Programming Guide

Programming Guide

Customer Support

MT6765/MT6762/MT6761

Doc No: CS6765-AZ6A-PGD-V1.0EN
Version: V1.0
Release date: 2018-06-06
Classification: internal

Keywords
Programming Guide

MediaTek Inc.

Postal address
No. 1, Dusing 1st Rd. , Hsinchu Science Park, Hsinchu City, Taiwan 30078

MTK support office address
No. 1, Dusing 1st Rd. , Hsinchu Science Park, Hsinchu City, Taiwan 30078

Internet
http://www.mediatek.com/

Document Revision History

Revision	Date	Author	Description
V1.0	2016-06-06	Weiqi Fu	Initial Release

Table of Contents

1 Introduction

☞ [Random filler text. Not intended for actual reading.] Must keep the chapter even it have empty content.

1.1 Purpose

The following topics are included in the documentation.

- MTK sensor architecture
- Porting guide
- API interface
- Build option
- Debug method

The Introductions are from AP side and from SCP side.

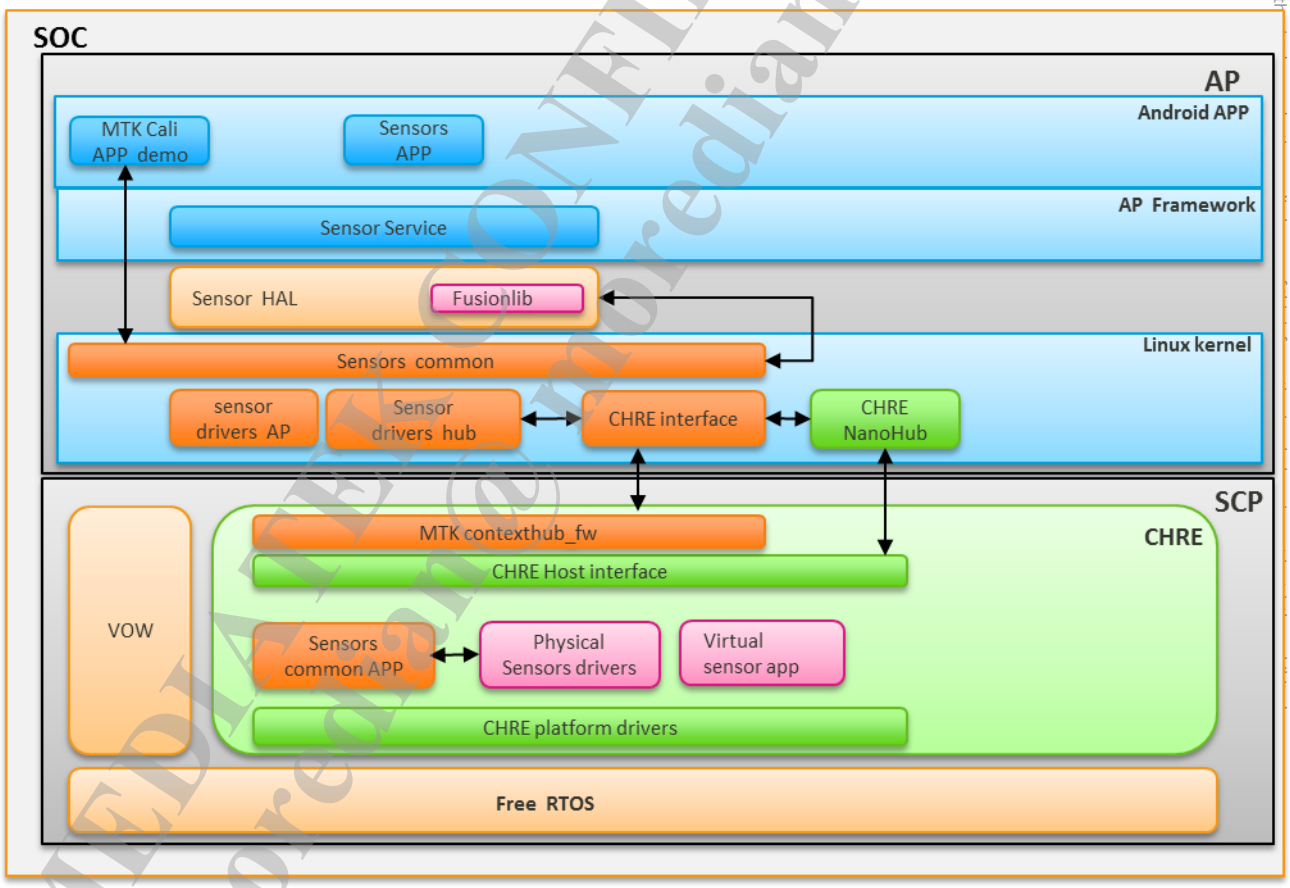
This doc is based on MTK 6762 platform

2 Overview

This chapter first gives a brief description of the modules of the system and the relationship of the modules.

2.1 Architecture

Note:
Orange: MTK maintain
Purple: MTK or 3rd party
Blue: Google default
Green: Google CRHE AOSP
CHRE: Open source



MTK Sensor can be divided into two parts: AP(ca5x, CA7x seires main processor), SCP(CM4 coprocessor) . They work together to deal with sensor data.

Another scheme is to use AP only without SCP to implement sensor function

2.2 Source Code Organization

Sensor Hub

- alps/vendor/mediatek/proprietary/tinysys/freertos/source

Kernel code

- alps/kernel-3.18/drivers/misc/mediatek/sensors-1.0/
- alps/kernel-4.4/drivers/misc/mediatek/sensors-1.0/
- alps/device/mediatek/common/kernel-headers/linux/hwmsensor.h
- kernel-4.9/drivers/staging/nanohub

HAL code

- alps/vendor/mediatek/proprietary/hardware/sensor
- alps/vendor/mediatek/proprietary/hardware/libsensor (third party library)
- alps/device/mediatek/MTxxxx/device.mk
- alps/device/mediatek/MTxxxx/manifest.xml
- alps/device/mediatek/MTxxxx/init.sensors_1_0.rc
- alps/device/mediatek/common/kernel-header/linux/hwmsensor.h
- alps/hardware/interface/sensors/
- alps/device/mediatek/sepolicy/basic/non-plat/

Figure 2-1. Source Code Directory Structure

3 AP sensors introduction

This section specifies sensor architecture in AP part which include: kernel driver and interface, HAL Layer and porting guide.

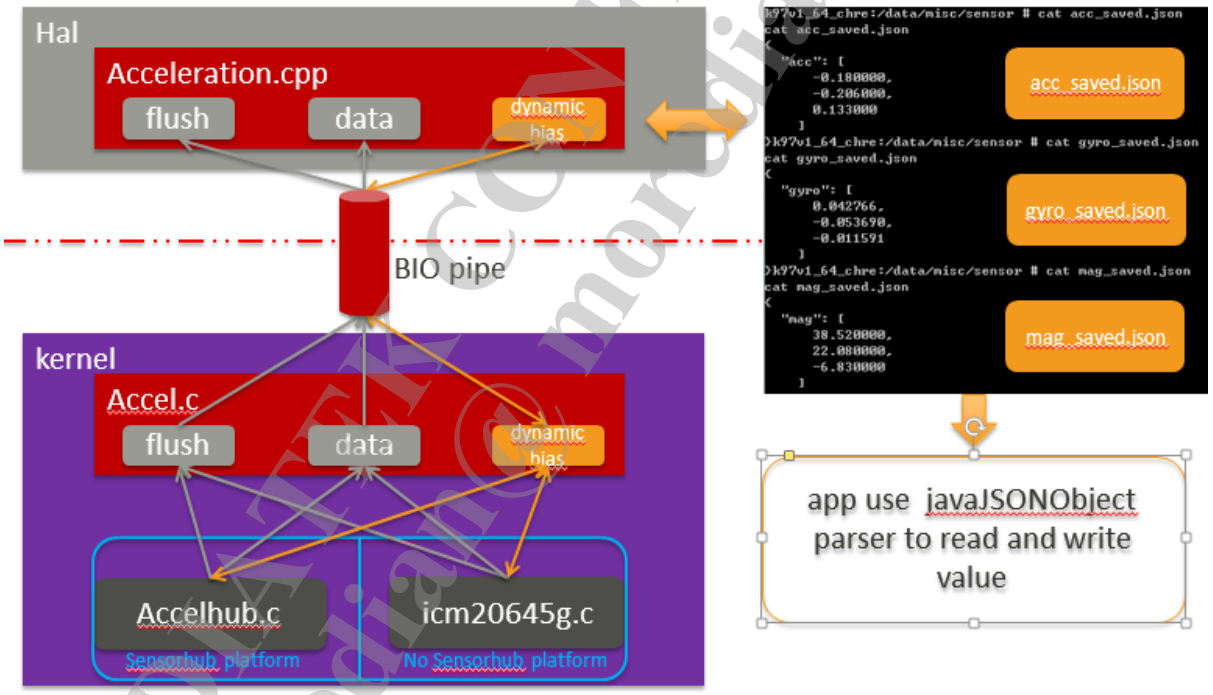
3.1 Linux sensors drivers interface

This section introduces MTK kernel common sensor Interface， and porting guide.

3.1.1 MTK sensor common layer introduction

The purpose of common layer is to simplify porting work.

On the other hand, the communications between common layer and MTK HAL is using MTK exclusive interface (BIO pipe) which is much more efficient than input event.



3.1.2 Common layer API Introduction

The next part introduces how to use common API and accelerometer is used as an example.

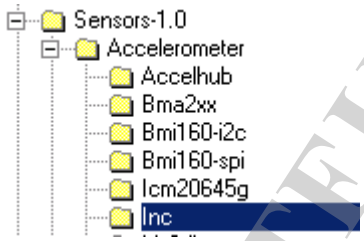
APIs are provided by two header files.

1. **Accel.h,** provide data flow and control flow API to Android layer.
2. **Acc_factory.h** provide data flow and control flow 的 API to MTK factory mode.

Accel.h imlements API as following. It provides data report patch to Android layer:

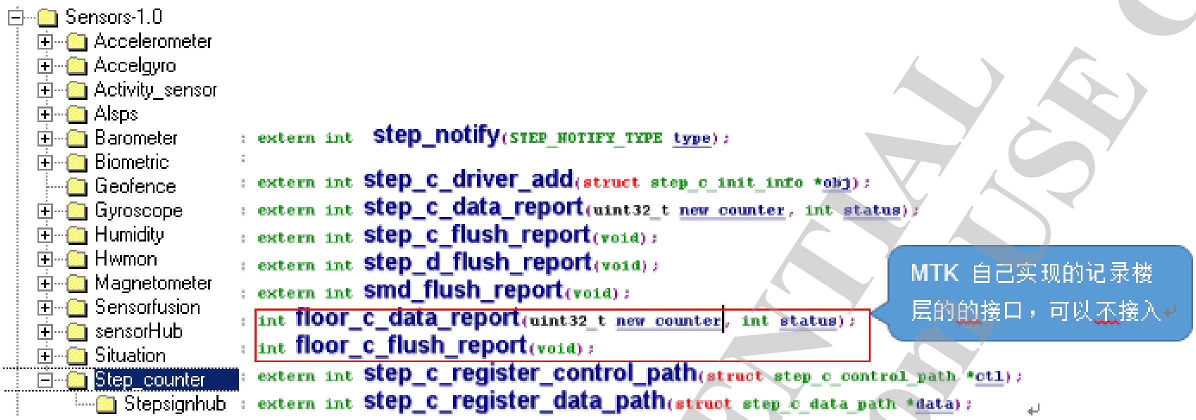
API Name	Parameter description
acc_driver_add(struct acc_init_info* obj)	acc_init_info includes sensor related information. Sensor auto detect can be implements by registering this structure to common driver.
acc_register_data_path(struct acc_data_path *data)	acc_data_path includes a function to get sensor data and two parameters to normalize sensor raw data from different vendor. These two parameters are registered to common sensor driver architecture to get sensor data for MTK architecture.
acc_register_control_path(struct acc_control_path *ctl)	acc_control_path includes 3 functions and a boolean parameter to control sensor to enable/disable, setDelay and so on.
acc_data_report()	Report data
Acc_flush_report()	Report flush

The structure to pass to the API could be referred to an already implemented driver.



3.1.3 Step counter access

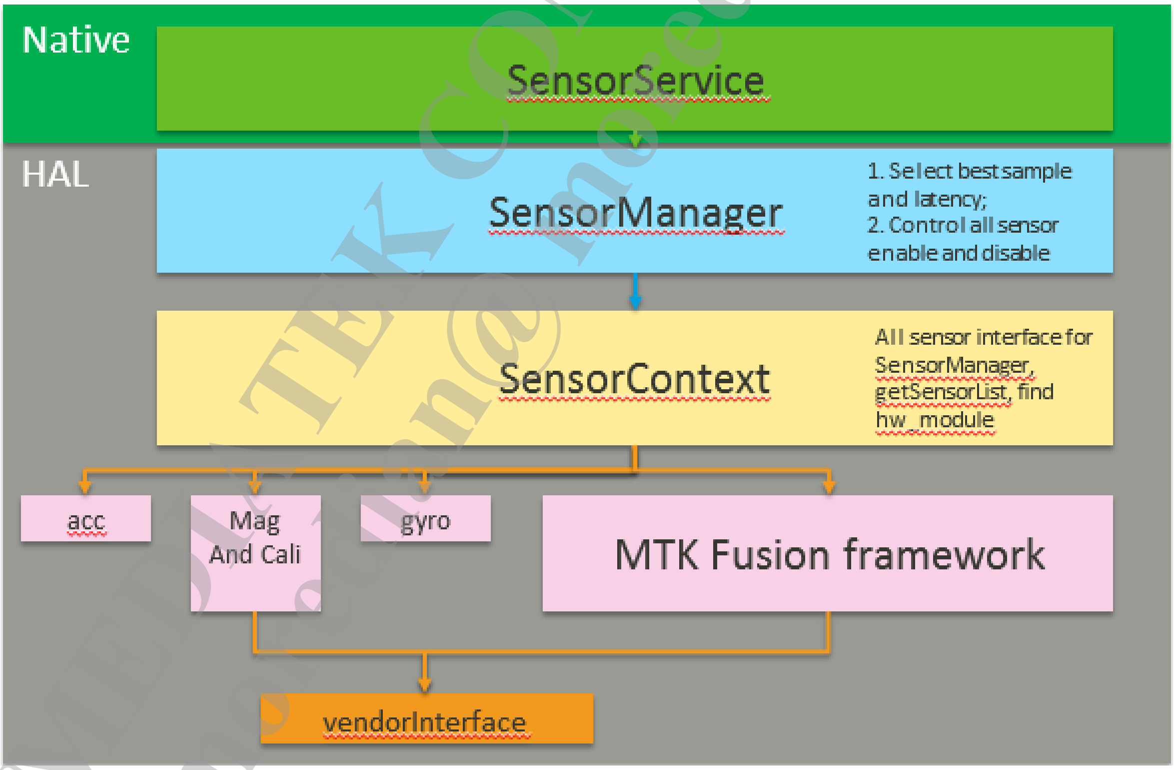
This section introduces, based on accelerometer which comes with acc, how to access step counter data. The interface in the following folder could be referred and is similar to the acc common interface as introduced before.



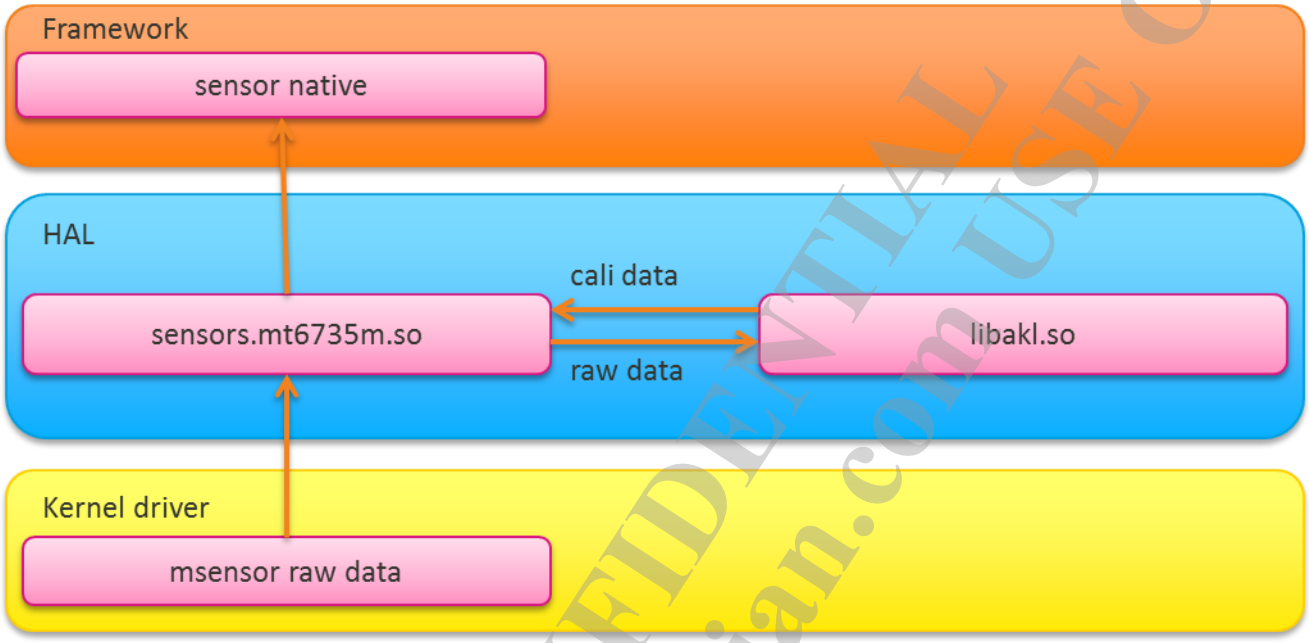
3.2 Msensor lib porting guide

MTK HAL provides third-party algorithm interface according to different needs from customers. Customers may want to implement a series of algorithm such as fusion, virtual gyro and so on in native layer.

