MT7681 IoT FAQ

Version:	0.05
Release date:	2014-7-

Revision History

Date	Revision	Author	Description
04.12.2014	First v0.01	Jinchuan	Initial draft for MT7681 IoT FAQ
05.17.2014	V0.02	Jinchuan	
05.28.2014	V0.03	Jinchuan	Add APK compile Error in Android4.2
06.16.2014	V0.04	Jinchuan	Update "4.3 How to change MAC Address" with Hex input Add "4.6 XIP, OVL mechanism"
07.09.2014	V0.05	Jinchuan	Enable TCP Tx ReTransmit, and Http Client Update: 5.11 take CFG_SUPPORT_MTK_SMNT=1 as the control macro of smart connection function Add: 5.12 Wifi State Machine Flow. Add: 6.9 Enable TCP Tx ReTransmit, and Http Client Add: 4.6 HW Timer1 Add: Capter7: Interface Customization 7.1 PWM Level 7.2 Set UartTx as Interrupt Mode or Poll mode
09.20.2014	V0.05	xThinkLab	Translate into English.

Contents

1 SOURCE CODE COMPILE

1.1 How to Setup AndeSight SDK

Refer to the document: MTK_AndesToolChains_Usage_v0.0*_*******.pdf

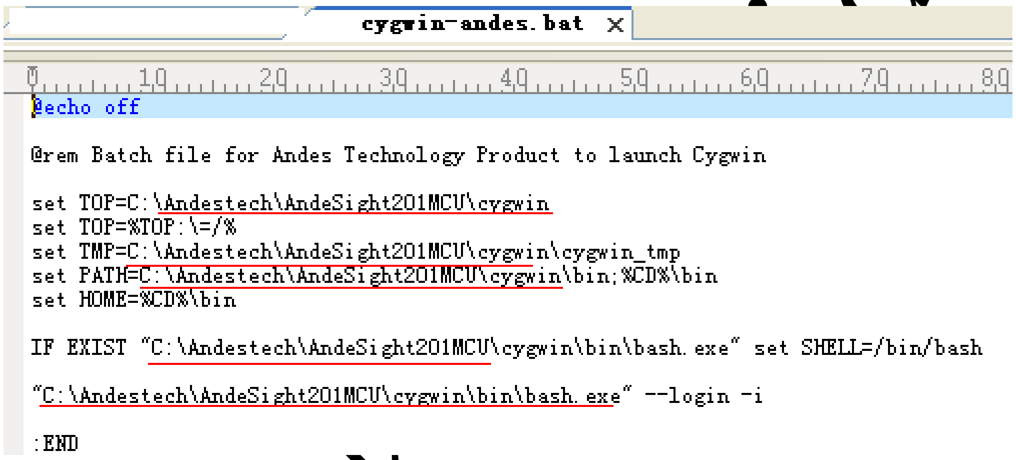
Q1: The AndeSight IDE version I have downloaded is : Andestech\AndeSight201MCU , but you mentioned in document is C:\Andestech\AndeSight14\, any problem ?

A1: No problem , the newest version of 2.01 you downloaded certainly can be used ,and the configure method is same as V1.4

Q2: Refer to method of《MTK_AndesToolChains_Usage_20140212.pdf》,using "nds32le-elf-newlib-V2j" from MTK instead of original toolchain, start and log off。

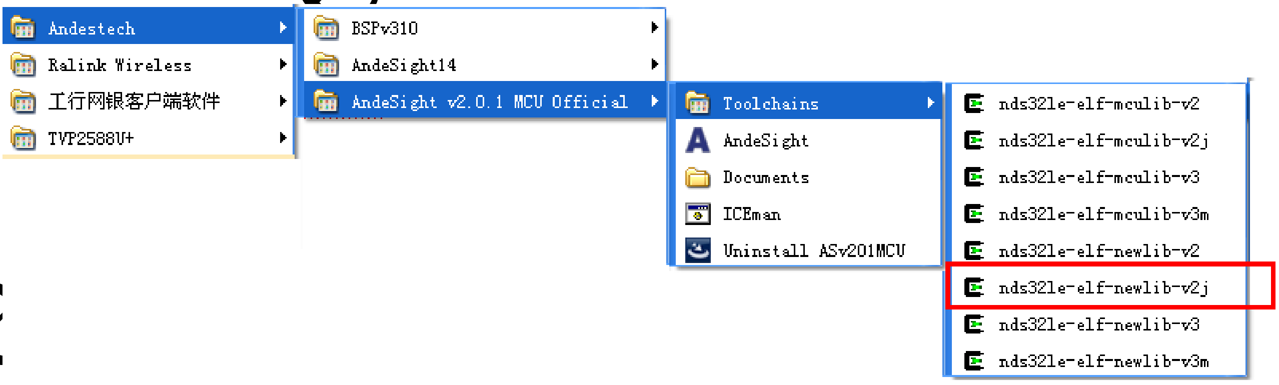
A2: There are no problem to install AndeSight201MCU on XP 32bit OS(someone installed on Win7 64bitOS successfully),use default install path C:\Andestech\AndeSight201MCU\toolchains

If your installation and Toolchain path are same as me ,please refer to the modification screenshot of "cygwin-andes.bat" below .
C:\Andestech\AndeSight201MCU\toolchains\nds32le-elf-newlib-v2j



```
@echo off

@rem Batch file for Andes Technology Product to launch Cygwin

set TOP=C:\Andestech\AndeSight201MCU\cygwin
set TOP=%TOP:\=/%
set TMP=C:\Andestech\AndeSight201MCU\cygwin\cygwin_tmp
set PATH=C:\Andestech\AndeSight201MCU\cygwin\bin;%CD%\bin
set HOME=%CD%\bin

IF EXIST "C:\Andestech\AndeSight201MCU\cygwin\bin\bash.exe" set SHELL=/bin/bash

"C:\Andestech\AndeSight201MCU\cygwin\bin\bash.exe" --login -i

:END
```

Then ,Click the link marked with red box below and start it .



1.2 How to create new project in AndeSight / How to import MTK7681 SDK to AndeSight?

We ONLY use the cygwin compile environment of AndeSight SDK,but not create new project with it .

1.3 How to compile source code for Recovery FW , Sta FW, AP FW

make b=0 clean;make b=0 -> create recovery bin

make b=1 clean;make b=1 -> create sta bin

make b=2 clean;make b=2 -> create ap bin

1.4 Ap/Sta/recovery/all.bin are created after compile done， which binary we can use

The Readme-ForEachBinDescript.xlsx file in MT7681_IoT_Package_v1.10\Src folder will show you the binary code .

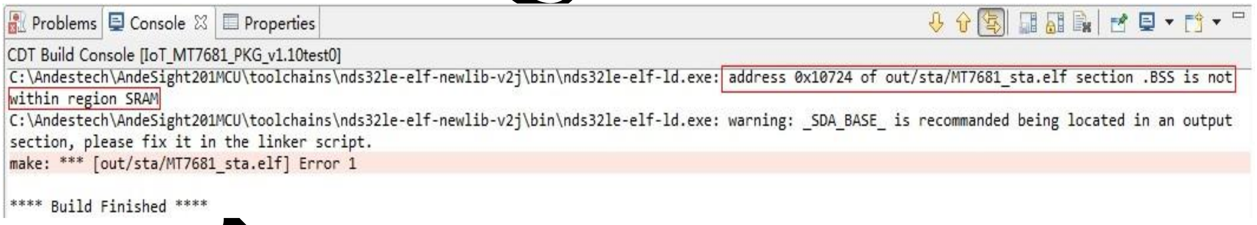
compile done then generate : MT7681_sta.bin， MT7681_recovery_old.bin， MT7681_ap.bin ， those files will Merge (loader_0322_94973.bin， MT7681E2_EEPROM layout_20140330.bin) as one binary file , MT7681_all.bin. MT7681_all.bin is the all in one binary that can flash via the Flash Writer .

The generated file include :
MT7681_recovery_header.bin,MT7681_sta_header.bin,MT7681_ap_header.bin,all these binary firmware marked with "_header" which can be used as upgrading files via UART interface .

The detail of upgrading method and the difference between ***_Header_**.bin and ***.bin, please refer to
MT7681_IoT_Package_v1.10\Doc\MT7681_Uart_Firwmare_Upgrade_v0.0*_********.pdf .

1.5 Compile Source Code fail -- .BSS is not within region SRAM

It shows the program is out of ram space



```
CDT Build Console [IoT_MT7681_PKG_v1.10test0]
C:\Andestech\AndeSight201MCU\toolchains\nds32le-elf-newlib-v2j\bin\nds32le-elf-ld.exe: address 0x10724 of out/sta/MT7681_sta.elf section .BSS is not
within region SRAM
C:\Andestech\AndeSight201MCU\toolchains\nds32le-elf-newlib-v2j\bin\nds32le-elf-ld.exe: warning: _SDA_BASE_ is recommanded being located in an output
section, please fix it in the linker script.
make: *** [out/sta/MT7681_sta.elf] Error 1

**** Build Finished ****
```

In Package v1.10 SDK :

In default configuration,flag_sta.mk (for Station mode)， the free space for user is 11KB

In default configuration, flag_ap.mk (for AP mode)， the free space for user is 13KB

2 FW UPGRADE

2.1 FW upgrade method

1: FW upgrade by Flash Writer

2: FW upgrade by Uart

Just like descript in "**<u>1.4</u>**" , please refer to MT7681_Uart_Firwmare_Upgrade_v0.0*_********.pdf .

* Not support FOTA yet， but developers can complete function using current API

2.2 How to change AP mode and Station Mode

using AT#FLASH -s98305 –v* command ， change the value of Flash Offset: 0x18001, shutdown and reboot ,

[0x18001]=[0x00] > > Boot as STA mode,

[0x18001]=[0x01] > > Boot as AP mode

Example: Switching to AP Mode is simple , just modify the value of Flash Offset: 0x18001 to 1

Step1: flash the MT7681_all_v1.10.bin via Flash Writer to Flash

Step2: Power on, display the message below as STA Mode:

```
==> Recovery Mode
<== RecoveryMode
(-)
SM=0, Sub=0
SM=1, Sub=0
[WTask]9811
```

Step3: Read BootIndex value of 0x18001 via AT#Flash command

```
AT#FLASH -r98305
[0x18001]=[0x00]  > > Boot as STA mode,     if [0x18001]=[0x01], Boot as AP mode
```

Step4: Modify BootIndex value of 0x18001 via AT#Flash command to 1

```
AT#FLASH -s98305 -v1
```

Step5: Power on MT7681 again, it will boot in AP mode,

SSID name 为"MT7681_Softap", 只 Support Open Mode

```
==> Recovery Mode
<== RecoveryMode
(-)
APStartUp ok
Start AP ...
[WTask]9318
[WTask]14322
```

If smart phone connect MT7681 at the moment , it will show message below :

```
Assoc request sanity success
i = 5, j = 0
client ip addr: 192.168.81.2
[WTask]39372
```

2.3 Why there is a Recovery Mode

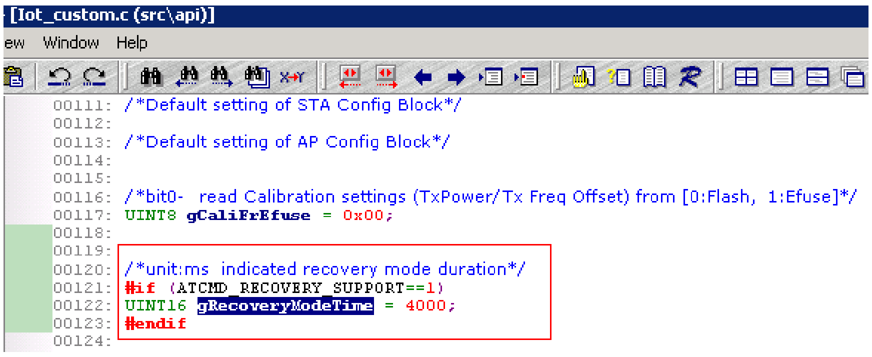
The main purpose is Uart FW Upgrade and Production Calibration

It will get into Recovery Mode automatically after powering on or Reboot, and wait 4s to receive command , we can enter AT#UpdateFW to start Uart FW upgrade Process within 4s.we can enter AT#ATECAL -S to get into Calibration Mode and start Tx/Rx Calibration

We should consider how the system recover , if the STAFW is failing to update, at this time, the existence of Recovery Mode is very important. The wait time of Recovery Mode is 4s, because recently the SDK v1.0 can not be connected to Uart Rx when the system is start, and there must be enough time to make sure the AT#UpdateFW instruction is input after power on.

