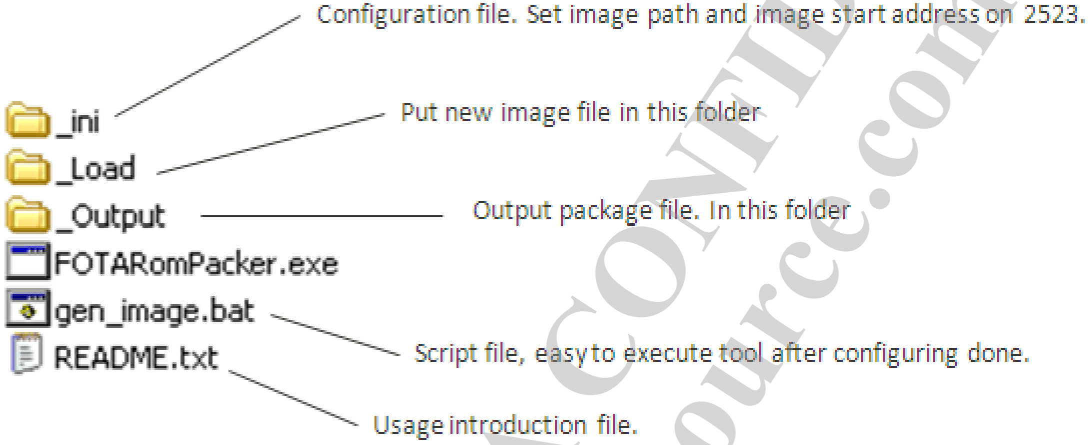
Appendix 2
FOTA of MT7687

FOTA architecture layout



Generate Package File

- FOTARomPacker Tool (Win OS)



_ini — Configuration file. Set image path and image start address on 2523.

_Load — Put new image file in this folder

_Output — Output package file. In this folder

FOTARomPacker.exe

gen_image.bat — Script file, easy to execute tool after configuring done.

README.txt — Usage introduction file.

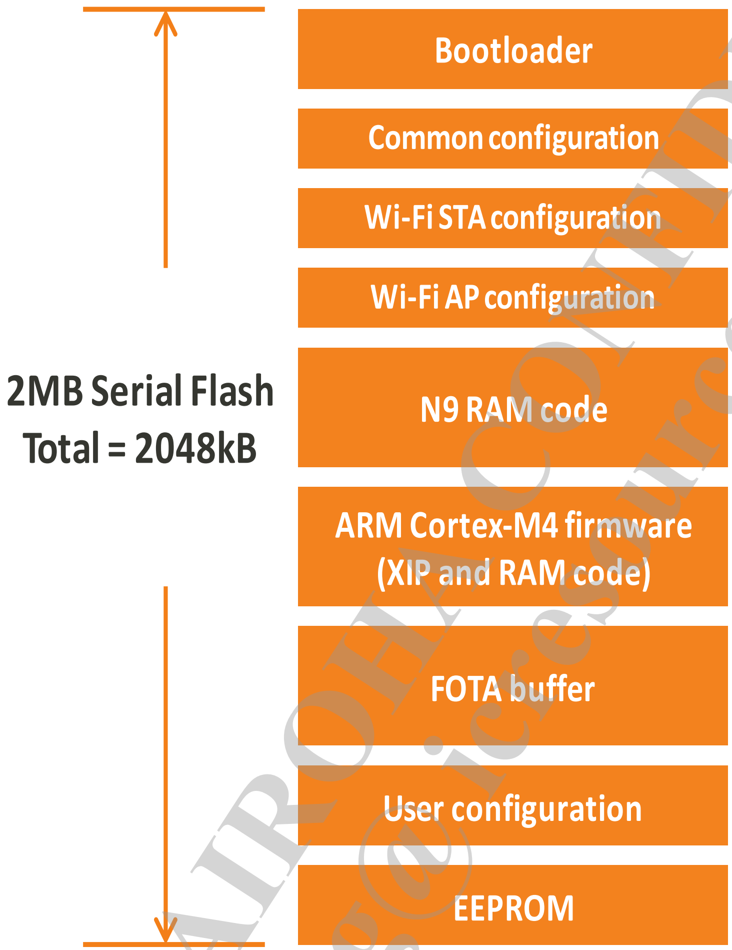
Path: gva/tools/fota

- Follow these steps as below:
  - 1. Put new bin in "_Load" folder.
  - 2. Configure FOTARomPacker.ini file in "_ini" folder.
    - a) set load path in "general setting" line.
    - b) set bin file name and the start address of this bin in flash. (For7687, CM4 start addr is 0x7c000)
  - 3. Click "gen_image.bat", find the generated FOTA package file in "_Output folder".