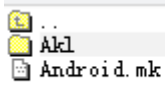
MTK HAL architecture is as following：



Msensor lib communicate with MTK lib as following:



1. Put msensor souce and lib at following folder
   vendor/mediatek/proprietary/hardware/libsensor



2. device/mediatekprojects/&project/ProjectConfig.mk
   Or device/mediateksample/&project/ProjectConfig.mk

3. vendor/mediatek/proprietary/hardware/libsensor/akl/Android.mk

```
LOCAL_SHARED_LIBRARIES := liblog
LOCAL_MODULE := libakl
LOCAL_PROPRIETARY_MODULE    := true
LOCAL_MODULE_OWNER := mtk
LOCAL_MODULE_TAGS := optional
```

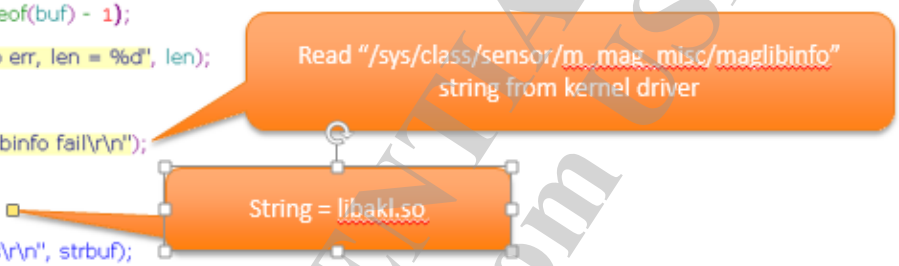
4. vendor/mediatek/proprietary/hardware/sensor/sensors-1.0/Android.mk

```
LOCAL_MODULE := sensors.$(TARGET_BOARD_PLATFORM)
LOCAL_PROPRIETARY_MODULE := true
LOCAL_MODULE_OWNER := mtk
LOCAL_MODULE_TAGS := optional
$(foreach custom_hal_msensorlib,$(CUSTOM_HAL_MSENSORLIB),$(eval LOCAL_REQUIRED_MODULES += lib$(custom_hal_msensorlib)))
```

   Build sensors.&project.so auto link libakl.so

5. vendor/mediatek/proprietary/hardware/sensor/sensors-1.0/VendorInterface.cpp

```
#ifndef CUSTOM_KERNEL_SENSORHUB
    char libinfos[64] = {"/sys/class/sensor/m_mag_misc/maglibinfo"};
    char buf[20] = {0};
    char strbuf[64] = {0};
    int len = 0;
    fd = open(libinfos, O_RDWR);
    if (fd >= 0) {
        len = read(fd, buf, sizeof(buf) - 1);
        if (len <= 0) {
            ALOGD("read libinfo err, len = %d", len);
        }
        close(fd);
    } else
        ALOGE("open vendor libinfo fail\r\n");

    strcpy(strbuf, "lib");
    strcat(strbuf, buf);
    strcat(strbuf, ".so");
    //ALOGE("yue strbuf=%s\r\n", strbuf);

    //void *handle = dlopen("libcalmodule_akm.so", RTLD_NOW);
    void *handle = dlopen(strbuf, RTLD_NOW);
    if (!handle) {
        ALOGE("dlopen fail\n");
        return;
    }
```



6. Lib interface, path:

vendor/mediatek/proprietary/hardware/sensor/sensors-1.0/include/mag_calibtation_lib.h

```
struct magCaliDataOutPut {
    int64_t timeStamp;
    float x, y, z;
    float x_bias, y_bias, z_bias;
    int8_t status;
};
struct magCaliDataInPut {
    int64_t timeStamp;
    float x, y, z;
    int32_t status;
};
struct magChipInfo {
    int32_t layout;
    int32_t deviceid;
};
struct mag_lib_interface_t {
    const char *module;
    int (*initLib)(struct magChipInfo *info);
    int (*caliApiGetOffset)(float offset[3]);
    int (*caliApiSetOffset)(float offset[3]);
    int (*caliApiSetGyroData)(struct magCaliDataInPut *inputData);
    int (*caliApiSetAccData)(struct magCaliDataInPut *inputData);
    int (*caliApiSetMagData)(struct magCaliDataInPut *inputData);
    int (*doCaliApi)(struct magCaliDataInPut *inputData,
        struct magCaliDataOutPut *outputData);
    int (*getGravity)(struct magCaliDataOutPut *outputData);
    int (*getRotationVector)(struct magCaliDataOutPut *outputData);
    int (*getOrientation)(struct magCaliDataOutPut *outputData);
    int (*getLinearaccel)(struct magCaliDataOutPut *outputData);
    int (*getGameRotationVector)(struct magCaliDataOutPut *outputData);
    int (*getGeoMagnetic)(struct magCaliDataOutPut *outputData);
    int (*getVirtualGyro)(struct magCaliDataOutPut *outputData);
};
```

Interface implementation in your libxxx.so, path:vendor/mediatek/proprietary/hardware/libsensor/xxx

```
const struct mag_lib_interface_t MAG_LIB_API_INTERFACE = {
    .module = "libakl",
    .initLib = AKMInitLib,
    .caliApiGetOffset = AKMCaliApiGetOffset,
    .caliApiSetOffset = AKMCaliApiSetOffset,
#if 0
    .caliApiSetGyroData = AKMCaliApiSetGyroData,
    .caliApiSetAccData = AKMCaliApiSetAccData,
#endif
    .doCaliApi = AKMDoCaliApi,
};
```

This can't be changed

7. Set follow macro in vendor/mediatek/proprietary/custom/{project}/hal/sensors/sensor/hwmsen_custom.h

- MAG_CALIBRATION_IN_SENSORHUB // need define if do mag cali in sensor hub
- MAG_CALIBRATION_FAST // if use acc and gyro do mag cali need define this
- FUSION_ALGORITHM_IN_SENSORHUB // if implement fusion lib in sensor hub ,need define
- FUSION_SUPPORT_9D_ALGORITHM //open it if AP fusion use gyro , beside acc and mag
- VIRTUAL_GYROSCOPE_ALGORITHM // if support Virtual gyro we need define this
- ACC_GYRO_CALIBRATION_IN_SENSORHUB // no need define if no sensor hub
- ACC_GYRO_CALIBRATION_SUPPORT // no need define

3.3 Sensors Calibration Interface

MTK provides Calibration interface that can be used by Android APP. Now it supports ACC and Gyro zero calibration, and Proximity threshold calibration. MTK provides engineer mode APK as an example to use these APIs.

3.3.1 Debug Command

Using adb cmd to do accel and gyroscope calibration(the calibration is done in adb command window):

1、adb root

2、adb shell

3、Case Accel calibration ：cd sys/bus/platform/drivers/gsensor

Case Gyro calibration ：cd sys/bus/platform/drivers/gyroscope

4、Do Calibration. Put the phone flat, and DO NOT HAVE any even slight movement ： echo 1 > test_cali

```
k71v1_64_bsp:/vendor/nvcfg/sensor #
```

5. Check calibration result ：

```
1|k71v1_64_bsp:/vendor/nvcfg/sensor # cat *
{
  "acc_bias": [
      0.187000,
      -0.459000,
      0.008000
    ]
}{
  "acc_cali": [
      185,
      -460,
      10
    ]
}k71v1_64_bsp:/vendor/nvcfg/sensor # cd ..
```

3.3.2 Accel & Gyro Zero Calibration Interface

Native API Path：vendor\mediatek\proprietary\external\sensor-tools\include\libhwm.h

Accel Calibration API

Int gsensor_start_static_calibration(void);

- ✓ Return zero or non-zero. If zero: success; if non-zero: failure.

Int gsensor_get_static_calibration(sensorData *sensorDat);

- ✓ Return zero or non-zero. If zero: success; if non-zero: failure.

  Passed parameter is used to send back the calibration data to be displayed on the UI.,x=sensorDat->data[0], y=sensorDat->data[1], z=sensorDat->data[2]

- ✓ Only when gsensor_start_static_calibration returns true, this api can be used to get static calibration data.

Gyroscope Calibration API

Int gyroscope_start_static_calibration (void);

- ✓ Return zero or non-zero. If zero: success; if non-zero: failure.

Int gyroscope_get_static_calibration (sensorData *sensorDat);

- ✓ Return zero or non-zero. If zero: success; if non-zero: failure.
- ✓ Passed parameter is used to send back the calibration data to be displayed on the UI,x=sensorDat->data[0], y=sensorDat->data[1], z=sensorDat->data[2]
- ✓ Only when gyroscope_start_static_calibration, this api can be used to get static calibration data.

3.3.3 Proximity Threshold Calibration Interface

Proximity threshold calibration API locates in the same place as Accel and Gyro.

Demo：Enter Engineer Mode: Phone Call Display + “*#*#3646633#*#*” to enter Engineer Mode

Ehen encountering proximity issue, the first interface is used to do calibration and the second one is used to set threshold manually.

Native API Path：vendor\mediatek\proprietary\external\sensor-tools\include\libhwm.h

int get_psensor_data(void)

- ✓ Return psensor raw data

Int set_psensor_threshold(int high, int low)

- ✓ High and low two data are stored in the nvram and can be passed to bottom driver.
- ✓ For example in 36658.c

  When phone power on to home screen, the scp dirver will received the Cali data you reserved at last calibration.

```
/* ps cali state */
sensorFsmCmd(STATE_PS_CALI, CHIP_PS_CALI_DONE, cm36558_ps_cali),
/* ps cfg state */
sensorFsmCmd(STATE_PS_CFG, CHIP_PS_CFG_DONE, cm36558_ps_cfg),
```

3.3.4 Ligth calibration interface

Int als_start_static_calibration(void);

- ✓ Call this native API , SCP driver will received the Cali cmd

✓ For example 36658 driver ‘s function cm36658_als_cali will be called

```
/* als cali state */
sensorFsmCmd(STATE_ALS_CALI, CHIP_ALS_CALI_DONE, cm36558_als_cali),
```

You can implement your cali algo in the below function cm36558_als_cali(), this is a demo function , you need to do the cali function by your self

```
static int cm36558_als_cali(I2cCallbackF i2cCallBack, SpiCbkF spiCallBack, void *next_state,
                            void *inBuf, uint8_t inSize, uint8_t elemInSize,
                            void *outBuf, uint8_t *outSize, uint8_t *elemOutSize)
{
    //sensorFsmEnqueueFakeI2cEvt(i2cCallBack, next_state, SUCCESS_EVT);
    int32_t alsCali[2];
    float   alsCali_mi;

    alsCali_mi = 0.345;
    alsCali[0] = alsCali_mi * ALS_INCREASE_NUM_AP;
    alsCali[1] = 0;

    sendAlsPsCalibrationResult(SENS_TYPE_ALS, alsCali);
    sensorFsmEnqueueFakeI2cEvt(i2cCallBack, next_state, SUCCESS_EVT);
    return 0;
}
```

After calibration，when power on you can get the cali value in cm36558_als_cfg()

```
static int cm36558_als_cfg(I2cCallbackF i2cCallBack, SpiCbkF spiCallBack, void *next_state,
                           void *inBuf, uint8_t inSize, uint8_t elemInSize,
                           void *outBuf, uint8_t *outSize, uint8_t *elemOutSize)
{
    int ret = 0;
    struct alspsCaliCfgPacket caliCfgPacket;
    double  alsCali_mi;

    ret = rxCaliCfgInfo(&caliCfgPacket, inBuf, inSize, elemInSize, outBuf, outSize, elemOutSize);

    if (ret < 0) {
        sensorFsmEnqueueFakeI2cEvt(i2cCallBack, next_state, ERROR_EVT);
        osLog(LOG_ERROR, "cm36558_als_cfg, rx inSize and elemSize error\n");
        return -1;
    }
    alsCali_mi = (double)((float)caliCfgPacket.caliCfgData[0] / ALS_INCREASE_NUM_AP);
    osLog(LOG_INFO, "cm36558_als_cfg: [%f]\n", alsCali_mi);
    mTask.alsCali = caliCfgPacket.caliCfgData[0];
    sensorFsmEnqueueFakeI2cEvt(i2cCallBack, next_state, SUCCESS_EVT);
    return 0;
} ? end cm36558_als_cfg ?
```

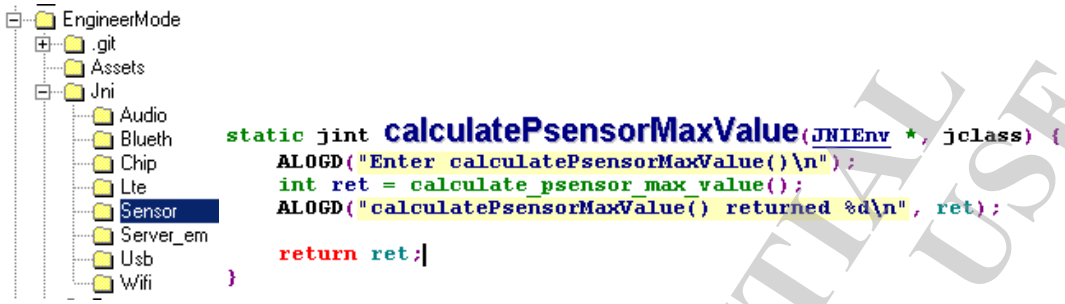
Int als_get_static_calibration(void);

You can call this function to show the cali value

```
struct caliData caliData;
int ret = als_get_static_calibration (&caliData)
```

3.3.5 Engineer mode Demo code example（*#*#3646633#*#*）

- Native Layer interface using example : JIN encapsulates API to be used by APP.

