IMU380ZA-200 USER MANUAL

Document Part Number: 7430-3810-01

MEMSIC, Inc., 1759 McCarthy Blvd, Milpitas, CA 95035
Tel: 408-964-9700, Fax: 408-854-7702
email: infoca@memsic.com, website: www.memsic.com

⚠ WARNING

This product has been developed by MEMSIC exclusively for commercial applications. It has not been tested for, and MEMSIC makes no representation or warranty as to conformance with, any military specifications or that the product is appropriate for any military application or end-use. Additionally, any use of this product for nuclear, chemical, biological weapons, or weapons research, or for any use in missiles, rockets, and/or UAV's of 300km or greater range, or any other activity prohibited by the Export Administration Regulations, is expressly prohibited without the written consent of MEMSIC and without obtaining appropriate US export license(s) when required by US law. Diversion contrary to U.S. law is prohibited.

Table of Contents

About this Manual

The following annotations have been used to provide additional information.

NOTE

Note provides additional information about the topic.

☑ EXAMPLE

Examples are given throughout the manual to help the reader understand the terminology.

IMPORTANT

This symbol defines items that have significant meaning to the user

⚠ WARNING

The user should pay particular attention to this symbol. It means there is a chance that physical harm could happen to either the person or the equipment.

The following paragraph heading formatting is used in this manual:

1 Heading 1

1.1 Heading 2

1.1.1 Heading 3

Normal

1 Introduction

1.1 Manual Overview

This manual provides a comprehensive introduction to the MEMSIC IMU380ZA-200 product. The following table highlights the content in each section and suggests how to use this manual.

Table 1: Manual Content

Manual Section	**Who Should Read?**
Section 1: Introduction	All customers should read section 1.
Section 2: Interface	Customers designing the electrical and mechanical interface to the IMU380ZA-200 should read Section 2.
Section 3: Theory of Operation	All customers should read Section 3. Section 3 provides an overview of all of the functions and features of the IMU380ZA-200.
Section 4: SPI Port Interface	Customers designing the software interface to the IMU380ZA-200 SPI Port should review Section 4.
Sections 5-7: UART Port Interface	Customers designing the software interface to the IMU380ZA-200 UART Port should review Sections 5-7.

1.2 Overview of the IMU380ZA-200 Inertial System

This manual provides a comprehensive introduction to the MEMSIC IMU380ZA-200, and is intended to be used as a detailed technical reference and operating guide. MEMSIC's IMU380 is a 6DOF Digital IMU that combines the latest in commercial MEMS (Micro-electromechanical Systems) sensors and digital signal processing techniques to provide a small, cost-effective alternative to existing IMU Systems.

The IMU380ZA-200 is designed for OEM applications. At the core of the IMU380 is a rugged 6-DOF (Degrees of Freedom) MEMS inertial sensor board. The 6-DOF MEMS inertial sensor board includes three axes of MEMS angular rate sensing and three axes of MEMS linear acceleration sensing. These sensors are based on rugged, field proven silicon bulk micromachining technology. Each sensor within the board is individually factory calibrated for temperature and non-linearity effects during MEMSIC's manufacturing and test process using automated thermal chambers and rate tables. Coupled with the 6-DOF MEMS inertial sensor board is a high performance processor (CPU) that utilizes the inertial sensor measurements to accurately compute angular rate and linear acceleration.

The IMU380ZA-200 is packaged in a light-weight, rugged, unsealed metal enclosure that is designed for cost-sensitive commercial and OEM applications. The IMU380 can be configured to output data over a SPI Port or a low level UART serial port. The port choice is user controlled by grounding the appropriate pin on the connector.

The IMU380ZA-200 low level UART output data port is supported by MEMSIC's NAV-VIEW 3.X, a powerful PC-based operating tool that provides complete field configuration, diagnostics, charting of sensor performance, and data logging with playback.

2 Interface

2.1 Electrical Interface

2.1.1 Connector and Mating Connector

The IMU380ZA-200 main connector is a SAMTEC FTM-110-02-F-DV-P defined in Figure 1. The mating connector that can be used is the SAMTEC CLM-110-02.

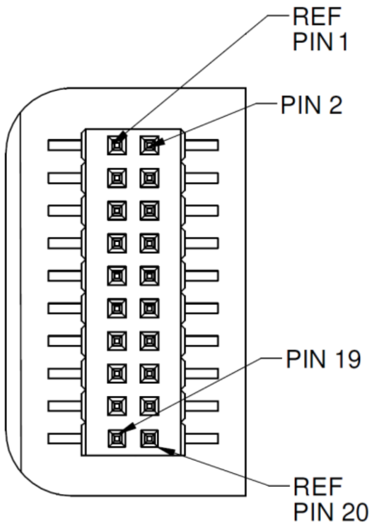


Figure 1: IMU380ZA-200 Interface Connector

The connector pin definitions are defined in Table 2.

Table 2: IMU380ZA-200 Interface Connector Pin Definition

Pin	Signal
1	Reserved – factory use only
2	Synchronization Input (1KHz Pulse used to synchronous SPI Com)
3	SPI Clock (SCLK) / UART TX
4	SPI Data Output (MISO) / UART RX
5	SPI Data Input (MOSI)
6	SPI Chip Select (SS)
7	Data Ready (SPI Communication Data Ready) / SPI-UART Port Select
8	Reserved – factory use only
9	Reserved – factory use only
10	Power VIN (3-5 VDC)
11	Power VIN (3-5 VDC)
12	Power VIN (3-5 VDC)
13	Power GND
14	Power GND
15	Power GND
16	Reserved – factory use only
17	Reserved – factory use only
18	Reserved – factory use only
19	Reserved – factory use only
20	Reserved – factory use only

2.1.2 Power Input and Power Input Ground

Power is applied to the IMU380ZA-200 on pins 10 through 15. Pins 13-15 are ground; Pins 10-12 accepts 3 to 5 VDC unregulated input. Note that these are redundant power ground input pairs.

⚠ WARNING

Do not reverse the power leads or damage may occur.

2.1.3 Serial Data Interface

The user can select the serial interface port for the IMU380ZA-200. If the connector pin 7 is left floating then the IMU380ZA-200 is configured for SPI communications on pins 3-6. Pin 7 is then used as the DATA READY discrete for SPI communications. The user can also synchronize the output data on the SPI port by providing a 1 KHz input pulse on Pin 2. For the complete SPI interface definition, please refer to Section 4. If the connector pin 7 is grounded then the IMU380ZA-200 is configured for low-level UART output on pins 3 and 4. This is a standard UART asynchronous output data port. For the complete UART interface definition, please refer to Sections 5-7. Note that the two output port interface methods are controlled independently from each other. The UART port output controls apply only to data being output over the UART port, and the SPI output controls apply only to data being output over the SPI port.

2.1.4 Reserved – Factory Use Only

During normal operation of the IMU380ZA-200, no connection is made to the Reserved – factory use only pins. These pins have internal pull-up mechanisms and must have no connections for the IMU380ZA-200 to operate properly.

2.2 Mechanical Interface

The IMU380ZA-200 mechanical interface is defined by the outline drawing in Figure 2.

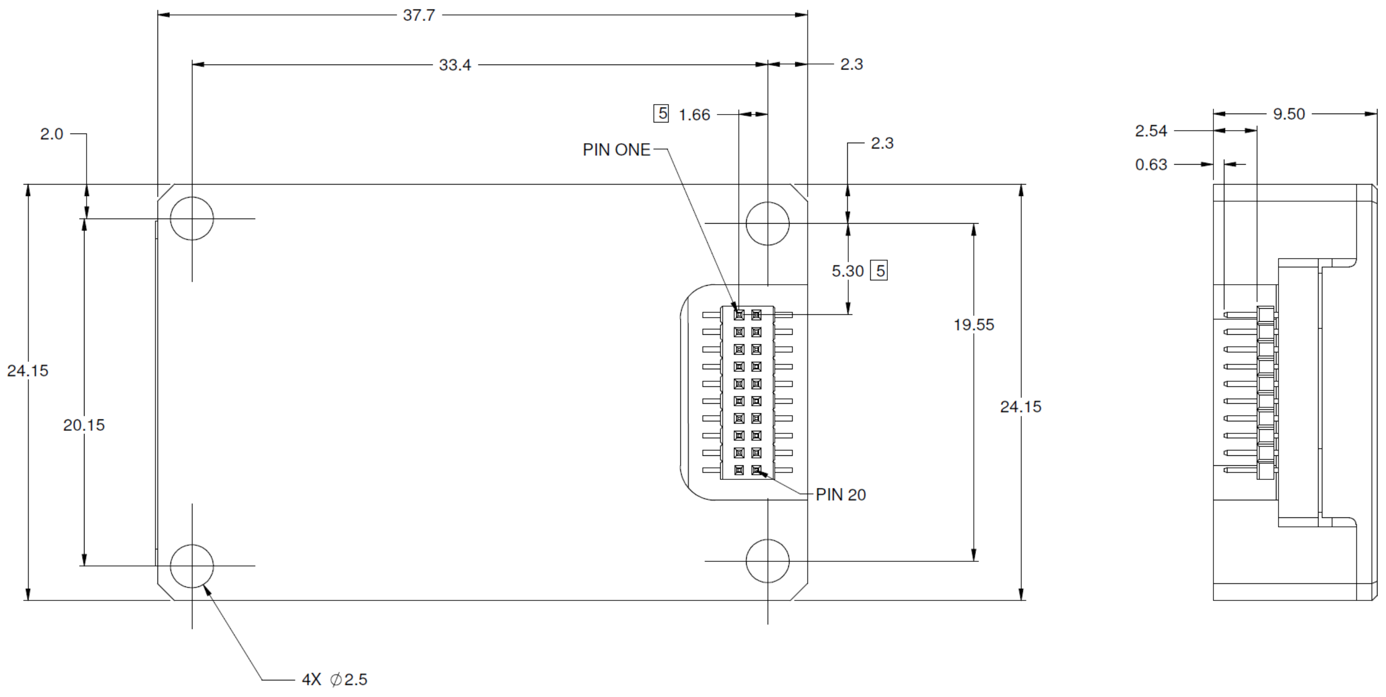


Figure 2: IMU380ZA-200 Outline Drawing

NOTES UNLESS OTHERWISE STATED:

1) INTERPRET DWG. PER ANSI Y14.5M-1994
2) CONTROLLING DIMENSION: MM
3) ALL DIMENSIONS ARE BASIC
4) MATING CONNECTOR: SAMTEC CLM-110-02
5) DIMENSION TO CENTROID OF PIN ONE $\boxed{5}$

3 Theory of Operation

This section of the manual covers detailed theory of operation for the IMU380ZA-200 with its associated features, output, and performance.