2.4 How to set recovery mode duration

Now in the mode of recovery, if there are not any operations, after 4s, it will quit from recovery mode. During the 4s, it can be controlled by set variable in v1.2 SDK.



```
/*Default setting of STA Config Block*/

/*Default setting of AP Config Block*/


/*bit0-  read Calibration settings (TxPower/Tx Freq Offset) from [0:Flash, 1:Efuse]*/
UINT8 gCaliFrEfuse = 0x00;


/*unit:ms  indicated recovery mode duration*/
#if (ATCMD_RECOVERY_SUPPORT==1)
UINT16 gRecoveryModeTime = 4000;
#endif

```

3 APK INSTALL AND USAGE

3.1 Install "IoTManager_v0.94_1_android4.0.apk", show "some problems when packages are resolved"

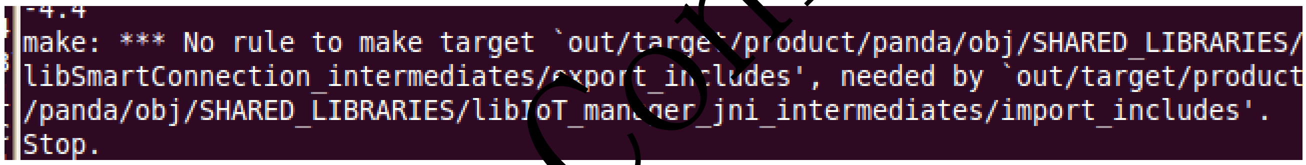
Install "IoTManager_v0.94_1_android4.0.apk", show "some problems when packages are resolved", IoTManager_v0.94_1_android4.0.apk is for Android 4.0 above.

APK SourceCode is also in the Package v1.10, you can compile the corresponding APK for demo.

3.2 Compile APK Source Code failure in Android Codebase 4.2

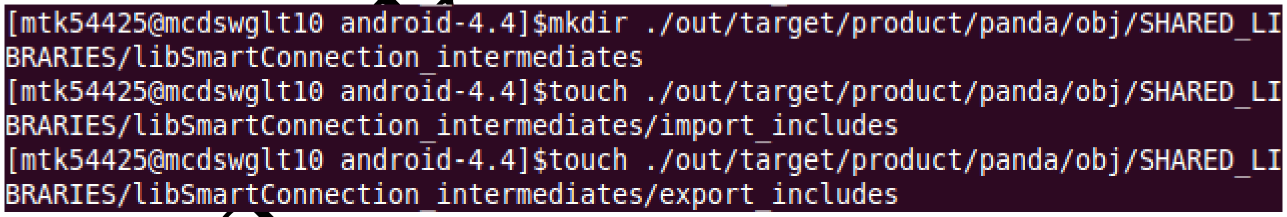
There is no source code for release SmartConnection.lib in the APK, so there may be the following problems when you compile in the codebase for android4.2.

1:it hints there is no export includes



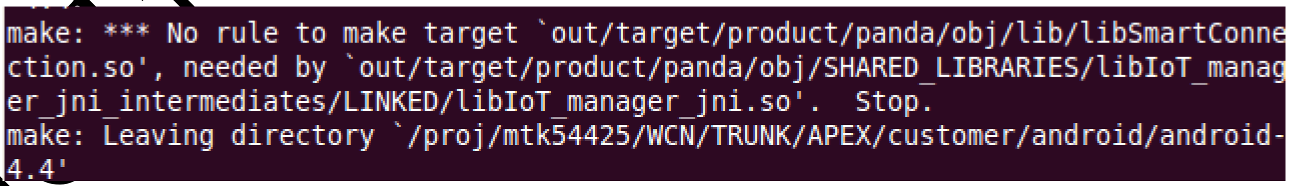
```
-4.4
make: *** No rule to make target `out/target/product/panda/obj/SHARED_LIBRARIES/
libSmartConnection_intermediates/export_includes', needed by `out/target/product
/panda/obj/SHARED_LIBRARIES/libIoT_manager_jni_intermediates/import_includes'.
Stop.
```

It is because there is no lib of buildSmartConnection, the solution is to create two files manually.



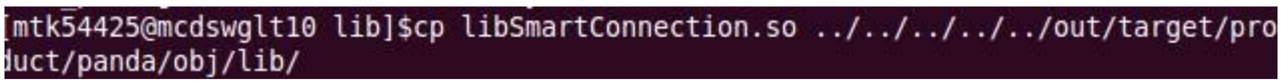
```
[mtk54425@mcdswglt10 android-4.4]$mkdir ./out/target/product/panda/obj/SHARED_LI
BRARIES/libSmartConnection_intermediates
[mtk54425@mcdswglt10 android-4.4]$touch ./out/target/product/panda/obj/SHARED_LI
BRARIES/libSmartConnection_intermediates/import_includes
[mtk54425@mcdswglt10 android-4.4]$touch ./out/target/product/panda/obj/SHARED_LI
BRARIES/libSmartConnection_intermediates/export_includes
```

2: it hints there is no libSmartConnection.so



```
make: *** No rule to make target `out/target/product/panda/obj/lib/libSmartConne
ction.so', needed by `out/target/product/panda/obj/SHARED_LIBRARIES/libIoT_manag
er_jni_intermediates/LINKED/libIoT_manager_jni.so'.  Stop.
make: Leaving directory `/proj/mtk54425/WCN/TRUNK/APEX/customer/android/android-
4.4'
```

The solution is to copy the IoTManager/lib/libSmartConnection.so to the corresponding path, as following:



```
[mtk54425@mcdswglt10 lib]$cp libSmartConnection.so ../../../../../out/target/pro
duct/panda/obj/lib/
```

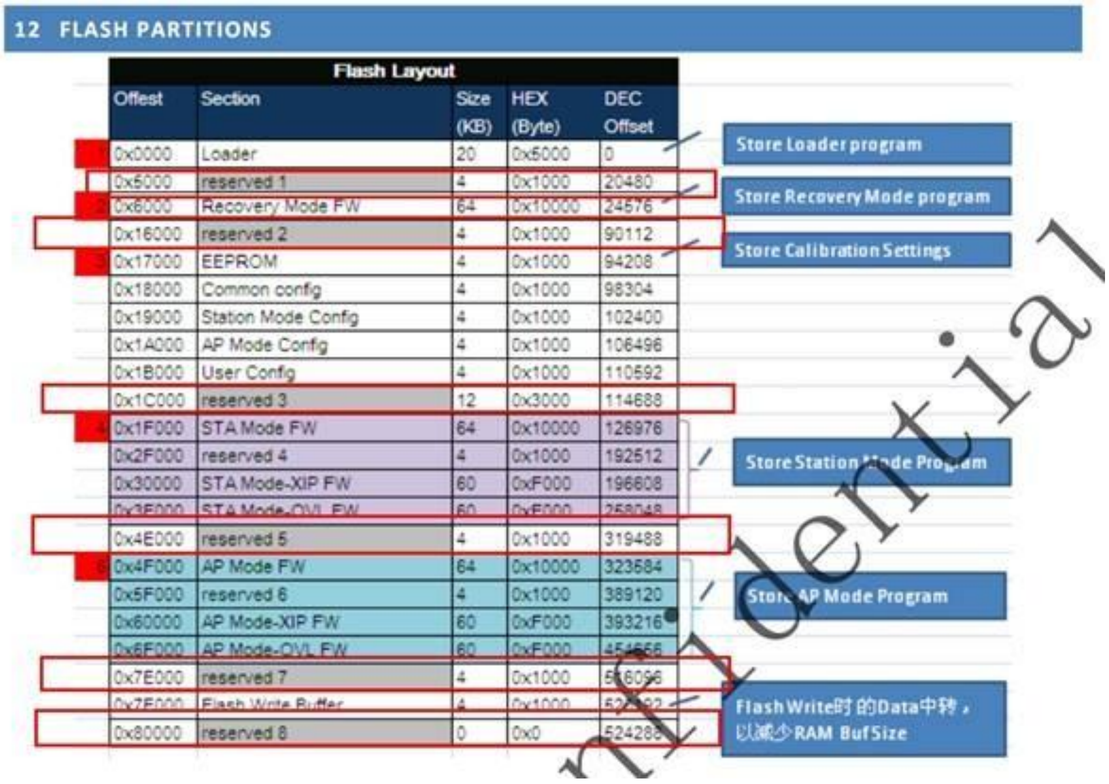
4 SYSTEM CODING

4.1 Printf_High() /DBGPRINTF_HIGH() ,the switch of log output

In the V1.2SDK, the Boolean global variable is added to Iot_custom.c to control if the LOG is printed.

```
/*TRUE: Printf_High()/DBGPRINT_HIGH() is enabled,   FALSE: Printf_High/DBGPRINT_HIGH()  is disabled*/
BOOLEAN          PRINT_FLAG = TRUE;
```

4.2 The usage of "reserved" region in flash？ is possible to store other data in it


12 FLASH PARTITIONS
Store Loader program
Store Recovery Mode program
Store Calibration Settings
Store Station Mode Program
Store AP Mode Program
Flash Write時的Data中转，以減少RAM Buf Size

Flash Layout

Offest	Section	Size (KB)	HEX (Byte)	DEC Offset
0x0000	Loader	20	0x5000	0
0x5000	reserved 1	4	0x1000	20480
0x6000	Recovery Mode FW	64	0x10000	24576
0x16000	reserved 2	4	0x1000	90112
0x17000	EEPROM	4	0x1000	94208
0x18000	Common config	4	0x1000	98304
0x19000	Station Mode Config	4	0x1000	102400
0x1A000	AP Mode Config	4	0x1000	106496
0x1B000	User Config	4	0x1000	110592
0x1C000	reserved 3	12	0x3000	114688
0x1F000	STA Mode FW	64	0x10000	126976
0x2F000	reserved 4	4	0x1000	192512
0x30000	STA Mode-XIP FW	60	0xF000	196608
0x3F000	STA Mode-OVL FW	60	0xF000	258048
0x4E000	reserved 5	4	0x1000	319488
0x4F000	AP Mode FW	64	0x10000	323584
0x5F000	reserved 6	4	0x1000	389120
0x60000	AP Mode-XIP FW	60	0xF000	393216
0x6F000	AP Mode-OVL FW	60	0xF000	454656
0x7E000	reserved 7	4	0x1000	516096
0x7F000	Flash Write Buffer	4	0x1000	520192
0x80000	reserved 8	0	0x0	524288

Reserved is mainly used to isolated all area.

