IC4 Interface Control Document (ICD)

Revision history:

Revision	Date	Comments
1.5	2014-05-13	Initial Release

Table of Contents

Figures

Tables

1 Introduction

This document defines the protocol interface of the IC4 with the host system. Electrical and mechanical interfaces are described separately in the IC4 datasheet, which should be used in conjunction with this ICD. This IC4 interface control document fully describes the packet structure and all commands, responses and output data packet formats that are applicable to the IC4.

The IC4 commands and responses are listed in **Table 8** in Section 2.3. For basic operation, one needs only to apply power, and then send the Start Streaming command, which will cause the IC4 to start streaming out the default data at the default data rate and baud rates. Optionally, before entering streaming mode, the Set Register command can be used to configure data packet items, data rate, baud rate, etc. The output data packet documented in Section 2.5 is a user-configurable format.

1.1 LED Behavior

The IC4 board contains one LED to indicate the voltage level and a second LED on Bluetooth models to indicate the Bluetooth status. Wired IC4s do not currently display LEDs as the board is contained inside a metal or plastic housing. Note that Bluetooth models only display the LEDs when the power switch is in the ON position, and that it is recommended to charge the sensors with the power turned off, as it will charge more quickly that way (an additional green LED near the charge connector indicates that the sensor is charging when it is illuminated).

Table 1: Power LED status

LED Status	Charging?	Power LED description
Solid orange	Yes	Battery charging, leave power cable plugged in
Solid green	Yes	Battery fully charged, power cable can be unplugged
Solid green	No	Battery voltage stable for normal use
Solid red	No	Power level low, needs recharging
Blink red	No	Power level very low, needs immediate recharging

Table 2: Bluetooth LED Status

LED Status	Bluetooth LED description
Solid blue	Bluetooth connection established successfully
Fast blink blue	Bluetooth command mode (times out)
Slow blink blue	Bluetooth not connected and not in command mode

1.2 Interfaces

IC4 supports a UART interface via RS232 or USB. The USB interface is provided by a Silicon Labs chipset, and requires drivers to communicate with the host OS (this is currently provided by Silicon Labs for Windows, Mac OS X, Windows CE, and in the Linux kernel, though usage on Android and possibly other Linux variants requires a rebuild of the Linux kernel to include the driver).

1.2.1 RS-232/USB Communication

The IC4 RS-232 UART external communication interface is a full-duplex serial communication port. The default baud rate is 115,200 bps with 1 start bit, 1 stop bit, and no parity. The UART receives commands on the RX pin, and transmits outgoing messages on the TX pin, including both command replies and streaming data packets.

When using the USB UART interface (which requires a Silicon Labs Virtual COM Port driver), it is still necessary to set the USB Virtual COM Port parameters to match the IC4 baud rate (as set by the baud rate register), to match.

1.3 Modes

The IC4 implements a partitioned firmware for operation in Bootloader Mode and Operating Mode. illustrates the different modes and possible transition paths between them.

