MTK Sensors Customer Document

Version:	1.0.3
Release date:	2017-12-15

目录

Document Revision History

Revision	Date	Author	Description
1.0	2017-11-24	MTK	
1.0.1	2017-12-07	MTK	修正 Sensor Calibration 校准章节，loadable 说明章节
1.0.2	2017-12-10	MTK	增加 Tinysis 添加 driver 章节,和实现 CHRE APP 章节
1.0.3	2017-12-13	MTK	更新 SCP 关联 AP 重启章节

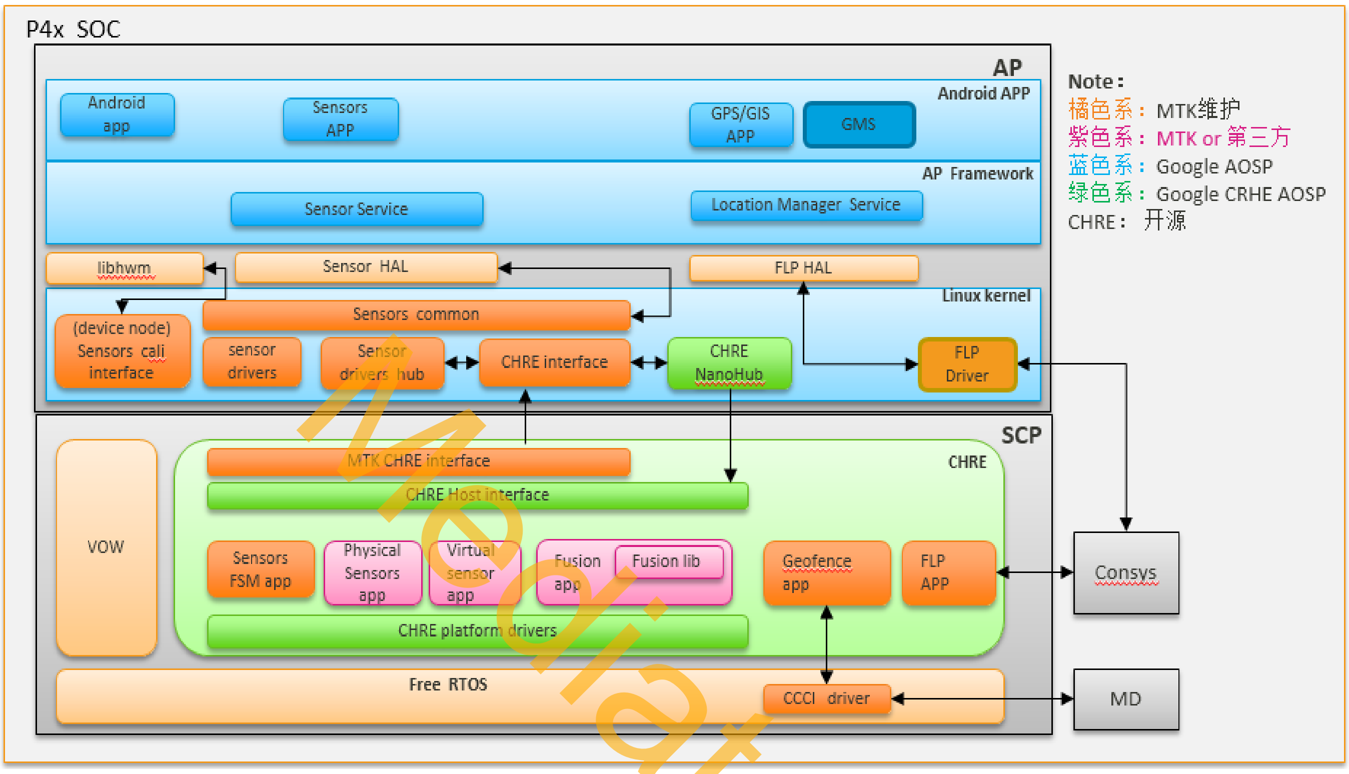
1 概述

1.1 主要内容

文档主要内容包括，MTK Sensor 的架构说明，porting Guild ，提供的 API 接口，build option 以及 debug 方式。
分为 AP side 和 SCP side 介绍
本文档适用 MTK P40 平台

2 Architecture overview

2.1 Architecture



MTK Sensor 分为 AP 和 SCP 两大部分，AP (CA5x ，CA7x 系列 主芯片)， SCP（CM4）协处理器。负责处理 Sensor 数据。实际使用中，也可以关闭 SCP 只走 AP 这路实现 sensor 功能

3 AP sensors introduction

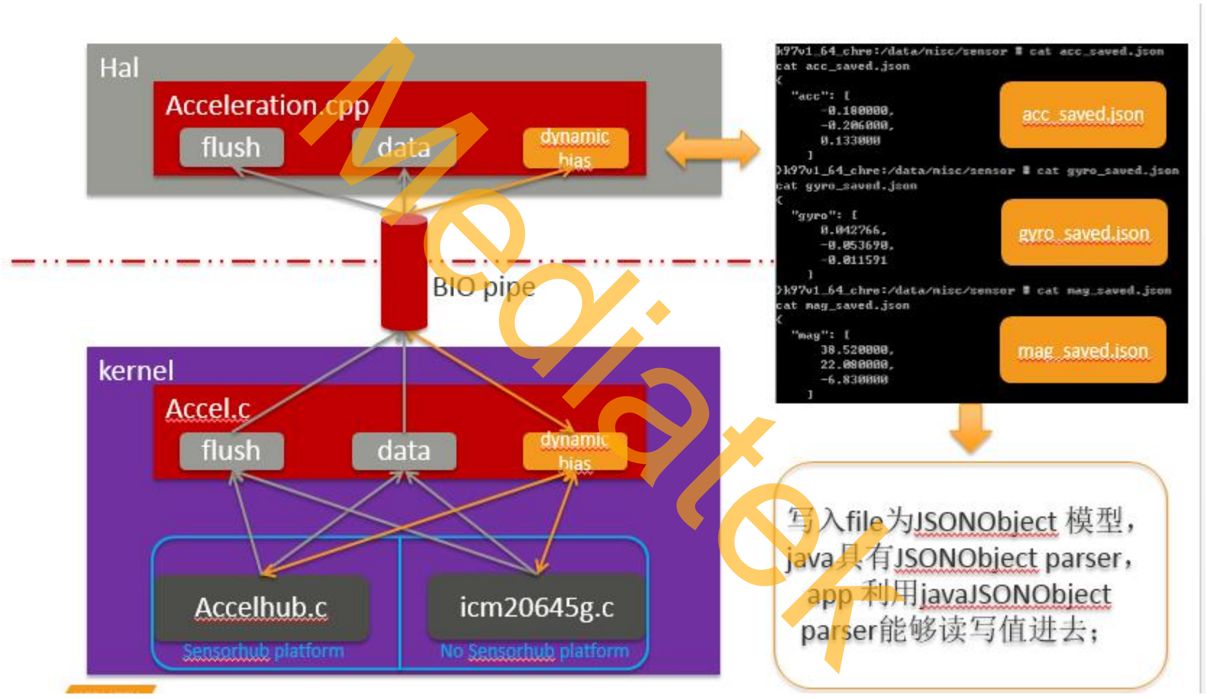
主要介绍 HAL 层， kernel 层的接口，以及 AP 端实现 sensor 功能的 porting Guild

3.1 Linux sensors drivers interface

本章节介绍 MTK kernel common sensor 的 Interface，以及 porting guiled

3.1.1 MTK sensor common layer introduction

实现 common 层的目的是简化客户 porting 时间。
另外 common 层对接 MTK HAL 使用的是 MTK 私有的 Interface (BIO pipe)效率大大优于 input event。



```
k97v1_64_chre:/data/misc/sensor # cat acc_saved.json
cat acc_saved.json
{
  "acc": [
       -0.180000,
       -0.206000,
       0.133000
    ]
}k97v1_64_chre:/data/misc/sensor # cat gyro_saved.json
cat gyro_saved.json
{
  "gyro": [
       0.042766,
       -0.053670,
       -0.011591
    ]
}k97v1_64_chre:/data/misc/sensor # cat mag_saved.json
cat mag_saved.json
{
  "mag": [
       38.520000,
       22.080000,
       -6.830000
    ]
```

写入file为JSONObject 模型，java具有JSONObject parser，app 利用javaJSONObject parser能够读写值进去；

3.1.2 Common layer API 介绍

下面以加速计为例，介绍 common API 的用法。
API 主要由两个头文件提供。

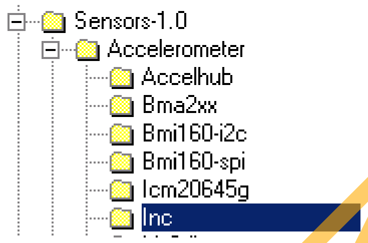
1. **Accel.h,** 提供接入 Android 层的 data flow 和 contrl flow 的 API
2. **Acc_factory.h** 提供接入 MTK 工厂模式的 data flow 和 contrl flow 的 API

Accel.h 实现 API 如下，即可打通上报道 Android 的数据通路：

API Name	Parameter description
acc_driver_add(struct acc_init_info* obj)	acc_init_info 包含了 sensor 的相关信息，通过将这个结构体注册到 common driver 实现 sensor auto detect 的功能
acc_register_data_path(struct acc_data_path *data)	acc_data_path 中包含了一个获取 sensor 数据的 function 和一个用于归一化不同厂商 sensor raw data 的

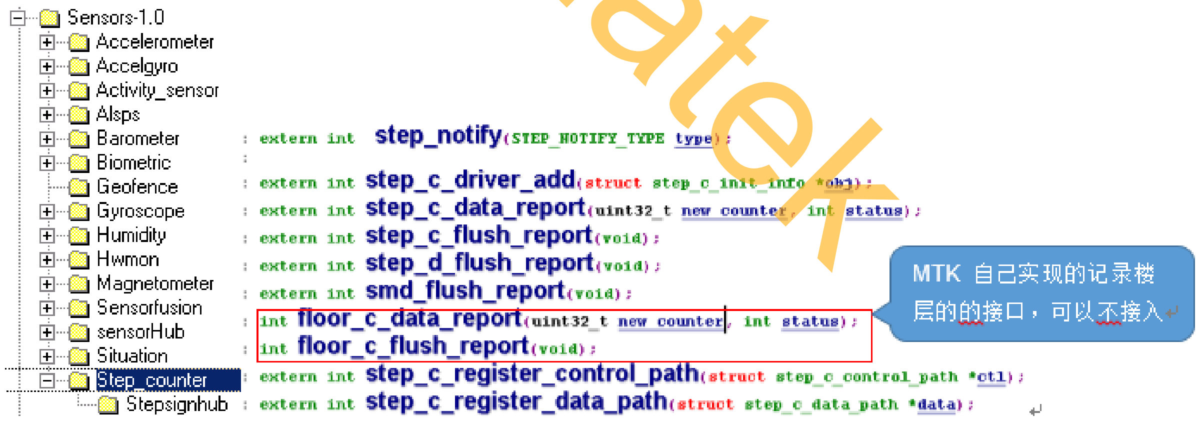
	算术值，这两个数据将被注册到 common sensor driver 的架构中去，为 MTK 架构获取 sensor 数据
acc_register_control_path(struct acc_control_path *ctl)	acc_control_path 包含了 3 个 function 和一个布尔型参数，用于控制 sensor 的 enable/disable，setDelay 等
acc_data_report()	用于上报数据
Acc_flush_report()	上报 flush

调用 API 要传入的结构体可以参考已经完成的 driver 实现。



3.2 Step counter 接入

介绍使用 acc 自带加速度计，如何将 step counter 数据接入可以参考如下 fodler 的接口，和前面介绍的 acc common 接口类似。

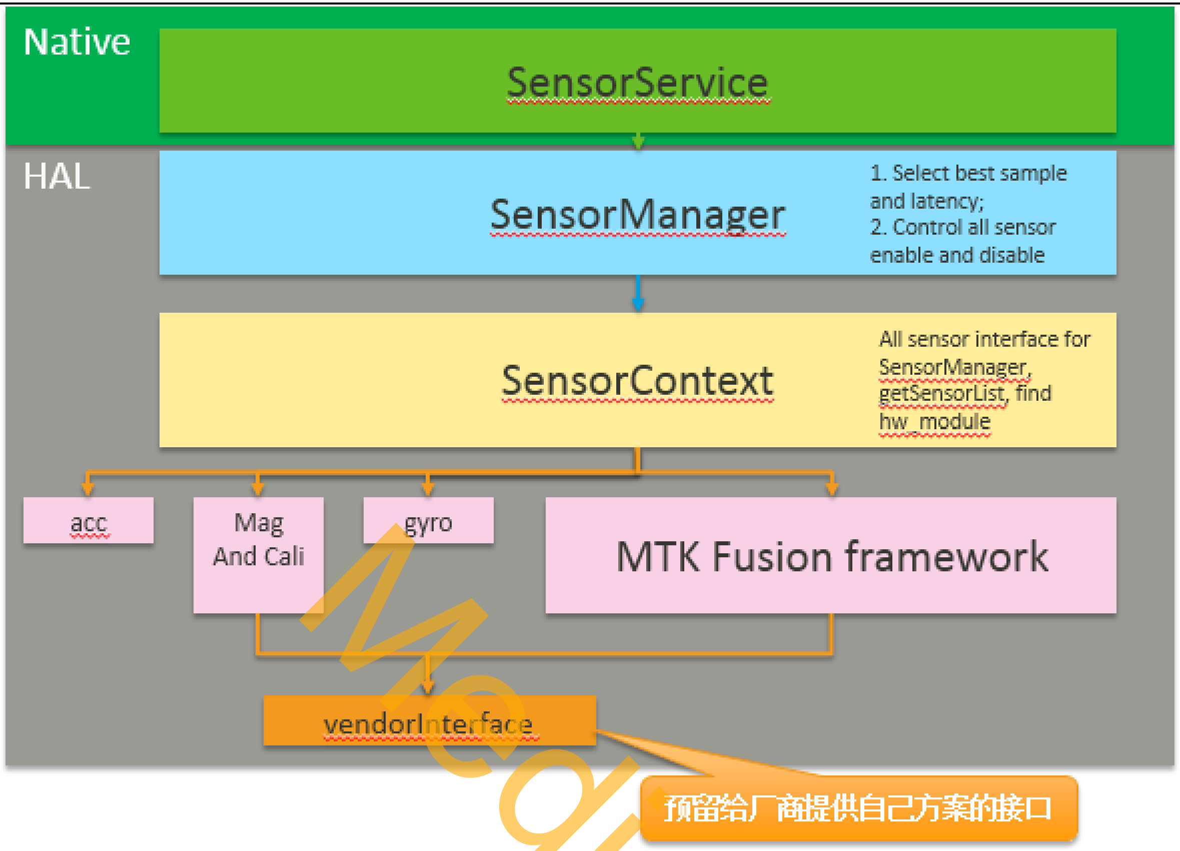


```
extern int  step_notify(STEP_NOTIFY_TYPE type);

extern int step_c_driver_add(struct step_c_init_info *obj);
extern int step_c_data_report(uint32_t new_counter, int status);
extern int step_c_flush_report(void);
extern int step_d_flush_report(void);
extern int smd_flush_report(void);
int floor_c_data_report(uint32_t new_counter, int status);
int floor_c_flush_report(void);
extern int step_c_register_control_path(struct step_c_control_path *ctl);
extern int step_c_register_data_path(struct step_c_data_path *data);
```