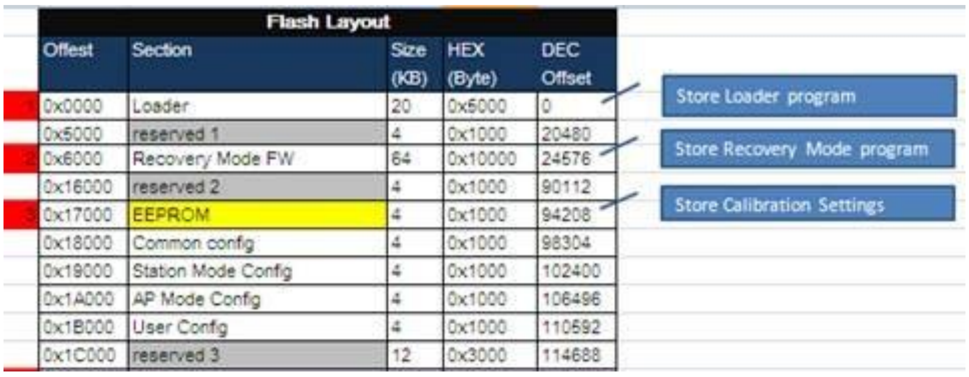
If you want extra space in the flash, you can used the reserved in 0x1C000, total 12KB

But please do not use the 0x1F000, STA FW .

And the Flash size which Chip can access to is 1MB, now what the Flash Layout define is 512KB. If your product is adopted 1MB Flash, the [0x8000 ~ 0xFFFFF] can also be used.

4.3 How to change MAC Address

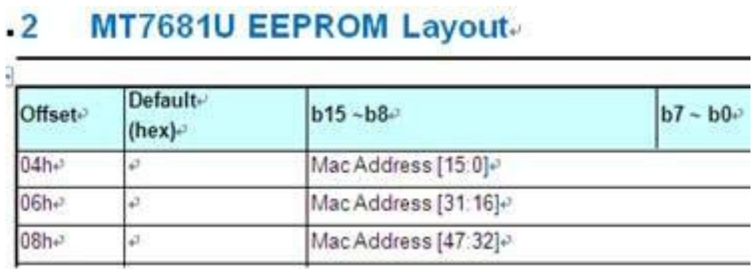
By default, Mac address is stored in EEPROM area.


Store Loader program
Store Recovery Mode program
Store Calibration Settings

Flash Layout

Offest	Section	Size (KB)	HEX (Byte)	DEC Offset
0x0000	Loader	20	0x5000	0
0x5000	reserved 1	4	0x1000	20480
0x6000	Recovery Mode FW	64	0x10000	24576
0x16000	reserved 2	4	0x1000	90112
0x17000	EEPROM	4	0x1000	94208
0x18000	Common config	4	0x1000	98304
0x19000	Station Mode Config	4	0x1000	102400
0x1A000	AP Mode Config	4	0x1000	106496
0x1B000	User Config	4	0x1000	110592
0x1C000	reserved 3	12	0x3000	114688

EEPROM Layout is described in the MT7681U-EEPROM Content _***.docx, the MAC Address is exist in the EEPROM 0x04~0x09.


.2 MT7681U EEPROM Layout

Offset	Default (hex)	b15 ~b8	b7 ~ b0
04h		Mac Address [15:0]	
06h		Mac Address [31:16]	
08h		Mac Address [47:32]	

The AT Command for reading Flash MAC address:

```
AT#FLASH –r94212          LOG output  [0x17004]=[0x00]
AT#FLASH –r94213          LOG output  [0x17005]=[0x0c]
AT#FLASH –r94214          LOG output  [0x17006]=[0x43]
AT#FLASH –r94215          LOG output  [0x17007]=[0x26]
AT#FLASH –r94216          LOG output  [0x17008]=[0x60]
AT#FLASH –r94217          LOG output  [0x17009]=[0x40]
```

From v1.30 SDK, it support AT#FLASH –r0x17004, hexadecimal format.

The AT Command for Flash MAC address setting :

```
AT#FLASH –s94212   -vX   x is the set point
AT#FLASH –s94213   -vX   x is the set point
AT#FLASH –s94214   -vX   x is the set point
AT#FLASH –s94215   -vX   x is the set point
AT#FLASH –s94216   -vX   x is the set point
AT#FLASH –s94217   -vX   x is the set point
```

The above -r, -s, -v are all decimal.

From v1.30 SDK, it support AT#FLASH -s0x17004 -v0x0c, hexadecimal format.

After set the parameter, reboot the system, and then the it will use the new MAC address, and assign to gCurrentAddress.

4.4 Open Macro:CFG_SUPPORT_TCPIP, Compile Error

Since we have opened the SourceCode of uIP.

ATCMD_TCPIP_SUPPORT is not maintained, so it is closed.

4.5 XIP, Overlay Mechanism

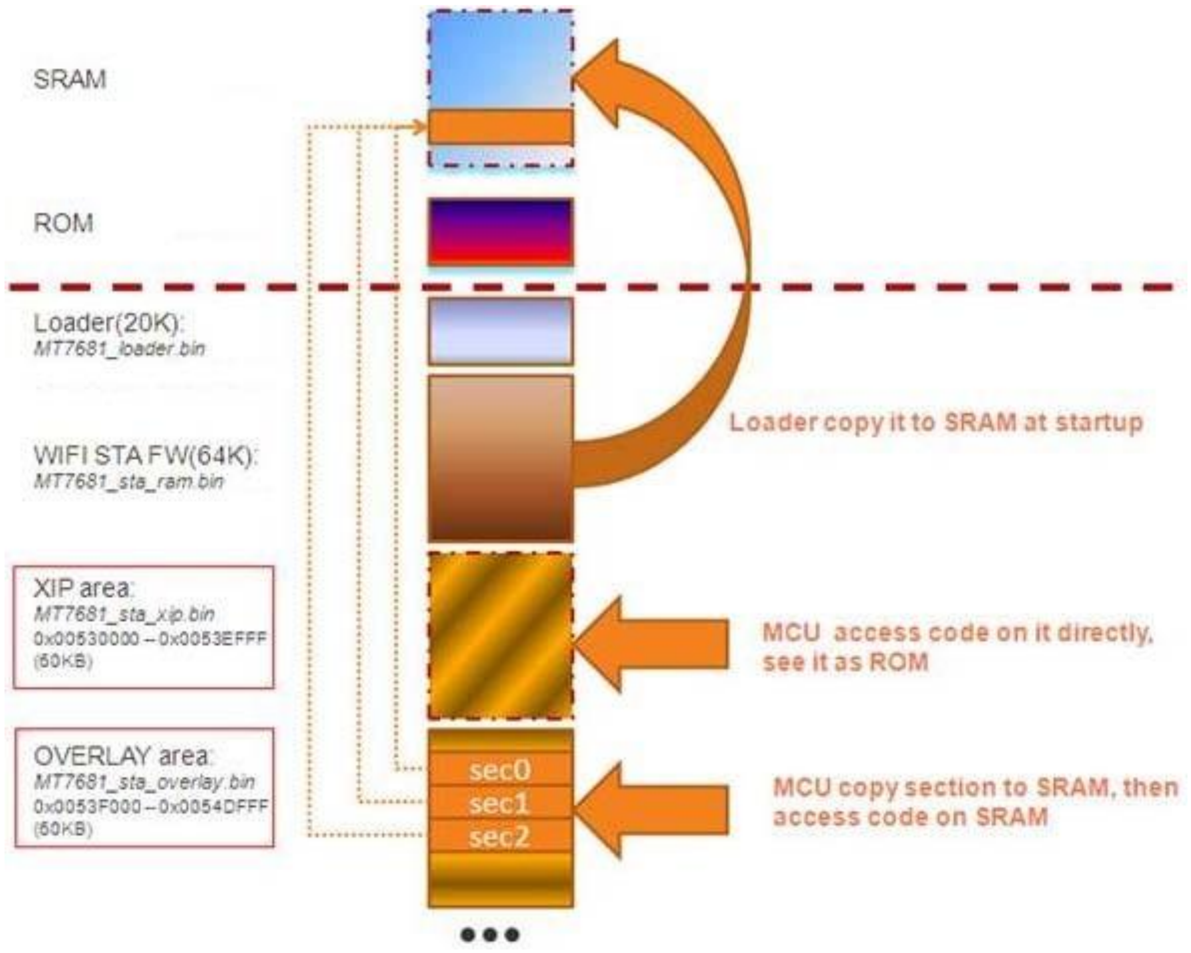
Now some customers need customized code, it may exceed the range of Ram, so there are some introductions about the two mechanisms, XIP and OVL.

The steps are as follows:

* The Function defined as XIP will run on the Flash.
The whole function will not have conflict.
The XIP function definition is at the function statement, and add XIP_ATTRIBUTE(".xipsec0"),.
But it is better not define XIP in the following conditions
1 r/w function for the flash
2: the function need high real-time/effectiveness.

* The function is defined as Overlay, because several functions will used the same RAM, and there will have conflict. We should do function review one by one.

To make sure this function is not be used by other overlay function, or called by each other, it is more and more difficult later, so suggest the customers not use this method.