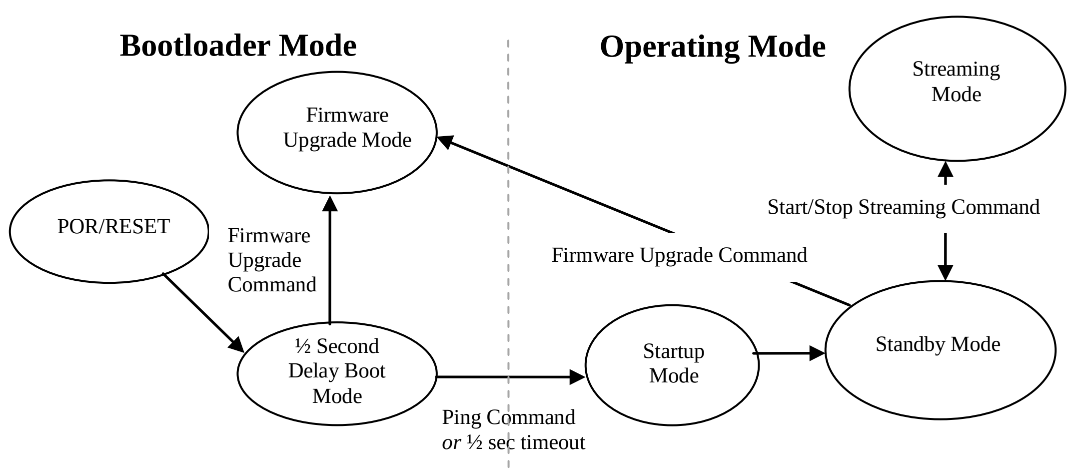

Figure 1: IC4 mode transition diagram

1.3.1 Bootloader Mode

The IC4 firmware is in-system programmable. On power-up the IC4 executes a ½ second Delay Boot Mode where it is waiting for either a Ping command to exit Bootloader Mode and jump to Operating Mode or a Firmware Upgrade command to jump to Firmware Upgrade Mode. Firmware upgrade can be performed by customers using an InterSense-provided upgrade utility (currently this is the *DeviceTool2* application, which is Windows-only).

1.3.2 Operating Mode

Once the IC4 exits the ½ second Delay Boot Mode it enters Startup Mode where it initializes the hardware and performs quick built-in-tests (QBIT) on the sensors. IC4 has two other modes of operation, Standby Mode and Streaming Modes. In Streaming Mode, IC4 actively acquires data from the sensors and transmits data. In Standby Mode, the IC4 performs the same acquisition and processing in order to stabilize the temperature while it waits for external commands, but does not stream output data. Refer to **Table 3** for typical start up sequence and timing.

Table 3: IC4 start-up sequence timing

Time (ms)	IC4 Status	Comment
0	Power applied	½ second delay boot starts
T_1 (max 500)	Switch from bootloader to Startup Mode	T_1=500 ms, or earlier if Ping command received.
T_1+200	Switch from Startup Mode to Standby Mode	Hardware initialized, QBIT complete, ready to stream valid data
$T_2 > T_1 + 200$	Enter Streaming Mode	By default, T_2 is the time when Start Streaming command is received. If auto-streaming parameter has been configured and saved, the device will automatically switch to streaming mode upon completion of startup mode, and T_2=T_1+200.

2 Communications protocol

2.1 Packet structure

Commands begin with a start byte and a header byte, formatted as shown in **Table 4**. Some commands include a body, typically containing additional command parameters and data. A checksum is added to the end of the command. The checksum is the two's complement of the sum of all preceding bytes including the start byte and header. Little-endian format is used for multi-byte words for communication and addressing within the IC4. All signed integers use the 2's complement format.

2.1.1 Command packet structure

The *start byte* is always 0xA5. The *header* byte consists of an *address* nibble and a *command* nibble.

Address specifies the recipient device. Default address is zero. In some designs, the IC4 can share the interface with other devices at different addresses.

Table 4: Command packet structure

<table>
<tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td colspan="4">0xA</td><td colspan="4">0x5</td></tr>
<tr><td>Spare</td><td colspan="3">Address (0-7)</td><td colspan="4">Command (0-15)</td></tr>
<tr><td colspan="8">Body (command-dependent)</td></tr>
<tr><td colspan="8">Checksum</td></tr>
</table>

The commands supported by the IC4 are listed in **Table 8**.

2.1.2 Reply packet structure

Replies start with an echo of the command header byte and may be followed by additional data bytes depending on the command. Replies that include a body (typically containing data) also have a checksum appended, which includes all preceding bytes.

Table 5: Reply packet structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Command header echo							
Reply data byte(s)*							
Checksum*							
Note: *Present only when reply includes a body.							

