SENtral™-A2

Motion Coprocessor for Android™

General Description

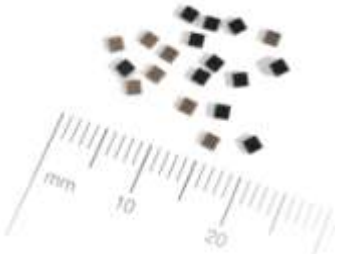
SENtral-A2 is an ultra-low power sensor hub to support all Android-compliant sensors, providing a full solution of processing plus algorithm in one tiny and efficient package.

SENtral-A2 embeds the complete range of Android sensor features in hardware on a low-power coprocessor, allowing wearable OEMs to support always-on context aware applications such as Google Now - without having to worry about power consumption.

SENtral-A2 offers a single solution for both enabling hardware and software for Android and "always-on" operation of the highest performance motion, context and location algorithms.

Compared to FLASH-based sensor hub microcontrollers (MCUs) with on-board floating point units (FPUs), SENtral-A2 operates at < 1/10 power. It can routinely run over 140'000 floating point operations per second at ~200 µAmps average current consumption at 1.8V.

SENtral-A2 accommodates up to 6 different sensor inputs and is the only sensor hub in the market that can run all the Android sensors simultaneously for under 200 µAmps.



Features

- Heading Accuracy of up to 2° rms.
- Ultra Low Power Consumption
- Continuous Soft and Hard-Iron Magnetic Calibration
- Magnetic Anomaly Compensation
- Small Form-Factor
- Sensor Flexibility
- Simultaneous output of all Android Sensor Functions at 200Hz
- Dual-buffering

Applications

- Wearables
- Smartphones & Tablet
- 3D pointers, Air Mice
- Motion and gesture recognition
- Location based services
- AR/VR
- Gaming and TV remotes

Ordering Information

Item	Part #	Quantity	Package
SENtral-A2	13896P	<4000	Cut-Tape
SENtral-A2	13896	4000	Tape & Reel

Table of Contents

1 DEVICE CHARACTERISTICS

The following sections describe the hardware, electrical, timing characteristics of the SENtral-A2 chip.

NOTE: Unless otherwise noted "SENtral" shall mean SENtral-A2 in this document.

2 BLOCK DIAGRAM

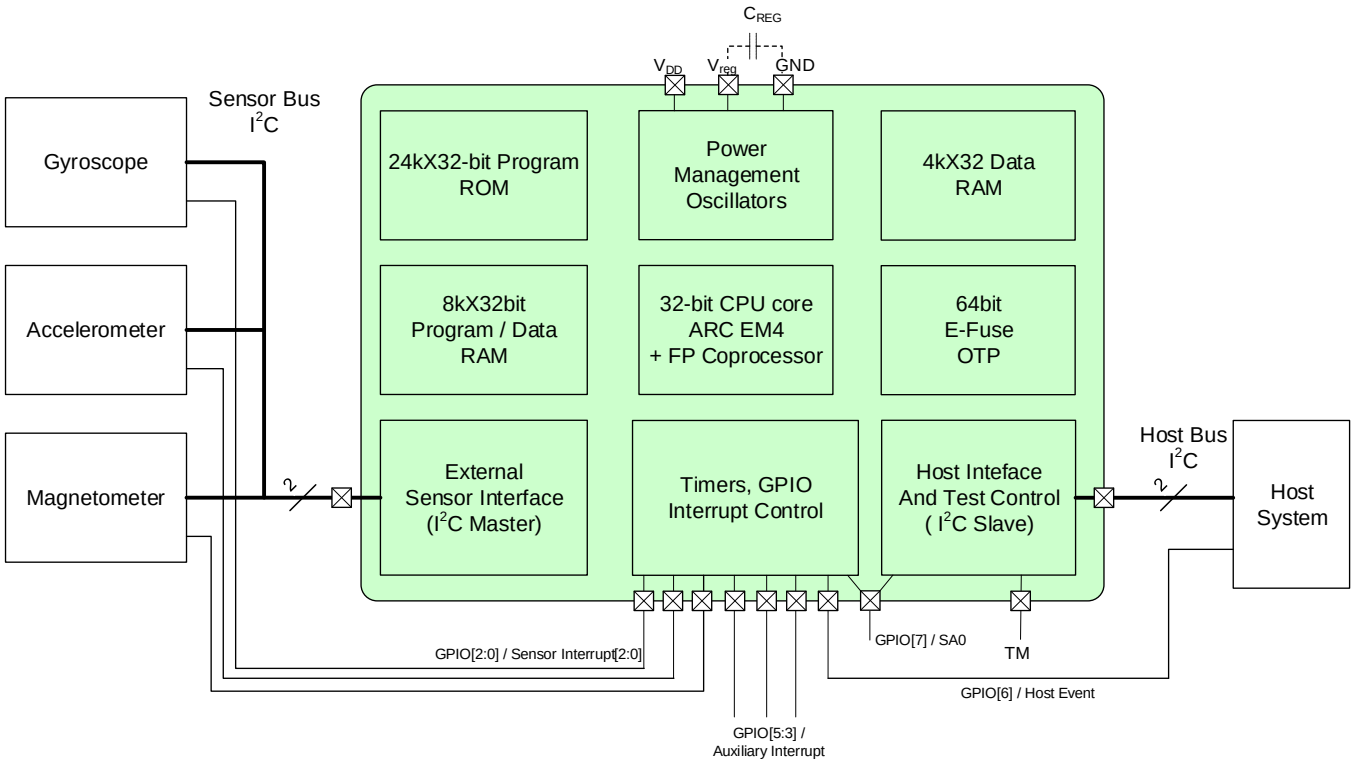


Figure 1: SENtral-A2 Block Digarm

2.1 ABSOLUTE MAXIMUM RATINGS

Stresses above these listed maximum ratings may cause permanent damages to the device. Exposure beyond specified operating conditions may affect device reliability or cause malfunction.

Table 1: Absolute Maximum Ratings

Parameter	Symbol	Conditions
Supply Voltage	V_{DD}	-0.3V to +3.6V
Voltage at remaining pin	V_{PIN}	-0.3V to V_{dd} +0.3V
Storage Temperature	T_{STORE}	-50°C to +150°C

2.2 HANDLING PROCEDURES

This device has built-in protection against high static voltages or electric fields; however, anti-static precautions must be taken as for any other CMOS component. Unless otherwise specified, proper operation can only occur when all terminal voltages are kept within the voltage range. Unused inputs must always be tied to a defined logic voltage level.

2.3 OPERATING CONDITIONS

Table 2: Operating Conditions

Parameter	Symbol	Min.	Max.	Unit
Supply Voltage	V_{DD}	1.6	+3.3	V
Operating Temperature	T_{OP}	-40	+85	°C
Regulator Decoupling Capacitor (ESR < 2Ω)	C_{REG}	0.33	1.8	µF

2.4 ELECTRICAL CHARACTERISTICS

Table 3: Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Power on Reset Threshold	V_{POR}	$V_{REG}>V_{POR}$		V_{REG}-125mV		V
High Level Input Voltage	V_{IH}		0.7*$_{VDD}$		V_{DD}	V
Low Level Input Voltage	V_{IL}		0		0.3*V_{DD}	V

Low Level Output Current	I_{OL}	V_O = 0.3V	1			mA
High Level Output Current	I_H	V_O=V_{DD}-0.3V			-1	mA
Current Consumption, Run[1]	I_{RUN}	0°C to + 40°C, (1)		800		µA
Current Consumption, Normal Operation[2]	I_{OPER}	0°C to + 40°C, (2)		350		µA
Current Consumption, Sleep[3]	I_{SLEEP}	0°C to + 40°C, (3)		40		µA
Current Consumption, Deep Sleep[4]	I_{DSLEEP}	0°C to + 40°C, (4)		7		µA
Current Consumption, Idle[5]	I_{IDLE}	0°C to + 40°C, (5)		6		µA

Notes:
Current consumption when CPU is running and executing from ROM.
Current consumption in normal operation is average consumption when Gyro rate is 200 Hz, Accel rate is 100 Hz, Mag rate is 30 Hz
Sleep mode is entered when CPU and I2C are idle and TIMOSC and SYSOSC are enabled
In Deep Sleep mode, only TIMOSC is enabled while SYSOSC is disabled
In Idle mode, no operations are performed, all oscillators are disabled

2.5 TIMING CHARACTERISTICS

Table 4: Timing Characteristics

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
System Clock Frequency (SYSOSC)	F_{SYS}		9	10	11	MHz
Timing Clock Frequency (TIMOSC)	F_{TIM}		124	128	132	KHz
I^2C Slave Port Operating Frequency	F_{SCLS}	(1)(2)	100		2000	KHz
I^2C Master Port Operating Frequency	F_{SCLM}	(1)(3)	100		1000	KHz
I^2C Operating Frequency Pass-Through Mode	F_{SCLPT}	(1)(4)	100		400	KHz

Notes:

1. I^2C Protocol Implementation is compliant with UM10204 I^2C -bus Specification/User Manual, Rev. 04, Feb 13th, 2012
2. I^2C Slave port supports Standard, Fast, Fast+ and High Speed Mode. High Speed Mode is supported with a reduced range of VDD and bus capacitance. Host I^2C transfer speed during continuous FIFO reads must be limited to 2Mb/s.
3. I^2C Master port supports Standard, Fast and Fast+ Mode

4. In pass-through mode, I²C Slave and Master ports form a single I2C bus. Standard & Fast modes are supported.

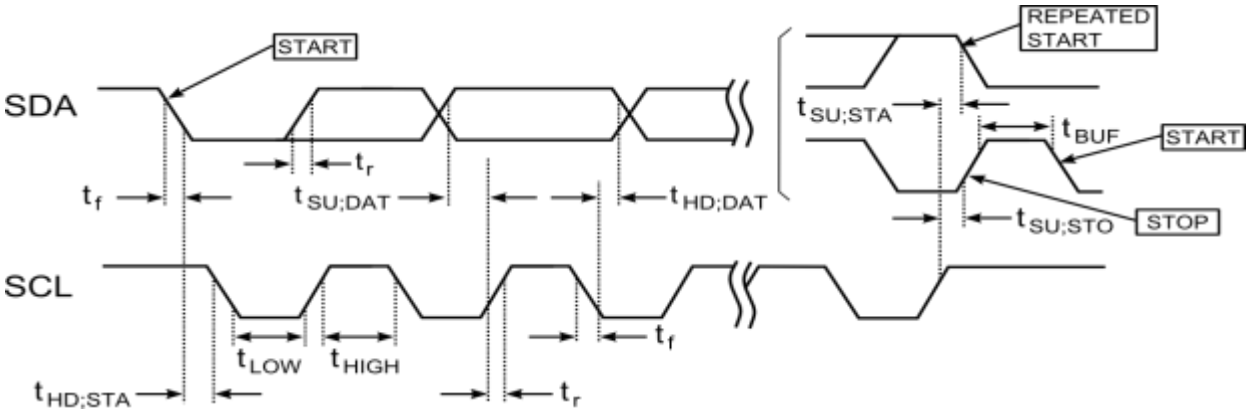


Figure 2: I2C Timing

Table 5: I²C Timing Parameters

Parameter	Symbol	Standard		Fast		Fast Plus		High Speed		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
SCL Clock	f_{SCL}	0	100	0	400	0	1000	0	2000	kHz
SDA & SCL Rise Time	t_r	-	1000	20	300		120	10	40_{SCL} 80_{SDA}	ns
SDA & SCL Fall Time	t_f	-	300	20 * (V_{DD}/5.5V)	300	20 * (V_{DD}/5.5V)	120	10	40_{SCL} 80_{SDA}	ns
LOW period of SCL Clock	t_{LOW}	4.7	-	1.3	-	0.5	-	0.16	-	µs
HIGH period of SCL Clock	t_{HIGH}	4.0	-	0.6	-	0.26	-	0.16	-	µs
Hold Time (repeated) START	$t_{HD;STA}$	4.0	-	0.6	-	0.26	-	0.16	-	µs
Data Hold time	$t_{HD;DAT}$	0	-	0	-	0	-	0	0.7	µs
Data set-up time	$t_{SU;DAT}$	250	-	100	-	40	-	10	-	ns
Set-Up time for repeated Start	$t_{SU;STA}$	4.7	-	0.6	-	0.26	-	0.16	-	µs
Stop set-up time	$t_{SU;STA}$	4.0	-	0.6	-	0.26	-	0.16	-	µs
Bus free time	t_{buf}	4.7	-	1.3	-	0.5	-	1.3	-	µs

between STOP & START										

3 FUNCTIONAL DESCRIPTION

3.1 PRODUCT OVERVIEW

SENtral-A2 is an integrated circuit that makes it easy to quickly integrate, optimize and operate multiple sensors on mobile consumer electronics devices. SENtral-A2 manages and uses data from a list of specified sensors defined under the Android operating system, including physical and virtual sensors.

SENtral-A2 provides reliable motion tracking and an accurate compass heading using 3-axis accelerometer, gyroscope, and magnetometer sensors (9 axes total), while consuming a fraction of the power of a comparable sensor fusion microprocessor. The primary data outputs are quaternions, which uniquely define device orientation, or Euler angles (heading, pitch, and roll). The quaternions easily can be converted to Euler angles, the rotation vector, and the rotation matrix.

SENtral-A2 is the interface between off-chip sensors and a host MCU, but it can also operate as a standalone part in some applications. Its processing architecture is designed to support sensor fusion algorithms and is particularly optimized for battery operated low power applications.

SENtral-A2 is typically connected as an I²C bus master to an external accelerometer, an external gyroscope and an external magnetometer. It polls the individual sensors in response to data ready interrupt signals from them – integrating, fusing and filtering their raw data. Then it buffers results to output sensor data to the host MCU through a programmable Slave I²C bus. The amount of buffering (batching) varies based on the exact combination of inertial sensors in use.

4 FEATURES AND BENEFITS

Features and benefits of the SENtral-A2 include:

- **Low power consumption.** Offloads sensor processing from the less efficient host CPU, consuming <10% of the power of a standard Cortex CPU running a comparable sensor fusion algorithm. Provides the ability to tailor the tradeoff between power consumption and motion-tracking performance.
- **Heading accuracy.** Unparalleled heading accuracy for consumer electronics applications.
- **Continuous hard and soft-iron magnetic auto-calibration.** Provides continual background calibration of the sensors.

- **Magnetic anomaly compensation.** Heading and motion tracking is unaffected by magnetic anomalies such as rebar in buildings, desks, speakers etc., that can easily throw off the accuracy. SENtral-A2 recognizes and compensates for these anomalies.
- **Sensor flexibility.** Works with the most common consumer electronic MEMS motion sensors, so system designers can choose the sensors most appropriate for their systems.
- **Small form-factor.** 1.7x1.68x0.5 mm chip-scale package on 0.4 mm pitch. Uses little PCB real estate, allowing for painless integration.
- **I^2C interface.** Uses the industry-standard I^2C protocol in a proprietary low-power implementation to interface to the sensors and the host, so system integration is straightforward. Standard, Fast, Fast Plus, and High Speed are supported on the host bus.
- **Outputs.** SENtral-A2 natively outputs quaternions, rotational velocity, linear acceleration, and magnetic field.
- **Virtual Sensors.** The full set of Android virtual sensors is supported. These are described in section 9, and include: Proximity, Tilt detection, step detector, step counter, and many others. Each virtual sensor has an independently programmable sample rate and batching timeout.

4.1 FUNCTIONAL BLOCKS

The block diagram below shows the primary blocks, a brief description of these functional blocks follows.

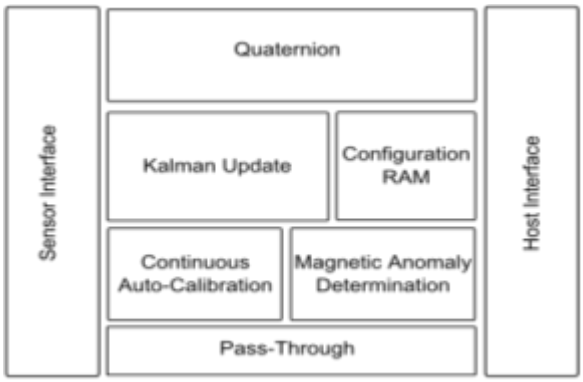


Figure 3: SENtral-A2 Functional Blocks

- **Quaternion** provides the orientation output and is updated at a rate limited to the gyro output data rate (ODR), up to a maximum of 200 Hz.
- **Kalman Update** fuses data from the 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer, plus data from the magnetic anomaly determination and continuous auto-calibration blocks to generate intelligent orientation updates. The Kalman update involves an advanced multi-state Kalman algorithm.
- **Continuous Hard and Soft-Iron Auto-Calibration**. SENtral-A2 is the only product in the market that auto-calibrates for both hard-iron and soft-iron magnetic distortions. While others may

calibrate for hard-iron distortion, soft-iron distortion is more difficult to correct for, and it can be caused by EMI shielding tape and other shielding materials widely used in mobile and consumer electronic devices. It is important to correct for soft-iron distortions since they can contribute up to 90° of error. Additionally, since a host system's magnetic signature can change over time and temperature, SENtral-A2's continuous auto-calibration ensures accuracy all the time.

- **Magnetic Anomaly Determination** establishes if a transient magnetic distortion is present and accounts for it.
- **Configuration RAM** allows for customizing SENtral to match the specific sensors being used and allows the user to tailor certain parameters for their specific system. This is where the physical and many virtual sensor drivers are located.
- **Pass-Through** allows for direct communication with devices on the sensor bus by connecting SENtral's I^2C Host Interface to the Sensor Interface.
- **Host Interface** communicates with the host system. Data is transmitted between the host and SENtral via the host I^2C bus, in which the host acts as the master and SENtral acts as a slave device. Sensor data transfer follows the messaging specifications under Android and operates in batch or non-batch modes.
- **Sensor Interface** communicates primarily with the sensors. Sensor data is transmitted from the sensors to SENtral via the sensor I^2C bus, in which SENtral acts as the master and the sensors as the slave devices.

4.2 I^2C SERIAL INTERFACE

Communication with the host processor and sensors is via an I2C interface and interrupt lines. The SENTRAL-A2 Motion Coprocessor acts as the I2C master with the sensors and as a slave with the host processor.

4.2.1 EXTERNAL SENSOR INTERFACE – I2C MASTER

SENtral communicates with the accelerometer, gyroscope, and magnetometer and other sensors over the sensor bus, where SENtral acts as the I^2C master and the sensors act as the I^2C slaves. On the sensor bus, SENtral initiates data transfer and generates the serial clock. SENtral's I^2C sensor interface supports Standard mode with a rate up to 100 kbit/s, Fast mode with a rate up to 400 kbit/s, and Fast Plus mode with a rate up to 1000 kbit/s.

The two wires comprising the sensor bus are SDAM, the serial data line, and SCLM, the serial clock. Both are bidirectional and driven by open drain transistors within SENtral. Each line should be attached to a pull-up resistor.

The I^2C Master interface supports the features and the I^2C clock speeds listed in Table 6 below.

Table 6: I²C Master Interface Features

Feature	Single Master
START condition	YES
STOP condition	YES
Acknowledge	YES
Synchronization	N/A
Arbitration	N/A
Clock stretching	YES
7-bit Slave address	YES
10-bit Slave address	NO
General Call address	NO
Software reset	NO
START byte	N/A
Device ID	N/A
Burst length	up to 16-bytes

4.2.2 TRANSFER FORMATS – I2C MASTER

The following sequence describes how firmware built into the SENTRAL-A2 writes to a slave device attached to the I²C Master interface. An example of a write sequence is in Figure 4 and Figure 5 , below.

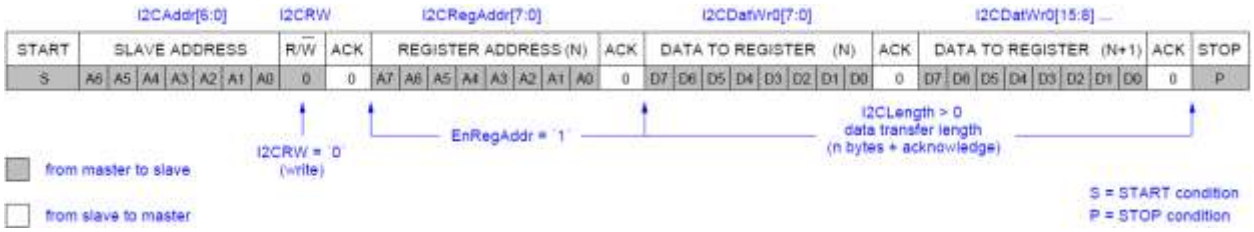


Figure 4: Master Write Sequence

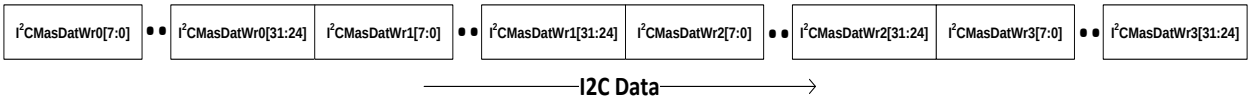


Figure 5: I²C MasDatWrx Register Mapping to I²C Data

The following sequence describes how SENtral's firmware reads a slave device attached to the I²C Master interface. An example of a read sequence using repeated START conditions is in Figure 6 and Figure 7 below.