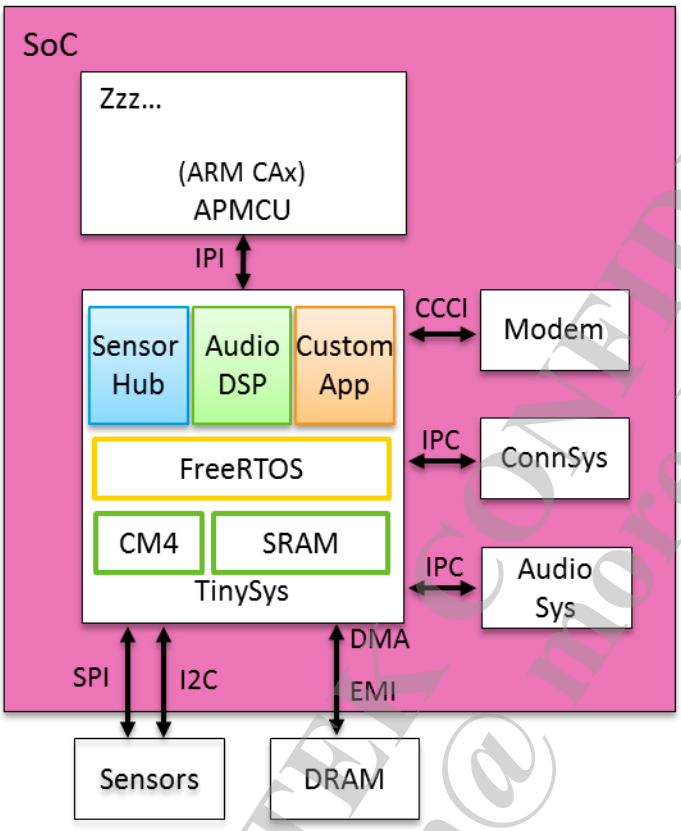


Reference Path：vendor\mediatek\proprietary\external\apps\engineerMode

4 SCP Introduction

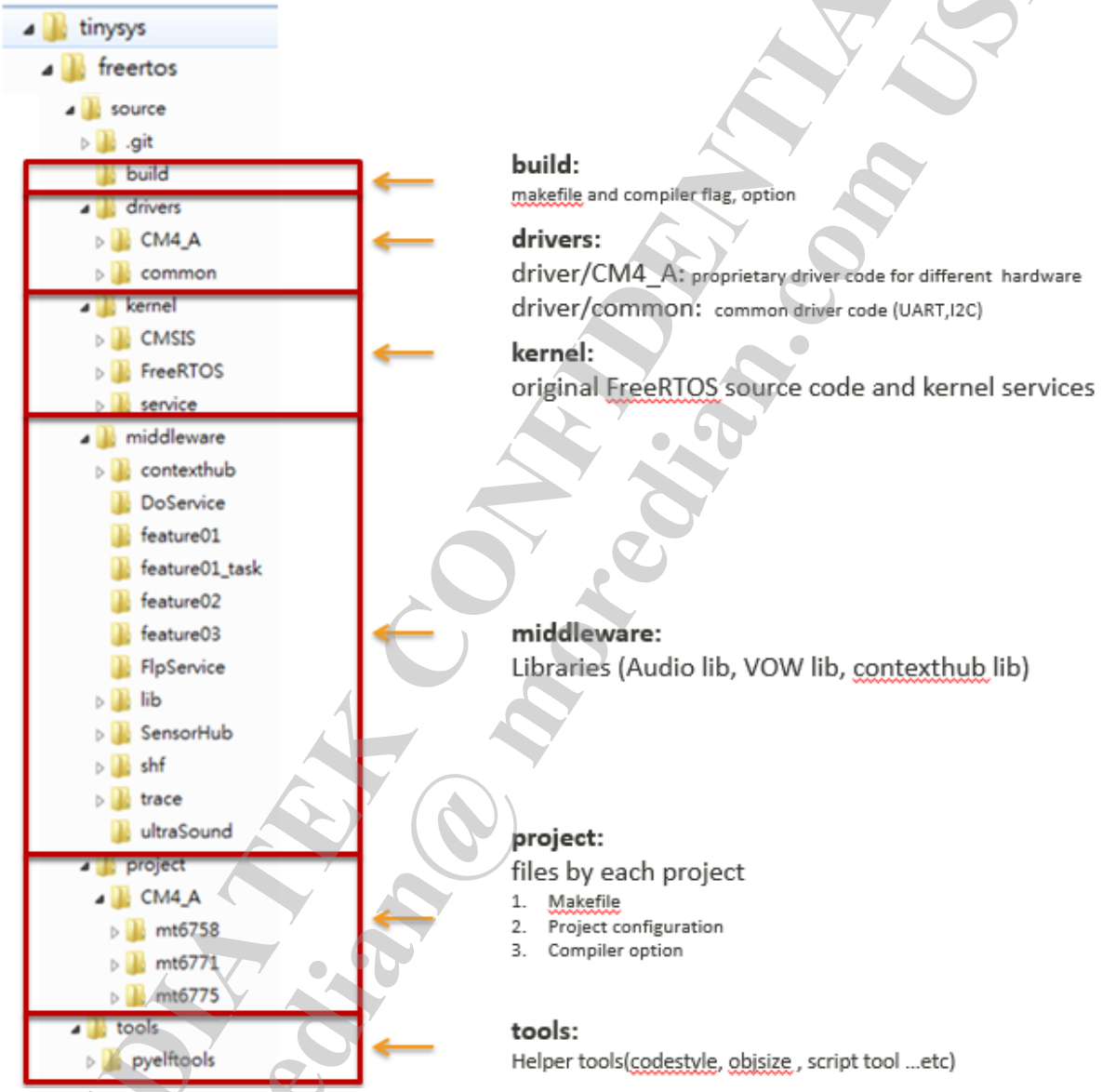
SCP （Tinysys）co-processor is in charge of sensor and audio related features and other customized feature.

MTK SCP chooses FreeRTOS as os and CHRE is a specialized Task to deal with Sensor related data in FreeRTOS. Audio feature is developed on FreeRTOS directly.



4.1 Tinysys introduction

4.1.1 Folder Structure



4.1.2 Configuration files

1. **Platform Configuration file**

   Required LDFLAGS, headers or C files of the platform

   Default configurations of the platform

   Path：project/$(PROCESSOR)/$(PLATFORM)/platform/platform.mk

   As following：

```
##############################################################
# Mandatory platform-specific resources
##############################################################
INCLUDES += \
  $(PLATFORM_DIR)/inc \
  $(SOURCE_DIR)/kernel/service/common/include \
  $(SOURCE_DIR)/kernel/CMSIS/Device/MTK/$(PLATFORM)/Include \
  $(SOURCE_DIR)/middleware/SensorHub \
  $(DRIVERS_PLATFORM_DIR)/feature_manager/inc

C_FILES += \
  $(PLATFORM_DIR)/src/main.c \
  $(PLATFORM_DIR)/src/platform.c \
  $(PLATFORM_DIR)/src/interrupt.c \
  $(SOURCE_DIR)/kernel/service/common/src/mtk_printf.c \
  $(SOURCE_DIR)/kernel/service/common/src/wakelock.c \
  $(PLATFORM_DIR)/src/scp_it.c \
  $(DRIVERS_PLATFORM_DIR)/feature_manager/src/feature_manager.c
```

2. **Project Configuration file**

ProjectConfig.mk will **overwriting** options in platform.mk

```
CFG_MTK_VOW_SUPPORT = no

CFG_CHRE_SUPPORT = yes
CFG_CONTEXTHUB_FW_SUPPORT = yes
CFG_ACCGYRO_SUPPORT = yes
CFG_LSM6DSM_SUPPORT = yes
CFG_ALSPS_SUPPORT = yes
CFG_CM36558_SUPPORT = yes
CFG_MAGNETOMETER_SUPPORT = yes
```

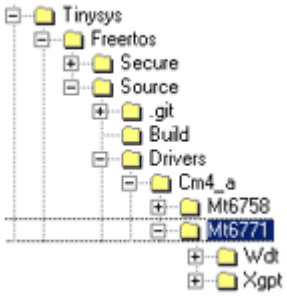
4.1.3 How to add freeRTOS driver under Tinysys

1 put the code in the appropriate folder

For driver code, put your file at following path:

drivers/$(PROCESSOR)/$(PLATFORM)/(New_Driver_Folder)

- Example: drivers/CM4_A/mt6771/XGPT



2. add a new compiler option in platform.mk

project/$(PROCESSOR)/$(PLATFORM)/platform/platform.mk

- example：project/mt6771/platform/platform.mk

```
ifeq ($(CFG_XGPT_SUPPORT),yes)
INCLUDES += $(DRIVERS_PLATFORM_DIR)/xgpt/inc/
C_FILES  += $(DRIVERS_PLATFORM_DIR)/xgpt/src/xgpt.c
C_FILES  += $(SOURCE_DIR)/kernel/service/common/src/utils.c
endif
```

3. add a new configuration in platform.mk or ProjectConfig.mk

- project/$(PROCESSOR)/$(PLATFORM)/platform/platform.mk
- project/$(PROCESSOR)/$(PLATFORM)/$(PROJECT)/ProjectConfig.mk
- example：project/mt6771/platform/platform.mk

```
################################################
# SCP internal feature options
################################################
CFG_TESTSUITE_SUPPORT = no
CFG_MODULE_INIT_SUPPORT = yes
CFG_XGPT_SUPPORT = yes
CFG_UART_SUPPORT = no
CFG_MTK_SCPUART_SUPPORT = yes
```

4.1.4 SCP code size limitation mechanism

1. memoryReport.py is a script which use to limit code size at the build time.

   If code size over your settings, it will cause build errors.

   (script: vendor/mediatek/proprietary/tinysys/freertos/source/tools/**memoryReport.py**)

2. This script is hooked by tinysys scp make file:

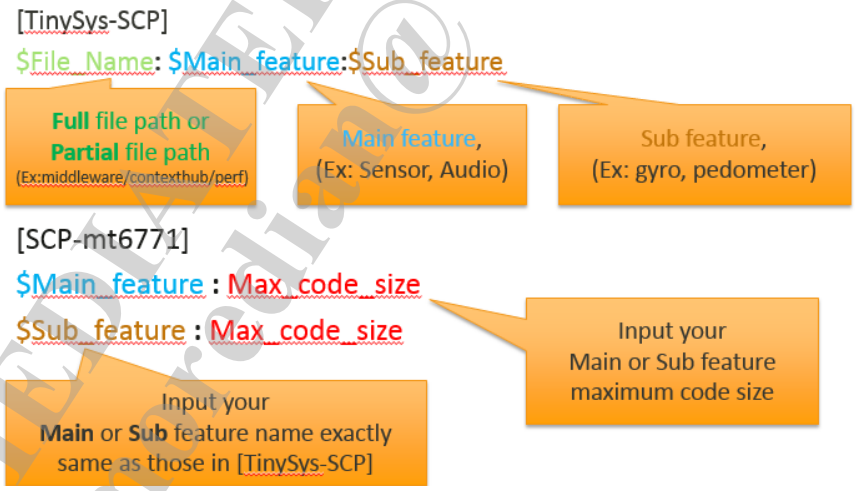
   vendor/mediatek/proprietary/tinysys/freertos/source/build/**config.mk**

```
-180,8 +180,8 @@ $($(PROCESSOR).ELF_FILE): $(MY_LIBFLAGS_SEARCH_FILE)
$(PROCESSOR).ELF_FILE): $(DEPS)
    @echo '$(TINYSYS_SCP): ELF      $@'
    $(hide)$(CC) $(PRIVATE_LDFLAGS) $(PRIVATE_OBJS) -Wl,-Map=$(PRIVATE_MAP_FILE) -o
    @echo '$(TINYSYS_SCP): Memory Check'
    $(hide)PLATFORM=$(PLATFORM) $(MCHECK) SCP $(SETTING) $(PRIVATE_MAP_FILE)
```

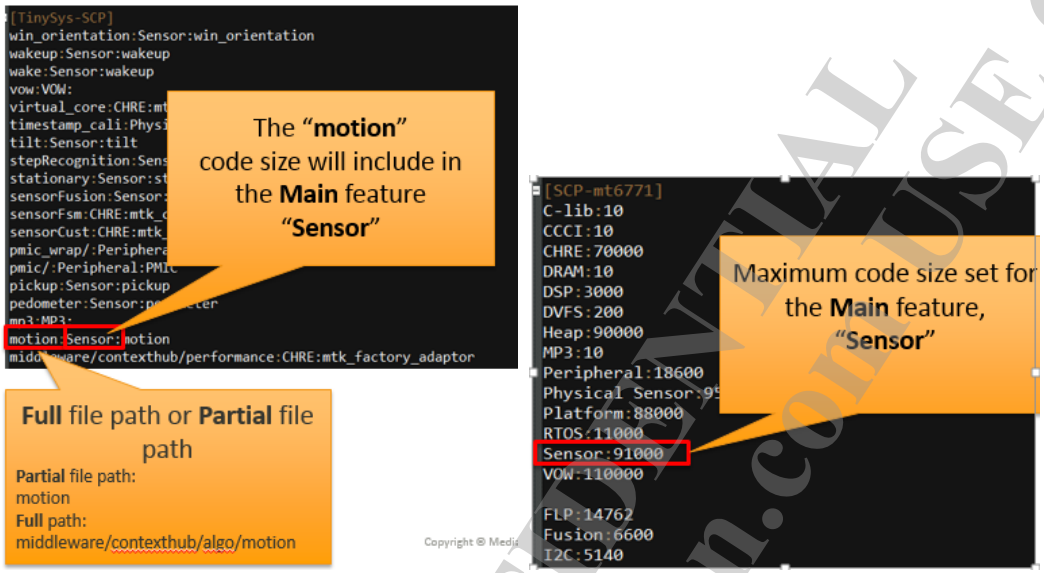
3. Configuration file at following path :

   vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/**$PLATFORM**/platform/**Setting.ini**

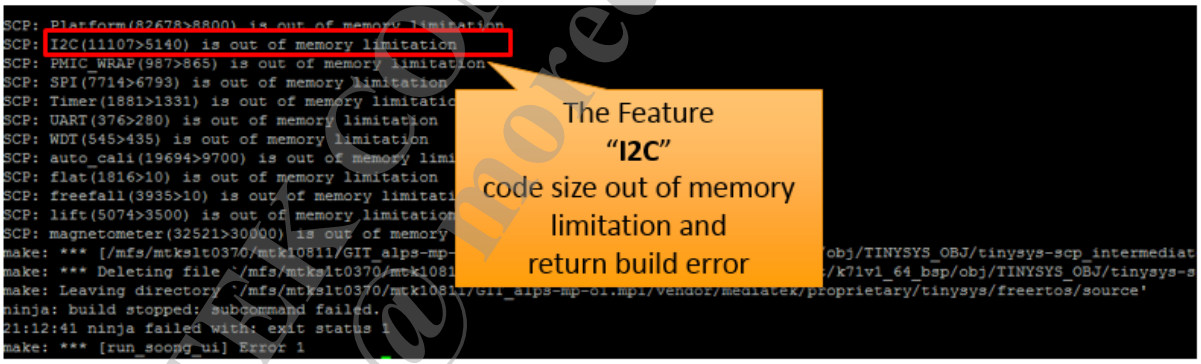
4. **Setting.ini format**



Example：



If code size exceeds limit, there will be build error warning as following：



5 CHRE sensors introduction

5.1 CHRE Introduction

Under SCP， MTK sensor hub feature is developed on Google CHRE architecture.

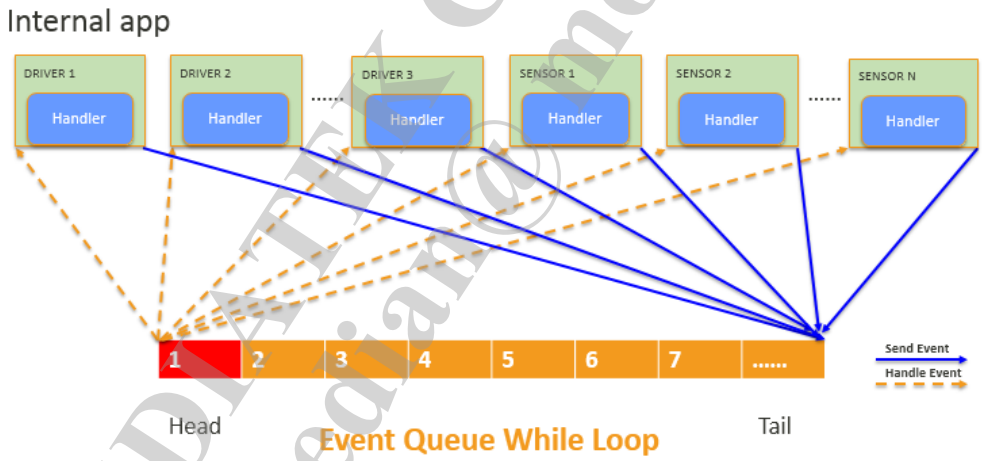
CHRE (Context Hub Runtime Environment) is an event-driven architecture， and can also be treated as an OS。

The yellow part is the Event Queue. CHRE only has a while loop to handle the head event in the event queue. One task in the queue cannot be invoked by the CHRE if a previous invocation hasn't completed. So there's no concept of priority and the current event queue processing can only be interrupted by interrupts. CHRE supports maximum 512 events in Event Queue in default.

The purpose of CHRE is to be real-time and lightweight, so all tasks invoked in Event Queue must run quickly.

The driver implementation in CHRE is called nano hub app. The following section will explain how to implement a nano hub app in detail.