2.2 Data and baud rates

The IC4 can transmit data at any of the following data rates:

Table 6: IC4 data output rate

data rate = data rate max/divisor	
data rate divisor	**output data rate**
1	1000 Hz (Max)
2	500 Hz
3	333.333 Hz
4	250 Hz
5	200 Hz (default)
6	166.667 Hz
7	142.857 Hz
8	125 Hz
9	111.111 Hz
10	100 Hz
…	…
…	…
32	31.25 Hz (Min)

For packets containing **$\Delta\theta$** and **ΔV** data, the minimum data rate is 100 Hz to avoid overflow.

Communication can occur at a maximum baud rate of 921,600 or any of the following submultiples listed in **Table 7**. Other divisors will not be rejected, but only the divisors listed in **Table 7** are officially supported.

Table 7: IC4 baud rates

baud rate = baud rate max/divisor	
baud rate divisor	**output baud rate**
1	921,600 (Max)
2	460,800
4	230,400
8	115,200 (default)
24	38,400

Be aware when programming the communications parameters that the baud rate must be high enough to support the chosen data rate and packet type combination. For a packet type with total length N bytes, the length will be 10*N bits (including the start and stop bit), so the baud rate should be at least 20% higher than 10*N*data rate. If not, the IC4 will drop data packets whenever the serial port transmission cannot keep up with the rate at which new data is being generated. Because data is output only on integer millisecond intervals, this will have the effect of reducing the effective data output rate to the highest rate that is possible to transmit at a given baud rate, however operation in this mode is not recommended.

2.3 IC4 command set

Refer to the key below for notation frequently used for command syntax.

Key:

`<s>`	Start byte *s* = 0xA5
`<ax>`	Header byte where *a* = device address, *x* = command
`<d>`	Data byte
`<ct>`	Total number of bytes in a command, including header and checksum
`<cs>`	Checksum byte (two's complement of the sum of all preceding bytes)

Table 8: IC4 command set

Command	Command Syntax	Reply Syntax
Ping/Stop Streaming	`<s><a0><cs>`	`<a0>` if not streaming, else stops streaming but no reply.
Get Register[1,2]	`<s><a1><ma><cs>` Refer to **Table 9** for meaning of the bytes of the Configuration Registers. $0 \leq ma \leq 255$	`<a1><d><cs>`
Set Register[1,2]	`<s><a2><ma><d><cs>` Refer to **Table 9** for meaning of the bytes of the Configuration Registers. $0 \leq ma \leq 255$ Set the address *ma* to 0xFF and *d* to 0 to save the updated data to FLASH memory. Set the address *ma* to 0xFF and *d* to 1 to restore the registers to default values.	`<a2>`
Start Streaming	`<s><a5><cs>`	Starts streaming data packets. No command acknowledgement. If Keep-Alive is set, this command must be sent within the Keep-Alive time period to keep data streaming. If Keep-Alive expires, it is set to zero and must be reset to use Keep-Alive.

Notes:

1. These commands read/write to RAM, and updates received in the commands are updated only in the RAM copy of the page until saved. Care should be taken to perform all the updates to the RAM copy and then save the changes to flash, before doing any other operation, or all changes will be lost.
2. Not available in Streaming Mode, and will be ignored if accidentally sent.

2.4 Configuration Register Set

The Configuration Register Set is a 256-byte block of memory containing all of the IC4's current operating state information, including constants, user-configured parameters, and built-in self-test results. It is important to note that some registers may be different for firmware version 4 and lower; any sensors with older firmware that need to use this ICD must upgrade to version 5 or later in order to use the ICD (this may be obtained free of charge from InterSense by contacting Technical Support, techsupport@intersense.com). There are three ways to read information out of the Configuration Registers:

1) Get Register command reads out any byte(s) in any order.
2) In Streaming Mode, the Discrete Flag Byte contains an "S-bit" which cycles through the first 256 bits (32 bytes) of the Configuration Register Set, allowing the receiving program to reconstruct the most important status information once per 256-packet frame (e.g. 1.28 seconds at 200 Hz, more rapidly at higher rates).
3) In Streaming Mode, the output packet can be configured to play back a byte at a time, allowing the receiving program to reconstruct the entire Configuration Register Set once per 256-packet frame (the register sent is the register corresponding to the packet ID number).