Compile options for the FOTA update

Option	Description	Build the ARM Cortex-M4	Build the bootloader
MTK_FOTA_ENABLE	FOTA module on/off	=y	=y
MTK_TFTP_ENABLE	TFTP module on/off	=y	=n
MTK_FOTA_CLI_ENABLE	Command line on/off	=y	=n
MTK_BL_FOTA_LOG_ENALBE	Bootloader logging on/off	=n	=y
MTK_HAL_PLAIN_LOG_ENABLE	HAL debug	=n	=y

FOTA buffer partition configuration



2MB Serial Flash
Total = 2048kB

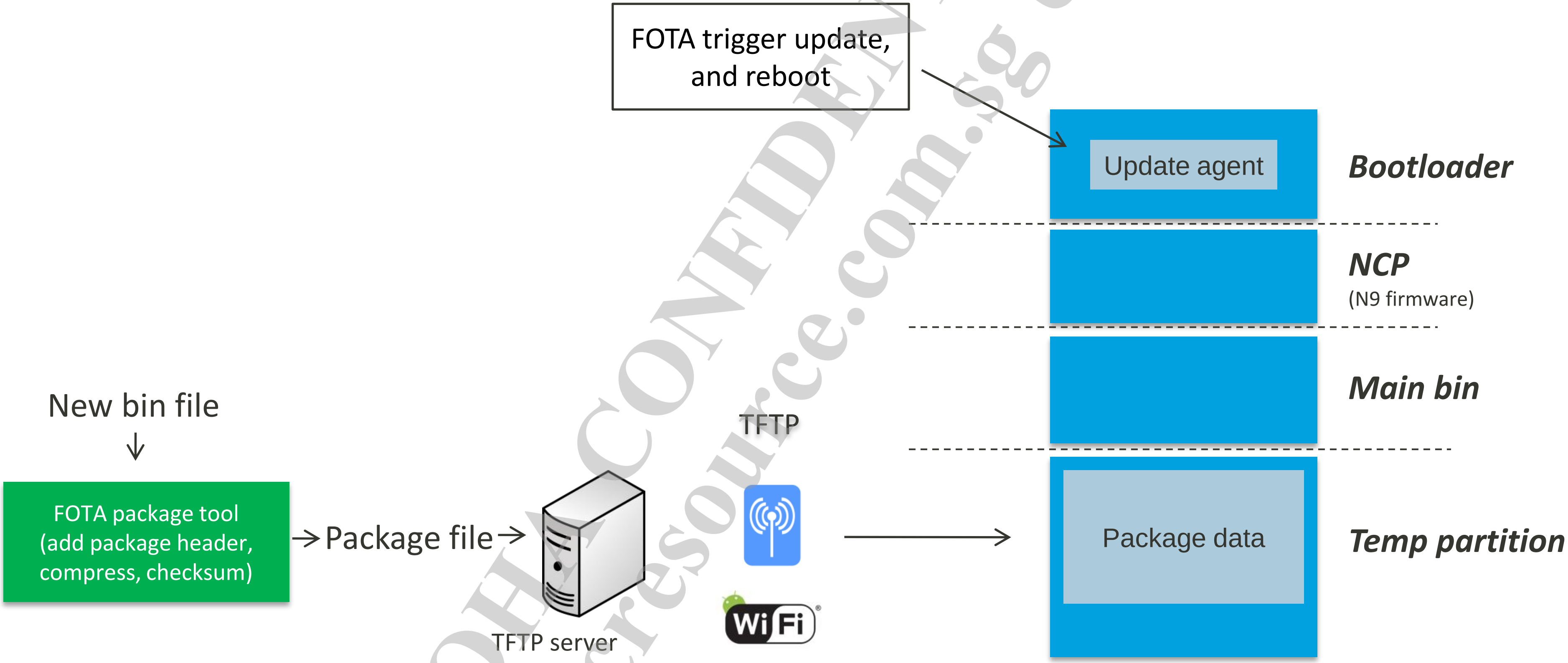
Bootloader	32kB, this size is extendable.
Common configuration	4kB, includes the FOTA settings.
Wi-Fi STA configuration	4kB
Wi-Fi AP configuration	4kB
N9 RAM code	452kB, includes the patch size of 64kB and N9 Wi-Fi RAM of size 388kB.
ARM Cortex-M4 firmware (XIP and RAM code)	764kB, includes BSP and Customer SDK of 500kB
FOTA buffer	764kB
User configuration	20kB, for custom configuration
EEPROM	4kB