3.3 Sensor HAL Interface

MTK HAL 层提供接入第三方算法的接口，因应客户的不同需求。如客户希望在 native 层实现 fusion，virtual gyro 等一系列的算法。

MTK HAL 架构如下：


Native
SensorService
HAL
SensorManager
1. Select best sample and latency;
2. Control all sensor enable and disable
SensorContext
All sensor interface for SensorManager, getSensorList, find hw_module
acc
Mag And Cali
gyro
MTK Fusion framework
vendorInterface
预留给厂商提供自己方案的接口

3.3.1 SensorManger user manual

- Get sensormanager interface
  - mSensorManager = SensorManager::getInstance();
- Create sensor connection
  - mSensorConnection = SensorManager::createSensorConnection(magnetic);
- Enable sensor
  - mSensorManager->batch(mSensorConnection, ID_ACCELEROMETER, 20000000, 0);
  - mSensorManager->activate(mSensorConnection, ID_ACCELEROMETER, true);
- Disable sensor
  - mSensorManager->activate(mSensorConnection, ID_ACCELEROMETER, false);
- Remove sensor connection
  - mSensorManager->removeSensorConnection(mSensorConnection);

```
struct SensorManager {
    static SensorManager *getInstance();
    SensorConnection* createSensorConnection(int mSensorMoudle);
    void removeSensorConnection(SensorConnection* connection);
    void addSensorsList(sensor_t const *list, size_t count);
    int activate(SensorConnection *connection, int32_t sensor_handle, bool enabled);
    int batch(SensorConnection *connection, int32_t sensor_handle,
            int64_t sampling_period_ns,
            int64_t max_report_latency_ns);
    int flush(SensorConnection *connection, int32_t sensor_handle);
    int pollEvent(sensors_event_t* data, int count);
    int setEvent(sensors_event_t* data, int moudle);
    void setNativeConnection(SensorConnection *connection);
    void setSensorContext(sensors_poll_context_t *context);
```

3.3.2 Vendor Interface user manual

- Get vendor interface
  - mVendorInterface = VendorInterface::getInstance();

```
struct VendorInterface {
public:
    static VendorInterface *getInstance();
    ~VendorInterface();
    int setMagOffset(float offset[3]);
    int getMagOffset(float offset[3]);
    int setGyroData(struct magCaliDataInPut *inputData);
    int setAccData(struct magCaliDataInPut *inputData);
    int setMagData(struct magCaliDataInPut *inputData);
    int magCalibration(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);
```

3.4 Sensors Calibration 接口

MTK 提供可供 Android APP 调用的 Calibration 接口，目前支持 ACC，和 Gyro 的零值校准。支持 Proximity 的 threshold 设定。 MTK 提供工模 APK 作为 API 使用范例。

3.4.1 Debug 命令

用命里进行 accel and gyroscope calibration 过程以下命令均在 cmd 命令窗口进行）：

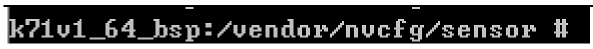
1、adb root

2、adb shell

3、若需要 debug Accel 的 calibration ： cd sys/bus/platform/drivers/gsensor

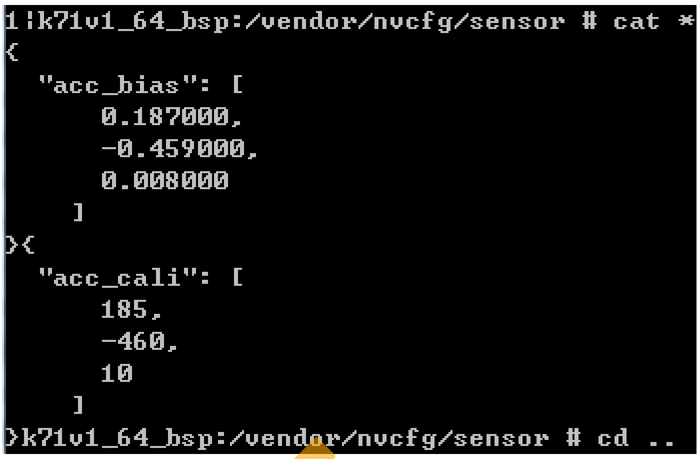
若需要 debug Gyro 的 calibration ： cd sys/bus/platform/drivers/gyroscope

4、进行 Calibration，手机放平，不能有任何微小移动 ： echo 1 > test_cali



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

5. 查看 calibration 结果：



```
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 ..
```

3.4.2 Accel & Gyro 零值校准接口

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

Accel Calibration API

Int gsensor_start_static_calibration(void);

- ✓ 返回值为 0 或非 0，0：成功；非 0：失败

Int gsensor_get_static_calibration(sensorData *sensorDat);

- ✓ 返回值为 0 或非 0，0：成功；非 0：失败
  传入参数为回传的校准数据用于显示在 UI 界面上面,x=sensorDat->data[0], y=sensorDat->data[1], z=sensorDat->data[2]
- ✓ 当 gsensor_start_static_calibration 返回 true 时方可调用此 api 获取静态校准数据

Gyroscope Calibration API

Int gyroscope_start_static_calibration (void);

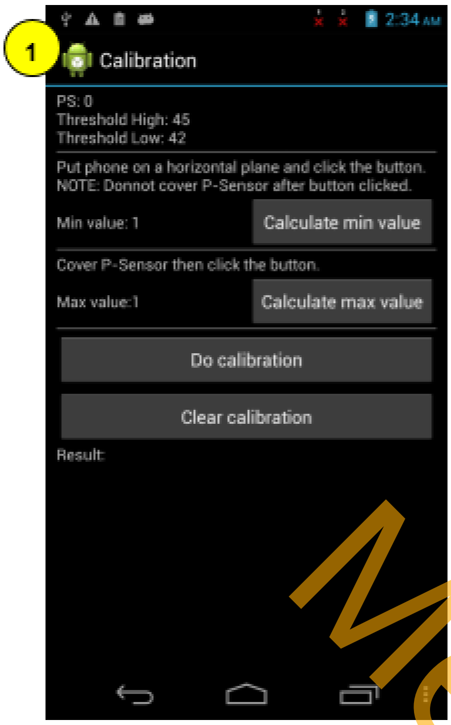
- ✓ 返回值为 0 或非 0，0：成功；非 0：失败

Int gyroscope_get_static_calibration (sensorData *sensorDat);

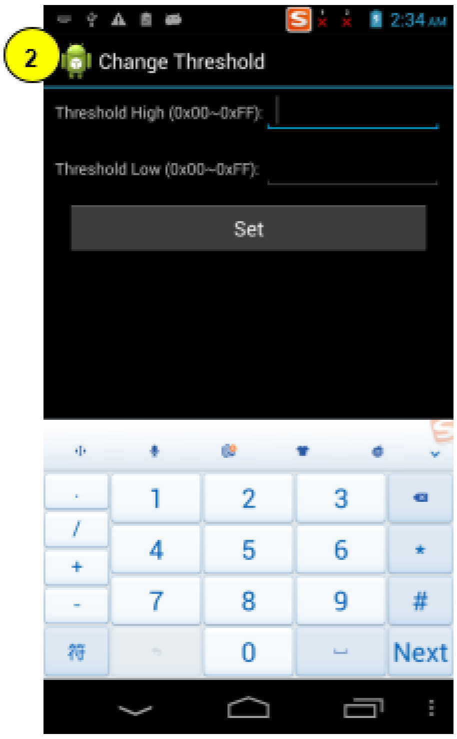
- ✓ 返回值为 0 或非 0，0：成功；非 0：失败
- ✓ 传入参数为回传的校准数据用于显示在 UI 界面上面,x=sensorDat->data[0], y=sensorDat->data[1], z=sensorDat->data[2]
- ✓ 当 gyroscope_start_static_calibration 返回 true 时方可调用此 api 获取静态校准数据

3.4.3 Proximity 门限值校准接口

Proximity 门限值校准的 API 调用位置同 Accel 和 Gyro。
Demo 范例：工模进入方式，Phone Call Display + “*#*#3646633#*#*”进入 engineer mode
当遇到问题手机时，使用标号 1 中的界面进行校准修正问题，标号 2 中的界面用于手动设定 threshold，



PS: 0
Threshold High: 45
Threshold Low: 42
Put phone on a horizontal plane and click the button.
NOTE: Donnot cover P-Sensor after button clicked.
Min value: 1 | Calculate min value
Cover P-Sensor then click the button.
Max value:1 | Calculate max value
Do calibration
Clear calibration
Result:



Threshold High (0x00~0xFF):
Threshold Low (0x00~0xFF):

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

int get_psensor_data(void)

✓ 返回 psensor 的 raw data

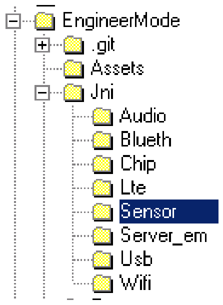
Int set_psensor_threshold(int high, int low)

✓ 可以将 high 和 low 两个数值保存在 nvram，并且传送给底层 driver 中

Native 层获取 psensor 接近远离状态，使用 Android 标准 native API 即可

3.4.4 工模代码范例（*#*#3646633#*#*）

- Native Layer interface 使用范例 ：使用 JIN 封装提供给 APP 使用



```
static jint calculatePsensorMaxValue(JNIEnv *, jclass) {
    ALOGD("Enter calculatePsensorMaxValue()\n");
    int ret = calculate_psensor_max_value();
    ALOGD("calculatePsensorMaxValue() returned %d\n", ret);

    return ret;
}
```

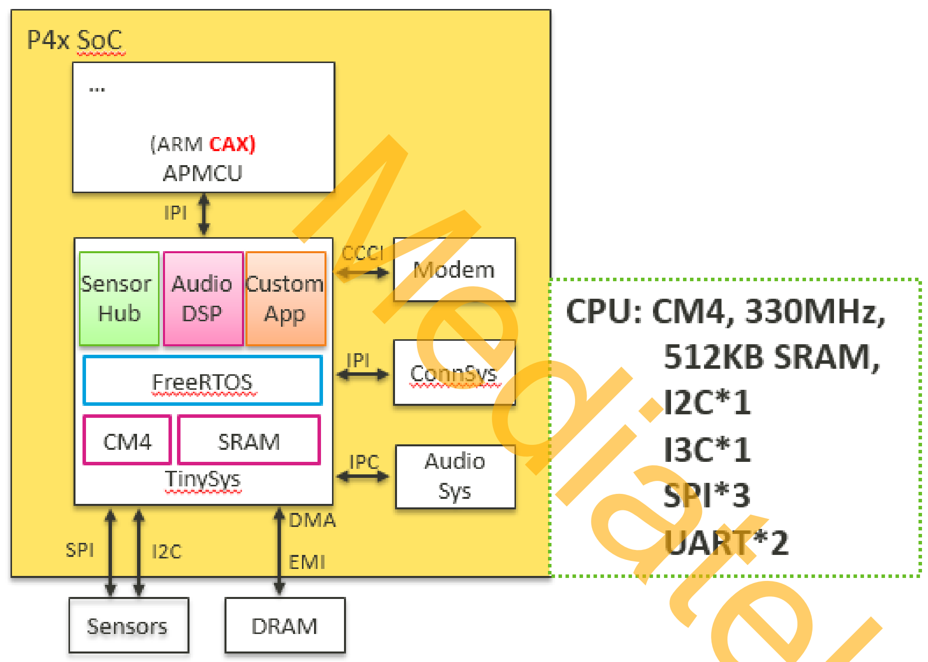
参考路径：vendor\mediatek\proprietary\external\apps\engineerMode

3.5 Device tree introduction

4 SCP Introduction

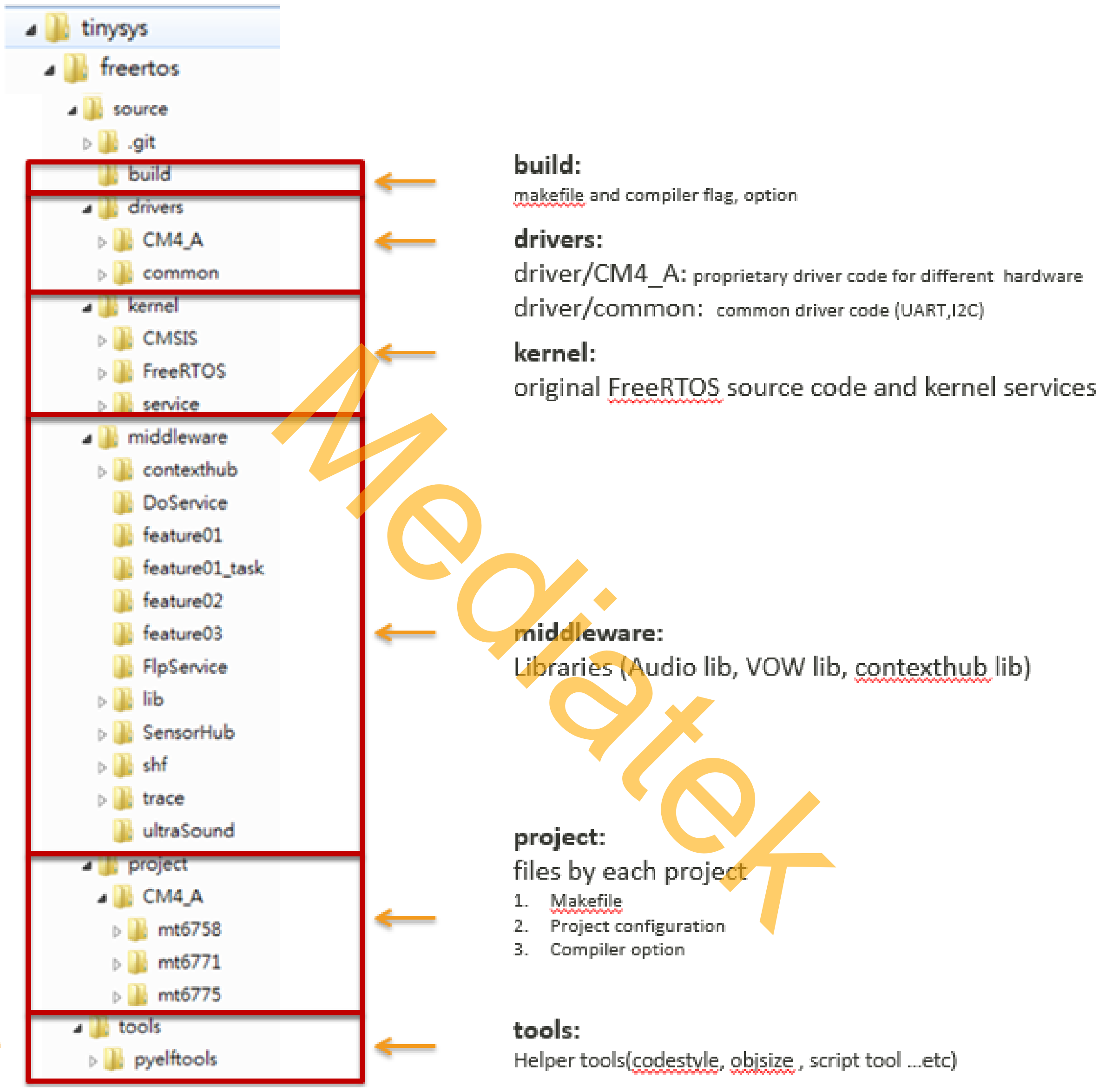
SCP （Tinysys）协处理器，负责 sensor ，audio 的相关 feature，以及可以扩客户私有的 feature。MTK SCP 的 系统选用的是 FreeRTOS，其中 CHRE 是 FreeRTOS 的一个 Task 专门处理 Sensor 相关数据。Audio feature 直接基于 FreeRTOS 进行实作

