Revision History

Revision	Date (mm/dd/yyyy)	Author	Reviewer	Comments
1.0	01/12/2014	Yintong	N/A	Initial draft
1.1	05/21/2014	Yintong	N/A	Enhance at command format Support concurrent test flow
1.2	06/08/2014	Yintong	N/A	Support Audio loopback interface Enhance Touch Panel test flow
1.3	07/25/2014	Yintong	N/A	Enhance Audio loopback interface

Contents

Introduction

This document mainly introduce the following three parts:

1） How to trigger MTK Phone boot up into handset testing mode（factory mode）
2） Command list for each test items and the analysis of the command format
3） The structure of response data

1 How to trigger MTK phone boot into handset testing mode

To trigger target boot into testing mode, the dlls and USB driver provided by mediate are needed.Please look for them in the tool package released by mediatek.
DLLs includes FlashToolLib.dll & SPMETA_DLL.dll & SLA_Challenge.dll
USB driver is SP_Drivers.rar

1.1 SP_Preloader_BootMode

Please use SP_Preloader_BootMode provided by falshtoolib to trigger target boot into testing mode. After the target successfully boot up into testing mode,the kernel comport will enumerate,and then you can send command to target by this comport.

[Function prototype]

```
LIBEXPORT int __stdcall SP_Preloader_BootMode(SP_BOOT_ARG_S * pArg);
```

[Structure of SP_BOOT_ARG_S]

```
typedef struct
{
   //Old parameters
   SP_BBCHIP_TYPE m_bbchip_type;
   SP_EXT_CLOCK m_ext_clock;
   unsigned int m_ms_boot_timeout;
   unsigned int m_max_start_cmd_retry_count;

   //New parameters
   unsigned int m_uTimeout;
   unsigned int m_uRetryTime;
   unsigned int m_uInterval;
   unsigned int m_uBaudrate;
   int * m_pStopFlag; // if m_pStopFlag = 0x9876， that means stop booting into testing mode

   bool m_bIsUSBEnable;
   bool m_bIsSymbolicEnable;
   bool m_bIsCompositeDeviceEnable; // true:  enable adb log， false : disable adb log
```

```
    SP_BOOT_MODE m_euBootMode;
    unsigned short m_uMDMode;
    unsigned int m_uPortNumber; // Preloader comport number
    char m_szPortSymbolic[256];

    // Serial Link Authentication
    SP_AUTH_HANDLE_T  m_auth_handle;
    SP_SCERT_HANDLE_T m_scert_handle;
    SP_CALLBACK_SLA_CHALLENGE  m_cb_sla_challenge;
    void *  m_cb_sla_challenge_arg;
    SP_CALLBACK_SLA_CHALLENGE_END  m_cb_sla_challenge_end;
    void *  m_cb_sla_challenge_end_arg;

} SP_BOOT_ARG_S;
```

1.2 Demo code

```
bool ATA_DLL_Handle::EnterFactoryModeByUSB (unsigned int &comport)
{
        SP_BOOT_ARG_S stArg;

        memset (&stArg, 0x0, sizeof(SP_BOOT_ARG_S));
        stArg.m_bbchip_type = SP_AUTO_DETECT_BBCHIP;
        stArg.m_ext_clock = SP_AUTO_DETECT_EXT_CLOCK;
        stArg.m_ms_boot_timeout = SP_BOOT_INFINITE;
        stArg.m_max_start_cmd_retry_count = SP_DEFAULT_BROM_START_CMD_RETRY_COUNT;

        //New parameters
        stArg.m_uTimeout = 20000;
        stArg.m_uRetryTime = 2000;
        stArg.m_uInterval = 10;
        stArg.m_uBaudrate = CBR_115200;

        // Serial Link Authentication
        stArg.m_auth_handle = SPATE_Get_AuthHandle();
        stArg.m_scert_handle = SPATE_Get_ScertHandle();
        stArg.m_cb_sla_challenge = SLA_Challenge;
        stArg.m_cb_sla_challenge_arg = NULL;
        stArg.m_cb_sla_challenge_end = SLA_Challenge_END;
        stArg.m_cb_sla_challenge_end_arg = NULL;

        stArg.m_pStopFlag = m_commonCFG.stop_flag;
        stArg.m_bIsUSBEnable = true;
        stArg.m_bIsSymbolicEnable = false;
        stArg.m_bIsCompositeDeviceEnable = m_commonCFG.com_port_info.bIsComposite ? true : false;
        stArg.m_euBootMode = SP_FACTORY_BOOT;
        stArg.m_uPortNumber = m_commonCFG.com_port_info.preloader_com_port;

        Eboot_DebugOn ();
        SP_META_DebugOn ();
        SP_Brom_DebugOn ();

        if (SP_S_DONE != SP_Preloader_BootMode(&stArg))
        {
                return false;
        }

        SP_META_DebugOff ();
        Eboot_DebugOff ();
        SP_Brom_DebugOff ();

        comport = m_commonCFG.com_port_info.kernel_com_port;
        return true;
}
```