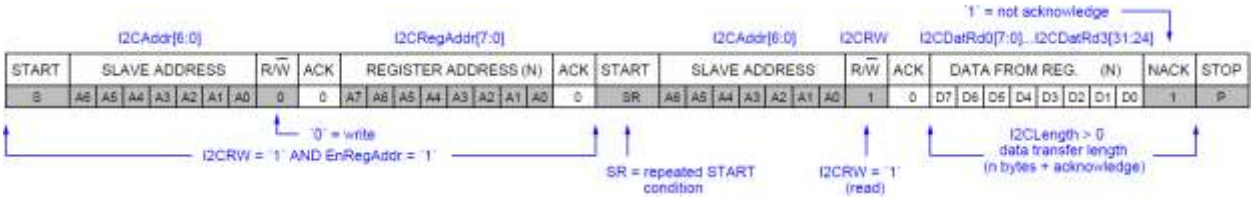


Figure 6: I²C Master Read Example, Combined Format with Repeated START

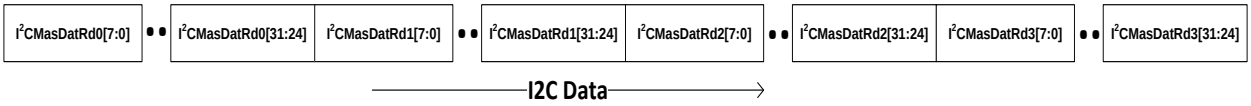


Figure 7: I²C Data Mapping to I2CMasDatWrx register

Other read sequences are possible such as sending the register address in a separate sequence than the data. Shown below

- Only the register address is written into the Slave device. Register I2CMasCtrl[I2CLength]=0 (Figure 8)
- After the register address is sent the Master can begin reading from this address (Figure 9)

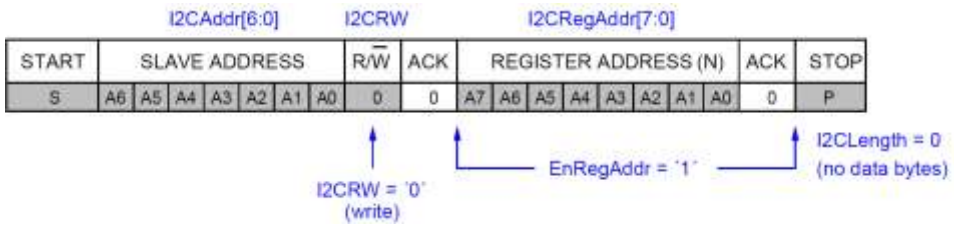


Figure 8: I2C Master Write Register Address Only

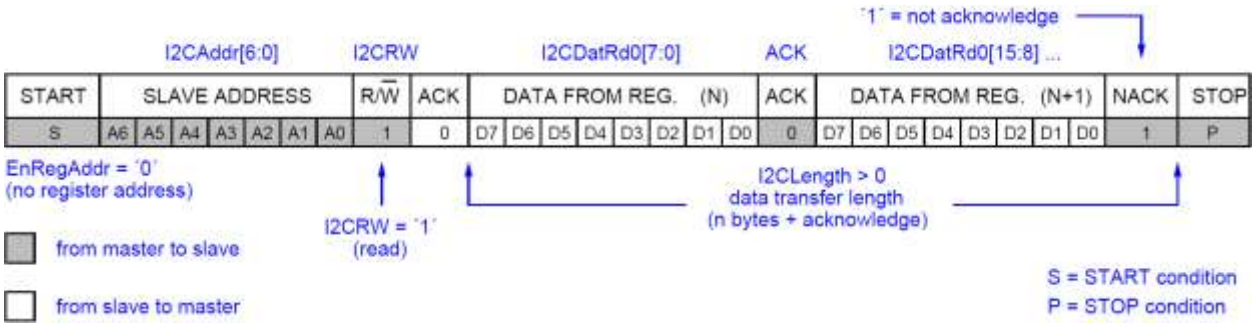


Figure 9: I²C Master Read from Current Address

4.2.3 EXTERNAL HOST INTERFACE – I2C SLAVE

The host controls SENTRAL-A2 on the host bus via SENTRAL-A2 's I2C host interface. The host interface consists of 2 wires: the serial clock, SCLS, and the serial data line, SDAS. Both lines are bi-directional. SENTRAL-A2 is connected to the host bus via the SDAS and SCLS pins, which incorporate open drain drivers within the device. The host bus lines must be externally connected to a positive supply voltage (DVIO) via a pull-up resistor. The I2C Slave interface supports the features listed in Table 7 below.

Table 7: I2C Slave Features

Feature	Slave
START condition	YES
STOP condition	YES
Acknowledge	YES
Synchronization	N/A
Arbitration	N/A
Clock stretching	No
7-bit Slave address	YES
10-bit Slave address	NO
General Call address	NO
Software reset	NO
START byte	N/A
Device ID	N/A

4.2.4 TRANSFER FORMATS – I²C SLAVE

Figure 10, below illustrates an example of how to write data to registers in single-byte or multiple-byte mode.

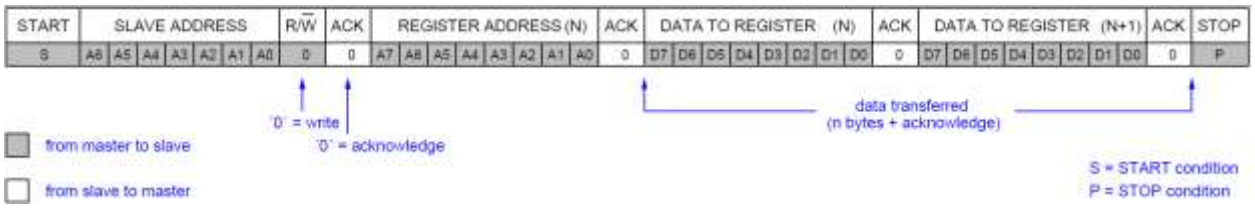


Figure 10: I²C Slave Write Example

The slave address consists of the fixed most significant bits A6:A1 = 010100 and the least significant bit A0 = pad SA0. If SA0 is pulled to ground, then the 7 bit slave address would be 0x28 (hexadecimal). If pulled up, it would be 0x29.

Similar to the I²C Master, the I²C Slave supports both a read sequence using repeated START conditions and a sequence in which the register address is sent in a separate sequence than the data.

Figure 11: I²C Slave Read Example, Combined Format with Repeated START

Figure 12: I²C Slave Write Register Address Only

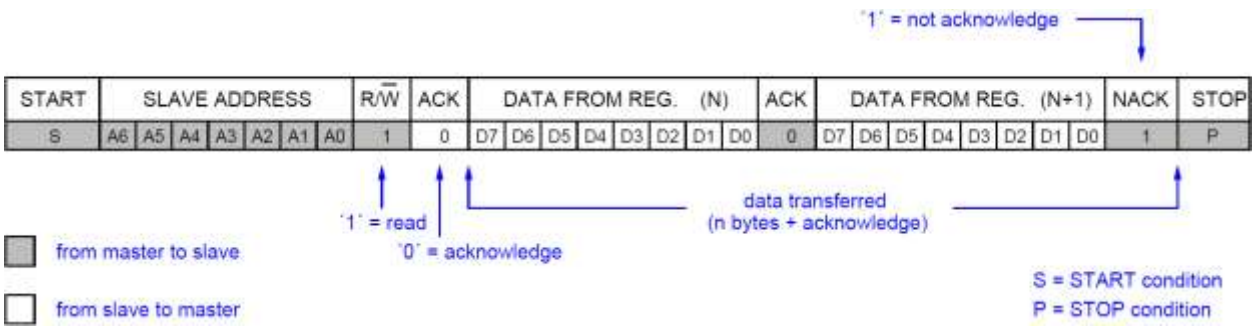


Figure 13: I²C Slave Read Register from Current Address

4.2.5 PASS-THROUGH MODE – I²C

The block diagram in Figure 3 shows the SENtral-A2 Motion Coprocessor with the Host and Sensor interface I²C buses. These I²C buses are independent in normal mode operation.

The Host I²C bus is connected to the I²C Slave interface of the SENtral through the two IO pins named SDAS and SCLS. The Sensor I²C bus is connected to the I²C Master interface of the SENTRAL-A2 through the two IO pins named SDAM and SCLM.

SENtral-A2 includes a Pass-through mode where the Host directly communicates with Sensors through the SENtral device. The I²C Slave interface of SENtral-A2 stays active and the Host has full access to the Slave register field in Pass-through mode.

4.2.6 I²C PULL-UP RESISTANCE

The pull-up resistor values for both host and sensor bus will depend on the I²C data rate and the number of devices on the bus. Table 8 provides the maximum acceptable bus capacitance, as a function of bus rate, which can be accommodated with a 4.7 kΩ or 2.4 kΩ pull-up resistor. As a general rule, each device connected to the bus represents 10 pF of capacitance on the bus, so a bus with 4 devices would require a "Max Cb" value of >40 pF.

Table 8: I²C Pull-Up Resistance Table

I²C Mode		Rate (Kbit/s)	Rise Time (ns)	Max Cb (pF)	
				4.7 kΩ pull-up	2.4 kΩ pull-up
Standard		100	1000	251.1	491.8
Fast		400	300	75.3	147.5
Fast Plus		1000	120	30.1	59.0

I²C Mode		Rate	Rise Time	Max Cb (pF)	
High Speed-1.7 MHz	Clock	1700	80	20.1	39.3
	Data	1700	160	40.2	78.7
High Speed-3.4 MHz	Clock	3400	40	10.0	19.7
	Data	3400	80	20.1	39.3

As the table implies, for most Standard and Fast Mode implementations a 4.7 kΩ pull-up should work well, while a 2.4 kΩ pull-up normally should be used for Fast Plus. See Section 7.1 of NXP's UM10204 specification for additional information.

5 HOST INTERFACE

This section describes the host interface, the mechanism by which the application processor (AP) of an Android device can communicate with the SENtral-A2. This same interface can be used by non-Android systems.

5.1 ANDROID FEATURES

Compared with earlier versions of Android, and compared with the capabilities of the original SENtral (PNI p/n 13658), Android dramatically expands the list of available sensors to include a number of virtual sensors as well as uncalibrated versions of the gyroscope and magnetometer.

Most sensors are continuous output sensors; the rate at which they output data to the host is determined by the host by means of a sample period setting.

Some sensors are on change sensors. These use the sample period setting to determine the fastest rate that changes should be sent to the host if the value actually changes that quickly; otherwise, they only update the host as the data changes. They must send a sample when they are first activated. In Android, on change sensors are protected from data loss if the non-wakeup FIFO overflows.

One shot sensors disable themselves and then send a single event upon detection of the event. They ignore the sample period.

1. Accelerometer
2. Magnetometer
3. Orientation (deprecated in Android SDK but not HAL; azimuth / pitch / roll)
4. Gyroscope (temperature compensated)

5. Light – on change
6. Barometer
7. Temperature* (deprecated)
8. Proximity – on change
9. Gravity
10. Linear Acceleration
11. Rotation Vector (9DOF)
12. Humidity – on change
13. Ambient Temperature* – on change
14. Magnetometer Un-calibrated
15. Game Rotation Vector (6DOF accelerometer + gyroscope)
16. Gyroscope Un-calibrated
17. Significant Motion – one shot
18. Step Detector – special (like on change but ignores sample rate)
19. Step Counter – on change
20. Geomagnetic Rotation Vector (6DOF accelerometer + magnetometer)
21. Heart Rate – on change
22. Tilt Detector - special
23. Wake Gesture – one-shot
24. Glance Gesture – one-shot
25. Pick Up Gesture – one-shot
26. Reserved
27. Reserved
28. Reserved
29. Reserved
30. Reserved
31. Activity (not an official sensor type, but used to report the separate activity state)

**Discouraged in the Android 4.4 Compatibility Definition, Revision 1, November 27, 2013, section 7.3.6. SENtral-A2 supports them anyway.*

Further, Android defines a mechanism for storing up, or batching, results for later host retrieval.

5.2 SENSOR OUTPUTS

The 9DOF quaternion sensors, such as Rotation Vector, use 16-bit fixed point integers for each vector sensor axis (X, Y, Z, W). The packing of other sensor types into three 16 bit unsigned integers is used for most of the rest of the sensors. The Light, Proximity, Ambient temperature, Humidity, Step Detector, and Step Counter use 16-bit signed or unsigned integers. Due to the high resolution required to measure vertical change in location to a resolution less than one floor in a building, Barometer readings are reported to 24 bits of resolution, which maps well to single precision floats as that is the size of the significand. The uncalibrated sensors include three additional 16 bit signed values (for a total of six).

Output data will be sent in a continuous stream, with each sensor sample uniquely identified. Because many sensors may end up producing output at the same time, the timestamp is only output once at the start of a series of sensor samples that occur at the same time.

Table 9: Sensor Types

Sensor Name	Sensor Value	Format
Rotation Vector Game Rotation Vector Geomagnetic Rotation Vector	X, Y, Z, W Quaternion Estimated Accuracy	16-bit fixed point 16-bit integer
Accelerometer Magnetometer Orientation Gyroscope Gravity Linear Acceleration	X, Y, Z Vector Status	16-bit signed integer, scaled to maximize dynamic range 8 bit unsigned integer indicating accuracy of measurement (low, medium, high, unreliable)
Light Proximity Humidity	Absolute Scalar	16-bit unsigned integer, scaled to maximize dynamic range
Step Counter	Absolute Scalar	16-bit unsigned integer
Temperature Ambient Temperature	Scalar	16-bit signed integer, scaled to maximize dynamic range
Barometer	Absolute Scalar	24 bit unsigned integer, scaled as required

Sensor Name	Sensor Value	Format
Significant Motion Step Detector Tilt Detector Wake Gesture Glance Gesture Pickup Gesture	Event	None
Uncalibrated Magnetometer Uncalibrated Gyroscope	Uncalib X, Y, Z; bias X, Y, Z Status	16-bit signed integer, scaled to maximize dynamic range 8-bit unsigned integer indicating accuracy of measurement (low, medium, high, unreliable) (not required by Android but nice to have)
Heart Rate	Scalar	8 bit beats per minute
Activity	Activity On or Off	2 bytes (bits to indicate each type of activity)

5.3 SENSOR CONFIGURATION

Android does not require the ability to set the dynamic range or resolution of any sensors. The only configurable items at the HAL to sensor driver interface are below. SENtral-A2 does provide a dynamic range setting for custom use.

- **Activate** – enable or disable the sensor: Boolean
- **Batch** – enable by setting the maximum report latency or disable by setting to zero: Integer (nanosecond resolution or larger)
- **Delay (Sample Period)** – must be equal or less than the requested value, but no smaller than the Delay / 2: Integer (nanosecond resolution or larger)
- **Flush** – send all batched samples (if any) immediately, then send a special meta data type indicating the flush is complete
- **Poll** – read a specified number of samples: Integer
- **Wakeup** – if a sensor is opened as a wakeup sensor (using a flag), then it is allowed to interrupt the host, either when each sample is available or after the maximum report latency; if opened as a non-wakeup sensor, then they must not interrupt the host. Data for the wakeup vs. non-wakeup sensors go in separate FIFOs.

Since the HAL / driver will be given the sample rate (period) and max report latency at the same time by higher levels, and since they map to 8 bytes of data, they can be sent to SENtral-A2 in a single 8 byte parameter write. Each unique sensor (physical or virtual) is assigned a unique Parameter number.

Writing this Parameter modifies the values; reading this Parameter returns the actual sample rate and actual report latency. The actual sample rate will be greater than or equal to the requested rate, but no greater than two times the requested rate. While this ability to query the actual sample rate is not required by Android, it is useful for external testing, debugging, and non-Android applications. If the requested sensor is not present, the returned sample rate and report latency will be 0.

The flush mechanism is done using a single 8 bit GP register. Android defines a special meta-data value which will be placed in the buffer to indicate the flush of a specific sensor has been completed.

5.4 SENSOR STATUS INFORMATION

Android sensor drivers need to make the following information available about each supported sensor:

- **Name** – unique; if there are multiple sensors in the system of the same type, each one must have a unique name; this can be derived from the SENtral sensor's driver ID and slave I^2C address
- **Vendor** – vendor of the underlying hardware; this can be derived from the SENtral sensor's driver ID
- **Version** – version of the hardware + driver; changes when the driver's output changes in some way; obtain from SENtral sensor's driver version plus driver-specific version information
- **Type** – sensor type, e.g., SENSOR_TYPE_ROTATION_VECTOR; same as sensor data packet type
- **maxRange** – the maximum possible sensor value in SI units (e.g., an accelerometer set for a 4g range would report 4g here); derived from sensor dynamic range
- **resolution** – smallest difference between two values reported by this sensor (e.g., an accelerometer at 4g range and 16 bit signed values could report 4g/32767 = 1.2xe-4 g; derived from sensor dynamic range and bits of resolution
- **power** – rough estimate of sensor's power consumption in mA; this appears to be only queried at reboot by Android, and is defined to be the maximum current consumed when in use

- **minDelay** – continuous sensors report minimum period in microseconds; on-change sensors report 0; one-shot sensors report -1
- **maxDelay** – continuous sensors report maximum period in microseconds
- **fifoReservedEventCount** – number of events reserved for this sensor in the batch FIFO; since a single FIFO for all sensors is used in SENtral-A2, this implies that no area is reserved specially for any one sensor, so this returns 0
- **fifoMaxEventCount** – maximum number of events that could be batched; since the FIFO is shared, this is the size of the FIFO in bytes divided by the number of bytes per sample
- **flags** – only one is defined – wakeup

In SENtral-A2, these are combined into the following fields:

- **Sensor Type**: unsigned 8 bits
- **Driver ID:** unsigned 8 bits
- **Driver Version:** unsigned 8 bits
- **Max Range:** signed 16 bits; scaled the same as the sensor's corresponding data value
- **Resolution:** signed 16 bits; number of bits per sensor (axis) sample
- **Power:** unsigned 8 bits; multiples of 0.1 milliamps
- **Max Rate:** unsigned 16 bits; rate in Hz
- **Min Rate:** unsigned 9 bits; rate in Hz
- **FIFO Reserved:** unsigned 16 bits; 0
- **FIFO Max:** unsigned 16 bits; total FIFO size in bytes divided by size of data value
- **Status Bits:** unsigned 8 bits; while not required by Android, for testing and debugging the SENtral-A2 provides bit flags indicating:
  - data_available – there is at least one sample of this sensor type in the FIFO
  - i2c_nack – the physical sensor underlying this sensor is not responding
  - device_id_error – the physical sensor does not match the loaded driver
  - transient_error – temporary error, e.g., magnetic transient
  - data_lost – at least one sample was lost due to FIFO overflow
  - sensor_power_mode – the sensor is shutdown, standby, low power active, high power active

NOTE: the Status Bits will change dynamically at run time, but the other fields are fixed and only read by the Android HAL once at boot. Therefore, these per-sensor Status Bits are available through a separate mechanism than the static information, via the sensor status banks, described in section 5.11.1.3.

The remaining fields add up to 14 bytes of information. These values, plus the size of a given sensor's sample in the FIFO, can be read by the host by reading the Sensor Information structure for a specific sensor, as described in section 5.11.4.

Since not all of the possible sensor types will be present in all builds (e.g., one system might have a barometer but another may not), the host can query this data for all possible sensors, and based on the returned data, know which sensors are present. If a sensor is not present, all status fields will be returned with a value of 0. Alternatively, the host can query the Sensor Status Bits for each sensor; those sensors that are not available will return 0 for the power mode, indicating sensor not present.

5.5 FIFOS

Since many sensors now have both wakeup and non-wakeup versions, Android requires a minimum of two FIFOs.

Each version of a sensor has independent sample rate and report latency values.

Each FIFO has independent watermarks, interrupt control, and flush requests.

Data will be output to the host as a sum of the wakeup FIFO followed by the non-wakeup FIFO, unless the output was triggered by a flush request, in which case, the type of sensor requested in the flush determines which FIFO is output.

Meta events not related to a specific sensor will only be placed in the non-wakeup FIFO. These meta events are Error, Self-Test Results, and Initialized. However, these meta events are enabled by default, and wakeup the AP by default.

Meta events related to a specific physical sensor, such as the Sample Rate Changed, Power Mode Changed, and Dynamic Range Changed, are always sent to the non-wakeup FIFO.

5.6 NON-BATCH MODE

As required, any sensor with a 0 latency or batching timeout will be reported as soon as it is detected. This will of necessity result in many small FIFO transfers and an interrupt rate as high as the fastest non-batched sensor's sample rate. This may in reality be even more often than one might expect, due to slight variations in sensor sample rates.

The host can minimize interrupts while ensuring timely transfer of sensor data by setting all but the fastest enabled sensor to have a non-zero latency, large enough to not timeout before the next sample of the fastest sensor. In this configuration, data transfers from the FIFO will occur at the rate of the fastest sensor, with all slower sensors transferred together.

For example, if the Accelerometer is set for 60 Hz, and the Gyroscope and Magnetometer are set to 15 Hz, and the Gyroscope and Magnetometer are each set with a latency timeout of 21 ms, host transfers will occur like this:

Accel 1
Accel 2
Accel 3, Gyro 1, Mag 1
Accel 4
Accel 5
Accel 6, Gyro 2, Mag 2
…

5.7 BATCH MODE

SENtral-A2 fully supports Android batching requirements. Each sensor type has an independently settable latency or batching timeout.

The batched sensor data and other meta data is stored in a RAM-based FIFO, the size of which depends on the exact constellation of sensor drivers included in the executing RAM patch.

SENtral-A2 implements a single shared wakeup FIFO for wakeup sensors, and a single shared non-wakeup FIFO for non-wakeup sensors. As such, whichever sensor's batching timeout expires first will cause all events in the FIFOs to be sent to the host.

SENtral-A2 supports a host-settable Watermark value for each FIFO, which is used to ensure that batched data is not lost due to FIFO overflow, when the AP is outside of suspend mode.

When the AP is in suspend mode, the FIFO is allowed to overflow. SENtral will discard the oldest data to make room for new data as it arrives. As soon as the AP leaves suspend mode, SENtral will request a transfer of the entire contents of both FIFOs to the host.

SENtral-A2 supports the optional WAKE_UPON_FIFO_FULL feature when the AP is suspended, though depending on how many sensors are enabled and what sample rates they are set to, there may be less than 10 seconds of batching time. This option can be enabled by the host by setting a Watermark value, enabling the Watermark Meta Event, and setting the Watermark Meta Event to trigger an interrupt when it occurs.

5.8 REGISTER MAP

All interaction between the host and SENtral-A2 is through the register map, summarized below. Each I^2C register is unidirectional, so the MCU Access and I^2C Access indicates which side can write and which is read only.

In general all registers default to 0 upon reset. ROM Version, Product ID, Revision ID, and certain actively updated status bits are the exception.

