AI6060H User Manual

Revision History

Revision	Date	Author	Description
V.1	160308	PW	New Create

Contents:

1. Introduction:

AcSip Wi-Fi IOT AI6060H module is dedicated designed for IOT product. The single IOT Wi-Fi chip includes Wi-Fi driver, Embedded OS and TCP/IP stack.

In this document, we will focus on how to install the proper Acsip IOT developing environment

2. Start AI6060H development

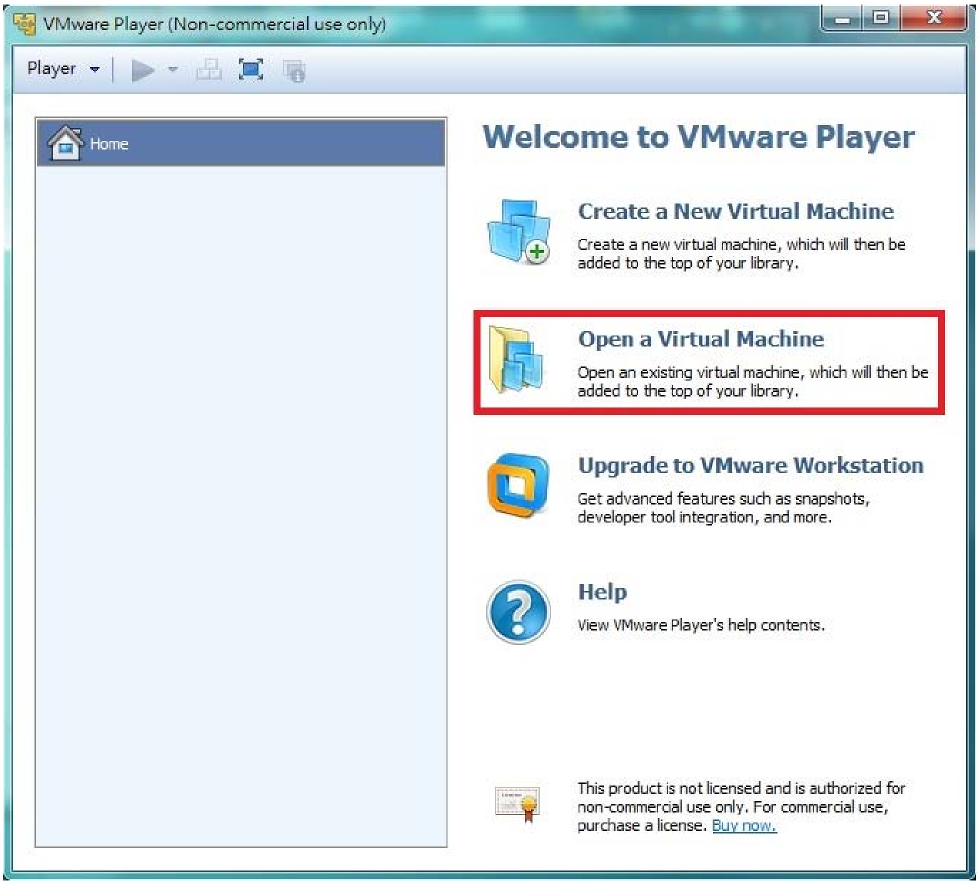
2.1 IDE Setup

The AI6060H firmware is developed under Ubuntu operation system.
The developing environment also shipped together with VMware Virtual Machine.
In order to use this virtual machine, please download VMware player and setup.

(1) Download VMware Player 6
Please reference the web page for download the VMware Player
https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0|PLAYER-605|product_downloads

(2) Prepare VMware Virtual Machine
Unzip the file “InstantContiki2.7-OEM” to local hard drive. This is the developing environment that VMware Player will run with.

(3) Execute Virtual Machine
Execute the VMware Player from “Desktop” or “Program files’’. Select “Open a Virtual Machine” on VMware Player.



Welcome to VMware Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

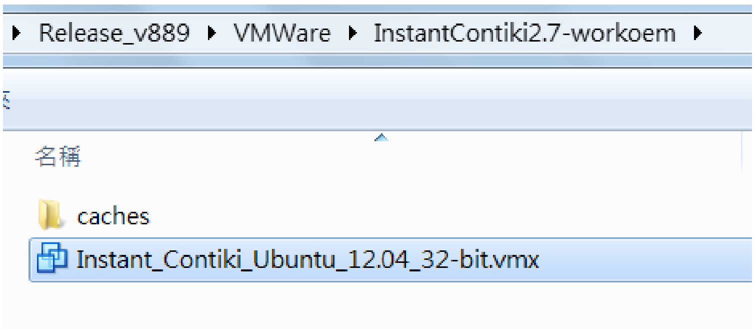
Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation
Get advanced features such as snapshots, developer tool integration, and more.

Help
View VMware Player's help contents.

This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

(4) Open the file folder and choose the virtual machine project file “Instant_Contiki_Ubuntu_12.04_32-bit.vmx”

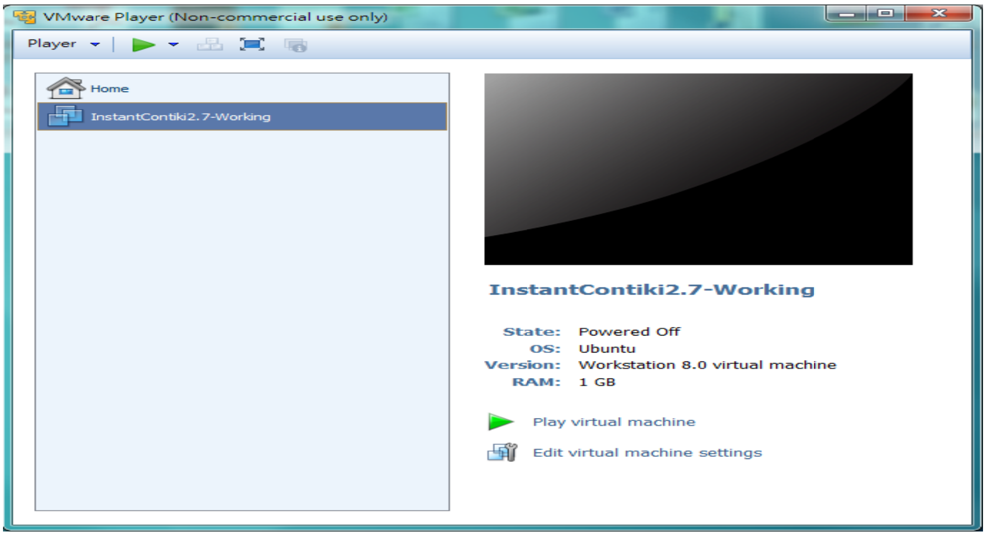

Release_v889 ▸ VMWare ▸ InstantContiki2.7-workoem ▸

名稱

caches

Instant_Contiki_Ubuntu_12.04_32-bit.vmx

(5) Double click the virtual machine name “InstantContiki2.7-Working”


VMware Player (Non-commercial use only)
Player
Home
InstantContiki2.7-Working

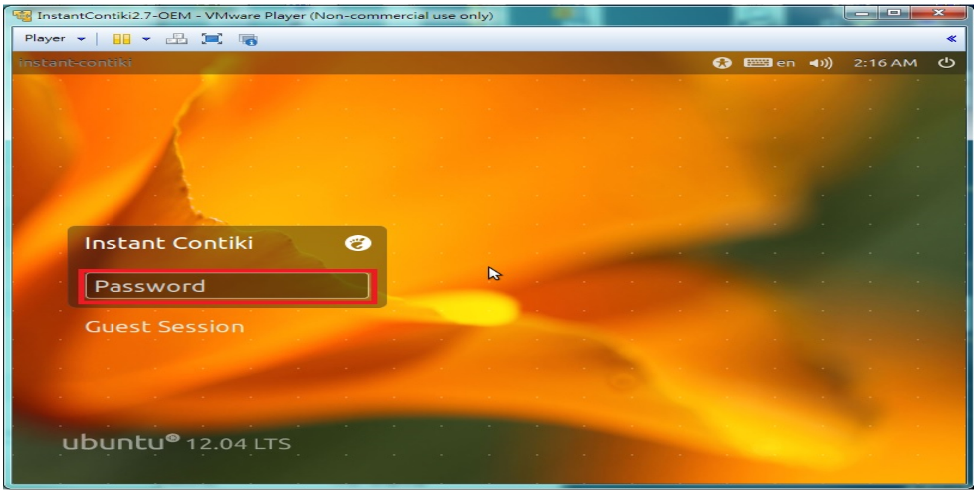
InstantContiki2.7-Working

State: Powered Off
OS: Ubuntu
Version: Workstation 8.0 virtual machine
RAM: 1 GB

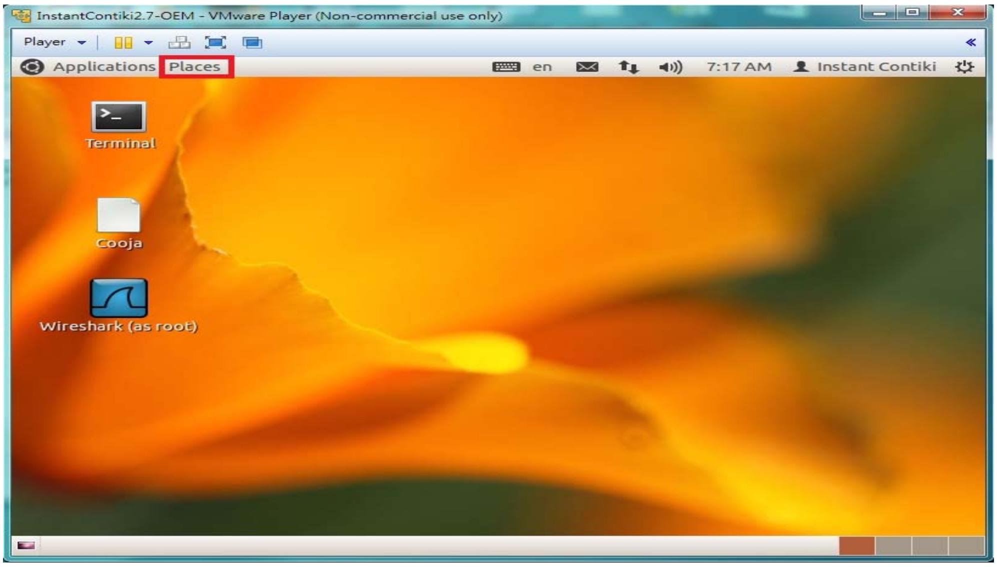
Play virtual machine
Edit virtual machine settings

2.2 F/W build

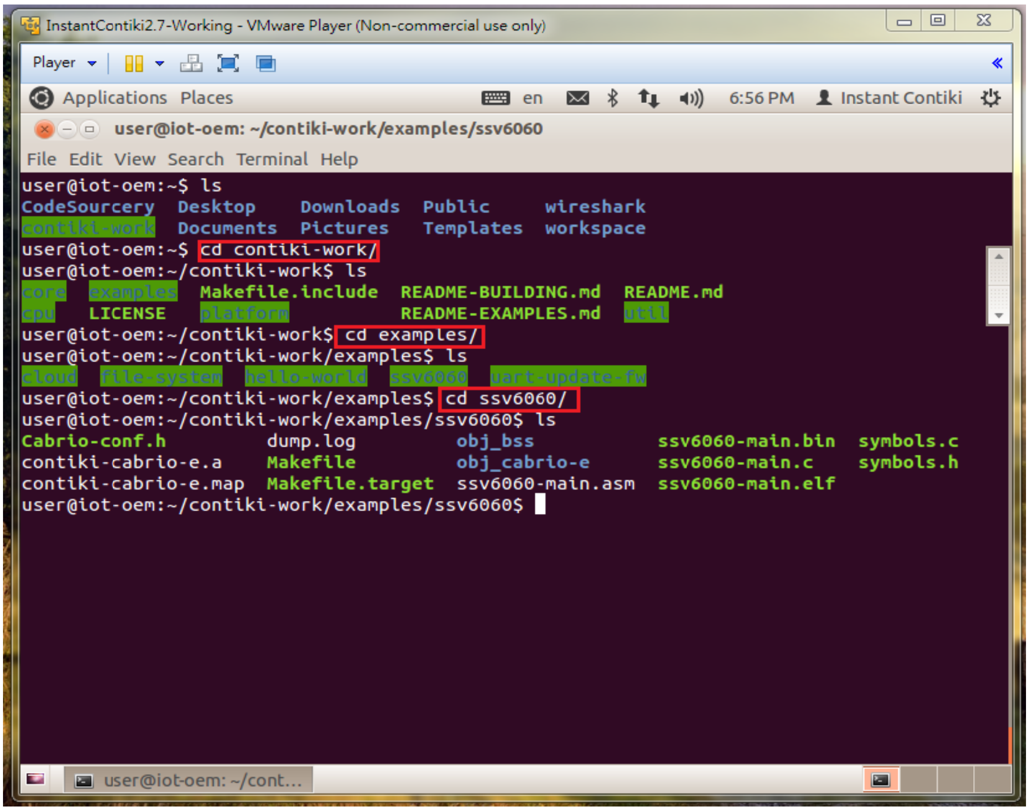
(1) Into Ubuntu developing environment. Type “user” for the password to get into Ubuntu environment.