HW Architecture



4.1 Tinysys introduction

4.1.1 Folder Structure



build:
makefile and compiler flag, option

drivers:
driver/CM4_A: proprietary driver code for different hardware
driver/common: common driver code (UART,I2C)

kernel:
original FreeRTOS source code and kernel services

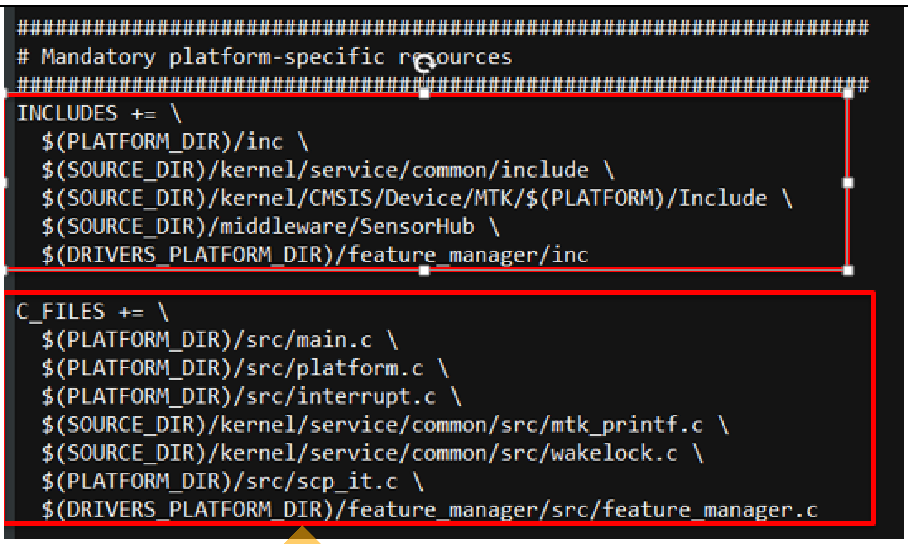
middleware:
Libraries (Audio lib, VOW lib, contexthub lib)

project:
files by each project
1. Makefile
2. Project configuration
3. Compiler option

tools:
Helper tools(codestyle, objsize , script tool …etc)

4.1.2 Configuration files

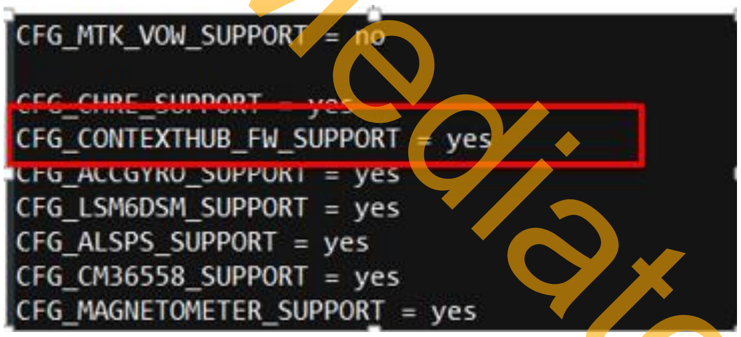
1. **Platform Configuration file**
   Required LDFLAGS, headers or C files of the platform
   Default configurations of the platform
   路径：**project/$(PROCESSOR)/$(PLATFORM)/platform/platform.mk**
   如下图：



```
######################################################################
# 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

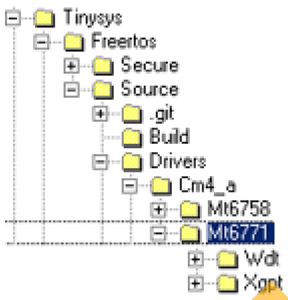


```
CFG_MTK_VOW_SUPPORT = no

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 如何在 Tinysis 下添加一新 freeRTOS driver

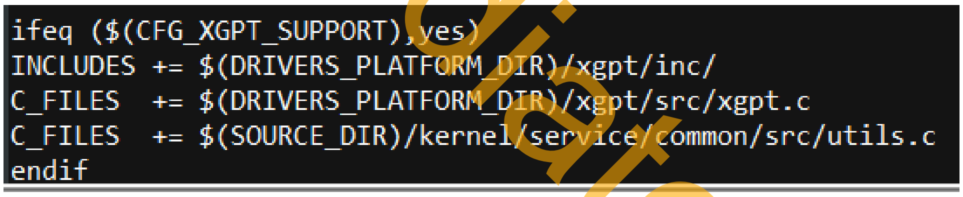
1 put the code in the appropriate folder
For driver code, put your file at following path:
drivers/$(PROCESSOR)/$(PLATFORM)/(New_Driver_Folder)

- Example: drivers/CM4_A/mt6771/XGPT



2. add a new compiler option in platform.mk
project/$(PROCESSOR)/$(PLATFORM)/platform/platform.mk

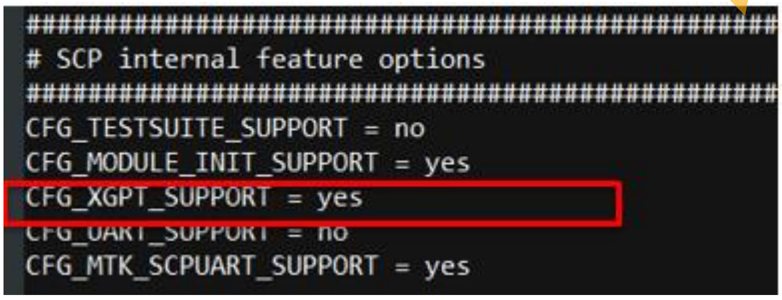
- example：project/mt6771/platform/platform.mk



```
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

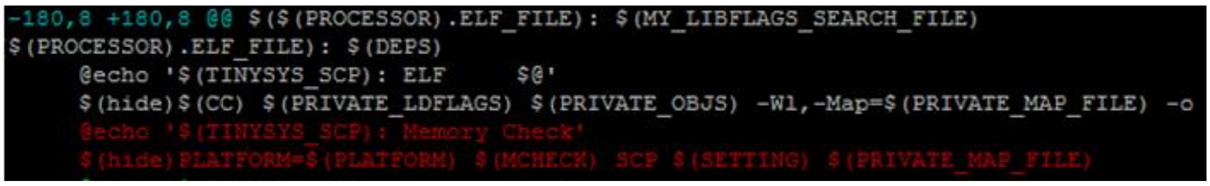


```
##################################################
# SCP internal feature options
##################################################
CFG_TESTSUITE_SUPPORT = no
CFG_MODULE_INIT_SUPPORT = yes
CFG_XGPT_SUPPORT = yes
CFG_UART_SUPPORT = no
CFG_MTK_SCPUART_SUPPORT = yes
```

4.1.4 SCP code size 限制机制

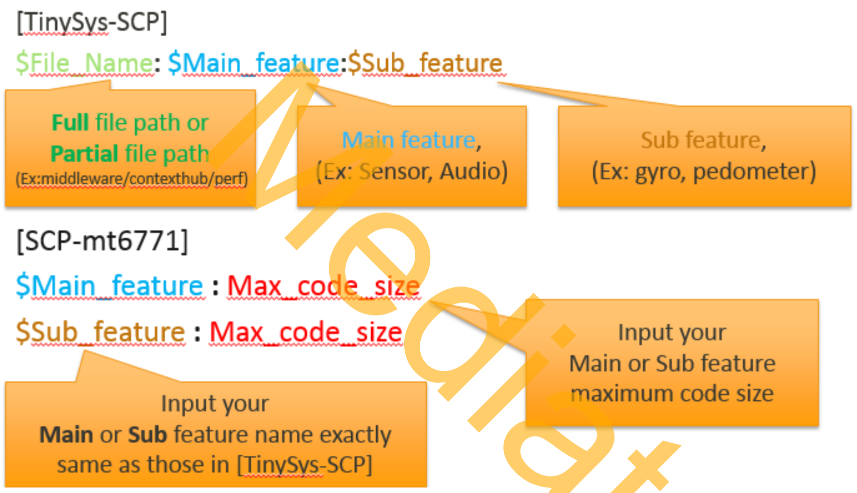
1. memoryReport.py is a script which use to limit code size at the build time.
If code size over your settings, it will cause build errors.
(script: vendor/mediatek/proprietary/tinysys/freertos/source/tools/**memoryReport.py**)
2. This script is hooked by tinysys scp make file:

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 格式说明**

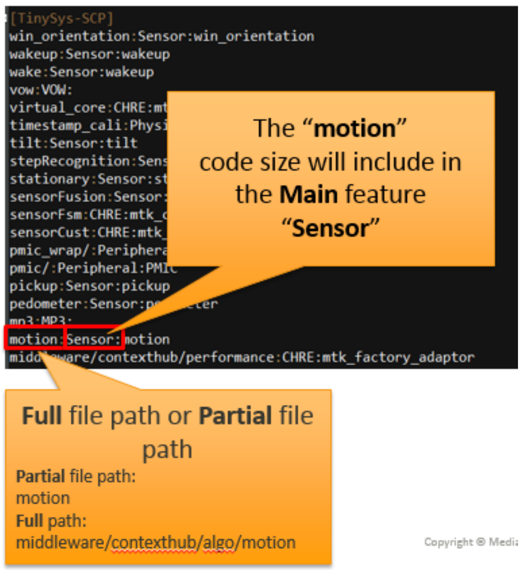


[TinySys-SCP]
$File_Name: $Main_feature:$Sub_feature

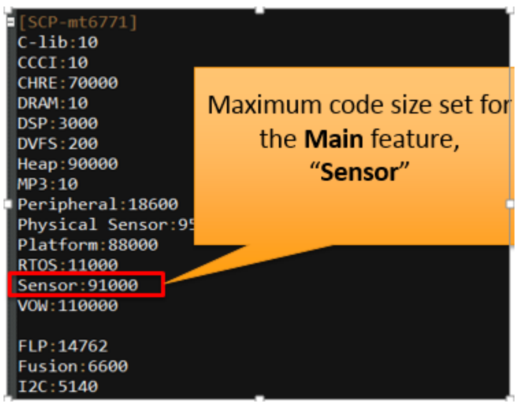
[SCP-mt6771]
$Main_feature : Max_code_size
$Sub_feature : Max_code_size

Input your
Main or **Sub** feature name exactly
same as those in [TinySys-SCP]
Input your
Main or Sub feature
maximum code size

举例说明：

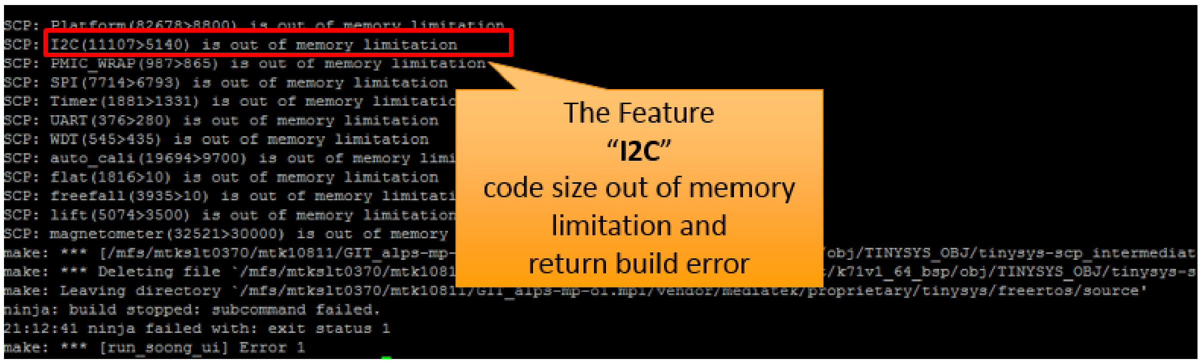


```
[TinySys-SCP]
win_orientation:Sensor:win_orientation
wakeup:Sensor:wakeup
wake:Sensor:wakeup
vow:VOW:
virtual_core:CHRE:mt
timestamp_cali:Physi
tilt:Sensor:tilt
stepRecognition:Sens
stationary:Sensor:st
sensorFusion:Sensor:
sensorFsm:CHRE:mtk_c
sensorCust:CHRE:mtk_
pmic_wrap/:Periphera
pmic/:Peripheral:PMIC
pickup:Sensor:pickup
pedometer:Sensor:pedometer
mp3:MP3:
motion:Sensor:motion
middleware/contexthub/performance:CHRE:mtk_factory_adaptor
```



```
[SCP-mt6771]
C-lib:10
CCCI:10
CHRE:70000
DRAM:10
DSP:3000
DVFS:200
Heap:90000
MP3:10
Peripheral:18600
Physical Sensor:95
Platform:88000
RTOS:11000
Sensor:91000
VOW:110000

