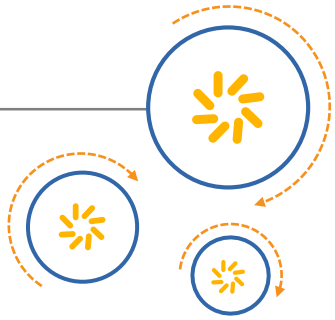
Qualcomm Technologies, Inc.



Snapdragon™ Sensors Core (SSC) New Sensor Driver Integration Guide for Linux Android

80-N7635-1 F

July 13, 2015

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	September 2011	Initial release
B	June 2012	Updated steps in Chapter 2
C	December 2012	Updated for MSM8974 family chipsets
D	February 2013	Added Sensor Single Image feature; added Chapter 2
E	November 2013	Numerous changes were made to this document; it should be read in its entirety.
F	July 2015	Updated for MSM8994/MSM8992, MSM8952/MSM8956/MSM8976, and MSM8996 family chipsets; updated Sections 1.1 and A.1, Table 2-1, and Chapter 3

Contents

Tables

1 Introduction

1.1 Purpose

This document provides guidelines and general instructions for integrating a new sensor driver to the Qualcomm® Snapdragon™ Sensors Core (SSC) on MSM8960, APQ8064, MSM8974, MSM8x26, APQ8084, MSM8994/MSM8992, MSM8952/MSM8956/MSM8976, and MSM8996 chipsets.

This document is intended for handset developers who need to add a new sensor to the SSC.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Shading indicates content that has been added or changed in this revision of the document.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Driver Integration Platforms

Device driver integration steps differ for various product lines. See Table 2-1 to determine the applicable instructions.

Table 2-1 Driver integration instructions

Platform	Instructions
MSM8974 (LA 2.0), MSM8x26, APQ8084, MSM8994/MSM8992, MSM8952/MSM8956/MSM8976, MSM8996	Chapter 3
MSM8974 (LA 1.0 ES5 and later releases)	Chapter 4
MSM8960, APQ8064, MSM8974 (LA 1.0 ES4 and earlier releases)	Chapter 5

Starting with the MSM8974 Linux Android (LA) ES5 (1015) release, a new feature, Sensors Single Image (SSI), is supported. SSI supports loading multiple sensor drivers into a single ADSP image. Different sensors may be selected without recompiling or reloading the ADSP image.

3 Integrating a New Sensor Driver with Configuration File-Supported SSI

NOTE: Numerous changes were made in this chapter.

By default, the SSC (ADSP/SLPI) build system assumes the configuration given in Table 3-1, Table 3-2, Table 3-5, and Table 3-6.

Table 3-1 Default configuration for MSM8974

Sensor type	Default sensor part	Default configuration option
Accelerometer	MPU6050	CONFIG_SUPPORT_MPU6050
Gyroscope	MPU6050	CONFIG_SUPPORT_MPU6050
Magnetometer	AK8963C	CONFIG_SUPPORT_AKM8963
Ambient light sensor/proximity sensor	APDS9900	CONFIG_SUPPORT_APDS99XX
Pressure	BMP180	CONFIG_SUPPORT_BMP085

Table 3-2 Default configuration for MSM8x26

Sensor type	Default sensor part	Default configuration option
Accelerometer	BMA222E	CONFIG_SUPPORT_BMA2X2
Gyroscope	MPU3050	CONFIG_SUPPORT_MPU3050
Magnetometer	HSCDTD007A	CONFIG_SUPPORT_HSCD008
Ambient light sensor/proximity sensor	TMD27713T	CONFIG_SUPPORT_APDS99XX

Table 3-3 Default configuration for MSM8994/MSM8992 and MSM8952

Sensor type	Default sensor part	Default configuration option
Accelerometer	BMI058	CONFIG_SUPPORT_BMA2X2
Gyroscope	BMI058	CONFIG_SUPPORT_BMG160
Magnetometer	HSCDTD008A	CONFIG_SUPPORT_HSCD008
Pressure	LPS25H	CONFIG_SUPPORT_LPS25H
Ambient light sensor/proximity/RGB	TMG3993	CONFIG_SUPPORT_TMG399X
Capacitive (only on MSM8994/MSM8992)	AD7146	CONFIG_SUPPORT_AD7146
Humidity (only on MSM8994/MSM8992)	SHTC1	CONFIG_SUPPORT_SHTC1

Table 3-4 Default configuration for MSM8996

Sensor type	Default sensor part	Default configuration option
Accelerometer	BMI160	CONFIG_SUPPORT_BMI160
Gyroscope	BMI160	CONFIG_SUPPORT_BMI160
Magnetometer	AK9915	CONFIG_SUPPORT_AKM099xx_FIFO
Pressure	BMP280	CONFIG_SUPPORT_BMP280
Ambient light sensor/proximity/RGB	APDS9960	CONFIG_SUPPORT_TMG399X
Capacitive	AD7146	CONFIG_SUPPORT_AD7146
Humidity	SHTW1	CONFIG_SUPPORT_SHTC1
HeartRate	ADPD142RI	CONFIG_SUPPORT_ADPD142
Hall effect	BU52053NVX	CONFIG_SUPPORT_BU52061NVX
Thermopile	DTS201A	CONFIG_SUPPORT_DTS201A
UV sensor	HSVDDD002A	CONFIG_SUPPORT_HSVDDD002A

To include a sensor part that is not listed in Table 3-1, Table 3-2, Table 3-5, or Table 3-6, perform the steps described in the following sections.