Figure 3 shows the IMU380ZA-200 hardware block diagram. At the core of the IMU380ZA-200 is a 6-DOF (Degrees of Freedom) MEMS inertial sensor cluster mounted on a rigid board. The 6-DOF MEMS inertial sensor cluster includes three axes of MEMS angular rate sensing and three axes of MEMS linear acceleration sensing. These sensors are based on rugged, field proven silicon bulk micromachining technology. Each sensor within the cluster is individually factory calibrated using MEMSIC's automated manufacturing process. Sensor errors are compensated for temperature bias, scale factor, non-linearity and misalignment effects using a proprietary algorithm from data collected during manufacturing. Accelerometer and angular rate sensor bias shifts over temperature (-40 ^{0}C to +85 ^{0}C) are compensated and verified using calibrated thermal chambers and rate tables. The 6-DOF sensor cluster data is fed into a high-speed data acquisition processor, which provides all the sensor compensation and communications handling.

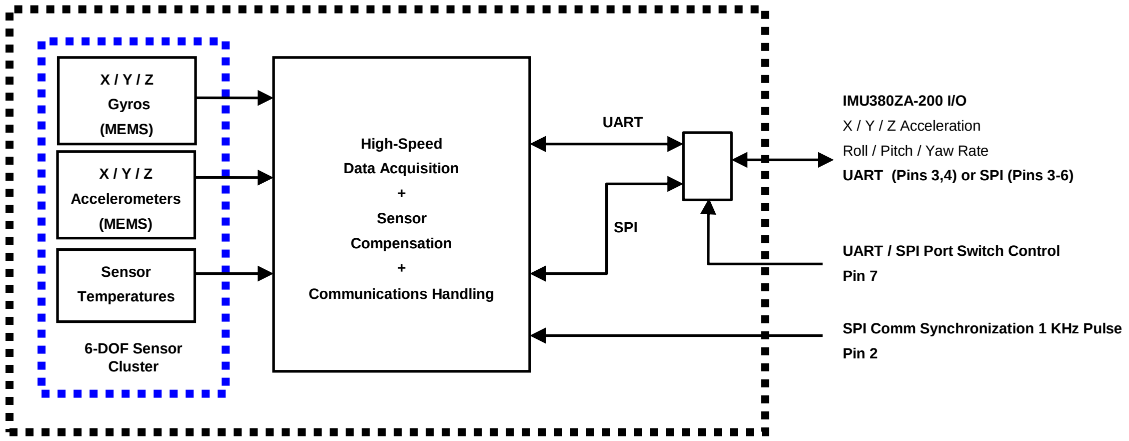


Figure 3: IMU380ZA-200 Hardware Block Diagram

The 6-DOF inertial sensor cluster data is fed into a high speed signal processing chain, which provides sensor compensation and digital filtering. Measurement data packets are available at fixed continuous output rates or on a polled basis from the SPI port or the UART port. The SPI port outputs data via registers, and the user can perform polled reads of each register, or a block burst read of a set of predefined registers. Output data over the SPI port can be synchronized to an external 1 KHz pulse. The complete SPI interface is defined in Section 4. The UART port outputs data packets are asynchronous and defined in Sections 5-7.

3.1 IMU380ZA-200 Default Coordinate System

The IMU380ZA-200 Inertial System has two default coordinate systems depending on the output data port selected. Figure 4 defines the fixed coordinate system for data output via the SPI Port.

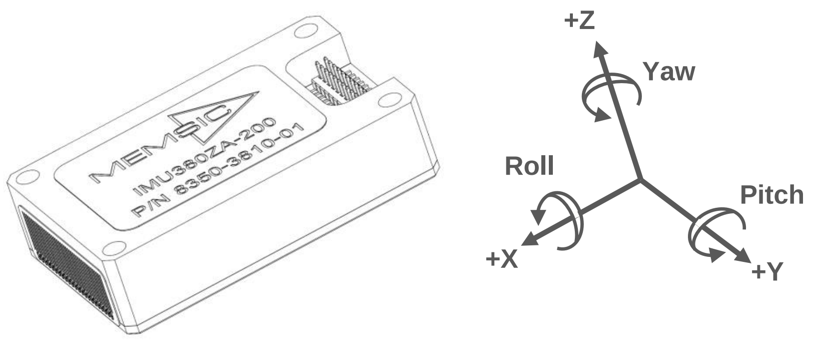


Figure 4: IMU380ZA-200 SPI Port Coordinate System

Figure 5 defines the default coordinate system for data output via the UART port. Note that for data output via the UART port, the coordinate system is configurable with either NAV-VIEW 3.X (see Appendix A) or by sending the appropriate serial commands.

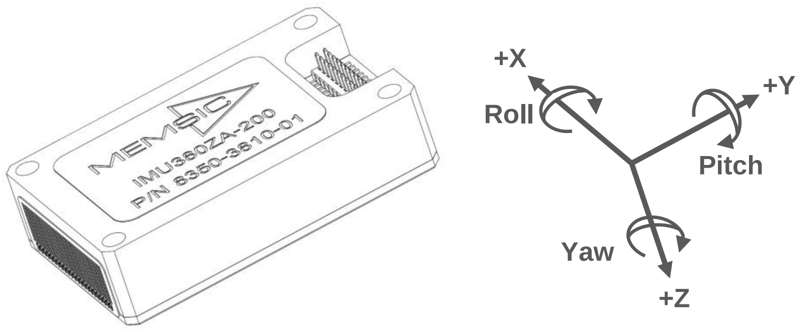


Figure 5: IMU380ZA-200 UART Port Default Coordinate System

For both output ports, the axes form an orthogonal SAE right-handed coordinate system. Acceleration is positive when it is oriented towards the positive side of the coordinate axis.

For example, if the IMU380ZA-200 is configured to output data over the UART port, and the default UART port coordinate frame is used, with an IMU380ZA-200 sitting on a level table, the system will measure zero g along the x- and y-axes and -1 g along the z-axis. Normal Force acceleration is directed upward, and thus will be defined as negative for the IMU380ZA-200 z-axis.

The angular rate sensors are aligned with these same axes. The rate sensors measure angular rotation rate around a given axis. The rate measurements are labeled by the appropriate axis. The direction of a positive rotation is defined by the right-hand rule. With the thumb of your right hand pointing along the axis in a positive direction, your fingers curl around in the positive rotation direction.

For example, if the IMU380ZA-200 is configured to output data over the UART port, and the default UART port coordinate frame is used, with the IMU380ZA-200 sitting on a level surface and rotated clockwise on that surface, the system will measure a positive rotation around the z-axis. The x- and y-axis rate sensors will measure zero angular rates, and the z-axis sensor will measure a positive angular rate.

Pitch is defined positive for a positive rotation around the y-axis (pitch up). Roll is defined as positive for a positive rotation around the x-axis (roll right). Yaw is defined as positive for a positive rotation around the z-axis (turn right).

3.2 IMU380 Theory of Operation

The product name, IMU380, stands for Inertial Measurement Unit 380, and the name is indicative of the inertial measurement unit functionality that the IMU380 provides by providing inertial rate and acceleration data in 6-DOF (six degrees of freedom). The IMU380 signal processing chain consists of the 6-DOF sensor cluster and the high speed data processor for sensor error compensation and low-pass digital filtering, as well as communications handling. The IMU380 can output data over a synchronous SPI port, or an asynchronous low-level UART serial communications port.

The rate and acceleration sensor signals are acquired at greater than 500Hz. The sensor data is filtered and down-sampled to 200Hz by the processor using FIR or IIR filters (user selectable – see Section 4 for the filter characterization) for the SPI port, and down sampled to 100Hz by the processor using an IIR (2nd order 10Hz Butterworth) filter for the UART port. The factory calibration data, stored in EEPROM, is used by the processor to remove temperature bias, misalignment, scale factor errors and non-linearity from the sensor data.

For data output over the SPI port, the IMU380ZA-200 provides a register read/write capability. The user may define single register reads, or read from a defined burst set of registers populated with the sensor data. The SPI port data output also uses two discrete lines to ensure timely availability of the data on the registers. The DATA READY line can be monitored to define when the data is available on the registers, and the 1KHZ PULSE line can be input to synchronize the data. See Section 4 of the manual for a detailed SPI communications definition.