FLP:14762
Fusion:6600
I2C:5140
```

如果 size 大小超过设定限制会出现如下 build error 提醒：



```
SCP: Platform(82678>8800) is out of memory limitation
SCP: I2C(11107>5140) is out of memory limitation
SCP: PMIC_WRAP(987>865) is out of memory limitation
SCP: SPI(7714>6793) is out of memory limitation
SCP: Timer(1881>1331) is out of memory limitatio
SCP: UART(376>280) is out of memory limitation
SCP: WDT(545>435) is out of memory limitation
SCP: auto_cali(19694>9700) is out of memory limi
SCP: flat(1816>10) is out of memory limitation
SCP: freefall(3935>10) is out of memory limitati
SCP: lift(5074>3500) is out of memory limitation
SCP: magnetometer(32521>30000) is out of memory
make: *** [/mfs/mtkslt0370/mtk10811/GIT_alps-mp-                                 /obj/TINYSYS_OBJ/tinysys-scp_intermediat
make: *** Deleting file `/mfs/mtkslt0370/mtk1081                                 :/k71v1_64_bsp/obj/TINYSYS_OBJ/tinysys-s
make: Leaving directory `/mfs/mtkslt0370/mtk10811/GIT_alps-mp-o1.mp1/vendor/mediatek/proprietary/tinysys/freertos/source'
ninja: build stopped: subcommand failed.
21:12:41 ninja failed with: exit status 1
make: *** [run_soong_ui] Error 1
```

5 CHRE sensors introduction

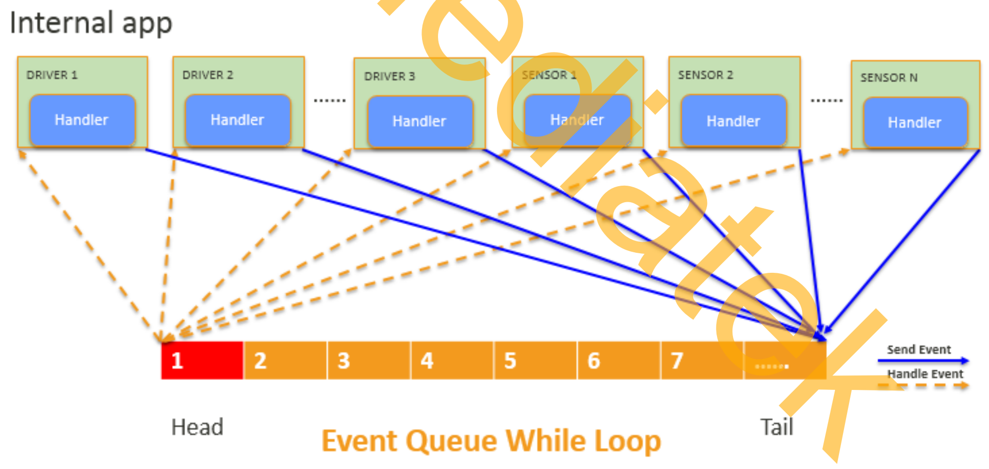
5.1 CHRE 简述

SCP 下面的，MTK sensor hub feature 基于 Google CHRE 架构实现.
CHRE (Context Hub Runtime Environment) 是一个事件驱动的架构，也可以作为独立的 OS。

黄色部分是 Event Queue，CHRE 只有一个 while 去处理排在头部的 Event Queue，这里没有优先级的概念，Event Queue 的处理是先来先服务，只有 interrupt 可以打断当前的 Event Queue 处理。CHRE 默认有 512 个 Event Queue。设计的目的是实时，且轻量级，所以 EventQueue 内部代码必须很快跑完。

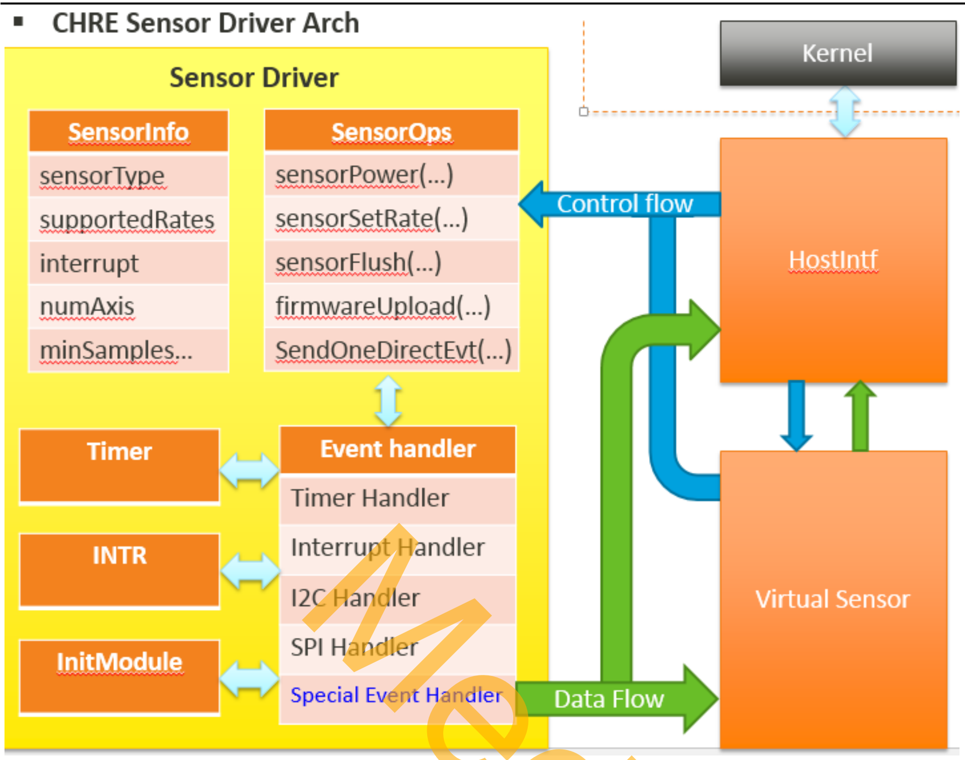
CHRE 内部实现的 driver，即用来处理事件的代码 Google 称之为 nano hub app。后面讲详细解释如何写一个 nano hub app

CHRE 消息机制简要做如下图示



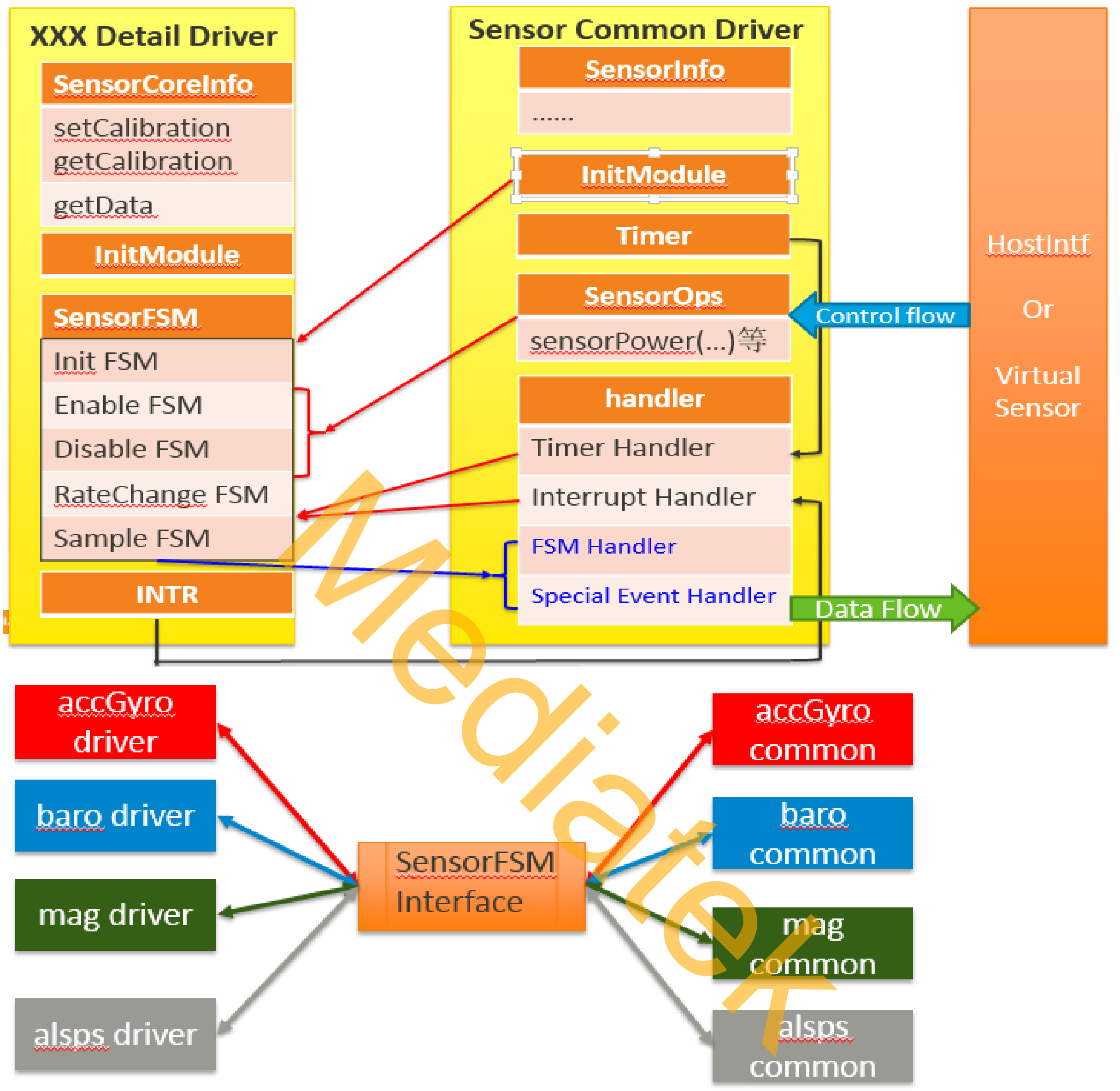
5.2 MTK CHRE Sensors Common Layer

CHRE 事件驱动的方式，实现一个 app 较为复杂，coding 代码不好理解。实现一个原生 CHRE app 架构如下：



App 需要实现 hostintf （hostintf 提供类似 AP 端 SensorManager，SensorService 功能，及 HAL 层接口）开出来的接口，对接好 control flow 和 data flow。编程人员必须熟悉 hostintf 内部流程。

为避免客户 porting 时间过长，已经出现 bug 不好查找，MTK 将物理 sensor 的逻辑部分抽出了单独的一层叫做 sensorFSM。和硬件相关的部分单独写成一份代码，提供给 sensorFSM 调用。框架如下：



拆出 common 层后，客户需要实现的是：

- Init 相关（加载客制化，auto detect 等)
- 实现有限个 sensorFSM
- 实现 SensorCoreInfo

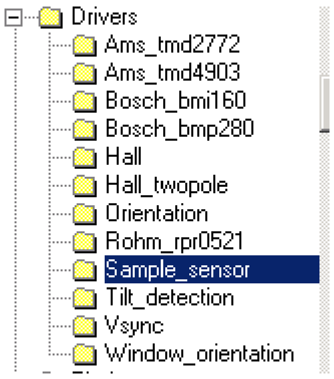
5.3 如何写一个 CHRE APP

MTK 提供的虚拟 sensor 都是按照标准 CHRE APP 的写法实现。客户可以标准 CHRE APP 的写法实现自己的虚拟 sensor 或者其他 sensor 相关用途的 APP。下面介绍一个例子，来讲述如何完成一个自定义的 APP。

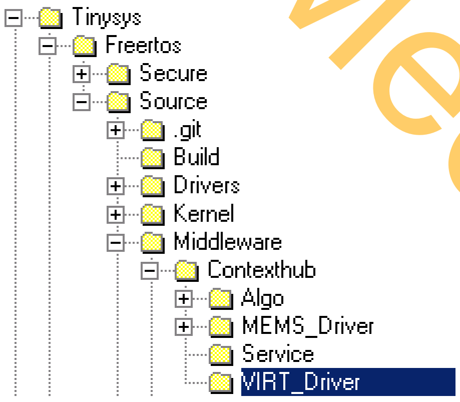
本例子是实现一个自定义的 sensor type flat，这个 type 从来检测手机是否是水平放置。

5.3.1 Copy 一份 demo driver 实现 CHRE APP 框架

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


Drivers
Ams_tmd2772
Ams_tmd4903
Bosch_bmi160
Bosch_bmp280
Hall
Hall_twopole
Orientation
Rohm_rpr0521
Sample_sensor
Tilt_detection
Vsync
Window_orientation

Copy 到如下路径：
vendor\mediatek\proprietary\tinysis\freertos\souce\middleware\virt_driver\


Tinysys
Freertos
Secure
Source
.git
Build
Drivers
Kernel
Middleware
Contexthub
Algo
MEMS_Driver
Service
VIRT_Driver

5.3.2 编译选项添加

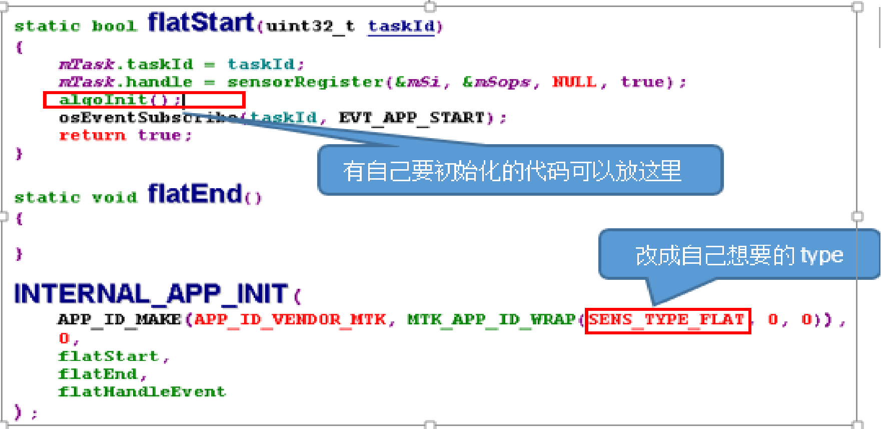
project/CM4_a/mt6771/platform/feature_config/chre.mk

```
######## 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 修改注意点

做如下框架内的调整：

1. 初始化部分，让 CHRE APP 运行起来



```
static bool flatStart(uint32_t taskId)
{
    mTask.taskId = taskId;
    mTask.handle = sensorRegister(&mSi, &mSops, NULL, true);
    algoInit();
    osEventSubscribe(taskId, EVT_APP_START);
    return true;
}

static void flatEnd()
{

}

INTERNAL_APP_INIT(
    APP_ID_MAKE(APP_ID_VENDOR_MTK, MTK_APP_ID_WRAP(SENS_TYPE_FLAT, 0, 0)),
    0,
    flatStart,
    flatEnd,
    flatHandleEvent
);
```