With MSM8996, SSC is supported by a dedicated processor called Sensors Low Power Island (SLPI); therefore, all the directory paths are referred to as <slpi_proc>. Other chipsets, i.e., MSM8994/MSM8992 and MSM8952, continue to have sensors core supported in the ADSP; therefore, the directory paths are referred to as <adsp_proc>. Paths of the various files to be updated are the same across ADSP and SLPI targets except for <adsp_proc> or <slpi_proc>.

3.1 Declare the entry function for the driver

Add the following entry function for the new driver in <adsp_proc/slpi_proc>\sensors\dd\qcom\inc\sns_dd.h:

```
/* Make the device function lists sharable with SMGR */.
extern sns_ddf_driver_if_s sns_dd_<new_sensor_model>_if;
…
```

The function name sns_<new_sensor_model>_driver_if is identical to the driver entry function defined in the new driver itself, e.g., for MPU6050, it is sns_dd_mpu6050_if.

3.2 Add the new driver files

1. Add the new driver files in <adsp_proc/slpi_proc>\sensors\dd\qcom\src in the ADSP/SLPI build, and update the <adsp_proc/slpi_proc>\sensors\dd\qcom\build\dd_qcom.scons file. Alternatively, add the driver in <adsp_proc/slpi_proc>\sensors\dd\vendor\src, and update the <adsp_proc/slpi_proc>\sensors\dd\vendor\build\dd_vendor.scons file.
2. Configure Sensors.scons to compile the new files. To include the driver, in <adsp_proc/slpi_proc>\sensors\build\Sensors.scons, add:

   ```
   env.Append(CPPDEFINES = [”CONFIG_SUPPORT_<NEW_SENSOR_MODEL>”])
   ```

Before compiling the ADSP/SLPI build, it is important to edit the Sensors.scons file to update all of the drivers being included in the image with the CONFIG_SUPPORT_* CPP defines.

The list of drivers in this file should be limited to those that are needed for the single image.

3.3 Generate the UUID to associate with the new driver

Licensees may need to (and may) create new registry entries for new drivers. UUID generation need not be controlled by QTI. However, QTI will add new UUIDs for future sensors as the sensor vendors make drivers available.

Commonly available UUID generation tools can be used to create the UUID, e.g., in Linux, the uuidgen, a command line utility, can be used to create new UUID values. Generating a UUID is not necessary to enable a driver that is already integrated. See Section 3.5 for more information.

1. Update the sns_reg_common.h files on both the ADSP/SLPI and APSS builds to add UUID:
   - ADSP/SLPI – <adsp_proc/slpi_proc>\Sensors\common\inc\sns_reg_common.h
   - APSS – android\vendor\qcom\proprietary\sensors\dsps\sensordaemon\common\inc\ sns_reg_common.h

```
#define SNS_REG_UUID_<NEW_SENSOR_MODEL> \

{0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0xE
E,0xFF,0x11}
```

2. Update the Sensors Manager (SMGR) function pointer map so that the SMGR can associate the new UUID with the new DD function pointer.

   For MSM8974, MSM8x26, and APQ8084, modify adsp_proc/Sensors/smgr/src/common/ sns_smgr_init.c, update smgr_sensor_fn_ptr_map[]:

```
#ifdef CONFIG_SUPPORT_<NEW_SENSOR_MODEL>
{ SNS_REG_UUID_XXXX, &sns_xxx_xxxx_driver_fn_list},
#endif
```

   For MSM8994/MSM8992, MSM8952, and MSM8996, modify <adsp_proc\slpi_proc>/ Sensors/smgr/src/sns_smgr_reg.c, update smgr_sensor_fn_ptr_map[]:

```
#ifdef CONFIG_SUPPORT_<NEW_SENSOR_MODEL>
{ SNS_REG_UUID_XXXX, &sns_xxx_xxxx_driver_fn_list},
#endif
```

3.4 Configure the registry

NOTE: It is *mandatory* to write the proper selection of drivers and parameters into the registry for the sensors to operate. This can be done at runtime, or an sns.reg file can be added to the build and loaded as part of the image.

Default values for the registry items listed in Table 3-5 must be configured for new sensor integration.

Table 3-5 Device-specific registry items

Registry ID	Registry item	Description	Source
<base index> + 0	UUID_HIGH	High 8-bytes of the UUID mapping to DD	Unique ID (can be generated using a utility like uuidgen in Linux)
< base index > + 1	UUID_LOW	Low 8-bytes of the UUID mapping to DD	
< base index > + 2	OFF_TO_IDLE	Configurable delay time for SMGR (microseconds)	Defined by the sensor part specification (from the vendor datasheet)
< base index > + 3	IDLE_TO_READY	Configurable delay time for SMGR (microseconds)	Defined by the sensor part specification (from the vendor datasheet)
< base index > + 4	I2C_BUS/SPI_BUS	I2C bus/SPI bus	Customer hardware – BLSP bus number to which the sensor is connected, e.g., BLSP 1-12 on MSM8994; for a sensor connected to BLSP5 as I2C, it is 5. For SPI – For example, with MSM8996, a sensor connected to SSC_SPI1 is 0x1001 (and 0x1002 if connected to SSC_SPI2)
< base index > + 5	REG_GROUP_ID	Registry items associated with this driver	Defined by the SSC framework per sensor – Can keep the same as the default sensor_def_qcomdev.conf for each sensor type
< base index > + 6	CAL_PRI_GROUP_ID	Registry calibration for this driver	
< base index > + 7	GPIO1	GPIO for device interrupt	MSM™ GPIO number to which the sensor interrupt is connected
< base index > + 8	GPIO2	GPIO for second device interrupt	
< base index > + 9	SENSOR_ID	SMGR ID associated with sensor type	Defined by the SSC framework for each sensor type (SNS_SMGR_ID_*_V01 in sns_smgr_common_v01.h)