For the UART port, IMU data can be output at a selectable fixed rate (100, 50, 25, 20, 10, 5 or 2 Hz) or on an as requested basis using the GP, 'Get Packet' command. The UART port IMU data is available in the Scaled Sensor Data ('S1' Packet) measurement packet format, which is output in scaled engineering units. See Section 6 of the manual for a detailed description of the data S1 data packet.

4 IMU380ZA-200 SPI Port Interface Definition

The IMU380ZA-200 provides a SPI port interface for data communications. This section of the manual will define the IMU380ZA-200 register map, register control capabilities, and the data register reading and writing methodology.

4.1 IMU380ZA-200 Register Map

Table 3 contains the register map defined for the IMU380ZA-200.

Table 3: IMU380ZA-200 Registry Map

Name	Read/Write	Address	Default	Function
Reserved	N/A	0x00 to 0x03	N/A	
X_RATE	R	0x04	N/A	X-Axis Rate-Sensor Output (Most-Significant Byte)
Y_RATE	R	0x06		Y-Axis Rate-Sensor Output (Most-Significant Byte)
Z_RATE	R	0x08		Z-Axis Rate-Sensor Output (Most-Significant Byte)
X_ACC	R	0x0A	N/A	X-Axis Accelerometer Output (Most-Significant Byte)
Y_ACC	R	0x0C		Y-Axis Accelerometer Output (Most-Significant Byte)
Z_ACC	R	0x0E		Z-Axis Accelerometer Output (Most-Significant Byte)
Reserved	N/A	0x10 to 0x15	N/A	
RATE_TEMP	R	0x16	N/A	Temperature measured by the rate-sensor
BOARD_TEMP	R	0x18	N/A	Temperature measured by the board temperature-sensor
Reserved	N/A	0x1A to 0x33	N/A	Reserved for future use (algorithm output)
SELF_TEST	R/W	0x34/0x35	0x00	See Table 12: Initiate self-test
DATA_READY	R/W	0x35/0x34	0x04	See Table 12: Configure Data-Ready output signal
OUTPUT_DATA_RATE	R/W	0x36/0x37	0x01	See Table 13: Sets Output Data Rate (ODR)
SYSTEM_CLOCK	R/W	0x37/0x36	0x01	See Table 13: Sets the system clock (internal/external)*
RS_DYNAMIC_RANGE	R/W	0x38/0x39	0x02	See Table 14: Set the dynamic range of the rate-sensor
LOW_PASS_FILTER	R/W	0x39/0x38	0x06	See Table 14: Select the digital filter
Reserved	N/A	0x3A to 0x3B	N/A	
STATUS	R	0x3C	N/A	See Table 10: Diagnostic register
BURST_MODE	R	0x3E	N/A	Burst-Mode Command
Reserved	N/A	0x40 to 0x51	N/A	
MANUF_CODE	R	0x52	0x1310	Manufacturing code indicating year and location
UNIT_CODE	R	0x54	0x0000	Unit information code
PRODUCT_ID	R	0x56	0x3810	Product identification code
SERIAL_NUMBER	R	0x58	Varies	Serial number
Reserved	N/A	0x5A to 0x7F	N/A	

* If an external sync pulse is applied, then the system cannot return to using internal timing without resetting the system and removing the sync signal

4.2 IMU380ZA-200 SPI Register Read Methodology

The IMU380ZA-200 SPI port uses registers to store two types of data:

1) Sensor data
2) Configuration/Status information

The user master device accesses this information via the SPI bus in one of two ways:

1) Polled-Mode
2) Burst-Mode

In polled-mode, the IMU380ZA-200 transfers information from any memory location back to the master in two (or more) SPI cycles[1]. In Burst-Mode, the IMU380ZA-200 transfers a predefined block of data in one contiguous group of nine SPI cycles.

4.2.1 IMU380ZA-200 SPI Port Polled-Mode Read

In polled-mode, data transfer begins when the SPI master sets the chip-select line (nSS) low and clocks a 16-bit word (the combination of the register-address byte and a byte of "don't care" bits) across the MOSI line. For example, the master sends the command 0x5800 to request the unit's serial number, stored in register 0x58. The IMU380ZA-200 returns information from this address across the MISO line during the subsequent 16 clock-cycles.

The commands sent by the master to the IMU380ZA-200 during the subsequent SPI cycle consist of either:

1) Sixteen "don't care" bits (0x0000, for example) to complete the read of a single register.
2) The address of another register followed by a byte of "don't care" values, e.g. 0x00. This permits back-to-back reads from non-contiguous registers.

Figure 6 illustrates a polled-mode read of a single-register (x-axis rate-sensor data), which is composed of two bytes starting at register address 0x04.

The SPI master initiates a register read by clocking in the address followed by 0x00, i.e. 0x0400, via MOSI; this combination is referred to as a read-command[2]. This is followed by an additional 16-bits (0x0000, in this case) to complete the SPI data-transfer cycle. As the master transmits the read command on MOSI, the IMU380ZA-200 begins to transmit data back over MISO. The data-word that is transmitted, as the read-command is being sent, consists of 16 "don't care" bits. The subsequent 16-bit message contains the x-axis rate-sensor information (most significant byte followed by least-significant byte).

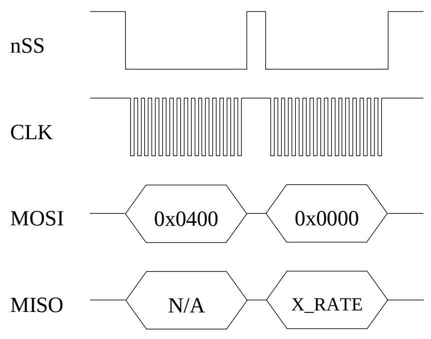


Figure 6: Single Register Read via Polled-Mode

Figure 7 illustrates a multiple register polled-read. As in the previous example, the SPI-master transmits the read command (the desired register-address appended by 0x00) across the MOSI line followed by any number of additional read-commands (one for each register of interest). The IMU380ZA-200 transfers the requested information concurrently across the MISO line to the master. To complete the data transfer, the final read-command must be followed by an additional 16 clock cycles to transfer the last 16-bits of data.

In this example, the master requests data from four separate registers: x-axis rate (0x0400), y-axis rate (0x0600), z-axis acceleration (0x0E00), and system status (0x3C00). The transfer of 0x0000 across MOSI line completes the read by returning the status data across the MISO line.

[1] A SPI cycle consists of 16 clock cycles.

[2] A read-command consists of an 8-bit register address and a zero byte (0x00).

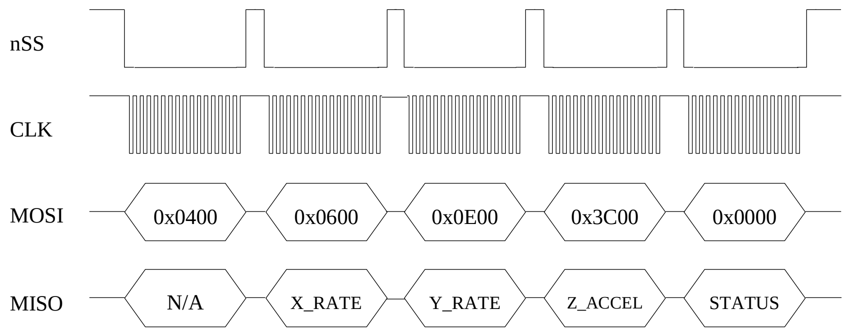


Figure 7: Multiple Register Read via Polled-Mode

4.2.2 *IMU380ZA-200 SPI Port Burst-Mode Read*

In burst-mode, the IMU380ZA-200 returns data from eight predefined registers across the SPI bus without the need to provide individual register addresses. Table 4 lists the registers returned during a burst-mode read in the order in which they are sent.

Table 4: IMU380ZA-200 Burst-Mode Output Registers

Register Name	**Register Address**	**Description**
STATUS	0x3C	System Status
X_RATE	0x04	Rate Sensor Output (X-Axis)
Y_RATE	0x06	Rate Sensor Output (Y-Axis)
Z_RATE	0x08	Rate Sensor Output (Z-Axis)
X_ACCEL	0x0A	Accelerometer Output (X-Axis)
Y_ACCEL	0x0C	Accelerometer Output (Y-Axis)
Z_ACCEL	0x0E	Accelerometer Output (Z-Axis)
BOARD_TEMP	0x18	System Temperature

Burst-mode begins when the master requests a read from register 0x3E. Eight additional SPI cycles completes the read. Note: exactly eight SPI cycles must follow the burst-mode command or the system will remain in burst-mode until the eighth cycle is complete; subsequent reads/writes will be out of sync with the SPI transfer cycle of the IMU380ZA-200. Figure 8 illustrates the burst-mode sequence.

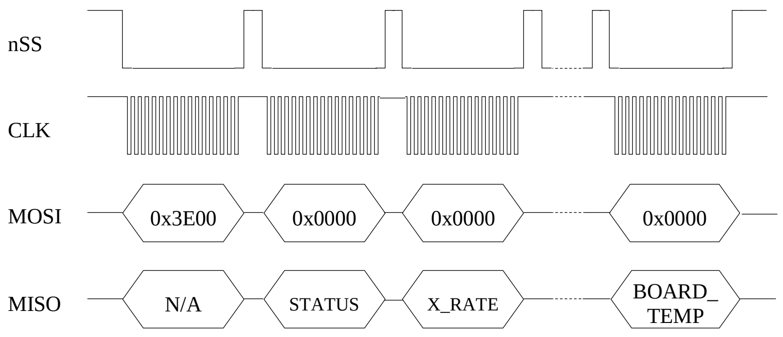


Figure 8: Multiple Register Read via Burst-Mode

Note: During this time the chip-select line (nSS) can be toggled between 16-bit words or held low the entire time. However, after the transfer is complete, chip-select must be set high.

