INTERNAL USE

SP ATA Customization Guide v2.0

Goal

- **This document is mainly**

  --To introduce how to add a new test item for ATA2.0 both in target side and in tool side.

  --To list the function that need customer to modify according to their design.

  -- The software of target are based on ALPS.L1.XXX,if your version is not L1.XXX,there maybe some difference with this document. But the whole ideas are the same, you can make the appropriate adjustments for your software.

Steps

- Add New Test Item into Factory Mode(target side)
- Add Test Item into SP ATA Test Mode(target side)
- ATA Tool Customization Guide(Tool side)

Add New Test Item into Factory Mode

--Target Side

Steps(1/7)

- **Step 1:Create your feature definition in “custom/{project}/factory/inc/cust.h”**

  e.g. FEATURE_FTM_HALL

```
//Add hall test
#define FEATURE_FTM_HALL
```

- **Step 2:Create your factory test item id in “factory/inc/common.h”**

  e.g. ITEM_HALL

```
ITEM_HEART_MONITOR,
ITEM_HALL,
ITEM_MAX_IDS
```

Note:please add the item id in the last of the enum

Steps(2/7)

- **Step 3:Add factory test menu item to array in “factory/src/item.cpp”**

  item_t **ftm_test_items**[] =

  {.....

```
#ifdef FEATURE_FTM_HALL
    item(ITEM_HALL, uistr_hall_test),
#endif
}
```

  If the test method of this item is “auto test”, you should also add it into **ftm_auto_test_items**[]

- **Step 4:Add english string and chinese string in “factory/inc/uistrings.h ”and “factory/inc/uistrings_chn.h”**

```
#define uistr_rf_test     "RF Test"
#define uistr_rf_c2k_test     "C2K RF test"
#define uistr_hall_test     "HALL Test"
```

```
#define uistr_rf_test     "RF Test"
#define uistr_rf_c2k_test     "C2K RF test"
#define uistr_hall_test     "霍尔测试"
```

Steps(3/7)

- **Step 5:Add module's implementation file and define your module's init function**

e.g. Add file ftm_hall.c under "factory/src/test/ftm_hall.c"

```
int hall_init(void)
{
 ......

if (!mod)
    return -ENOMEM;

ret = ftm_register(mod, hall_entry, (void*)dat);

return ret;

}
```

Steps(4/7)

- **Step 6:Accomplish your module's entry function and other functions(if necessary). Each module's entry function can be viewed as the main function of this module.**

e.g. In file ftm_hall.c

```
int hall_entry(struct ftm_param *param, void *priv)
{
    char *ptr;
    int chosen;
    struct acc_data *dat = (struct acc_data *)priv;
    struct textview *tv;
    struct itemview *iv;
    struct statfs stat;
    int err;

    LOGD(TAG "%s\n", __FUNCTION__);

    init_text(&dat->title, param->name, COLOR_YELLOW);
```

Steps(5/7)

- **Step 7:Add initialize code for hall_init in file “ftm_mods.cpp”**

e.g. In file ftm_modes.cpp

```
extern "C"{
#endif
extern int mcard_init(void);
extern int battery_init(void);
extern int gsensor_init(void);
extern int hall_init(void);
extern int gs_cali_init(void);
extern int msensor_init(void);
extern int flash_init(void);
```

ftm_init_fn **ftm_init_funcs[]**={

```
#ifdef FEATURE_FTM_HALL
    hall_init,
#endif

    NULL,
};
```

Steps(6/7)

- **Step 8: Add your module's source code file to the makefile**

file: factory/Android.mk

```
TEST_SRC_FILES := \
    src/test/ftm.cpp\
    src/test/ftm_sp_ata.cpp\
    src/test/ftm_mods.cpp\
    src/test/ftm_keys.c\
    src/test/ftm_lcd.c\
    src/test/ftm_lcm.c\
    src/test/ftm_backlight.c\
    src/test/ftm_led.c\
    src/test/ftm_memcard.c\
    src/test/ftm_rtc.cpp\
    src/test/ftm_gsensor.c\
    src/test/ftm_gs_cali.c\
    src/test/ftm_msensor.c\
    src/test/ftm_hall.c\
    src/test/ftm_touch.c\
    src/test/ftm_touch_auto.c\
```

Steps(7/7)

- **Step 9: Add test item in " proprietary/custom/{project}/factory/factory.ini" and "proprietary/custom/{project}/factory/factory.chn.ini"**