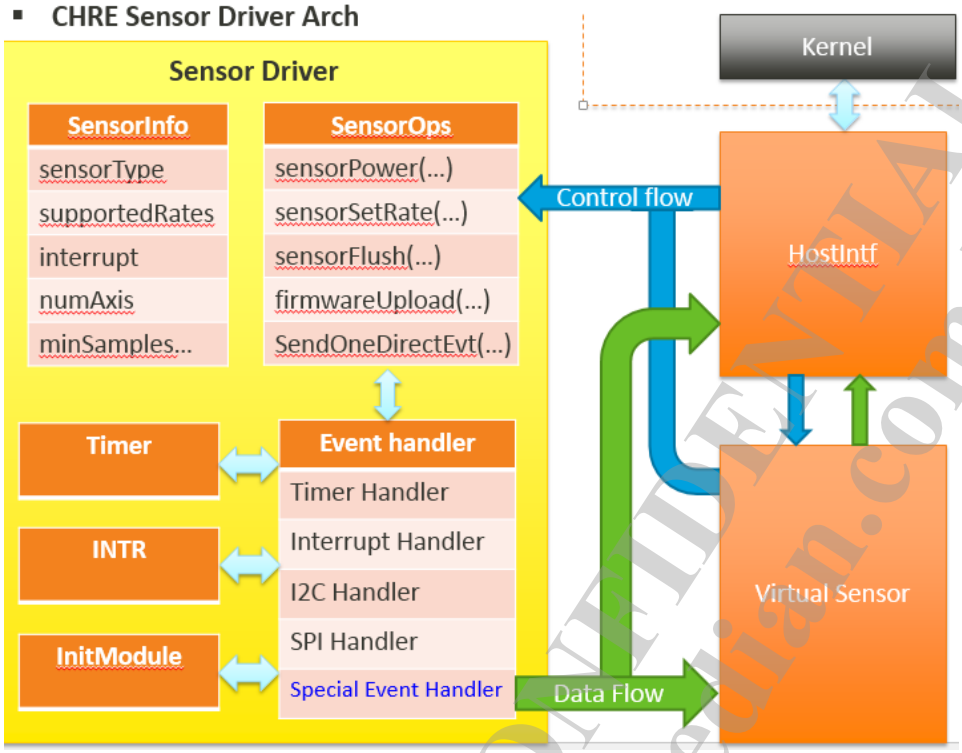
CHRE message mechanism is briefly shown in the figure.



5.2 MTK CHRE Sensors Common Layer

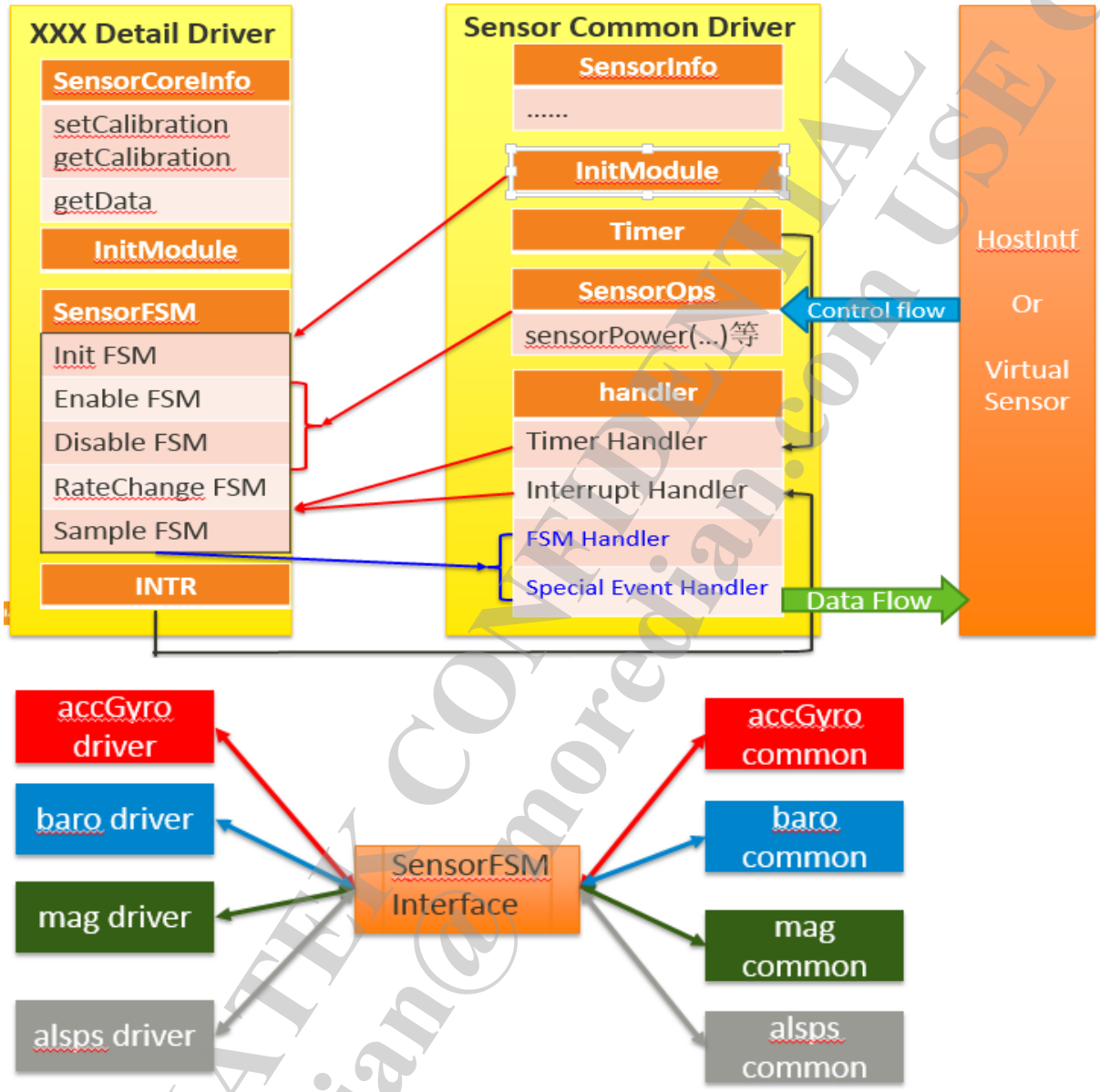
Because of the event-driven mechanism in CHRE, it is quite complex to implement an app. The native CHRE app architecture is as following:

- CHRE Sensor Driver Arch



App needs to implements hostintf (hostintf provide functions which are similar to SensorManager, SensorService and HAL layer interface as well in AP side) to perform control/data flow as shown in the figure. Programmers are required to be familiar with hostintf internal flow.

To avoid too much effort to do porting and bug hunting, MTK separates physical sensor logical part as a single layer called sensorFSM. Hardware related part is in other files and evoked by sensorFSM. The architecture is as following:



After seperating common layer , what customers need to implement are:

- Init flow (load customization, auto detect…)
- Implement finite sensor FSM
- Implement sensorCoreInfo

5.3 How to implement a CHRE APP

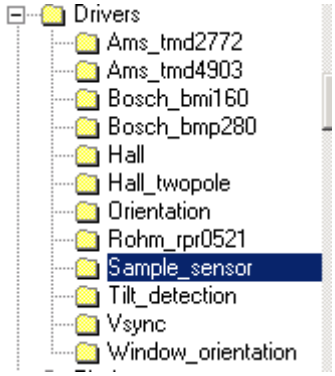
MTK virtual sensor is implemented following CHRE APP standardization. Customers could implement their own virtual sensors or other sensor related APP. The following part introduces an example to show how to implement a CHRE APP.

This example implements a custom sensor typle flat. This sensor type is used to check whether the phone is horizontal placement.

5.3.1 Copy a demo driver to implement CHRE APP architecture

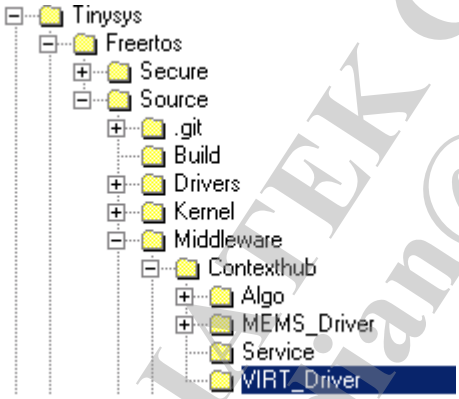
路径：

vendor\mediatek\proprietary\hardware\contexthub\firmware\src\drivers\sample_sensor\sample_sensor.cpp



Copy to the following path：

vendor\mediatek\proprietary\tinysis\freertos\souce\middleware\virt_driver\



5.3.2 Add compile options

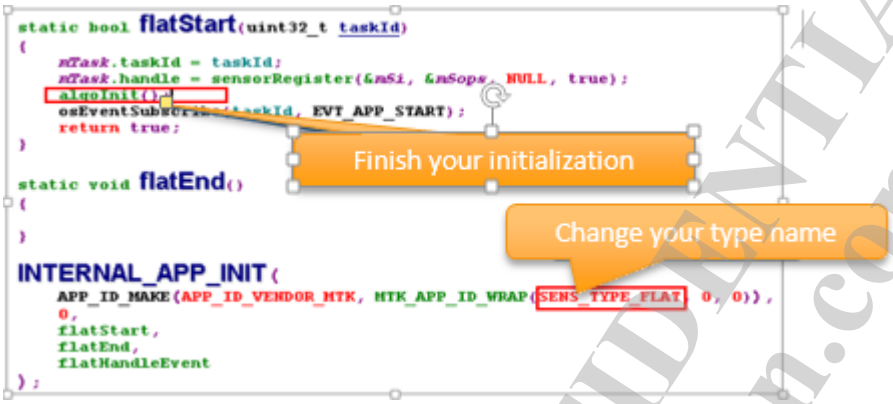
project/CM4_a/mt6771/platform/feature_config/chre.mk

```
######## flat ########
ifeq ($(CFG_FLAT_SUPPORT),yes)
INCLUDES += -I$(SOURCE_DIR)/middleware/contexthub/algo/common
C_FILES  += $(SOURCE_DIR)/middleware/contexthub/VIRT_Driver/flat.c
C_FILES  += $(SOURCE_DIR)/middleware/contexthub/VIRT_Driver/algoDataResample.c
LIBFLAGS += -L$(SOURCE_DIR)/middleware/contexthub/algo/common -lmath
endif
```

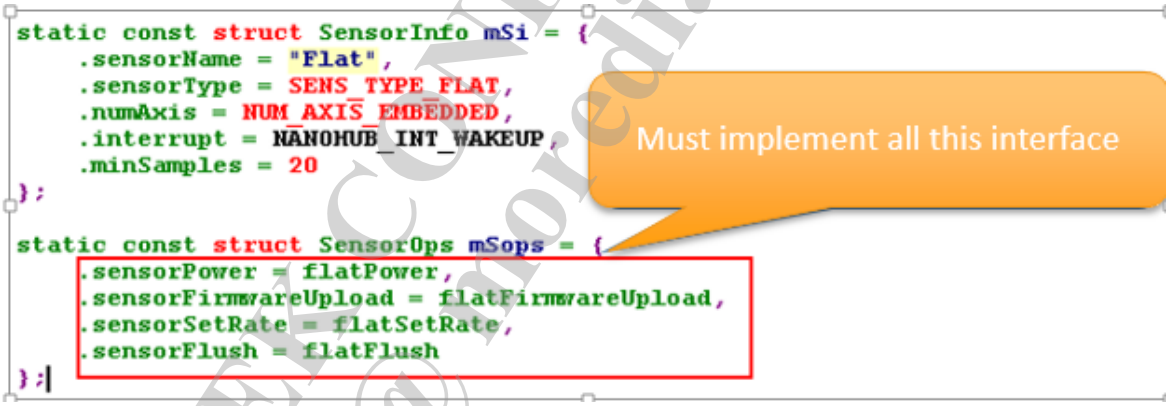
5.3.3 APP modification tips

Do the following modification:

1. Initialization, start the CHRE APP.



2. Implement a global structure in APP and register a sensor type

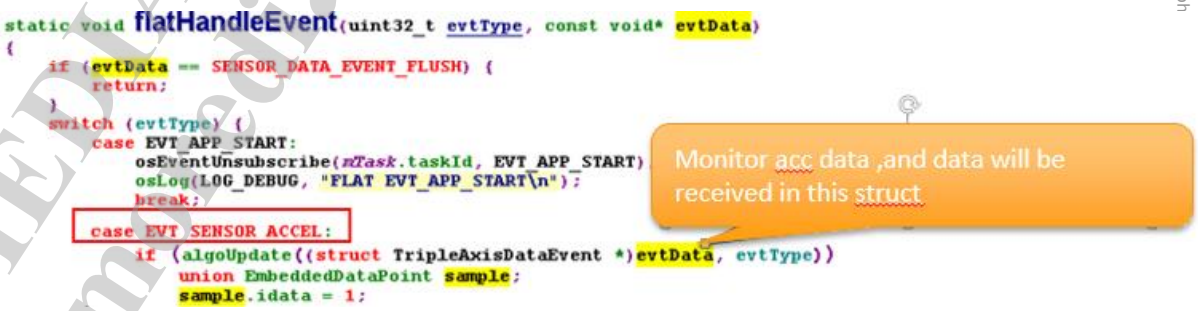


3. Subscribe events

   Flat needs to subscribe ACC raw date to implement its own algorithm.

```
osEventSubscribe(mTask.taskId, EVT_SENSOR_ACCEL);
```

4. Handle subscribed events



5. CHRE sensor data structure introduction：

   Path：vendor\mediatek\proprietary\hardware\contexthub\firmware\inc\sensors.h

```
struct RawTripleAxisDataPoin
    union {
        uint32_t deltaTime; // delta since last sample, for 0th sample this is firstSample
        struct SensorFirstSample firstSample;
    };
    int16_t ix;
    int16_t iy;
    int16_t iz;
} ATTRIBUTE_PACKED;
SET_PACKED_STRUCT_MODE_OFF

struct RawTripleAxisDataEvent {
    uint64_t referenceTime;
    struct RawTripleAxisDataPoint samples[];
};
```

Calc the delta between two sensor data

Record this batch include how many data

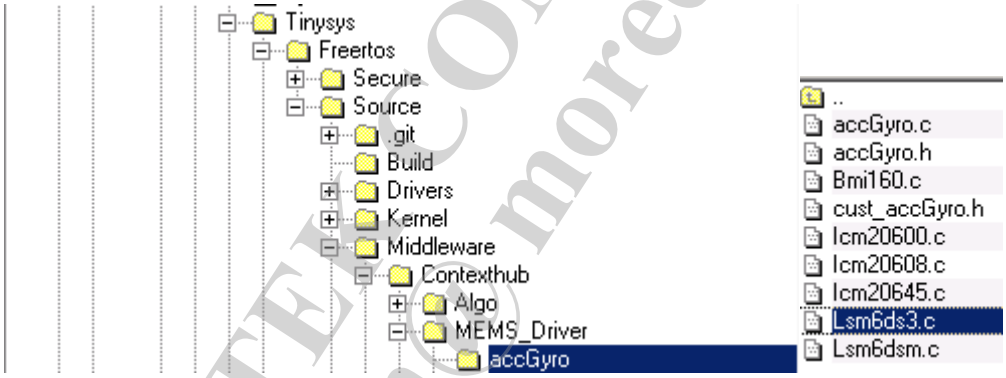
The timestamp when fifo interrupt up

5.4 A+G driver porting guide

5.4.1 A+G initialization

Here a ST driver is employed as an example:

Copy an implemented driver to do modification. Please copy from the same series to save effort to do modification.

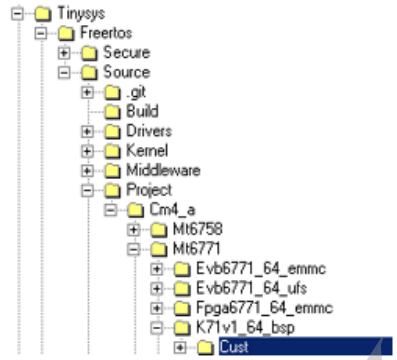


```
int lsm6ds3Init(void)
{
    int ret = 0;
    enum SensorIndex i;

    insertMagicNum(&mTask.accGyroPacket);
    mTask.hw = get_cust_accGyro("lsm6ds3");

    if (NULL == mTask.hw) {
        osLog(LOG_ERROR, "get_cust_acc_hw fail\n");
        return 0;
    }

    osLog(LOG_INFO, "acc spi_num: %d\n", mTask.hw->i2c_num);

    if (0 != (ret = sensorDriverGetConvert(mTask.hw->direction, &mTask.cvt))) {
        osLog(LOG_ERROR, "invalid direction: %d\n", mTask.hw->direction);
    }
    osLog(LOG_ERROR, "acc map[0]:%d, map[1]:%d, map[2]:%d, sign[0]:%d, sign[1]:%d, sign[2]:%d\n\r",
        mTask.cvt.map[AXIS_X], mTask.cvt.map[AXIS_Y], mTask.cvt.map[AXIS_Z],
        mTask.cvt.sign[AXIS_X], mTask.cvt.sign[AXIS_Y], mTask.cvt.sign[AXIS_Z]);

    mTask.sensors[ACC].sensitivity = 65536 / (8 * 2);
    mTask.sensors[GYR].sensitivity = 1000 / 70;
```

Get customize info

Change sensitivity

Custom info location:




```
#include "cust_accGyro.h"

struct accGyro_hw cust_accGyro_hw[] __attribute__((section(".cust_accGyro"))) = {
#ifdef CFG_LSM6DSM_SUPPORT
    {
        .name = "lsm6dsm",
        .i2c_num = 0,
        .direction = 7,
        .i2c_addr = {0, 0},
        .eint_num = 10,
    },
#endif
};
```

Configure customize info here

If use SPI，we still need to add SPI bus info to i2c_num section, we reuse this item

1. SensorFSM array

Definition： head-tail event

Head event，such as STAT_ENABLE，STAT_SAMPLE

Tail event，the last event to execute after finishing a specific action，such as ENABLE_DONE，DISABLE_DONE，RATECHG_DONE, SAMPLE_DONE and so on. These events usually come along with interaction with top layer.