4.3 Output Data Registers

The output data registers hold the sensor information as it is measured; they are overwritten only when new data is available. Table 5 lists each register, its memory address and its conversion factor.

Table 5: IMU380ZA-200 Data Output Registers

Name	Read Address	Function
X_RATE	0x04	X, Y, Z-axis rate-sensor information, twos complement format, conversion factor: 200 LSB/[°/sec] (default); changes with selected dynamic range (Table 11)
Y_RATE	0x06	
Z_RATE	0x08	
X_ ACCEL	0x0A	X, Y, Z-axis accelerometer information, twos complement format, conversion factor: 4000 LSB/g
Y_ ACCEL	0x0C	
Z_ACCEL	0x0D	
RATE_TEMP	0x16	Rate-sensor temperature information, twos complement format, conversion: Tout [°C] = Vout · 0.7311 [°C/LSB] + 31.0 [°C]
BOARD_TEMP	0x18	System temperature information, twos complement format, conversion: Tout [°C] = Vout · 0.7311 [°C/LSB] + 31.0 [°C]

4.4 System Registers

In addition to the output data registers, there are five additional read-only registers that provide IMU380ZA-200 system information to the SPI master. Table 6 provides a description of each along with their read-addresses.

Table 6: IMU380ZA-200 System Registers

Name	Read Address	Function
DIAGNOSTIC_STATUS	0x3C	See Table 10
MANUF_CODE	0x37	Contains the product manufacturing code
UNIT_CODE	0x36	Additional information about product manufacturing
PRODUCT_ID	0x39	Contains the product ID (0x3810)
SERIAL_NUMBER	0x38	Unique product identification number

4.4.1 Diagnostic Status Register

The diagnostic status register contains information describing the results of the self-test as well as sensor over-range information and is defined in Table 7.

Table 7: Diagnostic Status Register

(Base Address: 0x3C), Read-Only	
Bits	**Description (Default: 0x0000)**
15	Accelerometer Z-Axis self-test bit 0: Pass, 1: Fail
14	Accelerometer Y-Axis self-test bit 0: Pass, 1: Fail
13	Accelerometer X-Axis self-test bit 0: Pass, 1: Fail
12	Rate-Sensor Z-Axis self-test bit 0: Pass, 1: Fail
11	Rate-Sensor Y-Axis self-test bit 0: Pass, 1: Fail
10	Rate-Sensor X-Axis self-test bit 0: Pass, 1: Fail
[9:6]	Unused
5	Self-Test Success/Failure bit 0: Success, 1: Failure
4	Sensor over-range bit (a 1 indicates one or more sensors have over-ranged)
[3:0]	Unused

4.5 IMU380ZA-200 SPI Register Write Methodology

The user SPI master configures the IMU380ZA-200 by writing to specified registers (Table 6). However, unlike reads, writes are performed one byte at a time. The specific registers that affect system configuration are listed in Table 8 along with their write-addresses.

Table 8: IMU380ZA-200 Configuration Registers

Name	Write Address	Function
SELF_TEST	0x35	See Table 12: Initiate self-test
DATA_READY	0x34	See Table 12: Configure Data-Ready output signal
OUTPUT_DATA_RATE	0x37	See Table 13: Sets Output Data Rate (ODR)
SYSTEM_CLOCK	0x36	See Table 13: Sets internal/external system clock
RS_DYNAMIC_RANGE	0x39	See Table 14: Set the rate-sensor dynamic range
LOW_PASS_FILTER	0x38	See Table 14: Select the digital filter

The following example highlights how write-commands are formed in order to initiate a sensor self-test:

- Select the write address of the desired register, e.g. 0x35 for self-test
- Change the most-significant bit of the address to 1 (the write-bit), e.g. 0x35 becomes 0xB5
- Create the write command by appending the write-bit/address combination with the value to be written to the register, e.g. 0xB504 (see Table 10 for a description of the self-test register)

Figure 9 illustrates the sensor self-test command sent over SPI.

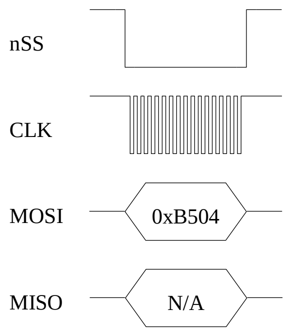


Figure 9: Single Register Write to Initiate Self-Test

As described in Table 9 below, the self-test command bit remains set until the test completes. The master must read from register 0x34 to assess if the test is complete (Figure 10). Note: as described in the Register Reads section, a register read returns two bytes, in this case a read from register 0x34 returns data from registers 0x34 (self-test information) and 0x35 (data-ready settings). The value read from the IMU380ZA-200 must be parsed according to Table 9 to determine self-test status.

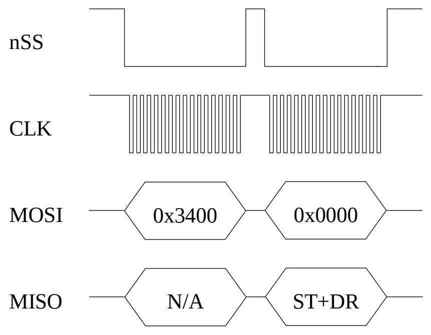


Figure 10: Polled-Read of the Self-Test/Data-Ready Register

4.6 Configuration Registers

4.6.1 Self-Test/Data-Ready

Self-test and data-ready registers are combined into a single 16-bit register at memory location 0x34; individual bits are assigned according to Table 9.

Table 9: Self-Test/Data-Ready Register

(Base Address: 0x34), Read/Write	
Bits	**Description (Default: 0x0004)**
[15:11]	Unused
10	Unit self-test bit (bit reset upon completion of self-test) 0: Disabled (default) 1: Enabled
[9:8]	Unused
[7:3]	Unused
2	Data-ready enable bit 0: Disabled

	1: Enabled (default)
1	Data-ready line polarity 0: Low upon data-ready (default) 1: High upon data-ready
0	Unused

The self-test enables the system to test individual sensors by applying a temporary bias to determine if they are responding correctly. Once self-test completes, the self-test bit (bit 10) is reset to indicate that the test is finished. Results of the self-test are store in the status register, 0x3C. To initiate self-test, the master sends 0xB504 across the SPI bus.

The data-ready bits enable the master to enable or disable the data-ready signal provided on pin 7 of the IMU380ZA-200 and to set the data-ready signal polarity (high or low). To enable data-ready with a high signal, the master sends 0xB406.

4.6.2 Output Data Rate/Clock Configuration

Output data rate (ODR) and system clock configuration are combined into a single 16-bit register at memory location 0x36; individual bits are assigned according to Table 10. Note that these settings apply only to data being output via the IMU380ZA-200 SPI port and do not affect the low-level UART output port.

Table 10: Output Data Rate/Clock Configuration Register

(Base Address: 0x36), Read/Write	
Bits	**Description (Default: 0x0101)**
[15:12]	Unused
[8:11]	System Output Data Rate 0x00 (0): Data output suppressed 0x01 (1): 200 Hz (default) 0x02 (2): 100 Hz 0x03 (3): 50 Hz 0x04 (4): 25 Hz 0x05 (5): 20 Hz 0x06 (6): 10 Hz 0x07 (7): 5 Hz 0x08 (8): 4 Hz 0x09 (9): 2 Hz 0x10 (10): 1 Hz
[7:1]	Unused
0	System Clock (currently unused) 0: External Clock 1: Internal Clock (default)

The ODR enables the master to specify the output rate of data provided by the IMU380ZA-200. Setting this register directly affects the data-ready signal. The default ODR is 200 Hz; to change the ODR to 100 Hz, the master sends 0xB702.

The system clock setting (bit 0) switches between the internal (IMU380ZA-200) system clock and the external 1 kHz sync signal. To switch to an external clock, the master sends 0xB600.

4.6.3 Rate-Sensor Scaling/Low-Pass Filter

The rate-sensor scaling and digital low-pass filter configuration are combined into a single 16-bit register at memory location 0x38; individual bits are assigned according to Table 11. Note that these settings apply only to data being output via the IMU380ZA-200 SPI port and do not affect the low-level UART output port.

Table 11: Sensor Scaling/Digital Low-Pass Filter Register

(Base Address: 0x38), Read/Write	
Bits	**Description (Default: 0x0206)**
[15:12]	Unused
[8:11]	Rate-Sensor Scaling/Dynamic Range Selector 0x1 (1): +/-62.5°/sec 0x2 (2): +/-125.0°/sec (default) 0x4 (4): +/-250.0°/sec
[7:0]	Digital Low-Pass Filter 0x00 (0): Unfiltered 0x03 (3): 40 Hz Bartlett 0x04 (4): 20 Hz Bartlett 0x05 (5): 10 Hz Bartlett 0x06 (6): 5 Hz Bartlett (default) 0x30 (48): 50 Hz Butterworth 0x40 (64): 20 Hz Butterworth 0x50 (80): 10 Hz Butterworth 0x60 (96): 5 Hz Butterworth

The rate-sensor scaling selector adjusts the output scaling applied to the rate-sensor as well as the limits that control the sensor over-range bit in the diagnostic status register (Table 7); if the system undergoes motion that exceeds this limit, the over-range bit is set. The default scaling is 125.0°/sec; to change the scaling to 62.5°/sec, the master sends 0xB901.