Change the start address and length of the FOTA buffer

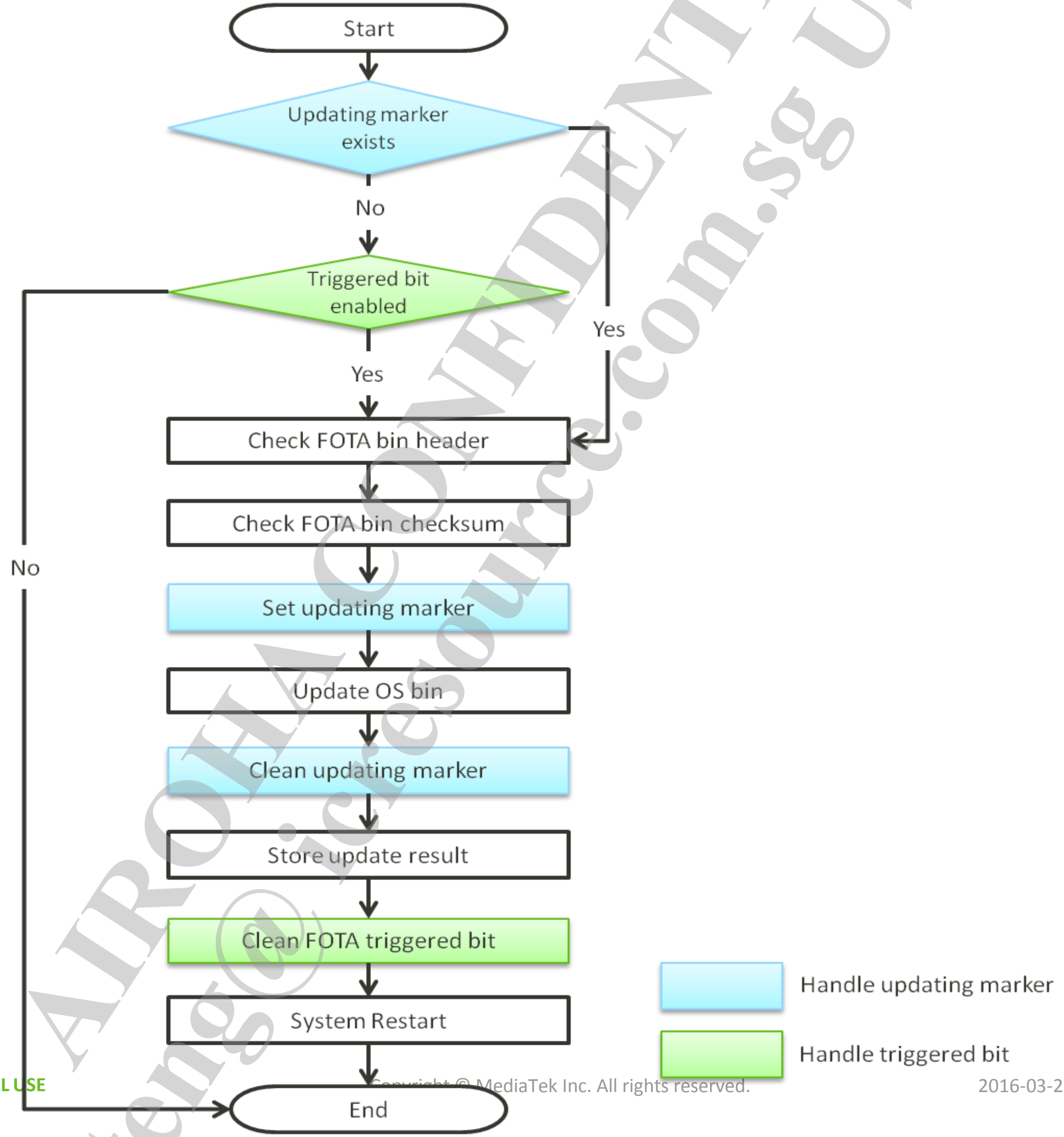
- driver\chip\mt7687\inc\flash_map.h

```
#define FLASH_LOADER_SIZE           0x8000        /*  32kB */
#define FLASH_COMM_CONF_SIZE        0x1000        /*   4kB */
#define FLASH_STA_CONF_SIZE         0x1000        /*   4kB */
#define FLASH_AP_CONF_SIZE          0x1000        /*   4kB */
#define FLASH_N9_RAM_CODE_SIZE      0x71000       /* 452kB */
#define FLASH_CM4_XIP_CODE_SIZE     0xBF000       /* 764kB */
#define FLASH_TMP_SIZE              0xBF000       /* 764kB */
#define FLASH_USR_CONF_SIZE         0x5000        /*  20kB */
#define FLASH_EEPROM_SIZE           0x1000        /*   4kB */
#define CM4_FLASH_LOADER_ADDR       0x0
#define CM4_FLASH_COMM_CONF_ADDR    (CM4_FLASH_LOADER_ADDR     + FLASH_LOADER_SIZE)
#define CM4_FLASH_STA_CONF_ADDR     (CM4_FLASH_COMM_CONF_ADDR  + FLASH_COMM_CONF_SIZE)
#define CM4_FLASH_AP_CONF_ADDR      (CM4_FLASH_STA_CONF_ADDR   + FLASH_STA_CONF_SIZE)
#define CM4_FLASH_N9_RAMCODE_ADDR   (CM4_FLASH_AP_CONF_ADDR    + FLASH_AP_CONF_SIZE)
#define CM4_FLASH_CM4_ADDR          (CM4_FLASH_N9_RAMCODE_ADDR + FLASH_N9_RAM_CODE_SIZE)
#define CM4_FLASH_TMP_ADDR          (CM4_FLASH_CM4_ADDR        + FLASH_CM4_XIP_CODE_SIZE)
#define CM4_FLASH_USR_CONF_ADDR     (CM4_FLASH_TMP_ADDR        + FLASH_USR_CONF_SIZE)
#define CM4_FLASH_EEPROM_ADDR       (CM4_FLASH_USR_CONF_ADDR   + FLASH_EEPROM_SIZE)
```