Table 9: IC4 Configuration Registers description

Byte Address	Access Permission	Bit Fields	Function	
0	Read		Device type (23 for IC4)	Basic Device Info Registers
1	Read		Firmware minor version[0]	
2	Read		Firmware major version[0]	
3	Read		NVRAM size in 64-byte blocks	
4-6	Read		Serial number portion A[1]	
7	Read		Reserved	
8	Read / Write		Device address (0-7, default 0)[2]	
9	Read		Reserved	
10-11	Read		FrameID (Increments when the ***PacketID*** rolls over)	Operating State Registers
12-13	Read		Serial number portion B[1]	
14	Read / Write		Baud rate divisor (see **Table 7**)	
15	Read / Write		Data rate divisor (see **Table 6**)	
16	Read		Reserved	

17	Read / Write	7	Reserved	
		6	Reserved	
		5	Reserved	
		4	Reserved	
		3	Reserved	
		2	Reserved	
		1	Reserved	
		0	Stream on power-up[3] (default=0)	
18	Read	7	Reserved	
		6	Reserved	
		5	Reserved	
		4	Reserved	
		3	Reserved	
		2	Streaming now	
		1:0	Reserved	
19-21	Read		Reserved	
22-24	Read		Reserved	Health Registers
25-27	Read	23:12	Temperature (signed integer representing Celsius temperature in 0.05°C increments)	
		11:0	Vin (unsigned integer representing volts in 1.4648mV increments)	
28-31	Read		Reserved	
32-35	Read / Write		Data Item List for output packet (**Table 12**)	Extended Registers
36-85	Read		Reserved	
86-88	Read		Calibration Date[4]	
89	Read		Discrete flag byte (**Table 10**)	
90-91	Read		Calibration Revision	
92-158	Read		Reserved	
159	Read / Write		Keep-Alive timeout (in increments of 0.1 seconds). Not saved, cleared if Keep-Alive times out.	
160-161	Read		Reserved	

162	Read / Write	7	Reserved	
		6	Reserved	
		5	Reserved	
		4	Reserved	
		3	Reserved	
		2	Reserved	
		1	Reserved	
		0	AHRS Compass (On/Off)	
163-254	Read		Reserved	
255	Read/Write (self-clears)		Save/Restore configuration register set[5]	Save Registers[6]

Notes:

[0] The version is registers 1:2 as a little-endian short (MINOR + MAJOR*256)

[1] The Serial Number is a 7 digit decimal number in the format YYMMSSS. Left shift ***portion A*** (registers 4-6, interpreted as little endian format) by 8 bits and add ***portion B*** (registers 12-13) to them to get the serial number. For example, if registers 4-6 are [0x61, 0x15, 0x00] and registers 12-13 are [0x92, 0x00], the serial number would be (0x001561 << 8) + 0x0092= 0x156192 (or 1401234 decimal). The decimal number, **1401234**, will match the numeric portion of the serial number label on the IC4.

[2] Changing the device address may prevent detection by the InterSense library (DLL)

[3] Boot to streaming mode instead of standby mode.

[4] The Calibration Date is saved as a decimal number in the format MM/DD/**2**YYY, and is encoded with register 86 being MM, register 87 being DD, and register 88 being YYY (all interpreted as decimal numbers with range 0-255).

[5] This byte is self-clearing on the execution of the command.

[6] Value 0: Save CRS to flash, 1: Restore CRS to default

2.5 AHRS Data

Firmware versions 5 and later support on-board processing of sensor data to provide AHRS data (Attitude and Heading Reference System), using a Kalman filter very similar to the one used in the InterSense library. This allows embedded systems to obtain the orientation of the sensor (as a rotation matrix, Euler angles, or quaternions), with optional magnetic heading correction. This feature is automatically enabled when data items representing Euler angles, quaternions or any of the rotation matrix rows (bits 10-14 of the register 35-32 bit field) are enabled.