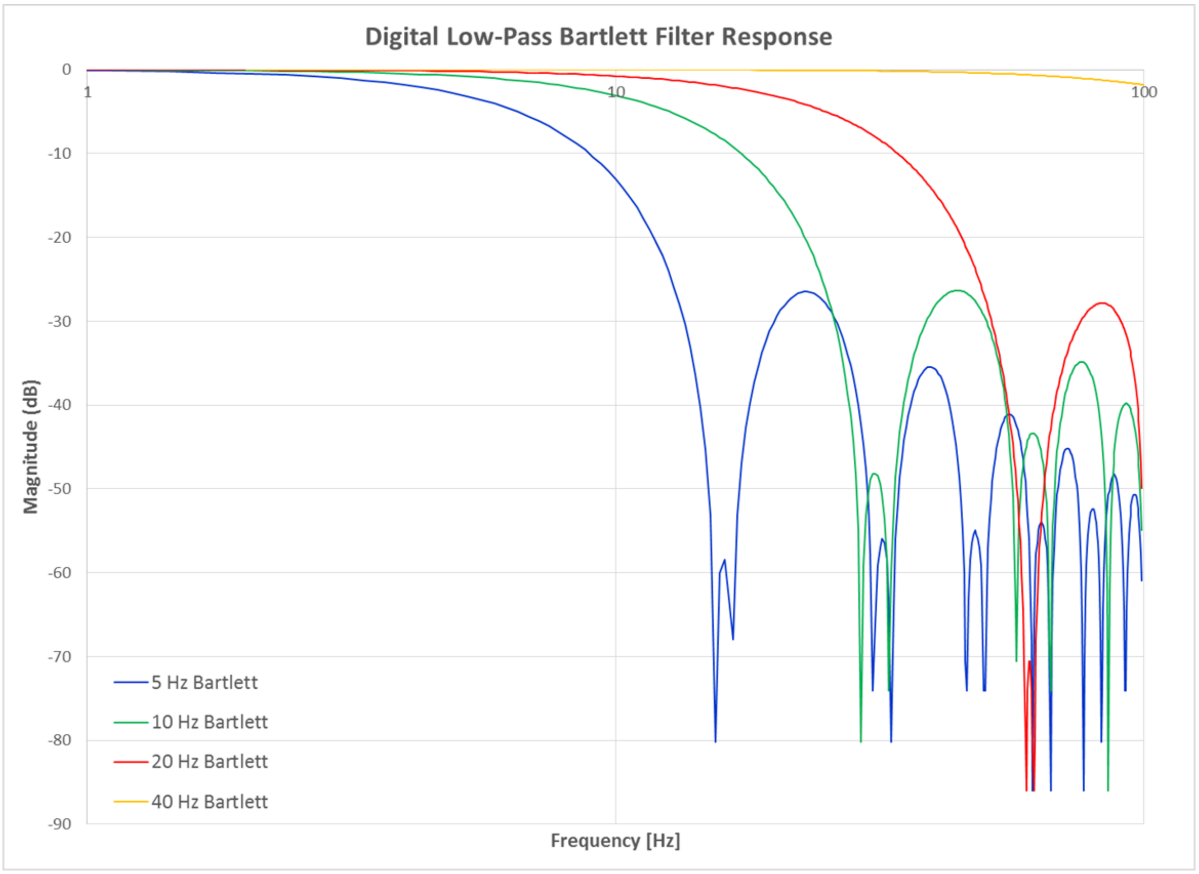
The rate sensor dynamic range selection maps to a bit-weight scale factor as defined in Table 12.

Table 12: Rate-Sensor Scaling Factor

Dynamic Range	Scale Factor
+/-62.5°/sec	400 LSB/(°/sec)
+/-125.0°/sec	200 LSB/(°/sec)
+/-250.0°/sec	100 LSB/(°/sec)

The digital low-pass filter register sets the type and cutoff frequency of the filter applied to the scaled sensor data. The default setting is a 5 Hz Bartlett filter; to switch to a 20 Hz Butterworth filter, the master sends 0xB840. Figure 11 describes the output response of the different Bartlett filter settings.

Figure 11: IMU380ZA-200 Bartlett Filter Response



5 IMU380ZA-200 UART Port Interface Definition

The IMU380ZA-200 supports a common MEMSIC packet structure that includes both command, also referred to as input data packets (data sent to the IMU380ZA-200), and measurement output, also referred to as response packet formats (data sent from the IMU380ZA-200), over the UART port. This section of the manual explains these packet formats as well as the supported commands. NAV-VIEW 3.X also features a number of tools that can help a user understand the packet types available and the information contained within the packets. This section of the manual assumes that the user is familiar with ANSI C programming language and data type conventions. For an example of the code required to parse data packets over the UART port, please refer to Appendix B.

For qualified commercial OEM users, a source code license of NAV-VIEW 3.X can be made available under certain conditions. Please contact your MEMSIC representative for more information.

5.1 General Settings

The UART serial port settings are 1 start bit, 8 data bits, no parity bit, 1 stop bit, and no flow control. Standard baud rates supported are: 9600, 19200, 38400, and 57600.

Common definitions include:

- A word is defined to be 2 bytes or 16 bits.
- All communications to and from the unit are packets that start with a single word alternating bit preamble 0x5555. This is the ASCII string "UU".
- All multiple byte values are transmitted Big Endian (Most Significant Byte First).
- All communication packets end with a single word CRC (2 bytes). CRC's are calculated on all packet bytes excluding the preamble and CRC itself. Input packets with incorrect CRC's will be ignored.
- Each complete communication packet must be transmitted to the IMU380ZA-200 inertial system within a 4 second period.

5.2 Default UART Port Settings

The IMU380 default settings are described in Table 13 below. All of the IMU380 UART port's advanced settings are accessible thru NAV-VIEW 3.X under the Configuration Menu -> Unit Configuration settings.

Table 13: IMU380 UART Port Default Settings

Setting	Default	Comments
Baud Rate	38400	9600, 19200, 57600 also available
Packet Type	S1	This is the only data packet available for output over the UART port.
Packet Rate	100Hz	This parameter sets the rate at which the selected Packet Type packets are output. If polled mode is desired, then select Quiet. If Quiet is selected, the IMU380 will only send measurement packets in response to GP commands.
Orientation	See Figure 5	To configure the axis orientation, select the desired measurement for each axis: NAV-VIEW 3.X will show the corresponding image of the IMU380ZA-200 so it easy to visualize the mode of operation. Refer to Section 7.3 Orientation Field settings for the twenty four possible orientation settings. The default setting points the connector to the right of the vehicle.

5.3 Number Formats

Number Format Conventions include:

- 0x as a prefix to hexadecimal values
- single quotes (‘’) to delimit ASCII characters
- no prefix or delimiters to specify decimal values.

Table 14 defines number formats:

Table 14: Number Formats

Descriptor	Description	Size (bytes)	Comment	Range
U1	Unsigned Char	1		0 to 255
U2	Unsigned Short	2		0 to 65535
U4	Unsigned Int	4		0 to 2^32-1
I2	Signed Short	2	2's Complement	-2^15 to 2^15-1
I2*	Signed Short	2	Shifted 2's Complement	Shifted to specified range
I4	Signed Int	4	2's Complement	-2^31 to 2^31-1
F4	Floating Point	4	IEEE754 Single Precision	-1*2^127 to 2^127
SN	String	N	ASCII	

5.4 Packet Format

All of the Input and Output packets, except the Ping command, conform to the following structure:

0x5555	*<2-byte packet type (U2)>*	*<payload byte-length (U1)>*	*<variable length payload>*	*<2-byte CRC (U2)>*

The Ping Command does not require a CRC, so an IMU380ZA-200 unit can be pinged from a terminal emulator. To Ping an IMU380ZA-200 unit, type the ASCII string 'UUPK'. If properly connected, the IMU380ZA-200 unit will respond with 'PK'. All other communications with the IMU380ZA-200 unit require the 2-byte CRC. {Note: An IMU380ZA-200 unit will also respond to a ping command using the full packet formation with payload 0 and correctly calculated CRC. Example: 0x5555504B009ef4 }.

5.4.1 Packet Header

The packet header is always the bit pattern 0x5555.

5.4.2 Packet Type

The packet type is always two bytes long in unsigned short integer format. Most input and output packet types can be interpreted as a pair of ASCII characters. As a semantic aid consider the following single character acronyms:

P = packet

F = fields

Refers to Fields which are settings or data contained in the unit

E = EEPROM

Refers to factory data stored in EEPROM

R = read

Reads default non-volatile fields

G = get

Gets current volatile fields or settings

W = write

Writes default non-volatile fields. These fields are stored in non-volatile memory and determine the unit's behavior on power up. Modifying default fields take effect on the next power up and thereafter.

S = set

Sets current volatile fields or settings. Modifying current fields will take effect immediately by modifying internal RAM and are lost on a power cycle.

5.4.3 Payload Length

The payload length is always a one byte unsigned character with a range of 0-255. The payload length byte is the length(in bytes) of the *<variable length payload>* portion of the packet ONLY, and does not include the CRC.

5.4.4 Payload

The payload is of variable length based on the packet type.

5.4.5 16-bit CRC-CCITT

Packets end with a 16-bit CRC-CCITT calculated on the entire packet excluding the 0x5555 header and the CRC field itself. A discussion of the 16-bit CRC-CCITT and sample code for implementing the computation of the CRC is included at the end of this document. This 16-bit CRC standard is maintained by the International Telecommunication Union (ITU). The highlights are:

Width = 16 bits

Polynomial 0x1021

Initial value = 0xFFFF

No XOR performed on the final value.