Table 10: SENtral-A2 Registers

Register Name	I^2C Address	ARC Access	I^2C Access
Data Transfer Area [00:03]	0-3	RW	RO
Data Transfer Area [04:07]	4-7	RW	RO
Data Transfer Area [08:11]	8-B	RW	RO
Data Transfer Area [12:15]	C-F	RW	RO
Data Transfer Area [16:17]	10-11	RW	RO
Data Transfer Area [18:19]	12-13	RW	RO
Data Transfer Area [20:21]	14-15	RW	RO
Data Transfer Area [22:23]	16-17	RW	RO
Data Transfer Area [24:25]	18-19	RW	RO
Data Transfer Area [26:27]	1A-1B	RW	RO
Data Transfer Area [28:29]	1C-1D	RW	RO
Data Transfer Area [30:31]	1E-1F	RW	RO
Data Transfer Area [32:33]	20-21	RW	RO
Data Transfer Area [34:35]	22-23	RW	RO
Data Transfer Area [36:37]	24-25	RW	RO
Data Transfer Area [38:39]	26-27	RW	RO
Data Transfer Area [40:41]	28-29	RW	RO
Data Transfer Area [42:43]	2A-2B	RW	RO
Data Transfer Area [44:45]	2C-2D	RW	RO
Data Transfer Area [46:47]	2E-2F	RW	RO
Data Transfer Area [48:49]	30-31	RW	RO
FIFO Flush	32	RO	RW

Register Name	I²C Address	ARC Access	I²C Access
Reserved	33	RO	RW
Chip Control	34	RO	RW
Host Status	35	RW	RO
Int Status	36	RW	RO
Chip Status	37	RW	RO
Bytes Remaining LSB	38	RW	RO
Bytes Remaining MSB	39	RW	RO
Parameter Acknowledge	3A	RW	RO
Saved Parameter Byte 0	3B	RW	RO
Saved Parameter Byte 1	3C	RW	RO
Saved Parameter Byte 2	3D	RW	RO
Saved Parameter Byte 3	3E	RW	RO
Saved Parameter Byte 4	3F	RW	RO
Saved Parameter Byte 5	40	RW	RO
Saved Parameter Byte 6	41	RW	RO
Saved Parameter Byte 7	42	RW	RO
Saved Parameter Byte 8	43	RW	RO
Saved Parameter Byte 9	44	RW	RO
Saved Parameter Byte 10	45	RW	RO
Saved Parameter Byte 11	46	RW	RO
Saved Parameter Byte 12	47	RW	RO
Saved Parameter Byte 13	48	RW	RO
Saved Parameter Byte 14	49	RW	RO
Saved Parameter Byte 15	4A	RW	RO
GP20	4B	RW	RO
GP21	4C	RW	RO
GP22	4D	RW	RO
GP23	4E	RW	RO
GP24	4F	RW	RO
Error Register	50	RW	RO

Register Name	I²C Address	ARC Access	I²C Access
Interrupt State	51	RW	RO
Debug Value	52	RW	RO
Debug State	53	RW	RO
Parameter Page Select	54	RO	RW
Host Interface Control	55	RO	RW
GP31	56	RO	RW
GP32	57	RO	RW
GP33	58	RO	RW
GP34	59	RO	RW
GP35	5A	RO	RW
GP36	5B	RO	RW
Load Parameter Byte 0	5C	RO	RW
Load Parameter Byte 1	5D	RO	RW
Load Parameter Byte 2	5E	RO	RW
Load Parameter Byte 3	5F	RO	RW
Load Parameter Byte 4	60	RO	RW
Load Parameter Byte 5	61	RO	RW
Load Parameter Byte 6	62	RO	RW
Load Parameter Byte 7	63	RO	RW
Parameter Request	64	RO	RW
GP46	65	RO	RW
GP47	66	RO	RW
GP48	67	RO	RW
GP49	68	RO	RW
GP50	69	RO	RW
GP51	6A	RO	RW
GP52	6B	RO	RW
Host IRQ Timestamp	6C-6F	RW	RO
ROM Version	70-73	RW	RO
	74-8F	No access	0

Register Name	I²C Address	ARC Access	I²C Access
Product ID	90	No access	RO
Revision ID	91	No access	RO
	92	No access	RO
	94-95	No access	RW
Upload Data	96	No access	RW
CRC Host	97-9A	No access	RO
Reset Request	9B	No access	RW
	9C-9D	No access	0
Pass-Through Ready	9E	No access	RO
SCL Low Cycles	9F	No access	RW
Pass-Through Config	A0	No access	RW
	A1-A2	No access	RW
	A3-AA	No access	RO
	AB-B5	No access	RW

5.9 REGISTERS

A full description of all SENtral-A2 registers follows.

5.9.1 0X00 TO 0X31: DATA TRANSFER AREA

This range of registers is used for data transfers from the FIFO to the host. Access to this area must be done in a specific manner as described in later sections. The general procedure is, however, that the host must read the Bytes Remaining registers to determine the current number of pending bytes in the FIFO, and then should read that many bytes from this register block.

5.9.2 0X32: FIFO FLUSH

This allows the host to request that a single sensor's FIFO (batch mode) be flushed, or all sensors. Since SENtral-A2 implements shared FIFOs, specifying a Sensor ID has the same effect as specifying 0xFF to flush all sensors. This is an optional mechanism; if the host does not use this register, then the watermark, sensor latency, and wakeup and non-wakeup FIFO interrupt disable bits, as well as which sensors are enabled, determine when the host interrupt occurs and whether the data stream will include both the wakeup and non-wakeup FIFOs.

Register Name	Register Address	Register Value
FIFO Flush	0x32	Sensor ID or 0xFF to flush all samples for both FIFOs 0x00: NOP 0xFE: clears out (discards, not sent to host) both FIFOs 0xFD: flushes (sends) the wakeup FIFO only 0xFC: flushes (sends) the non-wakeup FIFO only 0xFB: clears out (discards) the wakeup FIFO 0xFA: clears out (discards) the non-wakeup FIFO

5.9.3 0X34: CHIP CONTROL

This register provides bits that control fundamental behavior of the chip.

Register Name	Register Address	Bit Number	Register Value
Chip Control	0x34	0 1 2-7	CPU Run Request Host Upload Enable Reserved

- CPU Run Request controls whether the internal MCU is running or not
- Host Upload Enable controls the RAM patch upload mechanism; see Appendix A for details

5.9.4 0X35: HOST STATUS

Algorithm Standby will be set to confirm that the host's previous write of a 1 to the Algorithm Standby Request bit in the Host Interface Control register (described in section 5.9.13) has taken effect. Reset is set after power-on reset, watchdog reset, or reset invoked by the host by means of the Reset Request register.

Register Name	Register Address	Bit Number	Register Value
Host Status	0x35	0 1 2-4	Reset Algorithm Standby Host Interface ID: 0 = Android KitKat [7184] 1 = Android Lollipop [n/a]

	5-7	2 = Android Lollipop Extended [7186]) Algorithm ID: 1 = SENtral

5.9.5 0X36: INT STATUS

This provides a way for a host, which does not use the host interrupt GPIO pin on the SENTRAL-A2, to determine when it is safe to read the Bytes Remaining registers and start a read of data from the output buffer area. NOTE: the time at which the host interrupt was asserted can be queried via the Host IRQ Timestamp parameter of the System parameter page or via the Host IRQ Timestamp registers.

The Host Interrupt bit reflects the state of the host interrupt GPIO pin. The Wakeup and Non-Wakeup Watermark bits are set if the watermark for their respective FIFOs was reached. The Wakeup and Non-Wakeup latency bits are set if a timeout on a sensor in their respective FIFOs expired. The Wakeup and Non-Wakeup Immediate bits are set if a sensor event has occurred which was configured with no latency.

The Host Interface Control interrupt disable bits (one each for the two FIFOs), when asserted, will cause ONLY the Host Interrupt bit and Host IRQ signal to deassert if asserted. The other bits will remain asserted, so that the host can still learn that there is a pending interrupt by reading this register. If any bits are set, there is a pending interrupt.

Register Name	Register Address	Bit Number	Register Value
Int Status	0x36	0	Host Interrupt
		1	Wakeup Watermark
		2	Wakeup Latency
		3	Wakeup Immediate
		4	Non-Wakeup Watermark
		5	Non-Wakeup Latency
		6	Non-Wakeup Immediate
		7	Reserved

Wakeup or Non-Wakeup Immediate indicates that one or more sensors in non-batch mode (latency = 0) generated a sample, leading to an interrupt.

5.9.6 0X37: CHIP STATUS

This register reflects fundamental behavior of the chip during boot up.

Register Name	Register Address	Bit Number	Register Value
Chip Status	0x37	0	EEPROM Detected
		1	EE Upload Done
		2	EE Upload Error
		3	Firmware Idle (halted)
		4	No EEPROM
		5-7	Defined by Software

See Appendix A for usage of these bits.

5.9.7 0X38-0X39: FIFO AVAILABLE TRANSFER LENGTH

This indicates how many bytes are available in the FIFO buffer. This can vary from the size of the smallest single FIFO event (a sensor sample or other event type) to the size of the entire FIFO. The maximum FIFO size can be queried using the FIFO Control Parameter in the System Parameter Page; see section 5.11.1.2.

The value of this register pair is only updated by SENtral at the following times:

- Immediately prior to asserting the host interrupt
- Upon demand, by means of the host writing a 1 to the Update Transfer Count bit of the Host Interface Control register.

This means that during normal operation, when the host receives an interrupt from SENtral, it should read the Bytes Remaining registers, and then read that many bytes from the FIFO. At that point, the host interrupt will de-assert, indicating that SENtral acknowledges that all the data the host knew about has been read.

If there is new data in the FIFO that arrived while the host was reading from the FIFO, after the host finishes reading the original quantity and SENtral has de-asserted the interrupt, the SENtral will update the Bytes Remaining registers and then reassert the host interrupt (depending on batching and water mark settings). This could occur immediately, or could occur later.

5.9.8 0X3A: PARAMETER ACKNOWLEDGE

After the host has written the desired page number to the Parameter Page Select register, and then has written the desired parameter number to the Parameter Request register, it should poll the Parameter Acknowledge register until it matches the Parameter Request register, or until it equals the special error

value 0x80. The error value means that the requested parameter page or parameter number is unsupported.

5.9.9 0X3B-0X4A: SAVED PARAMETER BYTES 0-15

The Saved Parameter Bytes area is large enough to report an entire sensor information structure in one transfer.

The host must request a parameter to be saved (read) by writing the page number to the Parameter Page Select register, and then writing the parameter number to the Parameter Request register. The host should then poll the Parameter Acknowledge register until it matches the desired parameter number (or until it equals the special error value 0x80). At that point, the Saved Parameter Bytes area will contain the value of the parameter.

The host can read more parameters within the same page by writing a new Parameter Request register value, polling for a match in the Parameter Acknowledge register, then reading the new parameter's value from the Saved Parameter Bytes area.

The host ends the parameter transfer procedure by writing the Parameter Request register with 0.

5.9.10 0X4B-0X4F: UNUSED ARC WRITEABLE

These are available for customer use in custom drivers or extensions. They are all read-only from the I2C host but writable from the ARC core.

5.9.11 0X50-0X53: ERROR, INTERRUPT STATE, DEBUG VALUE, DEBUG STATE

The Interrupt State, Debug Value, and Debug State registers are for EM Microelectronic troubleshooting. The Error register reports an internal error code. Some of this information is also reported in the FIFO using the Error Meta Event.

Error recovery strategies are described in section 5.10.3.

Table 11: Error Codes

Error Register Values	Description	Error Category

Error Register Values	Description	Error Category
0x00	No Error	
0x21	Sensor Init Failed: Unexpected Device ID	Hardware
0x22	Sensor Init Failed: No Response	Hardware
0x23	Sensor Init Failed: Unknown	Hardware
0x24	Sensor Error: No Valid Data	Programming
0x27	Stack Overflow	Fatal
0x30	Math Error	Temporary
0x40	Memory Error	Fatal
0x41	SWI3 Error	Fatal
0x42	SWI4 Error	Fatal
0x4F	Illegal Instruction Error	Fatal
0x60	Sensor Self-Test Failure	Hardware
0x61	Sensor Self-Test X Axis Failure	Hardware
0x62	Sensor Self-Test Y Axis Failure	Hardware
0x64	Sensor Self-Test Z Axis Failure	Hardware
0x70	Host Buffer Corrupt During Discard	Fatal
0x71	Host Buffer Corrupt During Retrieve	Fatal
0x72	Host Buffer Not Initialized	Programming
0x73	Insufficient Program RAM	Programming
0x74	Code RAM and Data RAM Not Adjacent	Programming
0x75	No Host Interrupt Set	Fatal
0x90	Master I2C Queue Overflow	Fatal
0xA0	Timer Scheduling Error	Fatal

5.9.12 0X54: PARAMETER PAGE SELECT

This register allows the host to select one of 15 different possible pages of parameters, up to 127 parameters in each page. Specific page numbers and parameters are pre-allocated, as indicated below.

Register Name	Register Address	Bits 7-4	Bits 3-0
Parameter Page Select	0x54	Parameter Size	Parameter Page

The least significant nibble contains the parameter page number, described below.

The most significant nibble contains the desired transfer size in bytes; if 0, the default of the entire space (16 bytes for saving / reading, 8 bytes for loading / writing) is selected. Size will be limited to the 16 bytes saving / 8 bytes loading should the host specify a larger value.

Parameter Page	Name	Description
0	None	The host must write this value after accessing the Algorithm Parameter Page in order for the internal processor to know it is safe to copy back the algorithm data structures.
1	System	This page contains parameters which affect the whole system, such as meta event enables, sensor status, FIFO watermark control, etc.
2	Algorithm	This page contains sensor fusion algorithm warm start parameters. The host can save these to non-volatile storage, and then restore them later after a reset to speed up calibration.
3	Sensor Information	This page contains parameters for every sensor (real or virtual), both for reading their status and for configuring their operation.
4	Reserved	This is reserved for future use.
5	Sensor Configuration	This page contains parameters for every sensor (real or virtual), wakeup and non-wakeup, for writing their Sensor Configuration structures or reading back the actual configuration.
6-8	Reserved	These are reserved for future use.
9-12	Customer Pages	These are reserved for customers for custom pages.
13	Algorithm Knobs	These parameters allow the host to configure advanced features of sensor fusion algorithm.
14-15	Reserved	These are reserved for future use.

5.9.13 0X55: HOST INTERFACE CONTROL

This register allows the host to rapidly request specific actions that affect the state of SENtral-A2 host interface.

NOTE: Abort Transfer and Update Transfer Count bits do not auto-clear. It is up to the host to set these two bits correctly every time it writes this register. It should not, however, due to race conditions, clear either of these bits immediately after setting either of them.

Register Name	Register Address	Bit Number	Register Value
Host Interface Control	0x55	0	Algorithm Standby Request
		1	Abort Transfer
		2	Update Transfer Count
		3	Wakeup FIFO Host Interrupt Disable
		4	NED coordinates
		5	AP Suspended
		6	Request Sensor Self-Test
		7	Non-Wakeup FIFO Host Interrupt Disable

- **Algorithm Standby** requests the algorithm to prepare itself to pause, then shuts down all sensors in order to save power. When this bit is de-asserted, any sensors previously enabled by the host will be restarted, and the operation of the algorithm will resume. This is a simpler way to temporarily conserve power without requiring the host to disable all active virtual sensors individually.
- **Abort Transfer** indicates the host does not intend to complete reading out the FIFO; all pending data in the 100 byte data transfer register area is discarded, as well as any partial sensor sample that remains, the host interrupt line is de-asserted, and the Bytes Remaining value is set to 0. If there is more data in the FIFO, SENtral will soon request another transfer. It is up to the host to recover properly from this; see section 5.13.9.
- **Update Transfer Count** can be used by the host to request a new value be written to the Bytes Remaining registers, such that data that has arrived since the last time Bytes Remaining was written, and data that has been removed, shall be accounted for. However, this does not extend the length of any pending or on-going transfer; it is merely an approximation of how much more there is in the FIFO.
- **Wakeup FIFO Host Interrupt Disable** is a master interrupt disable bit; setting this bit deasserts the host interrupt and prevents further interrupts, while clearing this bit (the default) allows it to be asserted whenever conditions warrant; this controls interrupt generation due to the wakeup FIFO.
- **NED Coordinates** selects the North East Down coordinate system instead of the default Android East North Up (ENU) system.
- **AP Suspended** affects when it is ok for SENtral to issue a host interrupt; when true, only Proximity and Significant Motion sensor events may wake the AP; when false, any sensor event may trigger a host interrupt if so configured.

- **Request Self-Test** is used by the host to inform us that a self-test should be performed when transitioning out of standby; any physical sensor drivers that implement self-test control of their sensor chips will request it, then report a Self-Test Results meta event with the results.
- **Non-Wakeup FIFO Host Interrupt Disable** disables host interrupts due to the non-wakeup FIFO.

5.9.14 0X56-0X5B: UNUSED HOST WRITEABLE

These registers are available for customer use in custom drivers or extensions. This range is writeable from the I^2C host and readable by the ARC processor.

5.9.15 0X5C-0X63: LOAD PARAMETER BYTES 0-7

The host uses this area as follows. Before writing the Parameter Request register with the MSB set, it should write the data it wishes to write to the parameter to this Load Parameter Bytes area. It should then write the Parameter Page Select register, followed by writing the Parameter Request register with the most significant bit set to indicate a load operation, to start the parameter transfer. It should then poll the Parameter Acknowledge register until its value matches the Parameter Request register, or equals 0x80 indicating an unsupported page and/or parameter number.

The host may write another parameter in the same page by repeating the procedure: write the new data to the Load Parameter Bytes area, write the new parameter number to the Parameter Request register with the MSB set, and then poll the Parameter Acknowledge register until it matches or equals 0x80 indicating an error.

The host ends the parameter transfer by writing the Parameter Request register with 0.

5.9.16 0X64: PARAMETER REQUEST

This register is where the host specifies which parameter for the selected page it would like to either load or save. Loading vs. saving is controlled by the MSB of this register. It should be clear to Save (read) and set to Load (write). The Parameter Acknowledge register will match this when the transfer is complete.

To end a series of parameter transfers, the Parameter Request should be written with a 0, causing Parameter Acknowledge to return a 0 on next read. Having the Parameter Acknowledge reset to 0 allows the host to determine on the next parameter I/O request whether the request was successful; if it was, the Parameter Acknowledge will match the Parameter Request; if it was not successful, the Parameter Acknowledge will equal the special error value 0x80.

5.9.17 0X65-0X6B: UNUSED HOST WRITEABLE

This range is available for customer use in custom drivers or extensions. This block is writeable by the I^2C host and readable by the ARC core.

5.9.18 0X6C-0X6F: HOST IRQ TIMESTAMP

This area is written with the timestamp of the host IRQ assertion. The write occurs within 5 µs of the rise of the Host IRQ line. The host must wait at least this amount of time before reading these registers in order to guarantee that all four bytes are from the same timestamp value (coherent). This same value can be queried by means of the System Page Timestamps Parameter.

5.9.19 0X70-0X71: ROM VERSION

This 16 bit register contains the build number corresponding to the firmware placed in ROM.

SENtral DI01: 0x07A8
SENtral DI02: 0x09E6
SENtral-A DI01: 0x1F9D
SENtral-A2: 0x27F2 (this IC)

5.9.20 0X72-0X73: RAM VERSION

This 16-bit register contains the build number corresponding to the RAM patch firmware, if any. If none is present, this will read back 0.

NOTE: if a watchdog timeout occurs during normal operation, the CPU will be reset, after which the RAM version will again read back as 0. This is an indication that the host will need to reload the RAM patch and reconfigure the sensors. See section 5.15.1 for details.

5.9.21 0X74-0X8F: RESERVED

5.9.22 0X90: PRODUCT ID

This identifies the overall product number for the chip.

SENtral: 0x80
SENtral-A: 0x84
SENtral-A2: 0x86

5.9.23 0X91: REVISION ID

This identifies the hardware revision for the chip.

EM7180: 0x02 (DI02)
EM7184: 0x01 (DI01)
SENTRAL-A2: 0x01 (DI01)

5.9.24 0X92-0X93: RESERVED

5.9.25 0X94-0X95: UPLOAD ADDRESS

This 16 bit register lets the host specify the starting address for a RAM patch. By default it is 0. After a RAM upload, it will not be 0, so a subsequent RAM upload procedure will need to start by writing this to 0.

5.9.26 0X96: UPLOAD DATA

Once the host has entered upload mode by writing a 1 to the Host Upload Enable bit of the Chip Control register, it may burst the RAM image to this register. Note that the RAM patch file format starts with a 16 byte header which must be skipped. Every 4 bytes of data in the file after the header must be byte swapped before upload to this register.

5.9.27 0X97-0X9A: DATA CRC

After the host has transferred all data from the RAM patch file to the Upload Data register, the Data CRC register will contain a 32 bit CRC of the data. The host should compare this to a calculated CRC to determine whether the upload was successful. If so, the host should disable upload mode and start firmware execution by writing a 0 to the Host Upload Enable bit and a 1 to the CPU Run Request bit of the Chip Control register.

5.9.28 0X9B: RESET REQUEST

The host writes a 1 to this register to trigger a hardware reset of the ARC processor core. This bit automatically clears to 0.

5.9.29 0X9C-0X9D: RESERVED

5.9.30 0X9E: PASS-THROUGH READY

This register returns a 1 when I^2C pass-through mode is active, and 0 when it is not. Pass-through logically connects (in hardware) the master and slave I2C busses, allowing the host to have direct access to the sensors. During this mode, the internal processor is prevented from performing its scheduled access to the sensors, so pass-through mode should only be done for a short time.

See section 5.15.2 for more details about pass-through mode.

5.9.31 0X9F: SCL LOW CYCLES

If pass-through mode with clock stretching is turned on, this register specifies the minimum period that SCL should be low.