Please note that this feature is limited to data rates of 200 Hz and below, and will cause significantly more current draw due to the extra computation required to process the data and provide AHRS outputs. Therefore, it is recommended not to enable these bits unless they are being used, to reduce power consumption. Although all items may be output simultaneously, additional items will require additional computation, so it is recommended to only output those items that are required (e.g. only Euler angles or only quaternions). Additionally, the full data resolution is internally available to the Kalman filter, so the AHRS data is not limited by maximum accel/gyro rates at low data rates.

InterSense library versions 4.2393 and later may also be configured to pass the AHRS output data through to the library for quick tests using existing software such as ISDEMO. To do this, it is possible to use the InterSense API, or to create a file called **isense1.ini** in the application directory, which contains the following text (case sensitive):

```
#STATION
GetAHRSData=1
```

If the file above is used with older sensors, no data will be displayed; if used with an older DLL, AHRS will not be enabled on the sensor. The **Compass** setting in the API/applications is also honored by the DLL, and may be changed while the sensor is running.

Approximate current draws for Euler angle output at various data rates are presented below in Table 10 for reference. Disabling the compass increases current draw by around 3mA (due to increased Kalman filter update rates from the IMU).

Table 10: Approximate AHRS current draw

AHRS Current Draw @ 6V DC for IC4 FW 5		
Data Rate	*AHRS Disabled*	*Euler angles with Compass*
200 Hz	37 mA	57 mA
125 Hz	36 mA	51 mA
100 Hz	35 mA	49 mA
50 Hz	34 mA	44 mA

2.6 Data Packets Description

The IC4 supports a user-configurable data output packet type for different applications. Configuring the packet through the Data Item List register causes the IC4 to provide the selected type of output data.

2.6.1 Glossary

PacketID is an 8-bit sequence counter of the packet number relative to the start of this 256-record data frame. This can be used to decode the Configuration Registers one bit or one byte at a time during streaming mode (see the ***S*** flag description in **Table 10**).

DeltaTheta is an incremental rotation vector over the data integration period, including coning compensation if enabled. Angular integrals are maintained internally with higher precision, and any truncated remainders are carried forward and added to the next output packet, so that numerical round-off will not affect the long-term integration accuracy.
DeltaTheta (16-bit signed integer): Each bit represents 0.00625 mRad, with ±32768 bits range, so the maximum rotation per update period is 0.2048 Rad. In a high-dynamic application with angular rates up to 20 Rad/s, the update rate must be at least 100 Hz to prevent overflow.

DeltaV is an integral of accelerometer measurements over the data integration period, including coning and sculling compensation if enabled. Velocity integrals are maintained internally with higher precision, and any truncated remainders are carried forward and added to the next output packet, so that numerical round-off will not affect the long-term integration accuracy.
DeltaV (16-bit signed integer): Each bit represents 39.0625e-6 m/s, with ±32768 bits range, so the maximum velocity change per update period is 1.28 m/s. In a high dynamic application with linear accelerations up to the maximum 120 m/s/s that the IC4 can measure, the update rate must be at least 100 Hz to prevent overflow.

$\boldsymbol{Mag_I}$ is the compensated magnetic measurement in Gauss for the I^{th} axis. The value of I sequences to select x(i=1),y(i=2) and z(i=3) axis as new data is being acquired from the magnetometer sensor. When no new data is available, or in products that do not contain magnetometers $I = 0$ and $\boldsymbol{Mag_I} = 0$.
Mag (16-bit signed integer): Each bit represents 0.25e-3 Gauss with ±32768 bits range, so the maximum measurement range is +/-8.192 Gauss. The index I specifies the current axis which is contained in byte 4 of the packet.

Euler angles Roll (φ), Pitch (θ) and Yaw (ψ) (16-bit signed integer): Each bit represents 0.1e-3 Rad, with ±32768 bits range.