有自己要初始化的代码可以放这里

改成自己想要的 type

2. 实现 APP 内的全局结构体，并注册一个 sensor 类型

```
static const struct SensorInfo mSi = {
    .sensorName = "Flat",
    .sensorType = SENS_TYPE_FLAT,
    .numAxis = NUM_AXIS_EMBEDDED,
    .interrupt = NANOHUB_INT_WAKEUP,
    .minSamples = 20
};

static const struct SensorOps mSops = {
    .sensorPower = flatPower,
    .sensorFirmwareUpload = flatFirmwareUpload,
    .sensorSetRate = flatSetRate,
    .sensorFlush = flatFlush
};
```

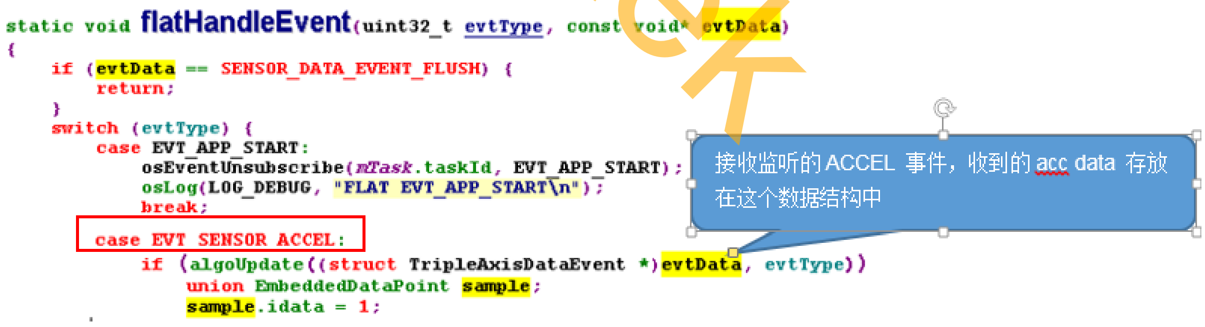
这些接口都要实现，至少要按照范例发一个 event，确保 CHRE 的 flow 可以跑下去

3. 订阅消息

Flat 需要订阅 ACC 的 raw data 来完成自己的算法

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

4. 接收订阅的消息

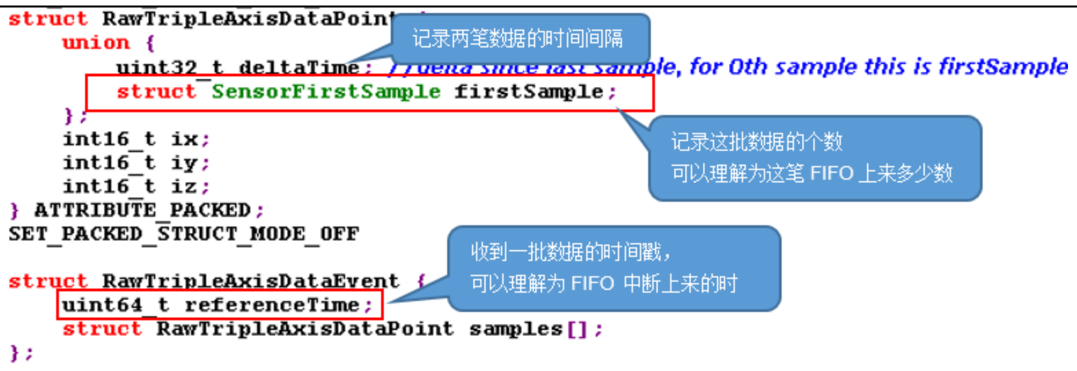


```
static void flatHandleEvent(uint32_t evtType, const void* evtData)
{
    if (evtData == SENSOR_DATA_EVENT_FLUSH) {
        return;
    }
    switch (evtType) {
        case EVT_APP_START:
            osEventUnsubscribe(mTask.taskId, EVT_APP_START);
            osLog(LOG_DEBUG, "FLAT EVT_APP_START\n");
            break;

        case EVT_SENSOR_ACCEL:
            if (algoUpdate((struct TripleAxisDataEvent *)evtData, evtType))
                union EmbeddedDataPoint sample;
                sample.idata = 1;
```

接收监听的 ACCEL 事件，收到的 acc data 存放在这个数据结构中

5. CHRE sensor 数据结构说明：

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



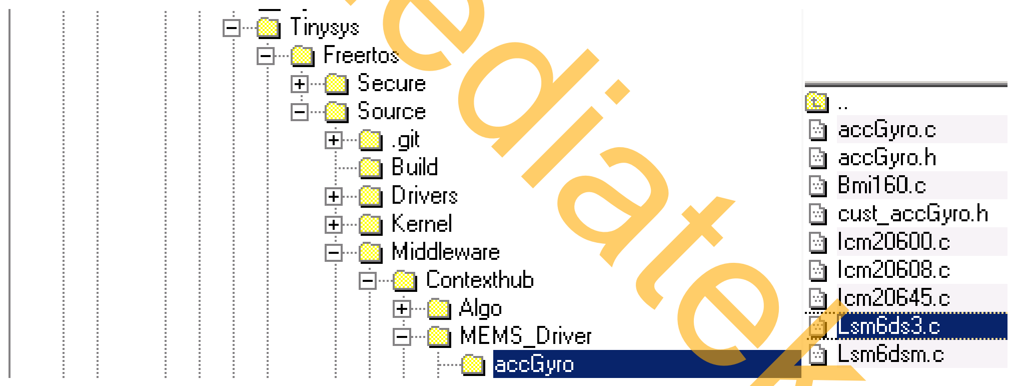
```
struct RawTripleAxisDataPoint {
    union {
        uint32_t deltaTime; //delta since last sample, for 0th sample this is firstSample
        struct SensorFirstSample firstSample;
    };
    int16_t ix;
    int16_t iy;
    int16_t iz;
} ATTRIBUTE_PACKED;
SET_PACKED_STRUCT_MODE_OFF

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

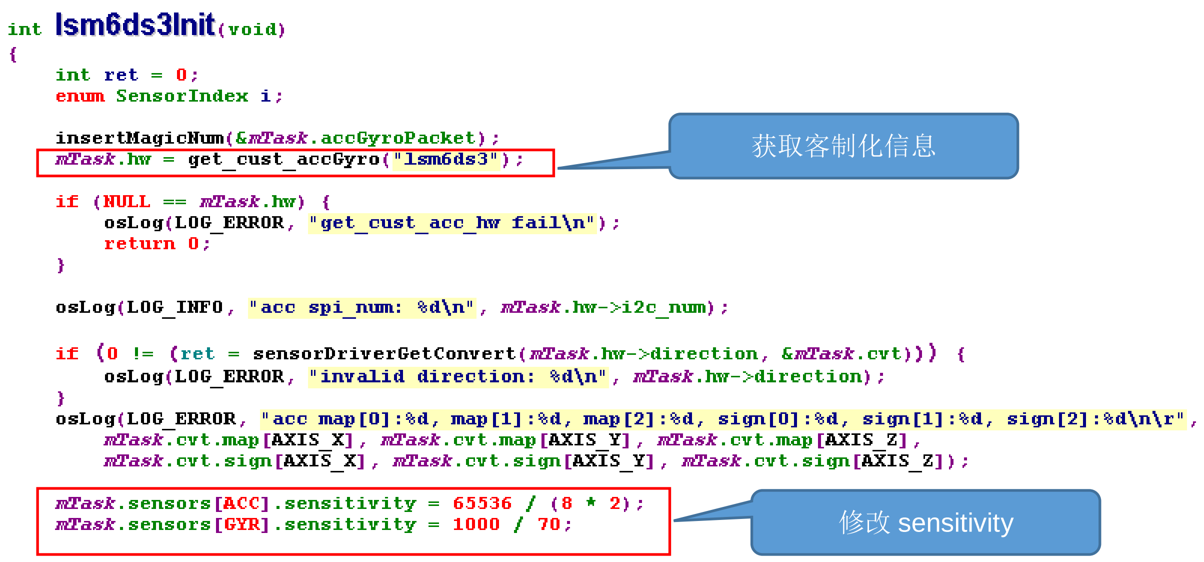
5.4 A+G driver porting guide

5.4.1 A+G initialization

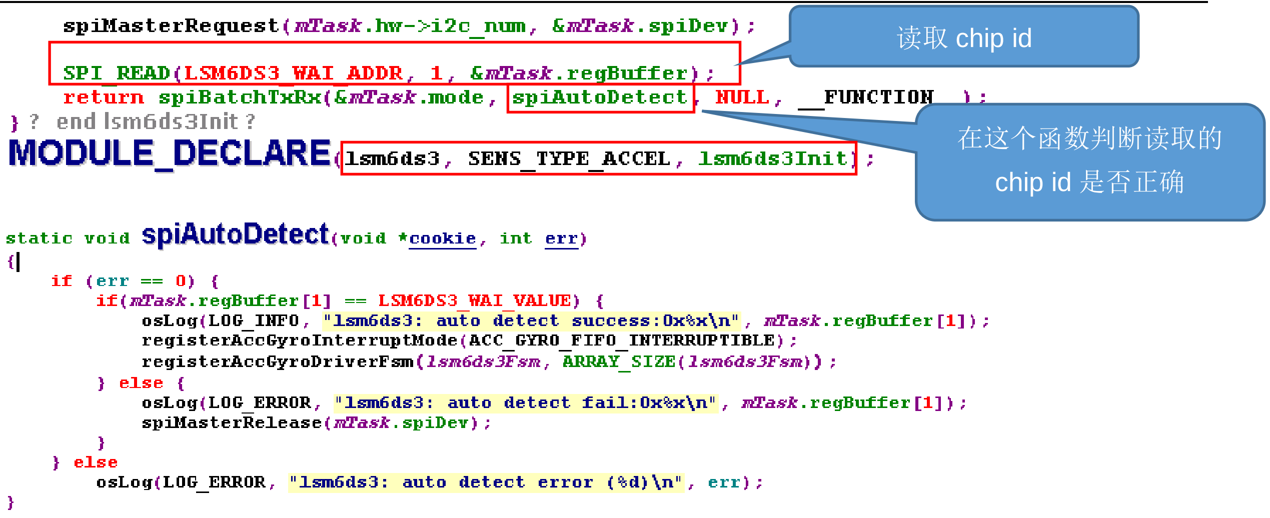
这里用 ST 的 driver 做个例子。

Copy 一份已经写好的 driver 进行修改，一定要基于同系列的进行 copy 改动会小。



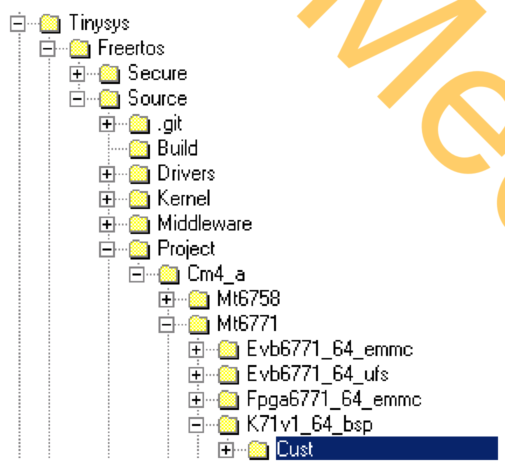


```
int lsm6ds3Init(void)
{
    int ret = 0;
    enum SensorIndex i;

    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;
```

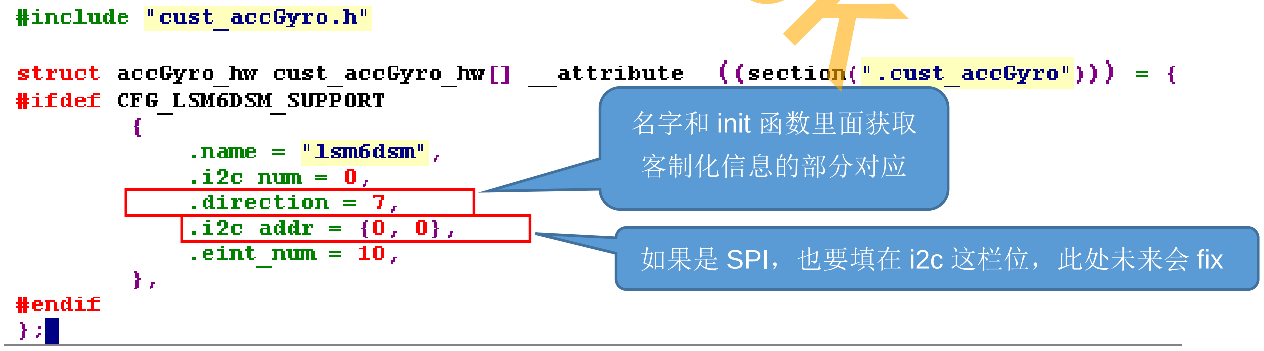


```
    spiMasterRequest(mTask.hw->i2c_num, &mTask.spiDev);

    SPI_READ(LSM6DS3_WAI_ADDR, 1, &mTask.regBuffer);
    return spiBatchTxRx(&mTask.mode, spiAutoDetect, NULL, __FUNCTION__);
} ? end lsm6ds3Init ?
MODULE_DECLARE(lsm6ds3, SENS_TYPE_ACCEL, lsm6ds3Init);

static void spiAutoDetect(void *cookie, int err)
{
    if (err == 0) {
        if(mTask.regBuffer[1] == LSM6DS3_WAI_VALUE) {
            osLog(LOG_INFO, "lsm6ds3: auto detect success:0x%x\n", mTask.regBuffer[1]);
            registerAccGyroInterruptMode(ACC_GYRO_FIFO_INTERRUPTIBLE);
            registerAccGyroDriverFsm(lsm6ds3Fsm, ARRAY_SIZE(lsm6ds3Fsm));
        } else {
            osLog(LOG_ERROR, "lsm6ds3: auto detect fail:0x%x\n", mTask.regBuffer[1]);
            spiMasterRelease(mTask.spiDev);
        }
    } else
        osLog(LOG_ERROR, "lsm6ds3: auto detect error (%d)\n", err);
}
```

客制化信息填写的位置：



```
Tinysys
  Freertos
    Secure
    Source
      .git
      Build
      Drivers
      Kernel
      Middleware
      Project
        Cm4_a
          Mt6758
          Mt6771
            Evb6771_64_emmc
            Evb6771_64_ufs
            Fpga6771_64_emmc
            K71v1_64_bsp
              Cust
```


名字和 init 函数里面获取客制化信息的部分对应
如果是 SPI，也要填在 i2c 这栏位，此处未来会 fix