5.9.32 0XA0: PASS-THROUGH CONFIG

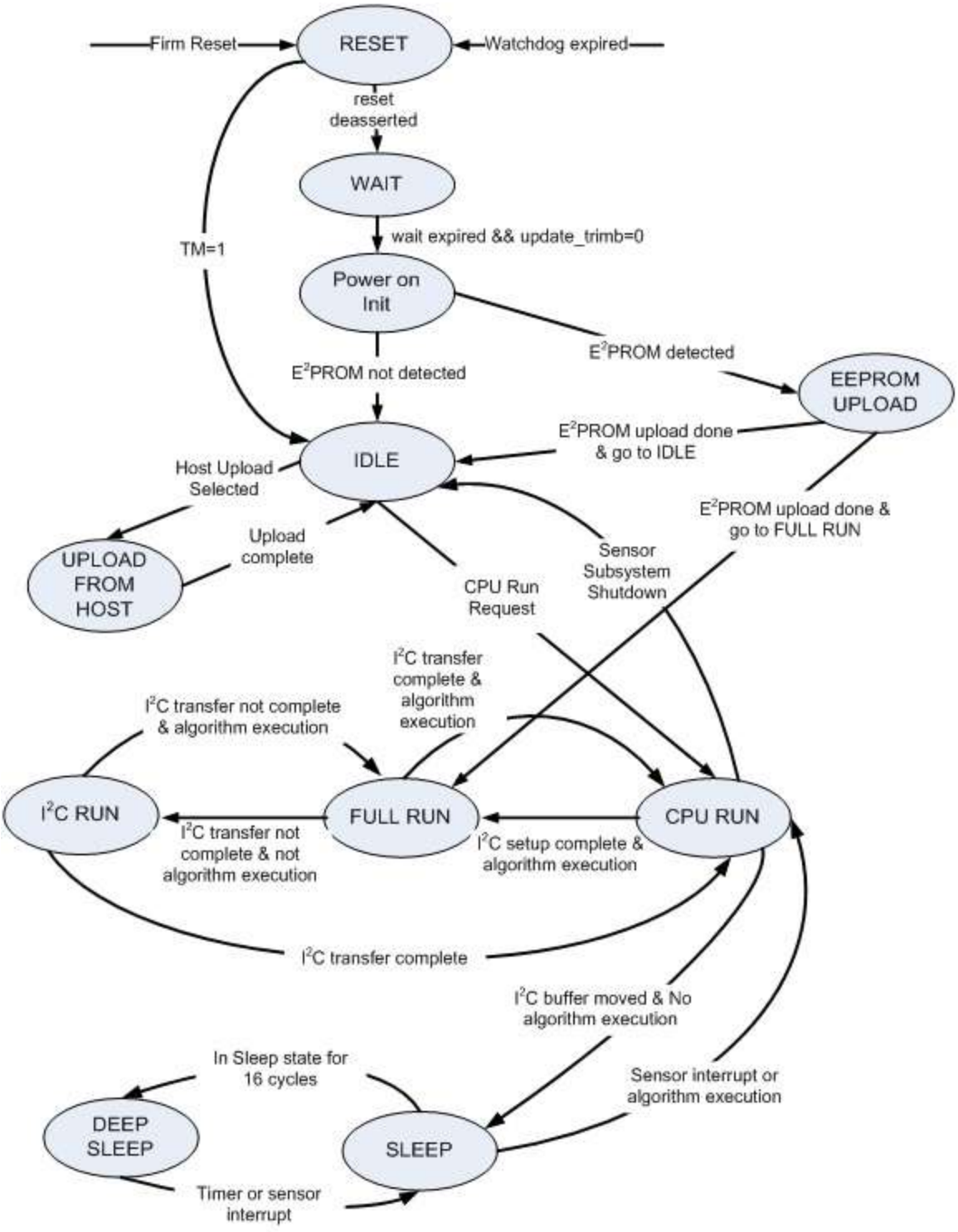
This register controls pass-through mode. Bit 0 is the pass-through enable bit, and bit 1 is the clock stretching enable bit. The latter is only effective during pass-through mode, and only applies to sensors which require clock stretching.

5.9.33 0XA1-0XB5: FACTORY USE ONLY

5.10 EXECUTION MODES AND ERROR HANDLING

5.10.1 EXECUTION FLOW CHART

The flow chart below shows all relevant operational states and state transition when starting up and operating SENtral-A2


Firm Reset
RESET
Watchdog expired
reset deasserted
WAIT
wait expired && update_trimb=0
TM=1
Power on Init
E²PROM detected
E²PROM not detected
EEPROM UPLOAD
E²PROM upload done & go to IDLE
IDLE
Host Upload Selected
E²PROM upload done & go to FULL RUN
UPLOAD FROM HOST
Upload complete
Sensor Subsystem Shutdown
CPU Run Request
I²C transfer complete & algorithm execution
I²C transfer not complete & algorithm execution
I²C RUN
FULL RUN
CPU RUN
I²C transfer not complete & not algorithm execution
I²C setup complete & algorithm execution
I²C transfer complete
I²C buffer moved & No algorithm execution
In Sleep state for 16 cycles
Sensor interrupt or algorithm execution
DEEP SLEEP
SLEEP
Timer or sensor interrupt

5.10.1.1 RESET

The system remains in this state while Power-on reset is asserted. Oscillator SYSOSC is enabled to synchronize the release of reset. When reset is released the system transitions to the WAIT state.

5.10.1.2 WAIT

The system remains in this state to allow for the oscillators to stabilize after which the system transitions to the POWER ON INIT state. SYSOSC is enabled in this state.

5.10.1.3 POWER ON INIT

After reset is released, the internal MCU starts executing a predefined start-up operation from ROM. The presence of an EEPROM is determined. Trim values are loaded. If an EEPROM is detected the device enters the EEPROM UPLOAD state, otherwise, the device enters the IDLE state. If the watchdog timer expires or register ResetReq is asserted the device transitions to POWER ON INIT.

5.10.1.4 EEPROM UPLOAD

The internal MCU uploads firmware from EEPROM to program RAM in this state. When upload is complete, indicated by register field Status.EEUploadDone = 1, the firmware sets register field Status.EEUploadError as appropriate. It is anticipated that a field in the EEPROM will have an expected CRC value. When upload is complete, the device transitions to state IDLE if register field Control.IDLEAfterUpload = 1, state CPU RUN otherwise.

5.10.1.5 IDLE

In this state the internal processor is halted. The Host can initiate the upload process to program RAM over the I2C Slave bus by writing register field ChipControl.HostUpload = 1 or command the processor to start execution from beginning of ROM or RAM by writing register field ChipControl.CPURunReq = 1.

5.10.1.6 UPLOAD FROM HOST

In this state the program RAM is uploaded with instructions to be executed by the internal processor. Instructions are sent by the Host over the I2C Slave bus. The internal processor remains halted during state Host UPLOAD. Once the upload is completed, indicated by the host writing register field ChipControl.HostUpload = 0, the chip returns to the IDLE state.

5.10.1.7 SLEEP

If a sensor interrupt occurs or it is time to execute the sensor fusion algorithm the system transitions to the CPU RUN state. If the system remains in this state for 16 clock cycles the system transitions to the DEEP SLEEP state.

5.10.1.8 DEEP SLEEP

SYSOSC is disabled while TIMOSC is running. SYSOSC is re-enabled when an external interrupt or timer interrupt occurs.

5.10.1.9 CPU RUN

Execution starts in the beginning of program RAM. Once initialization is complete the internal processor executes the sensor fusion algorithm, setting up the I^2C Master auxiliary registers prior to a transfer, or moving sensor data from the auxiliary registers to Data RAM after a transfer.

If it is time to execute the sensor fusion algorithm and the I^2C Master setup is complete the system transitions to state FULL RUN. If it is not time to execute the fusion algorithm and the i^2C Master data buffer has been moved to Data RAM the system transitions to state SLEEP.

If it is time to execute the SENtral algorithm and the system did not transition to this state due to a sensor interrupt or due to the I^2C Master setup being complete the system remains in this state.

The Host can request sensor subsystem shutdown by writing register field *ChipControl.CPURunReq = 0.*

5.10.1.10 FULL RUN

In this state the ARC executes the sensor fusion algorithm and an I^2C Master transaction is occurring. If the fusion algorithm is not complete and the I^2C Master transfer is complete the system transitions to state CPU RUN. If the fusion algorithm is complete and the I^2C Master transfer is not complete the system transitions to state I2C RUN. If the SpacePoint algorithm is not complete and the I^2C Master transfer is not complete the system remains in this state.

5.10.1.11 I^2C RUN

In this state an I^2C Master transaction is occurring. When the I^2C Master transfer is complete the system transitions to state CPU RUN. If the I^2C Master transfer is not complete and it is time to execute the fusion algorithm the system transitions to state FULL RUN. If the I^2C Master transfer is not complete and it is not time to execute the fusion algorithm the system remains in this state.

5.10.2 EXECUTION MODES

SENtral-A2 has two distinct modes of execution.

The ROM is essentially split into two parts – the small boot loader and the large set of libraries and drivers which can be used by a RAM-based program or "patch." It is this latter part of the ROM which provides most of the functionality required for sensor fusion, host interface interactions, data batching, and so on. However, without a RAM patch, none of these more advanced behaviors can occur. This is where boot loading comes in.

5.10.2.1 BOOT MODE

When SENtral first comes out of reset (due to power on reset, watchdog reset, or host-initiated reset), it executes a small portion of its ROM: the boot loader.

The boot loader loads and makes use of factory trim values, initializes the host interrupt GPIO line, then automatically scans the master I^2C bus for an attached EEPROM.

If an EEPROM is found, it will read the contents of the EEPROM into program RAM, check the CRC to confirm it is valid, and if so, automatically start execution of the program RAM (if so specified in the EEPROM contents), or interrupt the host and halt. If the contents are invalid, it will indicate an error in the Chip Status register, interrupt the host and halt.

If no EEPROM is found, it will interrupt the host and halted.

If halted, for any of these reasons, the host may directly load a RAM patch using the firmware image uploading procedure.

Whatever the source of the RAM patch, the host continues to the main execution mode by writing a 1 to the Chip Control register bit 0, CPU Run Request. Prior to running the RAM patch, the RAM Version registers will contain 0.

The complete initialization sequence is shown below:

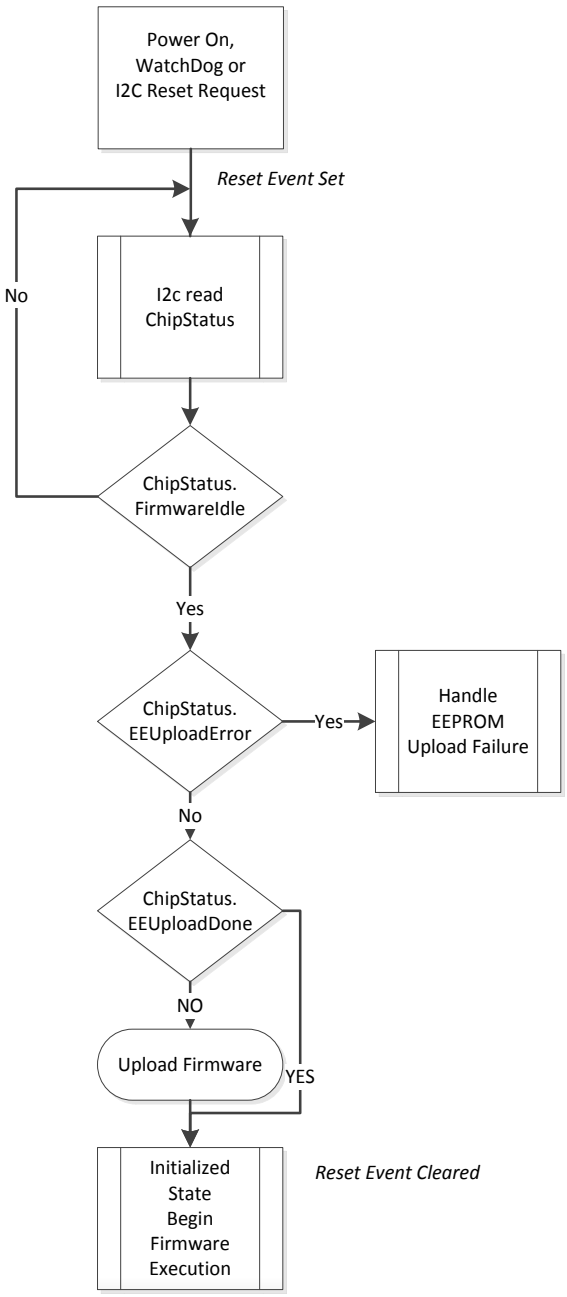


Figure 14: Boot Mode Flow Chart

5.10.2.2 MAIN EXECUTION MODE

Once in this mode, the full Android host interface and sensor suite is available. SENtral-A2 indicates it is ready by inserting an Initialized meta event in the FIFO, setting the Bytes Remaining registers to the size of this event plus timestamp, then asserting the host interrupt. The host should wait for this before attempting to query or configure sensors or other features.

If an incorrect RAM patch has been loaded (for example, is built for a different sensor suite), the FIFO will instead contain one or more Sensor Error or Error meta events.

In the nominal case, however, the host is now free to query which sensors are present by reading the Sensor Status bits, learn the details of each sensor by querying the Sensor Information parameters, load any Warm Start values using the Algorithm Warm Start parameters, and/or configure sensors to start generating output using the Sensor Configuration parameters.

The host may also wish to configure which meta events will appear in the FIFO, such as FIFO Overflow, Watermark, or many others. It can specify whether certain meta events can cause an immediate host interrupt, or are batched until later. Finally, the host may wish to configure the optional Watermark value using the FIFO Control parameter. This allows the host to be informed that the FIFO is full enough that the host may wish to read its contents before data is lost. This is especially useful when the Application Processor is asleep.

5.10.3 ERROR DETECTION AND RECOVERY

It is possible, in the Main Execution Mode, for SENtral to reach a fatal error condition. The host software or driver will need to be able to detect and recover from this condition.

It is recommended that a production driver self-monitor the recovery process. It should back off recovery attempts over longer and longer time intervals, and eventually stop recovery attempts after some reasonable amount of time, and log a fatal error. This is to prevent a continuous fatal hardware error or corrupted firmware file on the host system from triggering an infinite loop of recovery attempts.

These are the three levels which should be addressed in the host:

1) Watchdog / Power On Reset occurred during operation
   a) Host IRQ is received with 0 bytes to transfer
   b) Firmware Idle bit is set in the Chip Status register, or RAM version number is now 0
   c) Reaction should be:
      i) log registers 0x34 to 0x39, 0x50 to 0x53, and 0x6C to 0x73
      ii) reload RAM patch

iii) reconfigure system parameters (FIFO watermark, etc.)
iv) restart virtual sensors

2) Errors detected by EM7186 firmware
   a) Error Event in the FIFO (**do not mask** the error meta event)
   b) Or non-zero state of Error register
   c) Reaction should be:
      i) Fatal errors: issue reset over I2C (write 0x01 to register 0x9B), then the same as item 1
      ii) Hardware errors: same as previous point
      iii) Programming errors: should never occur in the field; recovery same as previous point
      iv) Temporary errors: ignore and continue

3) Unexpected / Unknown State / Live-lock
   a) Detection only by software watchdog in the driver (as described below)
   b) Reaction should be:
      i) Follow the same recovery as item 2 – Fatal Errors

5.10.4 HOST SOFTWARE WATCHDOG

A software watchdog in the host software or driver would by definition handle the cases (1) and (2) as well as (3).

The goal of such a system should be to detect and recover from unusual conditions, but it must be careful not to do this under normal operating conditions. So, one needs to specify "how long should the wait be before the decision to recover?" We suggest at least 1 second for param I/O, 1 or 2 seconds for streaming data with no batching, or a multiple of the batching period.

There are three sources of information that could be used to trigger a software watchdog failure:

Timeout waiting for param I/O handshaking to complete. The key value is the timeout duration itself; under no circumstances should param I/O take more than 1 second (very worst case) – it normally completes in tens of milliseconds. On timeout, declare "live-lock."

Timeout waiting for continuous output sensor data with no batching; AP Suspend mode off. Because sensor start up can take some time, it would be safest to ignore this until the first data arrives from a sensor; if no sensors are on, of course ignore this as well. Keep track of the fastest rate "FR" of all continuous output sensors. Usually this would be much higher than 1 Hz. If FR is > 1 Hz, and no host interrupt occurs within a timeout (1 or 2 seconds, or longer to be extra careful), declare "live-lock." If FR = 1 Hz, increase the timeout to 5 or 10 seconds.

Timeout on continuous output sensor data with batching/watermark; AP Suspend mode off. This is harder to do without false triggering. Keep track of the shortest latency setting (SLS) for all continuous

output sensors that are enabled and for which some data has arrived (since they were turned on). Use this SLS to set a timeout that is a multiple of SLS.

AP Suspend mode. In AP Suspend mode, detection and recovery is up to the customer. If the sensor hub is the only mechanism to wake up the AP, then perhaps a periodic timer set for a long time interval should occur which will check the state of the SENTRAL-A2 , and recover if conditions 1 or 2 have occurred. If other mechanisms such as hardware buttons can wake the AP, then perhaps no timer should be implemented in order to conserve battery power.

5.11 PARAMETER I/O

Except for a few features which the host can request using the Chip Control and Host Interface Control registers, or which it can query from other registers, the primary control channel for configuring and querying the state of the system and the sensors is done using Parameter I/O.

Parameter I/O in SENtral-A2 is implemented using a mail box protocol. This protocol includes a full handshake between the host and SENtral to synchronize access to the data transfer registers; these registers are used to carry control information to SENtral from the host or status information to the host from SENtral.

5.11.1 PARAMETER PAGE 1: SYSTEM

These parameters control general system-wide features.

Status banks 0 and 1 are for the nonwakeup sensors, and 2 and 3 are for the wakeup sensors.

Meta Event Control parameter 1 is for nonwakeup FIFO meta events, and Meta Event Control parameter 29 is for wakeup FIFO meta events.

Table 12: System Parameters, Parameter Page 1

Parameter Number	Parameter Name	Value Saved From	Value Loaded Into
1	Non-wakeup FIFO Meta Event Control		
2	FIFO Control		
3	Sensor Status Bank 0	Status Bits: Sensors 1-16	Not used
4	Sensor Status Bank 1	Status Bits: Sensors 17-32	Not used

Parameter Number	Parameter Name	Value Saved From	Value Loaded Into
5	Sensor Status Bank 2	Status Bits: Sensors 33-48	Not used
6	Sensor Status Bank 3	Status Bits: Sensors 49-64	Not used
7	Sensor Status Bank 4	Status Bits: Sensors 65-80	Not used
8	Sensor Status Bank 5	Status Bits: Sensors 81-96	Not used
9	Sensor Status Bank 6	Status Bits: Sensors 97-112	Not used
10	Sensor Status Bank 7	Status Bits: Sensors 113-128	Not used
11-28	Reserved		
29	Meta Event Control for Wakeup FIFO		
30	Host IRQ Timestamp		
31	Physical Sensor Status	Physical Sensor Status	Not Used
32	Physical Sensors Present	Bitmap of Available Physical Sensors	Not Used
33-96	Physical Sensor Info	Orientation Matrix, etc.	Not Used

5.11.1.1 META EVENT CONTROL (NON-WAKEUP AND WAKEUP)

The 8 bytes in this writeable parameter are divided into 32 two-bit sections. Each section controls whether the corresponding Meta Event will be enabled (so that it will appear in the output FIFO when it occurs), as well as whether that will lead to an immediate host interrupt. See section 5.12.9 for a list of Meta Events. Each specific Meta Event will have a unique default enable state, also specified in that section.

Parameter 1 is used to control Non-Wakeup FIFO Meta Events, while Parameter 29 controls Wakeup Meta Events.

> ***NOTE:*** *the sensor-related Meta Events like Sample Rate Changed, Power Mode Changed, and Dynamic Range Changed are always placed in the Non-Wakeup FIFO, even if the host had only turned on the Wakeup version of a given sensor. The Initialized Meta Event is always placed in the Wakeup FIFO. FIFO Watermark, FIFO Overflow, Flush Complete, and Transfer Cause will appear in the appropriate FIFO.*

The MSB in each Bit Range is the event enable, and each LSB is the event interrupt enable. The following table applies to both Parameter 1 and Parameter 29.

Load Parameter Byte	Enable Bit	Int Enable Bit	Meta Event
0	1	0	Meta Event 1
0	3	2	Meta Event 2
0	5	4	Meta Event 3
0	7	6	Meta Event 4
…			
7	1	0	Meta Event 29
7	3	2	Meta Event 30
7	5	4	Meta Event 31
7	7	6	Meta Event 32

5.11.1.2 FIFO CONTROL

This parameter provides a mechanism for the host to set a target number of bytes the main FIFO buffer should contain before it asserts the host interrupt signal.

Set this value to 0 to disable this feature, or a non-zero value to set the watermark. Any value larger than the size of the FIFO will be treated the same as a value exactly equal to the size of the FIFO.

Data loss will likely occur with watermark values that are too high. It is up to the customer to determine, in their application, based on the maximum I^2C host rate and host interrupt response time, what a safe maximum watermark level might be.

> ***NOTE:*** *a non-zero Watermark has no effect if all enabled sensors have 0 latencies (batch timeouts) and the AP is active (the Host Interface Control register's AP Suspend bit is 0). As soon as any one sensor generates a sample, and that sensor's latency is 0, a host interrupt will be generated. The Watermark is only useful when all enabled continuous output sensors are configured with non-zero latencies, or the AP is suspended, and no wakeup events occur (Proximity or Significant Motion).*

The size of the FIFO can be retrieved by the host by reading this same parameter; it is returned in bytes 2 and 3 for the Wakeup FIFO and bytes 6 and 7 for the Non-Wakeup FIFO. This size is determined at compile time of the RAM patch; additions of customer code or additional features or bug fixes will reduce the amount of RAM available for the FIFOs.

Parameter Byte	FIFO	Field Name	Description	Direction
0	Wakeup	Watermark LSB	Number of bytes in FIFO before	Read/Write

Parameter Byte	FIFO	Field Name	Description	Direction
1	Wakeup	Watermark MSB	interrupt is asserted	
2	Wakeup	FIFO Size LSB	Size of FIFO in bytes	Read only
3	Wakeup	FIFO Size MSB		
4	Non-Wakeup	Watermark LSB	Number of bytes in FIFO before interrupt is asserted (Least Significant Byte)	Read/Write
5	Non-Wakeup	Watermark MSB	Number of bytes in FIFO before interrupt is asserted (Most Significant Byte)	Read/Write
6	Non-Wakeup	FIFO Size LSB	FIFO size in bytes (Least Significant Byte)	Read Only
7	Non-Wakeup	FIFO Size MSB	FIFO size in bytes (Most Significant Byte)	Read Only

5.11.1.3 SENSOR STATUS BANKS

Each byte in a 16 byte Sensor Status Bank corresponds to a specific sensor type, starting with 1 (since sensor type 0 is reserved).

Sensor Status Bank	Saved Parameter Byte	Sensor Type
0	0	Sensor Type 1
0	…	
0	15	Sensor Type 16
1	0	Sensor Type 17
1	…	
1	15	Sensor Type 32
2	0	Sensor Type 33
2	…	
2	15	Sensor Type 48
3	0	Sensor Type 49
3	…	
3	15	Sensor Type 64

Sensor Status Bank	Saved Parameter Byte	Sensor Type
4	0	Sensor Type 65
4	…	
4	15	Sensor Type 80
5	0	Sensor Type 81
5	…	
5	15	Sensor Type 96
6	0	Sensor Type 97
6	…	
6	15	Sensor Type 112
7	0	Sensor Type 113
7	..	
7	15	Sensor Type 128

Each of these bytes contains the Sensor Status Bits for a given sensor. These reflect various pieces of information about a sensor that used to be scattered among a number of different registers.