(2) Press "Places" icon to have a look for the development folders which is located at "contiki-work"

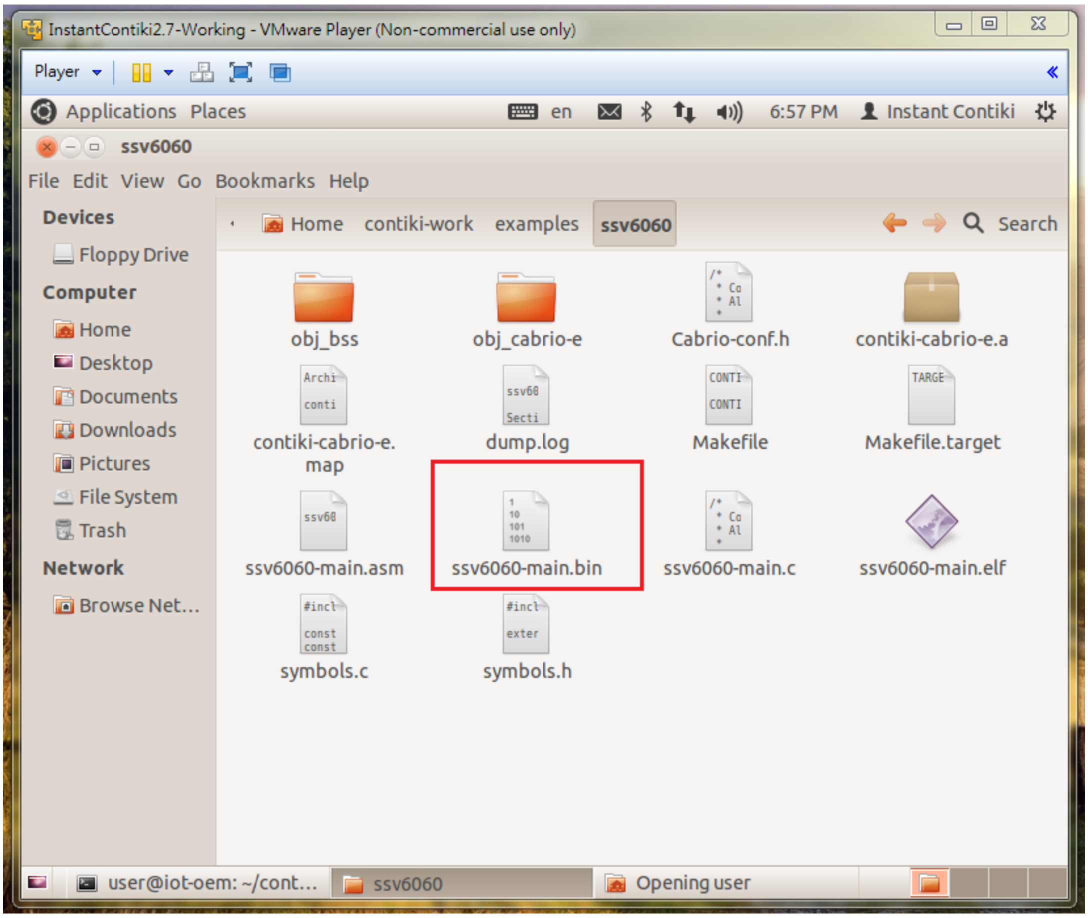


(3) Execute the "Terminal" icon on the Ubuntu Desktop.
Use "**make clean**" to clean up the compiler environment and "**make**" to rebuild the examples project.


InstantContiki2.7-Working - VMware Player (Non-commercial use only)
Player
Applications Places
en
6:56 PM
Instant Contiki
user@iot-oem: ~/contiki-work/examples/ssv6060
File Edit View Search Terminal Help
user@iot-oem: ~/cont...

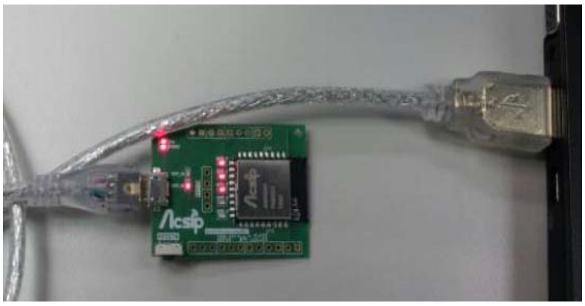
```
user@iot-oem:~$ ls
CodeSourcery  Desktop    Downloads  Public     wireshark
contiki-work  Documents  Pictures   Templates  workspace
user@iot-oem:~$ cd contiki-work/
user@iot-oem:~/contiki-work$ ls
core  examples  Makefile.include  README-BUILDING.md  README.md
cpu   LICENSE   platform          README-EXAMPLES.md  util
user@iot-oem:~/contiki-work$ cd examples/
user@iot-oem:~/contiki-work/examples$ ls
cloud  file-system  hello-world  ssv6060  uart-update-fw
user@iot-oem:~/contiki-work/examples$ cd ssv6060/
user@iot-oem:~/contiki-work/examples/ssv6060$ ls
Cabrio-conf.h         dump.log         obj_bss           ssv6060-main.bin  symbols.c
contiki-cabrio-e.a    Makefile         obj_cabrio-e      ssv6060-main.c    symbols.h
contiki-cabrio-e.map  Makefile.target  ssv6060-main.asm  ssv6060-main.elf
user@iot-oem:~/contiki-work/examples/ssv6060$
```

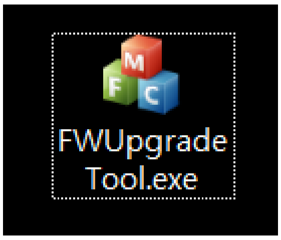
(4) Get a ***.bin** file which is the F/W for IoT module



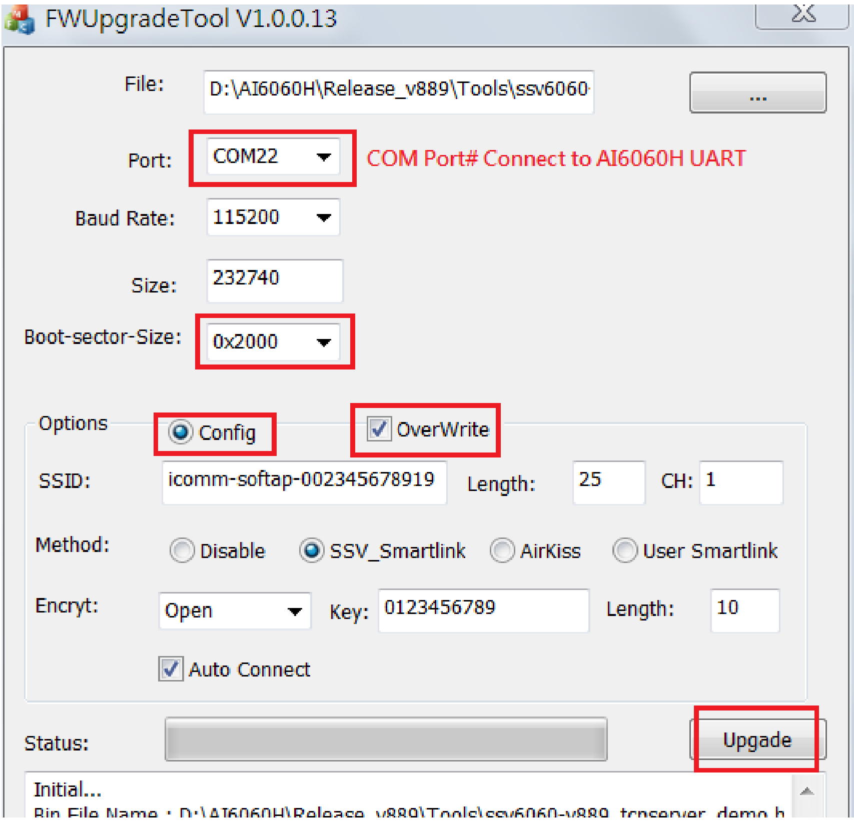
2.3 Update F/W

(1) Connect board and execute F/W upgrade tool





(2) Open tool then update “ **ssv6060-main.bin** ”,



File: D:\AI6060H\Release_v889\Tools\ssv6060

Port: COM22

COM Port# Connect to AI6060H UART

Baud Rate: 115200

Size: 232740

Boot-sector-Size: 0x2000

Options ● Config ☑ OverWrite

SSID: icomm-softap-002345678919 Length: 25 CH: 1

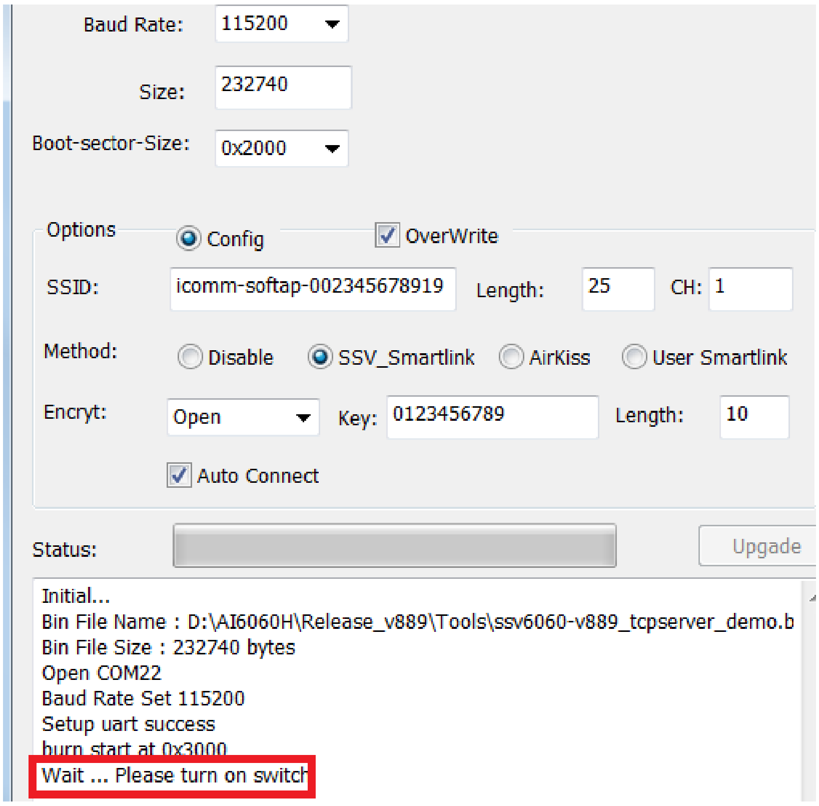
Method: ○ Disable ● SSV_Smartlink ○ AirKiss ○ User Smartlink

Encryt: Open Key: 0123456789 Length: 10

☑ Auto Connect

Status:

```
Initial...
Bin File Name : D:\AI6060H\Release_v889\Tools\ssv6060-v889_tcpserver_demo.b
```