  The item name should be same as which define in **ftm_test_items[]**

```
MenuItem=EXT BUCK(A);
MenuItem=RF Test(A);
MenuItem=OTG(A);
MenuItem=HALL Test(M);
```

(M) — means mauanl test
(A) — means auto test

```
MenuItem=EXT BUCK(A);
MenuItem=RF Test(A);
MenuItem=OTG(A);
MenuItem=霍尔测试(M);
```

Add Test Item into SP ATA Test Mode

--Target Side

Steps(1/2)

- **Step1: Add test item and AT command in cmd_hldr[] under "factory/src/test/ftm_sp_ata.cpp"**

--The test item id must be the same as the id defined in **ftm_test_items[]** and **common.h**

--The command string must be the same as AT command received from PC tool.

static ftm_cmd_hdlr **cmd_hdlr**[]={

……

```
#ifdef FEATURE_FTM_VIBRATOR
    {18, ITEM_VIBRATOR, "AT+VIBRATOR", (hdlr)ftm_item_add_cb},
#endif

#ifdef FEATURE_FTM_LED
    {19, ITEM_LED, "AT+LED", (hdlr)ftm_item_add_cb},
#endif

#ifdef FEATURE_FTM_RECEIVER
    {20, ITEM_RECEIVER, "AT+RECEIVER", (hdlr)ftm_item_entry_cb},
#endif

#ifdef FEATURE_FTM_HALL
    {37, ITEM_HALL, "AT+HALL", (hdlr)ftm_item_entry_cb},
#endif
```

ftm_item_add_cb—For items need to be stoped by command"AT+XXXX=STOP"

ftm_item_entry_cb—for other items no need to be stoped by sending"AT+XXXX=STOP"

Steps(2/2)

- **Step2:This step is optional. If the test item need to return some data to ata tool for further judge the result,You should following below steps.**

---In module's source file ftm_xxx.c (e.g. ftm_gsensor.c)

Add "**extern sp_ata_data return_data**;" in the begin.

--Assign return value in the array "**return_data** "at the property place where need to return data.

e.g. in ftm_gsensor.c, return coordinate to pc side

```
//add sensor data to struct sp_ata_data for PC side
return_data.gsensor.g_sensor_x = acc->evt.x;
return_data.gsensor.g_sensor_y = acc->evt.y;
return_data.gsensor.g_sensor_z = acc->evt.z;
return_data.gsensor.accuracy = 3;
```

ATA Tool Customization Guide

--Tool Side

Environment

- **ATA Tool Development environment is Microsoft Visual C++ 6.0**

- **Options for build ATA2.0 tool**

  --In file “ATA_DLL.h” ,set options as below. Please note that the code is wrapped in these option.

```
//#define INTERNEL_DEBUG_VERSION
#define __ATA_CONCURRENT_FLOW__
#define __ATA_LOAD30_TEST20__
#define __ATA20__
//#define __ATA30__
//#define __ATA_MT6795_SMT__
```

Steps(1/10)

- **Step 1: UI exe—Config Dialog**

  --Add new check box to indicate the new test item.

  --Add new variable in MFC ClassWizard.

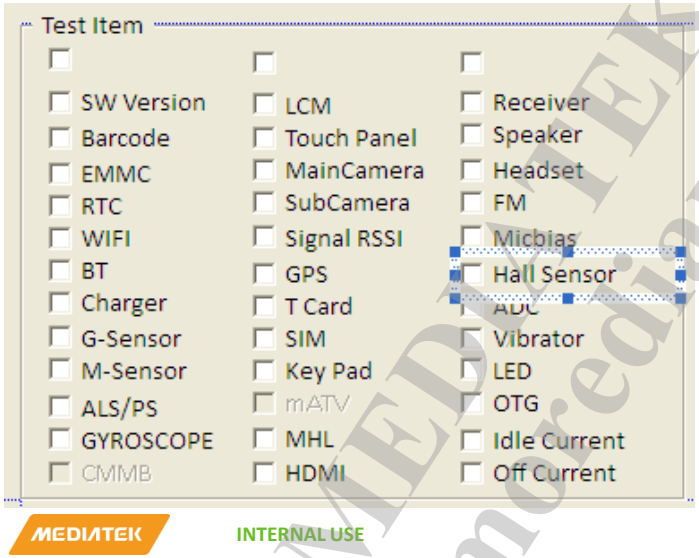
  --Update related functions in TestItemConfig.cpp