Bit	Field Name	Description
0	Data Available	One or more samples in output buffer
1	I^2C NACK	Sensor did not acknowledge transfer
2	Device ID Error	WHO_AM_I register mismatch
3	Transient Error	e.g., magnetic transient
4	Data Lost	FIFO overflow
5-7	Sensor Power Mode	The Sensor Power Mode values in bits 5-7 are: 0: Sensor Not Present 1: Power Down 2: Suspend 3: Self-Test 4: Interrupt Motion 5: One Shot 6: Low Power Active 7: Active

5.11.1.4TIMESTAMPS

In order to provide a mechanism for the host to translate sensor data timestamps to host-relative timestamps, this parameter may be read to determine the time at which the last host-interrupt was asserted, as well as the current system time. NOTE: the Host IRQ Timestamp can be read more efficiently from the Host IRQ Timestamp registers 0x6C-0x6F.

Parameter Byte	Field Name	Description	Direction
0	Host IRQ Timestamp LSB	Time (in units of 1/32000 seconds) that the last host interrupt was triggered	Read only
1	Host IRQ Timestamp B2		
2	Host IRQ Timestamp B3		
3	Host IRQ Timestamp MSB		
4	Current Timestamp LSB	Time (in units of 1/32000 seconds) for current system time	
5	Current Timestamp B2		
6	Current Timestamp B3		
7	Current Timestamp MSB		

5.11.1.5PHYSICAL SENSOR STATUS

This parameter is provided for debugging. It lets the host find out the actual underlying physical sensor settings such as sample rate, dynamic range, interrupt enable, and power mode.

Parameter Byte	Field Name	Type	Description
0	Accel Sample Rate	Unsigned 16 bit	Actual sample rate in Hz
1			
2	Accel Dynamic Range	Unsigned 16 bit	Actual dynamic range in gs
3			
4	Accel Flags	Unsigned 8 bit	bit 0: interrupt enable bits 5-7: Sensor Power Mode (same as Sensor Status Bits 5-7)
5	Gyro Sample Rate	Unsigned 16 bit	Actual sample rate in Hz
6			
7	Gyro Dynamic Range	Unsigned 16 bit	Actual dynamic range in gs

Parameter Byte	Field Name	Type	Description
8			
9	Gyro Flags	Unsigned 8 bit	bit 0: interrupt enable bits 5-7: Sensor Power Mode (same as Sensor Status Bits 5-7)
10	Mag Sample Rate	Unsigned 16 bit	Actual sample rate in Hz
11			
12	Mag Dynamic Range	Unsigned 16 bit	Actual dynamic range in gs
13			
14	Mag Flags	Unsigned 8 bit	bit 0: interrupt enable bits 5-7: Sensor Power Mode (same as Sensor Status Bits 5-7)

5.11.1.6 PHYSICAL SENSORS PRESENT

This parameter contains a 64 bit bitmap, where a set bit indicates the corresponding physical sensor is present in the system.

For example, if a physical accelerometer, magnetometer, and humidity sensor were the only physical sensors present, the bit map would have bits set for sensor ID 1 (accelerometer), 2 (magnetometer) and 12 (humidity). In this example, the bit map would be:

Byte 0: 0000 0110 (binary; left most bit is bit 7, right most is bit 0)
Byte 1: 0001 0000 (left most is bit 15; right most is bit 8)
Byte 2: 0000 0000
Byte 3: 0000 0000
Byte 4: 0000 0000
Byte 5: 0000 0000
Byte 6: 0000 0000
Byte 7: 0000 0000

5.11.1.7 PHYSICAL SENSOR INFORMATION

This structure is returned for any parameters 33-96 when the corresponding physical sensor is present. If not present, this structure returns all 0s.

This is an enhanced version of the earlier Physical Sensor Status structure. This new structure also provides access to the orientation matrix, for those sensors that include them, such as 3 axis accelerometers, magnetometers, and gyroscopes.

Parameter Byte	Field Name	Type	Description
0	Sensor Type	Unsigned 8 bit	Same as parameter number - 32
1	Driver ID	Unsigned 8 bit	Unique per driver / vendor / part number
2	Driver Version	Unsigned 8 bit	Denotes notable change in behavior
3	Current	Unsigned 8 bit	0.1 mA
4-5	Current Range	Unsigned 16 bit	Current dynamic range of sensor in SI units
6	Flags	Unsigned 8 bit	Bit 0: IRQ enabled Bits 1-4: reserved Bits 5-7: power mode (same as sensor status bits 5-7)
7	Reserved		
8-9	Current Rate	Unsigned 16 bit	Current Sample Rate in Hz
10	Number of Axes	Unsigned 8 bit	Number of Axes (e.g., X/Y/Z = 3)
11-15	Orientation Matrix	4 bits per element	See below

The calibration matrix is output in the same order as the elements are listed in the board .cfg file used to generate a firmware image, described in the SENTRAL-A2 Application Note 005 Programming Guide, where each matrix element is stored in successive nibbles.

For example, if the board .cfg file contains:

```
#DriverID,Addr,GPIO,C0,C1,C2,C3,C4,C5,C6,C7,C8,Off0,Off1,Off2,Range
a9, 24, 3, 1, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0
```

Or, if the stuffelf utility were used to generate the .fw file using the command line:

```
stuffelf outerloop.elf –a -d24 -p3 -c1,0,0,0,-1,0,0,0,-1
```

Then bytes 11-15 of the Physical Sensor Information structure would be:

Byte 11: 01 (hexadecimal)

Byte 12: 00

Byte 13: 0F

Byte 14: 00

Byte 15: 0F

The matrix is used to correct the orientation of physical sensor axes to match the required ENU (east north up) orientation required by Android. The calculation is performed as:

$$|X\ Y\ Z| = |X_s\ Y_s Z_s| \cdot \begin{vmatrix} C_0 C_1 C_2 \\ C_3 C_4 C_5 \\ C_6 C_7 C_8 \end{vmatrix}$$

5.11.2 PARAMETER PAGE 2: ALGORITHM WARM START

The algorithm parameters split into two pages. The first page below is called Algorithm Warm Page and it is for reading and writing Warm Start parameters (Parameter Number 1-29) and Calibration Matrices parameters (Parameter Number 30-50). The rest of parameters are reserved for future usage (Parameter Number 51-127).

The Algorithm Warm Start page uses the Load and Save Parameter Byte that are 8 bytes wide. Thus the array coefficients are packed more tightly (usually two float values at one parameter number). All parameters are float except where noted other type (in column Note).

Please note that values presented in the table use MATLAB matrix notation starting at 1 index instead of 0 index (C notation).

Table 13: Algorithm Warm Start Parameters

Parameter		Value			Note
#	Name	Bytes	Saved From	Loaded Into	
Warm Start Params I: 1-29					
1 … 29	Warm start	0-7	The warm start parameters 1-29 should be saved and restored by the host without regard to the internal meanings of each parameter. Simply treat them as a block of binary data. See SENtral Warm Start Procedure document for more details.		
Warm Start Params II, NED Calibration Matrices: 30-50					
30 … 50	Calibration	0-7	Details of these matrices are hardware specific.		
Warm Start Params III, Reserved parameters 51-127					
51-127	Reserved	0-7	Not used	Not used	

5.11.3 PARAMETER PAGE 13: ALGORITHM KNOBS

The second algorithm page is called Algorithm Knobs Page and it is for reading and writing Knobs parameters defined in the following table – Knob Parameters (Parameter Number 1-87). The rest of parameters are reserved for future usage (Parameter Number 88-127).

Table 14: Algorithm Knob Parameters

Parameter			Value		Note
#	Name	Bytes	Saved From	Loaded Into	
Knob Parameters I: 1-87					
1 … *87*		varies	Details of the algorithm knobs and how to use them are available in App Note: Sentral Parameters.		
Knob Parameters II, Reserved parameters 88-127					
88 – 127	Reserved	0-7	Not used	Not used	

5.11.4 PARAMETER PAGE 3: SENSOR INFORMATION

The table below of sensor parameters is split into two logical areas. The first area, Config Numbers 1-63, is for reading the Sensor Information structure of a specific non-wakeup sensor. The second area, Config Numbers 65-127, is for reading the Sensor Information structure of a specific wakeup sensor.

Table 15: Sensor Information Parameters

Parameter Number	Sensor Name	Value Saved From	Value Loaded Into
Non-Wakeup Sensor Information			
1	Accelerometer	Sensor Information	Not used
2	Magnetometer	Sensor Information	Not used
3	Orientation	Sensor Information	Not used
4	Gyroscope	Sensor Information	Not used
5	Light	Sensor Information	Not used

Parameter Number	Sensor Name	Value Saved From	Value Loaded Into
6	Barometer	Sensor Information	Not used
7	Temperature	Sensor Information	Not used
8	Proximity	Sensor Information	Not used
9	Gravity	Sensor Information	Not used
10	Linear Acceleration	Sensor Information	Not used
11	Rotation Vector	Sensor Information	Not used
12	Humidity	Sensor Information	Not used
13	Ambient Temperature	Sensor Information	Not used
14	Uncalibrated Magnetometer	Sensor Information	Not used
15	Game Rotation Vector	Sensor Information	Not used
16	Uncalibrated Gyroscope	Sensor Information	Not used
17	Significant Motion	Sensor Information	Not used
18	Step Detector	Sensor Information	Not used
19	Step Counter	Sensor Information	Not used
20	Geomagnetic Rotation Vector	Sensor Information	Not used
21	Heart Rate	Sensor Information	Not used
22	Tilt Detector	Sensor Information	Not used
23	Wake Gesture	Sensor Information	Not used
24	Glance Gesture	Sensor Information	Not used
25	Pick Up Gesture	Sensor Information	Not used
26-30	Reserved	Sensor Information	Not used
31	Activity	Sensor Information	Not used
32-63	Reserved	Sensor Information	Not used
Wakeup Sensor Information			
64	Reserved		
65	Accelerometer	Sensor Information	Not used
66	Magnetometer	Sensor Information	Not used
67	Orientation	Sensor Information	Not used
68	Gyroscope	Sensor Information	Not used

Parameter Number	Sensor Name	Value Saved From	Value Loaded Into
69	Light	Sensor Information	Not used
70	Barometer	Sensor Information	Not used
71	Temperature	Sensor Information	Not used
72	Proximity	Sensor Information	Not used
73	Gravity	Sensor Information	Not used
74	Linear Acceleration	Sensor Information	Not used
75	Rotation Vector	Sensor Information	Not used
76	Humidity	Sensor Information	Not used
77	Ambient Temperature	Sensor Information	Not used
78	Uncalibrated Magnetometer	Sensor Information	Not used
79	Game Rotation Vector	Sensor Information	Not used
80	Uncalibrated Gyroscope	Sensor Information	Not used
81	Significant Motion	Sensor Information	Not used
82	Step Detector	Sensor Information	Not used
83	Step Counter	Sensor Information	Not used
84	Geomagnetic Rotation Vector	Sensor Information	Not used
85	Heart Rate	Sensor Information	Not used
86	Tilt Detector	Sensor Information	Not used
87	Wake Gesture	Sensor Information	Not used
88	Glance Gesture	Sensor Information	Not used
89	Pick Up Gesture	Sensor Information	Not used
90-94	Reserved	Sensor Information	Not used
95	Activity	Sensor Information	Not used
96-127	Reserved	Sensor Information	Not used

5.11.4.1 SENSOR INFORMATION STRUCTURE

This structure reports everything that Android needs to know about a sensor type. If the requested sensor is not supported by the current firmware image, then all fields will be reported as zero.

For physical sensors, the Max Range field will be set to the current Dynamic Range setting. This is a constant value that does not change based on the current Dynamic Range setting. It is stored in Android-appropriate units (m/s^2, radians/s, µT). As this is a 16 bit integer field, the value should be rounded up to the next nearest integer.

The Resolution field is provided so that the host can determine the "smallest difference between two values reported by this sensor." It contains the number of bits of resolution. With that, the host can determine the floating point resolution value in SI units by dividing the Max Range or current Dynamic Range (in SI units) by $2^{Resolution}$.

Parameter Byte	Field Name	Type	Description
0	Sensor Type	Unsigned 8 bit	Defensive programming measure – repeat the requested sensor type
1	Driver ID	Unsigned 8 bit	Unique per driver / vendor / part number
2	Driver Version	Unsigned 8 bit	Denotes notable change in behavior
3	Power	Unsigned 8 bit	0.1 mA
4	Max Range	Unsigned 16 bit	maximum range of sensor data in SI units
5			
6	Resolution	Unsigned 16 bit	number of bits of resolution of underlying sensor
7			
8	Max Rate	Unsigned 16 bit	Hz
9			
10	FIFO Reserved	Unsigned 16 bit	FIFO size in bytes reserved for this sensor divided by data packet size in bytes; **if a single shared FIFO, this can be 0**
11			
12	FIFO Max	Unsigned 16 bit	Entire FIFO size in bytes divided by data packet size in bytes
13			
14	Event Size	Unsigned 8 bit	Number of bytes for sensor data packet (including Sensor Type)
15	Min Rate	Unsigned 8 bit	Hz

5.11.5 PARAMETER PAGE 4: RESERVED

5.11.6 PARAMETER PAGE 5: SENSOR CONFIGURATION

The table below of sensor parameters is split into two logical areas. The first area, Config numbers 1-63, is for writing the Sensor Configuration structure of a specific non-wakeup sensor, or for reading the actual sample rates, latencies, dynamic ranges, and sensitivities. The second area, Config numbers 65-127, is for writing the Sensor Configuration structure of a specific wakeup sensor or for reading the actual settings.

Table 16: Sensor Configuration Parameters

Parameter Number	Sensor Name	Value Saved From	Value Loaded Into
Non-Wakeup Sensor Configuration			
1	Accelerometer	Actual Configuration	Configuration
2	Magnetometer	Actual Configuration	Configuration
3	Orientation	Actual Configuration	Configuration
4	Gyroscope	Actual Configuration	Configuration
5	Light	Actual Configuration	Configuration
6	Barometer	Actual Configuration	Configuration
7	Temperature	Actual Configuration	Configuration
8	Proximity	Actual Configuration	Configuration
9	Gravity	Actual Configuration	Configuration
10	Linear Acceleration	Actual Configuration	Configuration
11	Rotation Vector	Actual Configuration	Configuration
12	Humidity	Actual Configuration	Configuration
13	Ambient Temperature	Actual Configuration	Configuration
14	Uncalibrated Magnetometer	Actual Configuration	Configuration
15	Game Rotation Vector	Actual Configuration	Configuration
16	Uncalibrated Gyroscope	Actual Configuration	Configuration
17	Significant Motion	Actual Configuration	Configuration
18	Step Detector	Actual Configuration	Configuration
19	Step Counter	Actual Configuration	Configuration
20	Geomagnetic Rotation Vector	Actual Configuration	Configuration
21	Heart Rate	Actual Configuration	Configuration

Parameter Number	Sensor Name	Value Saved From	Value Loaded Into
22	Tilt Detector	Actual Configuration	Configuration
23	Wake Gesture	Actual Configuration	Configuration
24	Glance Gesture	Actual Configuration	Configuration
25	Pick Up Gesture	Actual Configuration	Configuration
26-30	Reserved	Actual Configuration	Configuration
31	Activity	Actual Configuration	Configuration
32-63	Reserved	Actual Configuration	Configuration
Wakeup Sensor Configuration			
64	Reserved		
65	Accelerometer	Actual Configuration	Configuration
66	Magnetometer	Actual Configuration	Configuration
67	Orientation	Actual Configuration	Configuration
68	Gyroscope	Actual Configuration	Configuration
69	Light	Actual Configuration	Configuration
70	Barometer	Actual Configuration	Configuration
71	Temperature	Actual Configuration	Configuration
72	Proximity	Actual Configuration	Configuration
73	Gravity	Actual Configuration	Configuration
74	Linear Acceleration	Actual Configuration	Configuration
75	Rotation Vector	Actual Configuration	Configuration
76	Humidity	Actual Configuration	Configuration
77	Ambient Temperature	Actual Configuration	Configuration
78	Uncalibrated Magnetometer	Actual Configuration	Configuration
79	Game Rotation Vector	Actual Configuration	Configuration
80	Uncalibrated Gyroscope	Actual Configuration	Configuration
81	Significant Motion	Actual Configuration	Configuration
82	Step Detector	Actual Configuration	Configuration
83	Step Counter	Actual Configuration	Configuration
84	Geomagnetic Rotation Vector	Actual Configuration	Configuration

Parameter Number	Sensor Name	Value Saved From	Value Loaded Into
85	Heart Rate	Actual Configuration	Configuration
86	Tilt Detector	Actual Configuration	Configuration
87	Wake Gesture	Actual Configuration	Configuration
88	Glance Gesture	Actual Configuration	Configuration
89	Pick Up Gesture	Actual Configuration	Configuration
90-94	Reserved	Actual Configuration	Configuration
95	Activity	Actual Configuration	Configuration
96-127	Reserved	Actual Configuration	Configuration

5.11.6.1 SENSOR CONFIGURATION STRUCTURE

The meaning of each field below is slightly different depending on whether this is being loaded (written) or saved (read); see the description for details.

Changes to the Sample Rate field take effect quickly, but not immediately. If the host wishes to know when the rate change is complete, it can enable and wait for the Sample Rate Changed meta event. The actual rate selected is only guaranteed to be equal to or greater than the requested rate and twice that rate:

Requested Rate <= Actual Rate < Requested Rate X 2

Changes to the Max Report Latency take effect immediately. If a timer for the sensor using a different Max Report Latency is running, it will be modified. It is possible due to timing for a change to be slightly too late to effect the current timer, but will affect subsequent samples. If Max Report Latency is set to 0 when it was previously not 0, then this will be treated the same as a flush request.

The Dynamic Range field for the virtual Accelerometer, Gyroscope, and Magnetometer sensors controls the actual dynamic range settings in the corresponding physical sensors. A value of 0 requests the default. The algorithm will be informed of the request to change the dynamic range, and, the corresponding scale factor for the sensor data outputs will change accordingly. The host may then read back the Sensor Configuration structure to determine the actual current dynamic range.

NOTE: *the host should enable and watch for the Dynamic Range Changed meta event so it can apply the correct scale factor before and after the dynamic range change, if the change is made while the sensor is already enabled.*

Reading back the parameter is especially important if the host sets a dynamic range for other virtual sensors that share the same underlying physical sensor. SENtral-A2 will select the largest requested dynamic range of all virtual sensors that share that physical sensor.

For instance, the virtual Accelerometer, Gravity, and Linear Acceleration sensors all share the physical accelerometer. The virtual Gyroscope and Uncalibrated Gyroscope both share the physical gyroscope. The virtual Magnetometer and Uncalibrated Magnetometer share the physical magnetometer. If the host does not specify a dynamic range for a specific virtual sensor (by setting it to 0), then only the virtual sensors with non-zero dynamic range requests from the host will be considered in selecting the actual dynamic range for the physical sensor.

For example, setting the dynamic range for the Accelerometer sensor to 156.93 m/s^2 while setting the dynamic range of the Linear Acceleration sensor to 78.46 m/s^2 and the Gravity sensor to 0 results in an actual dynamic range for all three sensors as well as the physical sensor of 156.93 m/s^2, which is the same as 16 Earth g-s.

The dynamic range will determine the scale factor for the sensor data, based on the number of bits and signed-ness of the data. See section 5.12.5.

The units of measurement for dynamic range here are not the same as the Android units for those sensors. We have chosen to use units that are commonly used by sensor manufacturers in their data sheets when discussing range settings.

- Accelerometer: Earth g-s
- Gyroscope: degrees/second
- Magnetometer: µT

Parameter Byte	Field Name	Type	Description
0	Sample Rate	Unsigned 16 bit	Rate in Hz; 1 ÷ Android sample period; reads back actual sensor rate
1			
2	Max Report Latency	Unsigned 16 bit	0 for non-batch mode; if nonzero, Sample Rate must also be nonzero; milliseconds; reads back actual latency
3			

Parameter Byte	Field Name	Type	Description
4	Change Sensitivity	Unsigned 16 bit	Scaled same as sensor's data value; for future Win8/10 support; not implemented
5			
6	Dynamic Range	Unsigned 16 bit	range setting for physical setting in appropriate units
7			

5.12 SENSOR DATA OUTPUT FORMAT

The format for data being stored in the FIFO buffer is the same as the format mentioned in section 9 with the addition of a sensor ID. All multi-byte fields are little-endian.

Sensor IDs for non-wakeup sensors will match the Android numbering; these are currently numbered 1 through 31, but can expand up to 63 to include new Android sensors as well as customer developed sensors. Any new sensor IDs that represent unique events that are not to be mapped to host-side sensors will be numbered starting at 254, decreasing towards the Android numbers. Sensor IDs for wakeup sensors are specified using the Android number plus 64.

If the host and SENtral become out of sync, the host can regain synchronization as described in section 5.13.7. Any data that arrives after that will start with a whole sensor data packet, with the Sensor Type as the first byte.