Baud Rate: 115200

Size: 232740

Boot-sector-Size: 0x2000

Options ● Config ☑ OverWrite

SSID: icomm-softap-002345678919 Length: 25 CH: 1

Method: ○ Disable ● SSV_Smartlink ○ AirKiss ○ User Smartlink

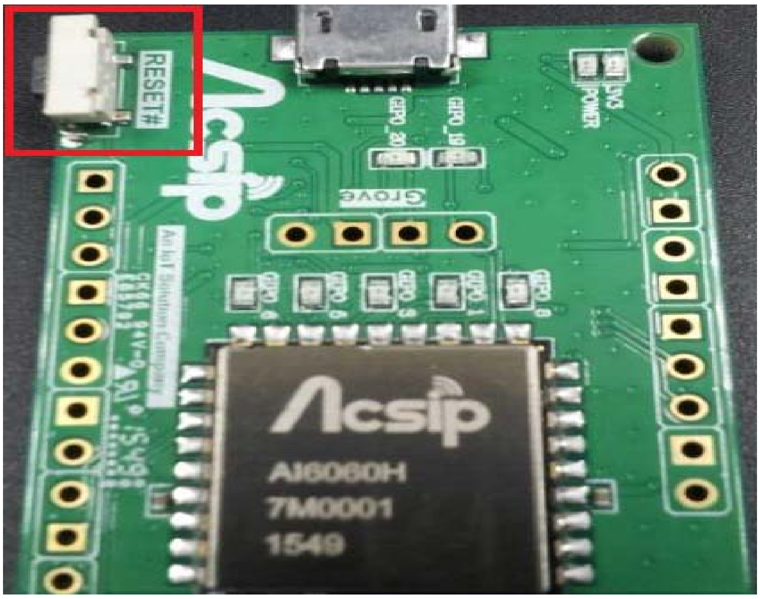
Encryt: Open Key: 0123456789 Length: 10

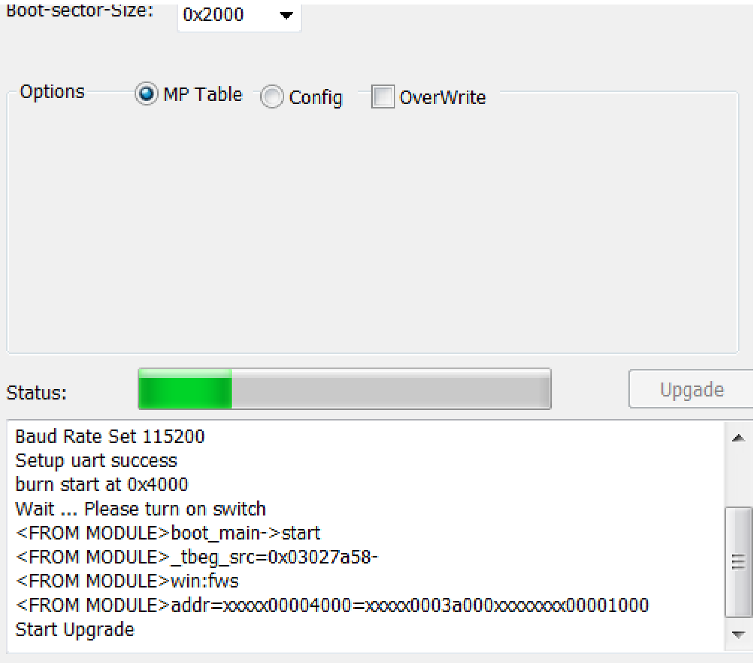
☑ Auto Connect

Status:

```
Initial...
Bin File Name : D:\AI6060H\Release_v889\Tools\ssv6060-v889_tcpserver_demo.b
Bin File Size : 232740 bytes
Open COM22
Baud Rate Set 115200
Setup uart success
burn start at 0x3000
Wait ... Please turn on switc
```

(3) Press board RESET# button




Boot-sector-Size: 0x2000

Options ● MP Table ○ Config □ OverWrite

Status:

```
Baud Rate Set 115200
Setup uart success
burn start at 0x4000
Wait ... Please turn on switch
<FROM MODULE>boot_main->start
<FROM MODULE>_tbeg_src=0x03027a58-
<FROM MODULE>win:fws
<FROM MODULE>addr=xxxxx00004000=xxxxx0003a000xxxxxxxx00001000
Start Upgrade
```

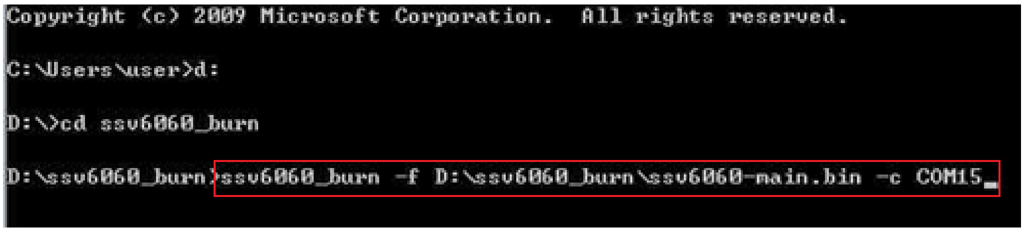
2.4 Burn in command line

(1) Open Windows “Terminal” and run **ssv6060_burn.exe**

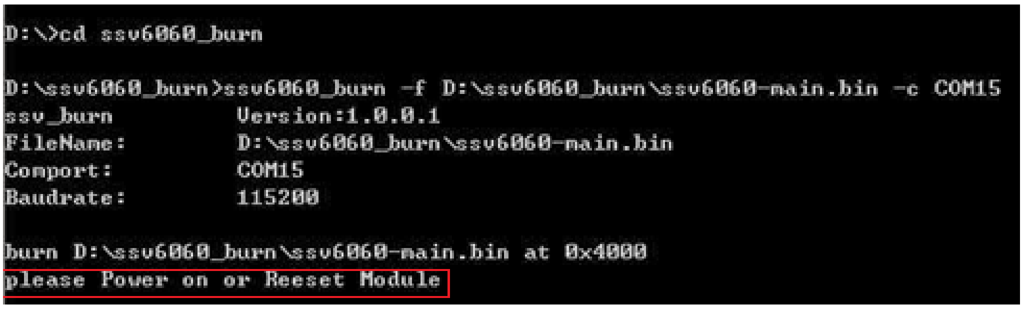
(2) -f: file path , -b: Baud-rate , -c: port

-f , -c is needed ,

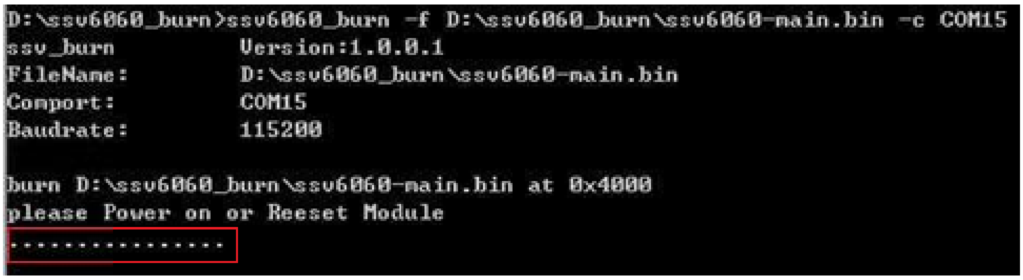
ex：-f D:\ssv6060_burn\ssv6060-main.bin



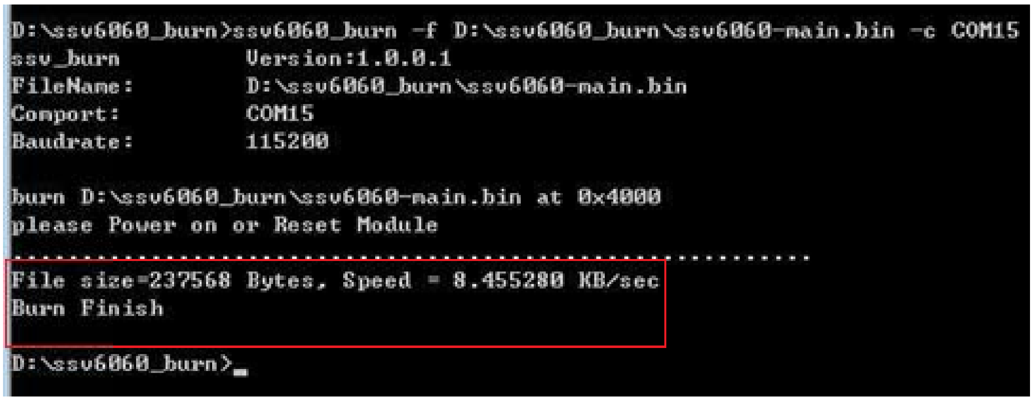
```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>d:

D:\>cd ssv6060_burn

D:\ssv6060_burn>ssv6060_burn -f D:\ssv6060_burn\ssv6060-main.bin -c COM15_
```



```
D:\>cd ssv6060_burn

D:\ssv6060_burn>ssv6060_burn -f D:\ssv6060_burn\ssv6060-main.bin -c COM15
ssv_burn          Version:1.0.0.1
FileName:         D:\ssv6060_burn\ssv6060-main.bin
Comport:          COM15
Baudrate:         115200

burn D:\ssv6060_burn\ssv6060-main.bin at 0x4000
please Power on or Reeset Module
```



```
D:\ssv6060_burn>ssv6060_burn -f D:\ssv6060_burn\ssv6060-main.bin -c COM15
ssv_burn          Version:1.0.0.1
FileName:         D:\ssv6060_burn\ssv6060-main.bin
Comport:          COM15
Baudrate:         115200

burn D:\ssv6060_burn\ssv6060-main.bin at 0x4000
please Power on or Reeset Module
.................
```



```
D:\ssv6060_burn>ssv6060_burn -f D:\ssv6060_burn\ssv6060-main.bin -c COM15
ssv_burn          Version:1.0.0.1
FileName:         D:\ssv6060_burn\ssv6060-main.bin
Comport:          COM15
Baudrate:         115200

burn D:\ssv6060_burn\ssv6060-main.bin at 0x4000
please Power on or Reset Module
.............................................................
File size=237568 Bytes, Speed = 8.455280 KB/sec
Burn Finish

D:\ssv6060_burn>_
```

3 AT command Description

3.1 system reboot

Description	AT+REBOOT
command	none
Return	none

3.2 Get Firmware Version

Description	AT+VERSION=?
Parameters	none
Return	+VERSION:SSV6060.Z0.799.0

3.3 Get Manufacture information

Description	AT+MF_INFO=?
Parameters	none
Return	+MF_INFO AT+VERSION=OK

3.4 Get network configuration

Description	AT+GET_CONFIG=?
Parameters	None
Return	+GET_CONFIG= wifi_mode, AP_SSID, key, key number, dhcp,IP,Submask ,Gateway
	If dhcp =1 +GET_CONFIG=0, 7298A,12345678,8,1,0.0.0.0,0.0.0.0,0.0.0.0 If dhcp =0 +GET_CONFIG=0,7298A, 12345678,8,0,192.168.55.243,255.255.255.0,192.168.55.1

3.5 Set network configuration

Description	AT+SET_IFCONFIG= <dhcp> , <IP> , <Submask> , <Gateway>
Parameters	<dhcp>: 1:Auto , 0: manual <IP>:xxx.xxx.xxx.xxx (if Dhcp =0) <Submask >: xxx.xxx.xxx.xxx (if Dhcp =0) <Gateway>: xxx.xxx.xxx.xxx (if Dhcp =0)
Return	AT+SET_ IFCONFIG =OK
	AT+SET_IFCONFIG=0,192.168.55.243,255.255.255.0,192.168.55.1 AT+SET_IFCONFIG=1

3.6 Set Wi_Fi configuration

Description	AT+SET_WIFICONFIG= <mode>, <AP_SSID>, <Key>
Parameters	<mode>: 0: STA-mode <AP_SSID> : AP name <Key> : AP password
Return	AT+SET_WIFICONFIG=OK
	AT+SET_WIFICONFIG=0,7298A,88888888

3.7 Scanning

Description	AT+SCAN=?
Parameters	None
Return	+SCAN:7298A, SSV-AP5_2.4G, QA.DIR524, SSV_AP2, TP-LINK_45FE5E, D-Link_DIR-Jay, APET, test, OOX, for.interchannel.wr845n, for.interchannel.c8, Eric_Fw, SSV_AP2, icomm-softap-002345678916, Winnie_NB-PC, EnGenius_wpa2aes, AT+SCAN=OK

3.8 Wi_Fi AP connection

Description	AT+WIFICONNECT
Parameters	None
Return	AT+WIFICONNECT=OK
	If dhcp =1 Got IP address 192.168.43.16 Got netmask 255.255.255.0 Got DNS server 192.168.43.1 Got default router 192.168.43.1

3.9 Wi_Fi AP Disconnection

Description	AT+WIFIDISCONNECT
Parameters	None
Return	AT+WIFIDISCONNECT=OK

3.10 Show connect AP

Description	AT+SHOWCONNECTAP
Parameters	None
Return	[0]7298A, ch: 1, rssi: -29 dBm, rssiLevel: 4, security_type = WPA2/AES , HT-MM SGI MCS7