4.6 HW timer1 EINT Freq Adjustment

The Frequency for hardware timer 1 interrupt, Range [1~10]

#define TICK_HZ_HWTIMER1 10 /*T = 1/TICK_HZ_HWTIEMR1*/

Above example: IoT_Cust_HW_Timer1_Hdlr will be triggered every 100ms (That is T=1/10)

5 CONNECTION DEVELOPMENT

5.1 MT7681 Support mode and Bandwidth

In the Package v1.10, channel choose API: IoT_Cmd_Set_Channel

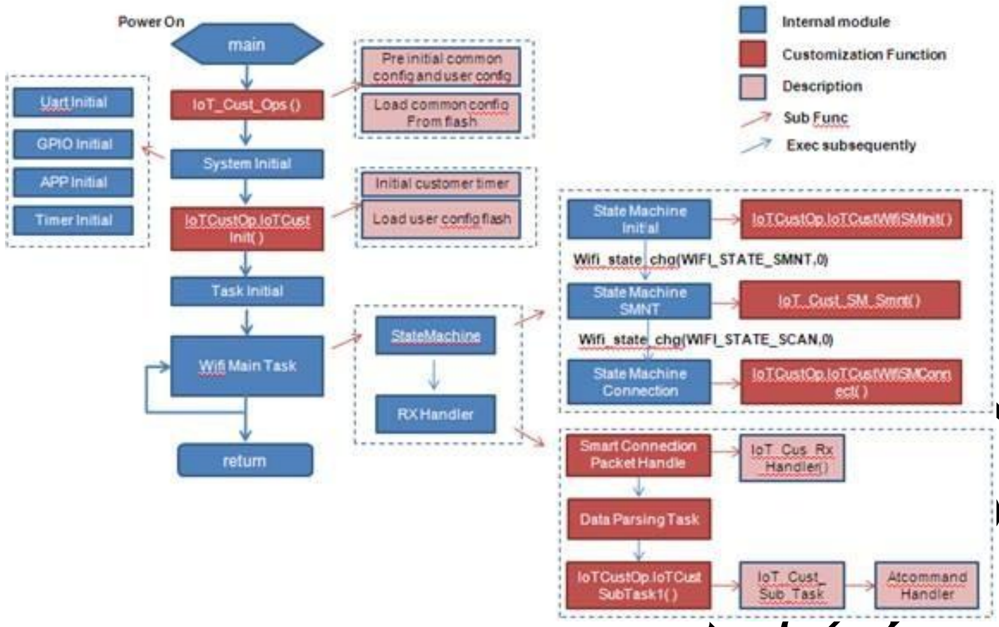
For Mode and Bandwidth: now Support 80.211 b/g ， BW_20M

5.2 Generally, C has entry point, main(), where is entry poi of a IoT application

yes, we have main(), but we do not open the whole main function

but we built hook function in order to make customers compile their own program in the iot_custom.c

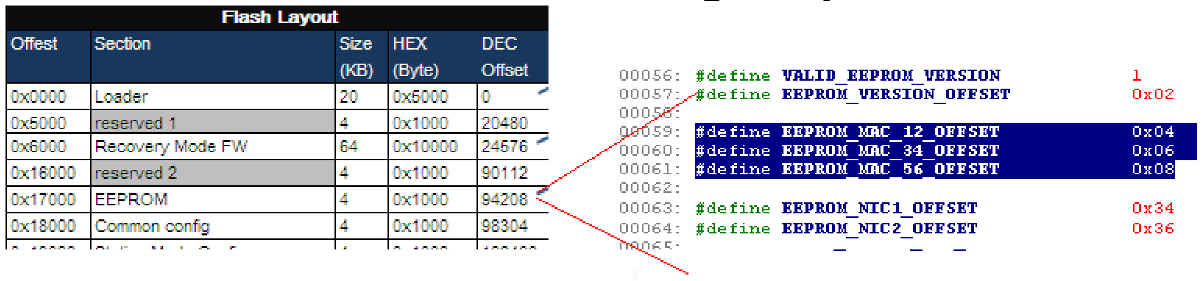
refer to MT7681_IoT_WIFI_Firmware_Programming_Guide_v0.0*.pdf 的 Section 5 “Customer Hook Function”



5.3 How to get mt7681 MAC address

The MAC address of 7681 is stored on Flash offset [0x17004~0x17009],

and this MAC shall be set to global parameter gCurrentAddress[MAC_ADDR_LEN]



Flash Layout				
Offest	Section	Size (KB)	HEX (Byte)	DEC Offset
0x0000	Loader	20	0x5000	0
0x5000	reserved 1	4	0x1000	20480
0x6000	Recovery Mode FW	64	0x10000	24576
0x16000	reserved 2	4	0x1000	90112
0x17000	EEPROM	4	0x1000	94208
0x18000	Common config	4	0x1000	98304
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

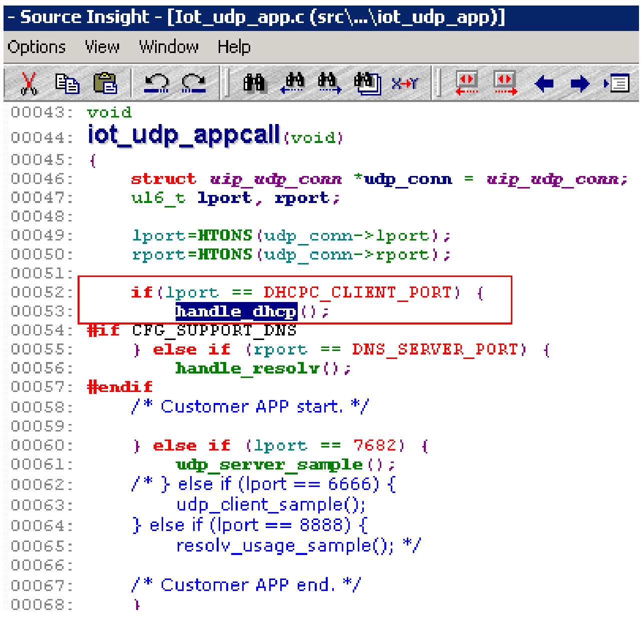
```
#define VALID_EEPROM_VERSION          1
#define EEPROM_VERSION_OFFSET         0x02

#define EEPROM_MAC_12_OFFSET          0x04
#define EEPROM_MAC_34_OFFSET          0x06
#define EEPROM_MAC_56_OFFSET          0x08

#define EEPROM_NIC1_OFFSET            0x34
#define EEPROM_NIC2_OFFSET            0x36
```

5.4 In the mode of sta connected to ap, what can I do to know if the connection is ok?

A1: Now the system has a process that Initial-> Smart Connection -> Scan -> Auth-> Assoc -> 4 way -> Connected, the state will set to pIoTMlme->CurrentWifiState . so it can control the flow of state machine, if pIoTMlme->CurrentWifiState = WIFI_STATE_CONNED (6), it means the connection of AP is ok, the next step is to acquire IP.

5.5 How can the IP acquired automatically, is it used the dhcpc in the uip?

A2: After get into the connected state, the TCP/UDP is interaction by calling uip_periodic_timer, it will do DHCPC to acquire IP, the file and functions are as follows:



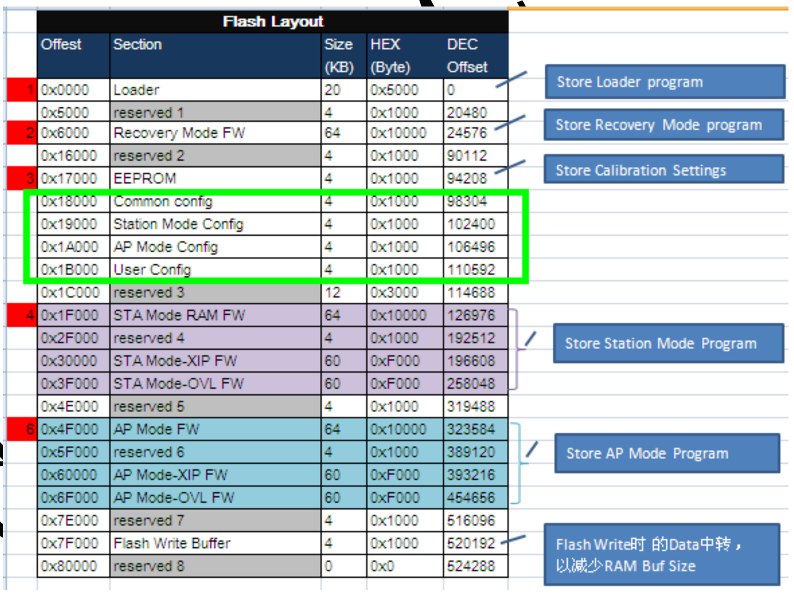
```
void
iot_udp_appcall(void)
{
    struct uip_udp_conn *udp_conn = uip_udp_conn;
    u16_t lport, rport;

    lport=HTONS(udp_conn->lport);
    rport=HTONS(udp_conn->rport);

    if(lport == DHCPC_CLIENT_PORT) {
        handle_dhcp();
#if CFG_SUPPORT_DNS
    } else if (rport == DNS_SERVER_PORT) {
        handle_resolv();
#endif
    /* Customer APP start. */

    } else if (lport == 7682) {
        udp_server_sample();
    /* } else if (lport == 6666) {
        udp_client_sample();
    } else if (lport == 8888) {
        resolv_usage_sample(); */

    /* Customer APP end. */
    }
```

5.6 How to reset IP/SSID and other parameter to default

Now the code has offer the AT#Default function.

For parameter set, we can save it in the green format of flash as follows,
In the MT7681_IoT_WIFI_Firmware_Programming_Guide_v0.0*.pdf Section “11 FLASH PARTITIONS “, there are the definition of every Byte.

Otherwise, in the iot_customer.c, The common cfg, AP cfg, user cfg parameter are completed in load and reset functions. For the station cfg is initialized in reset_sta_cfg() function .



	Flash Layout				
	Offest	Section	Size (KB)	HEX (Byte)	DEC Offset
1	0x0000	Loader	20	0x5000	0
	0x5000	reserved 1	4	0x1000	20480
2	0x6000	Recovery Mode FW	64	0x10000	24576
	0x16000	reserved 2	4	0x1000	90112
3	0x17000	EEPROM	4	0x1000	94208
	0x18000	Common config	4	0x1000	98304
	0x19000	Station Mode Config	4	0x1000	102400
	0x1A000	AP Mode Config	4	0x1000	106496
	0x1B000	User Config	4	0x1000	110592
	0x1C000	reserved 3	12	0x3000	114688
4	0x1F000	STA Mode RAM FW	64	0x10000	126976
	0x2F000	reserved 4	4	0x1000	192512
	0x30000	STA Mode-XIP FW	60	0xF000	196608
	0x3F000	STA Mode-OVL FW	60	0xF000	258048
	0x4E000	reserved 5	4	0x1000	319488
6	0x4F000	AP Mode FW	64	0x10000	323584
	0x5F000	reserved 6	4	0x1000	389120
	0x60000	AP Mode-XIP FW	60	0xF000	393216
	0x6F000	AP Mode-OVL FW	60	0xF000	454656
	0x7E000	reserved 7	4	0x1000	516096
	0x7F000	Flash Write Buffer	4	0x1000	520192
	0x80000	reserved 8	0	0x0	524288

Store Loader program
Store Recovery Mode program
Store Calibration Settings
Store Station Mode Program
Store AP Mode Program
Flash Write时 的Data中转，以减少RAM Buf Size

5.7 There is no “mem_alloc, mem_free” API in SDK v1.10

Because not use the operation system , we do not have the whole mem_alloc and mem_free API
Now the function of malloc free is definite, it can not be called by nesting, as follows:

malloc(A) -> free(A) -> malloc(B) -> free(B) === OK
malloc(A) -> malloc(B) -> free(A) -> free(B) === NG

so the API can not be used widely, but the MTK and customers all can develop code, so suggest you not use this API.