Registry ID	Registry item	Description	Source
< base index > + 10	I2C_ADDRESS/ SPI CS#	I2C address of the device (7-bit) For SPI, this registry item indicates the ChipSelect (CS) pin number to which the sensor is connected	I2C address – Defined by the sensor part specification (from the vendor datasheet) SPI CS# – Based on the OEM hardware to which the CS is connected; e.g., on the MSM8996, accel is connected to CS0 of SPI1 (i.e., SSC_8), it is 0; mag is connected to CS1 of SPI1 (i.e., SSC_0), it is 1
< base index > + 11	DATA_TYPE1	Primary data type	Primary and secondary data types provided by the vendor based on driver implementation in compliance with DDF
< base index > + 12	DATA_TYPE2	Secondary data type	
< base index > + 13	RELATED_ SENSOR_INDEX	Indicates related sensor; -1 for no related sensor	Vendor software implementation; -1 by default for no related sensor
< base index > + 14	SENSITIVITY_ DEFAULT	Sensitivity setting	Vendor software implementation
< base index > + 15	FLAGS	SMGR flags to indicate DRI vs. Polling mode and other SMGR functionality	▪ Polling 0x00 ▪ DRI 0x80 ▪ FIFO 0xD0 – Required for sensors with FIFO
1982 to 1986; 3682 to 3686	DEVICE_SELECT	Device select is applicable to some device drivers when multiple sensors are supported by a single driver	Vendor software implementation; 0 by default

Registry indexes are defined in vendor/qcom/proprietary/sensors/dsps/api/sns_reg_api_v02.h. By default, the SSC supports 15 configurations that have base registry indexes listed in Table 3-6.

Table 3-6 Base registry indexes

Configuration ID	Base registry index
0	1902
1	1918
2	1934
3	1950
4	1966
5	3602
6	3618
7	3634
8	3650
9	3666
10	5502
11	5518

Configuration ID	Base registry index
12	5534
13	5550
14	5566

3.4.1 Update the configuration file

For MSM8974 LA 2.0, MSM8x26, APQ8084, MSM8994/MSM8992, MSM8952, and MSM8996 builds, the default registry values are defined in a .conf file: vendor\qcom\proprietary\sensors\dsps\reg_defaults\sensor_def_qcomdev.conf.

1. Create a new .conf file in vendor\qcom\proprietary\sensors\dsps\reg_defaults based on the sensors_def_qcomdev.conf example.

   The filename should have the format sensor_def_<oem>.conf.

Table 3-7 Configuration file keys

Key	Purpose
:version	A positive, nonzero version number for the file. It should be equal to the largest version number of any item.
:hardware	Items from the configuration file will only be used if the hardware string value is a substring of the /proc/cpuinfo string. It can be used more than once so that one file can support more than one hardware. If set to an empty string, it indicates common item values for all hardware.
:platform	The string must match the text of either /sys/devices/soc0/hw_platform or /sys/device/soc0/platform_subtype, or can be an empty string to indicate common item values for all platforms.
:property	Has two inputs, a system property key and a desired property value. Items will only be applied if the system property value matches the desired value. Can be set to an empty string to indicate common item values for all properties.

2. Update the registry items corresponding to the driver requirements. Be sure to edit the entries for the appropriate hardware in order for changes to take effect.

   The general format of the .conf file entries is:

   <Registry ID> <Value> <Version>

   where, the Registry ID corresponds to the value defined in vendor\qcom\proprietary\sensors\dsps\api\sns_reg_api_v02.h (see Table 3-5).

   For example, the following sets configuration 0 to use the new sensor with the UUID defined in Section 3.3.

NOTE: The UUID is stored in Little Endian format.

```
:version 0x00010001
:hardware 8974
…
# SSI SMGR Cfg 0
1903 0x11FFEEDDCCBBAA99 0x00010001 # Low UUID
1902 0x9988665544332211 0x00010001 # High UUID
```

```
1906 12 0x00010001 # I2C Bus
1907 1000 0x00010001 # Registry Group ID
1908 0 0x00010001 # Calibration Group ID
1909 66 0x00010001 # GPIO 1
1910 0xFFFF 0x00010001 # GPIO 2
1904 100000 0x00010001 # Off to Idle Time
1905 250000 0x00010001 # Idle to Ready Time
1911 0 0x00010001 # Sensor ID
1912 0x68 0x00010001 # I2C Address
1913 1 0x00010001  # Data Type 1
1914 0 0x00010001 # Data Type 2
1915 1 0x00010001 # Related Sensor Index
1916 1 0x00010001 # Sensitivity Default
1917 0x40 0x00010001 # Flags
1982 0 0x00010001 # Device Select
```

3. Push the updated sensor_def_<oem>.conf file to the following directories:
   - For MSM8974, MSM8x26, APQ8084 – /etc/sensor_def_<oem>.conf
   - For MSM8994/MSM8992, MSM8952, MSM8996 – /etc/sensors/sensor_def_<oem>.conf

   In the case of multiple sensor_def_<xxx>.conf files in the above directories, the files are parsed in alphabetical order. Based on the parsing rules described in Section 3.4.2, any registry item that has an equal or lesser version number than those in files that precede it in alphabetical order will not be applied. For this reason, it is recommended to remove the sensor_def_qcomdev.conf file from the above directories if it is present and have only sensor_def_<oem>.conf.