3.11 Show All AP

Description	AT+SHOWALLAP
Parameters	None
Return	[0]7298A ch: 1, rssi: - 26 dBm, rssiLevel: 4, security_type = WPA2/AES [1]AP5_2.4G ch: 1, rssi: - 42 dBm, rssiLevel: 4, security_type = WPAWPA2/AES [2]QA.DIR524 ch: 1, rssi: - 47 dBm, rssiLevel: 4, security_type = OPEN/NONE [3]D-Link_DIR-Jay ch: 2, rssi: - 80 dBm, rssiLevel: 2, security_type = OPEN/NONE

3.12 TCP connect

Description	AT+TCPCONNECT=<IP>,<PORT>
Parameters	IP : xxx.xxx.xxx.xxx,ex:192.168.112.10 PORT : xxxx,ex:2000
Return	socket number : create socket:0

3.13 TCP Send

Description	AT+TCPSEND=<socket number>,<data>
Parameters	socket number : Use TCPCONNECT to get socket number data : string,ex:12345678
Return	AT+TCPSEND=OK

3.14 TCP Disconnect

Description	AT+TCPDISCONNECT=<socket>
Parameters	None
Return	AT+TCPDISCONNECT=OK
	ssv6060>:AT+TCPDISCONNECT=0 ssv6060>:socked:0 closed AT+TCPDISCONNECT=OK

3.15 TCP SERVER

Description	AT+TCPLISTEN=<PORT>
Parameters	PORT: number ; ex :2000
Return	AT+TCPLISTEN=OK
	ssv6060>:AT+TCPLISTEN=2000 AT+TCPLISTEN=OK ssv6060>:new connected to listen port(2000), socket:0

3.16 Clean TCP SERVER

Description	AT+TCPUNLISTEN=<PORT>
Parameters	PORT: number ; ex :2000
Return	AT+TCPUNLISTEN=OK

3.17 UDP Create

Description	AT+UDPCREATE=<PORT>
Parameters	PORT: number ; ex :2000
Return	Socket number : create socket:12
	ssv6060>:AT+UDPCREATE=2000 create socket:12 AT+UDPCREATE=OK

3.18 UDP Send

Description	AT+UDPSEND=<socket>,<Client IP>,<port>,<Data>
Parameters	<socket> : Use UDPCREATE to get socket number <Client IP> : xxx.xxx.xxx.xxx;ex： 192.168.112.10 <port>: port number <Data>: string
Return	None
	AT+UDPSEND=12,192.168.43.23,11111,asdfgh ssv6060>:UDP socked:12 recvdata:fgshfshfh from 192.168.43.23:61148

3.19 Close UDP

Description	AT+UDPCLOSE =<socket>
Parameters	<socket> : Use UDPCREATE to get socket number
Return	AT+UDPCLOSE=OK
	ssv6060>:AT+UDPCLOSE=12 AT+UDPCLOSE=OK

3.20 GPIO Setting

Description	AT+SET_GPIO=<id>,<mode>
Parameters	<id>: GPIO ID. <mode>: 0:input ; 1:output.
Return	AT+SET_GPIO=OK
	ssv6060>:AT+SET_GPIO=2,1

3.21 GPIO Write

Description	AT+WRITE_GPIO=<id>,<value>
Parameters	<id>: GPIO ID. <value>: 0:low ; 1:high.
Return	AT+WRITE_GPIO=OK
	ssv6060>:AT+WRITE_GPIO=2,1

3.22 GPIO Read

Description	AT+READ_GPIO=<Number>
Parameters	<Number>
Return	On: 1 ; Off: 0
	ssv6060>:AT+READ_GPIO=3 +GPIO:1

3.23 PWM setting

Description	AT+SET_PWM=<id>,<Enable>,<Cycle>
Parameters	<id>:GPIO ID = 5 (default GPIO8) <Enable>: 0 , 1 <Cycle>: 0 ~ 10
Return	ssv6060>:AT+SET_PWM=5,1,0 +OK

3.24 **SMART LINK** mode

Description	AT+ENABLE_SMARTREBOOT=<type>
Parameters	<type> : 0 : NO Smart Link 1 : ICOMM Smart Link 2 : WECHAT 3 : USER
Return	None
	Need to reboot

3.25 AP mode

Description	AT+AP
Parameters	None
Return	[At_AP] : +++

3.26 EXIT AP mode

Description	AT+AP_EXIT
Parameters	None
Return	[At_AP_EXIT] : +++

3.27 Set AP SSID

Description	AT+SET_AP_SSID=<name>
Parameters	<name>:SSID
Return	<gconfig_set_softap_ssid> new_softap_ssid=ABC <gconfig_set_softap_ssid> i_config.softap_ssid=ABC
	ssv6060>:AT+SET_AP_SSID=ABC [At_SET_AP_SSID] : +++ <gconfig_set_softap_ssid> new_softap_ssid=ABC <gconfig_set_softap_ssid> i_config.softap_ssid=ABC

3.28 Start RF

Description	AT+RADIO_RF_START=<range>
Parameters	<range> = 0
Return	None
	AT+RADIO_RF_START=0

3.29 Set RF CHANNEL

Description	AT+RADIO_CHANNEL=<range>
Parameters	<range> =1~13
Return	None
	AT+RADIO_CHANNEL=12

3.30 WIFI PACKET Format

Description	AT+RADIO_RF_RATE=<range>
Parameters	<range>:0~30
Return	None
	AT+RADIO_RF_RATE=22

3.31 G/N Mode Gain

Description	AT+RADIO_RF_GNGAIN= <range>
Parameters	<range>:5~15
Return	None
	AT+RADIO_RF_GNGAIN=7

3.32 B Mode Gain

Description	AT+RADIO_RF_BGAIN= <range>
Parameters	<range>:5~15
Return	None
	AT+RADIO_RF_BGAIN=2

3.33 IC Temp.

Description	AT+RADIO_RF_READ_TEMPCS=?
Parameters	None
Return	Temp:°C Ex: AT+RADIO_RF_READ_TEMPCS=28

3.34 Stop RF

Description	AT+RADIO_RF_STOP
Parameters	None
Return	None

3.35 RF TCSR

Description	AT+RADIO_RF_ENABLE_TCSR=<on/off>
Parameters	<on/off>: 0:Off ; 1:On
Return	None

3.36 RF frequency OFFSET

Description	AT+RADIO_RF_FREQOFFSET=<p range>,<n range>
Parameters	<p range>:1~16 <n range>:1~16
Return	None

3.37 Dump all RF OFFSET

Description	AT+RADIO_RF_DUMP
Parameters	None
Return	列出 ALL RF 補償值 Ex: =============== channel = 1 B power = 11 G/N power = 9 Freq. Offset = 2,2 LDO = 4b661c PA1 = 3d5e84ff PA2 = 1457d79 PA3 = fcccce27 Boundary = 0,0 ===============

3.38 RF LDO

Description	AT+RADIO_RF_LDO=<n range>,<m range>
Parameters	<n range>:1~8 <m range>:5~8
Return	None

3.39 RF_PA1 TUNING

Description	AT+RADIO_RF_PA1= <range>
Parameters	<range>:6~12
Return	None

3.40 RF_PA2 TUNING

Description	AT+RADIO_RF_PA2= <range>
Parameters	<range>:1~7
Return	None

3.41 RF_PA3 TUNING

Description	AT+RADIO_RF_PA3= <range>
Parameters	<range>:7~14
Return	None

3.42 RF_IQ PHASE

Description	AT+RADIO_RF_IQPHASE=<range>
Parameters	<range>: (-7~7)
Return	None

3.43 RF_IQ AMP

Description	AT+RADIO_RF_IQAMP=<range>
Parameters	<range>: (-7~7)
Return	None

4. API Description:

4.1 TAG_CABRIO_CONFIGURATION data structure

The data structure of TAG_CABRIO_CONFIGURATION is used to store the necessary information for the operation of WiFi.

```
typedef struct t_TAG_CABRIO_CONFIGURATION
{
   u8_t   wifi_mode;          // 1: AP mode, 0: STA-mode, IBSS, Infrastructure
   char   wifi_ssid[32];      // WiFi AP name or STA name
   u8_t   wifi_ssid_len;      // The length of SSID
   u8_t   wifi_mac[6];        // MAC address of Wifi AP
   u8_t    wifi_privacy;
   TAG_SECURITY   wifi_security;
   u8_t   wifi_wepkeylen;   // 5, 13
   char   wifi_wepkey[64]; // 40bit and 104 bit
   char   wifi_pmk[32];       // 40bit and 104 bit
   u8_t   wifi_channel;       //  WiFI  channel  for  scanning  or  connect  to  WiFi
AP

   u8_t   dhcp_enable;        // 1: DHCP Enable 0: Fix IP
   u8_t   connect_mode;      // 0: Client , 1: Server
   uip_ip4addr_t local_ip_addr; // IP address of local host
   uip_ip4addr_t net_mask;        // Net mask of local host
   uip_ip4addr_t gateway_ip_addr;    // Gateway IP address of local host
   u8_t   gateway_mac[6];          // MAC addrsss of gateway
   u8_t   gateway_mac_set;

   uip_ip4addr_t   remote_ip_addr;   //   Remote   IP   addrss   for   TCP/UDP
connection
   u16_t port;       // Remote IP port number for TCP/UDP connection

   u8_t   local_mac[6];       // MAC address of local host

   u8_t   intoSmartLink;      // Activate of Smartlink process
   u8_t   debugLevel;
} TAG_CABRIO_CONFIGURATION
```

4.2 Reboot the firmware

Description :

The command will reboot the firmware of SSV6060.

Syntax :

void

bss_mgmt_reboot (void)

Parameters :

None

Return Value :

None

Remark :

None

4.3 Core Library Initialize

Description :

The command initializes the necessary parameters of 802.11 core library. This should be called every time.

Syntax :

void

bss_mgmt_init (void)

Parameters :

None

Return Value :

None

Remark :

None

4.4 WiFi AP Scan

Description:

The command will search the nearby WiFi Aps

Syntax:

void

bss_mgmt_scan (void)

Parameters:

None

Returns:

The return value will be send to the PROCESS_THREAD “Cabrio_ate_process” and call the function “At_ParserInd”. This thread will continue to listen the AT command and the result from 802.11 core library.

Remark :

```
    #define MAX_AP_LIST        100  // Maximum number of WiFi AP stored
    // The scanned of WiFi AP’s information will be store into “ap_list”.
    typedef struct t_TAG_AP_INFO
    {
            u8_t            name[32];
            u8_t            name_len;
            u8_t            mac[6];
            u8_t            channel;
    } TAG_AP_INFO;
TAG_AP_INFO ap_list[MAX_AP_LIST];
```

4.5 WiFi AP Connection

Description :

The command will connect the desired WiFi AP.

Syntax :

```
void
bss_mgmt_connect (void)
```

Parameters :

None

Returns :

None

Remark :

While happens to connection, some member of TAG_CABRIO_CONFIGURATION need to be pre-stored. Such as …

```
u8_t   wifi_mode;          // 1: AP mode, 0: STA-mode, IBSS, Infrastructure
```

```
char   wifi_ssid[32];     // WiFi AP name or STA name
u8_t   wifi_ssid_len;     // The length of SSID
u8_t   wifi_mac[6];       // MAC address of Wifi AP
```

The return value will be send to the PROCESS_THREAD “Cabrio_ate_process” and call the function “At_ParserInd” with message ID “MSG_ATE_CONNECT”.

4.6 WiFi AP Disconnect

Description :

The command will disconnect the connection of WiFi AP.

Command :

```
void
bss_mgmt_disconnect (void)
```

Parameters :

None

Returns :

None

Remark :

The return value will be send to the PROCESS_THREAD “Cabrio_ate_process” and call the function “At_ParserInd” with message ID “MSG_ATE_DISCONNECT”.

4.7 TCP Connection

Description :

The command will connects to remote IP’s TCP port. Please make sure the SSV6060 is connected to a WiFi AP before calling TCP connection.

Command :

```
Struct uip_conn *
tcp_connect (
uip_ipaddr_t *ripaddr,
uint16_t port,
void *appstate)
```

Parameters :

ripaddr

The remote IP address for TCP connection

port

The remote IP's port number for TCP connection

appstate

NULL for this field

Returns :

uip_conn * will be return for the handle of TCP connection.

Remark

4.8 TCP Disconnection

Description :

The command will disconnect the connection of remote IP's TCP port.

Command :

```
unsigned char
tcpclient_close (
struct tcpclient_state *s)
```

Parameters :

*s

The handle of current TCP connection.