2 Test Command

Command type	AT Command	Return	Description
Common	AT+START	pass or fail	enter automatic testing mode
	AT+STOP	pass or fail	exit automatic testing mode
	AT+REQUESTDATA	all the test result are record in one structure and return	get test result
Module	AT+FM	pass or fail	run FM testing
	AT+MEMCARD	pass or fail	run Memory card testing
	AT+SIM	pass or fail	run SIM card testing
	AT+KEY	pass or fail	run Keypad testing
	AT+MTOUCH	pass	runTouch panel test by drawing lines
	AT+RINGTONE=1,2	pass	Mic->Speaker loopback testing
	AT+RECEIVER=1,2	pass	Mic->Receiverloopback testing
	AT+HEADSET=1,2	pass	Mic->Headset loopback testing
	AT+LCD(=STOP)	pass or fail	switch the screen picture
	AT+GPS=1,3	pass or fail	run GPS testing
	AT+EMMC	pass or fail	run EMMC testing
	AT+MAINCAMERA	pass or fail	photographed with rear camera
	AT+SUBCAMERA	pass or fail	photographed with front camera
	AT+CAMERADATA	Camera block	transfer picture taked by camera
	AT+WIFI=1,3	pass or fail	run WIFI testing
	AT+BT=1,3	pass or fail	run BT testing
	AT+LED(=STOP)	pass or fail	open/close three-color light and virual key light
	AT+CHARGER	pass or fail	run Charger testing
	AT+GSENSOR(=STOP)	not test	open/close Gsensor
	AT+MSENSOR(=STOP)	not test	open/close M-sensor
	AT+ALSPS(=STOP)	not test	open/close ALSPS sensor
	AT+GYROSCOPE(=STOP)	not test	open/close GYROSCOPE sensor
	AT+VIBRATOR(=STOP)	not test	Open/close Vibrator
	AT+CHARGER	pass or fail	Read the battery's voltage,charge current and battery's temperature
	AT+VERSION	pass or fail	Read software version and IMEI,use"AT+REQUESTDATA" to get the detail information
	AT+READBARCODE	barcode	read barcode

1) BT & WIFI & GPS & Signal Test support background parallel testing to save time。
2) In order to prevent interference between the items' return values when doing parallel test，item id will be added at the front of the test result.
   - For example, after main camera test,target will retrun "0:pass" or "0:fail"
   - The corresponding id of each item， please refer to section 3.3

2.1 AT+START

This is the first command of automatic testing,when the target boot up into testing mode,this command should be sent in 1500 ms, otherwise,the target will not be automatically tested.

If this command is successfully accepted,target will respond “pass”, otherwise respond “fail” .

2.2 AT+STOP

This command is used to make target exit automatically test. After this command,taget will be in normal testing mode.

If this command is successfully accepted,target will respond “pass”, otherwise respond “fail” .

2.3 AT+REQUESTDATA

This command is used to get all test result of the items. After sending this command to target, the test result will be stored in the struct “sp_ata_data” and return.

The format of the struct “sp_ata_data” will be referred at section 4.2.

If this command is successfully accepted,target will respond a sting with “sp_ata_data” type, otherwise respond “fail”.

2.4 AT+FM

This command is used to make the target doing FM test,when the target receive this command,it will search FM signal.And If the target search for right signal,it will respond “pass”,otherwise respond “fail”.

The following command is to set the searching frequency (For example,if you want to set 998 MHz, frenquency should be 9980）.

```
AT+PROPERTY=0,1,"FMRadio.CH1,frequency"
```

The following command is to set the play time in seconds.

```
AT+PROPERTY=0,1,"FMRadio.PlayTime,time”
```

The two commands should be sent before “AT+FM”.

2.5 AT+MEMCARD

This command is to make the target checking memory card.If target detected memory card exist , it will respond “pass”,otherwise respond “fail”.