   Currently, up to three sensor_def_<xxx>.conf files are supported. If more than three sensor_def_<xxx>.conf files are present, all registry items for files parsed after the third (in alphabetical order) are applied regardless of the version number:

```
adb root
adb remount
adb shell rm /etc/sensors/sensor_def_qcomdev.conf
adb push sensors_def_<oem>.conf /etc/sensors/sensor_def_<oem>.conf
adb shell chmod 644 /etc/sensors/sensor_def_<oem>.conf
```

   Ensure that the configuration file is updated, if the defaults change. These default values are programmed into the registry when the registry is created. However, they may be overwritten later. If the licensee is using a fixed set of drivers, this is a good way to ensure that the fixed set of drivers is used.

3.4.2 How the configuration file is applied

As the configuration file is parsed, each item version is compared with the previously saved file version number. If the item version is greater, it overwrites the item with the specified value. The registry stores the specified :version set in the configuration file after it processes the file.

3.4.2.1 Initial parsing

A registry (sns.reg) is created from the sensor_def_<oem>.conf file on boot-up time by the sensors daemon in the following directory paths:

- For MSM8974, MSM8x26, APQ8084 – /data/misc/sensors/sns.reg
- For MSM8994/92, MSM8952, MSM8996 – /persist/sensors/sns.reg

The above directory path is referred to as <SNS_REG_PATH> in rest of this document.

For the initial parsing of the sensor_def_<oem>.conf file, the file version number has an implicit version of 0 since it has not been parsed before. Since every item has a version that is either 1 or greater, every item will be applied. On the second boot, the registry skips all items with version numbers less than or equal to the specified file version number.

To ensure that every item in the configuration file is applied, remove the existing registry file from the <SNS_REG_PATH> directory (or whichever directory has been configured to store the registry):

```
adb shell rm /<SNS_REG_PATH>/sns.reg
adb shell sync
adb reboot
```

3.4.2.2 Using version numbers to update specific registry items

An over-the-air update could change the default registry values by incrementing the file :version key, and any new/updated items. The next time the sensor service starts, it parses the configuration file and overwrites the items with version numbers greater than the previously stored file version number.

To overwrite the I2C address for configuration 0 that was applied from the example in Section 3.4.1, increment the file and item version numbers:

```
:version 0x00010002
:hardware 8974
…
# SSI SMGR Cfg 0
1903 0x11FFEEDDCCBBAA99 0x00010001 # Low UUID
1902 0x9988665544332211 0x00010001 # High UUID
1906 3 0x00010002 # I2C Bus – updating from 12 to 3
```

3.4.2.3 Using version numbers to reapply a registry item after every reboot

Version numbers can also be used to reapply a registry item after every boot. For example, to zero out all gyroscope calibration data after every reboot, the corresponding registry item version numbers could be set to 999, while the file version is left at 2 (or some other value less than 999). Since the item version number is always greater than the stored file version number, the gyroscope calibration items are reapplied every time the sensor service starts:

```
:version 0x00010001
:hardware 8974
…
306 0 0x00011111 # Gyro X-axis Bias
307 0 0x00011111 # Gyro Y-axis Bias
308 0 0x00011111 # Gyro Z-axis Bias
309 0 0x00011111 # Gyro X-axis Scale
310 0 0x00011111 # Gyro Y-axis Scale
311 0 0x00011111 # Gyro Z-axis Scale
```

3.5 Enabling an already integrated driver

Enabling a driver that was already integrated can be done using the configuration file.

3.5.1 Ensure that the driver is being compiled

In <adsp_proc/slpi_proc>\Sensors\build\Sensors.scons, verify that the driver configuration is appended and not commented out:

```
env.Append(CPPDEFINES = ["CONFIG_SUPPORT_XXXX"])
```

If it is not included in the image, uncomment the line, recompile, and push the resulting ADSP/SLPI binaries to the device.

3.5.2 Update the configuration file

Following the instructions in Section 3.4.1, update the configuration file with the UUID associated with the driver and apply the correct configuration settings.

4 Integrating a New Sensor Driver with SSI

By default, the ADSP build system assumes the configuration given in Table 4-1.

Table 4-1 Default configuration for MSM8974

Sensor type	Default sensor part	Default configuration option
Accelerometer	MPU6050	CONFIG_SUPPORT_MPU6050
Gyroscope	MPU6050	CONFIG_SUPPORT_MPU6050
Magnetometer	AK8963C	CONFIG_SUPPORT_AKM8963
Ambient light sensor/proximity sensor	APDS9900	CONFIG_SUPPORT_APDS99XX
Pressure	BMP180	CONFIG_SUPPORT_BMP085

To include a sensor part that is not listed in Table 4-1, perform the steps described in the following sections.