Events betewen head and tail, it defined by demand according to specific hw spec. Ususally, one i2c/SPI transfer is corresponded to one state.

To implement a new driver , customer just need to finish the FMS array

This is a typical example:

```
static struct sensorFsm lsm6dsmFsm[] = {
    sensorFsmCmd(STATE_SW_RESET, STATE_INIT_REG, lsm6dsmSwReset),
    sensorFsmCmd(STATE_INIT_REG, STATE_SENSOR_REGISTRATION, lsm6dsmInitReg),
    sensorFsmCmd(STATE_SENSOR_REGISTRATION, STATE_EINT_REGISTRATION, lsm6dsmSensorRegistration),
    sensorFsmCmd(STATE_EINT_REGISTRATION, STATE_INIT_DONE, lsm6dsmEintRegistration),

    sensorFsmCmd(STATE_ACC_ENABLE, STATE_ACC_ENABLE_DONE, lsm6dsmAccPowerOn),
    sensorFsmCmd(STATE_ACC_DISABLE, STATE_ACC_DISABLE_DONE, lsm6dsmAccPowerOff),

    sensorFsmCmd(STATE_ACC_RATECHG, STATE_ACC_RATECHG_DONE, lsm6dsmAccRate),

    sensorFsmCmd(STATE_GYRO_ENABLE, STATE_GYRO_ENABLE_DONE, lsm6dsmGyroPowerOn),
    sensorFsmCmd(STATE_GYRO_DISABLE, STATE_GYRO_DISABLE_DONE, lsm6dsmGyroPowerOff),

    sensorFsmCmd(STATE_GYRO_RATECHG, STATE_GYRO_RATECHG, lsm6dsmGyroRate),

    sensorFsmCmd(STATE_HW_INT_STATUS_CHECK, STATE_HW_INT_HANDLING, lsm6dsmIntStatusCheck),
    sensorFsmCmd(STATE_HW_INT_HANDLING, STATE_HW_INT_HANDLING_DONE, lsm6dsmIntHandling),

    sensorFsmCmd(STATE_SAMPLE, STATE_FIFO, lsm6dsmSample),
    sensorFsmCmd(STATE_FIFO, STATE_CONVERT, lsm6dsmReadFifo),
    sensorFsmCmd(STATE_CONVERT, STATE_SAMPLE_DONE, lsm6dsmConvert),

    /* For Anymotion */
    sensorFsmCmd(STATE_ANYMO_ENABLE, STATE_ANYMO_ENABLE_DONE, anyMotionPowerOn),
    sensorFsmCmd(STATE_ANYMO_DISABLE, STATE_ANYMO_DISABLE_DONE, anyMotionPowerOff),
};
```

initialization

Enable disable

Rate change

Handle interrupt

Sample data

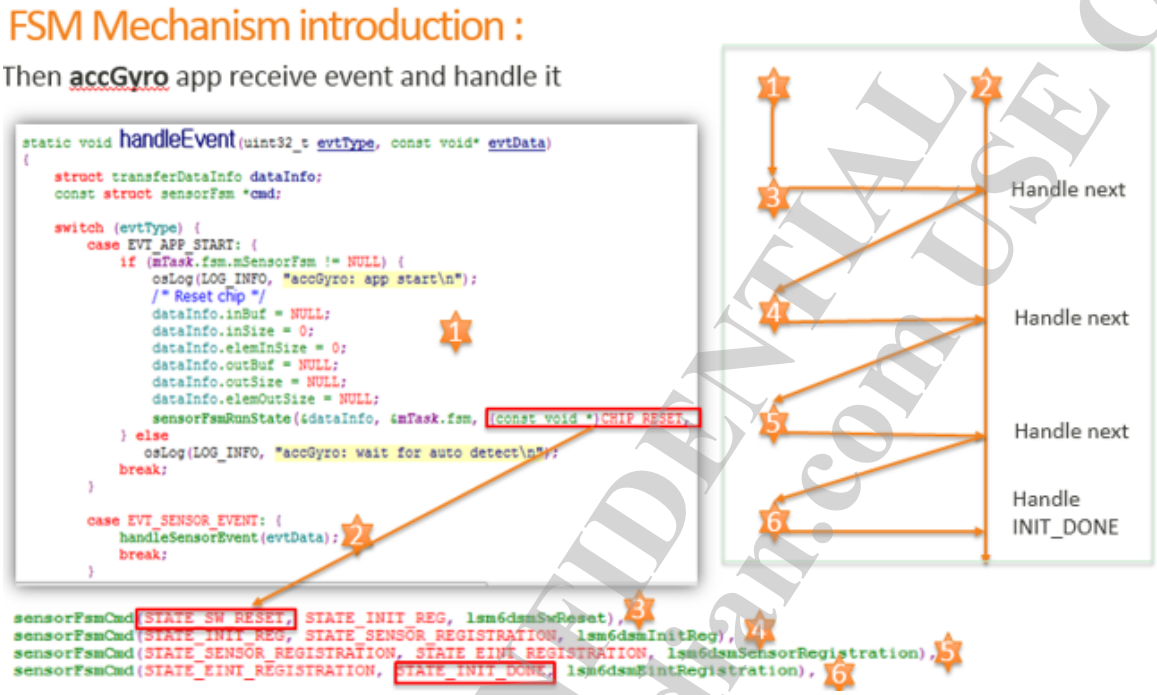
```
enum LSM6DSMState {
    STATE_SAMPLE = CHIP_SAMPLING,
    STATE_FIFO = CHIP_FIFO,
    STATE_CONVERT = CHIP_CONVERT,
    STATE_SAMPLE_DONE = CHIP_SAMPLING_DONE,
    STATE_ACC_ENABLE = CHIP_ACC_ENABLE,
    STATE_ACC_ENABLE_DONE = CHIP_ACC_ENABLE_DONE,
    STATE_ACC_DISABLE = CHIP_ACC_DISABLE,
    STATE_ACC_DISABLE_DONE = CHIP_ACC_DISABLE_DONE,
    STATE_ACC_RATECHG = CHIP_ACC_RATECHG,
    STATE_ACC_RATECHG_DONE = CHIP_ACC_RATECHG_DONE,
    STATE_GYRO_ENABLE = CHIP_GYRO_ENABLE,
    STATE_GYRO_ENABLE_DONE = CHIP_GYRO_ENABLE_DONE,
    STATE_GYRO_DISABLE = CHIP_GYRO_DISABLE,
    STATE_GYRO_DISABLE_DONE = CHIP_GYRO_DISABLE_DONE,
    STATE_GYRO_RATECHG = CHIP_GYRO_RATECHG,
    STATE_GYRO_RATECHG_DONE = CHIP_GYRO_RATECHG_DONE,

    STATE_ANYMO_ENABLE = CHIP_ANYMO_ENABLE,
    STATE_ANYMO_ENABLE_DONE = CHIP_ANYMO_ENABLE_DONE,
    STATE_ANYMO_DISABLE = CHIP_ANYMO_DISABLE,
    STATE_ANYMO_DISABLE_DONE = CHIP_ANYMO_DISABLE_DONE,

    STATE_HW_INT_STATUS_CHECK = CHIP_HW_INT_STATUS_CHECK,
    STATE_HW_INT_HANDLING = CHIP_HW_INT_HANDLING,
    STATE_HW_INT_HANDLING_DONE = CHIP_HW_INT_HANDLING_DONE,

    STATE_INIT_DONE = CHIP_INIT_DONE,
    STATE_IDLE = CHIP_IDLE,
    STATE_SW_RESET = CHIP_RESET,
    STATE_INIT_REG,
    STATE_SENSOR_REGISTRATION,
    STATE_EINT_REGISTRATION,
};
```

Note: In the driver, head-tail MUST be defined first.

Head-tail events are already defined in the common header file which can be used directly. But these must be put in front.

Your own definitions at the end

The following is the introduction of an init flow.



5.4.2 Enable/Disable

```
sensorFsmCmd(STATE_ACC_ENABLE, STATE_ACC_ENABLE_DONE, lsm6dsmAccPowerOn),
sensorFsmCmd(STATE_ACC_DISABLE, STATE_ACC_DISABLE_DONE, lsm6dsmAccPowerOff),

sensorFsmCmd(STATE_GYRO_ENABLE, STATE_GYRO_ENABLE_DONE, lsm6dsmGyroPowerOn),
sensorFsmCmd(STATE_GYRO_DISABLE, STATE_GYRO_DISABLE_DONE, lsm6dsmGyroPowerOff),
```

Note: if there's FIFO open/close operation, the following function should be called immediately after FIFO opened. This function will calibrate FIFO data timestamps.

```
registerAccGyroFifoInfo((mTask.sensors[ACC].hwRate == 0) ? 0 : 1024000000000 / mTask.sensors[ACC].hwRate,
    (mTask.sensors[GYR].hwRate == 0) ? 0 : 1024000000000 / mTask.sensors[GYR].hwRate);
```

Implement the function according to the vendor's data sheet.

5.4.3 Report rate

In A+G two in one driver, rate setting needs to pay attention. If both Acc and Gyro are enabled, the rates should be kept in consistent. The reason could be referred to FIFO setting section.

```
static int lsm6dsmAccRate(I2cCallbackF i2cCallBack, SpiCbkF spiCallBack, void *next state,
                          void *inBuf, uint8_t inSize, uint8_t elemInSize,
                          void *outBuf, uint8_t *outSize, uint8_t *elemOutSize)
{
```

```
odr = lsm6dsmCalcuOdr(&mTask.sensors[ACC].rate, &sampleRate);
if (odr < 0) {
    sensorFsmEnqueueFakeSpiEvt(spiCallBack, next_state, ERROR_EVT);
    osLog(LOG_ERROR, "lsm6dsmAccRate, calcu odr error\n");
    return -1;
}

if (odr < 2)
    sampleRate = SENSOR_HZ(26.0f / 2.0f);
mTask.sensors[ACC].preRealRate = sampleRate;
```

Convert the input rate to a closest hardware-supported report rate.

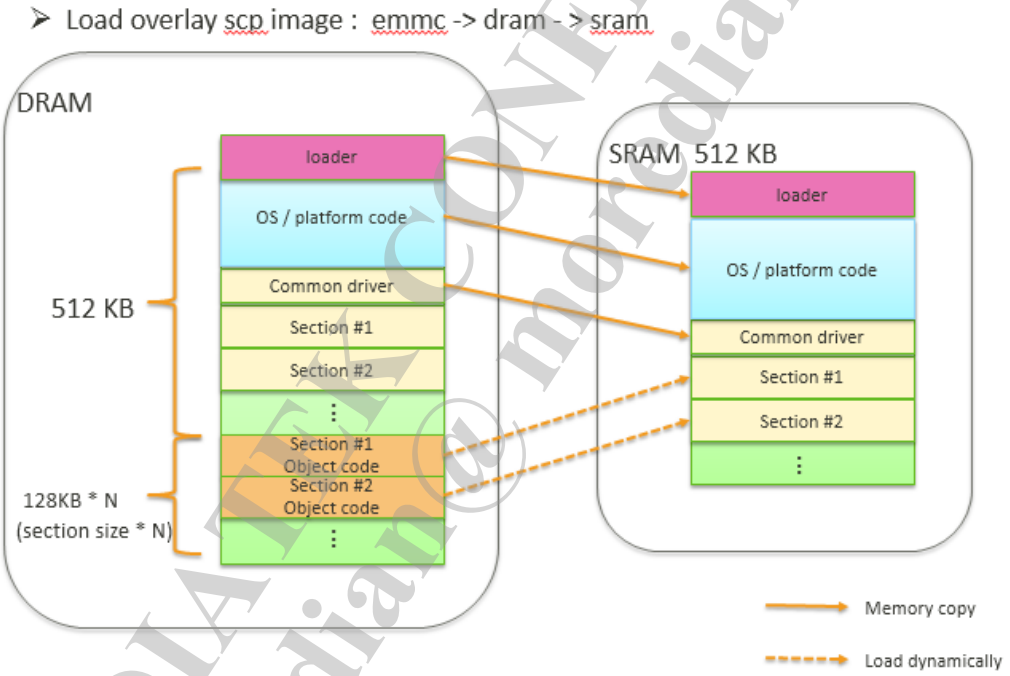
Record the report rate temporarily and the final rate still needs to be calculated. Ex. is the acc is already

```
if (mTask.sensors[GYR].configed) {
    maxRate = max(sampleRate, mTask.sensors[GYR].preRealRate);  //choose with preRealRate
    if ((maxRate != mTask.sensors[ACC].hwRate) || (maxRate != mTask.sensors[GYR].hwRate)) {
        mTask.sensors[ACC].hwRate = maxRate;
        mTask.sensors[GYR].hwRate = maxRate;

        odr = lsm6dsmCalcuOdr(&maxRate, &sampleRate);
        if (odr < 0) {
            sensorFsmEnqueueFakeSpiEvt(spiCallBack, next_state, ERROR_EVT);
            osLog(LOG_ERROR, "lsm6dsmAccRate, calcu odr error\n");
            return -1;
        }

        regValue = LSM6DSMImuRatesRegValue[odr];

        //delay = LSM6DSMGyroRatesSamplesToDiscard[odr] * (1024000000 / maxRate);
        mTask.sensors[ACC].samplesToDiscard = LSM6DSMAccelRatesSamplesToDiscard[odr];
        mTask.sensors[GYR].samplesToDiscard = LSM6DSMGyroRatesSamplesToDiscard[odr];

        SPI_WRITE(LSM6DSM_CTRL1_XL_ADDR, LSM6DSM_CTRL1_XL_BASE | regValue, 30);
        SPI_WRITE(LSM6DSM_CTRL2_G_ADDR, LSM6DSM_CTRL2_G_BASE | regValue, 30);
        accelOdrChanged = true;
    } ? end if (maxRate! =mTask.senso... ?  else {
        accelOdrChanged = false;
    }
} ? end if mTask.sensors[GYR].co... ?  else {
    if ((sampleRate != mTask.sensors[ACC].hwRate)) {
        mTask.sensors[ACC].hwRate = sampleRate;
        regValue = LSM6DSMImuRatesRegValue[odr];

        //delay = LSM6DSMAccelRatesSamplesToDiscard[odr] * (1024000000 / maxRate);
        mTask.sensors[ACC].samplesToDiscard = LSM6DSMAccelRatesSamplesToDiscard[odr];

        SPI_WRITE(LSM6DSM_CTRL1_XL_ADDR, LSM6DSM_CTRL1_XL_BASE | regValue, 30);
        accelOdrChanged = true;
    } else {
        accelOdrChanged = false;
    }
}
```

HW ODR needs to be reset if ACC retae is not

5.5 Sensor driver overlay

Purpose: customers need two materials. One type sensor may come from two different vendor. If putting these two drivers in SCP SRAM to do auto detect, it consumes SRAM. So MTK solution locates two drivers in the DRAM and one driver is loaded when SCP boots.