Returns :

1 : for the success of disconnection

0 : fail of diconnection

Remark

4.9 GPIO setting

Description :

The command is used for setting the attribute of GPIO.

Command :

```
void
pinMode(
    PIN_ID id,
    PIN_MODE mode)
```

Parameters :

```
    id
            typedef enum t_PIN_ID
            {
                    PIN_11          = 0,
                    PIN_13,
                    PIN_15,
                    PIN_16,
                    PIN_30,
                    PIN_31,
                    PIN_33,
                    PIN_34
            } PIN_ID;
    mode
            typedef enum t_PIN_MODE
            {
                    INPUT           = 0,
                    INPUT_PULL,
                    OUTPUT,
                    OUTPUT_DRIVINGUP
            } PIN_MODE;
```

Returns :

None

Remark :

4.10 GPIO Write

Description :

The command will send the data to GPIO.

Command :

```
    void
```

```
digitalWrite(
PIN_ID id,
u32 data)
```

Parameters :

id

```
typedef enum t_PIN_ID
{
        PIN_11          = 0,
        PIN_13,
        PIN_15,
        PIN_16,
        PIN_30,
        PIN_31,
        PIN_33,
        PIN_34
} PIN_ID;
```

data

The data is going send to GPIO

Returns :

None

Remark :

4.11 GPIO Read

Description :

Read data from GPIO.

Command :

```
u32
digitalRead(
PIN_ID id)
```

Parameters :

id

```
typedef enum t_PIN_ID
{
        PIN_11          = 0,
        PIN_13,
        PIN_15,
        PIN_16,
        PIN_30,
        PIN_31,
        PIN_33,
        PIN_34
} PIN_ID;
```

Returns :

Return data value in 32-bit data.

Remark :

4.12 PWM delay

Description :

Control PWM behavior..

Command:

```
int
enablePWM(
PIN_ID id,
u8 dutycycle)
```

Parameters :

id

```
typedef enum t_PIN_ID
{
        PIN_11          = 0,
        PIN_13,
        PIN_15,
        PIN_16,
        PIN_30,
        PIN_31,
        PIN_33,
        PIN_34
    } PIN_ID;
```

dutycycle

Duty cycle.

Returns :

0 : Success

-1 : fail

Remark :

4.13 Disable PWM

Description :

Disable PWM function.

Command :

```
void
disablePWM(
PIN_ID id)
```

Parameters :

id

```
typedef enum t_PIN_ID
{
    PIN_11          = 0,
    PIN_13,
    PIN_15,
    PIN_16,
    PIN_30,
    PIN_31,
    PIN_33,
    PIN_34
} PIN_ID;
```

Returns :

None

Remark :

5. Network programming

Because this is nonblack IO. We need to create a process to get message and confirm the TCP connection, send data is completed or there is new data or connection coming. There is a simple example show below. Process need to wait event PROCESS_EVENT_MSG and the data pointer type is SOCKETMSG. We can know this message come from which socket, port and status.

```
typedef enum t_SOCKETSTATE{
    SOCKET_CONNECTED,           ← TCP socket is connected.
    SOCKET_CLOSED,              ← TCP connection is closed.
    SOCKET_SENDACK,             ← The send data procedure is completed.
    SOCKET_NEWDATA,             ← There is new data coming.
    SOCKET_NEWCONNECTION,       ← A new connection from listening port is
created.
}SOCKETSTATE;

typedef struct t_SOCKETMSG
{
    NETSOCKET    socket;
    U16          lport;
    SOCKETSTATE status;
}SOCKETMSG;

PROCESS_THREAD(tcp_connect_process, ev, data)
{
    PROCESS_BEGIN();
    SOCKETMSG msg;

    while(1)  {
        PROCESS_WAIT_EVENT();
        if(ev == PROCESS_EVENT_MSG) {
            msg = *(SOCKETMSG *)data;
            //Doing things depend on which message
        }
    }
    PROCESS_END();
}
```

5.1 TCP client programming

- **Create a TCP connection**

  Call function tcpconnect to start TCP connect to peer side and need wait for SOCKET_CONNECTED message .

```
    httpsock = tcpconnect( &gNetStatus.remote_ip_addr, gNetStatus.port,
&http_request_process);
    //wait for TCP connected or timeout.
    PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_MSG);
    msg = *(SOCKETMSG *)data;
    if(msg.status != SOCKET_CONNECTED) {
        printf("TCP connect fail! Post message type:%d\n", msg.status);
        goto dissconnect;
    }
```

- **Send data to the peer side**

  Call function tcpsend to send data to peer side and wait for acknowledge. Before get the SOCKET_SENDACK message, please do not modify the data buffer.

```
tcpsend(httpsock, httpstring, strlen (httpstring));
//Wait for data is transmitted or uip_timeout.
PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_MSG);
msg = *(SOCKETMSG *)data;
if(msg.status == SOCKET_SENDACK) {
    printf("TCP send successful\n");
}else{
    printf("TCP send fail! Post message type:%d\n", msg.status);
    goto dissconnect;
}
```

- **Receive data from peer side**

  Wait for SOCKET_NEWDATA message then call function tcprecv to get incoming data.

```
PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_MSG)
msg = *(SOCKETMSG *)data;
```

```
if(msg.status == SOCKET_NEWDATA) {
        recvlen = tcprecv(httpsock, buffer_in, MAX_SEND_BUFFER);
}
```

- Close connection

  Call function tcpclose to close connection.

  ```
  tcpclose(httpsock);
  ```

There is a simple example “HTTP request” to show how the TCP client working.

Please refer

\Cabrio contiki src\examples\socket_proc\socket_proc.c

PROCESS_THREAD(http_request_process, ev, data)

5.2 TCP server programming

- **Listen a TCP port**

  Call function tcplisten to listen the TCP port.

  ```
  tcplisten(localport, &tcp_connect_process)
  ```

- **Accept a new connection**

  Wait for SOCKET_NEWCONNECTION message and get the new socket number from message.

  ```
  PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_MSG)
  msg = *(SOCKETMSG *)data;
  if(msg.status == SOCKET_NEWCONNECTION) {
          gserversock = msg.socket;
  }
  ```

- **Attach another process.**

  The new connection will use the callback process register by tcplisten as default callback process. If we want another process to be the callback process of this socket, it can call function tcpattach to replace the callback function.

  ```
  tcpattch(gserversock, &client_handle_process);
  ```

- **Stop listen a TCP port**

Call function tcpunlisten to stop listening the TCP port.

```
tcpunlisten(localport);
```

5.3 UDP programming

- **Create a UDP socket**

Call function udpcreate to create a UDP socket.

```
gudpsock = udpcreate(gNetStatus.udpport, &udp_connect_process);
```

- **Send data to the peer side**

Call function udpsendto to send data to peer side.

```
udpsendto(gudpsock, pdata, strlen(pdata), &remote_ip_addr, rport)
```

- **Receive data from peer side**

Wait for SOCKET_NEWDATA message then call function udprecvfrom to get incoming data.

```
PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_MSG)
msg = *(SOCKETMSG *)data;
if(msg.status == SOCKET_NEWDATA) {
    recvlen = udprecvfrom (msg.socket, buffer_in, MAX_SEND_BUFFER,
&peeraddr, &peerport);
}
```

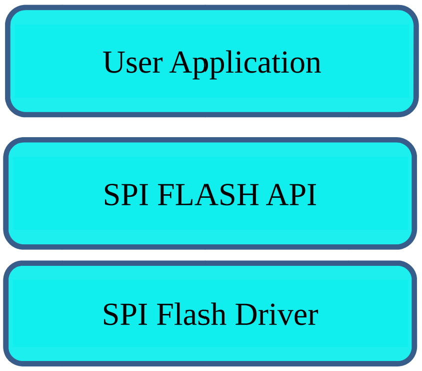
- **Close socket**

Call function udpclose to close socket.

```
udpclose(sock);
```

6. SPI Flash API

6.1 SPI Flash API structure



6.2 SPI Flash directory structure

API layer:
./ icomlib/include /flash_api.h
Driver layer:
. / icomlib/include /drv_flash.h

6.3 SPI Flash API introduction

Function name	Description
spi_flash_init()	Call driver initialization in this API.
spi_flash_read	Use this to read flash data into your reserved memory destination.
spi_flash_sector_erase	Use this api to erase sector (4KB) in spi flash. All data will be erase as 0xffff-ffff in whole sector.
spi_flash_write	Use this api to write data into spi flash driver’s interal 4KB cache buffer. But beware,data won’t be writtern into flash in this API.
spi_flash_finalize	Data in flash driver’s 4kB cache buffer will be written into flash by this API.

6.4 SPI Flash Driver introduction

Function name	Description
drv_flash_init	To setup spi command related information. In our driver,we now only setup following spi flash command: 1).read status1 2).read status2 3).write enable 4).erase sector 5).page program
drv_flash_get_cache_addr	Get start address of 4KB driver cache buffer. It's address is in sram range.
drv_flash_get_base_addr	Get start address of spi flash. In current tiramisu system,it's 0x300-0000
drv_flash_write_cmd	When all command or data related information being setup. This will triggr HW to send data into spi bus.
drv_flash_wait_spi_busy	Use this function to check if our HW busy.
drv_flash_wait_spi_status_regsiter	Use this function to check if SPI bus busy
drv_flash_write_enable	SPI flash need to write this command on spi bus first before programming data into spi flash.
drv_flash_sector_erase	Since IOT won't have large size flash. In our driver,we now only provide sector(4KB) level erase.
drv_flash_page_program	This function is the major function to write data. The flow would be "Copy data to specific SRAM" -> SPI bus->SPI flash.

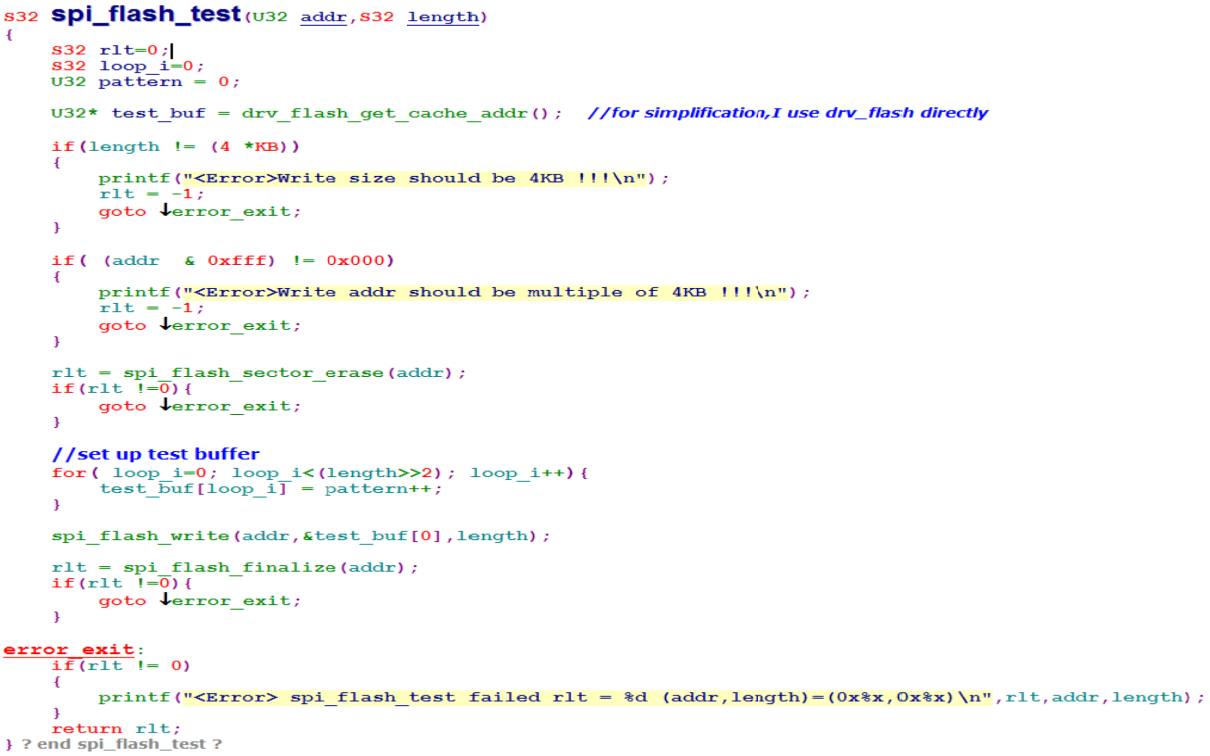
6.5 How SPI Flash Driver make use of HW

SPI command type	Procedures
Page program	Step1:initialization Set up register to let HW know the exact sram address of spi command. You can see the implementation in drv_flash_int(): In our implementation. I allocate two (256+4) byte buffers. REG32(SPI_ADDR_CMD_SRAM_ADDR) = (unsigned int)(&au8spi_cmd[0]); REG32(SPI_ADDR_DATAIN_SRAM_ADDR) = (unsigned int)(&au8spi_data[0]); Step2:write command into au8spi_cmd[0]. Step3:trigger HW to send command.It means you set up the command length,thus HW will know many bytes command need to be sent out.