4.1 Declare the entry function for the driver

Add the entry function for the new driver in adsp_proc\sensors\dd\qcom\inc\sns_dd.h:

```
/* Make the device function lists sharable with SMGR */.
extern sns_ddf_driver_if_s sns_dd_<new_sensor_model>_if;
…
```

The function name sns_<new_sensor_model>_driver_fn_list shall be identical to the driver entry function defined in the new driver itself, e.g., for MPU6050, it is sns_dd_mpu6050_if.

4.2 Add the new driver files

1. Add the new driver files in adsp_proc\sensors\dd\qcom\src in the ADSP build and update the adsp_proc\sensors\dd\qcom\build\dd_qcom.scons file. Alternatively, the driver can be added in adsp_proc\sensors\dd\vendor\src and the adsp_proc\sensors\dd\vendor\build\ dd_vendor.scons files.
2. Configure Sensors.scons to compile the new files. To include the driver, in adsp_proc\sensors\build\Sensors.scons, add:

   ```
   #env.Append(CPPDEFINES = ["CONFIG_SUPPORT_<NEW_SENSOR_MODEL>"])
   ```

   Before compiling the ADSP build, it is important to edit the Sensors.scons file to update the CONFIG_SUPPORT_* CPP defines.

   The list of drivers in this file should be limited to those that are needed for the single image.

4.3 Generate the UUID to associate with the new driver

Licensees may need to (and may) create new registry entries for new drivers. UUID generation does not *have* to be controlled by QTI, but QTI will add new UUIDs for future sensors as the sensor vendors make drivers available.

Commonly available UUID generation tools can be used to create the UUID, e.g., in Linux, uuidgen, a command line utility, can be used to create a new UUID value.

1. Update the sns_reg_common.h files on both the ADSP and APSS builds to add UUID:
   - ADSP – adsp_proc\Sensors\common\inc\sns_reg_common.h
   - APSS – android\vendor\qcom\proprietary\sensors\dsps\sensordaemon\common\inc\sns_reg_common.h

```
#define SNS_REG_UUID_<NEW_SENSOR_MODEL> \

{0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0xEE,0xFF,0x11}
```

2. Update the SMGR function pointer map so that the SMGR can associate the new UUID with the new DD function pointer. In adsp_proc/Sensors/smgr/src/common/sns_smgr_init.c, update smgr_sensor_fn_ptr_map[]:

```
#ifdef CONFIG_SUPPORT_XXXX
{ SNS_REG_UUID_XXXX, &sns_xxx_xxxx_driver_fn_list},
#endif
```

4.4 Configure the registry

NOTE: It is *mandatory* to write the proper selection of drivers and parameters into the registry for the sensors to operate. This can be done at runtime, or an sns.reg file can be added to the build and loaded as part of the image.

Default values for the registry items listed in Table 4-2 must be configured for new sensor integration.

Table 4-2 Device-specific registry items

Registry item	Description
UUID_HIGH	High 8-bytes of the UUID mapping to DD
UUID_LOW	Low 8-bytes of the UUID mapping to DD
OFF_TO_IDLE	Configurable delay time for SMGR
IDLE_TO_READY	Configurable delay time for SMGR
I2C_BUS	I2C bus (3 or 12)
REG_GROUP_ID	Registry items associated with this driver
CAL_PRI_GROUP_ID	Registry calibration for this driver
GPIO1	GPIO for device interrupt

Registry item	Description
GPIO2	GPIO for second device interrupt
SENSOR_ID	SMGR ID associated with sensor type
I2C_ADDRESS	I2C address of the device
DATA_TYPE1	Primary data type
DATA_TYPE2	Secondary data type
RELATED_SENSOR_INDEX	Indicates related sensor; -1 for no related sensor;
SENSITIVITY_DEFAULT	Sensitivity setting
FLAGS	SMGR flags, to indicate DRI vs. Polling mode, and other SMGR functionality
DEVICE_SELECT	Device select applicable to some device drivers when multiple sensors are supported by a single driver\

Registry indexes are defined in vendor/qcom/proprietary/sensors/dsps/api/sns_reg_api_v02.h. By default, the SSC supports five configurations that have base registry indexes listed in Table 3-6.

- For LA 1.0 builds, the default values for registry entries are defined in sns_reg_common.h. This file is common to both ADSP and APSS builds.
  - ADSP – adsp_proc\Sensors\common\inc\sns_reg_common.h
  - APSS – android\vendor\qcom\proprietary\sensors\dsps\sensordaemon\common\inc\ sns_reg_common.h
  - Update the corresponding registry items in the AP side file to the default configuration for the device driver.
  - The UUID_HIGH and UUID_LOW fields are Little Endian. For example, the UUID defined in Section 4.3 is configured as follows:

```
#define SNS_REG_DEF_ITEM_8974_SSI_SMGR_CFG0_UUID_HIGH
0x11FFEEDDCCBBAA99ULL
#define SNS_REG_DEF_ITEM_8974_SSI_SMGR_CFG0_UUID_LOW
0x8877665544332211ULL
```

  - Remove /data/misc/sensors/sns.reg file and reboot the device.
- Ensure that the apps code is updated if the defaults change.
  - See “default values for items in group SNS_REG_GROUP_SSI_SMGR” inside that file for a list of all defaults.
  - These default values are programmed into the registry when the registry is created, but may be overwritten later.
  - If the licensee is using a fixed set of drivers, this is a good way to ensure that the fixed set of drivers is used.
- With SSI, the configuration from sns_smgr_sensor_config.h no longer exists and fields in this file have now been moved to the registry, but have the same meanings with the following exceptions:
  - DD_FN_LIST_PTR has been replaced with a UUID in the registry to identify a driver.
  - REG_ITEM_TYPE has been removed, since the registry is not used or is a group entry.