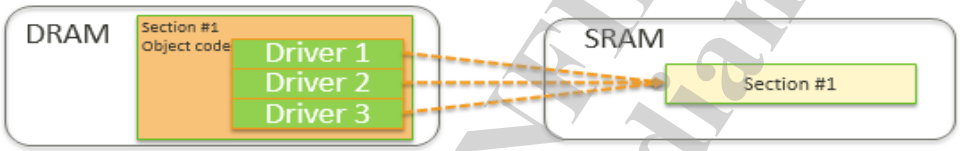


Overlay load flow

1) Memory copy loader code from dram to SRAM, then SCP run loader

2) SCP loader copy Tinysys common code from dram to SRAM, os run and sensor driver init

3) OverlayRemap copy sensor driver 1 to sram section and hw verify , if fail ,copy driver 2 and verify , if success , remap next sensor type



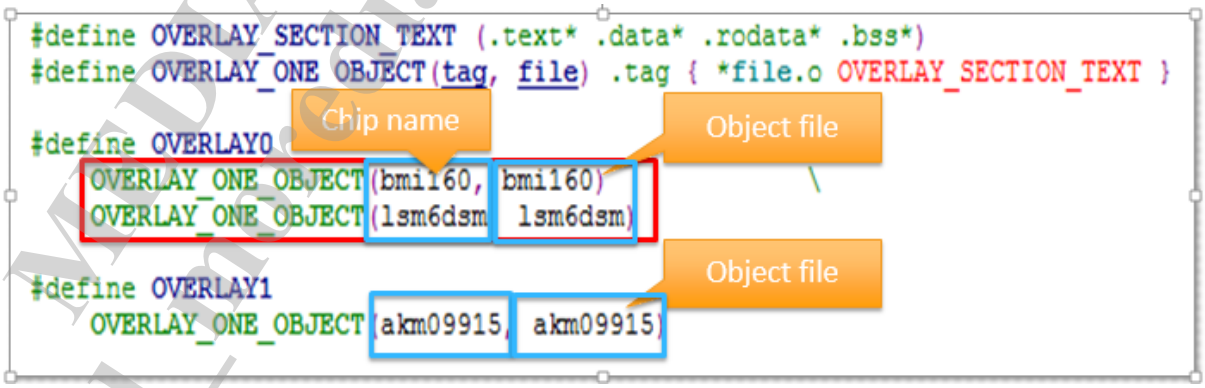
One section represent one sensor type ,may have multiple drivers (object code)

5.5.1 How to add overlay driver

1) ADD object in linker script

vendor\mediatek\proprietary\tinysys\freertos\source\project\CM4_A\#PLATFORM\$PROJECT\inc\overlay_sensor.h

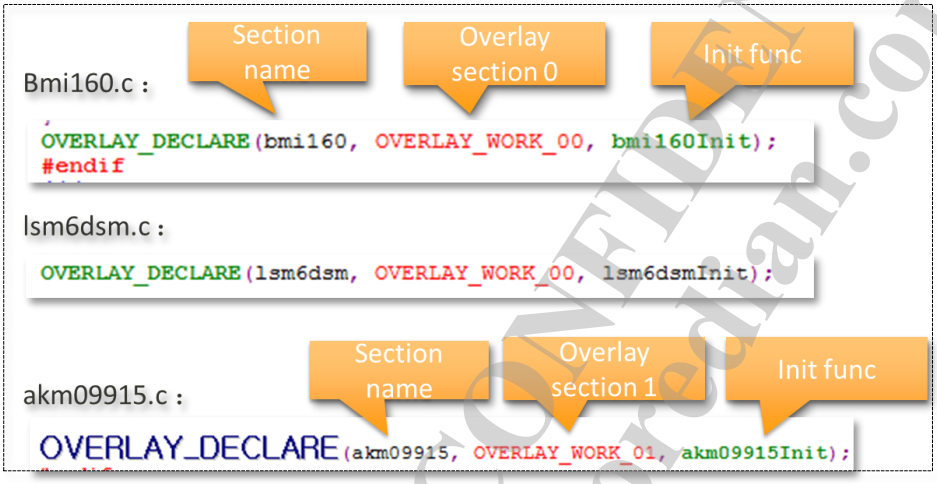
For A+G sensor type , may have bmi160 chip and lsm6dsm chip, add driver object file in overlay scp image, and for mag sensor type ,may have akm09915 chip.



For mag sensor type , there are several library file

```
#define OVERLAY_FIVE_OBJECT(tag, file1, file2, file3, file4, file5) \
    .tag { *file1.o OVERLAY_SECTION_TEXT } \
    .tag { *file2.o OVERLAY_SECTION_TEXT } \
    .tag { *file3.o OVERLAY_SECTION_TEXT } \
    .tag { *file4.o OVERLAY_SECTION_TEXT } \
    .tag { *file5.o OVERLAY_SECTION_TEXT }

#define OVERLAY1                                        \
    OVERLAY_FIVE_OBJECT(akm09915, akm09915, AkmApi, ParameterIO, Measure, Libakm09912)
```

2) ADD overlay declare in your overlay object (This is added in the specific driver)



3) In the driver, synchronized spi or i2c API is used in senor init flow.

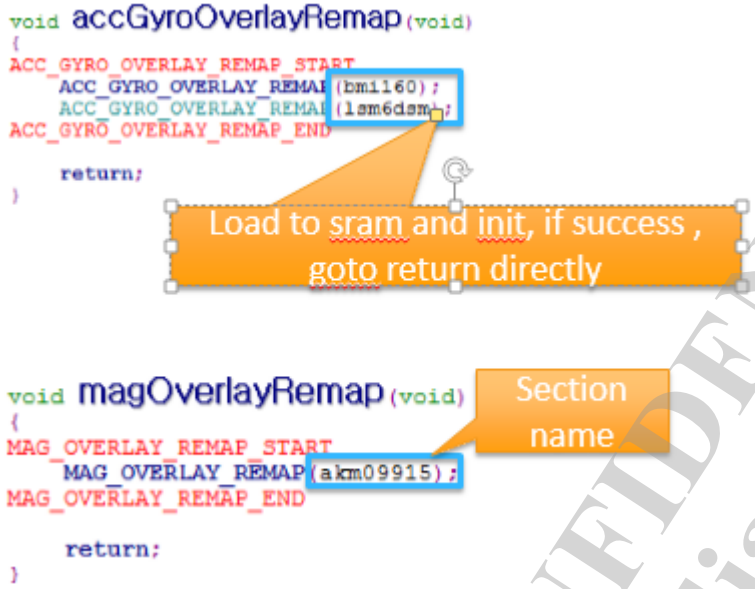
vendor\mediatek\proprietary\tinysys\freertos\source\middleware\contexthub\MEMS_Driver\

```
static int bmi160Init(void)
{
    // read the device ID for bmi160
    txData[0] = BMI160_REG_ID | 0x80;
    ret = spiMasterRxTxSync(T(spiDev), rxData, txData, 2);

    if (ret < 0 || (rxData[1] != BMI160_ID)) {
        ERROR_PRINT("failed id match: %02x, ret: %d\n", rxData[1], ret);
        spiMasterRelease(T(spiDev));
        goto err_out;
    }
    osLog(LOG_ERROR, "success id match: %02x\n", rxData[1]);
    SET_STATE(SENSOR_INITIALIZING);
    mTask.init_state = RESET_BMI160;
    registerAccGyroInterruptMode(ACC_GYRO_FIFO_INTERRUPTIBLE);
    registerAccGyroDriverFsm(bmi160Fsm, ARRAY_SIZE(bmi160Fsm));
err_out:
    return ret;
}
```

使用同步 API

4) ADD overlay remap for load and init in overlay.c

vendor\mediatek\proprietary\tinysys\freertos\source\project\CM4_A\$PLATFORM\$PROJECT \cust\overlay\



5) Enable overlay feature

vendor\mediatek\proprietary\tinysys\freertos\source\project\CM4_A\$PLATFORM\$PROJECT \Projectconfig.mk

```
CFG_OVERLAY_INIT_SUPPORT = yes
CFG_OVERLAY_DEBUG_SUPPORT = yes
```

5.6 CHRE I2C & SPI API

5.6.1 I2C API

1） Standard API

request i2c. Before Calingl I2C transfer API ，call this API firstly. This API is usually only used in user init function.

```
int i2cMasterRequest(uint32_t busId, uint32_t speedInHz);
```

release i2c

```
int i2cMasterRelease(uint32_t busId);
```

I2C Write

```
static inline int i2cMasterTx(uint32_t busId, uint32_t addr,
        const void *txBuf, size_t txSize, I2cCallbackF callback, void *cookie)
{
```

I2C Read

```
static inline int i2cMasterRx(uint32_t busId, uint32_t addr,
        void *rxBuf, size_t rxSize, I2cCallbackF callback, void *cookie)
```

I2C write and read

```
int i2cMasterTxRx(uint32_t busId, uint32_t addr, const void *txBuf, size_t txSize,
        void *rxBuf, size_t rxSize, I2cCallbackF callback, void *cookie);
```

2） MTK custom serial API

Int is used for sensor overlay

5.6.2 SPI API

Initialization

```
struct BMI160Task {
    uint32_t tid;
    struct BMI160Sensor sensors[NUM_OF_SENSOR];
    struct BMI160Sensor sensors_handle[NUM_OF_HANDLE];

    // time keeping.
    uint64_t last_sensortime;
    uint64_t frame_sensortime;
    uint64_t prev_frame_time[3];
    uint64_t time_delta[3];
    uint64_t next_delta[3];
    uint64_t tempTime;

    // spi and interrupt
    spi_cs_t cs;
    struct SpiMode mode;
    struct SpiPacket packets[SPI_PACKET_SIZE];
    struct SpiDevice *spiDev;
    time_sync_t gSensorTime2RTC;
```

定义 SPI 用的结构体类型

```
T(mode).speed = 8000000;    //8Mhz
T(mode).bitsPerWord = 8;
T(mode).cpol = SPI_CPOL_IDLE_LO;
T(mode).cpha = SPI_CPHA_LEADING_EDGE;
T(mode).nssChange = true;
T(mode).format = SPI_FORMAT_MSB_FIRST;
```

设置 mode，CS，速率

```
spiMasterRequest(T(hw)->i2c_num, &T(spiDev));
```

申请 SPI 使用权限

Synchronized SPI AP（only used in sensor initialization）

```
// read the device ID for bmi160
txData[0] = BMI160_REG_ID | 0x80;
ret = spiMasterRxTxSync(T(spiDev), rxData, txData, 2);

if (ret < 0 || (rxData[1] != BMI160_ID)) {
    ERROR_PRINT("failed id match: %02x, ret: %d\n", rxData[1], ret);
    ret = -1;
    spiMasterRelease(T(spiDev));
    goto err_out;
}
osLog(LOG_ERROR, "success id match: %02x\n", rxData[1]);
```

CHRE API

```
// perform soft reset and wait for 100ms
SPI_WRITE(BMI160_REG_CMD, 0xb6, 100000);
// dummy reads after soft reset, wait 100us
SPI_READ(BMI160_REG_MAGIC, 1, &mTask.dataBuffer, 100);
```

```
spiBatchTxRx(&mTask.mode, sensorSpiCallback, &mTask, "sensorInit RESET" );
```

6 Build and Debug

6.1 Source Code Structure & File Description

Kernel code

- alps/kernel-3.18/drivers/misc/mediatek/sensors-1.0/
- alps/kernel-4.4/drivers/misc/mediatek/sensors-1.0/
- alps/device/mediatek/common/kernel-headers/linux/hwmsensor.h

HAL code

- alps/vendor/mediatek/proprietary/hardware/sensor
- alps/vendor/mediatek/proprietary/hardware/libsensor (第三方算法库)
- alps/device/mediatek/MTxxxx/device.mk
- alps/device/mediatek/MTxxxx/manifest.xml
- alps/device/mediatek/MTxxxx/init.sensors_1_0.rc

6.2 How to build SCP

Build with Android environment

Before building with Android environment, initialization is required (except a standalone build without Android):

1. *$. build/envsetup.sh*
2. *$ lunch full_<PROJECT>-eng*

Step #1 is needed only once.

If you wish to change your project, re-run step #2 and replace <PROJECT> accordingly.

Full Android Build

$ mosesq make -j24

Module Build

*$ mosesq make **tinysys-scp** -j24*

Faster Module Build

$ vendor/mediatek/proprietary/tinysys/freertos/source/tools/build_tinysys.sh -j24

Built binary: **out/target/product/<PROJECT>/obj/TINYSYS_OBJ/tinysys-scp_intermediates/freertos/source/tinysys-scp.bin**
Check the built time of the binary if you wanna make sure the binary is updated:

6.3 How to build a sensor driver to binary

6.3.1 AccGyro

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/platform/feature_config/chre.mk

```
ifeq ($(CFG_ACCGYRO_SUPPORT),yes)
INCLUDES += -I$(SENDRV_DIR)/accGyro/
INCLUDES += -I$(SOURCE_DIR)/middleware/contexthub/algo/auto_cali
INCLUDES += -I$(SOURCE_DIR)/middleware/contexthub/algo/timestamp_cali
C_FILES  += $(SENDRV_DIR)/accGyro/accGyro.c
C_FILES  += $(SENCUST_DIR)/accGyro/cust_accGyro.c
LIBFLAGS += -L$(SOURCE_DIR)/middleware/contexthub/algo/auto_cali -lksensor
LIBFLAGS += -L$(SOURCE_DIR)/middleware/contexthub/algo/timestamp_cali -lktimestamp
ifeq ($(CFG_BMI160_SUPPORT),yes)
C_FILES  += $(SENDRV_DIR)/accGyro/bmi160.c
endif
endif
```

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/{PROJECT}/cust/accGyro/cust_accGyro.c

```
#include "cust_accGyro.h"

struct accGyro_hw cust_accGyro_hw[] __attribute__((section(".cust_accGyro"))) = {
#ifdef CFG_BMI160_SUPPORT
    {
        .name = "bmi160",
        .i2c_num = 1,
        .direction = 4,
        .i2c_addr = {0x68, 0},
        .eint_num = 7,
    },
#endif
};
```

6.3.2 Barometer

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/platform/feature_config/chre.mk

```
ifeq ($(CFG_BAROMETER_SUPPORT),yes)
INCLUDES += -I$(SENDRV_DIR)/barometer
C_FILES  += $(SENDRV_DIR)/barometer/barometer.c
C_FILES  += $(SENCUST_DIR)/barometer/cust_baro.c
ifeq ($(CFG_BMP280_SUPPORT),yes)
C_FILES  += $(SENDRV_DIR)/barometer/bosch_bmp280.c
endif
endif
```

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/{PROJECT}/cust/barometer/cust_baro.c

```
struct baro_hw cust_baro_hw[] __attribute__((section(".cust_baro"))) = {
#ifdef CFG_BMP280_SUPPORT
    {
        .name = "bmp280",
        .i2c_num = 1,
        .direction = 0,
        .i2c_addr = {0x77, 0},
    },
#endif
};
```

6.3.3 Magnetometer

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/platform/feature_config/chre.mk

```
ifeq ($(CFG_MAGNETOMETER_SUPPORT),yes)
INCLUDES += -I$(SENDRV_DIR)/magnetometer
C_FILES  += $(SENDRV_DIR)/magnetometer/magnetometer.c
C_FILES  += $(SENCUST_DIR)/magnetometer/cust_mag.c
ifeq ($(CFG_AKM09915_SUPPORT),yes)
C_FILES  += $(SENDRV_DIR)/magnetometer/akm09915.c
INCLUDES += -I$(SENLIB_DIR)/akm09912/
INCLUDES += -I$(SENLIB_DIR)/akm09912/include/
C_FILES  += $(SENLIB_DIR)/akm09912/AkmApi.c
C_FILES  += $(SENLIB_DIR)/akm09912/ParameterIO.c
C_FILES  += $(SENLIB_DIR)/akm09912/Measure.c
LIBFLAGS += -L$(SENLIB_DIR)/akm09912/include -lakm09912
```

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/{PROJECT}/cust/alsps/cust_mag.c

```
struct mag_hw cust_mag_hw[] __attribute__((section(".cust_mag"))) = {
#ifdef CFG_AKM09915_SUPPORT
    {
        .name = "akm09915",
        .i2c_num = 1,
        .direction = 4,
        .i2c_addr = {0x0c, 0},
    },
#endif
};
```

6.4 Build Option

6.4.1 Common build option

1. **Device config**

   Patch: /device/mediatekprojects/$project/ProjectConfig.mk

   MTK_TINYSYS_SCP_SUPPORT=yes

   MTK_SENSOR_SUPPORT =yes

   CUSTOM_KERNEL_SENSORHUB=yes

   MTK_SENSORS_1_0=yes

2. **Kernel config**

   Path:

   /kernel-4.4/arch/$TARGET_ARCH/configs/$project_defconfig

   /kernel-4.4/arch/$TARGET_ARCH/configs/$project_debug_defconfig

   CONFIG_MTK_TINYSYS_SCP_SUPPORT=y

   CONFIG_MTK_HWMON=y

   CONFIG_MTK_SENSOR_SUPPORT=y

   CONFIG_CUSTOM_KERNEL_SENSORHUB=y

   CONFIG_NANOHUB_MTK_IPI=y

   CONFIG_MTK_SENSORS_1_0=y

   CONFIG_NANOHUB=y

   CONFIG_IIO=y

3. **SCP config**

   Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt6758/工程名/projectconfig.mk

   CFG_CHRE_SUPPORT =yes

   CFG_CONTEXTHUB_FW_SUPPORT =yes

4. **LK config**

   Path: /vendor/mediatek/proprietary/bootable/bootloader/lk/project/$(project)

