item\_id == sns\_smgr.last\_requested\_sensor\_dep\_reg\_group\_id

--->sns\_smgr\_process\_sensor\_dep\_reg\_data( cfg\_group\_ptr );

--->sns\_smgr\_req\_reg\_data(cfg\_group\_ptr->next\_group\_id, SNS\_SMGR\_REG\_ITEM\_TYPE\_GROUP);

//如果next\_group\_id为0

则：sns\_smgr.all\_init\_state = SENSOR\_ALL\_INIT\_CONFIGURED;

否则：next+1循环

sns\_resp.sns\_err\_t != SENSOR1\_SUCCESS

Else处理

I2cDrv\_Init

//初始化互斥

--->I2cSys\_CreateCriticalSection(&i2cDrvSync);

------>qurt\_mutex\_init(&pCritSec->i2c\_mutex);

I2c初始化

--->init\_function:I2cBsp\_Init

i2c\_adsp\_8953.xml

i2c\_adsp\_8953.c

(item\_id == SNS\_REG\_GROUP\_SSI\_GPIO\_CFG\_V02) &&

(item\_type == SNS\_SMGR\_REG\_ITEM\_TYPE\_GROUP)

--->sns\_hw\_update\_ssi\_reg\_items(item\_id, data\_ptr);

RETURN\_IF\_INVALID\_BUS\_ID(eI2cBusId);

//i2c8对应eI2cBusId：7

I2cDrv\_AddDevice(eI2cBusId, pDesc);

I2cBsp\_Init

----->I2cDrv\_Init();

----->I2cDrv\_InitDevice(pDrvProperty->eDevId,pDesc);

( (item\_id >= SNS\_REG\_GROUP\_SSC\_GPIO\_CFG\_01\_V02) &&

(item\_id <= SNS\_REG\_GROUP\_SSC\_GPIO\_CFG\_30\_V02) ) &&

(item\_type == SNS\_SMGR\_REG\_ITEM\_TYPE\_GROUP)

----->做gpio配置

Mpu6050

----->sns\_dd\_mpu6xxx\_init

俩种传感器：类型3和4:SNS\_DDF\_SENSOR\_MAG

地址和i2c总线：

I2C addr: 0x68 bus instance: 0x8

sns\_smgr\_get\_attr

---->SMGR\_DRV\_FN\_PTR(sensor\_ptr)->get\_attrib

sns\_smgr

Sensor初始化入口：

drv\_status = SMGR\_DRV\_FN\_PTR()

sns\_smgr\_process\_reg\_ssi\_config

----->sns\_smgr\_create\_if\_handle

sns\_smgr\_parse\_reg\_devinfo\_resp

------>sns\_smgr\_create\_if\_handle

获取配置：cfg\_group\_ptr =

(sns\_reg\_ssi\_smgr\_cfg\_group\_s\*)data\_ptr;

===================

sns\_smgr\_process\_reg\_ssi\_config

---->sns\_hw\_update\_ssi\_reg\_items

=====

sns\_smgr\_process\_reg\_devinfo

---->sns\_hw\_update\_ssi\_reg\_items

sns\_smgr\_process\_reg\_data

---->sns\_smgr\_process\_reg\_devinfo

----->sns\_smgr\_process\_reg\_ssi\_config

========

---->Else的处理

=======

数据来源：

sns\_smgr\_process\_reg\_resp\_msg

sns\_smgr\_hw\_init

---->sns\_os\_sigs\_add--->SNS\_SMGR\_QMI\_MSG\_SIG

---->sns\_hw\_bd\_config();//SNS\_DBG\_MOD\_DSPS\_SMGR

---->sns\_smgr\_dd\_init();

===接受QMI消息直到SENSOR\_ALL\_INIT\_DONE

---->sns\_smgr\_process\_msg();//处理QMI消息

---->sns\_smgr\_dd\_init();//做初始化

===========

sns\_smgr\_ltcy\_measure\_check();

sns\_smgr\_process\_msg

---->sns\_smgr\_process\_reg\_resp\_msg(msg\_ptr);

---->sns\_smgr\_process\_reg\_data

sns\_smgr\_task

---->smgr\_init();

----->sns\_smgr\_sensor\_init();

---->sns\_smgr\_mr\_init(sns\_smgr.sig\_grp);

---->sns\_smgr\_dl\_init();

---->sns\_smgr\_hw\_init();

---->sns\_smgr\_dl\_deinit();

----->sns\_init\_done();

sns\_smgr\_task\_main();

sns\_smgr\_task\_main

接受信号处理：

SNS\_SMGR\_SENSOR\_STATE\_SIG--->smgr\_process\_sensor\_event()

SNS\_SMGR\_PWR\_RAIL\_SIG------>sns\_hw\_send\_powerrail\_msg\_tmr\_cb\_proc();

SNS\_SMGR\_QMI\_MSG\_SIG---->sns\_smgr\_process\_msg();

SNS\_SMGR\_DEPOT\_COPY\_SIG---->sns\_smgr\_copy\_uimg\_depots();