See Appendix C for sample code that implements the 16-bit CRC algorithm.

5.4.6 Messaging Overview

The following table summarizes the messages available by the IMU380ZA-200. Packet types are assigned mostly using the ASCII mnemonics defined above and are indicated in the summary table below and in the detailed sections for each command. The payload byte-length is often related to other data elements in the packet as defined in Table 15. The referenced variables are defined in the detailed sections following. Output messages are sent from the IMU380ZA-200 inertial system to the user system as a result of a poll request or a continuous packet output setting. Input messages are sent from the user system to the IMU380ZA-200 inertial system and will result in an associated Reply Message or NAK message. Note that reply messages typically have the same ***<2-byte packet type (U2)>*** as the input message that evoked it but with a different payload.

Table 15: Message Table

ASCII Mnemonic	***<2-byte packet type (U2)>***	***<payload byte-length (U1)>***	**Description**	**Type**
Link Test				
PK	0x504B	0	Ping Command and Response	Input/Reply Message
CH	0x4348	N	Echo Command and Response	Input/Reply Message
Interactive Commands				
GP	0x4750	2	Get Packet Request	Input Message
SR	0x5352	0	Software Reset	Input/Reply Message
NAK	0x1515	2	Error Response	Reply Message
Output Messages: Status & Other, (Polled Only)				
ID	0x4944	5+N	Identification Data	Output Message
VR	0x5652	5	Version Data	Output Message
Output				

Messages: **Measurement Data** **(Continuous or Polled)**				
S1	0x5331	24	Scaled Sensor 1 Data	Output Message
Advanced Commands				
WF	0x5746	numFields*4+1	Write Fields Request	Input Message
WF	0x5746	numFields*2+1	Write Fields Response	Reply Message
SF	0x5346	numFields*4+1	Set Fields Request	Input Message
SF	0x5346	numFields*2+1	Set Fields Response	Reply Message
RF	0x5246	numFields*2+1	Read Fields Request	Input Message
RF	0x5246	numFields*4+1	Read Fields Response	Reply Message
GF	0x4746	numFields*2+1	Get Fields Request	Input Message
GF	0x4746	numFields*4+1	Get Fields Response	Reply Message

6 IMU380ZA-200 Standard UART Port Commands and Messages

6.1 Link Test.

6.1.1 Ping Command

Ping ('PK' = 0x504B)			
Preamble	*Packet Type*	*Length*	*Termination*
0x5555	0x504B	-	-

The ping command has no payload. Sending the ping command will cause the unit to send a ping response. To facilitate human input from a terminal, the length and CRC fields are not required. (Example: 0x5555504B009ef4 or 0x5555504B))

Ping ('PK' = 0x504B)			
Preamble	*Packet Type*	*Length*	*Termination*
0x5555	0x504B	*0x00*	*<CRC (U2)>*

The unit will send this packet in response to a ping command.

6.1.2 Echo Command

Echo ('CH' = 0x4348)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x4348	N	*<echo payload>*	*<CRC (U2)>*

The echo command allows testing and verification of the communication link. The unit will respond with an echo response containing the *echo data*. The *echo data* is N bytes long.

Echo Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	echoData0	U1	-	-	first byte of echo data
1	echoData1	U1	-	-	Second byte of echo data
…	…	U1	-	-	Echo data
N-2	echoData...	U1	-	-	Second to last byte of echo data
N-1	echoData…	U1	-	-	Last byte of echo data

6.2 Interactive Commands

Interactive commands are used to interactively request data from the IMU380ZA-200, and to calibrate or reset the IMU380ZA-200.

6.2.1 Get Packet Request

Get Packet ('GP' = 0x4750)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x4750	0x02	*<GP payload>*	*<CRC (U2)>*

This command allows the user to poll for both measurement packets and special purpose output packets including 'VR' and 'ID'.

GP Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	requestedPacketType	U2	-	-	The requested packet type

Refer to the sections below for Packet Definitions sent in response to the 'GP' command

6.2.2 Software Reset Command

Software Reset ('SR' = 0x5352)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5352	0x00	-	*<CRC (U2)>*

This command performs a core CPU reset, functionally equivalent to a power cycle. All default power-up field settings will apply. The unit will respond with a software reset response before the system goes down.

Software Reset ('SR' = 0x5352)			
Preamble	*Packet Type*	*Length*	*Termination*
0x5555	0x5352	0x00	*<CRC (U2)>*

The unit will send this packet in response to a software reset command.

6.2.3 Error Response

Error Response (ASCII NAK, NAK = 0x1515)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x1515	0x02	*<NAK payload>*	*<CRC (U2)>*

The unit will send this packet in place of a normal response to a *failedInputPacketType* request if it could not be completed successfully.

NAK Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	failedInputPacketType	U2	-	-	the failed request

6.3 Output Packets (Polled)

The following packet formats are special informational packets which can be requested using the 'GP' command.

6.3.1 Identification Data Packet

Identification Data ('ID' = 0x4944)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x4944	5+N	*<ID payload>*	*<CRC (U2)>*

This packet contains the unit *serialNumber* and *modelString*. The model string is terminated with 0x00. The model string contains the programmed versionString (8-bit Ascii values) followed by the firmware part number string delimited by a whitespace.

ID Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	serialNumber	U4	-	-	Unit serial number
4	modelString	SN	-	-	Unit Version String
4+N	0x00	U1	-	-	Zero Delimiter

6.3.2 Version Data Packet

Version Data ('VR' = 0x5652)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5652	5	*<VR payload>*	*<CRC (U2)>*

This packet contains firmware version information. *majorVersion* changes may introduce serious incompatibilities. *minorVersion* changes may add or modify functionality, but maintain backward compatibility with previous minor versions.

patch level changes reflect bug fixes and internal modifications with little effect on the user. The build *stage* is one of the following: 0=release candidate, 1=development, 2=alpha, 3=beta. The *buildNumber* is incremented with each engineering firmware build. The *buildNumber* and *stage* for released firmware are both zero. The final beta candidate is v.w.x.3.y, which is then changed to v.w.x.0.1 to create the first release candidate. The last release candidate is v.w.x.0.z, which is then changed to v.w.x.0.0 for release.

VR Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	majorVersion	U1	-	-	Major firmware version
1	minorVersion	U1	-	-	Minor firmware version
2	patch	U1	-	-	Patch level
3	stage	-	-	-	Development Stage (0=release candidate, 1=development, 2=alpha, 3=beta)
4	buildNumber	U1	-	-	Build number

6.4 Output Packets (Polled or Continuous)

6.4.1 Scaled Sensor Data Packet 1 (Default IMU Data)

Scaled Sensor Data ('S1' = 0x5331)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5331	0x18	*<S1 payload>*	*<CRC (U2)>*

This packet contains scaled sensor data. Data involving angular measurements include the factor pi in the scaling and can be interpreted in either radians or degrees.

Angular rates: scaled to range of 3.5* [-pi,+pi) or [-630 deg/sec to +630 deg/sec)

Accelerometers: scaled to a range of [-10,+10)g

Temperature: scaled to a range of [-100, +100)°C

S1 Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	xAccel	I2	20/2^16	g	X accelerometer
2	yAccel	I2	20/2^16	g	Y accelerometer
4	zAccel	I2	20/2^16	g	Z accelerometer
6	xRate	I2	7*pi/2^16 [1260°/2^16]	rad/s [°/sec]	X angular rate
8	yRate	I2	7*pi/2^16 [1260°/2^16]	rad/s [°/sec]	Y angular rate
10	zRate	I2	7*pi/2^16 [1260°/2^16]	rad/s [°/sec]	Z angular rate
12	xRateTemp	I2	200/2^16	deg. C	X rate temperature
14	yRateTemp	I2	200/2^16	deg. C	Y rate temperature
16	zRateTemp	I2	200/2^16	deg. C	Z rate temperature
18	reserved	I2	200/2^16	deg. C	Not used
20	Counter	U2	-	packets	Output packet counter
22	reserved	U2	-	-	Not used

7 IMU380ZA-200 Advanced UART Port Commands

The advanced commands allow users to programmatically change the IMU380ZA-200 settings. This section of the manual documents all of the settings and options contained under the Unit Configuration tab within NAV-VIEW 3.X. Using these advanced commands, a user's system can change or modify the settings without the need for NAV-VIEW 3.X.

7.1 Configuration Fields

Configuration fields determine various behaviors of the unit that can be modified by the user. These include settings like baud rate, packet output rate and axes orientation. These fields are stored in EEPROM and loaded on power up. These fields can be read from the EEPROM using the 'RF' command. These fields can be written to the EEPROM affecting the default power up behavior using the 'WF' command. The current value of these fields (which may be different from the value stored in the EEPROM) can also be accessed using the 'GF' command. All of these fields can also be modified immediately for the duration of the current power cycle using the 'SF' command. The unit will always power up in the configuration stored in the EEPROM. Configuration fields can only be set or written with valid data from the table below.

Configuration fields	Field ID	Valid Values	Description
Packet rate divider	0x0001	0,1,2,5,10, 20, 25, 50	quiet, 100Hz, 50Hz, 25Hz, 20Hz, 10Hz, 5Hz, 2Hz
Unit BAUD rate	0x0002	0,1,2,3	9600, 19200, 38400, 57600
Continuous packet type	0x0003	Any output packet type	Not all output packets available for all products. See detailed product descriptions.
Orientation	0x0007	See below	Determine forward, rightward, and downward facing sides

Note: BAUD rate SF has immediate effect. Some output data may be lost. Response will be received at new BAUD rate.