```
#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
};
```

1. 构建 SensorFSM 数组

名词解释： 首尾事件

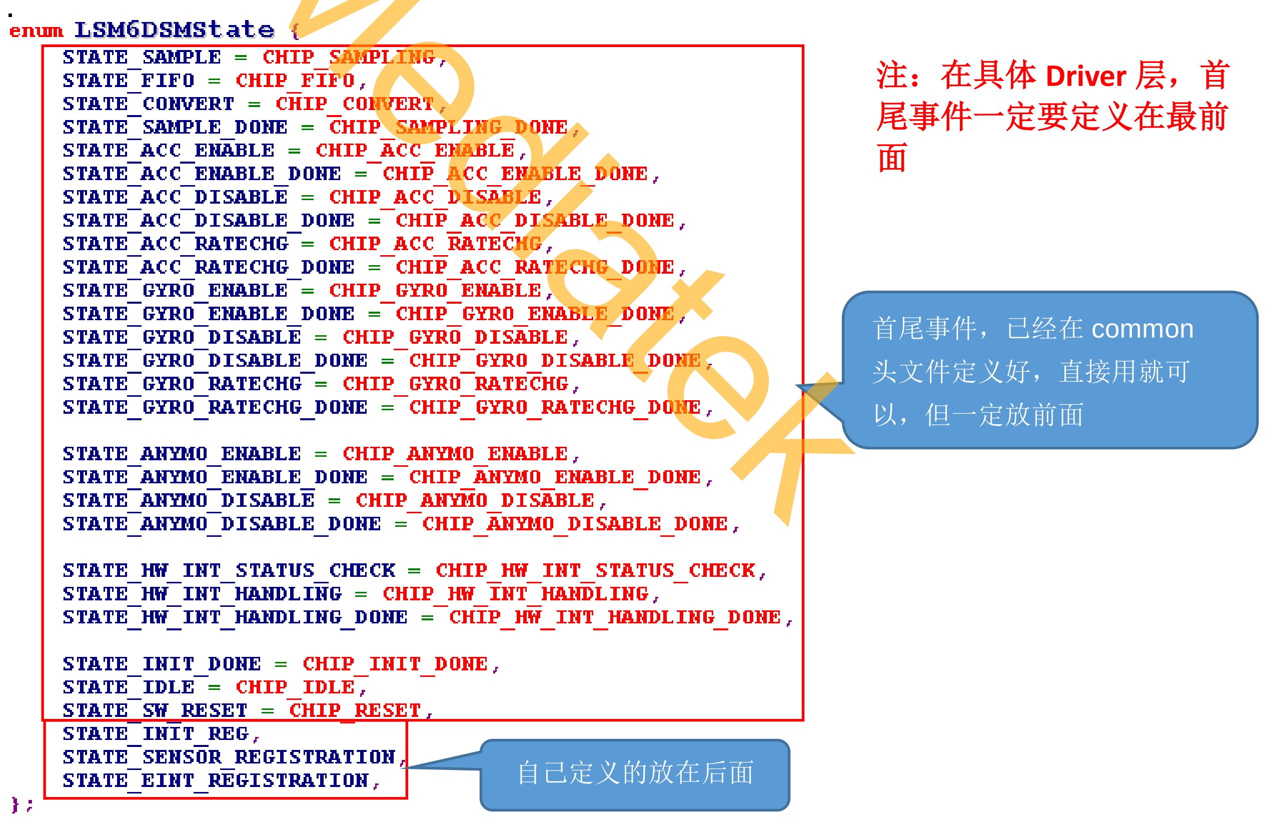
首事件，如 STAT_ENABLE，STAT_SAMPLE

尾事件，driver 在完成一个特定的动作时处理的最后一个 Event，例如，ENABLE_DONE，DISABLE_DONE，RATECHG_DONE, SAMPLE_DONE 等，这些 event 通常伴随着和上层的交互

首尾之间的事件，根据具体的 hw spec，按需求定义，一般按照一次 i2c/SPI transfer 一个 state 的方式定义。

如下面例子:

```
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),
};
```



```
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,
};
```

注：在具体 Driver 层，首尾事件一定要定义在最前面

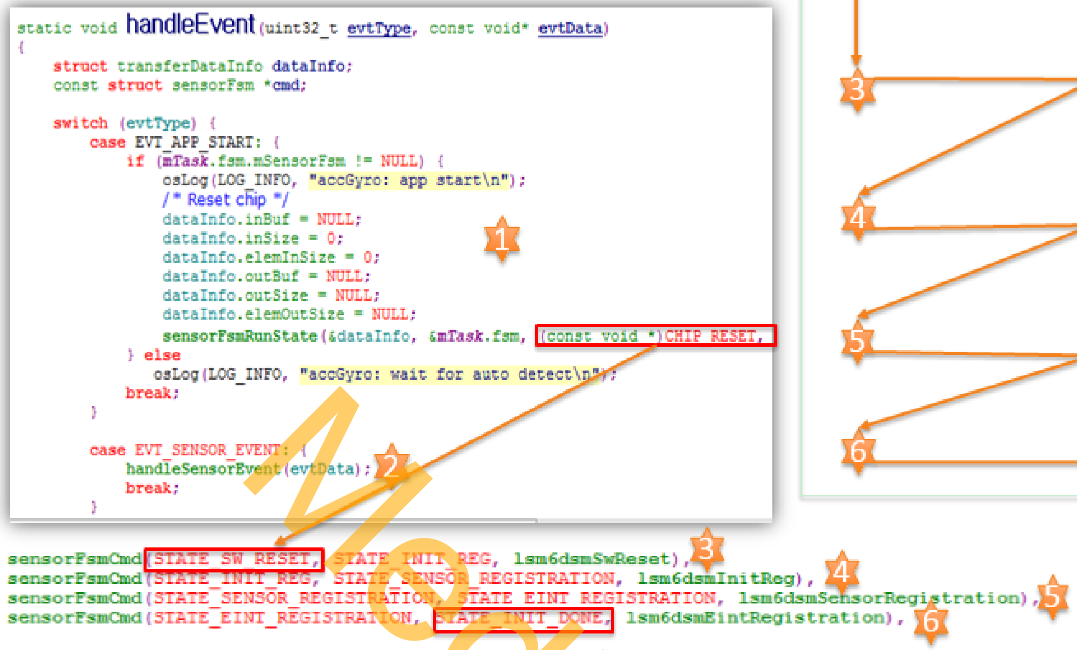
首尾事件，已经在 common 头文件定义好，直接用就可以，但一定放前面

自己定义的放在后面

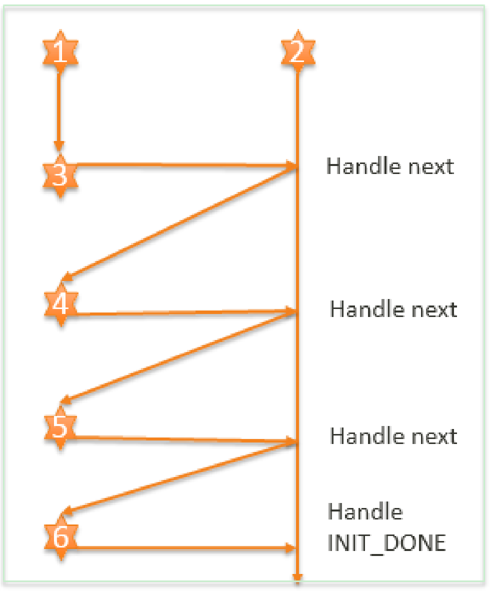
如下对一次 Init 操作流程做解释

FSM Mechanism introduction :

Then accGyro app receive event and handle it



```
static void handleEvent(uint32_t evtType, const void* evtData)
{
    struct transferDataInfo dataInfo;
    const struct sensorFsm *cmd;

    switch (evtType) {
        case EVT_APP_START: {
            if (mTask.fsm.mSensorFsm != NULL) {
                osLog(LOG_INFO, "accGyro: app start\n");
                /* Reset chip */
                dataInfo.inBuf = NULL;
                dataInfo.inSize = 0;
                dataInfo.elemInSize = 0;
                dataInfo.outBuf = NULL;
                dataInfo.outSize = NULL;
                dataInfo.elemOutSize = NULL;
                sensorFsmRunState(&dataInfo, &mTask.fsm, (const void *)CHIP_RESET,
            } else
               osLog(LOG_INFO, "accGyro: wait for auto detect\n");
            break;
        }

        case EVT_SENSOR_EVENT: {
            handleSensorEvent(evtData);
            break;
        }

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),
```



红框处是common层开始和结束fsm所用的state，driver function执行从3->6

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),
```

注意： 如果有 FIFO 开关的动作，一定要在打开 FIFO 后立刻调用如下函数，这个函数会进行 FIFO 数据时间戳的校准

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

根据不同厂商 data sheet 自行晚上要实现的函数即可。

5.4.3 Report rate

A+G 二合一 driver rate 的设定要注意，如果 Acc 和 Gyro 都打开了，频率要保持一致。原因参考 FIFO 配置章节。

```
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)
{
```

将传下来的 rate 转换成一个硬件支持的最接近的 report rate。

```
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;
```

将设定的先缓存下来，真正要设定 rate 还需要继续计算。如，acc 是否已经打开过了？

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

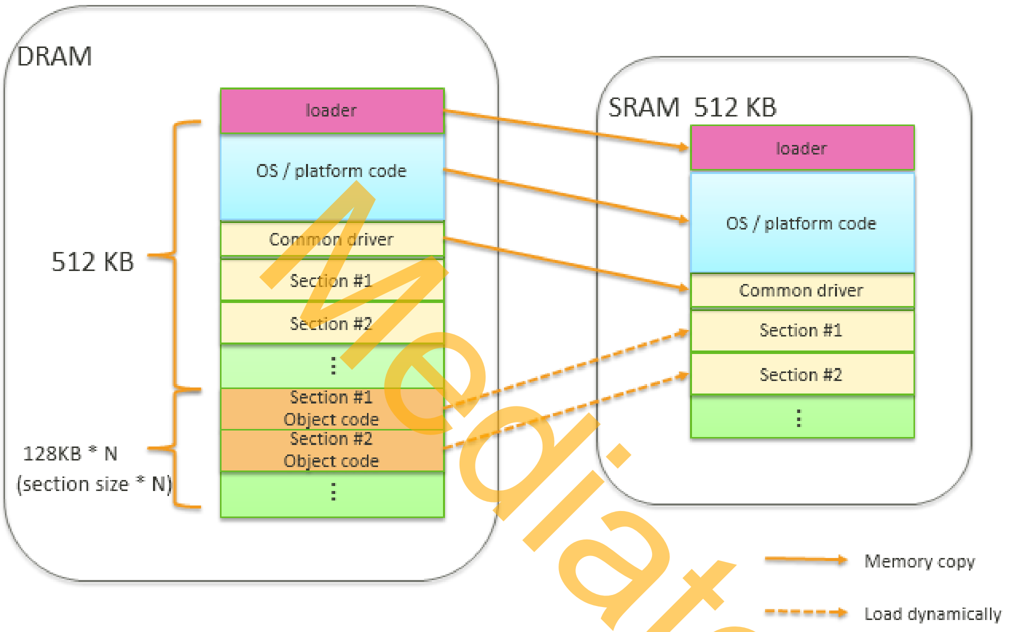
开了 gyro 要做同频处理

Acc rate 和当前不同才需要重新设定 HW 的 ODR

5.5 Sensor driver overlay

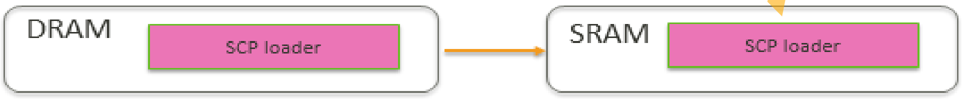
Purpose: 客户开案需要二供料件。 同一个类型的 sensor 会选择两家厂商的供货，如果同时在 SCP 写两个 driver 进行 auto detect 会占用 SCP 的 SRAM，因此选择将 driver 放在 DRAM 中，开机启动 SCP 时进行 load。

- Load overlay scp image : emmc -> dram - > sram

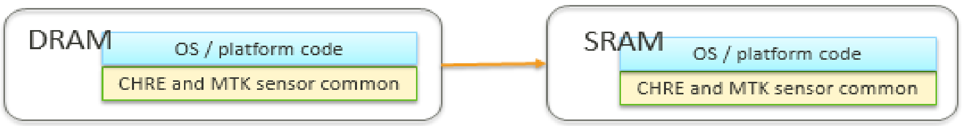


Overlay load flow

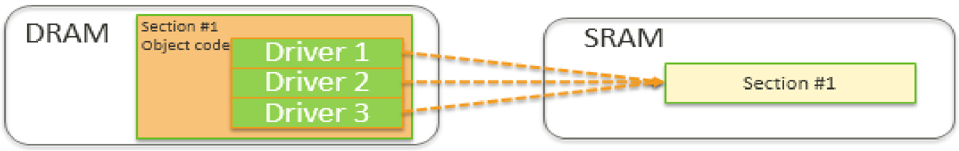
1) Memory copy loader code from dram to SRAM, then SCP run loader



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



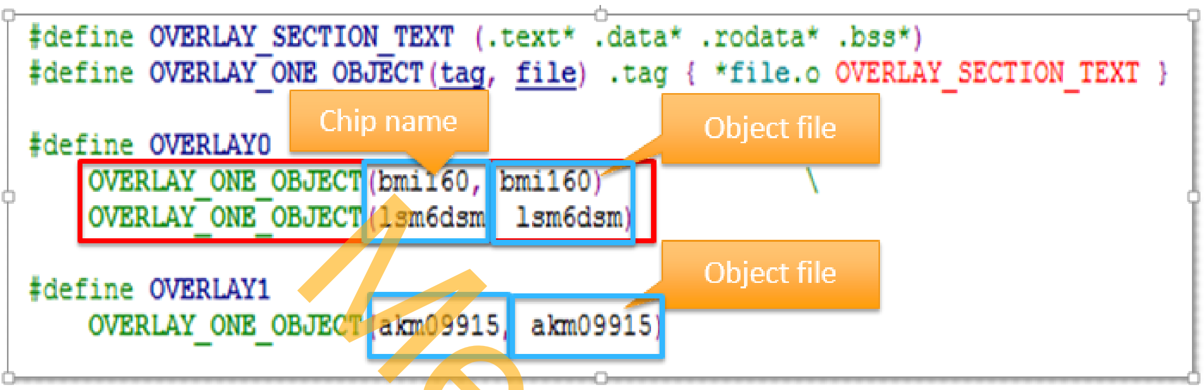
One section represent one sensor type ,may have multiple drivers (object code)

5.5.1 如何添加一个 overlay driver

1) ADD object in linker script

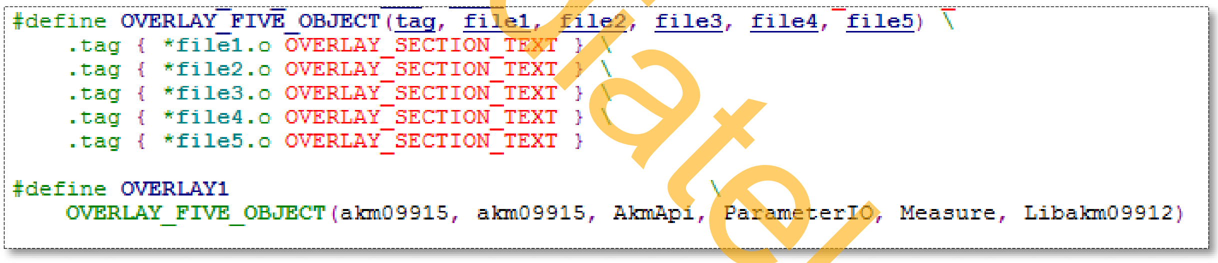
vendor\mediatek\proprietary\tinysys\freertos\source\project\CM4_A\#PLATFORM\$PROJECT\inc\overlay_sensor.h

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.