SNS\_SMGR\_VOTE\_MIPS\_BW--->sns\_smgr\_vote\_mips\_bw\_reqd();

SNS\_SMGR\_APP\_WAKE\_SIG--->sns\_smgr\_handle\_app\_wake\_sig();

SNS\_SMGR\_APP\_SLEEP\_SIG--->sns\_smgr\_handle\_app\_sleep\_sig();

sns\_smgr\_dd\_init();//做初始化

相关列表初始化：

1.设置为idle状态：

sns\_smgr\_set\_sensor\_state(sensor\_ptr, SENSOR\_STATE\_IDLE);

2.sns\_smgr\_init\_lpf\_table(sensor\_ptr);

sns\_smgr\_init\_odr\_table(sensor\_ptr);

sns\_smgr\_init\_fifo\_cfg(sensor\_ptr);

sns\_smgr\_init\_sensor\_info(sensor\_ptr);

3.初始化完成后设置为off：

sns\_smgr\_set\_sensor\_state(sensor\_ptr, SENSOR\_STATE\_OFF);

如果为SENSOR\_INIT\_FAIL则卸载驱动：

4.关闭电源和时钟：

sns\_hw\_set\_qup\_clk(false);

sns\_hw\_power\_rail\_config(SNS\_PWR\_RAIL\_OFF);

电源i2c的初始化：

1.初始化smgr\_sensor\_s结构体：

sns\_smgr\_sensor\_init\_post\_cfg

2.校准数据：

smgr\_send\_reg\_message\_for\_fac\_cal();

3.电源i2c:

sns\_hw\_power\_rail\_config(SNS\_PWR\_RAIL\_ON\_LPM);

sns\_hw\_set\_qup\_clk(true);

4.申请配置数据

sns\_smgr\_req\_reg\_data

5.gpio配置：

6.sensor初始化

SMGR\_DRV\_FN\_PTR

sns\_pm.core\_handle.pmic\_client\_lvs = npa\_create\_sync\_client (PMIC\_NPA\_GROUP\_ID\_SENSOR\_VDDIO, "Sensors", NPA\_CLIENT\_REQUIRED);

sns\_pm.core\_handle.pmic\_client\_ldo = npa\_create\_sync\_client (PMIC\_NPA\_GROUP\_ID\_SENSOR\_VDD, "Sensors", NPA\_CLIENT\_REQUIRED);

sns\_dd\_mpu6xxx\_set\_attr

.set\_attrib = &sns\_dd\_mpu6xxx\_set\_attr

set\_attrib

sns\_smgr\_set\_attr

------->SMGR\_DRV\_FN\_PTR(sensor\_ptr)->set\_attrib

sns\_dd\_mpu6xxx\_enable\_interrupt

<----sns\_dd\_mpu6xxx\_enable\_sched\_data

<----sns\_dd\_mpu6xxx\_enable\_dmp

<----sns\_dd\_mpu6xxx\_set\_attr

sns\_dd\_mpu6xxx\_enable\_interrupt

------>sns\_ddf\_signal\_register//注册中断

------>sns\_ddf\_write\_port\_byte//使能中断

流程分析：

1.sns\_hw\_bd\_config

----->sns\_smgr\_hw\_read\_ssc\_gpio\_config\_from\_reg();

----->sns\_smgr\_hw\_read\_ssi\_data\_from\_reg();

sns\_ddf\_signal\_register

----->GPIOInt\_RegisterIsr

------->sns\_ddf\_smgr\_notify\_irq

sns\_dd\_mpu6xxx\_enable\_sched\_data

.enable\_sched\_data = &sns\_dd\_mpu6xxx\_enable\_sched\_data,

1. sns\_smgr\_dd\_init------------->退出
2. sns\_smgr\_mr\_put\_msg 接收到SNS\_SMGR\_QMI\_MSG\_SIG信号。

sns\_smgr\_set\_attr

<------sns\_smgr\_set\_power\_attr

<-----sns\_smgr\_fifo\_set\_watermark

<----smgr\_build\_odr\_list

<----sns\_smgr\_config\_odr

<----sns\_smgr\_set\_lpf

<----sns\_smgr\_update\_md\_attribute

<----sns\_smgr\_process\_calibration\_req

<----sns\_smgr\_turn\_off\_sensor

<----sns\_smgr\_process\_wake\_up\_event

Sns\_smgr\_sensor.c

1. sns\_smgr\_hw\_init被SNS\_SMGR\_QMI\_MSG\_SIG唤醒并执行A：sns\_smgr\_process\_msg();B：sns\_smgr\_dd\_init();