   MTK_TINYSYS_SCP_SUPPORT=no

   MTK_TINYSYS_SCP_SUPPORT=yes

6.4.2 Physical sensor build option

1. **Device config**

Patch: /device/mediatek/$project/ProjectConfig.mk

ALS/PS	CUSTOM_KERNEL_ALSPS=yes
ACCELEROMETER	CUSTOM_KERNEL_ACCELEROMETER=yes
MAGNETIC_FIELD	CUSTOM_KERNEL_MAGNETOMETER=yes
GYROSCOPE	CUSTOM_KERNEL_GYROSCOPE=yes
BAROMETER	CUSTOM_KERNEL_BAROMETER=yes

2. Kernel config

Path:

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_defconfig

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_debug_defconfig

ALS/PS	CONFIG_CUSTOM_KERNEL_ALSPS=y CONFIG_MTK_ALSPSHUB=y
ACCELEROMETER	CONFIG_CUSTOM_KERNEL_ACCELEROMETER=y CONFIG_MTK_ACCELHUB=y
MAGNETIC_FIELD	CONFIG_CUSTOM_KERNEL_MAGNETOMETER=y CONFIG_MTK_MAGHUB=y
GYROSCOPE	CONFIG_CUSTOM_KERNEL_GYROSCOPE=y CONFIG_MTK_GYROHUB=y
BAROMETER	CONFIG_CUSTOM_KERNEL_BAROMETER=y CONFIG_MTK_BAROHUB=y

3. SCP config

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt6758/工程名/projectconfig.mk

ALS/PS	CFG_ALSPS_SUPPORT =yes CFG_CM36558_SUPPORT =yes
ACCELEROMETER	CFG_ACCGYRO_SUPPORT =yes CFG_BMI160_SUPPORT =yes

MAGNETIC_FIELD	CFG_MAGNETOMETER_SUPPORT =yes CFG_AKM09915_SUPPORT=yes
GYROSCOPE	same as acc
BAROMETER	CFG_BAROMETER_SUPPORT =yes CFG_BMP280_SUPPORT =yes

6.4.3 Fusion sensors build option

1. **Device config**

   Patch: /device/mediatek/$project/ProjectConfig.mk

ORIENTATION	CUSTOM_KERNEL_ORIENTATION_SENSOR=yes
GRAVITY	CUSTOM_KERNEL_GRAVITY_SENSOR=yes
LINEAR_ACCELERATION	CUSTOM_KERNEL_LINEARACCEL_SENSOR=yes
ROTATION_VECTOR	CUSTOM_KERNEL_RV_SENSOR=yes
MAGNETIC_FIELD_UNCALIBRATED	CUSTOM_KERNEL_UNCALI_MAG_SENSOR=yes
GAME_ROTATION_VECTOR	CUSTOM_KERNEL_GRV_SENSOR=yes
GYROSCOPE_UNCALIBRATED	CUSTOM_KERNEL_UNCALI_GYRO_SENSOR=yes
GEOMAGNETIC_ROTATION_VECTOR	CUSTOM_KERNEL_GMRV_SENSOR=yes

2. **Kernel config**

   Path:

   /kernel-4.4/arch/$TARGET_ARCH/configs/$project_defconfig

   /kernel-4.4/arch/$TARGET_ARCH/configs/$project_debug_defconfig

ORIENTATION	CONFIG_MTK_ORIENTHUB=y
GRAVITY	CONFIG_CUSTOM_KERNEL_GRAVITY_SENSOR=y CONFIG_MTK_GRAVITYHUB=y
LINEAR_ACCELERATION	CONFIG_CUSTOM_KERNEL_LINEARACCEL_SENSOR=y CONFIG_MTK_LINEARACCHUB=y

ROTATION_VECTOR	CONFIG_CUSTOM_KERNEL_RV_SENSOR=y CONFIG_MTK_ROTATVECHUB=y
MAGNETIC_FIELD_UNCALIBRATED	CONFIG_CUSTOM_KERNEL_UNCALI_MAG_SENSOR=y CONFIG_MTK_UNCALI_MAGHUB=y
GAME_ROTATION_VECTOR	CONFIG_CUSTOM_KERNEL_GRV_SENSOR=y CONFIG_MTK_GAMEROTVECHUB=y
GYROSCOPE_UNCALIBRATED	CONFIG_CUSTOM_KERNEL_UNCALI_GYRO_SENSOR=y CONFIG_MTK_UNCALI_GYROHUB=y
GEOMAGNETIC_ROTATION_VECTOR	CONFIG_CUSTOM_KERNEL_GMRV_SENSOR=y CONFIG_MTK_GMAGROTVECHUB=y

3. SCP config

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt6758/

工程名/projectconfig.mk

CFG_FUSION_SUPPORT=yes

6.4.4 Pedometer build option

1. Device config

Patch: /device/mediatek/$project/ProjectConfig.mk

SIGNIFICANT_MOTION	CUSTOM_KERNEL_SIGNIFICANT_MOTION_SENSOR=yes
STEP_DETECTOR	CUSTOM_KERNEL_STEP_COUNTER=yes
STEP_COUNTER	

2. Kernel config

Path:

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_defconfig

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_debug_defconfig

SIGNIFICANT_MOTION	CONFIG_CUSTOM_KERNEL_STEP_COUNTER=y CONFIG_MTK_STEPSIGNHUB=y
STEP_DETECTOR	
STEP_COUNTER	

3. **SCP config**

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt6758/工程名/projectconfig.mk

SIGNIFICANT_MOTION	CFG_SIGNIFICANT_MOTION_SUPPORT=yes
STEP_DETECTOR	CFG_STEP_COUNTER_SUPPORT=yes
STEP_COUNTER	CFG_STEP_DETECTOR_SUPPORT=yes

6.4.5 Situation & Gesture build option

1. **Device config**

Patch: /device/mediatek/$project/ProjectConfig.mk

TILT_DETECTOR	CUSTOM_KERNEL_TILT_DETECTOR_SENSOR=yes
WAKE_GESTURE	CUSTOM_KERNEL_WAKE_GESTURE_SENSOR=yes
GLANCE_GESTURE	CUSTOM_KERNEL_GLANCE_GESTURE_SENSOR=yes
PICK_UP_GESTURE	CUSTOM_KERNEL_PICK_UP_SENSOR=yes
DEVICE_ORIENTATION	CUSTOM_KERNEL_DEVICE_ORIENTATION=yes
STATIONARY_DETECT	CUSTOM_KERNEL_STATIONARY_SENSOR=yes
ANSWERCALL	CUSTOM_KERNEL_ANSWER_CALL_SENSOR=yes
MOTION_DETECT	CUSTOM_KERNEL_MOTION_DETECT=yes

2. **Kernel config**

Path:

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_defconfig

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_debug_defconfig

TILT_DETECTOR	CONFIG_MTK_TILTDETECTHUB=y
WAKE_GESTURE	CONFIG_MTK_WAKEHUB=y
GLANCE_GESTURE	CONFIG_MTK_GLGHUB=y

PICK_UP_GESTURE	CONFIG_MTK_PICKUPHUB=y
DEVICE_ORIENTATION	CONFIG_MTK_DEVICE_ORIENTATION_HUB=y
STATIONARY_DETECT	CONFIG_MTK_STATHUB=y
ANSWERCALL	CONFIG_MTK_ANSWER_CALL_HUB=y
MOTION_DETECT	CONFIG_MTK_MOTION_DETECT_HUB=y

3. **SCP config**

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt6758/工程名/projectconfig.mk

TILT_DETECTOR	CFG_TILT_SUPPORT=yes
WAKE_GESTURE	CFG_WAKEUP_SUPPORT=yes
GLANCE_GESTURE	CFG_SNAPSHOT_SUPPORT=yes
PICK_UP_GESTURE	CFG_PICKUP_SUPPORT=yes
DEVICE_ORIENTATION	CFG_WIN_ORIENTATION_SUPPORT=yes
STATIONARY_DETECT	CFG_STATIONARY_SUPPORT=yes
ANSWERCALL	CFG_ANSWERCALL_SUPPORT=yes
MOTION_DETECT	CFG_MOTION_SUPPORT=yes
Floor count	CFG_FLOOR_COUNT_SUPPORT=yes
Lift to wake	CFG_LIFT_SUPPORT =yes

6.4.6 Activity build option

1. **Device config**

Patch: /device/mediatek/$project/ProjectConfig.mk

CUSTOM_KERNEL_ACTIVITY_SENSOR=yes

2. **Kernel config**

Path:

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_defconfig

/kernel-4.4/arch/$TARGET_ARCH/configs/$project_debug_defconfig

CONFIG_CUSTOM_KERNEL_ACTIVITY_SENSOR=y

CONFIG_MTK_ACTIVITYHUB=y

3. **SCP config**

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt6758/

工程名/projectconfig.mk

CFG_ACTIVITY_NO_BARO_SUPPORT=yes

CFG_ACTIVITY_BARO_SUPPORT=yes

6.4.7 Enable SCP virtual gyro(based on AKM M-sensor)

SCP config

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt67xx/

工程名/projectconfig.mk

CFG_VIRTUAL_GYRO_SUPPORT = yes

CFG_AKM_FUSION_SUPPORT = yes

CFG_FUSION_SUPPORT = no // 关闭MTK自己的fusion算法，virtual gyro 配套AKMfusion

CFG_FAST_CALIBRATION_SUPPORT = no // 没有gyro，AKM不支持快速校准

6.4.8 Enable SCP MTK fusion algorithm (need physical gyro)

Patch: /vendor/mediatek/proprietary/tinysys/freertos/source/Project/cm4_a/mt67xx/

工程名/projectconfig.mk

CFG_VIRTUAL_GYRO_SUPPORT = no

CFG_AKM_FUSION_SUPPORT = no

CFG_FUSION_SUPPORT = yes // enable MTK fusion algorithm

CFG_FAST_CALIBRATION_SUPPORT = yes // AKM 的磁力方案，可以尝试开快速校准

6.5 Debug

6.5.1 How to enable SCP Uart

vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt67xx/platform/platform.mk

```
CFG_UART_SUPPORT = yes
```

Attention: uart DO NEED to be disabled during QA test, otherwise there will be performance issue.

6.5.2 SCP uart reuses AP uart

Enable by modify config

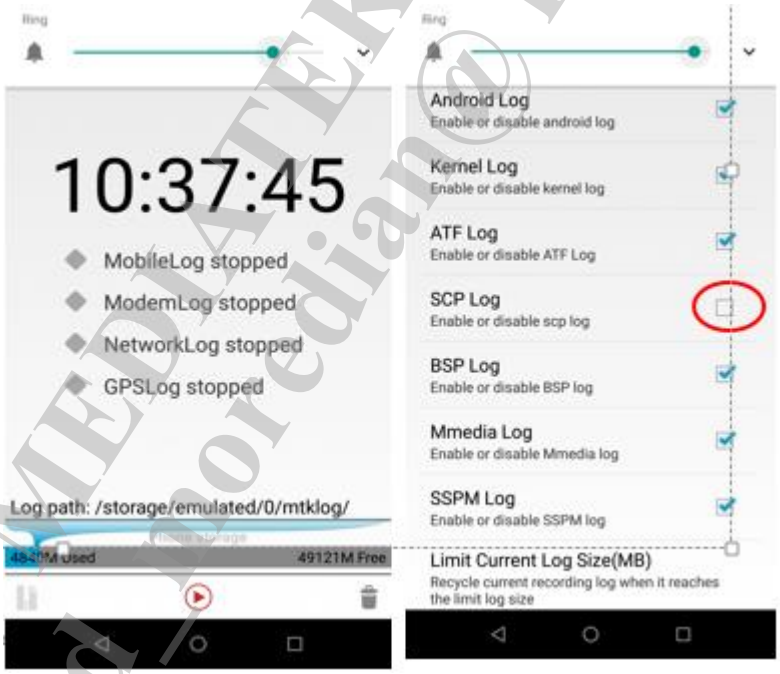
project/CM4_A/mt6771/platform/platform.mk

- Warning
  - DO NOT apply this change to ENG build, because AP and SCP log will mix together and hard to recognize.
  - DO NOT use this when measure suspend power, it keeps infra always on.

```
CFG_UART_SUPPORT = yes
CFG_MTK_SCPUART_SUPPORT = yes

# CFG_MTK_APUART_SUPPORT
# Do not use this with eng load or log may mix together and hard to recognzie
# Do not use this on lower power, it keeps infra always on
CFG_MTK_APUART_SUPPORT = no
```

6.5.3 usb outputs SCP log directly

1. Make sure mobile log (SCP part) is disabled
   1) All mobile log disable
   2) Or Disable SCP log



2. Enter adb shell and then type the following cmd

1) echo 1 > /sys/class/misc/scp/scp_mobile_log
2) while true; do cat /dev/scp;done

```
C:\Users\MTK11261>adb shell
k71v1_64_bsp:/ # echo 1 > /sys/class/misc/scp/scp_mobile_log
k71v1_64_bsp:/ # while true; do cat /dev/scp;done
103284, ap:39031836157017, ap_raw:39031836071402
raw_offset:2290053733, timestamp_offset_to_ap:2290053733
sync time counter_elapse:1126, ipi_transfer_time:86615
sync time scp:39039562168462, ap:39041852222325, ap_raw:39041852135710
raw_offset:2290053863, timestamp_offset_to_ap:2290053863
No sleep reasons: tmr=0, build=0, sema=0, lock=0, ipi=0, flag=4, slpbusy=0
sync time counter_elapse:1129, ipi_transfer_time:86846
sync time scp:39049578174332, ap:39051868228172, ap_raw:39051868141326
raw_offset:2290053840, timestamp_offset_to_ap:2290053840
sync time counter_elapse:1130, ipi_transfer_time:86923
sync time scp:39059594165433, ap:39061884219173, ap_raw:39061884132250
raw_offset:2290053740, timestamp_offset_to_ap:2290053740
sync time counter_elapse:1111, ipi_transfer_time:85461
sync time scp:39069610088456, ap:39071900142173, ap_raw:39071900056712
raw_offset:2290053717, timestamp_offset_to_ap:2290053717
No sleep reasons: tmr=0, build=0, sema=0, lock=0, ipi=0, flag=3, slpbusy=0
log en=1,update=1
sync time counter_elapse:1208, ipi_transfer_time:92923
sync time scp:39079626695403, ap:39081916749174, ap_raw:39081916656251
raw_offset:2290053771, timestamp_offset_to_ap:2290053771
^C
```

6.5.4 SCP open EE DB mechanism

- 1 .ensure AEE mechanism is enabled
  - Execute adb shell “aee -m 3”。Through adb shell “getprop persist.mtk.aee.mode”, check whether the return value is 3. If 3, the execution is successful.
- 2.ensure SCP db can dump info
  - Do adb command:
  - 1) adb shell cat /sys/class/misc/scp/scp_A_db_test (會看到返回 dumping SCP A db)
  - 2）find the db in the following path

sdcard/mtklog/aee_exp/data/aee_exp/

6.5.5 SCP reboot correlates AP reboot

Function：Force enable KE when SCP EE occur

- Default Status: DISABLE
  - How to switch: write the control node to turn on/off
    - How to use explain in next page
- When Enable:
  - Reset scope: Whole system (KE)
  - Debug info: Full RAM Dump (takes a long time) and mobile log
    - db = db.xx.EE & db.fatal.xx.KE
- When Disable:
  - Reset scope: SCP only (EE)
  - Debug info: SCP db and mobile log

- db = db.xx.EE

1. Control node:
   - Path: /sys/class/misc/scp/scp_ee_force_ke
   - Enable
     - echo 1 > /sys/class/misc/scp/scp_ee_force_ke
   - Disable
     - echo 0 > /sys/class/misc/scp/scp_ee_force_ke
2. Selinux permission settings

Must allow to access the path: /sys/class/misc/scp/scp_ee_force_ke

Ex： to enable eng_app access sysfs_scp should do the following

Under device/mediatek/sepolicy/basic/non_plat/eng_app.te

add your own file xxx.te, the eng_app in this example.