- REG_ITEM_ID is set to -1 if there is no calibration registry entry for this device; otherwise it is set to the group ID.
- CAL_PRI_TYPE has been removed, since the registry is not used or is a group entry.
- CAL_PRI_ID is set to -1 if there is no calibration registry entry for this device; otherwise it is set to the group ID.

4.5 Use the SSI command line utility to update the registry

This command line utility reads and writes to the SSI registry values to view or select sensor drivers and driver configurations.

The provided sns_regedit_ssi utility can be used as shown:

```
sns_regedit_ssi [-r|-w] [-s|-d <SENSOR>]
```

- -r – Read registry SSI data
- -w – Write SSI data
- -a – Autodetect (will overwrite existing SSI data)

By default, it selects the MSM8974 PoR sensors. Use -s or -d to pick different sensors.

Overwrite the default MSM8974 sensors by using:

- -s <SENSOR> – Configures to use a certain sensor part in Polling mode
- -d <SENSOR> – Configures to use a certain sensor part in Data-Ready-Interrupt (DRI) mode
- -f <SENSOR> – Configures to use a certain sensor part in FIFO mode

An example command line is:

```
adb shell sns_regedit_ssi -s LIS3DH -s LSM303DLHC -s LPS331AP -s L3GD20 –s
APDS99XX –w
```

The above command line is used to choose the following set of sensors in Polling (non-DRI) mode:

- accel – LIS3DH
- gyro – L3GD20
- mag – LSM303DLHC
- pressure – LPS331AP
- prox/light – APDS99xx

Currently, only the MSM8974 PoR sensors support DRI. Use Polling mode if unsure whether DRI is supported. This writes the proper configuration for those sensors into the registry, which are read by the SMGR when the ADSP is next booted.

4.6 Limitations

- Sensors Algorithm Manager (SAM) algorithms
  - Mag cal – SSI loads the correct mag cal library, if present in /system/lib on the device, based on the UUID in the registry.
  - AMD/RMD – The sns_regedit_ssi utility writes the tuning parameters for these algorithms for some sensors; however, these algorithms have not been tuned for all sensors.
  - Other algorithms – Other SAM algorithms (except AMD/RMD) are not updated based on registry settings; each algorithm should be tuned with the final set of sensors on the end-user device.
- Sensor-specific registry data and calibration data
  - Per sensor, there is only one location in the registry to store calibration data and sensor data. It is *not* cleared or updated if sensors are changed and an old registry is left on the device. It is up to the licensee to clear the registry if the sensors change; otherwise incorrect or old data may be used.

4.7 Edit the SSI command line utility to support new drivers

1. To use the sns_regedit_ssi command line with a new driver, define an initialization function with the device registry configurations:

```
int init_oemxxxx_config( sns_reg_ssi_smgr_cfg_group_s *cfg_array,
dev_info_t *dev_info, regitems_t *regitems );
```

2. Update the uuid_array in vendor\qcom\proprietary\sensors\dsps\test\src\sns_regedit_ssi.c with the driver-associated UUID and initialization function, e.g.:

```
#ifdef CONFIG_SUPPORT_XXXX
{ OEMXXXX, SNS_REG_UUID_XXXX, &sns_xxx_xxxx_driver_fn_list},
#endif
```

3. Recompile the sns_regedit_ssi.c file.
4. Push the resulting binary to the device:

```
adb push sns_regedit_ssi /system/bin/sns_regedit_ssi
adb shell chmod 777 /system/bin/sns_regedit_ssi
```

5. Execute the command to configure the new device driver:

```
adb shell sns_regedit_ssi –s OEMXXXX –w
```

6. Reboot the device.

5 Integrating a New Sensor Driver

This chapter provides the procedure for integrating a new sensor driver into the build for MSM8974 software releases prior to 1015 (ES5) and all MSM8960 and APQ8064 software releases. For MSM8974 1015 (ES5) and later releases, see Chapter 2.

By default, the DSPS build system assumes the configuration given in Table 5-1.

Table 5-1 Default configuration

Sensor type	Default sensor part	Default configuration option
MSM8960 and APQ8064		
Accelerometer	LIS3DH	CONFIG_USE_LIS3DH
Gyroscope	MPU3050	CONFIG_USE_MPU3050
Magnetometer	AK8975C	CONFIG_USE_AK8975C
Ambient light sensor/proximity sensor	ISL29028	CONFIG_USE_ISL29028
MSM8974		
Accelerometer	MPU6050	CONFIG_USE_MPU6050[1]
Gyroscope	MPU6050	CONFIG_USE_MPU6050[1]
Magnetometer	AK8963C	CONFIG_USE_AK8963C[1]
Ambient light sensor/proximity sensor	APDS9900	CONFIG_USE_APDS9900[1]
Pressure	BMP180	CONFIG_USE_BMP180[1]
[1]See Step 3 of Section 5.3.		