Table 17: Sensor IDs and Data Formats

Sensor ID Non Wakeup	Sensor ID Wakeup	Size in FIFO	Contents	Scale Factor	Sensor Value	Format
		1	**Padding**			Unsigned 8 bit value 0 means NOP
11 15 20	75 79 84	11	**Rotation Vector** **Game Rotation Vector** **Geomagnetic Rotation Vector**	2^{14}	X, Y, Z, W Quaternion Estimated Accuracy (radians)	16 bit signed fixed point 16 bit signed fixed point integer
1 2 3 4 9 10	65 66 67 68 73 74	8	**Accelerometer** **Magnetometer** **Orientation*** **Gyroscope** **Gravity** **Linear Acceleration**	dynamic dynamic $360° / 2^{15}$ dynamic dynamic dynamic	X, Y, Z Vector Status	16 bit signed integer, scaled to current dynamic range 8 bit unsigned integer indicating accuracy of measurement (low, medium, high, unreliable)

Sensor ID Non Wakeup	Sensor ID Wakeup	Size in FIFO	Contents	Scale Factor	Sensor Value	Format
5 8 12	69 72 76	3	**Light** **Proximity** **Humidity**	10000Lux / 2^{16} 100cm / 2^{16} 1%RH	Absolute Scalar	16 bit unsigned integer, scaled to maximize dynamic range
19	83	3	**Step Counter**	None	Absolute Scalar	16 bit unsigned integer
7 13	71 77	3	**Temperature** **Ambient Temperature**	500LSB/°C centered at 24°C	Scalar	16 bit signed integer, scaled to maximize dynamic range
6	70	4	**Barometer**	1/128 Pa	Absolute Scalar	24 bit unsigned integer, scaled as required
17 18 22 23 24 25	81 82 86 87 88 89	1	**Significant Motion** **Step Detector** **Tilt Detector** **Wake Gesture** **Glance Gesture** **Pick Up Gesture**	None	Event	None
14 16	78 80	14	**Uncalibrated Magnetometer** **Uncalibrated Gyroscope**	dynamic dynamic	Uncalib X, Y, Z; Bias X, Y, Z Status	16 bit signed integer, scaled to current dynamic range 8 bit unsigned integer indicating accuracy of measurement (low, medium, high, unreliable)
21	85	2	**Heart Rate**	None	Scalar	8 bit unsigned integer
31	95	3	**Activity**	None	Scalar	Bits 0-7 specify the activity change off, bits 8-15 specify the activity change on; see below for bit definitions
245	n/a	14	**Debug**	None	Structure	Byte1: Bit 7: reserved Bit 6: binary format (string if 0, binary if 1) Bits 5-0: valid bytes Bytes 2-13: debug data
249 250 251	n/a	17	**Raw Gyro** **Raw Mag** **Raw Accel**	None	X, Y, Z Vector; Temperature	32 bit float; 32 bit temperature
252	246	3	**Timestamp LSW**	1/32000 seconds	Time	16 bit unsigned integer; counts at a 32KHz rate; applies to all following sensor samples; wraps every 2 seconds
253	247	3	**Timestamp MSW (overflow)**	65536/32000 seconds	Time	16 bit unsigned integer; counts at Timestamp overflow rate; wraps every 37.28 hours

Sensor ID Non Wakeup	Sensor ID Wakeup	Size in FIFO	Contents	Scale Factor	Sensor Value	Format
254	**248**	**4**	**Meta Events**	None	Event	8 bit unsigned integer event number 8 bit unsigned integer sensor type 8 bit unsigned integer event-specific value

*NOTE: X = azimuth = +/- 180°, Y = pitch = +/- 180°, Z = roll = +/-90°. The orientation sensor output is identical for both the default ENU (East North Up, the Android standard) and the optional NED (North East Down) coordinate systems.

"dynamic" means the scale factor is calculated from the dynamic range

5.12.1 QUATERNION DATA

For the three rotation vectors (Rotation Vector, Game Rotation Vector, Geomagnetic Rotation Vector), the following format is used. The host can convert the X, Y, Z, W, and Estimated Accuracy fields to floating point numbers like this:

```
unsigned char event[11];
… read in the event to the array above…
float x = ((float)(event[1] + ((unsigned int)event[2]) << 8)) / 16384;
… convert other fields…
```

The Estimated Accuracy in Radians is reported as 0 for the Game Rotation Vector.

Byte Number	Contents	Format
0	Sensor ID (Rotation Vector, etc.)	
1	X LSB	Signed 16 bit fixed point integer
2	X MSB	
3	Y LSB	Signed 16 bit fixed point integer
4	Y MSB	
5	Z LSB	Signed 16 bit fixed point integer
6	Z MSB	
7	W LSB	Signed 16 bit fixed point integer
8	W MSB	
9	ESTIMATED ACCURACY, RADIANS, LSB	Signed 16 bit fixed point

Byte Number	Contents	Format
10	ESTIMATED ACCURACY, MSB	integer

5.12.2 THREE AXIS DATA

For the many 3 axis sensors (Accelerometer, Magnetometer, Orientation, Gyroscope, Gravity, Linear Acceleration), the following layout is used.

Byte Number	Contents	Format
0	Sensor ID (Accelerometer, etc.)	
1	X LSB	Signed 16 bit integer
2	X MSB	
3	Y LSB	Signed 16 bit integer
4	Y MSB	
5	Z LSB	Signed 16 bit integer
6	Z MSB	
7	STATUS	Accuracy of measurement 0 Unreliable 1 Accuracy Low 2 Accuracy Medium 3 Accuracy High

5.12.3 UNCALIBRATED 3 AXIS DATA

For the two uncalibrated 3 axis sensors (Uncalibrated Magnetometer, Uncalibrated Gyroscope), the following layout is used.

Byte Number	Contents	Format
0	Sensor ID	
1	X LSB	Signed 16 bit integer
2	X MSB	
3	Y LSB	Signed 16 bit integer
4	Y MSB	

5	Z LSB	Signed 16 bit integer
6	Z MSB	
7	X BIAS LSB	Signed 16 bit integer
8	X BIAS MSB	
9	Y BIAS LSB	Signed 16 bit integer
10	Y BIAS MSB	
11	Z BIAS LSB	Signed 16 bit integer
12	Z BIAS MSB	
13	STATUS	Accuracy of measurement 0 Unreliable 1 Accuracy Low 2 Accuracy Medium 3 Accuracy High

5.12.4 DEBUG DATA

Customers using the SDK can use printf() to send ASCII strings to the host in the non-wakeup FIFO, or fwrite() to send custom binary data. The data will be sent with Sensor ID = Debug.

The lower 6 bits include the number of valid bytes in the packet with the range from 0 up to 12. For example, if a 20 character long string is printed out: “ABCDEFGHIJKLMNOPQRST”, the FIFO will contain:

Debug (=245), 0x0C, “ABCDEFGHIJKL”
Debug, 0x08, “MNOPQRST”

Byte Number	Bit	Contents
0	0-7	Sensor ID (Debug)
1	0-5	Number of valid bytes that follow (0-12)
	6	Binary Data (=0 for ASCII data)
	7	Reserved
2-13	0-7	Customer debug data

5.12.5 SCALE FACTORS

The table of sensor types and their data formats above indicates a fixed scale factor for each sensor's data for many sensors. The selection of this scale factor is meant to ensure the full dynamic range of a sensor can fit within the number of bits provided. The exceptions are the accelerometer-, gyroscope-, and magnetometer-derived sensors. Those sensors' scale factors are set dynamically based on the largest dynamic range setting for any related virtual sensors (see section 5.11.6).

5.12.6 DEFAULT SCALE FACTORS

The default scale factor specified for the Accelerometer, Gravity, and Linear Acceleration allows for a 4g range (29.2 m/s2).

The default scale factor for the Gyroscope allows for a 2000° / s range (34.9 radians / s).

The default scale factor for the Magnetometer is based on a dynamic range of 1000µT, despite the fact that many sensors can read larger values than this. The AK8963, for example, can measure fields up to 4912µT. In return, there will be 2 more bits of resolution for normal readings, keeping in mind that the Earth's magnetic field ranges from 25µT to 65µT. Android does not require any particular range for the magnetometer beyond this.

These sensors all use a scale factor that adjusts automatically to fit the actual dynamic range of each sensor, if changed from the defaults. The host needs to query the actual dynamic range and then perform a calculation to determine the scale factor as dynamic range / $2^{\text{number of bits}}$, where the number of bits is 15 for signed 16 bit integers and so on. Note that the accelerometer and gyroscope dynamic range settings, as discussed in section 5.11.6.1, need to be converted to m/s^2 and radians/s, before being used to scale the output data as described below.

For example, the Accelerometer produces 16 bit signed data for the X, Y, and Z axes. The scale factor for a dynamic range of 156.93 m/s^2 would be:

Accelerometer Scale Factor = Dynamic Range / Maximum Positive Value = 156.93 / 32767 = $4.789e^{-3}$ m/s^2.

5.12.7 SCALAR DATA

All of the scalar sensors (Light, Proximity, Humidity, Step Counter, Heart Rate, Temperature, Ambient Temperature, and Barometer) share a similar format. While there are differences in signed vs. unsigned and scale, the layout is similar for all of them.

Byte Number	Contents	Format
0	Sensor ID (Light, etc.)	
1	LSB	Signed or Unsigned 16 bit integer
2	MSB	

Byte Number	Contents	Format
0	Barometer	
1	LSB	Unsigned 24 bit integer
2	MIB	
3	MSB	

5.12.8 PARAMETERLESS SENSORS

Some sensors (Significant Motion, Step Detector, Tilt Detector, Padding) generate no actual data, so the FIFO will simply contain the sensor ID itself.

Byte Number	Contents	Format
0	Sensor ID (Significant Motion, etc.)	

5.12.9 META EVENTS

These indicate asynchronous, low periodicity events. These can be individually enabled or disabled in the Meta Event Control Parameter in the System Parameter Page. They can also be configured to enable or disable a host interrupt when they occur; this would occur even if there were no pending samples. The Initialized Meta Event will be the first event in the FIFO (following the current timestamp) after initialization. Once the host receives this, it is safe to interact with the parameter I/O system to configure the chip.

Byte Number	Contents	Format
0	META_EVENT	
1	Event ID	Unsigned 8 bit integer
2	Sensor ID / other	Unsigned 8 bit integer
3	Additional Info	Unsigned 8 bit integer

The possible Meta Events are listed below.

Table 18: Meta Event Types

Meta Event Type	Name	Byte 1	Byte 2	Default Enable State	Default Int Enable State
0	Not used				
1	Flush Complete	Sensor Type from FIFO_FLUSH register	Not used	Enabled	Disabled
2	Sample Rate Changed	Sensor Type	Not used	Enabled	Disabled
3	Power Mode Changed	Sensor Type	Power Mode	Disabled	Disabled
4	Error	Error Register	Debug State	Enabled	Enabled

Meta Event Type	Name	Byte 1	Byte 2	Default Enable State	Default Int Enable State
5	Magnetic Transient	0 = No transient	Not used	Enabled	Disabled
6	Cal Status Changed	Cal Status Value	Trans Comp	Disabled	Disabled
7	Stillness Changed	0 = not still 1 = now still 2 = long term still	Not used	Enabled	Disabled
8	Available				
9	Calibration Stable	1 = stable 0 = not stable	Not used	Enabled	Disabled
10	Reserved				
11	Sensor Error	Sensor Type	Sensor Status Bits	Enabled	Enabled
12	FIFO Overflow	Loss Count LSB	Loss Count MSB	Enabled	Disabled
13	Dynamic Range Changed	Sensor Type	Not used	Enabled	Disabled
14	FIFO Watermark	Bytes Remaining LSB	Bytes Remaining MSB	Disabled	Disabled
15	Self-Test Results	Sensor Type	Test Result	Enabled	Disabled
16	Initialized	RAM Ver LSB	RAM Ver MSB	Enabled	Enabled
17	Transfer Cause	Sensor Type	Not used	Enabled	Enabled

5.12.9.1 FLUSH COMPLETE

This meta event is the only official Android meta event. It will be inserted in the FIFO after a Flush FIFO request, whether there is any data in the FIFO or not. Flushing in Android means “transfer to host”, not “discard.” It will be inserted into the appropriate FIFO.

5.12.9.2 SAMPLE RATE CHANGED

This meta event occurs when a given sensor’s sample rate has been set for the first time, and/or when a requested change to the rate actually occurs. Byte 1 indicates the sensor type whose rate changed. It will be inserted only into the non-wakeup FIFO.

5.12.9.3 POWER MODE CHANGED

This meta event indicates when a given sensor powers up or down; the sensor type is passed in byte 1. It will be inserted only into the non-wakeup FIFO.

5.12.9.4 ERROR

The error meta event will only occur when unexpected internal firmware errors occur. It is important for the host to log such events, including byte 1 (error register) and byte 2 (debug state), to help with troubleshooting. The only proper recovery from this is to reset SENtral, reload the RAM patch, and reinitialize the sensors to the desired rates, latencies, and so on.

5.12.9.5 MAGNETIC TRANSIENT

This Magnetic Transient event reports when there is a significant change in the magnetic field around the mag sensor, such as electrical current turning on in a nearby device. The payload indicates 1 if transient is detected and 0 if back to normal. It will be inserted only into the nonwakeup FIFO.

5.12.9.6 CAL STATUS CHANGED

Payload bytes include the calibration status value and the transient compensation value. It will be inserted only into the non-wakeup FIFO.

5.12.9.7 STILLNESS CHANGED

This stillness changed meta event will be reported any time the internal stillness state has been updated. A payload of 0 indicates moving, 1 indicates initial stillness detected, and 2 indicates that long term stillness detected. Stillness is determined using the data from accel and gyro. If data is within +/- threshold, stillness is detected. For certain period of time, long term stillness is detected. Otherwise, moving is detected. It will be inserted only into the non-wakeup FIFO.

5.12.9.8 CALIBRATION STABLE

This meta event will be reported any time a calibration state has changes. A payload value of 1 indicates that calibrations is complete and stable, while a payload value of 0 indicates that calibration is ongoing and may not be reliable yet. It will be inserted only into the non-wakeup FIFO.

5.12.9.9 SENSOR ERROR

This meta event occurs when there is either a sensor mismatch between the RAM patch loaded and the hardware available, or, when a sensor fails to respond when expected. Byte 1 specifies the sensor type, and byte 2 specifies the sensor status bits for that sensor. This will indicate whether the error was due

to an I^2C NACK or Device ID mismatch. The Device ID mismatch error should never occur in a properly configured system in production. The I^2C NACK error indicates a hardware failure.

5.12.9.10 FIFO OVERFLOW

This meta event indicates when data loss has occurred due to the host being unable to read out FIFO data quickly enough. This may be intentional, such as when the AP is suspended, or it may be due to having too many sensors on at high sample rates with a slow host I^2C rate or slow driver implementation. It reports in bytes 1 and 2 a saturating count of lost bytes. Preceding this meta event, SENtral will insert an accurate Timestamp MSW and Timestamp LSW event, to ensure the host will be able to report accurate timestamps after the area of data loss.

SENtral, when it detects a FIFO overflow, automatically discards approximately 200 bytes of the oldest FIFO data in order to make room for the FIFO Overflow and Timestamp events, make room for new data, and ensure that at least some new data will appear in the FIFO between FIFO Overflow events, rather than become saturated with such events in worst case conditions.

It will be inserted into the appropriate FIFO.

5.12.9.11 DYNAMIC RANGE CHANGED

This event will be placed in the FIFO as soon as a requested change in dynamic range has occurred. The host may wish to wait for this event before changing the scale factor, in the event that a sensor whose dynamic range was changed was already on. Otherwise, the host could apply the wrong scale factor on some samples, and report invalid data as a result. It will be inserted only into the non-wakeup FIFO.

5.12.9.12 FIFO WATERMARK

This event occurs within a few dozen bytes of the specified watermark level. It will be inserted into the appropriate FIFO.

5.12.9.13 SELF-TEST RESULTS

Byte 1 indicates which sensor is reporting self-test results. It will be inserted only into the non-wakeup FIFO.

The Sensor Type field values for each physical sensor type that can perform self-test are:

- Accelerometer = ID 1
- Uncalibrated Gyroscope = ID 16
- Uncalibrated Magnetometer = ID 14

Byte 2 indicates the test status:

0: Test Passed
1: X Axis Failed
2: Y Axis Failed
3: X & Y Axes Failed
4: Z Axis Failed
5: X & Z Axes Failed
6: Y & Z Axes Failed
7: All Axes Failed or Single Test Failed (if testing of each axis cannot be done)

5.12.9.14 INITIALIZED

This is the first meta event reported after reset. The RAM version is reported in bytes 1 and 2, for convenience. It will be inserted only into the wakeup FIFO.

5.12.9.15 TRANSFER CAUSE

This special meta event is prepended to the FIFO so that the host gets an early indication of the reason for the host interrupt; this can be useful when in AP Suspend mode when there is a lot of batched data. Without this feature, the host will need to read and parse the entire FIFO before the reason for wakeup is apparent. This will appear if the reason was due to an on-change, special, or one-shot sensor having been triggered. The Sensor ID will indicate which sensor this was. It will be inserted into the appropriate FIFO.

If the transfer cause was due to a wakeup sensor, this meta event will be sent as a Wakeup Meta Event (ID 248); otherwise, it will be sent as a Non-Wakeup Meta Event (ID 254).

Note that the host will need to read and process the start of the FIFO before finishing the entire FIFO in order for this to be effective at providing low latency. Splitting a host transfer is done most easily if the number of bytes transferred (prior to the final transfer) is a multiple of 50 bytes. See section2.5 for more details. We recommend that the host only perform this split when coming out of suspend; when not suspended, the additional time required for processing multiple I^2C transfers for a single host interrupt will increase power requirements and reduce I^2C bus bandwidth utilization unnecessarily.

5.12.9.16 TIMESTAMP LSW

SENtral-A2 outputs the Timestamp LSW event before every sensor sample whose timestamp is different from the previous Timestamp LSW placed in the FIFO. This even contains the least significant 16 bits of a sensor sample's 32 bit timestamp. If a sensor sample is about to be inserted in the FIFO which shares the same timestamp, this event is not inserted, in order to conserve FIFO space.

5.12.9.17 TIMESTAMP MSW

This event will be output when the most significant 16 bits of a sensor sample timestamp differs from the previous Timestamp MSW event placed in the FIFO. It, and the Timestamp LSW event, will be placed in the FIFO after a FIFO overflow as well.

> ***NOTE:*** *there will be no notification when the Timestamp MSW itself overflows, which happens every 37.28 hours. It is up to the host software to properly handle such timestamp wraparounds. One way is to periodically wake up and query the current Timestamp parameter. This wraparound will be particularly apparent for long-sleeping situations with the APSuspended bit set and the FIFO accumulating only on-change sensor events.*

5.13 FIFO EMPTYING PROTOCOL

As mentioned previously, registers 0x00 to 0x31 form the data transfer area. Unlike the other registers, these have strict protocols for host access.

When the host receives the host interrupt from SENtral, it should first read the Bytes Remaining registers to determine the number of bytes currently in the FIFO. It should then read all those bytes in one or more consecutive I^2C read operations.

This area is double buffered. FIFO data is placed in the two buffers by the internal processor, as needed, to transfer the number of bytes in the Bytes Remaining registers. The host only needs to start reading at register 0, and continue reading until the entire transfer is complete. The register address being read auto-resets when reading past 0x31, which also causes the double buffers to swap; when this happens, the internal processor fills the just emptied buffer with the next set of FIFO data.

In this manner, the host will be able to set up a single I^2C transfer for the entire Bytes Remaining value.

The transfer should be started as:

<START>
<SENTRAL Address + W>
<0x00 (register address 0x00)>
<REPEATED_START>
<SENTRAL Address + R>
<read data…>
<STOP>

Due to the fact that the double buffers are 50 bytes in size, it is recommended that if the host needs to break up a long FIFO read into multiple smaller individual I^2C reads, it do so in multiples of 50 bytes, and start reading each one starting at register address 0.

The current hardware cannot tell the internal processor if the host has read less than 50 bytes in any given buffer. As a result, it is up to the host to: read in multiples of 50 bytes; read the entire buffer in one transfer; or follow the Pause and Resume Mechanism below to begin each transfer at the proper I^2C address.

NOTE: *reading more data than the Bytes Remaining registers indicate can lead to data corruption; always read the amount specified and no more.*

5.13.1 HOST INTERRUPT BEHAVIOR

The host interrupt asserts high on reset or power up, and is cleared by loading and executing a RAM patch or by reading the Host Status register (0x35).

During normal operation, it will also assert high when data is available in the FIFO and it is time to notify the host.

A host interrupt is generated when:

1. An enabled sensor has a zero max report latency (timeout) and has generated a sample
2. A sensor has a non-zero max report latency, it has a sample in the FIFO, and it has timed out before any other sensor with a shorter latency or zero latency generates a sample
3. Same as 2, but the FIFO Watermark is non-zero, and it has been exceeded before the latencies timed out
4. A meta event has occurred which has its interrupt enable set (by default, only the initialized event, the transfer cause event, internal firmware errors, or sensor hardware errors can generate interrupts)
5. The AP is in suspend mode, the Proximity and/or Significant Motion sensors are enabled, and only a Proximity or Significant Motion event has occurred

6. The AP is in suspend mode, the FIFO watermark is non-zero, and the FIFO watermark meta event is both enabled and interrupt enabled; when the watermark is reached, the AP will be woken up

The host interrupt will remain asserted until the host has emptied the FIFO or has aborted the transfer with the Abort Transfer bit in the Host Interface Control register (0x55). It may then reassert immediately depending on the notification criteria above.

The host interrupt will always go low for a minimum of 24 µs between the time the host receives it and empties the FIFO and any subsequent transfer. This is to ensure that either level or edge-triggered interrupts can be used in the AP.

The time at which the rising edge of the interrupt occurred will be accessible as the Host IRQ Timestamp parameter of the System Parameter Page. This timestamp is a 32 bit counter, incremented at 32 KHz, and synchronized with the real time counter used to timestamp sensor data.

5.13.2 PAUSE AND RESUME MECHANISM

The transfer can be paused by ending the I^2C transfer with a <STOP> condition, and resumed later by issuing:

<START>
<SENTRAL Address + W>
<restart_register_address>
<REPEATED_START>
< SENTRAL Address + R>
<read data…>
<STOP>

The restart_register_address must be calculated by the host as:

restart_register_address = bytes_transferred_so_far MOD 50

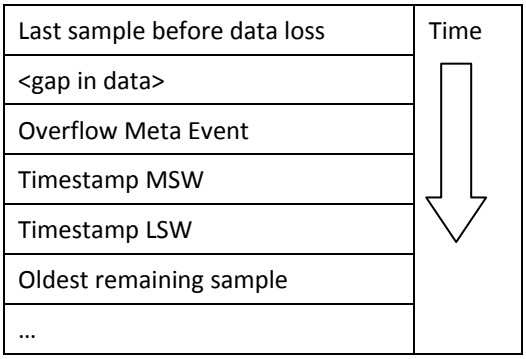
The Update Transfer Count bit in the Host Interface Control register can be used to get an updated value in registers 0x38/0x39 (Bytes Remaining). SENtral-A2 cannot detect the pause in transfer so it cannot automatically update the transfer count itself. However, this updated count will not affect the length of the current transfer; it simply advises the host of how much will eventually be transferred between the

current and next host interrupt. Again, reading more data from the FIFO than the original Bytes Remaining value is not recommended.

5.13.3 FIFO OVERFLOW HANDLING

In the event that the FIFO overflows, due to excessively high sample rates and/or slow reading by the host (or even the host being asleep), the oldest data will be will discarded first.

Once the host begins reading again, we will insert an Overflow Meta Event (if enabled) into the buffer at the point of data loss, followed immediately by the Timestamp MSW and Timestamp LSW values for the continuation point, followed finally by the oldest remaining data. In this way, the host can know that data was lost and know an accurate timestamp for the data that continues after the gap – after the lost data.