The content is as the following:

```
# Purpose: Allow eng_ap read /sys/class/misc/scp/scp_ee_force_ke
allow eng_ap sysfs_scp:dir r_dir_perms;
allow eng_ap sysfs_scp:file r_file_perms;
```

3. Property settings

device/mediatek/mt6771/init.mt6771.rc add contents as following：

```
# Add by MTK
  # SCP log
  ...
  chmod 0664 /sys/class/misc/scp/scp_ee_force_ke
  chown root system /sys/class/misc/scp/scp_ee_force_ke
```

6.5.6 Dynamic AP/SCP UART Switch

- Default enable AP uart support, switch with Fastboot cmd line
- Warning!! Limitation & side effect
  - extra code size require(+560 bytes)
  - Must disable AP uart log self
  - Timing impact, for debug only, can not use it on stress test
  - Power impact

(DO NOT apply this change to ENG build, because AP and SCP log will mix together and hard to recognize)

(DO NOT use this when measure suspend power, it keeps infra always on

1. How to use:

   Set CFG_MTK_DYNAMIC_AP_UART_SWITCH = yes

   (@ project/CM4_A/mt6771/platform/platform.mk)

```
################################################################################
# When the option CFG_MTK_DYNAMIC_AP_UART_SWITCH is set to "yes", the code
# of UART will be built into the SCP image. This leads to a larger image.
# Set this option to "yes" only when you really know what you are doing.
# Otherwise, set it to "no".
################################################################################
CFG_MTK_DYNAMIC_AP_UART_SWITCH = yes
ifeq ($(CFG_MTK_DYNAMIC_AP_UART_SWITCH), yes)
CFG_UART_SUPPORT = yes
CFG_MTK_SCPUART_SUPPORT = no
CFG_MTK_APUART_SUPPORT = yes
endif
```

2. How to dynamic switch
   1) enter lk fastboot
      - use adb reboot bootloader (@adb shell)
      - or booting menu (Power key + Volume- boot up) -> fastboot
   2) switch with fastboot cmd
      - enable: fastboot oem scp_log_thru_ap_uart 1
      - disable: fastboot oem scp_log_thru_ap_uart 0

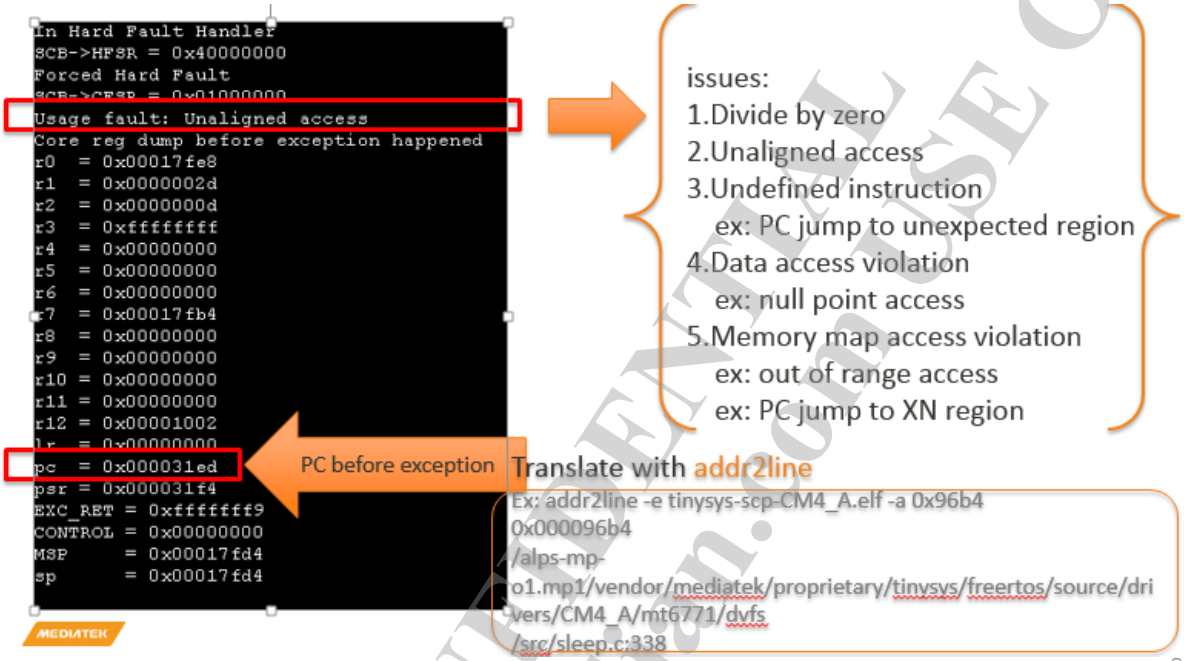
```
D:\>fastboot oem scp_log_thru_ap_uart 1
...
(bootloader) SCP log thru AP UART: on
(bootloader) Please reboot to apply the change.
OKAY [  0.010s]
finished. total time: 0.011s
```

   3) reboot (remember disable AP uart)

6.5.7 SCP Exception debug

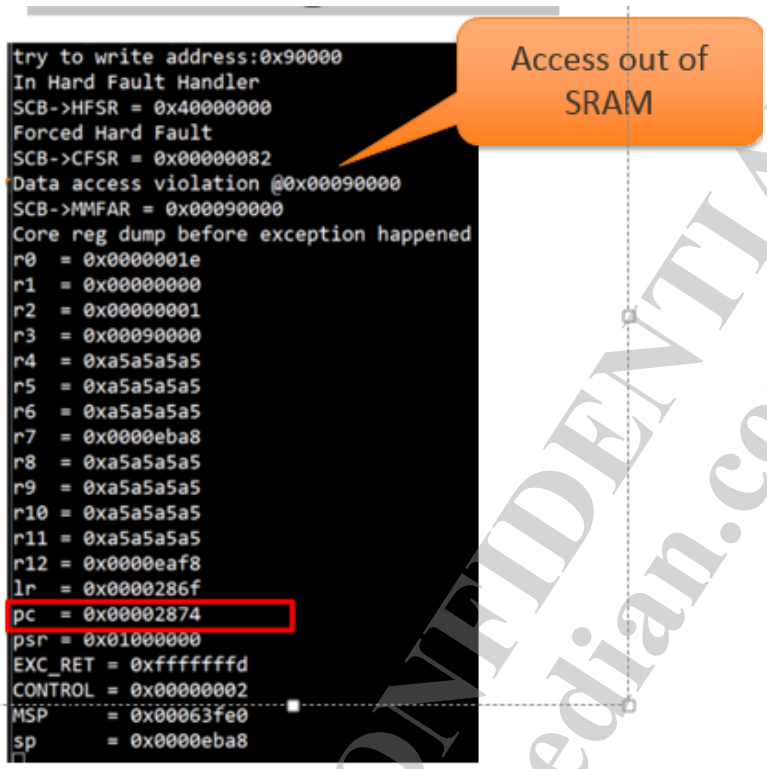
1. **How to get exception log**
   1) From uart/mobile log
   2) From db
      i. Extract SCP EE DB, it can see SYS_SCP_DUMP
2. **Exception category introduction**

If get exception log already

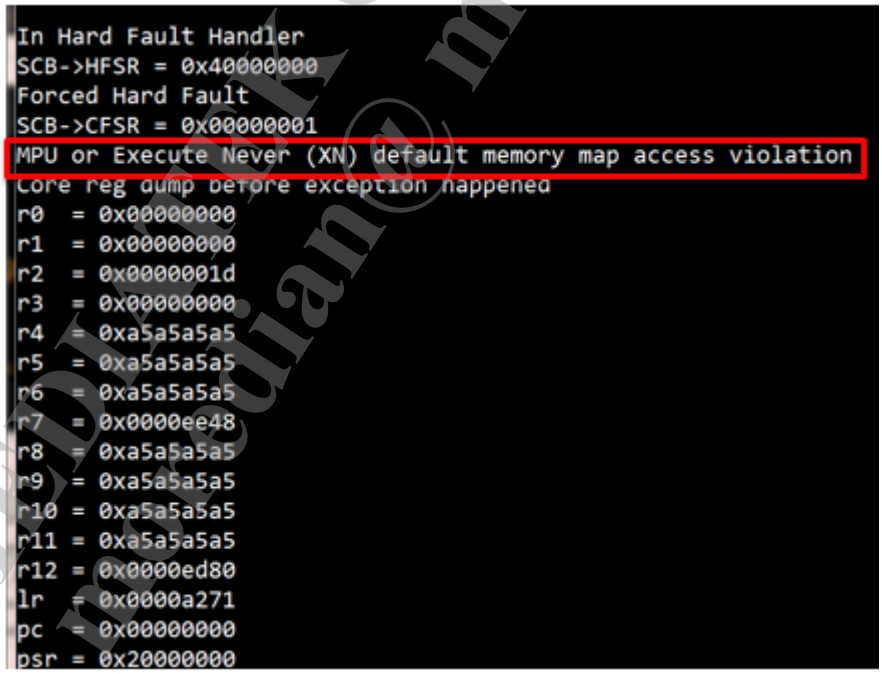


1) Divde by zero

2) Out of range access



3) Jump to XN region



4) Null pointer access

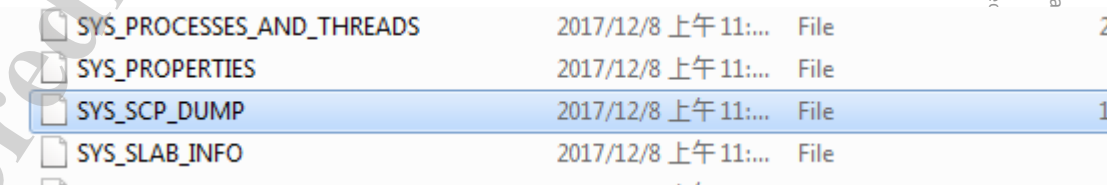
```
try to write address:0x0
In Hard Fault Handler
SCB->HFSR = 0x40000000
Forced Hard Fault
SCB->CFSR = 0x00000082
Data access violation @0x00000000
SCB->MMFAR = 0x00000000
Core reg dump before exception happened
r0  = 0x0000001a
r1  = 0x00000000
r2  = 0x00000001
r3  = 0x00000000
r4  = 0xa5a5a5a5
r5  = 0xa5a5a5a5
r6  = 0xa5a5a5a5
r7  = 0x0000eba8
r8  = 0xa5a5a5a5
r9  = 0xa5a5a5a5
r10 = 0xa5a5a5a5
r11 = 0xa5a5a5a5
r12 = 0x0000eaf8
lr  = 0x00002833
pc  = 0x00002836
psr = 0x01000000
EXC_RET = 0xffffffd
CONTROL = 0x00000002
MSP     = 0x00063fe0
sp      = 0x0000eba8
```

3. **Debug ram dump with gdb**

GDB download

- Get android ndk
  - https://developer.android.com/ndk/downloads/index.html
- You can find it in prebuild folder
  - Ex: prebuild/linux-x86_64/bin/gdb

1) Get ramdump
   i. Get SCP EE DB and extract it, it can see SYS_SCP_DUMP
   ii. If size of SYS_SCP_DUMP is 0, this issue is probably not an SCP issue



2) See last log

Run command, strings, to parse the ram dump:

```
strings SYS_SCP_DUMP | less
```

In this case, it shows “assert” in kernel/FreeRTOS/Source/timers.c:869.

We can know the failed point. The PC backtrace is also helpful. You can use addr2line to lo locate the problem

```
Init DRAMC OK
[DVFS-SCP] in dvfs_init
[DVFS-SCP] -INFO set_clk_sys_settle_time(0x1f)
[DVFS-SCP] -INFO set_clk_high_settle_time(0x7)
[DVFS-SCP] in get_cur_clk
[DVFS-SCP] in get_clk_div_select
[DVFS-SCP] -INFO cur_clk = 2, cur_div = 1
[DVFS-SCP] in use_SCP_ctrl_sleep_mode
[DVFS-SCP] in disable_SCP_sleep_mode
[CCCI0/INIT]register IPI 25, 0
 task:CC_I was created(traceTASK_CREATE())
[CCCI0/INIT]create IPI task 1 0xffa8
[CCCI0/INIT]rWrong parameters value: file ./kernel/FreeRTOS/Source/timers.c on line 869
==PC backtrace dump start==
0x00009b4c
0x00009b72
0x000071ec
0x0000452a
0x000072e0
0x00007baa
0x00002d56
0x00009aca
==PC backtrace dump end==
```

assert happened file and line

pc backtrace information

3) **Debug ram dump with gdb**

[Shell]$ ./gdb tinysys-scp-CM4_A.elf SYS_SCP_DUMP

Further cmds please reference GDB guide

Back trace: bt

```
GNU gdb (GDB) 7.11
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from tinysys-scp-CM4_A.elf...done.

warning: core file may not match specified executable file.
[New LWP 1]
Core was generated by `freertos8'.
#0  0x00015874 in parseRawData (timeStamp=341325329007, outBuf=<optimized out>) at middleware/contexthub/MEMS_Driver/magnetometer/magnetometer.c:645
645     middleware/contexthub/MEMS_Driver/magnetometer/magnetometer.c: Permission denied.
(gdb) bt
#0  0x00015874 in parseRawData (timeStamp=341325329007, outBuf=<optimized out>) at middleware/contexthub/MEMS_Driver/magnetometer/magnetometer.c:645
#1  handleSensorEvent (state=<optimized out>) at middleware/contexthub/MEMS_Driver/magnetometer/magnetometer.c:768
#2  handleEvent (evtType=<optimized out>, evtData=<optimized out>) at middleware/contexthub/MEMS_Driver/magnetometer/magnetometer.c:802
#3  0x6100f000 in ?? ()
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
```

GDB Basic cmds: p variable

```
(gdb) p mTask
$2 = {id = 261, magHandle = 17104901, magTimerHandle = 15205, rate = 51200, latency = 19999744, pendConfig = {late
  pendingFlushFifo = false, magNowOn = true, magReading = true, configed = true, fifoEnabled = false, flush = 0 '\
  elemOutSize = 12 '\f', timerDelay = 20000000, fifoDelay = 0, fifoStartTime = 0, prev_rtc_time = 341325329007, mD
    mSensorFsm = 0x732c0 <mmc3530Disable>, mCurrFsm = 0x732cc <mmc3530Disable+12>, mNextFsm = 0x0, mSensorFsmSize =
    caliApiSetOffset = 0x72fa9 <akm09915CaliApiSetOffset>, caliApiSetGyroData = 0x72d85 <akm09915CaliApiSetGyroDat
(gdb) p &mTask
$3 = (struct magTask *) 0x4ef08 <mTask>
```

Dump memory : x/FMT address

```
(gdb) x/32xw 0x4ef08
0x4ef08 <mTask>:        0x00000105      0x01050005      0x00003b65      0x0000c800
0x4ef18 <mTask+16>:     0x01312c00      0x00000000      0x00000000      0x00000000
0x4ef28 <mTask+32>:     0x00000000      0x00000000      0x00000000      0x00010101
0x4ef38 <mTask+48>:     0x00000300      0x00073368      0x00000c01      0x01312d00
0x4ef48 <mTask+64>:     0x00000000      0x00000000      0x00000000      0x00000000
0x4ef58 <mTask+80>:     0x7893326f      0x0000004f      0x00042120      0x00042510
0x4ef68 <mTask+96>:     0x00030209      0x000732c0      0x000732cc      0x00000000
0x4ef78 <mTask+112>:    0x00000809      0x00072d81      0x00072fa9      0x00072d85
```

6.5.8 Sensor driver debug trace

1. **SPI driver debug trace**

   Because CHRE SPI is not like i2c ，use SPI dump register need to follow the already existed sensor flow

   Example ：

   In a power on flow to judge debug trace（debug trace can define such as, 0x1,0x2 ……）, after debug trace was set , you can use synchronization SPI api to dump register,

```
static int lsm6dsmAccPowerOn(I2cCallbackF i2cCallBack, SpiCbkF spiCallBack, void *next_state,
                             void *inBuf, uint8_t inSize, uint8_t elemInSize,
                             void *outBuf, uint8_t *outSize, uint8_t *elemOutSize)
{
    osLog(LOG_INFO, "lsm6dsmAccPowerOn\n");
    int ret = 0;
    uint8_t txData[2] = {0}, rxData[2] = {0};
    if(mTask.debug_trace == 0x1)
    {
       //dump register
        osLog(LOG_ERROR, "lsm6dsm: fwq dump reg\n");

        txData[0] = LSM6DSM_WAI_ADDR | 0x80;
        ret = spiMasterRxTxSync(mTask.spiDev, rxData, txData, 2);
        osLog(LOG_ERROR, "lsm6dsm: device id: %02x , %d\n", rxData[1],ret);
       return 0;
    }
```

adb cmd：

```
/sys/bus/platform/drivers/gsensor # echo 0x1 > trace
```

Note ： after dump must you must return 0, and then sensor cannot work, you must restart the phone

2. **I2c driver debut trace**

   You can call i2c synchronization API in the debug trace call back function

```
static void ltr578SetDebugTrace(int32_t trace) {
    int ret = 0;
    mTask.debug_trace = trace;
    osLog(LOG_ERROR, "%s ==> trace:%d\n", __func__, mTask.debug_trace);
    // can use i2cMasterTxRxSync API dump register whitch you wanted
     ret = i2cMasterTxRxSync(mTask.hw->i2c_num, mTask.i2c_addr, mTask.txBuf, 1,
        &mTask.deviceId, 1, NULL, NULL);
    if (ret < 0) {
        osLog(LOG_ERROR, "bmp280 i2cMasterTxRxSync fail!!!\n");
        ret = -1;
        i2cMasterRelease(mTask.hw->i2c_num);
        goto err_out;
    }
}
```