The memory card’s specific capacity will be stored in “sp_ata_data”, and you can send “AT+REQUESTDATA” to get the detail information.

2.6 AT+SIM

This command is used to make the target checking SIM card.If target detected memory card exist , it will respond “pass”,otherwise respond “fail”.

2.7 AT+KEY

This command is used to test if the key is working properly.
Please first send the following command before "AT+KEY" to set test type,then the target will return the key code of each key when running keys test.

```
AT+PROPERTY=0,1,"FtmKeyType,1"
```

2.8 AT+MTOUCH

This command is used to test touchpanel by drawing lines.

2.9 AT+RINGTONE=1,2

This command is used to open "Mic -> Speaker" loopback. Before sending "AT+RINGTONE=1,2" to open the loopback,please set the input mic type by sending the following command.

AT+PROPERTY=0,1,"Audio.Manual.InputType,X"
X = 0，close the access
X = 1，set main mic as the input mic
X = 2，set sub mic as the input mic
X = 3，set headset mic as the input mic

2.10 AT+RECEIVER=1,2

This command is used to open "Mic -> Receiver" loopback. Before sending "AT+RECEIVER=1,2" to open the loopback,please set the input mic type by sending the following command .

AT+PROPERTY=0,1,"Audio. Manual.InputType,X"
X = 0，close the access
X = 1，set main mic as the input mic
X = 2，set sub mic as the input mic
X = 3，set headset mic as the input mic

2.11 AT+HEADSET=1,2

This command is used to open "Mic -> Headset" loopback. Before sending "AT+HEADSET=1,2" to open the loopback,please set the input mic type by sending the following command .

AT+PROPERTY=0,1,"Audio. Manual.InputType,X"
X = 0， close the access
X = 1，set main mic as the input mic

X = 2，set sub mic as the input mic
X = 3，set headset mic as the input mic

When complete the test by sending “AT+HEADSET”,the earphone plug state and key state are recorded in ftm_ata_headset of sp_ata_data. You can get the headset status by sending “AT+REQUESTDATA”.

```
typedef struct{
    int freqL;
    int amplL;
    int freqR;
    int amplR;
   int hds_state;  // headset plug state，1 : insert；  0 – pull out
   int hds_mic_state; // headset key state，  1 :pressed；  0 : released
} ftm_ata_headset;
```

2.12 AT+LCD(=STOP)

“AT+LCD” is used to make target doing screen color switch. Each time the command sent,the target’s screen swich to one color. The order of color-switching is red, green, blue, white, black.

To stop LCD test,please send “AT+LCD=STOP”.

2.13 AT+GPS=1,3

This command is used to make target search for GPS satellites,if any one can be found, target will return “pass” otherwise return “fail”..

This command support background parallel testing,so you can test other items at the same time.

2.14 AT+EMMC

This command is used to make the target checking EMMC.If target detected emmc exist , it will respond “pass”,otherwise respond “fail”.

The emmc’s specific capacity in GB will be stored in “sp_ata_data”, and you can send “AT+REQUESTDATA” to get the detail information.

2.15 AT+MAINCAMERA

This command is used to make target taking a picture with rear camera. Before this command,please first send the following command to set testing method.

```
AT+PROPERTY=0,1,"FTMCameraTest,1"
```

After taking a picture,you can get the picture in JPG format by sending "AT+CAMERADATA". Each time the command sent, data returns up to 150k size.Please check data integrity by comparing current_block and total_block. At last,compose a integral picture by the date received.

```
typedef struct{
    int total_block;
    int current_block;
    int block_size;
    char camera_data[1024*150];
} ftm_ata_camera_data;
```

2.16 AT+SUBCAMERA

This command is used to make target taking a picture with front camera. Before this command,please first send the following command to set testing method.

```
AT+PROPERTY=0,1,"FTMCameraTest,1"
```

The method of picture's transmission is the same as rear camera.

2.17 AT+WIFI=1,3

This command is used to make target search wifi hotspot and establish a connection.If success it will return pass otherwise return fail.

The wifi's ssid is set to mtkguest by default. Please set wifi hotspot name to mtkguest or change the searching ssid in target before testing.

This command support background parallel testing,so you can test other items at the same time.

2.18 AT+CHARGER

This command is used to make target detect battery's voltage, charging current and battery's temperature. If target detect the value, it will return pass otherwise return fail.

The detail value target detect will be stored in "sp_ata_data", please send "AT+REQUESTDATA" to get the information.