5.13.4 APPLICATION PROCESSOR SUSPEND PROCEDURE

When the application processor goes into suspend mode, it should tell SENtral by first writing a 1 to the AP Suspend bit in the Host Interface Control register, so that only wake-up sensors (Proximity and Significant Motion) issue a host interrupt. It is recommended that the host precede this with a FIFO Flush and complete emptying of the FIFO, so that the host is not immediately woken by mistake.

5.13.5 APPLICATION PROCESSOR WAKEUP PROCEDURE

When the AP wakes, it should notify SENtral by clearing the AP Suspend bit in the Host Interface Control register, so that all enabled sensors can (if so configured) issue a host interrupt as needed. If there is data pending, the SENTRAL-A2 will detect this action and start a host transfer by setting the Bytes Remaining registers and then assert the host interrupt as usual.

If the AP did not notify SENtral of entering and leaving suspend mode, but instead masked the host interrupt line, the AP may find the host-interrupt line already asserted when it comes out of suspend.

The Bytes Remaining registers will contain an old value, which, while valid, will likely be smaller than the current number of bytes in the FIFO. Once the host has transferred this smaller amount of bytes, the next host transfer will cover the remaining amount to be transferred.

5.13.6 NON-COMPLIANT HOSTS

Hosts must read the Bytes Remaining registers prior to starting the FIFO emptying protocol in order to read buffered data. Hosts must read at least that number of bytes of data; if it reads more, pad bytes (0s) will be returned. Hosts which read less than this will not receive a new host interrupt until they finish reading the remainder of the buffer, or until they request a transfer abort (which will cause loss of data). Hosts could read in fixed size blocks until, after parsing each block, it detects one or more pad bytes in the stream.

NOTE: *Any data read after the first pad byte should be discarded, as it may be a repetition of previous data. The host should stop reading as soon as this first pad byte is detected.*

5.13.7 RECOVERY FROM LOSS OF SYNC

If the host cannot make sense of the data it is reading from the FIFO, it may have lost sync. While this should never occur, if it does, the host should abort the transfer by writing a 1 to the Abort Transfer bit in the Host Interface Control register. The next transfer will start with a whole (not partial) block of data.

Alternatively, the host could simply finish the current transfer, discard what it is receiving, and then wait for the next transfer as signaled by a host interrupt. By definition, the start of a FIFO read will always begin at the start of a valid sensor event.

5.13.8 PADDING DATA

FIFO reads beyond the value previous read from the Bytes Remaining registers will result in the host reading 0s, which indicate end of data. A 0 sensor ID is defined to be a single byte NOP sensor event.

NOTE: *due to hardware limitations, if the amount of FIFO data to transfer modulus 50 is greater than 0 and less than 3, SENtral-A2 will add enough padding bytes to bring up the size of the final transfer to 4 bytes minimum.*

5.13.9 ABORTING A TRANSFER

The abort transfer action can be used to recover from loss of sync or for fast emergency shutdown. As mentioned previously, there will likely be approximately 100 bytes of FIFO data that are discarded when

the transfer is aborted, including any partial sensor sample or event that started in the 100 byte transfer registers and was scheduled to be finished in the next buffer.

If the host wishes to respond to the next host interrupt and parse out the remaining FIFO data, it will see a valid sample at the start, but, SENtral-A2 makes no attempt to provide an accurate starting timestamp in this case. So, the host needs to read, parse, and discard events up until the first Timestamp LSW event and/or Timestamp MSW event.

If there was one or more continuous output sensors enabled, then the host need only discard up to the first Timestamp LSW event; if the timestamp value is greater than the previous value seen before the abort, the host's knowledge of the timestamps of sensor samples that follow is accurate. However, if the first Timestamp LSW value seen after the abort is less than the previous value, the MSW has incremented, and the Timestamp MSW value was lost in the 100+ bytes that were discarded.

If there were only on-change, special, or one-shot sensors enabled, the host cannot be guaranteed to know the correct 32 bit timestamps for any events it sees after the abort until the next Timestamp MSW is read and parsed. It may still want to report the events it does see, as they may be important to the user, but the host will need to estimate the timestamps.

5.14 FIFO PARSING EXAMPLES

5.14.1 ACCELEROMETER AND STEP COUNTER

This example assumes the FIFO watermark is disabled, the Accelerometer is configured for 50Hz and 40 millisecond latency; the only other sensor on is the Step Counter, which is set for 0 latency. This is not the first sample; the last known Timestamp MSW = 0x0010 (~32 seconds after startup). At 50Hz, the timestamp will increment approximately 640 ticks between samples (640/32000 = 20ms = 1/50Hz).

Steps are listed in time order from the host's perspective.

1. Host interrupt received from SENTRAL-A2
2. Host reads registers 0x38-0x39; value read is 25
3. Host reads registers 0x00-0x18; data received:

Byte Number	Value	Meaning
1	252	Timestamp LSW Event
2	0xF8	LSB
3	0xFF	MSB; Time = 0x0010FFF8 / 32000 = 34.81575 seconds
4	1	Accelerometer Event
5	0xFE	X LSB
6	0xFF	X MSB; X value = 0xFFFE = -2 = -0.009578 m/s^2
7	0x05	Y LSB
8	0x00	Y MSB; Y value = 0x0005 = 5 = 0.023945 m/s^2
9	0x69	Z LSB
10	0x08	Z MSB; Z value = 0x0869 = 2153 = 10.310717 m/s^2
11	0x02	Status = medium accuracy
12	253	Timestamp MSW Event
13	0x11	LSB
14	0x00	MSB
15	252	Timestamp LSW Event
16	0x78	LSB
17	0x02	MSB; Time = 0x00110278 / 32000 = 34.83575 seconds
18	1	Accelerometer Event
19	0xFD	
20	0xFF	X value = -3 = -0.014367 m/s^2
21	0x08	
22	0x00	Y value = 8 = 0.038312 m/s^2
23	0xFC	
24	0x07	Z value = 0x07FC = 2044 = 9.798 m/s^2
25	0x02	Status = medium accuracy

4. The host decodes this, and as indicated in the meaning column, learns that:
    a. The FIFO contained the least significant word (LSW) of the timestamp; using that and the previous timestamp MSW gives us an actual timestamp for the next sample as 34.81575 seconds

b. There are two samples of the Accelerometer, which makes sense because the sample latency divided by the sample period is 2
c. The samples are separated by a full timestamp (both the MSW and the LSW) because the LSW overflowed between samples
d. Using the default scale factor of 4.789e-3 m/s2, it can calculate the actual values of the Accelerometer X, Y, and Z axes

5. The host now waits for the next interrupt
6. The human was stepping during this; right after the next accelerometer sample, the Step Counter outputs a new step count (NOTE: the accelerometer data is just dummy data; real data taken when a person is walking will of course change constantly on all axes more than this data does)
7. Host interrupt received from SENtral
8. Host reads registers 0x38-0x39; value read is 14
9. Host reads registers 0x00-0x0D; data received:

Byte Number	Value	Meaning
1	252	Timestamp LSW Event
2	0xF8	LSB
3	0x04	MSB; Time = 0x001104F8 / 32000 = 34.85575 seconds
4	1	Accelerometer Event
5	0xFF	X LSB
6	0xFF	X MSB; X value = 0xFFFE = -1 = -0.004789 m/s^2
7	0x11	Y LSB
8	0x00	Y MSB; Y value = 0x0011 = 17 = 0.081413 m/s^2
9	0x82	Z LSB
10	0x07	Z MSB; Z value = 0x0782 = 1922 = 9.204458 m/s^2
11	0x02	Status = medium accuracy
12	19	Step Counter Event
13	0x01	LSB
14	0x00	MSB = 1st step

10. The host decodes this, and learns that:
    a. There is a new timestamp, and the sample time is now 34.85575 seconds
    b. There is one accelerometer sample
    c. There is a step counter sample
    d. The step counter, since it had a latency of zero, flushed the FIFO earlier than it would have been by the accelerometer's latency of 40ms
11. Approximately 40 milliseconds later (the latency for the accelerometer), another host interrupt followed by reading of two accelerometer samples will occur (not shown)

5.15 FIRMWARE IMAGE UPLOAD AND FORMAT

Table 19 shows the SENtral-A2 firmware image format being used for the upload process. The whole image can be stored in the external EEPROM and uploaded by SENtral automatically. During upload from EEPROM, SENtral checks the beginning of firmware image magic numbers. If they match the expected value, it uploads the Firmware Image while it calculates CRC32 over incoming data. When SENtral is uploading the image from external EEPROM, the I^2C Speed is determined by the I2CClockSpeed Flag bits. Note that the I2CClockSpeed in the EEPROM image should be set with respect to the slowest device on I^2C sensor bus (including the EEPROM).

At the end of the upload, the calculated CRC32 is compared with its value stored in the header. If they match, and the EEPROM No Execute bit is not set in the header, the firmware image is executed.

When the direct upload from host to RAM is being used, the host software should check the magic numbers and compare the expected ROM version with the SENtral-A2 ROM Version register (Table 20). The Upload Image Length (UIL) is always a multiple of 4B. The host must use the correct byte ordering during upload as specified in Table 23. Data should be uploaded using one or more I^2C Write transactions to the Upload Data register. Each I^2C Write to the Upload Data register should contain one or more 4B data groups. For example, the upload of a 2048B image can be broken down into 128 I^2C Writes each containing 16B of the image.

At the end of upload, the host is responsible for comparing the calculated CRC32 with the one stored in the firmware image. I2CClockSpeed bits do not apply when the host is performing a direct upload. The host should determine the best host upload speed based on the host's I^2C bus parameters.

The last 44B of the uploaded firmware image contain the SENtral-A2 Configuration Data Structure, shown in Table 21 which can be used by the host to identify software versions and configuration.

Table 19: Firmware Image Format

Byte Offset	Contents	Note
0x00	Image Signature Lower Byte	0x2A
0x01	Image Signature Upper Byte	0x65

Byte Offset	Contents	Note
0x02-0x03	Flags	Bit 0 – EEPROM No Execute Bit 8…10 – I2CClockSpeed 000 – 1000 Kbit/s 001 – 833 Kbit/s 010 – 400 Kbit/s 011 – 333 Kbit/s 100 – 100 Kbit/s Others – 83 Kbit/s Bit 11…14 – ROM Version Expected Others - Reserved
0x04-0x07	CRC32 of uploaded image	Stored in Little Endian Form
0x08-0x0B	Reserved	0x00000000
0x0C-0x0D	Uploaded Image Length (UIL)	Stored in Little Endian Form
0x0E-0x0F	Reserved	0x0000
0x10-0x10+UIL-49	Uploaded Firmware Image (Instructions & Data)	CRC32 is calculated over uploaded image
0x10+UIL-48 0x10+UIL-1	Uploaded Firmware Image (SENtral Config Data Structure) *See* Table 21.	

Table 20: Expected ROM Version

ROM Version Expected	Content	ROM Version
0x00	Any ROM version	Any value
0x01	ROM version DI01	0x27F2
0x02	ROM version DI02	TBD

Table 21: SENtrasl-A2 Config Data Structure (CDS)

Byte Offset	Contents	Note
0x00	CDS Signature Lower Byte	0x8B
0x01	CDS Signature Upper Byte	0xC8
0x02-0x07	Reserved	
0x08-0x09	Ram Version	Firmware RAM version number

Byte Offset	Contents	Note
0x0A	CDS Version	Configuration data structure version
0x0B	Boot Protocol	Used for standalone applications
0x0C-0x13	Pin Selection	Bit 0-3 Host IRQ pin selection Others – reserved
0x14-0x1B	Pull Selection	0=no pulls, 1=pull-down, 2=pull-up, 3=keep firmware defaults Bit 0..1 GPIO0 pull selection Bit 2..3 GPIO1 pull selection Bit 4..5 GPIO2 pull selection Bit 6..7 GPIO3 pull selection Bit 8..9 GPIO4 pull selection Bit 10..11 GPIO5 pull selection Bit 12..13 GPIO6 pull selection Others – reserved
0x1C-0x2B	Device Name	16 character string

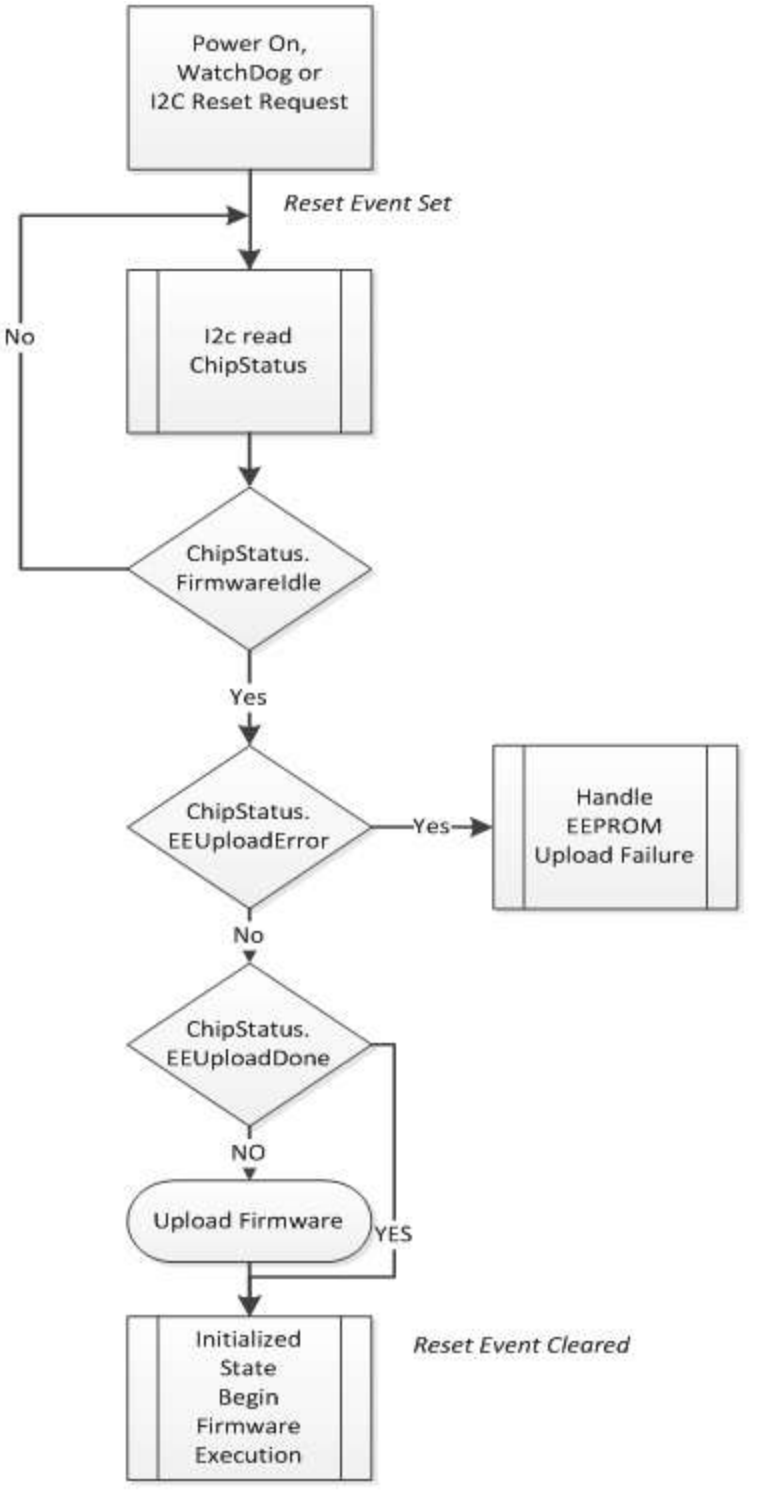


Figure 15: Initialization Sequence

5.15.1 EEPROM UPLOAD FAILURE

In the event that the firmware upload has failed via EEPROM, the host may take a number of steps as it sees fit:

- Upload firmware from the host into SENtral-A2 program RAM
- Download the firmware image from EEPROM and verify its contents
- Upload firmware from the host into the EEPROM and issue a reset request

If a direct firmware image upload is needed, the sequence and registers below should be used:

Table 22 : Upload Registers

Register Name	Register Address	Register Value
Chip Control	0x34	1 – CPU Run Request 2 – Host Upload enable
Upload Address	0x94-0x95	Initial RAM address (reset to 0x0000)
Upload Data	0x96	Data to be uploaded
CRC Host	0x97–0x9A	CRC32 of the uploaded data since host upload was enabled
Reset Request	0x9B	1 – Reset SENtral

The host should determine the I^2C upload speed based on the host's I^2C bus electrical parameters and attached devices.

1. Reset SENtral-A2. This is only needed if the previous event was not a reset event.
2. Verify the Firmware Image (see Table 19)
3. Image Signature in header matches expected value
4. Uploaded Image Length = Firmware Image File Size – 16B
5. Uploaded Image Length is multiple of 4B
6. Firmware version matches ROM version (see Table 20).
7. Enable upload mode (Chip Control)
8. Write initial RAM address into Upload Address register (this address is reset to 0x0000 by default)
9. Upload the Uploaded Image from range 0x10-0x10+UIL-1 to the Upload Data register using multiply of 4B during each write. Note that each group of 4B should be sent in byte reverse order (endian swapped) – see Table 23 for an example.
10. Verify the uploaded CRC matches the CRC in the header.
11. Disable upload mode.
12. Issue CPU run request only if uploaded CRC matches the value in the header.

Table 23: Host Upload Data Order

Byte order in firmware image	**i**	**i+1**	**i+2**	**i+3**	**i+4**	**i+5**	**i+6**	**i+7**
Firmware image example	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08
Byte order during host upload	**i+3**	**i+2**	**i+1**	**i**	**i+7**	**i+6**	**i+5**	**i+4**
Example during host upload	0x04	0x03	0x02	0x01	0x08	0x07	0x06	0x05

5.15.2 I2C PASS-THROUGH MODE

SENtral-A2 supports the use of pass-through mode, allowing the host to talk directly to sensors. Table 24 shows the registers used to enable and disable pass-through mode. Setting of registers for clock stretching support at different speeds is shown in Table 25.

Table 24: Pass-Through Registers

Register Name	Register Address	Register Value
Pass-Through Config	0xA0	2 - Sensor To Host En 1 - Pass-through enabled
SCL Low Cycles	0x9F	Minimal duration of SCL low period on I^2C Sensor bus when Sensor To Host En = 1
Pass-Through Ready	0x9E	1 – Pass-through ready 0 – Pass-through unavailable

Table 25: Recommended Setting for Pass-Through Mode

I^2C Speed	Communication without Clock Stretching Support		Communication with Clock Stretching Support	
	SCL Low Cycles	Pass-Through Config	SCL Low Cycles	Pass-Through Config
Standard Mode	0x00 (Not used)	0x01	46	0x03
Fast Mode			12	
Fast Mode +			4	

In order to use this mode, the following sequence must be performed:

1. Disable the SENtral algorithm by issuing a standby request or shutdown request.[1]
2. Set SCL Low Cycles (required only if target sensor supports SCL clock stretching.)
3. Enable pass-through mode (Write Pass-Through Config register.)
4. Check that pass-through mode has been enabled (Read Pass-Through Ready register)
5. Talk to any sensors attached to SENtral as if they were attached to the host.
6. Disable pass-through mode
7. Resume the SENtral algorithm if desired.

[1]*Algorithm standby is preferred when faster resume of operation is required (internal algorithm state is preserved). Shutdown request is typically used when complete restart is required after the end of pass-through mode (for example after the EEPROM is programmed).*

5.15.3 WRITING RAM PATCH INTO EEPROM

General steps to program the EEPROM attached to SENtral-A2 Sensor Bus are as follows:

1. Enter Pass-Through Mode (as described in the previous section). Make sure that SENtral is not in Normal Operation (it can be either in Initialized or Standby State).
2. Copy the contents of the RAM patch into EEPROM, starting with EEPROM byte address 0x0000. Please refer to EEPROM datasheet for details about correct timing between writes.
3. Verify the programming success by reading back the EEPROM content and comparing it to the original.
4. Leave Pass-Through Mode and issue I2C Reset Request (Write 0x01 into Reset Request register). This will force SENtral to reinitialize from EEPROM.

5.16 PARAMETER I/O PROCEDURE

This section describes the SENtral-A2 parameter transfer mechanism implemented in the SENtral-A2 and Host software. The Parameter Load and Save procedures can be invoked during normal algorithm operation. A Parameter is any fusion algorithm parameter or knob, or host interface configuration, information, or status value, which can be read or written by the Host. The implementation ensures atomicity of parameter reading and writing. Communication with the Host is handled by the internal MCU's I^2C Slave Write interrupt routine. Once the procedure is terminated (Parameter Request register is cleared) and the Host modified any of the parameters, new parameters are atomically applied into the fusion or other data structures.

U7184 Firmware Outerloop

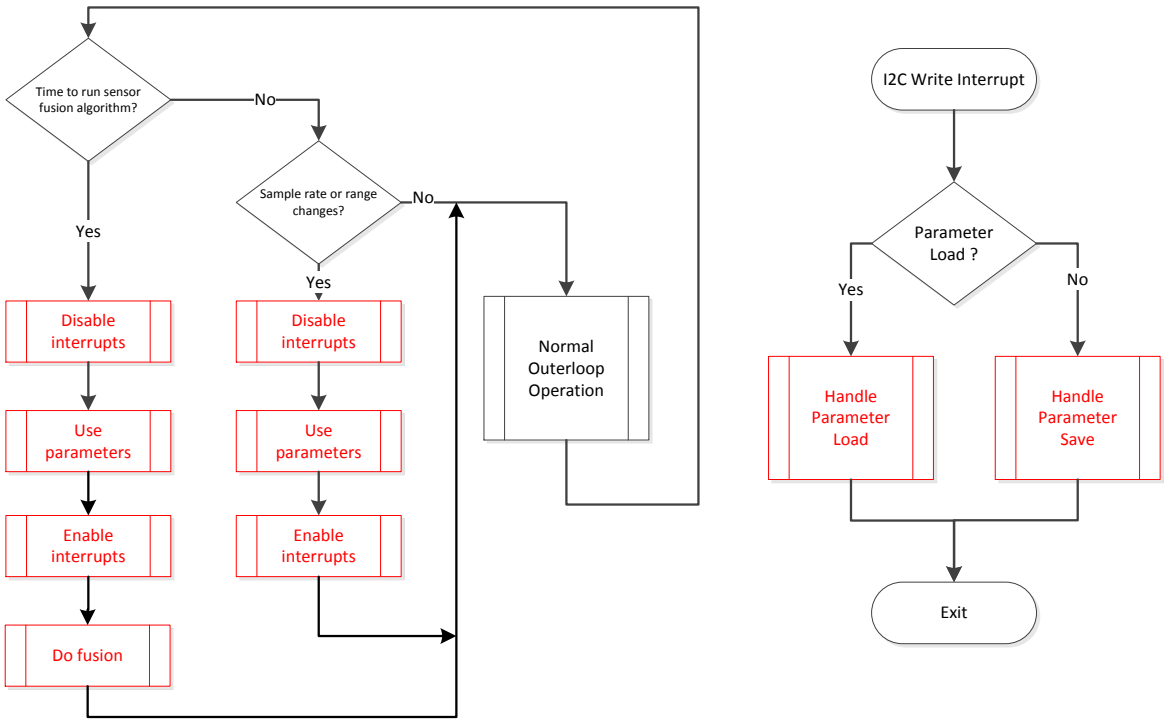


Figure 16 : SENtral-A2 Operation and Parameter Load/Save

5.16.1 PARAMETER LOAD AND SAVE IMPLEMENTATION

This section lists tasks that the host must perform in order to ensure proper operation with SENtral-A2.