To include a sensor part that is not listed in Table 5-1, perform the steps described in the following sections. Also, be sure to exclude the default configuration option from the build, as mentioned in Section 5.3.

5.1 Declare the entry function for the driver

For MSM8960 and APQ8064 builds, add the following line to declare the entry functions for the driver in the file dsps_proc\core\sensors\dd\inc\sns_dd.h:

```
extern sns_ddf_driver_if_s sns_<new_sensor_model>_driver_fn_list;
```

For MSM8974, include the interface functions in the file adsp_proc\Sensors\dd\inc\sns_dd.h:

```
/* Make device function lists sharable with SMGR */
extern sns_ddf_driver_if_s sns_dd_<new_sensor_model>_if;
…
```

NOTE: The function name sns_<new_sensor_model>_driver_fn_list shall be identical to the driver entry function defined in the new driver itself. For example, for MPU6050, it is sns_dd_mpu6050_if.

5.2 Add the new driver files

Include device driver source files in the dd.scons file.

1. For MSM8960 Linux Android (LA) 1.0, 1.04, 1.5, and 1.7 releases prior to 1720, and APQ8064 LA 1.0/1.1 releases prior to 1122, in the file <root>\dsps_proc\core\sensors\dd\build\dd.scons, add the new driver source file(s) under the section DD_8960_SOURCES =:

   ```
   "${BUILDPATH}/sns_dd_<new_sensor_model>.c",
   ```

   For MSM8960 LA 1.7 1720 and subsequent releases, and APQ8064 LA 1.0/1.1 1122 and subsequent releases, Step 1 is not required. All the driver files in the path <root>\dsps_proc\core\sensors\dd\src\ are included in the build by default. Building the files does not increase the size of the image, as the linker automatically excludes all unused object files from the final image.

2. For MSM8974 LA 1.0 1014 and subsequent releases, add the new driver source file(s) under section BRINGUP_8974 and DD_8660_FFA_SOURCES in the file adsp_proc\Sensors\dd\build\dd.scons:

   ```
   if env.has_key('BRINGUP_8974'):
      DD_8660_FFA_SOURCES = [
      "${BUILDPATH}/sns_dd_<new_sensor_model>.c",
   ….
   ….
   ]
   ```

   Ensure that only one sensor driver for each sensor type is included in the build. For example, for the accelerometer, if MPU6050 is included, do not include any other accelerometer driver files, such as LIS3DH.

NOTE: Ensure the driver source file is added in the path <root>\dsps_proc\core\sensors\dd\src\.

5.3 Configure the required sensors in the arm7.scons file

1. For MSM8960 LA 1.0, 1.04, 1.5, and 1.7 releases prior to 1720, and APQ8064 LA 1.0/ 1.1 releases prior to 1122, in the file <root>\dsps_proc\core\bsp\sensorsimg\build\arm7\ arm7.scons, add the line:

   ```
   env.Append(CPPDEFINES = [”CONFIG_USE_<NEW_SENSOR_MODEL>”])
   ```

2. For MSM8960 LA 1.7 1720 and subsequent releases, and APQ8064 1.0/1.1 1122 and subsequent releases, in the file <root>\dsps_proc\core\bsp\sensorsimg\build\arm7\ arm7.scons, add the new driver configuration line under the “else” section of the following structure:

   ```
   if ‘USES_SENSORS_GROUP_REFERENCE’ in env:
   # Not here
   elif ‘USES_SENSORS_GROUP_ALTERNATE’ in env:
   # Not Here
   else:
   # Please add here
   env.Append(CPPDEFINES = [“CONFIG_USE_<NEW_SENSOR_MODEL>”])
   ```

   Only *one* sensor driver type, e.g., an STM accelerometer, LIS3DH, is supported in the DSPS build configuration at a time. Therefore, if a new sensor driver is added, the default configuration must be disabled by commenting out the line:

   ```
   #env.Append(CPPDEFINES = [”CONFIG_USE_LIS3DH”])
   ```

3. The configuration option is not available until MSM8974 LA 1.0 1014; make sure only one device driver file of each sensor type is included in the build as mentioned in Step 3 of Section 5.2.

5.4 Configure the registry

The file <root>\dsps_proc\core\sensors\smgr\src\8960\sns_smgr_sensor_config.h defines the driver-specific configurations, e.g., I2C address, and the file is organized in sections of different sensor types.