	For detail implementation,please see drv_flash_page_program.

SPI command type	Procedures
Read status 1 register	Step1:initialization Set up register to let HW know the exact sram address of spi command. You can see the implementation in drv_flash_int(): In our implementation. I allocate two (256+4) byte buffers. REG32(SPI_ADDR_CMD_SRAM_ADDR) = (unsigned int)(&au8spi_cmd[0]); REG32(SPI_ADDR_DATAIN_SRAM_ADDR) = (unsigned int)(&au8spi_data[0]); Step2:write command into au8spi_cmd[0]. Step3:trigger HW to send command.It means you set up the command length,thus HW will know many bytes command need to be sent out. And then read data from au8spi_data byte buffer.

6.6 Example for using SPI Flash API



```
S32 spi_flash_test(U32 addr,S32 length)
{
    S32 rlt=0;
    S32 loop_i=0;
    U32 pattern = 0;

    U32* test_buf = drv_flash_get_cache_addr();   //for simplification,I use drv_flash directly

    if(length != (4 *KB))
    {
        printf("<Error>Write size should be 4KB !!!\n");
        rlt = -1;
        goto ↓error_exit;
    }

    if( (addr  & 0xfff) != 0x000)
    {
        printf("<Error>Write addr should be multiple of 4KB !!!\n");
        rlt = -1;
        goto ↓error_exit;
    }

    rlt = spi_flash_sector_erase(addr);
    if(rlt !=0){
        goto ↓error_exit;
    }

    //set up test buffer
    for( loop_i=0; loop_i<(length>>2); loop_i++){
        test_buf[loop_i] = pattern++;
    }

    spi_flash_write(addr,&test_buf[0],length);

    rlt = spi_flash_finalize(addr);
    if(rlt !=0){
        goto ↓error_exit;
    }

error_exit:
    if(rlt != 0)
    {
        printf("<Error> spi_flash_test failed rlt = %d (addr,length)=(0x%x,0x%x)\n",rlt,addr,length);
    }
    return rlt;
} ? end spi_flash_test ?
```

Note: Please beware that when program data into spi flash. Interrupt must be disabled or the System will han

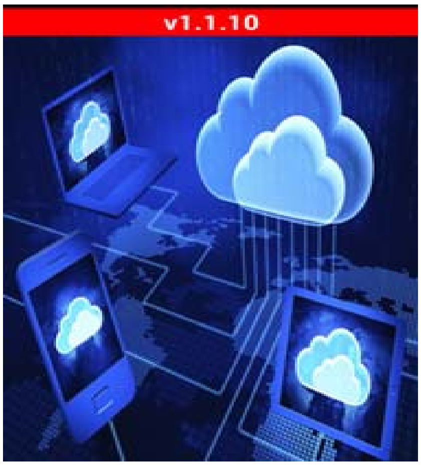
7. Smart_Link Guide

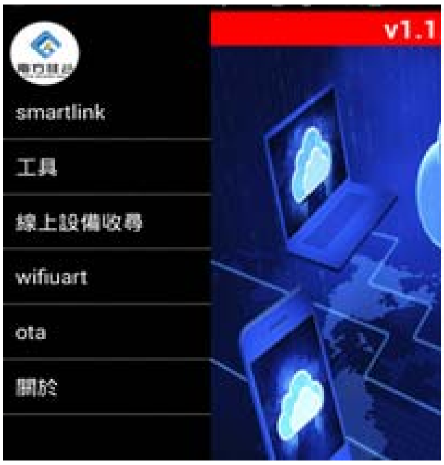
The APP(iComm_tool.apk) provide a simple way for the first initialization of AI 6060H through smartphone.

This APP will pass your currently connected WiFi AP’s name and password to AI6060H. As AI6060H received the message, AI6060H can connect to WiFi AP and will send success message back to the APP on smartphone. By this scenario, APP is capable to initialize multiple AI6060 H at the same time with few steps.

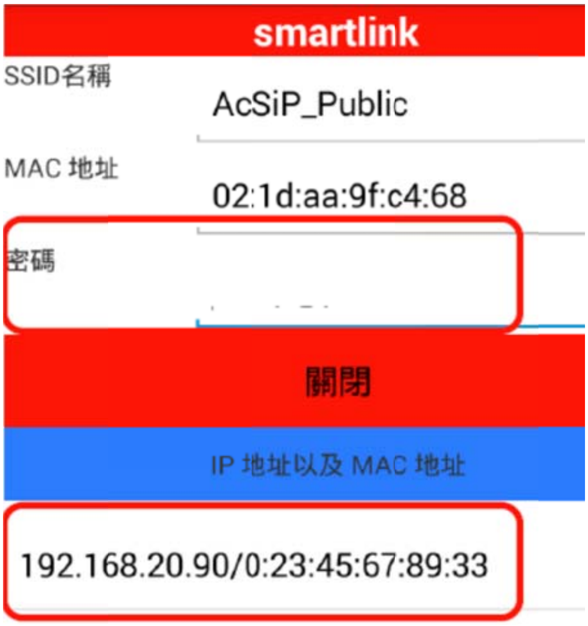
7.1 Install “ ICOM SmartLink.apk” and execute it:

7.2 Open APP





7.3 Tap “ smartlink” and key-in SSID/Password Then get the IP & Mac


smartlink

SSID名稱 AcSiP_Public

MAC 地址 02:1d:aa:9f:c4:68

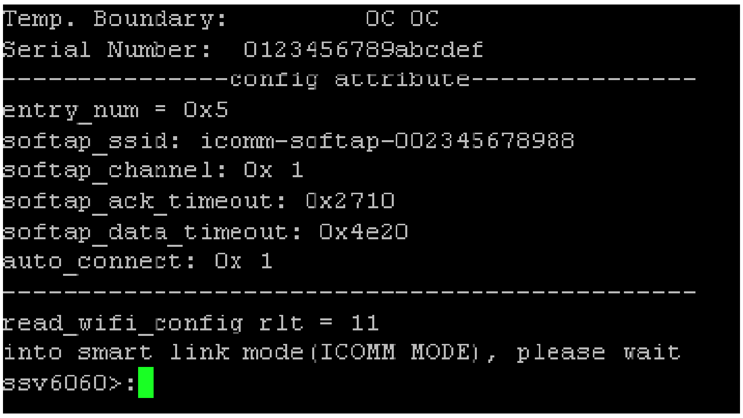
密碼

關閉

IP 地址以及 MAC 地址

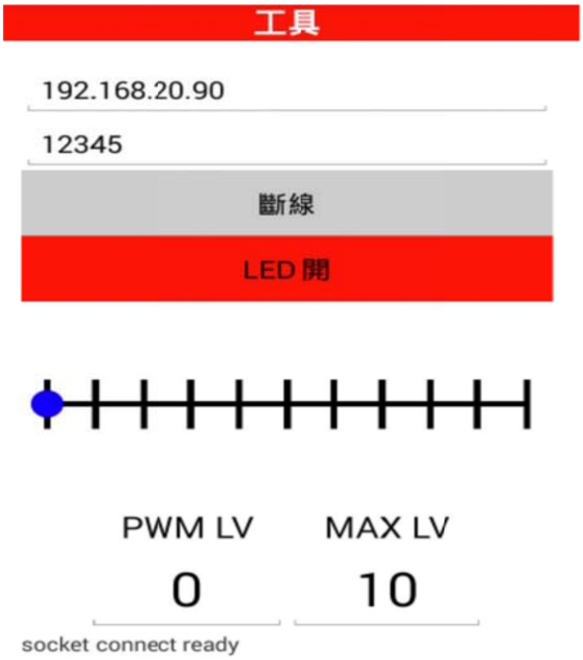
192.168.20.90/0:23:45:67:89:33

7.4 Console information



```
Temp. Boundary:          0C 0C
Serial Number:  0123456789abcdef
---------------config attribute---------------
entry_num = 0x5
softap_ssid: icomm-softap-002345678988
softap_channel: 0x 1
softap_ack_timeout: 0x2710
softap_data_timeout: 0x4e20
auto_connect: 0x 1
----------------------------------------------
read_wifi_config rlt = 11
into smart link mode(ICOMM MODE), please wait
ssv6060>:
```

7.5 Control LED on/off and PWM function



192.168.20.90

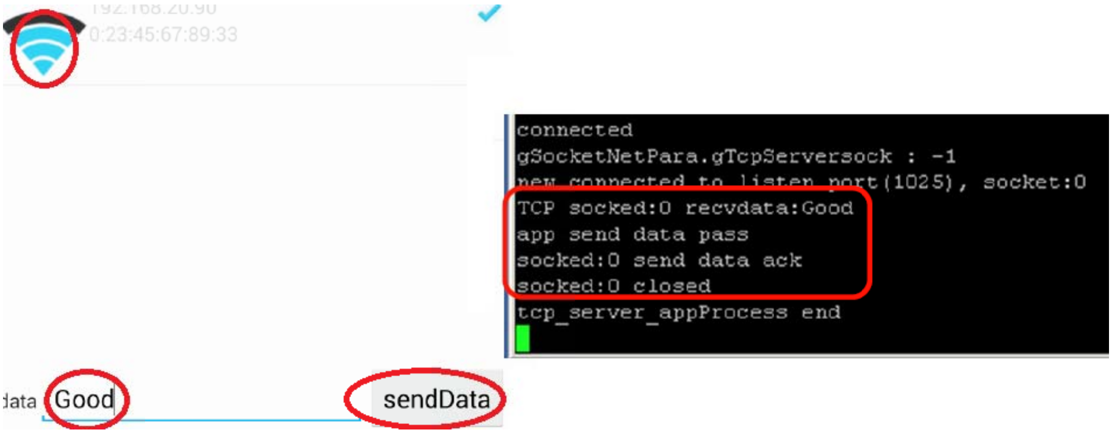
12345

PWM LV 0

MAX LV 10

socket connect ready

7.6 Wifiuart function:



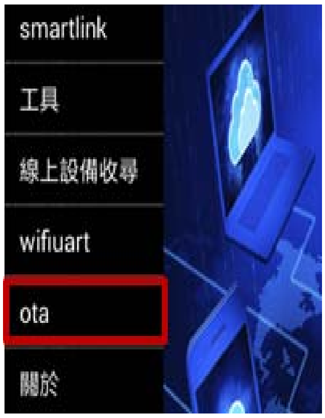
192.168.20.90
0:23:45:67:89:33

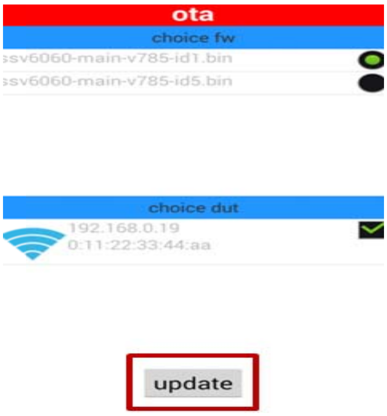
```
connected
gSocketNetPara.gTcpServersock : -1
new connected to listen port(1025), socket:0
TCP socked:0 recvdata:Good
app send data pass
socked:0 send data ack
socked:0 closed
tcp_server_appProcess end
```