A：

enable\_sched\_data

--->sns\_smgr\_dd\_enable\_sched\_data

sns\_dd\_mpu6xxx\_enable\_dmp

<---sns\_dd\_mpu6xxx\_set\_odr

sns\_ddf\_smgr\_notify\_irq

----->sns\_smgr\_signal\_me(SNS\_SMGR\_DD\_SERVICE\_SIG);

sns\_dd\_mpu6xxx\_set\_odr

<-----sns\_dd\_mpu6xxx\_set\_powerstate

<----sns\_dd\_mpu6xxx\_set\_attr

<----sns\_dd\_mpu6xxx\_handle\_irq

sns\_smgr\_dd\_enable\_sched\_data

<----sns\_smgr\_fifo\_set\_wm\_and\_intr

<----sns\_smgr\_fifo\_flush

<----sns\_smgr\_config\_odr

<----sns\_smgr\_turn\_off\_sensor

SNS\_SMGR\_DD\_SERVICE\_SIG

sns\_smgr\_task\_main

------>sns\_smgr\_dd\_service();

sns\_smgr\_config\_odr\_lpf

----->sns\_smgr\_config\_odr

sns\_dd\_mpu6xxx\_handle\_irq

<------.handle\_irq = &sns\_dd\_mpu6xxx\_handle\_irq

sns\_smgr\_dd\_service

sns\_ddf\_timer\_dispatch();//延时轮训处理

sns\_ddf\_signal\_dispatch();//中断处理

sns\_smgr\_config\_odr\_lpf

<-------sns\_smgr\_process\_wake\_up\_event

<------sns\_smgr\_process\_client\_update\_events

<-----sns\_smgr\_process\_md\_state\_events

sns\_ddf\_signal\_dispatch

sns\_ddf\_sig\_tb.sig\_info[sig\_num].driver->handle\_irq

sns\_smgr\_process\_wake\_up\_event

<-----sns\_smgr\_process\_idle\_state\_events

sns\_smgr\_process\_client\_update\_events

<----sns\_smgr\_process\_ready\_state\_events

sns\_smgr\_process\_ready\_state\_events

sns\_smgr\_process\_md\_state\_events

sns\_smgr\_process\_idle\_state\_events

<----sns\_smgr\_process\_individual\_sensor\_events

SNS\_SMGR\_SENSOR\_STATE\_SIG

<----sns\_rh\_process\_event\_gated\_buffering\_request

<----sns\_rh\_cancel\_service

<----sns\_rh\_process\_odr\_changed\_sig

<----sns\_rh\_process\_fifo\_flushed\_sig

<----sns\_rh\_handle\_modem\_wake\_sig

<----sns\_rh\_process\_buffering\_report\_sigs

<----sns\_rh\_cancel\_restricted\_service

<----sns\_rh\_sol\_schedule\_report

<---sns\_rh\_sol\_process\_periodic\_request

<----sns\_rh\_sol\_process\_buffering\_request

<---sns\_rh\_sol\_handle\_pending\_rpt

<---sns\_rh\_sol\_resume\_sensor\_clients

<---sns\_ddf\_smgr\_notify\_test\_complete

<---sns\_smgr\_notify\_event\_in\_bigimage

<---sns\_smgr\_fifo\_configure

<---sns\_smgr\_sensor\_ev\_timer\_cb

<---sns\_smgr\_sensor\_handle\_first\_valid\_sample

<---sns\_smgr\_put\_next\_request

sns\_smgr\_process\_individual\_sensor\_events

<----smgr\_process\_sensor\_event

smgr\_process\_sensor\_event

<-----sns\_smgr\_handle\_bigimage\_signals

sns\_smgr\_handle\_bigimage\_signals

<----sns\_smgr\_task\_main

SNS\_SMGR\_SENSOR\_STATE\_SIG

SNS\_SMGR\_ENABLE\_MD\_SIG

SNS\_SMGR\_DISABLE\_MD\_SIG

sns\_rh\_update\_sensor\_event

<---sns\_rh\_add\_event\_gated\_buffering\_request

<---sns\_rh\_process\_md\_int\_disabled\_sig

<---sns\_rh\_sol\_update\_sensor\_events

<---sns\_rh\_sol\_resume\_sensor\_clients

Main

--->sns\_init();

sns\_init

---->sns\_init\_once();

---->init\_ptrs[i]()

---->SNS\_INIT\_FUNCTIONS

---->sns\_init\_dsps

---->sns\_em\_init

---->sns\_smr\_init

---->sns\_dl\_init

----->sns\_rh\_init

----->sns\_debug\_init

----->smgr\_qmi\_ping\_client\_start

SNS\_SMGR\_ENABLE\_MD\_SIG

<-----sns\_rh\_update\_sensor\_event

sns\_rh\_add\_event\_gated\_buffering\_request

<--sns\_rh\_process\_event\_gated\_buffering\_request

<---sns\_rh\_process\_internal\_smgr\_request

<---sns\_rh\_process\_msg

<---sns\_rh\_process\_big\_image\_signals

<----sns\_rh\_task\_loop

<---sns\_rh\_task

<----sns\_rh\_init

<----SNS\_INIT\_FUNCTIONS

SNS\_PWR\_RAIL\_OFF

<----sns\_hw\_power\_rail\_config(SNS\_PWR\_RAIL\_OFF);

<----sns\_smgr\_dd\_init

<----power\_rail = SNS\_PWR\_RAIL\_OFF;

<--sns\_smgr\_update\_power\_state

SNS\_PWR\_RAIL\_OFF