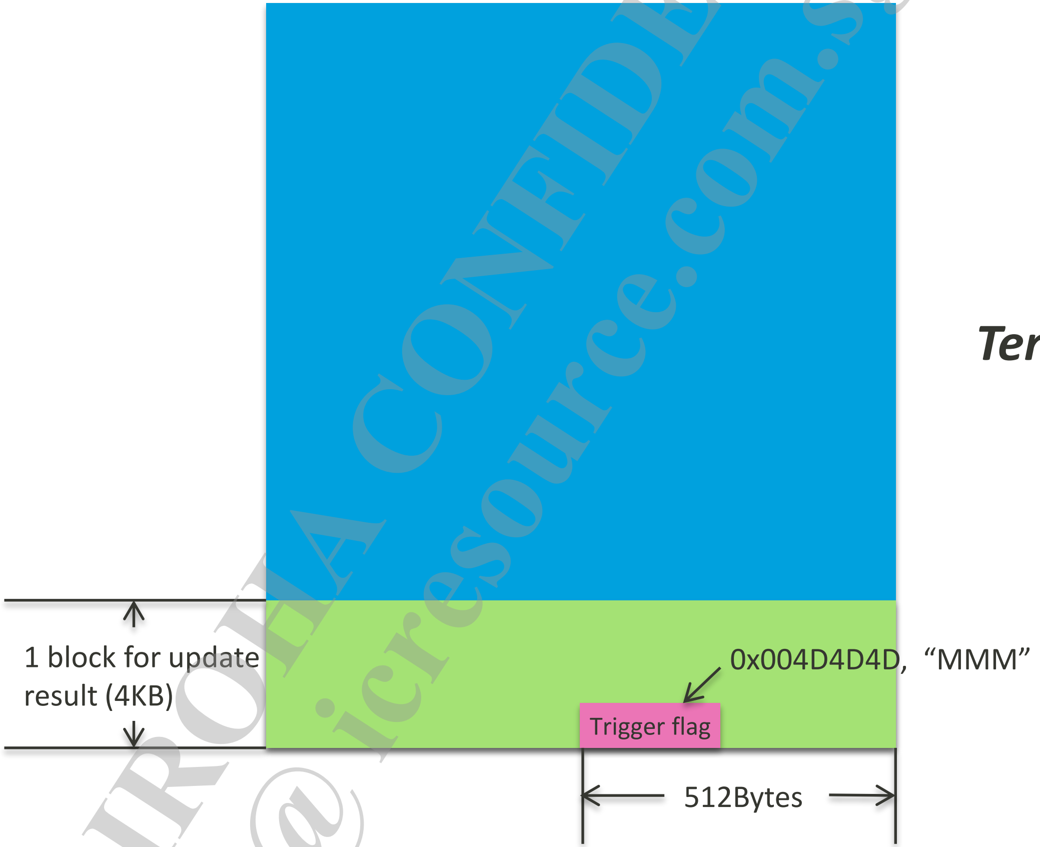
7687 FOTA Scenario



Bootloader Update Flow

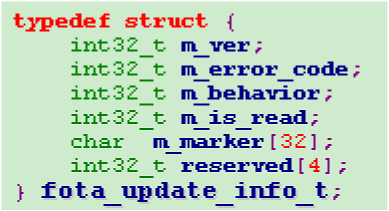


Trigger Flag & Update Result



Temp partition

Update result structure:



```
typedef struct {
    int32_t m_ver;
    int32_t m_error_code;
    int32_t m_behavior;
    int32_t m_is_read;
    char    m_marker[32];
    int32_t reserved[4];
} fota_update_info_t;
```