To replace a default sensor driver with a different driver:

1. Locate the appropriate section in the file.
2. Clone the section from an existing sensor of the same type.
3. Modify the configuration parameters as follows:

```
#ifdef (CONFIG_USE_<NEW_SENSOR_MODEL>)
  #define SNS_SMGR_SENSOR_<INDEX>_ENUM_CODE            <SENSOR_ID>

  #define SNS_SMGR_SENSOR_<INDEX>_DD_FN_LIST_PTR       &sns_
<new_sensor_model>_driver_fn_list

  #define SNS_SMGR_SENSOR_<INDEX>_DEVICE_ID            0

  #define SNS_SMGR_SENSOR_<INDEX>_BUS_ADDRESS        <7-bit I2C address>

  #define SNS_SMGR_SENSOR_<INDEX>_DATA_TYPE_1        <SENSOR_TYPE>

  #define SNS_SMGR_SENSOR_<INDEX>_DATA_TYPE_2        <SENSOR_TYPE>

  #define SNS_SMGR_SENSOR_<INDEX>_RANGE_TYPE         SNS_SMGR_DATA_
TYPE_PRIMARY_V01

#define SNS_SMGR_SENSOR_<INDEX>_SENSITIVITY_DEFAULT  <0..3>
  #define SNS_SMGR_SENSOR_<INDEX>_FLAGS              SNS_SMGR_NO_
SENSITIVITY

  #define SNS_SMGR_SENSOR_<INDEX>_OFF_TO_IDLE        <power on time in
microseconds>

  #define SNS_SMGR_SENSOR_<INDEX>_IDLE_TO_READY      <standby to ready
in microseconds>
  #define SNS_SMGR_SENSOR_<INDEX>_REG_ITEM_TYPE      SNS_SMGR_REG_
ITEM_TYPE_NONE

  #define SNS_SMGR_SENSOR_<INDEX>_REG_ITEM_ID        0

  #define SNS_SMGR_SENSOR_<INDEX>_CAL_PRI_TYPE       <CALIBRATION_
PRIORITY_TYPE>
```

```
  #define SNS_SMGR_SENSOR_<INDEX>_CAL_PRI_ID          <CALIBRATION_
PRIORITY_ID>
  #define SNS_SMGR_SENSOR_<INDEX>_GPIO_FIRST          <FIRST_GPIO_NUM>
  #define SNS_SMGR_SENSOR_<INDEX>_GPIO_SECOND         <SEC_GPIO_NUM>
#endif
```

Table 5-2 shows the sensor fields.

Table 5-2 Sensor fields

Field	Possible values
<INDEX>	Device Driver Framework (DDF) index for: ▪ 0 – Accelerometer ▪ 1 – Gyroscope ▪ 2 – Magnetometer ▪ 3 – Pressure ▪ 4 – Proximity/Ambient Light
<SENSOR_ID>	DDF Macro for Sensor ID: ▪ SNS_SMGR_ID_ACCEL_V01 – Accelerometer ▪ SNS_SMGR_ID_GYRO_V01 – Gyroscope ▪ SNS_SMGR_ID_MAG_V01 – Magnetometer ▪ SNS_SMGR_ID_PRESSURE_V01 – Pressure ▪ SNS_SMGR_ID_PROX_LIGHT_V01 – Proximity/Ambient Light
<SENSOR_TYPE>	DDF Macro for Sensor Type: ▪ SNS_DDF_SENSOR_ACCEL – Accelerometer ▪ SNS_DDF_SENSOR_GYRO – Gyroscope ▪ SNS_DDF_SENSOR_MAG – Magnetometer ▪ SNS_DDF_SENSOR_PRESSURE – Pressure ▪ SNS_DDF_SENSOR_PROXIMITY – Proximity ▪ SNS_DDF_SENSOR_AMBIENT – Ambient Light ▪ SNS_DDF_SENSOR__NONE – No sensor connected
<CALIBRATION_PRIORITY_TYPE>	See enums in <root>\dsps_proc\core\sensors\api\sns_reg_api_v01.h.
<CALIBRATION_PRIORITY_ID>	See enums in <root>\dsps_proc\core\sensors\api\sns_reg_api_v01.h.
<FIRST_GPIO_NUM>	The GPIO number is used for Data Ready Interrupt. If a different GPIO pin is to be used, change the number here. If no GPIO is assigned, use 0xFFFF.
<SECOND_GPIO_NUM>	Reserved for future purposes. Set to 0xFFFF.

For MSM8974, include the bus instance on which the sensor is connected:

```
#define SNS_SMGR_SENSOR_<INDEX>_BUS_INSTANCE              <Instance#>
```

Bus instances values are given in Table 5-3 (see *Presentation: MSM8974 Snapdragon™ Sensors Core (SSC) Deep Dive* (80-NA157-92)).

Table 5-3 Bus instances

BLSP	GPIO	Sensors connected	Bus instance
BLSP_I2C_SDA(**3**)	GPIO_10	▪ Light/Proximity – APDS9900 ▪ Pressure – BMP180 ▪ Mag. – AK8963	3
BLSP_I2C_SCL(3)	GPIO_11		
BLSP_I2C_SDA(12)	GPIO_87	▪ Accel and Gyro – MPU6050 ▪ (Accel – LIS3DH, not enabled by default in software)	12
BLSP_I2C_SCL(**12**)	GPIO_88		

A References

A.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
Application Note: Software Glossary for Customers	CL93-V3077-1
Presentation: MSM8974 Snapdragon™ Sensors Core (SSC) Deep Dive	80-NA157-92
Presentation: Sensors Deep Dive for MSM8994, MSM8992, MSM8952	80-NM328-74

A.2 Acronyms and terms

Acronym or term	Definition
APSS	Application Processor Subsystem Software
DRI	Data-Ready-Interrupt
LA	Linux Android
SAM	Sensors Algorithm Manager
SLPI	Sensor Low Power Island
SMGR	Sensors Manager
SSC	Snapdragon Sensors Core
SSI	Sensors Single Image
UUID	Universally Unique Identifier