data Good

8. OTA Guide

8.1 Open APP and choice F/W &　Dut





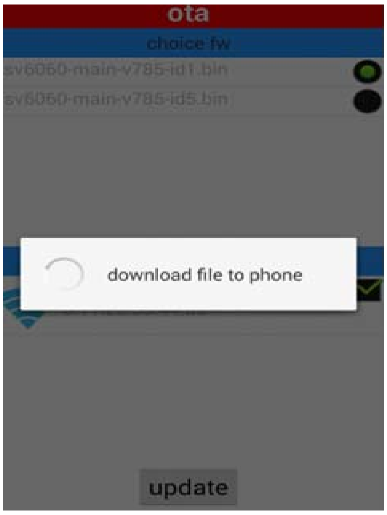
ssv6060-main-v785-id1.bin

ssv6060-main-v785-id5.bin

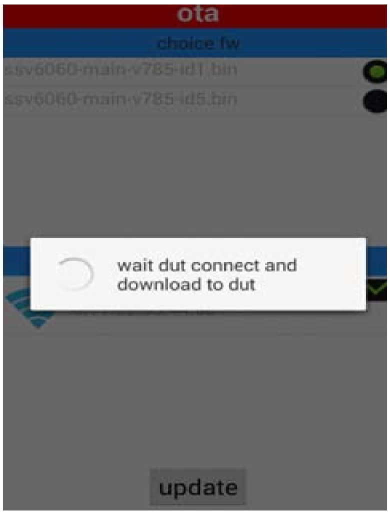
192.168.0.19

0:11:22:33:44:aa

8.2 download F/W and update to dut

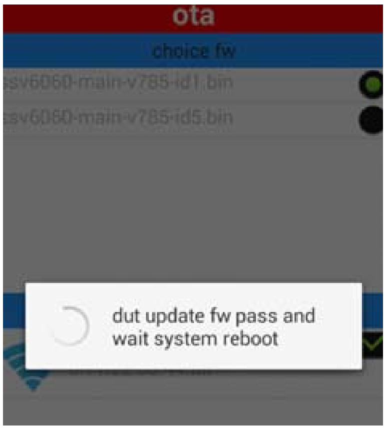


download file to phone



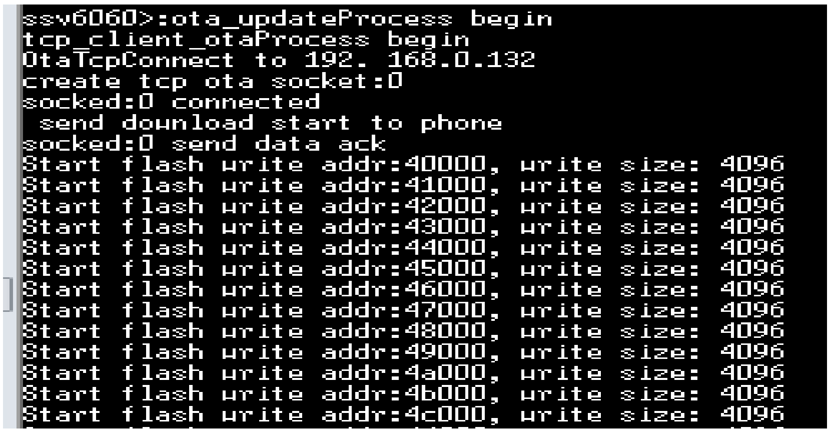
wait dut connect and download to dut

8.3 Update F/W Pass

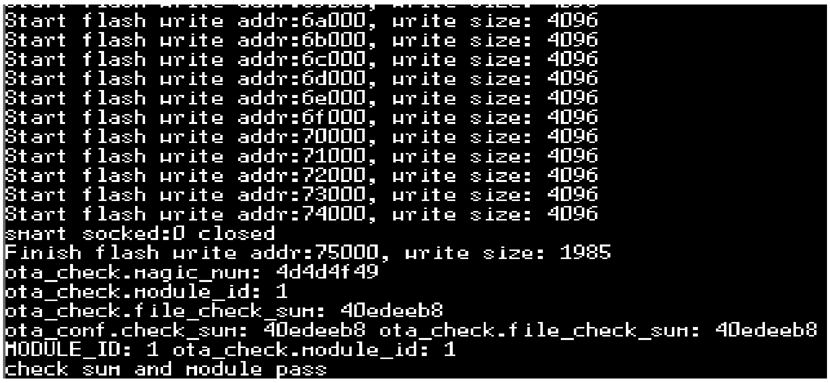


dut update fw pass and wait system reboot

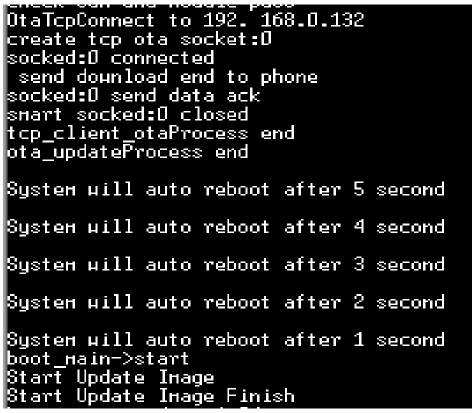
8.4 Console information



```
ssv6060>:ota_updateProcess begin
tcp_client_otaProcess begin
OtaTcpConnect to 192. 168.0.132
create tcp ota socket:0
socked:0 connected
 send download start to phone
socked:0 send data ack
Start flash write addr:40000, write size: 4096
Start flash write addr:41000, write size: 4096
Start flash write addr:42000, write size: 4096
Start flash write addr:43000, write size: 4096
Start flash write addr:44000, write size: 4096
Start flash write addr:45000, write size: 4096
Start flash write addr:46000, write size: 4096
Start flash write addr:47000, write size: 4096
Start flash write addr:48000, write size: 4096
Start flash write addr:49000, write size: 4096
Start flash write addr:4a000, write size: 4096
Start flash write addr:4b000, write size: 4096
Start flash write addr:4c000, write size: 4096
```



```
Start flash write addr:6a000, write size: 4096
Start flash write addr:6b000, write size: 4096
Start flash write addr:6c000, write size: 4096
Start flash write addr:6d000, write size: 4096
Start flash write addr:6e000, write size: 4096
Start flash write addr:6f000, write size: 4096
Start flash write addr:70000, write size: 4096
Start flash write addr:71000, write size: 4096
Start flash write addr:72000, write size: 4096
Start flash write addr:73000, write size: 4096
Start flash write addr:74000, write size: 4096
smart socked:0 closed
Finish flash write addr:75000, write size: 1985
ota_check.magic_num: 4d4d4f49
ota_check.module_id: 1
ota_check.file_check_sum: 40edeeb8
ota_conf.check_sum: 40edeeb8 ota_check.file_check_sum: 40edeeb8
MODULE_ID: 1 ota_check.module_id: 1
check sum and module pass
```



```
OtaTcpConnect to 192. 168.0.132
create tcp ota socket:0
socked:0 connected
 send download end to phone
socked:0 send data ack
smart socked:0 closed
tcp_client_otaProcess end
ota_updateProcess end

System will auto reboot after 5 second

System will auto reboot after 4 second

System will auto reboot after 3 second

System will auto reboot after 2 second

System will auto reboot after 1 second
boot_main->start
Start Update Image
Start Update Image Finish
```