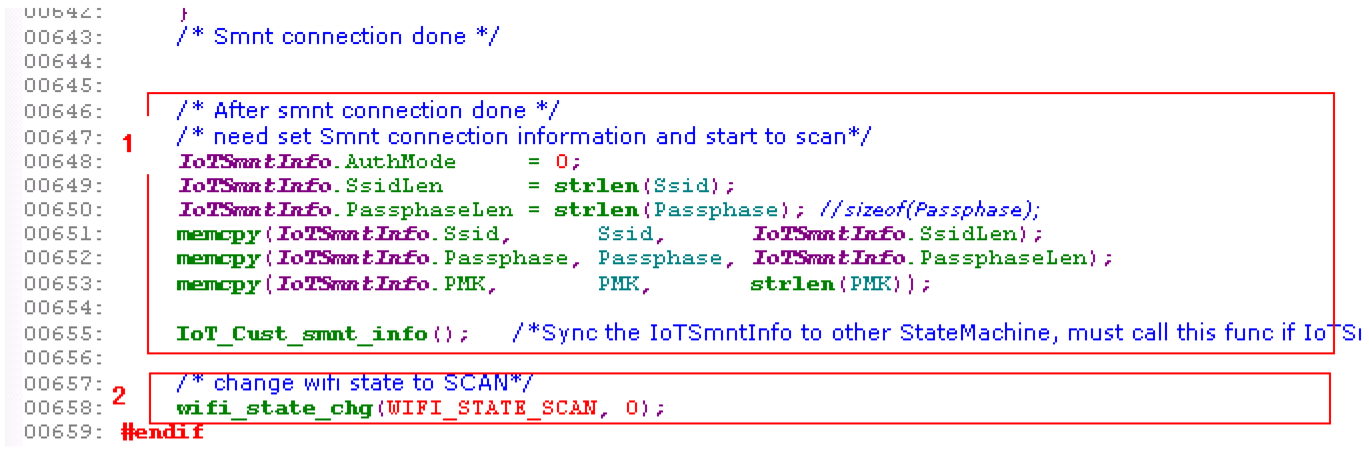
Now for the most cases, the buffer use global or local
array to apply.
So for the program, we will consider more the size of
array.

5.8 Structure _WIFI_STATE Introduction，The means for each state

```
typedef enum _WIFI_STATE{
WIFI_STATE_INIT = 0, －>//Initialize the wifi state machine, read sta cfg set in the
                                        flash, if it read an effective SSID, Password, PMK,
                                        AuthMode, then jump to SCAN stage, or it will jump
                                        to SMNT stage.
WIFI_STATE_SMTCNT, -> //Run smart connection, collect sta cfg set.
WIFI_STATE_SCAN, －>//By sta cfg acquired in the init or smnt stage to scan the ssid,
                                      and then fix channel.
WIFI_STATE_AUTH, －>//Send Auth Request frame to ssid AP and acquire Auth
                                      Response
WIFI_STATE_ASSOC, －>//Send Auth Request frame to ssid AP and acquire Assoc
                                      Response
WIFI_STATE_4WAY, －> //With ssid AP do 4 －way handshark to generate GTK,PTK
WIFI_STATE_CONNED －>          //Output dhcpc and acquire IP assigned by AP.
}WIFI_STATE;
```

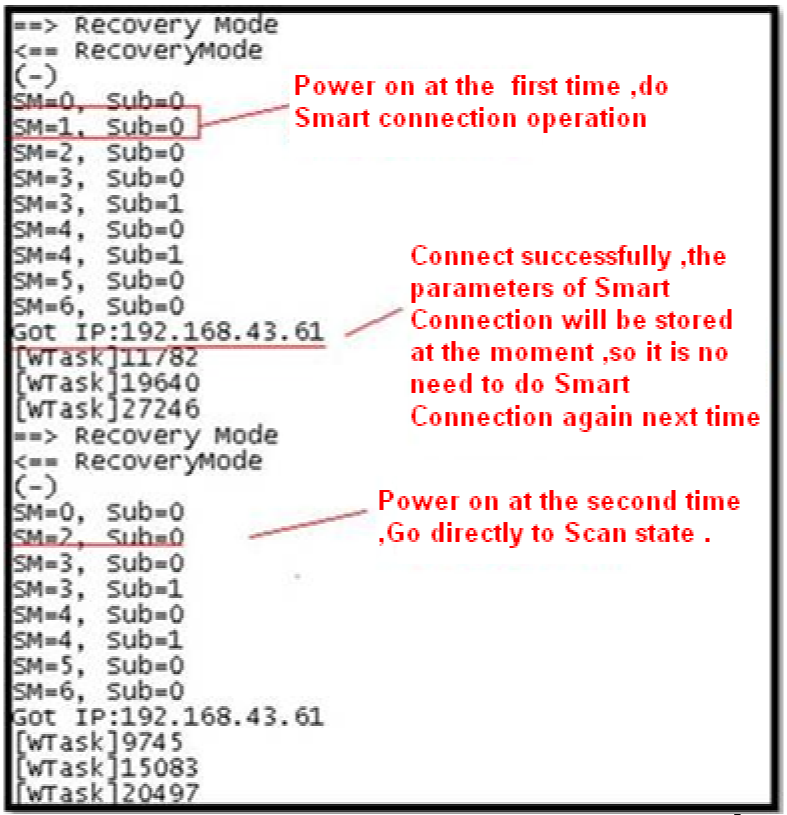
5.9 The data construction of station config in Smart Connection

If not use smart connection of MTK,
In the IoT_Cust_SM_Smnt, you can write IoTSmntInfo structure, and call wifi_state_chg to SCAN, and the system can complete process of Scan->Auth -> Assoc 。。。。。


```
    }
    /* Smnt connection done */


    /* After smnt connection done */
    /* need set Smnt connection information and start to scan*/
    IoTSmntInfo.AuthMode      = 0;
    IoTSmntInfo.SsidLen       = strlen(Ssid);
    IoTSmntInfo.PassphaseLen = strlen(Passphase); //sizeof(Passphase);
    memcpy(IoTSmntInfo.Ssid,      Ssid,      IoTSmntInfo.SsidLen);
    memcpy(IoTSmntInfo.Passphase, Passphase, IoTSmntInfo.PassphaseLen);
    memcpy(IoTSmntInfo.PMK,       PMK,       strlen(PMK));

    IoT_Cust_smnt_info();    /*Sync the IoTSmntInfo to other StateMachine, must call this func if IoTSı

    /* change wifi state to SCAN*/
    wifi_state_chg(WIFI_STATE_SCAN, 0);
#endif
```

5.10 The data access and delete for station config struct of Smart Connection.

Maybe, every customer has their own method to storage cfg
By default, we save SSID/Password/PMK… into flash, in the most cases, it can not be changed.
*Only in the case of deal with AT cmd: Default and Data Cmd: Offline, it can be delete
delete calling function: reset_sta_cfg ()
*Or use the flash original value, can not connect to AP, do Smart Connection again, connect to a new AP acquired IP, and lay over the old fash sta cfg value.
Calling function as follows: call IoT_Cust_smnt_info(VOID) at the smnt stage, and until IP is acquired, ws_got_ip() call to store_sta_cfg(VOID)



```
==> Recovery Mode
<== RecoveryMode
(-)
SM=0, Sub=0
SM=1, Sub=0
SM=2, Sub=0
SM=3, Sub=0
SM=3, Sub=1
SM=4, Sub=0
SM=4, Sub=1
SM=5, Sub=0
SM=6, Sub=0
Got IP:192.168.43.61
[WTask]11782
[WTask]19640
[WTask]27246
==> Recovery Mode
<== RecoveryMode
(-)
SM=0, Sub=0
SM=2, Sub=0
SM=3, Sub=0
SM=3, Sub=1
SM=4, Sub=0
SM=4, Sub=1
SM=5, Sub=0
SM=6, Sub=0
Got IP:192.168.43.61
[WTask]9745
[WTask]15083
[WTask]20497
```

Power on at the first time ,do Smart connection operation

Connect successfully ,the parameters of Smart Connection will be stored at the moment ,so it is no need to do Smart Connection again next time

Power on at the second time ,Go directly to Scan state .

5.11 The macro CFG_SUPPORT_MTK_SMNT=1 is used to control MTK smart connection

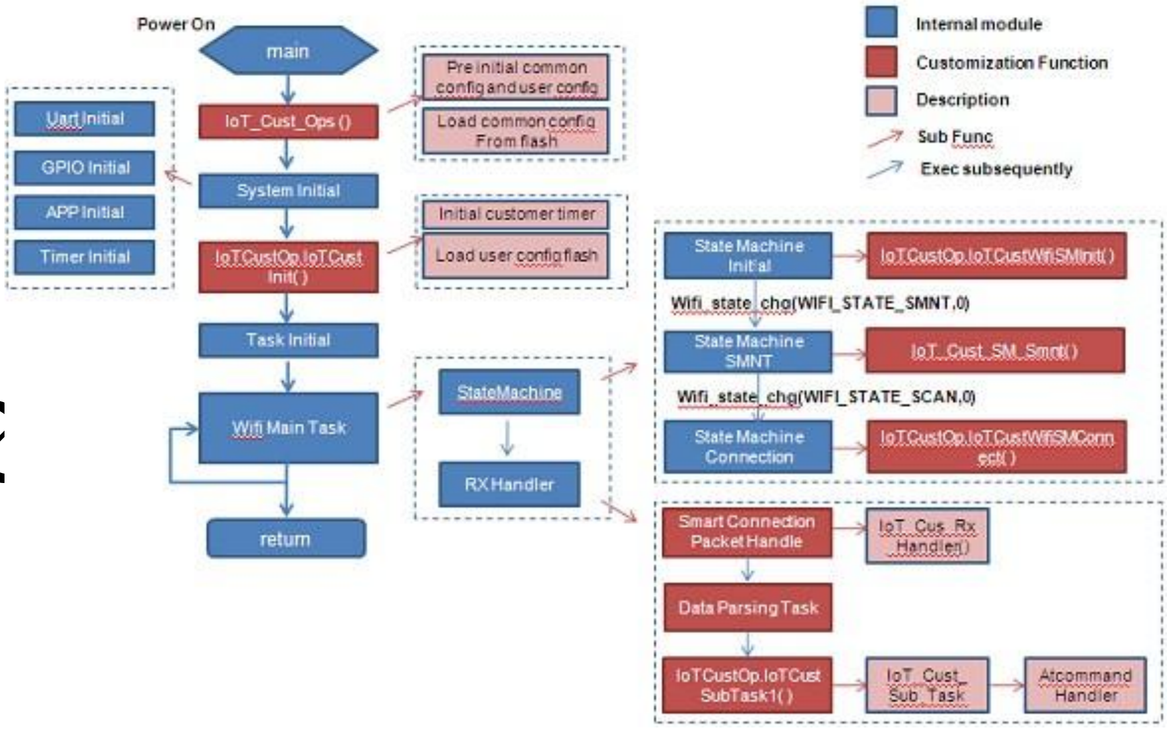
If the customer has his own Smart Connection, he can set the macro for 0.