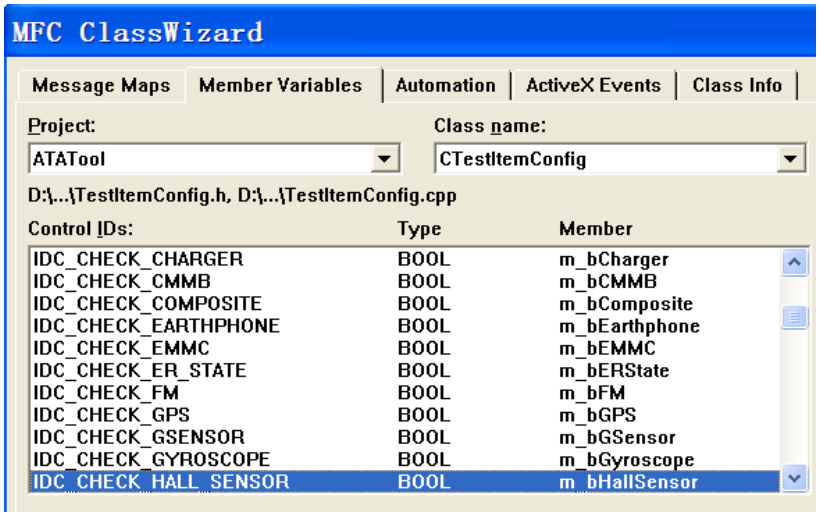
  void CTestItemConfig::OnSave()

  void CTestItemConfig::UpdateTestItem()

  --Add new property page to set limited value(optional)

  Max & min value for pass or fail





Steps(2/10)

- **Step 2: UI exe—Show Item**

  --Modify TestItem.ini which is under tool's folder.

  e.g. add **TestItem34**

  [Foreground Items]

  ….

  TestItem29 = LCDBacklight, 0

  TestItem30 = LED, 0

  TestItem31 = Mic Bias, 0

  TestItem32 = Off Current, 0

  TestItem33 = MHL, 0

  TestItem34 = Hall Sensor,0

Please note that the item id can not be repeated in the same section.

--Modify GetItemIndex() Function in ATA_DLL_Handle.cpp

```
E_ATDLL_TEST_ITEM_COUNT ATA_DLL_Handle::GetItemIndex(string item_name)
{
    E_ATDLL_TEST_ITEM_COUNT enum_index = E_TEST_ITEM_COUNT;
    for (enum_index = E_LCD; enum_index < E_TEST_ITEM_COUNT; enum_index = (E_ATDl
    {
        if (strstr(item_name.c_str(), test_item_name[enum_index]) != NULL)
        {
            bool bSelected = false;
            switch (enum_index)
            {
            case E_LCD:
                if (m_testItemCFG.b_LCD) bSelected = true;
                break;
```

Steps(3/10)

--In **ATA_DLL.h** add TestItem and item bool type.

```
    E_GYROSCOPE,

    E_OTG,
    E_HDMI,
    E_MHL,

    E_SLPMODE,
    E_OFFMODE,
    E_HALL,
    E_TEST_ITEM_COUNT  //
} ? end E_ATDLL_TEST_ITEM_COUNT ?  E_ATDLL_TEST_ITEM_COUNT;
```

```
    bool    b_MHL;
    bool    b_RSSI;
    bool    b_offMode;
    bool    b_hall;
} ? end {anonS_ATADLL_TEST_ITEM_T} ?  S_ATADLL_TEST_ITEM_T;
```

--Modify **ATA_DLL_Handle.cpp** and add item name in **test_item_name**

char test_item_name[E_TEST_ITEM_COUNT+1][64] =

{……

```
    "MHL",

    "Idle Current",
    "Off Current",

    "Hall Sensor",

    "End"
```

Please note that item's sequence in **test_item_name** is the same with that in **E_ATDLL_TEST_ITEM_COUNT** .The name in **test_item_name** should be same with **TestItem.ini**

Steps(4/10)

- **Step 3: return data and test result**

  --In **ATA_DLL.h**,add structure for return data.