Quaternion q, qi, qj and qk (16-bit signed integer): Each bit represents 3.05185095e-5 (1/32767), with a nominal range from -1 to +1.

Rotation matrix rows (16-bit signed integer): Each bit represents 3.05185095e-5 (1/32767), with a nominal range from -1 to +1. Transforms vectors from navigation frame to body frame.

Discrete flag byte

The following table defines certain bits or bit-fields that may appear in certain Data Packets.

Table 11: Discrete flag byte description

Item Name	Bits	Description
E1	7	The *E1* and *E2* flags can be used to notify events (reserved).
E2	6	
D	5	Reserved.
S	4	The *S* flag represents the value of the n[th] bit of the first 32 bytes (256 bits) of the Configuration Registers, where n is determined by PacketID. In other words, the Configuration Registers get played out once per 256-record frame, reading one bit at a time from left to right. The receiving software can reconstruct the Configuration Registers after every 256 records (e.g. once every 1.28 seconds at the default 200Hz data rate) if there are no data dropouts during that time.
F	3	The *F* flag signals a fault condition. The F flag will be high if there are any runtime warning flags.
R	2	The *R* flag is reserved for future use.
I	1:0	Index of $\boldsymbol{Mag_{I}}$.

2.6.2 Data Output Packet: User-Configurable Packet

Table 12: IC4 Data Output Packet

Byte No.	contents
0	Start byte (0x*a*5, a = device address)
1	Data Packet Type (0x64)
2	PacketID bits 7-0
3-?	data item 1
.	data item 2
.	…
N	Checksum of bytes 0 to N-1
Notes: Refer to **Table 12** for the definition of data fields.	

2.6.3 Data item definitions for Data Output Packet

The contents of the customizable Data Output Packet are defined by setting or clearing certain bits in the 32-bit "Data Item List" bit field located in bytes 32-35 of the Configuration Register Set.

Table 13: Data Output Packet data item definitions

Register	Bit[1]	Data Item Name	Data Item Type	Item Size (bytes)
35	31:24	Reserved	-	-
34	23:16	Reserved	-	-
33	15	Reserved	-	-
	14	Rotation matrix Cnb row 3 ordered Czx Czy Czz	(3x) 16-bit signed integer	6
	13	Rotation matrix Cnb row 2 ordered Cyx Cyy Cyz	(3x) 16-bit signed integer	6
	12	Rotation matrix Cnb row 1 ordered Cxx Cxy Cxz	(3x) 16-bit signed integer	6
	11	Quaternion ordered q, qi, qj and qk	(4x) 16-bit signed integer	8
	10	Euler angles ordered Roll(**φ**), Pitch(**θ**) and Yaw(**ψ**)	(3x) 16-bit signed integer	6
	9	Reserved	(4x) 8-bit unsigned char	4
	8	Temperature (0.05°C/LSB)	16-bit signed integer	2
32	7	Vin (1.4648mV/LSB)	16-bit unsigned integer	2
	6	Vex (23.4375mV/LSB)	8-bit unsigned integer	1
	5	Configuration register n (0-255)[2]	8-bit unsigned char	1
	4	Mag_I	16-bit signed integer	2
	3	$DeltaTheta_x$, $DeltaTheta_y$, $DeltaTheta_z$,	(3x) 16-bit signed integer	6
	2	$DeltaV_x$, $DeltaV_y$, $DeltaV_z$	(3x) 16-bit signed integer	6
	1	Reserved	16-bit unsigned integer	2
	0	E1	E2	D	S	F	R	I (2 bits)	discrete flag byte	1

Notes:

[1] Bit number in bytes 32-35 of the Configuration Register Set. Items are transmitted in order of increasing bit #, so it is possible to change the items sent, but not the item ordering.

[2] n^{th} byte of the Configuration Registers, where *n* is the packet's PacketID (range 0-255)

Bits 0,1,2,3,4 (of register 32) are enabled when the registers are restored to defaults