In the MT7681_IoT_WIFI_Firmware_Programming_Guide_v0.05.pdf, there are the callback corresponding to the present Smart connection, among it,

Before SDKv1.20: in the IoT_Cus_Rx_Handler(), it can be seen the content of every Rx Packet , and it can be deal with further.

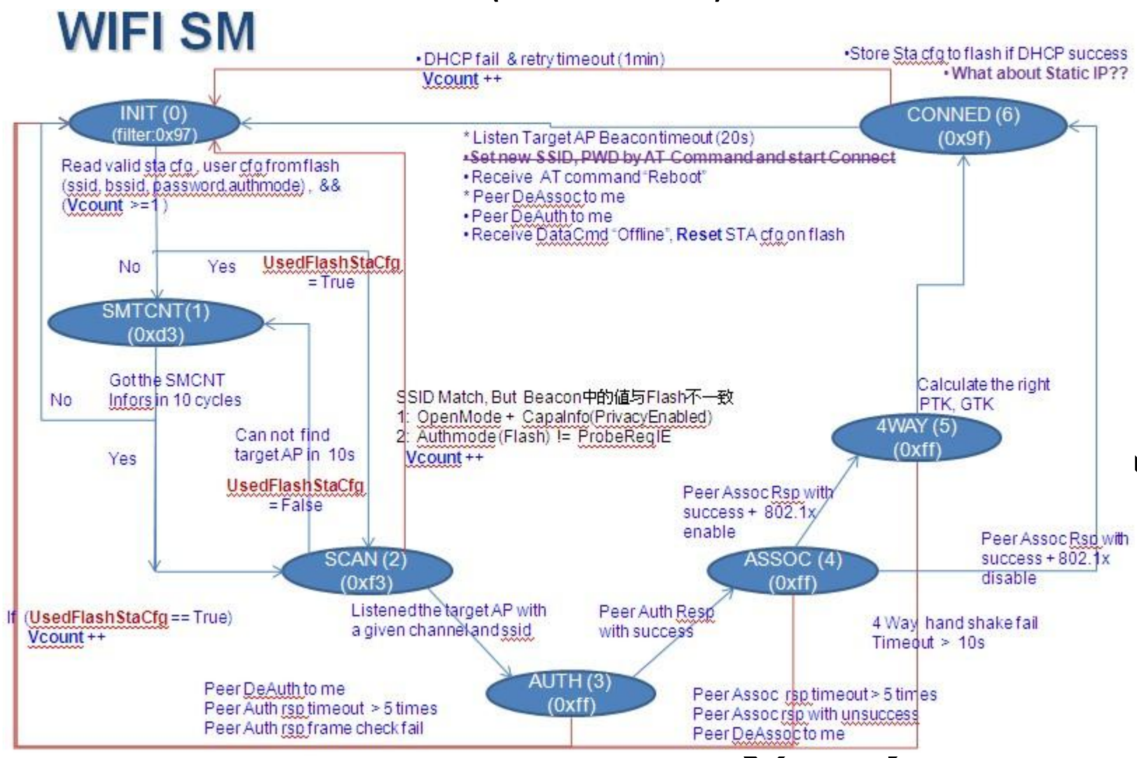
After SDKv1.30 : IoT_Cus_Rx_Handler() is replaced by STARxDoneInterruptHandle

IoT_Cut_SM_Smnt() is the state machine of smart connection .



5.12 Wifi Connection State Machine Flow

Wifi Connection State Machine Flow (STA Mode v1.30):


WIFI SM
•DHCP fail & retry timeout (1min)
Vcount ++
•Store Sta cfg to flash if DHCP success
•What about Static IP??
INIT (0)
(filter:0x97)
CONNED (6)
(0x9f)
* Listen Target AP Beacon timeout (20s)
~~•Set new SSID, PWD by AT Command and start Connect~~
•Receive AT command "Reboot"
* Peer DeAssoc to me
•Peer DeAuth to me
•Receive DataCmd "Offline", Reset STA cfg on flash
Read valid sta cfg, user cfg from flash
(ssid, bssid, password,authmode), &&
(Vcount >=1)
No
Yes
UsedFlashStaCfg
= True
SMTCNT(1)
(0xd3)
Got the SMCNT
Infors in 10 cycles
No
Yes
Can not find
target AP in 10s
UsedFlashStaCfg
= False
SSID Match, But Beacon中的值与Flash不一致
1: OpenMode + CapaInfo(PrivacyEnabled)
2: Authmode(Flash) != ProbeReqIE
Vcount ++
Calculate the right
PTK, GTK
4WAY (5)
(0xff)
Peer Assoc Rsp with
success + 802.1x
enable
SCAN (2)
(0xf3)
ASSOC (4)
(0xff)
Peer Assoc Rsp with
success + 802.1x
disable
If (UsedFlashStaCfg == True)
Vcount ++
Listened the target AP with
a given channel and ssid
Peer Auth Resp
with success
4 Way hand shake fail
Timeout > 10s
AUTH (3)
(0xff)
Peer DeAuth to me
Peer Auth rsp timeout > 5 times
Peer Auth rsp frame check fail
Peer Assoc rsp timeout > 5 times
Peer Assoc rsp with unsuccess
Peer DeAssoc to me

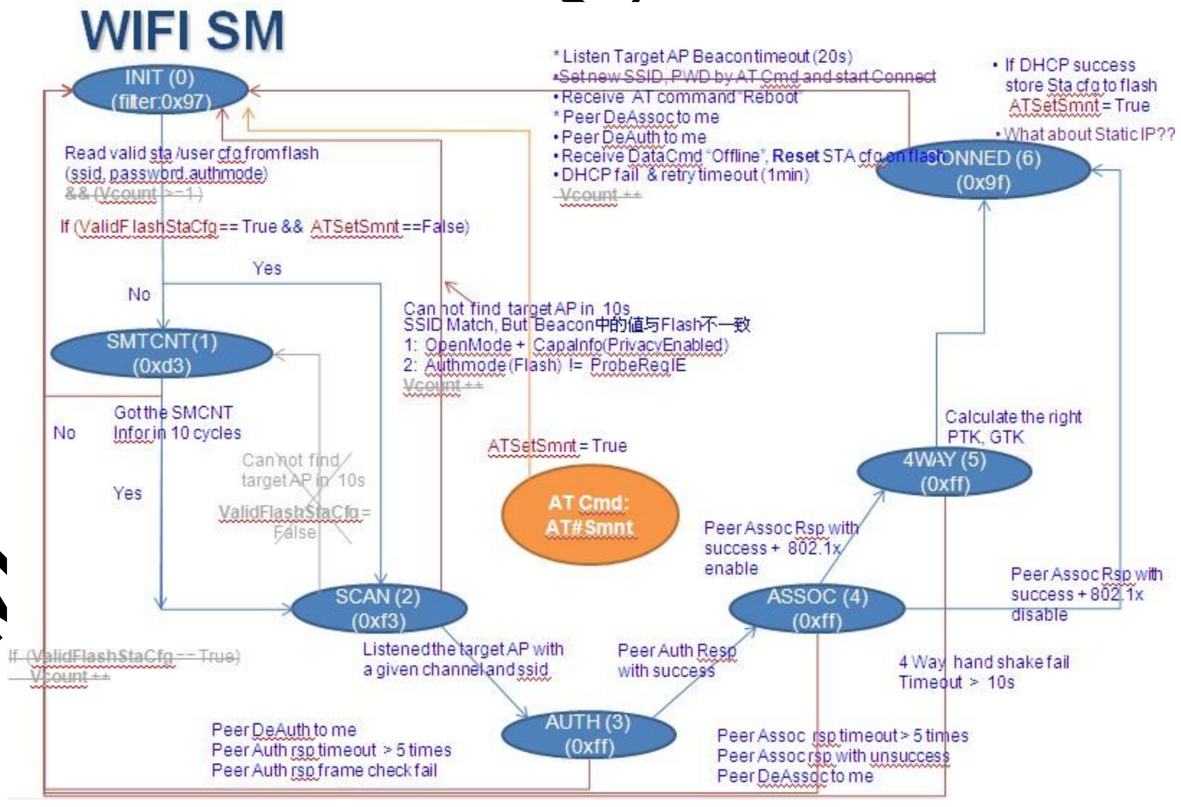
Wifi Connection State Machine Flow (STA Mode v1.40):

* when Scan,Auth,Assoc,4Way, DHCP fail , not back to smart connection state

* Smart Connection state only be triggered by following method:

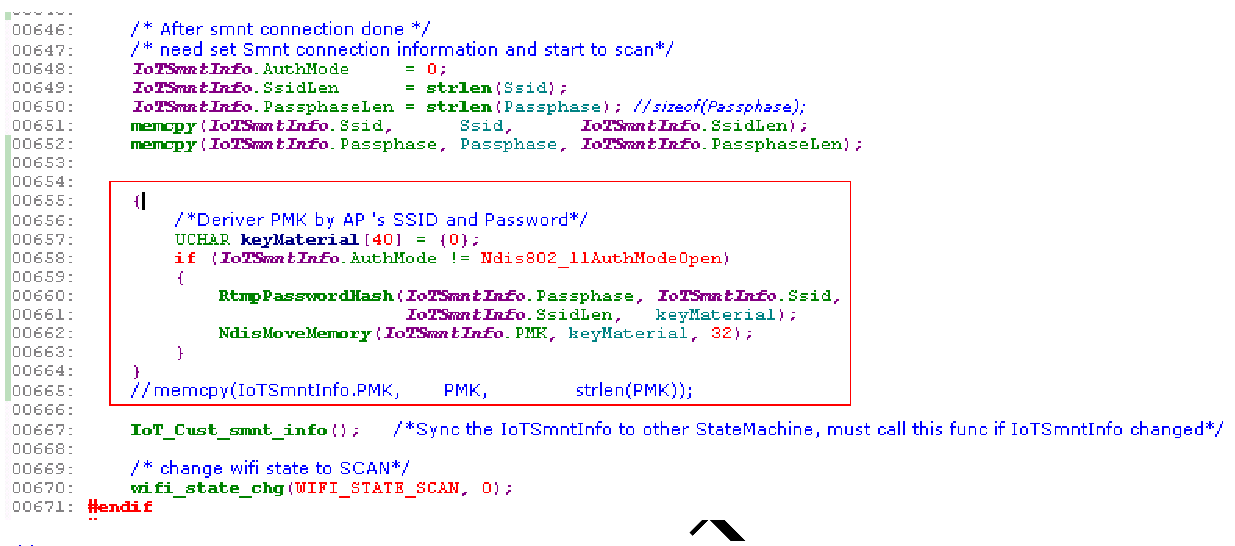
1: There is no valid content in Flash Sta Config region