2.19 AT+BT=1,3

This command is used to make target search for BT device nearby, if any one can be found, target will return "pass" otherwise return "fail".

You can set the max devices want to search by sending the following command.

```
AT+PROPERTY=0,1,"BT.DeviceNumber,X"
```

X, the number of device you want to search.

For example, AT+PROPERTY=0,1,"BT.DeviceNumber,3" means you just search max 3 devices.

If you don't set it ,it will search 10 devices by default.

This command support background parallel testing,so you can test other items at the same time.

2.20 AT+READBARCODE

This command is used to read barcode from target,if success it will return the barcode, otherwise return fail.

2.21 AT+LED(=STOP)

"AT+LED" is used to open three-color light and virual key light.
Send "AT+LED=STOP" to close the light.

2.22 AT+GSENSOR(=STOP)

"AT+GSENSOR" is used to open gsensor,and the coordinate values will be stored in sp_ata_data in real time.
"AT+GSENSOR=STOP" will close the sensor.
To get detail values ,please send "AT+REQUESTDATA".

2.23 AT+MSENSOR(=STOP)

"AT+MSENSOR" is used to open gsensor,and the coordinate values will be stored in sp_ata_data in real time.
"AT+MSENSOR=STOP" will close the sensor.
To get detail values ,please send "AT+REQUESTDATA".

2.24 AT+ALSPS(=STOP)

"AT+ALSPS" is used to open alsps sensor,and the coordinate values will be stored in sp_ata_data in real time.
"AT+ALSPS=STOP" will close the sensor.
To get detail values ,please send "AT+REQUESTDATA".

2.25 AT+GYROSCOPE(=STOP)

"AT+GYROSCOPE" is used to open gyroscope sensor,and the coordinate values will be stored in sp_ata_data in real time.

"AT+ GYROSCOPE =STOP" will close the sensor.
To get detail values ,please send "AT+REQUESTDATA".

3 Format of Response Data

3.1 structure of sp_ata_data

```
typedef struct{
        ftm_ata_fm fm;
        ftm_ata_wifi wifi;
        ftm_ata_bt_num bt;
        ftm_ata_version version;
        ftm_ata_gps gps;
        ftm_ata_speaker speaker;
        ftm_ata_receiver receiver;
        ftm_ata_headset headset;
        ftm_ata_headset headsetL;
        ftm_ata_headset headsetR;
        ftm_ata_battery battery;
        ftm_ata_gsensor gsensor;
        ftm_ata_msensor msensor;
        ftm_ata_alsps alsps;
        ftm_ata_gyroscope gyroscope;
        ftm_ata_vibrator vibrator;
        ftm_ata_freq_response rcv_response;
        ftm_ata_freq_response spk_response;
        ftm_ata_thd rcv_thd;
        ftm_ata_thd spk_thd;
        ftm_ata_thd headsetL_thd;
        ftm_ata_thd headsetR_thd;
        ftm_ata_memcard memcard;
        ftm_ata_emmc emmc;
} sp_ata_data;
```

3.2 Structure of test Items

```
typedef struct {
        int fm_rssi;
        int freq;
        int ampl;
} ftm_ata_fm;

typedef struct {
        char wifi_mac[33];
        char wifi_name[32];
        int wifi_rssi;
        int channel;
        int rate;
} ftm_ata_wifi;

typedef struct {
        char bt_mac[32];
        char bt_name[32];
        int bt_rssi;
} ftm_ata_bt;

typedef struct {
        char modem_ver[128];
        char sw_ver[128];
} ftm_ata_version;

typedef struct{
        int num;
        ftm_ata_bt bt[10];
} ftm_ata_bt_num;

typedef struct{
        float ratio;
        int offset;
        float drift;
        int mean;
        int sigma;
        int update_hz;
        int bitsync;
        int acquision;
        int svid;
} ftm_ata_gps;

typedef struct{
        int freqL;
        int amplL;
        int freqR;
        int amplR;
} ftm_ata_speaker;
```

```
typedef struct{
            int freqL;
    int amplL;
    int freqR;
    int amplR;
} ftm_ata_receiver;

typedef struct{
            int freqL;
             int amplL;
            int freqR;
            int amplR;
            int hds_state;
            int hds_mic_state;
} ftm_ata_headset;

typedef struct{
            int current;
            int voltage;
            int vbattemp;
} ftm_ata_battery;

typedef struct{
    float g_sensor_x;
    float g_sensor_y;
    float g_sensor_z;
    int accuracy;
} ftm_ata_gsensor;

typedef struct{
    int m_sensor_x;
    int m_sensor_y;
    int m_sensor_z;
    int accuracy;
} ftm_ata_msensor;

typedef struct{
    int als;
    int ps;
} ftm_ata_alsps;

typedef struct{
    float gyroscope_x;
    float gyroscope_y;
    float gyroscope_z;
    int accuracy;
} ftm_ata_gyroscope;

typedef struct{
    int freq;
    int ampl;
} ftm_ata_vibrator;

typedef struct{
    float mean;
    float deviation;
    float max;
    float min;
} ftm_ata_aud_perfromance;

typedef struct{
    ftm_ata_aud_perfromance thd;
} ftm_ata_thd;

typedef struct{
    ftm_ata_aud_perfromance freqresponse[5];
} ftm_ata_freq_response;

typedef struct{
            unsigned int sd1_total_size;
            unsigned int sd1_free_size;
            unsigned int sd2_total_size;
            unsigned int sd2_free_size;
} ftm_ata_memcard;

typedef struct{
            float capacity;
} ftm_ata_emmc;
```