9. FTP download link

ftp://AI6060H@ftp2.acsip.com.tw:2121

主機 =ftp2.acsip.com.tw

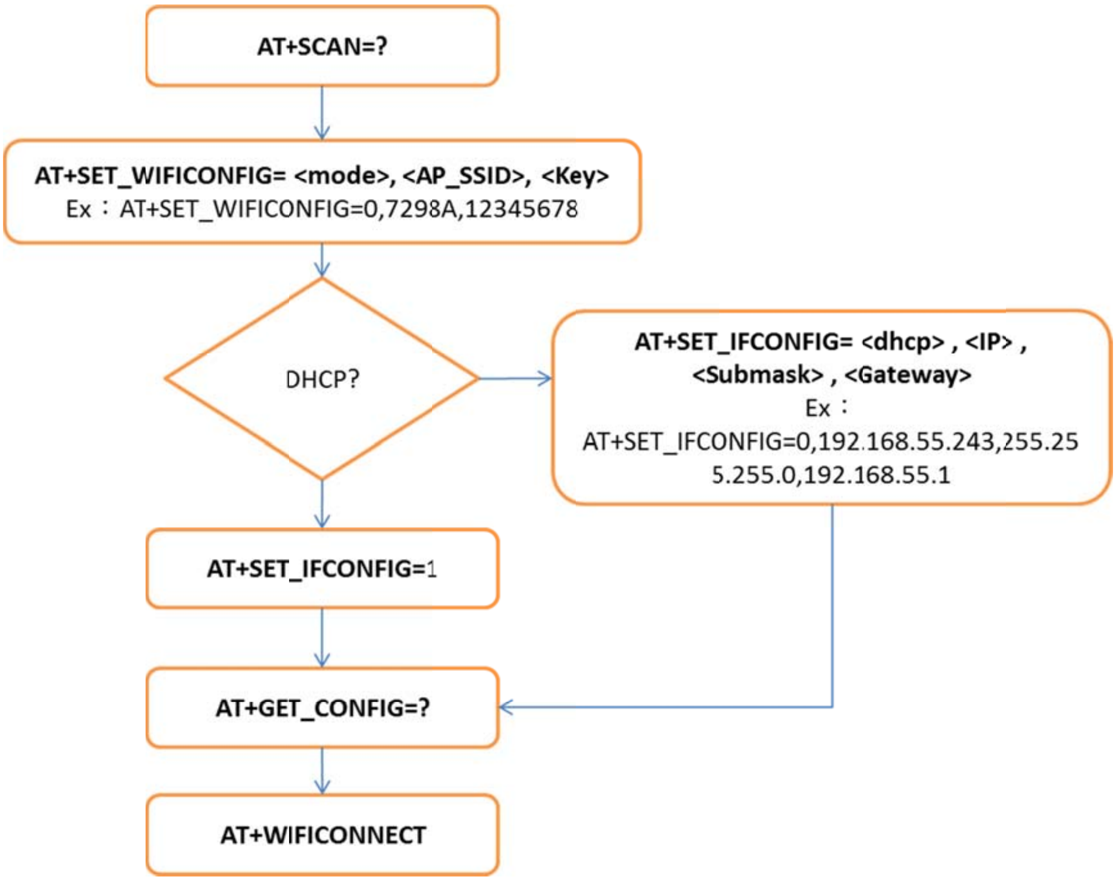
埠號 = 2121

帳號 = AI6060H

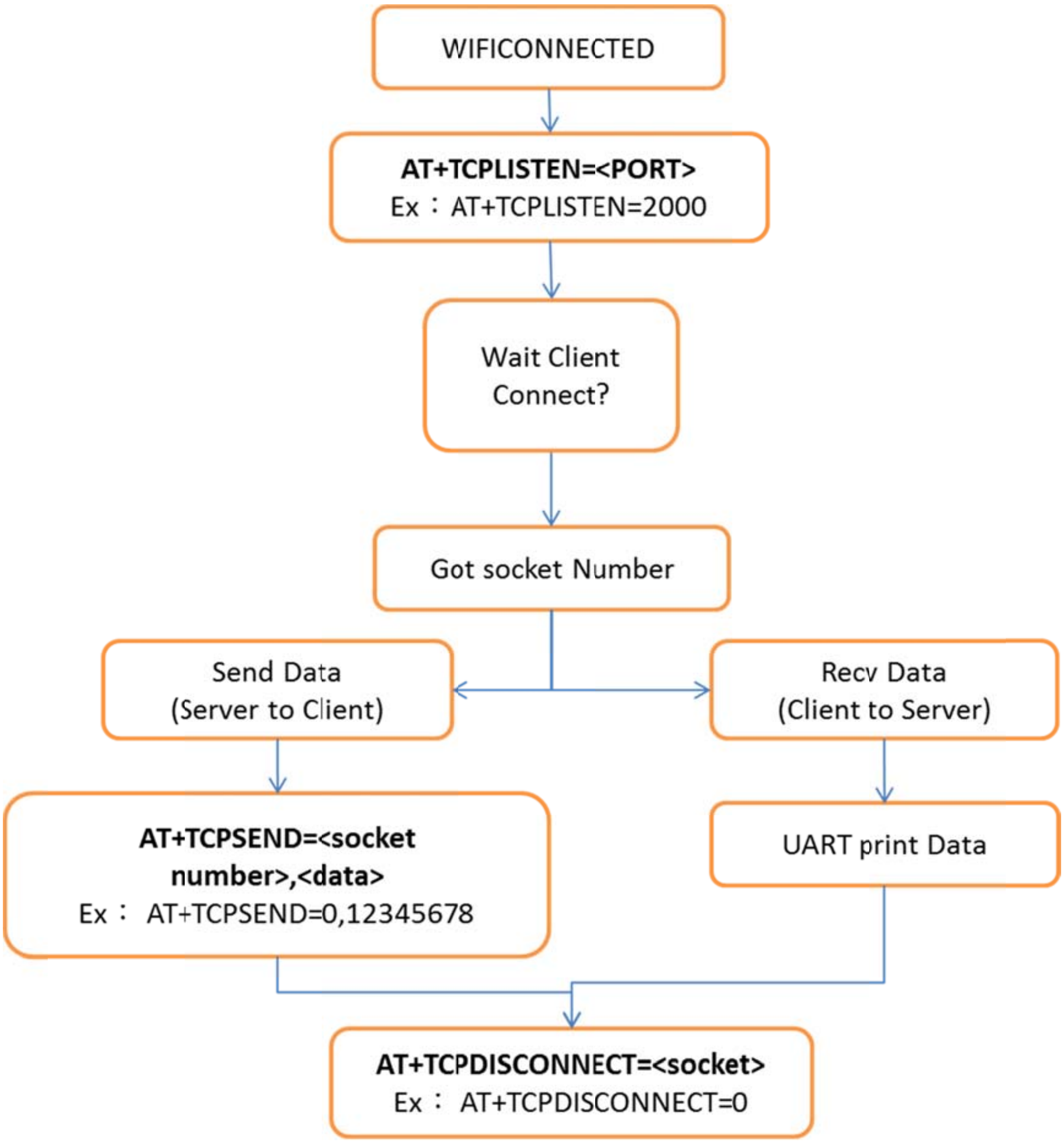
密碼 = bKNNc79A

10. Coding Flowchart:

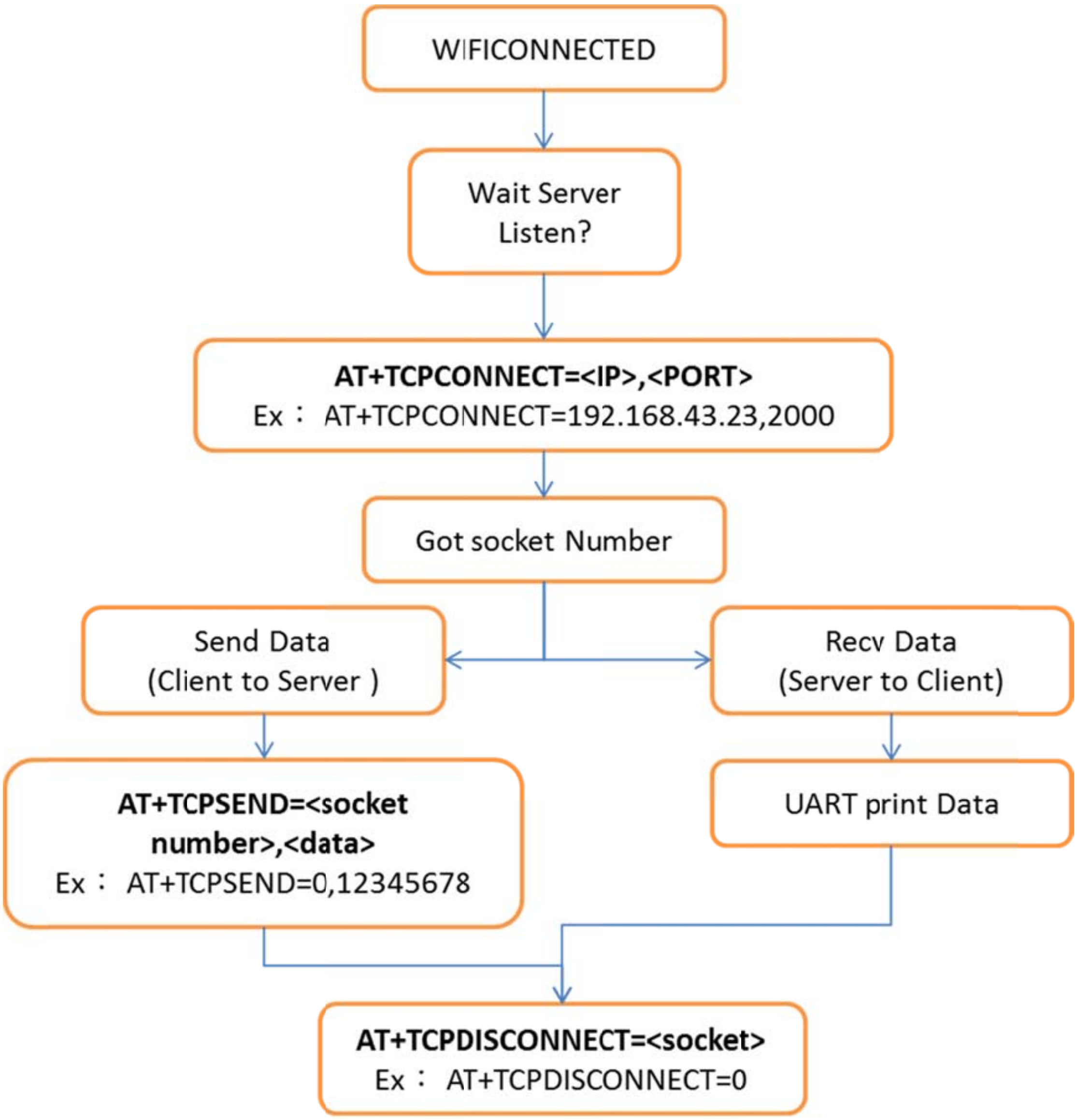
10.1 Wi_Fi Connect flow:


AT+SCAN=?
AT+SET_WIFICONFIG= <mode>, <AP_SSID>, <Key>
Ex：AT+SET_WIFICONFIG=0,7298A,12345678
DHCP?
AT+SET_IFCONFIG= <dhcp> , <IP> , <Submask> , <Gateway>
Ex：
AT+SET_IFCONFIG=0,192.168.55.243,255.255.255.0,192.168.55.1
AT+SET_IFCONFIG=1
AT+GET_CONFIG=?
AT+WIFICONNECT

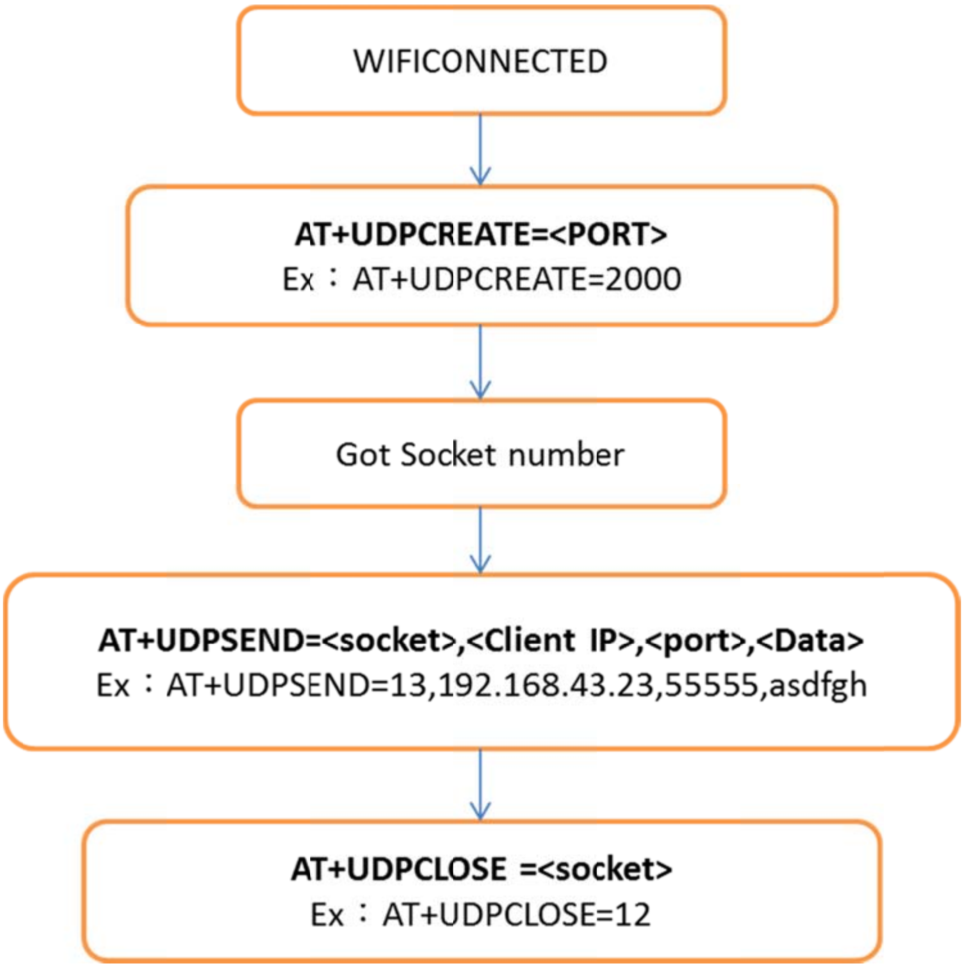
10.2 TCP SERVER @ STA MODE


WIFICONNECTED
AT+TCPLISTEN=<PORT>
Ex：AT+TCPLISTEN=2000
Wait Client Connect?
Got socket Number
Send Data (Server to Client)
Recv Data (Client to Server)
AT+TCPSEND=<socket number>,<data>
Ex： AT+TCPSEND=0,12345678
UART print Data
AT+TCPDISCONNECT=<socket>
Ex： AT+TCPDISCONNECT=0

10.3 TCP CLIENT @ STA MODE


WIFICONNECTED
Wait Server Listen?
AT+TCPCONNECT=<IP>,<PORT>
Ex： AT+TCPCONNECT=192.168.43.23,2000
Got socket Number
Send Data (Client to Server)
Recv Data (Server to Client)
AT+TCPSEND=<socket number>,<data>
Ex： AT+TCPSEND=0,12345678
UART print Data
AT+TCPDISCONNECT=<socket>
Ex： AT+TCPDISCONNECT=0

10.4 UDP @ STA MODE



11. Federal Communication Commission Interference Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FCC Caution:

- Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.
- This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Radiation Exposure Statement:

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

This device is intended only for OEM integrators under the following conditions:

1) The antenna must be installed such that 20 cm is maintained between the antenna and users, and the maximum antenna gain allowed for use with this device is -2.5dBi

2) The transmitter module may not be co-located with any other transmitter or antenna.

As long as 2 conditions above are met, further transmitter test will not be required. However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed

IMPORTANT NOTE: In the event that these conditions can not be met (for example certain laptop configurations or co-location with another transmitter), then the FCC authorization is no longer considered valid and the FCC ID can not be used on the final product. In these circumstances, the OEM integrator will be responsible for re-evaluating the end product (including the transmitter) and obtaining a separate FCC authorization.

End Product Labeling

This transmitter module is authorized only for use in device where the antenna may be installed such that 20 cm may be maintained between the antenna and users. The final end product must be labeled in a visible area with the following: "**Contains FCC ID: 2ADWC-AI6060H**". The grantee's FCC ID can be used only when all FCC compliance requirements are met.

Manual Information To the End User

The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user's manual of the end product which integrates this module. The end user manual shall include all required regulatory information/warning as show in this manual.

Figure 1 below details the standard product marking for all AcSiP Corp. products. Cross reference to the applicable line number and table for a full detail of all the variables.

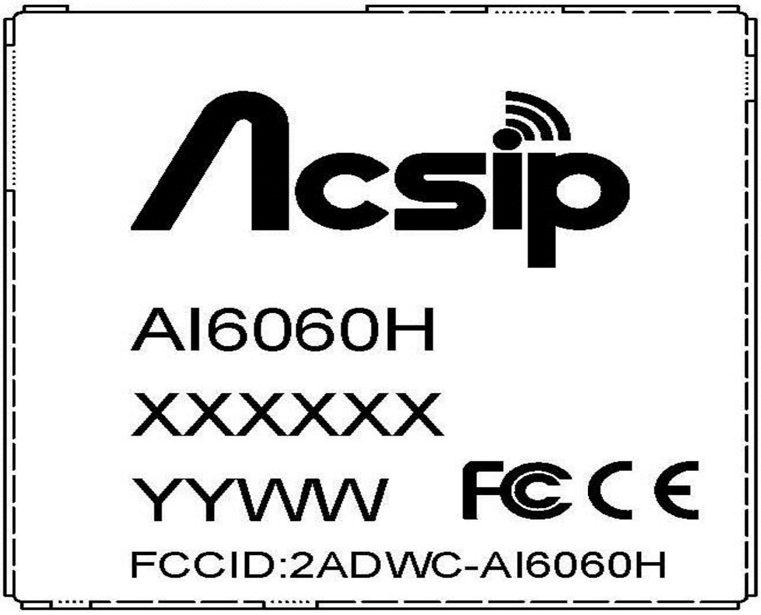


Figure 1 Standard Product Marking Diagram- TOP VIEW