7.2 Continuous Packet Type Field

This is the packet type that is being continually output. The supported packet depends on the model number. Please refer to Section 5.3 for a complete list of the available packet types.

7.3 Orientation Field

This field defines the rotation from the factory to user axis sets. This rotation is relative to the default factory orientation (connector aft, baseplate down). The default factory axis set is (Ux, Uy, Uz) defined by the connector pointing in the +Uy direction and the baseplate pointing in the +Uz direction. The user axis set is (X, Y, Z) as defined by this field. A depiction of the factory axis set is shown below in Figure 12.

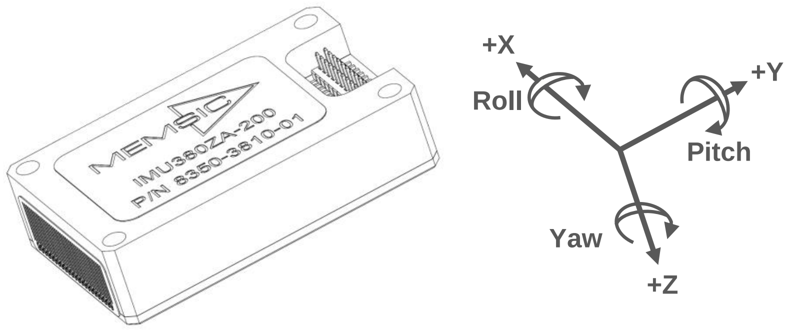


Figure 12: Default UART Port Orientation

Description	Bits	Meaning
X Axis Sign	0	0 = positive, 1 = negative
X Axis	1:2	0 = Ux, 1 = Uy, 2 = Uz, 3 = N/A
Y Axis Sign	3	0 = positive, 1 = negative
Y Axis	4:5	0 = Uy, 1 = Uz, 2 = Ux, 3 = N/A
Z Axis Sign	6	0 = positive, 1 = negative
Z Axis	7:8	0 = Uz, 1 = Ux, 2 = Uy, 3 = N/A
Reserved	9:15	N/A

There are 24 possible orientation configurations. Setting/Writing the field to anything else generates a NAK and has no effect.

Orientation Field Value	X Axis	Y Axis	Z Axis
0x0000	+Ux	+Uy	+Uz
0x0009	-Ux	-Uy	+Uz
0x0023	-Uy	+Ux	+Uz
0x002A	+Uy	-Ux	+Uz
0x0041	-Ux	+Uy	-Uz
0x0048	+Ux	-Uy	-Uz
0x0062	+Uy	+Ux	-Uz
0x006B	-Uy	-Ux	-Uz
0x0085	-Uz	+Uy	+Ux
0x008C	+Uz	-Uy	+Ux
0x0092	+Uy	+Uz	+Ux
0x009B	-Uy	-Uz	+Ux
0x00C4	+Uz	+Uy	-Ux
0x00CD	-Uz	-Uy	-Ux
0x00D3	-Uy	+Uz	-Ux
0x00DA	+Uy	-Uz	-Ux
0x0111	-Ux	+Uz	+Uy
0x0118	+Ux	-Uz	+Uy
0x0124	+Uz	+Ux	+Uy
0x012D	-Uz	-Ux	+Uy
0x0150	+Ux	+Uz	-Uy
0x0159	-Ux	-Uz	-Uy
0x0165	-Uz	+Ux	-Uy
0x016C	+Uz	-Ux	-Uy

7.4 Commands to Program Configuration

7.4.1 Write Fields Command

Write Fields ('WF' = 0x5746)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5746	*1+numFields*4*	*<WF payload>*	*<CRC (U2)>*

This command allows the user to write default power-up configuration fields to the EEPROM. Writing the default configuration will not take effect until the unit is power cycled. *NumFields* is the number of words to be written. The *field0, field1, etc.* are the field IDs that will be written with the *field0Data, field1Data, etc.*, respectively. The unit will not write to calibration or algorithm fields. If at least one field is successfully written, the unit will respond with a write fields response containing the field IDs of the successfully written fields. If any field is unable to be written, the unit will respond with an error response. Note that both a write fields and an error response may be received as a result of a write fields command. Attempts to write a field with an invalid value is one way to generate an error response. A table of field IDs and valid field values is available in Section 7.1.

WF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields to write
1	field0	U2	-	-	The first field ID to write
3	field0Data	U2	-	-	The first field ID's data to write
5	field1	U2	-	-	The second field ID to write
7	field1Data	U2	-	-	The second field ID's data
…	…	U2	-	-	…
numFields*4 -3	field…	U2	-	-	The last field ID to write
numFields*4 -1	field…Data	U2	-	-	The last field ID's data to write

7.4.2 Write Fields Response

Write Fields ('WF' = 0x5746)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5746	*1+numFields*2*	*<WF payload>*	*<CRC (U2)>*

The unit will send this packet in response to a write fields command if the command has completed without errors.

WF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields written
1	field0	U2	-	-	The first field ID written
3	field1	U2	-	-	The second field ID written
…	…	U2	-	-	More field IDs written
numFields*2 – 1	Field…	U2	-	-	The last field ID written

7.4.3 Set Fields Command

Set Fields ('SF' = 0x5346)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5346	*1+numFields*4*	*<SF payload>*	*<CRC (U2)>*

This command allows the user to set the unit's current configuration (SF) fields immediately which will then be lost on power down. *NumFields* is the number of words to be set. The *field0, field1, etc.* are the field IDs that will be written with the *field0Data, field1Data, etc.*, respectively. This command can be used to set configuration fields. The unit will not set calibration or algorithm fields. If at least one field is successfully set, the unit will respond with a set fields response containing the field IDs of the successfully set fields. If any field is unable to be set, the unit will respond with an error response. Note that both a set fields and an error response may be received as a result of one set fields command. Attempts to set a field with an invalid value is one way to generate an error response. A table of field IDs and valid field values is available in Section 7.1.

SF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields to set
1	field0	U2	-	-	The first field ID to set
3	field0Data	U2	-	-	The first field ID's data to set
5	field1	U2	-	-	The second field ID to set
7	field1Data	U2	-	-	The second field ID's data to set
…	…	U2	-	-	…
numFields*4 -3	field…	U2	-	-	The last field ID to set
numFields*4 -1	field…Data	U2	-	-	The last field ID's data to set

7.4.4 Set Fields Response

Set Fields ('SF' = 0x5346)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5346	*1+numFields*2*	*<SF payload>*	*<CRC (U2)>*

The unit will send this packet in response to a write fields command if the command has completed without errors.

SF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields set
1	field0	U2	-	-	The first field ID set
3	field1	U2	-	-	The second field ID set
…	…	U2	-	-	More field IDs set
numFields*2 – 1	Field…	U2	-	-	The last field ID set

7.4.5 Read Fields Command

Read Fields ('RF' = 0x5246)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5246	*1+numFields*2*	*<RF payload>*	*<CRC (U2)>*

This command allows the user to read the default power-up configuration fields from the EEPROM. *NumFields* is the number of fields to read. The *field0, field1, etc.* are the field IDs to read. RF may be used to read configuration and calibration fields from the EEPROM. If at least one field is successfully read, the unit will respond with a read fields response containing the field IDs and data from the successfully read fields. If any field is unable to be read, the unit will respond with an error response. Note that both a read fields and an error response may be received as a result of a read fields command.

RF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields to read

1	field0	U2	-	-	The first field ID to read
3	field1	U2	-	-	The second field ID to read
…	…	U2	-	-	More field IDs to read
numFields*2 - 1	Field…	U2	-	-	The last field ID to read

7.4.6 Read Fields Response

Read Fields ('RF' = 0x5246)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x5246	*1+numFields*4*	*<RF payload>*	*<CRC (U2)>*

The unit will send this packet in response to a read fields request if the command has completed without errors.

RF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields read
1	field0	U2	-	-	The first field ID read
3	field0Data	U2	-	-	The first field ID's data read
5	field1	U2	-	-	The second field ID read
7	field1Data	U2	-	-	The second field ID's data read
…	…	U2	-	-	…
numFields*4 -3	field…	U2	-	-	The last field ID read
numFields*4 -1	field…Data	U2	-	-	The last field ID's data read

7.4.7 Get Fields Command

Get Fields ('GF' = 0x4746)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x4746	*1+numFields*2*	*<GF Data>*	*<CRC (U2)>*

This command allows the user to get the unit's current configuration fields. *NumFields* is the number of fields to get. The *field0*, *field1, etc.* are the field IDs to get. GF may be used to get configuration, calibration, and algorithm fields from RAM. Multiple algorithm fields will not necessarily be from the same algorithm iteration. If at least one field is successfully collected, the unit will respond with a get fields response with data containing the field IDs of the successfully received fields. If any field is unable to be received, the unit will respond with an error response. Note that both a get fields and an error response may be received as the result of a get fields command.

GF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields to get
1	field0	U2	-	-	The first field ID to get
3	field1	U2	-	-	The second field ID to get
…	…	U2	-	-	More field IDs to get
numFields*2 – 1	Field…	U2	-	-	The last field ID to get

7.4.8 Get Fields Response

Get Fields ('GF' = 0x4746)				
Preamble	*Packet Type*	*Length*	*Payload*	*Termination*
0x5555	0x4746	*1+numFields*4*	*<GF Data>*	*<CRC (U2)>*

The unit will send this packet in response to a get fields request if the command has completed without errors.