  Please note that the structure should be same as that defined in target side(**ftm.h**)

  e.g. EMMC

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

```
        ftm_ata_thd headsetL_thd;
        ftm_ata_thd headsetR_thd;
#endif
        //ftm_ata_imei imei;
        ftm_ata_memcard memcard;
        ftm_ata_emmc emmc;

#ifdef __ATA_LOAD30_TEST20__
        ftm_ata_rf rf;
#endif
        ftm_ata_hall hall;
    } ? end {anonsp_ata_data} ?  sp_ata_data;
```

  --In ATA_DLL.h,add members in **S_ATADLL_TEST_CNF**

  e.g. EMMC

```
typedef struct
{
    bool result;
    ftm_ata_emmc emmc;
} S_ATADLL_EMMC_CNF;
```

```
    S_ATADLL_KPAD_CNF         kpad_cnf;
    S_ATADLL_ADC_CNF          adc_cnf;
    S_ATADLL_SLEEPMODE_CNF    slpmode_cnf;
    S_ATADLL_RSSI_CNF         rssi_cnf;
    S_ATADLL_EMMC_CNF         emmc_cnf;
    S_ATADLL_FLASH_CNF        flash_cnf;
    S_ATADLL_SWVER_CNF        sw_ver_cnf;
    S_ATADLL_CMMB_CNF         cmmb_cnf;
    S_ATADLL_LED_CNF          led_cnf;
    S_ATADLL_SENSOR_STATE     sensor_cnf;
    S_ATADLL_MICBIAS_CNF      micbias_cnf;
    S_ATADLL_OFFMODE_CNF      offmode_cnf;
    S_ATADLL_IMEI_CNF         imei_cnf;
    S_ATADLL_MEMCARD_CNF      memcard_cnf;
    S_ATADLL_OTG_CNF          otg_cnf;
    S_ATADLL_HDMI_CNF         hdmi_cnf;

    E_ATADLL_RESULT      test_result[E_TEST_ITEM_COUNT];
} ? end {anonS_ATADLL_TEST_CNF} ?  S_ATADLL_TEST_CNF;
```

Steps(5/10)

- **Step 4: Specific Design**

  --This step is needed if you want to judge result by comparing target return value with preset value in the UI

  --In ATA_DLL.h,modify **S_ATADLL_COMMON_CFG_T** and add item's special structure.

  e.g. G-Sensor

```
typedef struct
{
    bool bX;
    bool bY;
    bool bZ;

    bool bGSensorValueDiff;
} S_ATADLL_GSENSOR_CFG;
```

```
    S_ATADLL_ADC_CFG        adc_cfg;
    S_ATADLL_GSENSOR_CFG        gsensor_cfg;
    S_ATADLL_ALSPS_CFG          alsps_cfg;
    S_ATADLL_BARCODE_FLAG_CFG   barcodeFlag_cfg;
    S_ATADLL_VIBRATOR_CFG       vibrator_cfg;
    S_ATADLL_WIFI_CFG           wifi_cfg;
    S_ATADLL_CAMERA_CFG         camera_cfg;

    int     bt_spec_type;   // 0 : search all bt addr
    // 1 : search any bt addr then return

    int     *stop_flag;
    bool    stop_if_failed;
    int     waitSecBeforeTest;

    CallbackPreProcess          cbTestPreProcess;
    CallbackTestPregoress       cbTestProgess;
    CallbackQueryTestItemResult cbQueryTestItemResult;
    CallbackUpdateTestResultToUI    cbUpdateResult;
} ? end {anonS_ATADLL_COMMON_CFG_T} ?  S_ATADLL_COMMON_CFG_T;
```

Steps(6/10)

--In ATA_DLL.h,modify **S_ATADLL_TESTITEM_SPEC** and add item's special structure.

e.g. G-Sensor

```
typedef struct
{
    ftm_ata_gsensor max_gsensor;
    ftm_ata_gsensor min_gsensor;
} S_ATADLL_GSENSOR_SPEC;
```

```
    S_ATADLL_GSENSOR_SPEC    gsensor_spec;
    S_ATADLL_MSENSOR_SPEC    msensor_spec;
    S_ATADLL_ALSPS_SPEC      alsps_spec;
    S_ATADLL_GYROSCOPE_SPEC  gyroscope_spec;
    S_ATADLL_MICBIAS_SPEC    micbias_spec;
    S_ATADLL_OTG_SPEC        otg_spec;
    S_ATADLL_OFFMODE_SPEC    offmode_spec;
? end {anonS_ATADLL_TESTITEM_SPEC} ?  S_ATADLL_TESTITEM_SPEC;
```