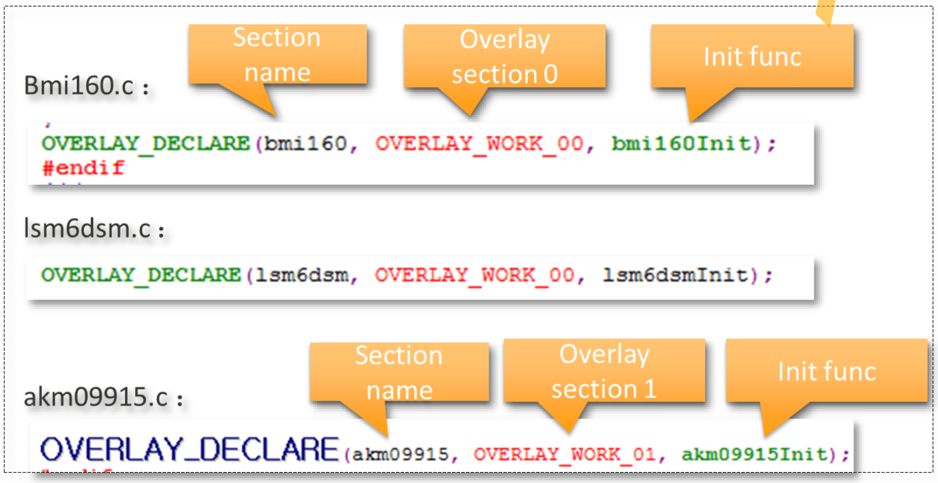
```
#define OVERLAY_SECTION_TEXT (.text* .data* .rodata* .bss*)
#define OVERLAY_ONE_OBJECT(tag, file) .tag { *file.o OVERLAY_SECTION_TEXT }

#define OVERLAY0
    OVERLAY_ONE_OBJECT(bmi160, bmi160)           \
    OVERLAY_ONE_OBJECT(lsm6dsm, lsm6dsm)

#define OVERLAY1
    OVERLAY_ONE_OBJECT(akm09915, akm09915)
```

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 (在具体 driver 中添加)



Bmi160.c :

```
OVERLAY_DECLARE(bmi160, OVERLAY_WORK_00, bmi160Init);
#endif
```

lsm6dsm.c :

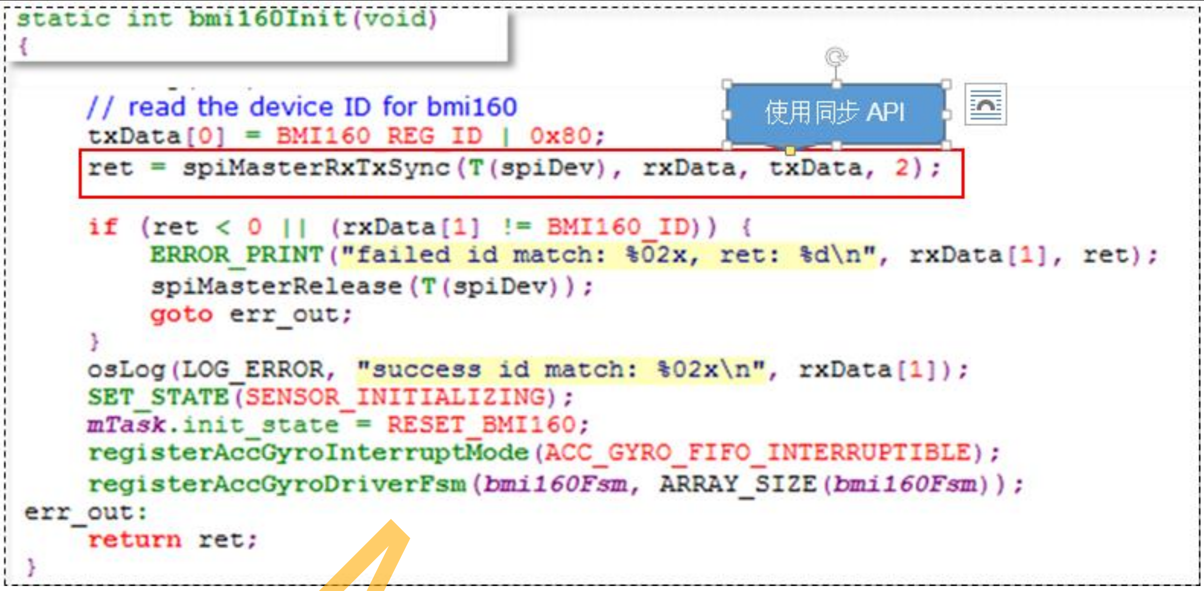
```
OVERLAY_DECLARE(lsm6dsm, OVERLAY_WORK_00, lsm6dsmInit);
```

akm09915.c :

```
OVERLAY_DECLARE(akm09915, OVERLAY_WORK_01, akm09915Init);
```

3) Driver 中要使用同步 SPI or i2c API 进行 init 的操作 (driver)

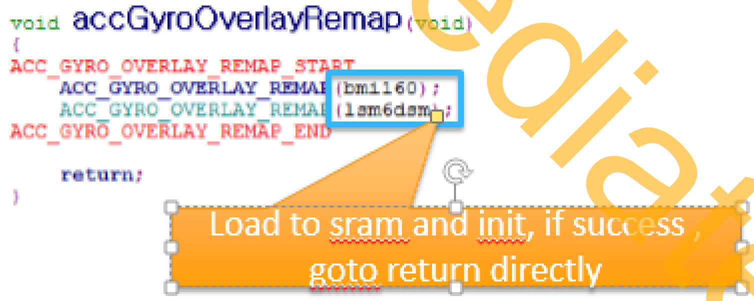
vendor\mediatek\proprietary\tinysys\freertos\source\middleware\contexthub\MEMS_Driver\



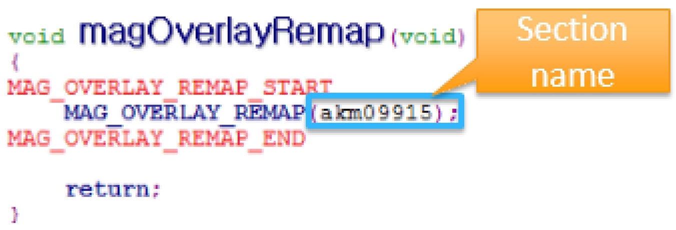
```
static int bmi160Init(void)
{

    // 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;
}
```

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

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



```
void accGyroOverlayRemap(void)
{
ACC_GYRO_OVERLAY_REMAP_START
    ACC_GYRO_OVERLAY_REMAP(bmi160);
    ACC_GYRO_OVERLAY_REMAP(lsm6dsm);
ACC_GYRO_OVERLAY_REMAP_END

    return;
}
```



```
void magOverlayRemap(void)
{
MAG_OVERLAY_REMAP_START
    MAG_OVERLAY_REMAP(akm09915);
MAG_OVERLAY_REMAP_END

    return;
}
```

5) 打开 overlay 的 feature 开关

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 使用

5.6.1 I2C API 说明

1） 标准 API

申请 i2c transfer，Call I2C transfer API 之前，先 call 此 API，但通常只在 user init 函数

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

释放 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

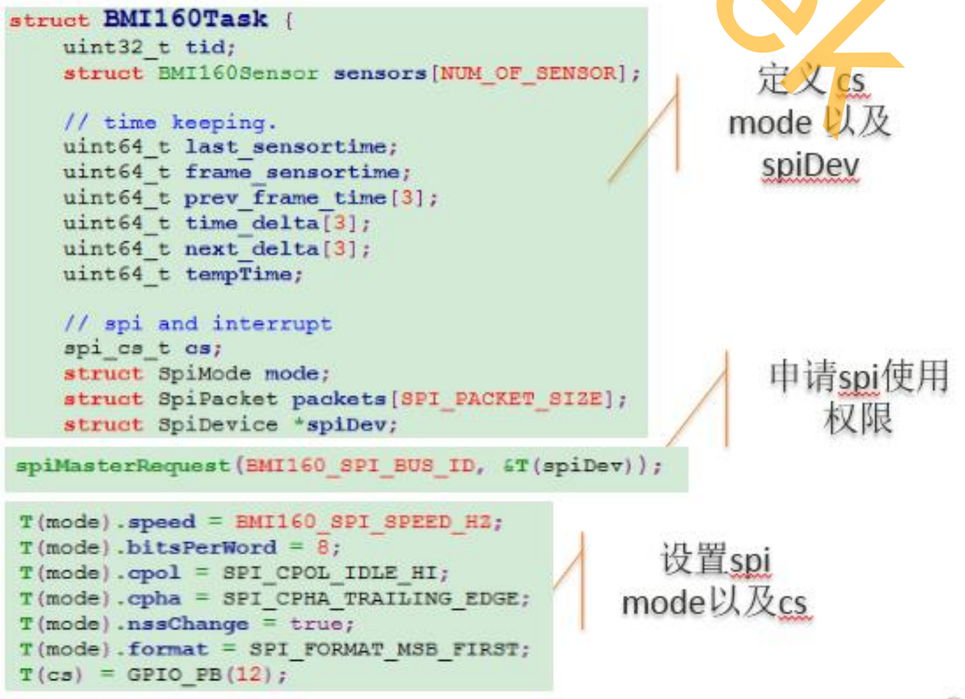
```
int i2cMasterTxRx(uint32_t busId, uint32_t addr, const void *txBuf, size_t txSize,
        void *rxBuf, size_t rxSize, I2cCallbackF callback, void *cookie);
```

2） MTK 自定义串行 API

用于 sensor overlay

5.6.2 SPI API

初始化配置



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

    // 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;
```

```
spiMasterRequest(BMI160_SPI_BUS_ID, &T(spiDev));
```

```
T(mode).speed = BMI160_SPI_SPEED_HZ;
T(mode).bitsPerWord = 8;
T(mode).cpol = SPI_CPOL_IDLE_HI;
T(mode).cpha = SPI_CPHA_TRAILING_EDGE;
T(mode).nssChange = true;
T(mode).format = SPI_FORMAT_MSB_FIRST;
T(cs) = GPIO_PB(12);
```

Write and read

```
// 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

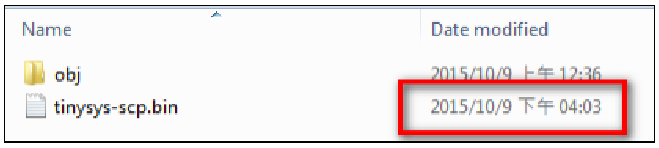
Module Build

$ 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:

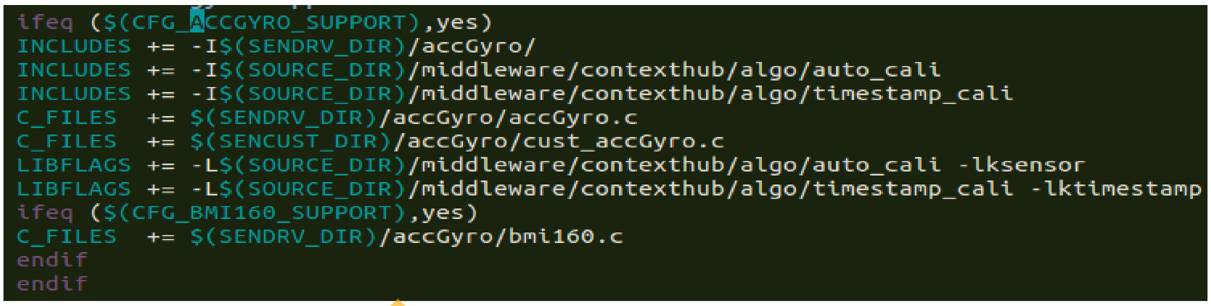


```
Name                      Date modified
obj                       2015/10/9 上午 12:36
tinysys-scp.bin           2015/10/9 下午 04:03
```