Table 26 shows the set of registers used for the Parameter Save/Load procedure. The procedure is invoked and terminated by setting or clearing the Parameter Request register. Three registers are used to implement a handshake mechanism between SENtral-A2 and the Host. Values shorter than 8B (Load) or 16B (Save) can be transferred by using only part of Save/Load Byte registers, by setting the number of bytes to transfer in the upper nibble of the Parameter Page Select register. Note that data must be always stored in Little Endian fashion.

Table 26: Registers used in Save and Load

Register Name	Register Address	INTERNAL Access Type	Host Access Type
Load Parameter Bytes 0-7	0x5C-0x63	R/O	R/W
Parameter Page Select	0x54	R/O	R/W
Parameter Request	0x64	R/O	R/W
Parameter Acknowledge	0x3A	R/W	R/O
Saved Parameter Bytes 0-15	0x3B-0x4A	R/W	R/O

5.16.2 PARAMETER LOAD PROCEDURE

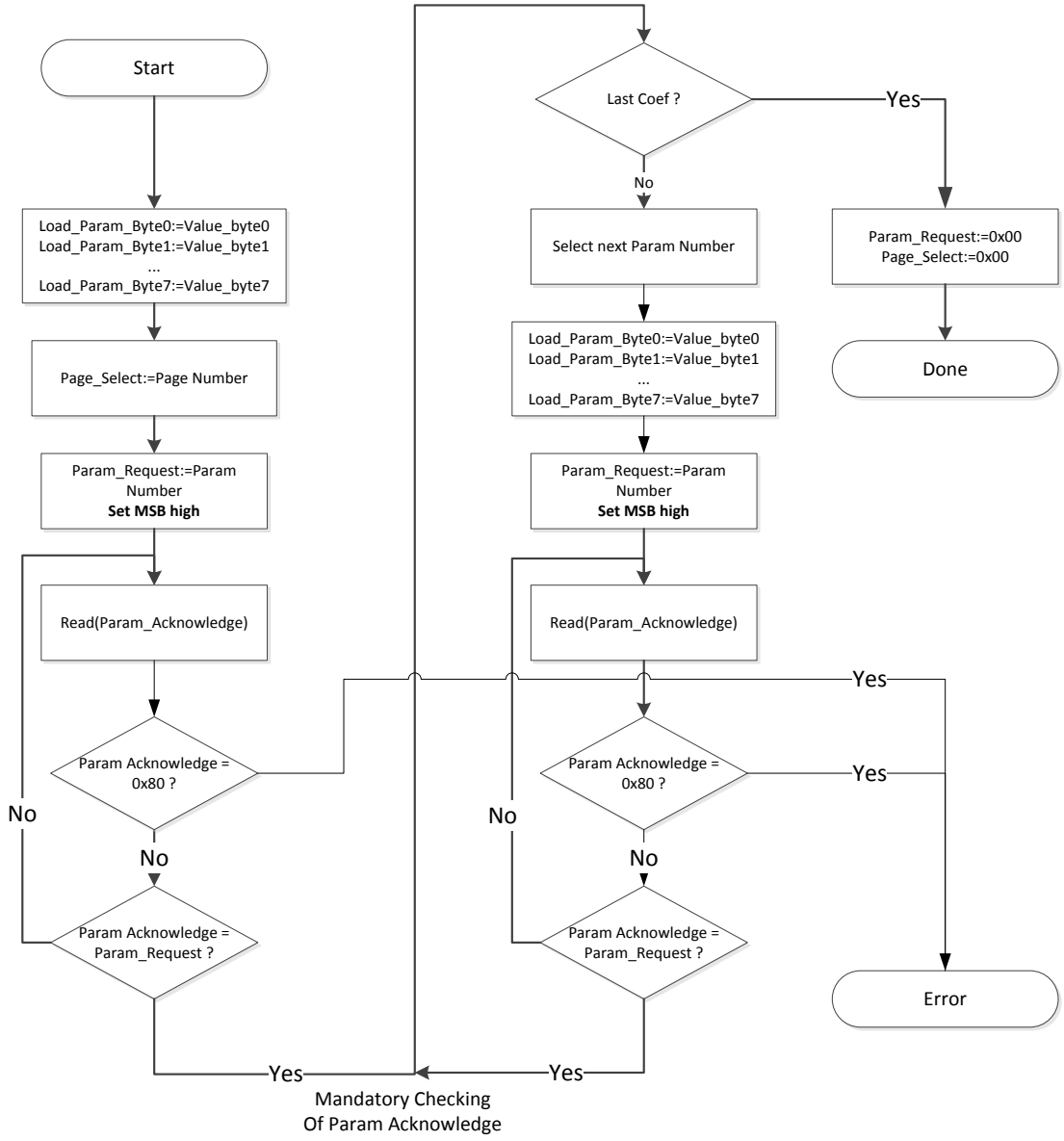


Figure 17: Parameter Load Procedure on Host

Figure 17, above, shows the procedure used by Host to Load Parameter data. The first parameter must be written into the Load Parameter Bytes registers together with the non-zero Page Number into the Parameter Page Select register, then the non-zero Parameter Number into Parameter Request register. MSB of Parameter Request register should be set high to indicate Load procedure. SENtral-A2 will

acknowledge the reception of the first Parameter by setting Parameter Acknowledge equal to Parameter Request. If the requested Page Number or Parameter Number is not implemented or supported, Parameter Acknowledge will instead equal 0x80. The Host should always check the Parameter Acknowledge value after the first Parameter since delay of code execution can be different between various software versions.

Once SENtral acknowledges the first parameter, Host can start writing additional Parameters in a loop. The Host terminates the Parameter Load procedure by setting Parameter Request:=0x00 and Parameter Page Select:=0x00.

5.16.3 PARAMETER SAVE PROCEDURE

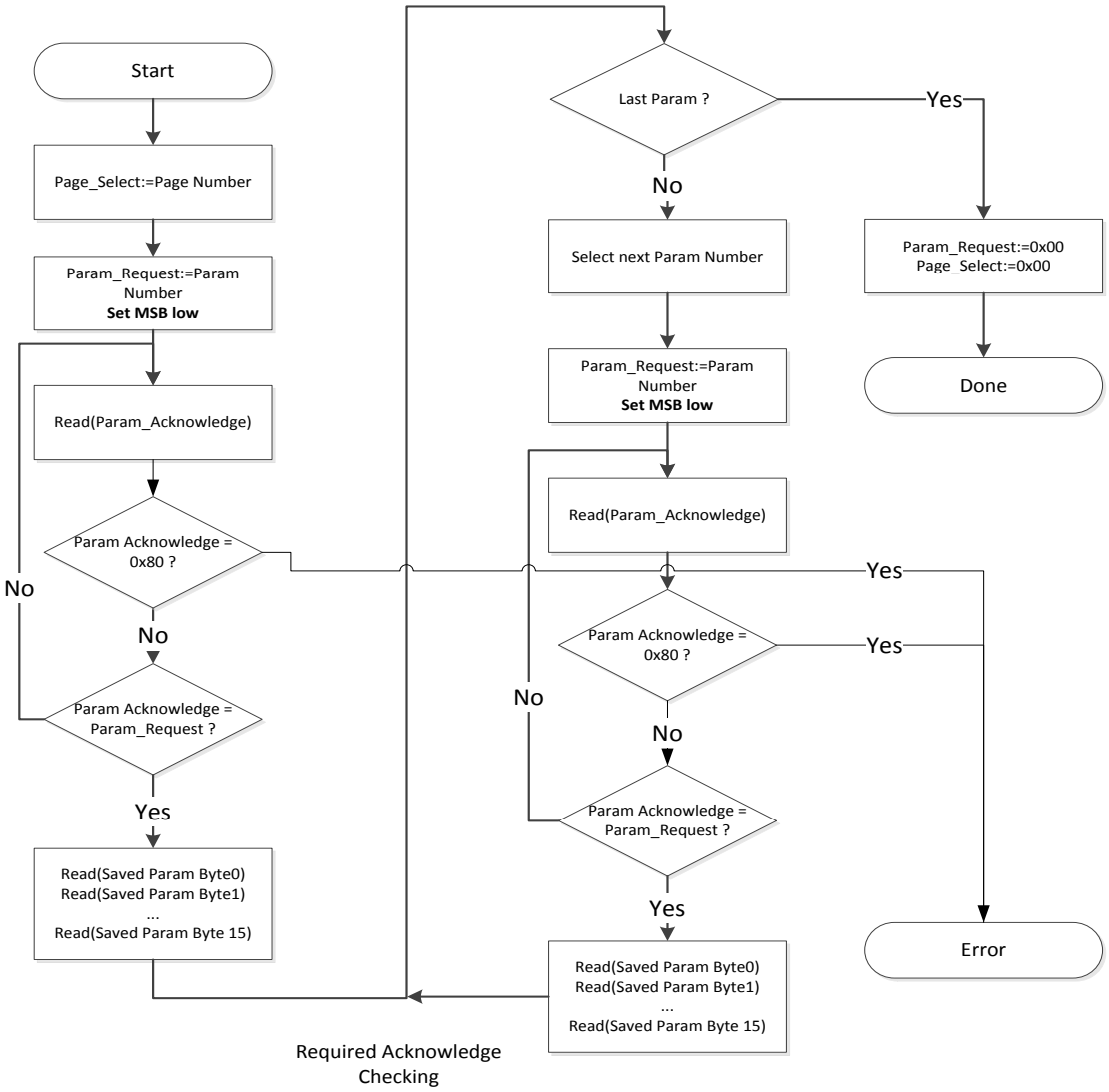


Figure 18: Parameter Save Procedure

The host starts the Parameter Save procedure when it writes the Parameter Page Select register with a non-zero Page Number, and then writes the Parameter Request register with a non-zero Parameter Number. The MSB of the Parameter Request register should be set low to select the Save procedure. The Host should perform repetitive reads of the Parameter Acknowledge register until it contains the

requested Parameter Number or 0x80 indicating an error (incorrect Page Number and/or Parameter number). If the requested Parameter Number is returned, it can then read the Saved Parameter Bytes 0-15 registers to obtain the Parameter value. (If the host does not require all 16 bytes of the parameter, it can set a smaller request size in the upper nibble of the Parameter Page Select register at the start of the procedure). Note that the Host can read the Parameter Acknowledge and Saved Parameter Bytes 0-15 registers using a single 17B I2C read transaction to improve efficiency. The Host can continue reading additional Parameters by setting a new value in the Parameter Request register and repeating the handshake. The Save Procedure is terminated by the Host by writing Parameter Request:=0x00 and Parameter Page Select:=0x00.

Duration of Load / Save Procedure

The parameter transfer time depends on CPU utilization and the time when the request is received. Based on usual sensor rates, the device should enter the procedure within __ ms after the request.

Once the procedure is invoked, the Loading of a single 8B parameter (Host to SENtral) requires the transmission of 92 bits (90 data bits + Start and Stop bit). The associated handshake adds 30 bits = 122 bits per parameter. Assuming a 1 Mbps I2C speed, the expected load time is 92 us per parameter. Saving a 16B parameter (from SENTRAL-A2 to Host) requires 144 bits. The associated handshake adds 30 bits = total 174 bits per parameter.

Parameter Number Usage

Note that the MSB of the Parameter Request register is used to select between the Load and Save operation. As well, a value of 0 has a special meaning. This allows space for up to 127 4B parameters. The Parameter Page Select register allows for up to 15 separate sets of 127 parameters.

5.16.4 WARM STARTUP PROCEDURE

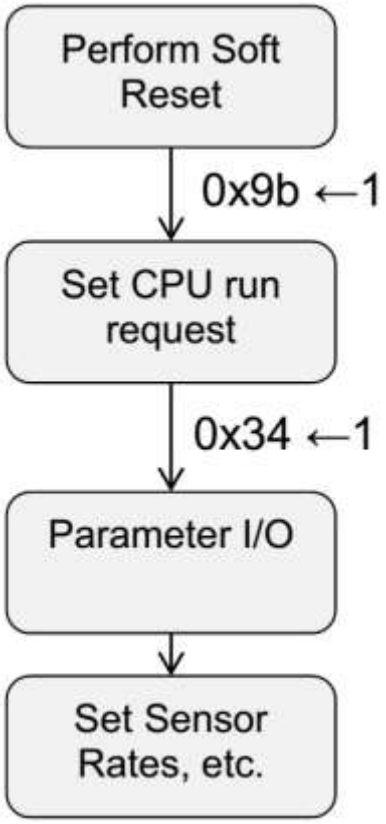


Figure 19: Warm Start Procedure

In order to perform a warm start, it is necessary to run the parameter I/O functions before the sensor fusion algorithm is allowed to run for the first time. This is because the sensor fusion algorithm state is only initialized during the initial run, and the internal structure in which sensor fusion algorithm states are stored is unavailable externally after this point. Therefore a warm start can be performed during initial startup of the system, or at any time with a soft reset (see Figure 20). The sensor fusion algorithm is prevented from execution until the non-zero rates are set after completion of the Parameter I/O procedure.

6 PACKAGE INFORMATION

6.1 PIN DESCRIPTIONS

Table 27 : SENtral-A2 Pin Description

Pin	Name	Application	I/O Type	Description
D1	VDD		PWR	External Supply Connection
D3	VREG		PWR	Regulator Output
D2	VSS		PWR	Ground Connection
B1	SCLS		IO	I^2C Host SCL Clock Line
A1	SDAS		IO	I^2C Host SDA Data Line
B4	SCLM		IO	I^2C Sensor SCL Clock Line
A4	SDAM		IO	I^2C Sensor SDA Data Line
D4	GPIO[0]	SIRQ[0]	IO / PUPD	GPIO[0] / Sensor Interrupt [0]
C4	GPIO[1]	SIRQ[1]	IO / PUPD	GPIO[1] / Sensor Interrupt [1]
A3	GPIO[2]	SIRQ[2]	IO / PUPD	GPIO[2] / Sensor Interrupt [2]
B3	GPIO[3]	GPIO	IO / PUPD	GPIO[3] / Auxiliary Interrupt[0]
A2	GPIO[4]	GPIO	IO / PUPD	GPIO[4] / Auxiliary Interrupt[1]
B2	GPIO[5]	GPIO	IO / PUPD	GPIO[5] / Auxiliary Interrupt[2]
C1	GPIO[6]	HIRQ	IO / PUPD	GPIO[6] / Host event
C3	GPIO[7]	GPIO/SA0	IO / PUPD	GPIO[7] / I^2C Host Slave Address bit [0]
C2	TM		I / PD	Test Mode Entry
PWR=Power Supply Connections; I=Digital Input; IO=Digital Input / Output; PU=Pull-Up; PD=Pull-Down				

6.2 BUMP LOCATION

(All dimensions in mm)

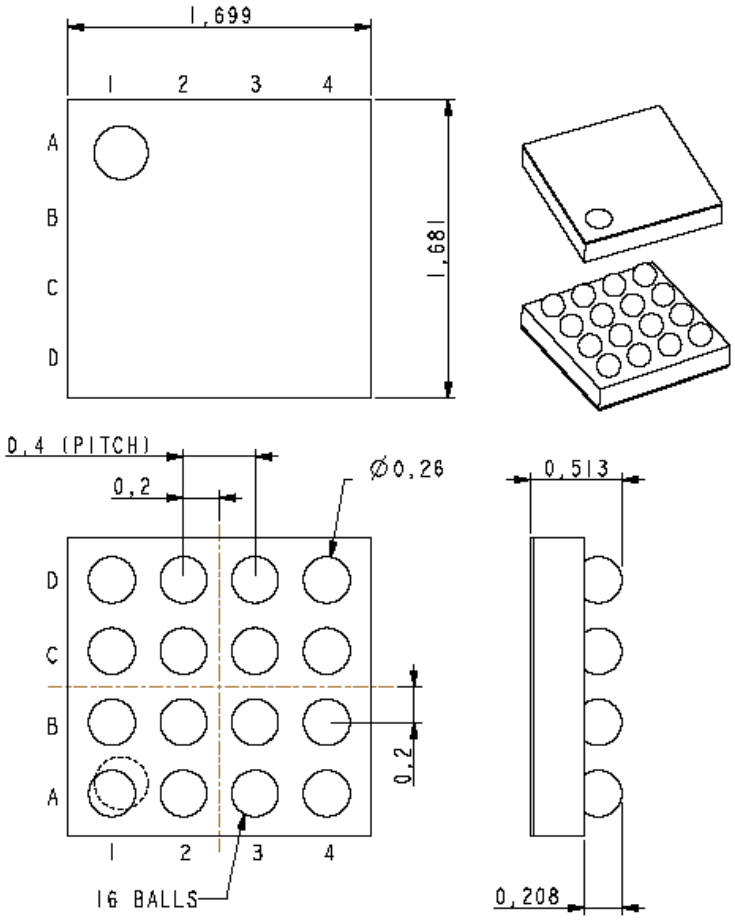


Figure 20: SENtral-A2 WL-CSP views

6.3 TAPE AND REEL

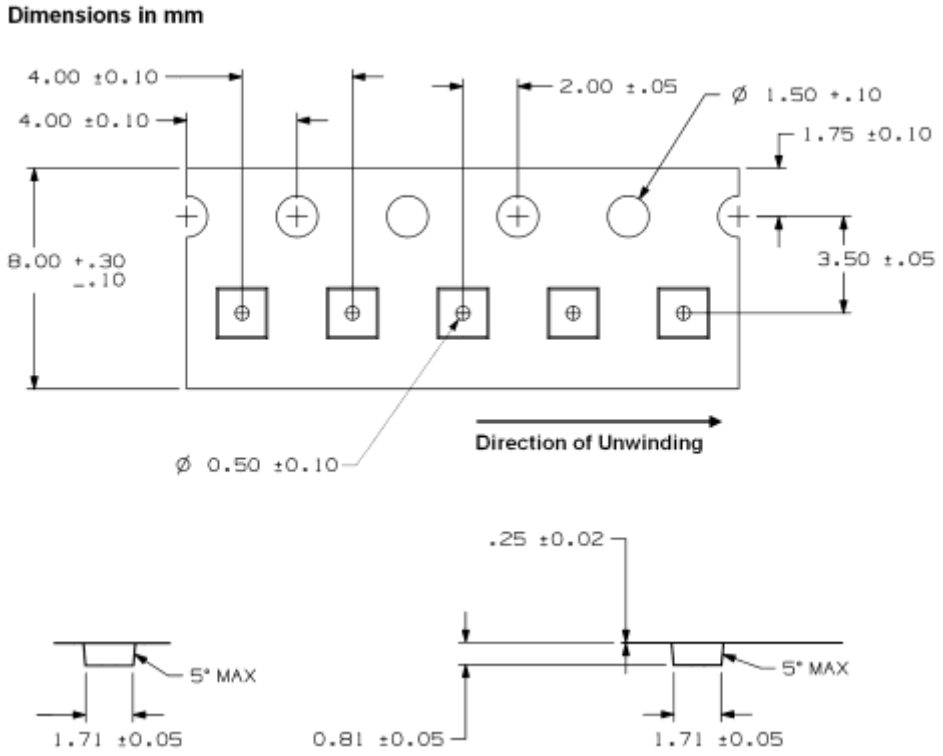


Figure 21: Tape Dimensions

Copyright & Warranty Info

Revised June 2016. For most recent version visit our website at http://www.pnicorp.com/

PNI Sensor Corporation
2331 Circadian Way
Santa Rosa, CA 95407, USA
Tel: +1 (707) 566-2260
Fax: +1 (707) 566-2261

Warranty and Limitation of Liability. PNI warrants that each PNI Product to be delivered hereunder, if properly used, will be free from defects in material and workmanship and will operate in accordance with PNI's published specifications and documentation for the Product in effect at time of order acceptance, if operated as directed by them for ninety (90) days following the date of shipment unless a different warranty time period for such Product is specified: (i) in PNI's Price List in effect at time of order acceptance; or (ii) on PNI's web site (www.pnicorp.com) at time of order acceptance. This warranty will be void if the products, serial number or other identification marks have been defaced, damaged or removed. This warranty does not cover wear and tear due to normal use, or damage to the product as the result of improper usage, neglect of care, alteration, accident or unauthorized repair.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. PNI NEITHER ASSUMES NOR AUTHORIZES ANY PERSON TO ASSUME FOR IT ANY OTHER LIABILITY.

If any PNI Product furnished hereunder fails to conform to the above warranty, Customer's sole and exclusive remedy and PNI's sole and exclusive liability will be, at PNI's option, to repair, replace or credit Customer's account with an amount equal to the price paid for any such Product which fails during the applicable warranty period provided that: (i) Customer promptly notifies PNI in writing that such Product is defective and furnishes an explanation of the deficiency; (ii) such Product is returned to PNI's service facility at Customer's risk and expense; and (iii) PNI is satisfied that claimed deficiencies exist and were not caused by accident, misuse, neglect, alteration, repair, improper installation or improper testing. The warranty excludes all cost of shipping, customs clearance and other related charges. PNI will have a reasonable time to make repairs or to replace Product or to credit Customer's account. PNI warrants any such repaired or replacement product to be free from defects in material and workmanship on the same terms as the product originally purchased.

Revision Control Block

Revision	Description of Change	Effective Date	Approval
DRAFT	Convert from EM to PNI Format	07-Dec-2015	D. Mckenzie
DRAFT	Minor edits and Sentral A2 name change	03-June-2016	E.Wang
DRAFT	More minor edits and formatting	29-June-2016	D. Vincent