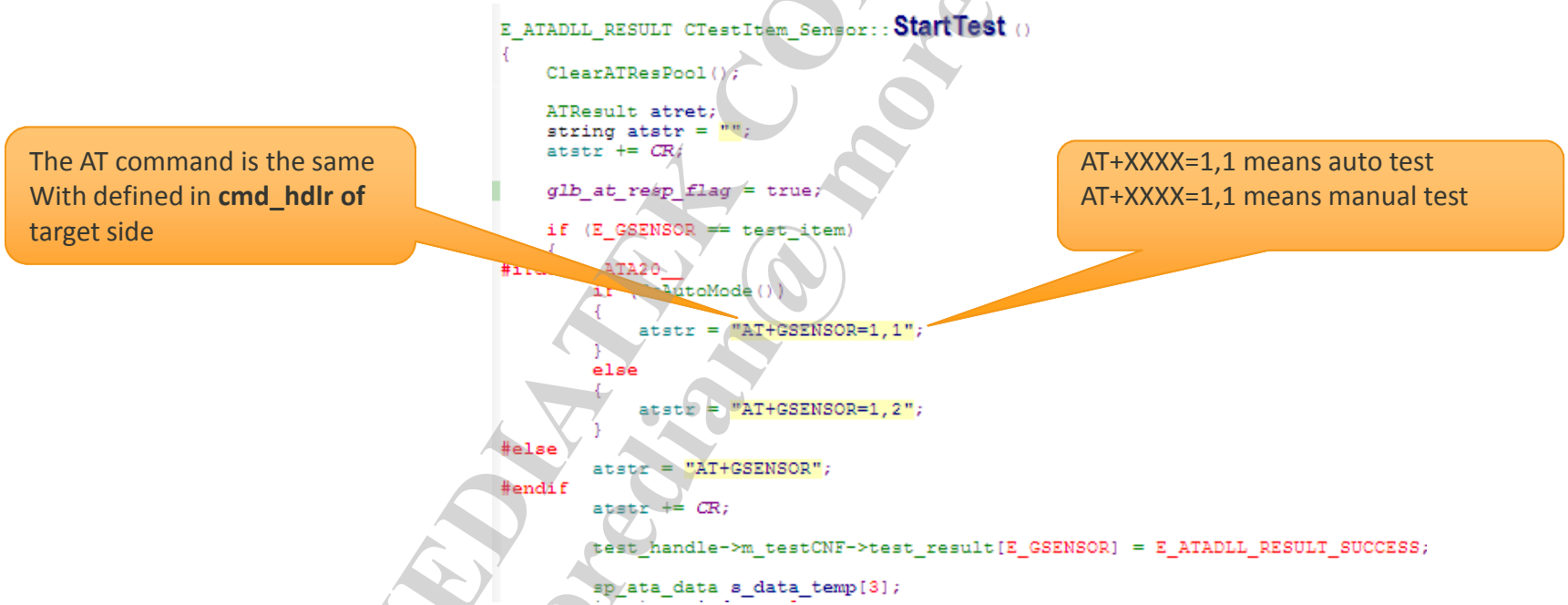
Steps(7/10)

- **Step 5: Add TestItem related files**

  --Add or modify test item related files under directory "**ATA_DLL/TestItem**"

  --If necessary,you can add new test item file named "**ATA_DLL_TestItem_xxxx.cpp** "and "**ATA_DLL_TestItem_xxxx.h**"

  --Add implement code in these files,you can refered to other test items.



Steps(8/10)

- **Step 6: Initialize object for new item in "ATA_DLL_Handle.cpp"**

  --bool ATA_DLL_Handle::**InitTestItemObj** ()

  {

  #ifdef __ATA_CONCURRENT_FLOW__

  ……

```
case E_HALL:
     g_TestItemObj[handle_index][E_HALL] =
        new CTestItem_Sensor(this, m_commonCFG.timeout[E_HALL], E_HALL, "Hall Sensor");

    if (g_TestItemObj[handle_index][E_HALL] == NULL)
    {
        return false;
    }
    break;
```

  }

Steps(9/10)

- **Step 7: Add ITEM_SEQUENCE and flag in "ATA_DLL_Handle.cpp"**

  Note:The item's sequence should be the same with target side's define (factory/inc/common.h). Otherwise, you may get mismatch result from target.

```
typedef enum
{
.......
} ITEM_SEQUENCE;

int g_callback_flag[ITEM_MAX_IDS] =
{
......
}
```

Steps(10/10)

- **Step 8:Quary and update result**

  ATA_DLL_Handle.cpp

  *--static void **ATACallBackOfAT(**ATResult& atret, int handle_index)*

  TestObject.cpp

  *--bool CTestObject::**QueryTestItemResult** (E_ATDLL_TEST_ITEM_COUNT item)*

  *--void CTestObject::**UpdateTestResultToCtrlList_Sub***
  (E_ATDLL_TEST_ITEM_COUNT item)

everyday genius