6.3 How to build a sensor driver to binary

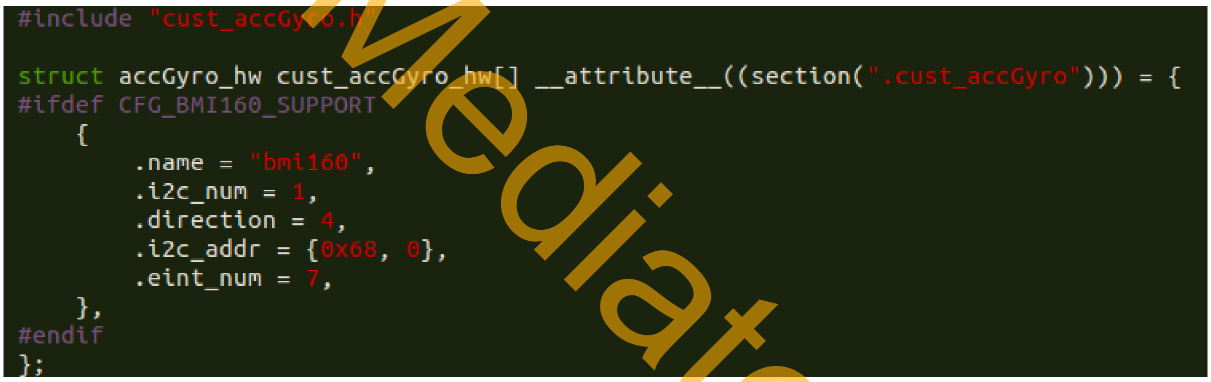
6.3.1 AccGyro

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
```

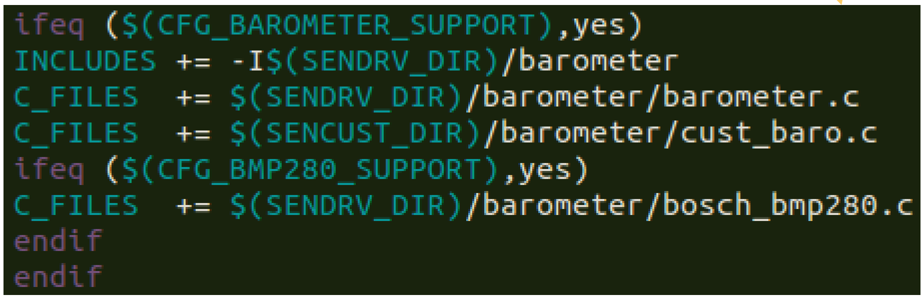
vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/{PROJECT}/cust/accGyro/cust_accGyro.c



```
#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
};
```

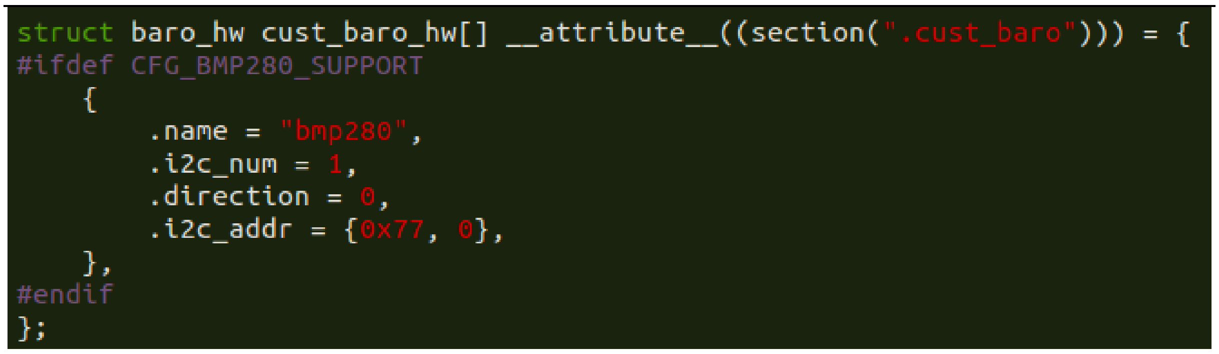
6.3.2 Barometer

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
```

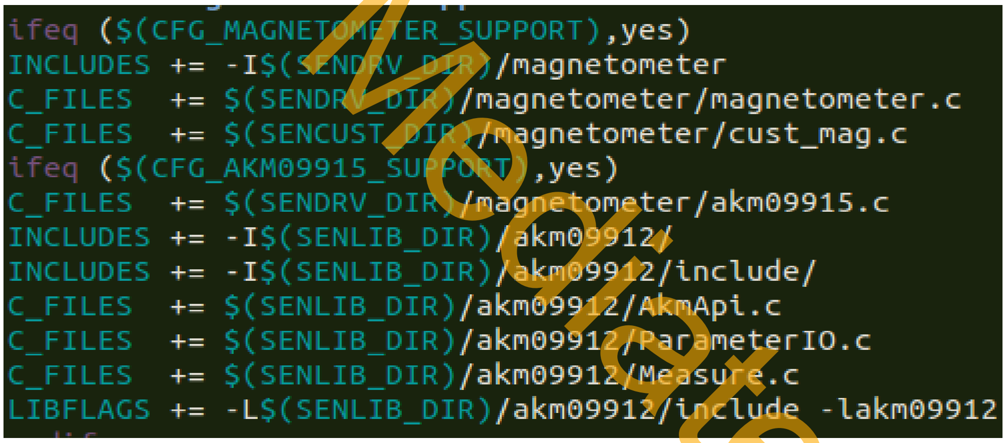
vendor/mediatek/proprietary/tinysys/freertos/source/project/CM4_A/mt6758/{PROJECT}/cust/barometer/cust_baro.c



```
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
};
```

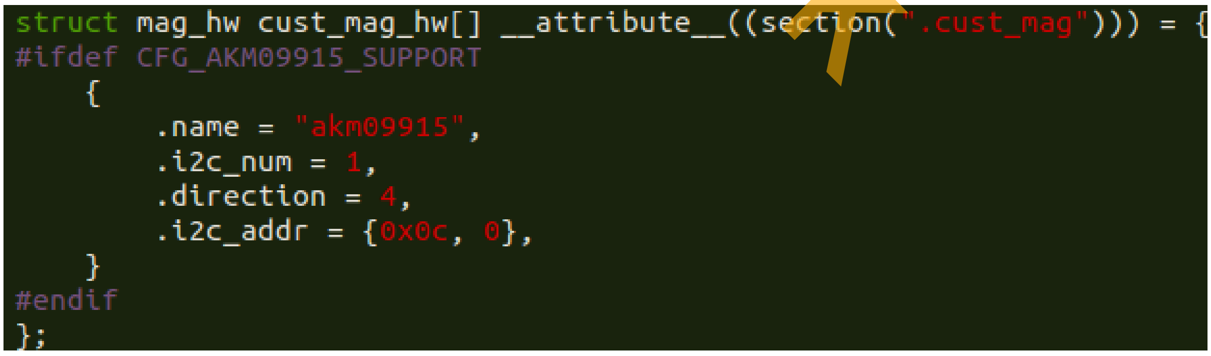
6.3.3 Magnetometer

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

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 打开 SCP 端虚拟 gyro（基于 AKM 的 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 打开 SCP 端 MTK fusion 算法（需要有物理 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     // 打开MTK自己的fusion算法
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 如何打开 SCP Uart

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

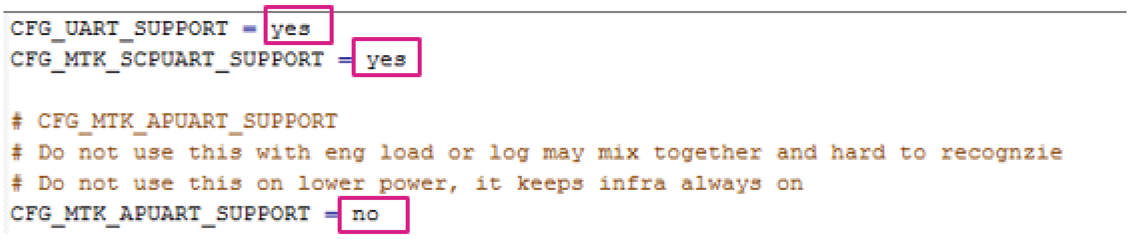
```
CFG_UART_SUPPORT = yes
```

注意，QA 测试时务必要关闭 uart，否则会有 performance 问题

6.5.2 SCP uart 如何复用 AP uart

Enable by modify config
project/CM4_A/mt6771/platform/platform.mk

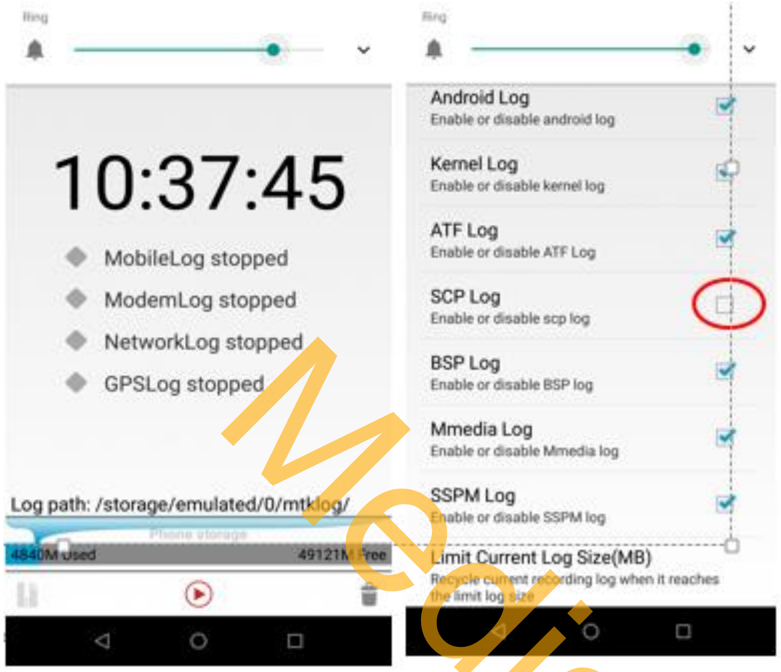
- Warning
  - DO NOT apply this change to ENG build, because AP and SCP log will mix together and hard to recognize.
  - DO NOT use this when measure suspend power, it keeps infra always on.



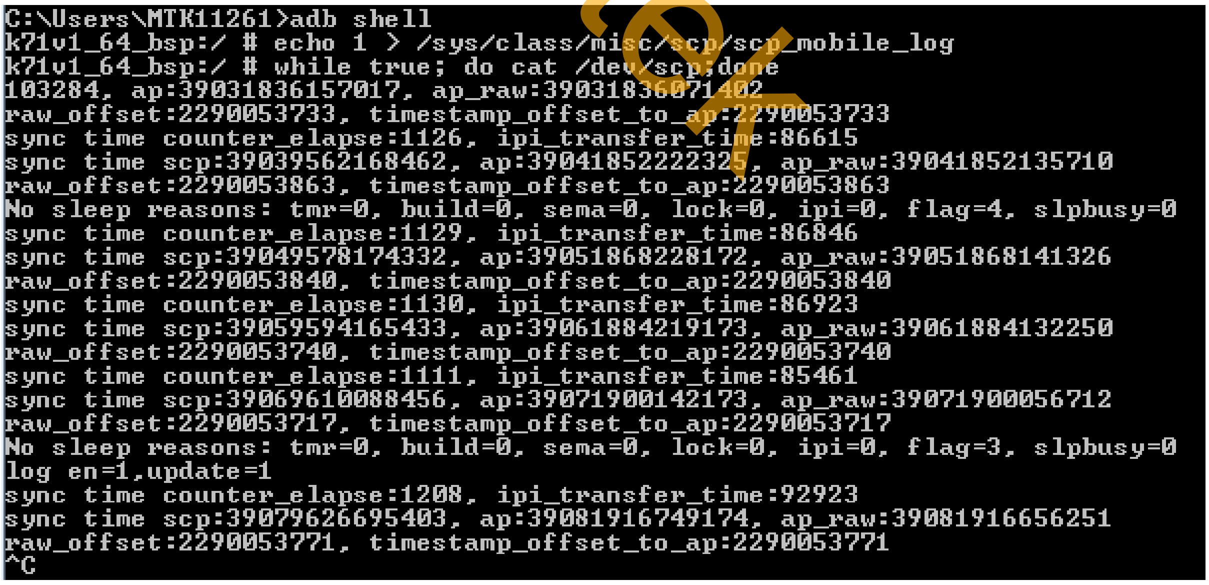
```
CFG_UART_SUPPORT = yes
CFG_MTK_SCPUART_SUPPORT = yes

# 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 直接输出 SCP log

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


Ring
10:37:45
MobileLog stopped
ModemLog stopped
NetworkLog stopped
GPSLog stopped
Log path: /storage/emulated/0/mtklog/
Phone storage
4840M used
49121M Free
Ring
Android Log
Enable or disable android log
Kernel Log
Enable or disable kernel log
ATF Log
Enable or disable ATF Log
SCP Log
Enable or disable scp log
BSP Log
Enable or disable BSP log
Mmedia Log
Enable or disable Mmedia log
SSPM Log
Enable or disable SSPM log
Limit Current Log Size(MB)
Recycle current recording log when it reaches the limit log size

2. Enter adb shell and then 输入如下命令
   - ✓ echo 1 > /sys/class/misc/scp/scp_mobile_log
   - ✓ 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 机制

- 1 .确认 AEE 机制打开

- 执行 adb shell “aee -m 3”。可以通过 adb shell “getprop persist.mtk.aee.mode”查看获得的值是否为 3，是 3 则说明执行成功

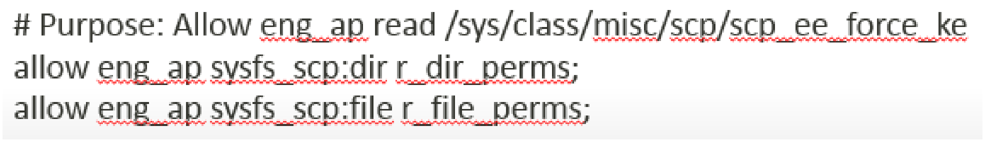
- 2.确认 SCP db 可以 dump
  - 執行 adb command:
  - 1) adb shell cat /sys/class/misc/scp/scp_A_db_test (會看到返回 dumping SCP A db)
  - 2）下面路徑要可找到 db
    sdcard/mtklog/aee_exp/data/aee_exp/

6.5.5 SCP 重启关联 AP 重启

功能说明：Force enable KE when SCP EE occur

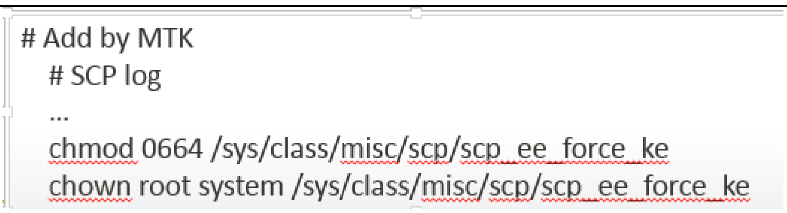
- Default Status: DISABLE
  - How to switch: write the control node to turn on/off
    - How to use explain in next page
- When Enable:
  - Reset scope: Whole system (KE)
  - Debug info: Full RAM Dump (takes a long time) and mobile log
    - db = db.xx.EE & db.fatal.xx.KE
- When Disable:
  - Reset scope: 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 设定权限
   Must allow to access the path: /sys/class/misc/scp/scp_ee_force_ke
   例如： 要允许 eng_app access sysfs_scp
   要在 device/mediatek/sepolicy/basic/non_plat/eng_app.te 下面
   增加自己的 xxx.te 文件，比如上面的 eng_app，
   编写内容内容如下



```
# 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 设定
   device/mediatek/mt6771/init.mt6771.rc 添加如下：



```
# 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
```