Applicable platform:

SDM670/SDM845

Issue/problem description:

There is no orientation sensor on SDM670/SDM845. However, many old apk are still using it.

Solution :

Create orientation.cpp under *vendor\qcom\proprietary\sensors-see\sensors-hal\sensors* and copy below code to it. we should be able to see orientation sensor after sensor hal is rebuilt.

```
/*
* Copyright (c) 2017 Qualcomm Technologies, Inc.
* All Rights Reserved.
* Confidential and Proprietary - Qualcomm Technologies, Inc.
*/

#include "sensor_factory.h"
#include "ssc_sensor.h"
#include "sns_std_sensor.pb.h"

static const char *SSC_DATATYPE_ROTV = "rotv";

#define RAD2DEG 180.0f/M_PI
#define M_PI 3.14159265358979323846
#define MIN_FLT_TO_AVOID_SINGULARITY 0.0001f

class orient : public ssc_sensor
{
public:
orient(sensor_uid suid, sensor_wakeup_type wakeup);
static const char* ssc_datatype() { return SSC_DATATYPE_ROTV; }

private:
virtual void handle_sns_std_sensor_event(
const sns_client_event_msg_sns_client_event& pb_event) override;

void QuatToRotMat(float rot_mat[9], float quat[4]);
void RotMatToOrient(float values[3], float rot_mat[9]);

};
```

```
orient::orient(sensor_uid suid, sensor_wakeup_type wakeup):
ssc_sensor(suid, wakeup)
{
set_type(SENSOR_TYPE_ORIENTATION);
set_string_type(SENSOR_STRING_TYPE_ORIENTATION);
}

static bool orient_module_init()
{
/* register supported sensor types with factory */
sensor_factory::register_sensor(
SENSOR_TYPE_ORIENTATION,
ssc_sensor::get_available_sensors<orient>);
sensor_factory::request_datatype(SSC_DATATYPE_ROTV);
return true;
}
void orient::handle_sns_std_sensor_event(
const sns_client_event_msg_sns_client_event& pb_event)
{
sns_std_sensor_event pb_sensor_event;
pb_sensor_event.ParseFromString(pb_event.payload());
sensors_event_t hal_event = create_sensor_hal_event(pb_event.timestamp());
float quat[4];
float rot_mat[9];
float orient[3];


quat[0] = pb_sensor_event.data(0);
quat[1] = pb_sensor_event.data(1);
quat[2] = pb_sensor_event.data(2);
quat[3] = pb_sensor_event.data(3);

QuatToRotMat(rot_mat, quat);
/* convert rotation matrix to orientation */
RotMatToOrient(orient, rot_mat);


hal_event.orientation.x = orient[0];
hal_event.orientation.y = orient[1];
hal_event.orientation.z = orient[2];
hal_event.orientation.status = pb_sensor_event.status();
```

```
submit_sensors_hal_event(hal_event);
}
/*===========================================================================
=
FUNCTION: QuatToRotMat
Convert quaternion to rotation matrix

Quaternion
Q = |X Y Z W|

Rotation Matrix
/ R[ 0] R[ 1] R[ 2] 0 \
| R[ 4] R[ 5] R[ 6] 0 |
| R[ 8] R[ 9] R[10] 0 |
\ 0 0 0 1 /

M = 1- 2(Y*Y + Z*Z) 2XY - 2ZW 2XZ + 2YW 0
2XY + 2ZW 1 - 2(XX + ZZ) 2YZ - 2XW 0
2XZ - 2YW 2YZ + 2XW 1 - 2(XX + ZZ) 0
0 0 0 1
===========================================================================
*/
void orient::QuatToRotMat(float rot_mat[9], float quat[4])
{
float X = quat[0];
float Y = quat[1];
float Z = quat[2];
float W = quat[3];

float xx = X * X;
float xy = X * Y;
float xz = X * Z;
float xw = X * W;
float yy = Y * Y;
float yz = Y * Z;
float yw = Y * W;
float zz = Z * Z;
float zw = Z * W;

sns_logv("%s: X=%f, Y=%f, Z=%f, W=%f", __FUNCTION__, X, Y, Z, W);

rot_mat[0] = 1 - 2 * ( yy + zz );
rot_mat[1] = 2 * ( xy - zw );
```

```
rot_mat[2] = 2 * ( xz + yw );
rot_mat[3] = 2 * ( xy + zw );
rot_mat[4] = 1 - 2 * ( xx + zz );
rot_mat[5] = 2 * ( yz - xw );
rot_mat[6] = 2 * ( xz - yw );
rot_mat[7] = 2 * ( yz + xw );
rot_mat[8] = 1 - 2 * ( xx + yy );
}

/*=============================================================================
=

FUNCTION: RotMatToOrient
Convert rotation matrix to Orientation Sensor as defined in Sensor.TYPE_ORIENTATION:

values[0]: Azimuth, angle between the magnetic north direction and the y-axis,
around the z-axis (0 to 359). 0=North, 90=East, 180=South, 270=West

values[1]: Pitch, rotation around x-axis (-180 to 180),
with positive values when the z-axis moves toward the y-axis.

values[2]: Roll, rotation around y-axis (-90 to 90),
with positive values when the x-axis moves toward the z-axis.

=============================================================================
*/
void orient::RotMatToOrient(float values[3], float rot_mat[9])
{
float xunit[3] = {rot_mat[0], rot_mat[3], rot_mat[6]};
float yunit[3] = {rot_mat[1], rot_mat[4], rot_mat[7]};
float zunit[3] = {rot_mat[2], rot_mat[5], rot_mat[8]};
float xnorm = sqrt(xunit[0]*xunit[0] + xunit[1]*xunit[1]);

if (fabs(zunit[2]) < MIN_FLT_TO_AVOID_SINGULARITY) {
zunit[2] = MIN_FLT_TO_AVOID_SINGULARITY * (zunit[2] < 0 ? -1 : 1);
}

if (fabs(xunit[0]) < MIN_FLT_TO_AVOID_SINGULARITY) {
xunit[0] = MIN_FLT_TO_AVOID_SINGULARITY * (xunit[0] < 0 ? -1 : 1);
}

if (fabs(xnorm) < MIN_FLT_TO_AVOID_SINGULARITY) {
xnorm = MIN_FLT_TO_AVOID_SINGULARITY * (xnorm < 0 ? -1 : 1);
}
```

```
values[0] = RAD2DEG * atan2(xunit[1], xunit[0]);
values[0] = fmodf(360.0f - values[0], 360.0f);
values[1] = -RAD2DEG * atan2(yunit[2], zunit[2]);
values[2] = RAD2DEG * atan2(xunit[2], xnorm);
}

SENSOR_MODULE_INIT(orient_module_init);
```