The actual structure maybe a little difference with the above-mentioned. You can refer to the file named “ftm.h” in your target’s software.

3.3 Item ID

The actual Item ID maybe a little difference with the below-mentioned. You can refer to the file named "common.h" in your target's software.

```
typedef enum {
    ITEM_MAIN_CAMERA= 0,
    ITEM_MAIN2_CAMERA,
    ITEM_SUB_CAMERA,
    ITEM_STROBE,
    ITEM_GPS,
    ITEM_NFC,
    ITEM_FM,
    ITEM_FMTX,
    ITEM_FLASH,
    ITEM_MEMCARD,
    ITEM_RTC,  //10
    ITEM_LCD,
    ITEM_LCM,
    ITEM_BACKLIGHT,
    ITEM_LED,
    ITEM_LOOPBACK,
    ITEM_LOOPBACK1,
    ITEM_LOOPBACK2,
    ITEM_LOOPBACK3,
    ITEM_BT,
    ITEM_WIFI, //20
    ITEM_KEYS,
    ITEM_LOOPBACK_PHONEMICSPK,
    ITEM_WAVEPLAYBACK,
    ITEM_ACOUSTICLOOPBACK,
    ITEM_GSENSOR,
    ITEM_GS_CALI,
    ITEM_MSENSOR,
    ITEM_ALSPS,
    ITEM_HEADSET,
    ITEM_HEADSET_DEBUG, // 30
    ITEM_USB,
    ITEM_OTG,
    ITEM_CLRFLASH,
    ITEM_CHARGER,
    ITEM_TOUCH,
    ITEM_TOUCH_AUTO,
    ITEM_SIM,
    ITEM_VIBRATOR,
    ITEM_RECEIVER,
    ITEM_RECEIVER_DEBUG,
    ITEM_SIMCARD,
    ITEM_IDLE,
    ITEM_TVOUT,
    ITEM_JOGBALL,
    ITEM_OFN,
    ITEM_MATV_NORMAL,
    ITEM_MATV_AUTOSCAN,
    ITEM_MUI_TEST,
    ITEM_FULL_TEST,
    ITEM_ITEM_TEST,
    ITEM_AUTO_TEST,
    ITEM_DEBUG_TEST,
    ITEM_VERSION,
    ITEM_REPORT,
    ITEM_UPDATE,
    ITEM_REBOOT,
    ITEM_BAROMETER,
    ITEM_GYROSCOPE,
    ITEM_GYROSCOPE_CALI,
    ITEM_SPK_OC,
    ITEM_SIGNALTEST,
    ITEM_CMMB,
    ITEM_EMMC,
    ITEM_EMI,
    ITEM_CLREMMC,
    ITEM_HDMI,
    ITEM_RECEIVER_PHONE,
    ITEM_HEADSET_PHONE,
    ITEM_LOOPBACK_PHONEMICSPK_PHONE,
    ITEM_VIBRATOR_PHONE,

    ITEM_CUSTOM_START,
    ITEM_CUSTOM_STOP,
    ITEM_CUSTOM_REQUESTDATA,
    ITEM_CUSTOM_VERSION,
    ITEM_CUSTOM_READBARCODE,
    ITEM_CUSTOM_WRITEBARCODE,
    ITEM_CUSTOM_CAMERADATA,
    ITEM_MAX_IDS
} ITEM_SEQUENCE;
```