2: 7681 receive AT Cmd: AT#Smnt

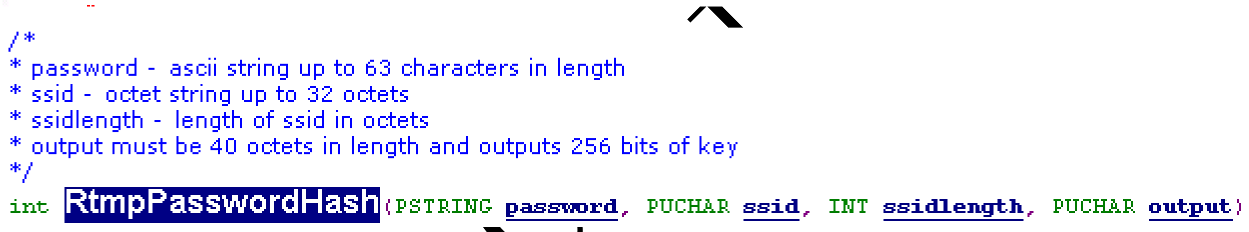


5.13 How to get PMK

PMK can be get from SSID, Passphase by SHA1 algorithm. We try 7618 software to get the value, it cost about 6s.
So we modified the algorithm.
Firstly, the PMK is calculated by mobile phone, secondly SSID, Passphase, PMK, AuthMode are passed out by smart connection. So the 7681 do not need to calculate the PMK, it save the time, and the PMK calculation function is deleted in 7681.
If you want to use it, you can call API: RtmpPasswordHash() in the IoT_Cust_SM_Smnt（）.



```
    /* After smnt connection done */
    /* need set Smnt connection information and start to scan*/
    IoTSmntInfo.AuthMode      = 0;
    IoTSmntInfo.SsidLen       = strlen(Ssid);
    IoTSmntInfo.PassphaseLen = strlen(Passphase); //sizeof(Passphase);
    memcpy(IoTSmntInfo.Ssid,      Ssid,       IoTSmntInfo.SsidLen);
    memcpy(IoTSmntInfo.Passphase, Passphase, IoTSmntInfo.PassphaseLen);


    {
        /*Deriver PMK by AP 's SSID and Password*/
        UCHAR keyMaterial[40] = {0};
        if (IoTSmntInfo.AuthMode != Ndis802_11AuthModeOpen)
        {
            RtmpPasswordHash(IoTSmntInfo.Passphase, IoTSmntInfo.Ssid,
                             IoTSmntInfo.SsidLen,    keyMaterial);
            NdisMoveMemory(IoTSmntInfo.PMK, keyMaterial, 32);
        }
    }
    //memcpy(IoTSmntInfo.PMK,     PMK,          strlen(PMK));

    IoT_Cust_smnt_info();    /*Sync the IoTSmntInfo to other StateMachine, must call this func if IoTSmntInfo changed*/

    /* change wifi state to SCAN*/
    wifi_state_chg(WIFI_STATE_SCAN, 0);
#endif
```

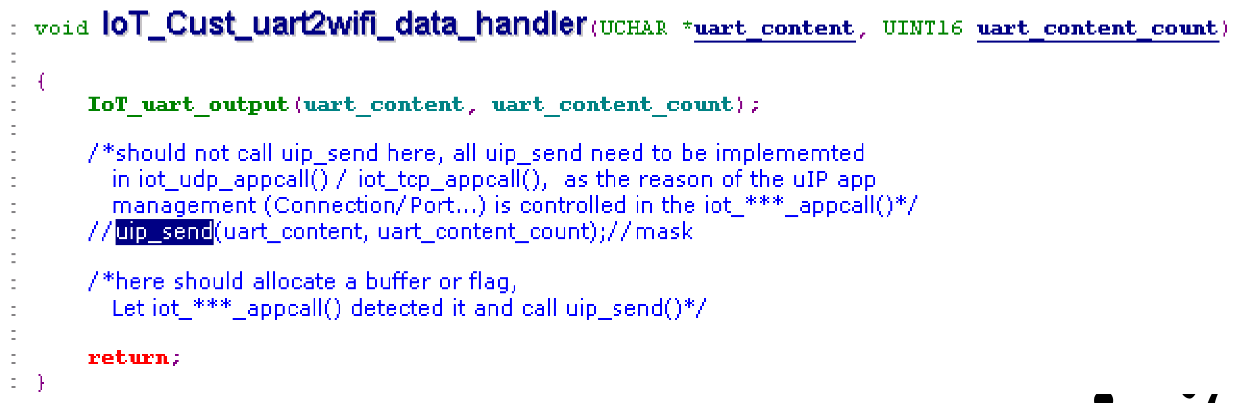


```
/*
* password - ascii string up to 63 characters in length
* ssid - octet string up to 32 octets
* ssidlength - length of ssid in octets
* output must be 40 octets in length and outputs 256 bits of key
*/
int RtmpPasswordHash(PSTRING password, PUCHAR ssid, INT ssidlength, PUCHAR output)
```

6 APPLICATION DEVELOPMENT

6.1 How to send TCP/UDP packet

The development of UDP、TCP app included connection building and packet sending should run in Tcpip_main.c à uip_process()-> UIP_APPCALL / UIP_UDP_APPCALL.Now the uIP code has been opened , it can be custom developed.
It will be failed the udp_send() is called directly to send Packet AT cmd handler.
Because the function is called directly, it is not assigned to send to which connection.



```
void IoT_Cust_uart2wifi_data_handler(UCHAR *uart_content, UINT16 uart_content_count)

{
    IoT_uart_output(uart_content, uart_content_count);

    /*should not call uip_send here, all uip_send need to be implememted
      in iot_udp_appcall() / iot_tcp_appcall(),  as the reason of the uIP app
      management (Connection/Port...) is controlled in the iot_***_appcall()*/
    //uip_send(uart_content, uart_content_count);//mask

    /*here should allocate a buffer or flag,
      Let iot_***_appcall() detected it and call uip_send()*/

    return;
}
```

6.2 Which APIs document should be refer to, if scan AP is built to connect AP, TCP + SSL .. is built in the IoT application.

For the MTK code, there is Initial-> Smart Connection-> Scan -> Auth-> Assoc-> 4 way -> DHCPC, in the mobile phone, the APK will send SSID, Password, PMK， MT7681, and at the stage of Smart Connect, the messages can be acquired, and according to it, scan the AP and connect,IP that get from AP. These are not customized, so it has not been developed corresponding API.

After get the IP, the uip is used to build TCP/UDP Connection.
SourceCode and Sample Code is developed in this part, the file is in the following path.
IoT_MT7681_PKG\cust\tcpip\,   IoT_MT7681_PKG\src\tcpip\
You can know more about it by referring to **MT7681_TCP_IP_*******_ToCust.pdf and those** Source File

6.3 What is the type of the data package when the data is sent to 7681 from server, and what is the type of the data after resolving the package?

You can refer to the IoT_Control_Protocol_v0.2.pdf in the Release Package v1.10 to know the type of the data package
The data after resolving package will be deal with further in the iot_parse.c IoT_process_app_packet(), the customer can define the type of the data.

6.4 The SDK has TCP and UDP mode, in which condition are they used?

Yes there are two mode, but it can be set by customers.
For the API and Source Code of TCP, UDP, we do not adopt the standard socket, but use uip. Because, the MT7681 is not adopted multi task or complicated OS.

The SourceCode and Sample Code are all published, the .c/ .h are all in the IoT_MT7681_PKG\cust\tcpip\, IoT_MT7681_PKG\src\tcpip\

You can refer to the MT7681_TCP_IP_*******_ToCust.pdf

And Source File in the Package v1.10.

6.5 What is the function of Internet Server besides ‘forwards’?

The same to 4.3, Internet Server is defined by customer, it can do forwards and many other things.

6.6 Is there the Support SNTP, NTP to get network time protocols?

There need SNTP, NTP, there is not above protocols in the MTK code.

6.7 Set up the TCP connection on MT7681

Behavior example:

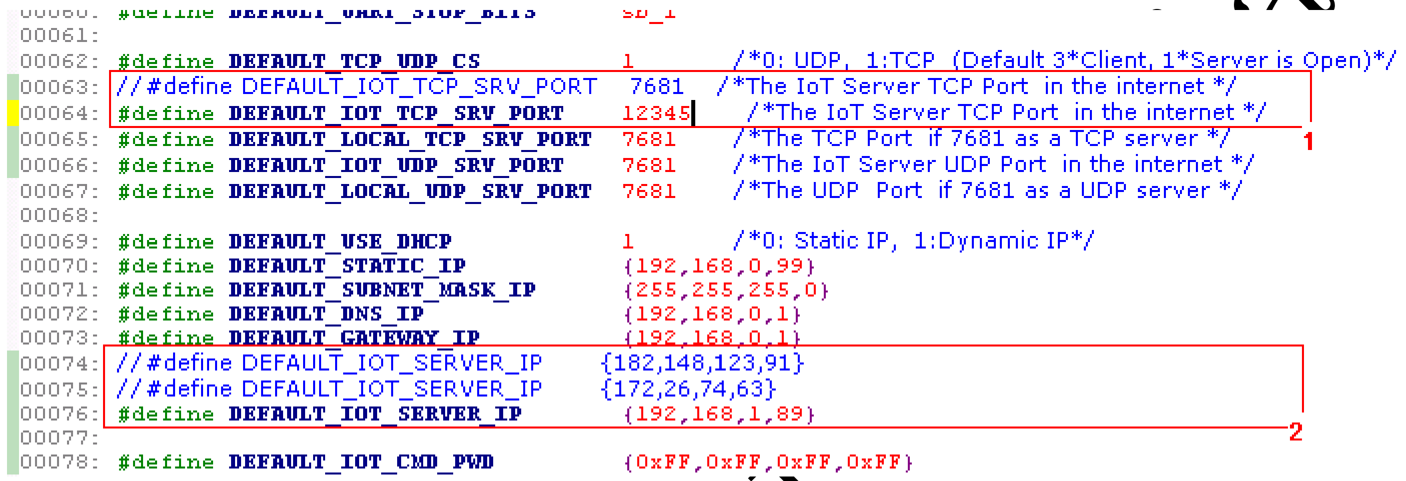
(1) Connection Type: TCP on port 12345 with data payload "0" or "1" only.

(2) Send data to MT7681 to Pull GPIO high/Low

The modify as follows

Step1:

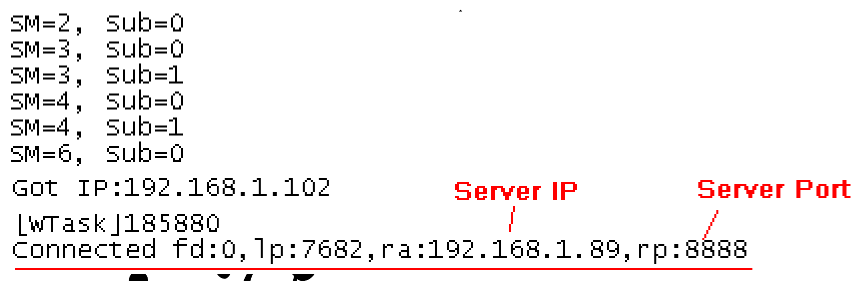
In the iot_custom.c , there is definition of default



```
#define DEFAULT_UART_STOP_BITS          SB_1

#define DEFAULT_TCP_UDP_CS              1        /*0: UDP, 1:TCP (Default 3*Client, 1*Server is Open)*/
//#define DEFAULT_IOT_TCP_SRV_PORT      7681    /*The IoT Server TCP Port in the internet */
#define DEFAULT_IOT_TCP_SRV_PORT        12345      /*The IoT Server TCP Port in the internet */
#define DEFAULT_LOCAL_TCP_SRV_PORT      7681     /*The TCP Port if 7681 as a TCP server */
#define DEFAULT_IOT_UDP_SRV_PORT        7681     /*The IoT Server UDP Port in the internet */
#define DEFAULT_LOCAL_UDP_SRV_PORT      7681     /*The UDP Port if 7681 as a UDP server */

#define DEFAULT_USE_DHCP                1        /*0: Static IP, 1:Dynamic IP*/
#define DEFAULT_STATIC_IP               {192,168,0,99}
#define DEFAULT_SUBNET_MASK_IP          {255,255,255,0}
#define DEFAULT_DNS_IP                  {192,168,0,1}
#define DEFAULT_GATEWAY_IP              {192,168,0,1}
//#define DEFAULT_IOT_SERVER_IP       {182,148,123,91}
//#define DEFAULT_IOT_SERVER_IP       {172,26,74,63}
#define DEFAULT_IOT_SERVER_IP           {192,168,1,89}

#define DEFAULT_IOT_CMD_PWD             {0xFF,0xFF,0xFF,0xFF}
```

Firstly, modify Server Port for 12345 in format1, and then modify Server IP in format2

The Server IP in format2 will be assigned to IoTpAdComCfg.IoT_ServeIP. now the IP address is acquired, and it will be connected to TCP server after the 7618 is power on, if everything is normal, there will be the hint as follows:

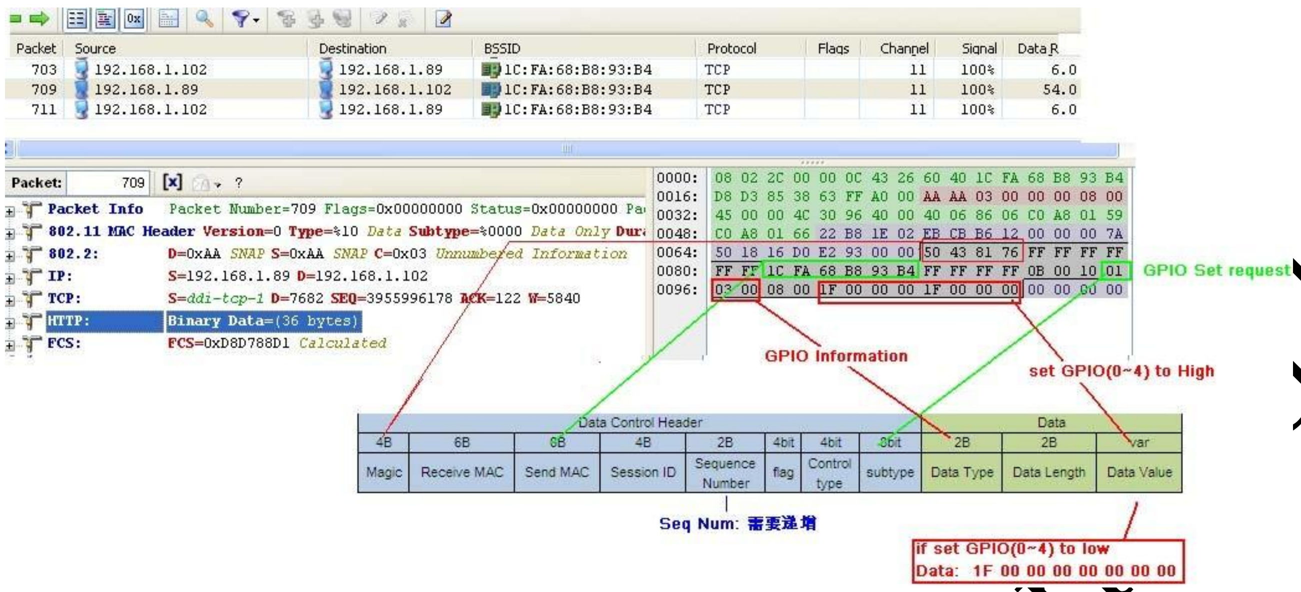


```
SM=2, Sub=0
SM=3, Sub=0
SM=3, Sub=1
SM=4, Sub=0
SM=4, Sub=1
SM=6, Sub=0
Got IP:192.168.1.102
[WTask]185880
Connected fd:0,lp:7682,ra:192.168.1.89,rp:8888
```

Step2:

After TCP Connection is connected, TCP Server can send Data Command to control GPIO of 7681, you can refer to **the** Protocol of Data Command **in the IoT_Control_Protocol_v0.*.pdf .**

There is an example that server send GPIO set commend to 7681.



```
Packet  Source          Destination     BSSID              Protocol  Flags  Channel  Signal  Data R
   703  192.168.1.102   192.168.1.89    1C:FA:68:B8:93:B4  TCP                  11    100%     6.0
   709  192.168.1.89    192.168.1.102   1C:FA:68:B8:93:B4  TCP                  11    100%    54.0
   711  192.168.1.102   192.168.1.89    1C:FA:68:B8:93:B4  TCP                  11    100%     6.0
```

```
Packet: 709
Packet Info         Packet Number=709 Flags=0x00000000 Status=0x00000000 Pa
802.11 MAC Header Version=0 Type=%10 Data Subtype=%0000 Data Only Dur
802.2:              D=0xAA SNAP S=0xAA SNAP C=0x03 Unnumbered Information
IP:                 S=192.168.1.89 D=192.168.1.102
TCP:                S=ddi-tcp-1 D=7682 SEQ=3955996178 ACK=122 W=5840
HTTP:               Binary Data=(36 bytes)
FCS:                FCS=0xD8D788D1 Calculated
```

```
0000: 08 02 2C 00 00 0C 43 26 60 40 1C FA 68 B8 93 B4
0016: D8 D3 85 38 63 FF A0 00 AA AA 03 00 00 00 08 00
0032: 45 00 00 4C 30 96 40 00 40 06 86 06 C0 A8 01 59
0048: C0 A8 01 66 22 B8 1E 02 EB CB B6 12 00 00 00 7A
0064: 50 18 16 D0 E2 93 00 00 50 43 81 76 FF FF FF FF
0080: FF FF 1C FA 68 B8 93 B4 FF FF FF FF 0B 00 10 01
0096: 03 00 08 00 1F 00 00 00 1F 00 00 00 00 00 00 00
```

GPIO Set request
GPIO Information
set GPIO(0~4) to High
Data Control Header
Data
4B | 6B | 6B | 4B | 2B | 4bit | 4bit | 8bit | 2B | 2B | var
Magic | Receive MAC | Send MAC | Session ID | Sequence Number | flag | Control type | subtype | Data Type | Data Length | Data Value
Seq Num: 需要递增
if set GPIO(0~4) to low
Data: 1F 00 00 00 00 00 00 00

6.8 Cannot use AT Cmd "Netmode ,Channel, SoftAPConf and TCP_Connect "…

What is the condition of Tcp_Connect command TCP_IP?
In the STA mode, smartconnection open the TCP_IP by default.
Can the Tcp_Connect TCP_IP be reset?

A3-1: SoftAPConf command is included by CONFIG_SOFTAP , the macro is only in the flag_ap.mk , so SoftAPConf is enable in the SoftAP mode.

A3-2: because UIP source code are all developed , Tcp_Connect is disable by default in the AT cmd .

6.9 Enable TCP Tx ReTransmit, and Http Client

In uip_conf.h

1: Enable TCP Re-Transmit

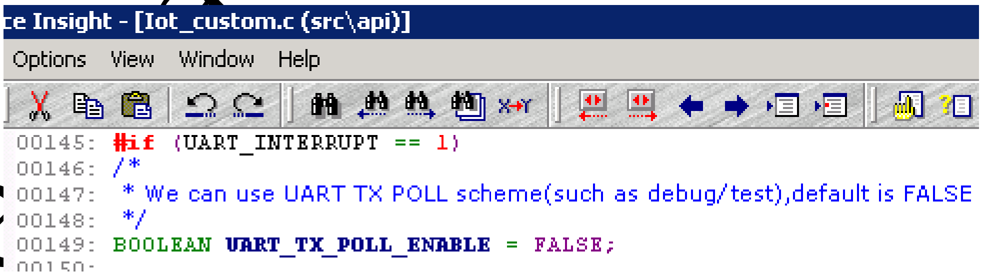
```
#define CFG_SUPPORT_TCP_REXMIT 1
```

2: Enable Http Client

```
#define UIP_HTTP_CLIENT_SUPPORT 1
```

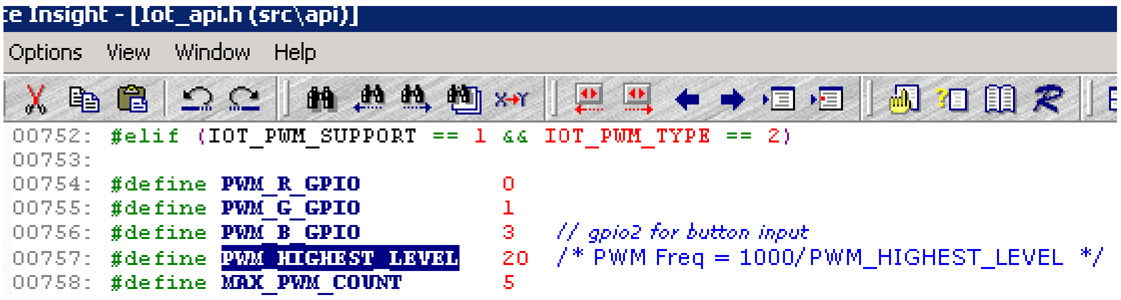
7 INTERFACE CUSTOMIZATION

7.1 Set UartTx as Interrupt Mode or Poll mode



```
#if (UART_INTERRUPT == 1)
/*
 * We can use UART TX POLL scheme(such as debug/test),default is FALSE
 */
BOOLEAN UART_TX_POLL_ENABLE = FALSE;
```

7.2 Set PWM Level



```
#elif (IOT_PWM_SUPPORT == 1 && IOT_PWM_TYPE == 2)

#define PWM_R_GPIO           0
#define PWM_G_GPIO           1
#define PWM_B_GPIO           3    // gpio2 for button input
#define PWM_HIGHEST_LEVEL    20   /* PWM Freq = 1000/PWM_HIGHEST_LEVEL */
#define MAX_PWM_COUNT        5
```

7.3 Why GPIO-x set Back to Low， after use IoT_gpio_output() to set GPIO-x to High

The reason is that, something call the IoT_gpio_input()

If GPIOx is Output mode, it will be set Input mode, and the value of input is 0

If GPIOx is Input mode, the input value is the state that GPIO get from outside circuit

It offer API in the v1.2 SDK, IoT_gpio_read (INT32 GPIONum, UINT8 *Val, UINT8 *Polarity) can do read operation to gpio, but it doesn't change the state or mode of gpio.