GF Payload Contents					
Byte Offset	*Name*	*Format*	*Scaling*	*Units*	*Description*
0	numFields	U1	-	-	The number of fields retrieved
1	field0	U2	-	-	The first field ID retrieved
3	field0Data	U2	-	-	The first field ID's data retrieved
5	field1	U2	-	-	The second field ID retrieved
7	field1Data	U2	-	-	The second field ID's data
…	…	U2	-	-	…
numFields*4 -3	field…	U2	-	-	The last field ID retrieved
numFields*4 -1	field…Data	U2	-	-	The last field ID's data retrieved

8 Warranty and Support Information

8.1 Customer Service

As a MEMSIC customer you have access to product support services, which include:

- Single-point return service
- Web-based support service
- Same day troubleshooting assistance
- Worldwide MEMSIC representation
- Onsite and factory training available
- Preventative maintenance and repair programs
- Installation assistance available

8.2 Contact Directory

United States:	Phone: 1-408-964-9700 (8 AM to 5 PM PST)
	Fax:1-408-854-7702 (24 hours)
	Email: techsupportca@memsic.com
Non-U.S.:	Refer to website www.memsic.com

8.3 Return Procedure

8.3.1 Authorization

Before returning any equipment, please contact MEMSIC to obtain a Returned Material Authorization number (RMA).

Be ready to provide the following information when requesting a RMA:

- Name
- Address
- Telephone, Fax, Email
- Equipment Model Number
- Equipment Serial Number
- Installation Date
- Failure Date
- Fault Description
- Will it connect to NAV-VIEW 3.X?

8.3.2 Identification and Protection

If the equipment is to be shipped to MEMSIC for service or repair, please attach a tag TO THE EQUIPMENT, as well as the shipping container(s), identifying the owner. Also indicate the service or repair required, the problems encountered, and other information considered valuable to the service facility such as the list of information provided to request the RMA number.

Place the equipment in the original shipping container(s), making sure there is adequate packing around all sides of the equipment. If the original shipping containers were discarded, use heavy boxes with adequate padding and protection.

8.3.3 Sealing the Container

Seal the shipping container(s) with heavy tape or metal bands strong enough to handle the weight of the equipment and the container.

8.3.4 Marking

Please write the words, "***FRAGILE, DELICATE INSTRUMENT***" in several places on the outside of the shipping container(s). In all correspondence, please refer to the equipment by the model number, the serial number, and the RMA number.

8.3.5 Return Shipping Address

Use the following address for all returned products:

MEMSIC, Inc.

1759 McCarthy Blvd.

Milpitas, CA 95035

Attn: RMA Number (XXXXXX)

8.4 **Warranty**

The MEMSIC product warranty is one year from date of shipment.

Appendix A: Installation and Operation of NAV-VIEW 3.X

NAV-VIEW 3.X allows users to control certain aspects of the IMU380ZA-200 operation including data recording, configuration and data transfer. Note that NAV-VIEW will control only the data being output over the UART port.

8.1 NAV-VIEW 3.X Computer Requirements

The following are minimum requirements for the installation of the NAV-VIEW 3.X Software:

• CPU: Pentium-class (1.5GHz minimum)
• RAM Memory: 500MB minimum, 1GB+ recommended
• Hard Drive Free Memory: 20MB
• Operating System: Windows XP™, Windows 7™, or Windows 8™

8.1.1 Install NAV-VIEW 3.X

To install NAV-VIEW 3.X onto your computer:

1. Insert the CD "MEMSIC Inertial Systems Product Support" in the CD-ROM drive.
2. Locate the "NAV-VIEW 3.X" folder. Double click on the "setup.exe" file.
3. Follow the setup wizard instructions. You will install NAV-VIEW 3.X and .NET 2.0 framework.

8.2 Connections

The IMU380ZA-200 evaluation kit contains a USB cable and interface board that enables the user to quickly connect the IMU380ZA-200 to a PC USB port.

1. The interface board contains a mating connector for the IMU380ZA-200 and a micro-USB connector. The board can be connected to the IMU380ZA-200 by physically attaching the IMU connector to the interface board connector. The board is designed to stay flat against the IMU380ZA-200. Connect the micro-USB port end of the cable to the interface board micro-USB connector.
2. Connect the standard USB port end of the cable marked "IMU380ZA-200 USB" to a standard USB port on your computer. If this is 1st time connecting this cable to your PC, then the built in cable electronics will identify itself to Windows and install the necessary interface driver so that Windows will enable it as a standard serial COM port. Subsequent connections to different USB ports will repeat this process. Subsequent connections to the same USB port should allow Windows to simply recognize the cable device and re-assign the same COM port.
3. Make sure the USB port on the PC is a "powered" USB port capable of supplying 5 volts (3-5 volts). The PC USB port will then power the system (IMU380ZA-200 and Adapter Board).

8.3 Setting up NAV-VIEW 3.X

With the IMU380ZA-200 and adapter board powered up and connected to your PC USB port, open the NAV-VIEW 3.X software application.

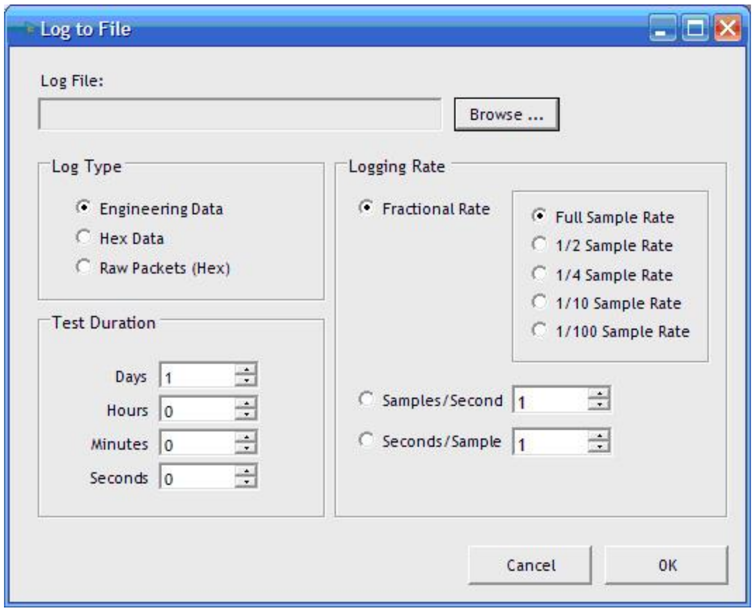
1. NAV-VIEW 3.X should automatically detect the IMU380ZA-200 product and display the serial number and firmware version if it is properly connected.
2. If NAV-VIEW 3.X does not connect, check that you have the correct COM port selected. You will find this under the "Setup" menu. Select the appropriate COM port and allow the unit to automatically match the baud rate by leaving the "Auto: match baud rate" selection marked. Discover the COM port set for the adapter cable in the standard Windows Control Panel -> System - > Device Manager application.
3. If the status indicator at the bottom is green and states, Unit Connected, you're ready to go. If the status indicator doesn't say connected and is red, check the connections between the IMU380ZA-200 product and the computer and verify that the COM port is not occupied by another device. If there is a COM port conflict, attempt to remap the adapted cable COM port selection using the standard Windows Control Panel -> System - > Device Manager application.
4. Under the "View" menu you have several choices of data presentation. Graph display is the default setting and will provide a real time graph of all the IMU380ZA-200 data. The remaining choices will be discussed in the following pages.

8.4 Data Recording

NAV-VIEW 3.X allows the user to log data to a text file (.txt) using the simple interface at the top of the screen. Customers can now tailor the type of data, rate of logging and can even establish predetermined recording lengths.

To begin logging data follow the steps below:

1. Locate the icon at the top of the page or select "Log to File" from the "File" drop down menu.
2. The following menu will appear, as shown in Figure 13.



Log to File

Log File:

Browse ...

Log Type
- [✓ Engineering Data]
- Hex Data
- Raw Packets (Hex)

Test Duration

Days	1
Hours	0
Minutes	0
Seconds	0

Logging Rate
- [✓ Fractional Rate]
  - [✓ Full Sample Rate]
  - 1/2 Sample Rate
  - 1/4 Sample Rate
  - 1/10 Sample Rate
  - 1/100 Sample Rate
- Samples/Second: 1
- Seconds/Sample: 1

Cancel OK

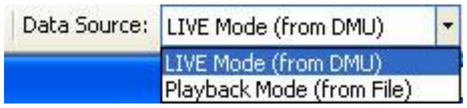
Figure 13: Log to File

3. Select the "Browse" box to enter the file name and location that you wish to save your data to.
4. Select the type of data you wish to record. "Engineering Data" records the converted values provided from the system in engineering units, "Hex Data" provides the raw hex values separated into columns displaying the value, and the "Raw Packets" will simply record the raw hex strings as they are sent from the unit.
5. Users can also select a predetermined "Test Duration" from the menu. Using the arrows, simply select the duration of your data recording.
6. Logging Rate can also be adjusted using the features on the right side of the menu.
7. Once you have completed the customization of your data recording, you will be returned to the main screen where you can start the recording process using the ● button at the top of the page or select "Start Logging" from the "File" menu. Stopping the data recording can be accomplished using the ■ button and the recording can also be paused using the ‖ button.

8.5 Data Playback

In addition to data recording, NAV-VIEW 3.X allows the user to replay saved data that has been stored in a log file.

1. To playback data, select "Playback Mode" from the "Data Source" drop down menu at the top.

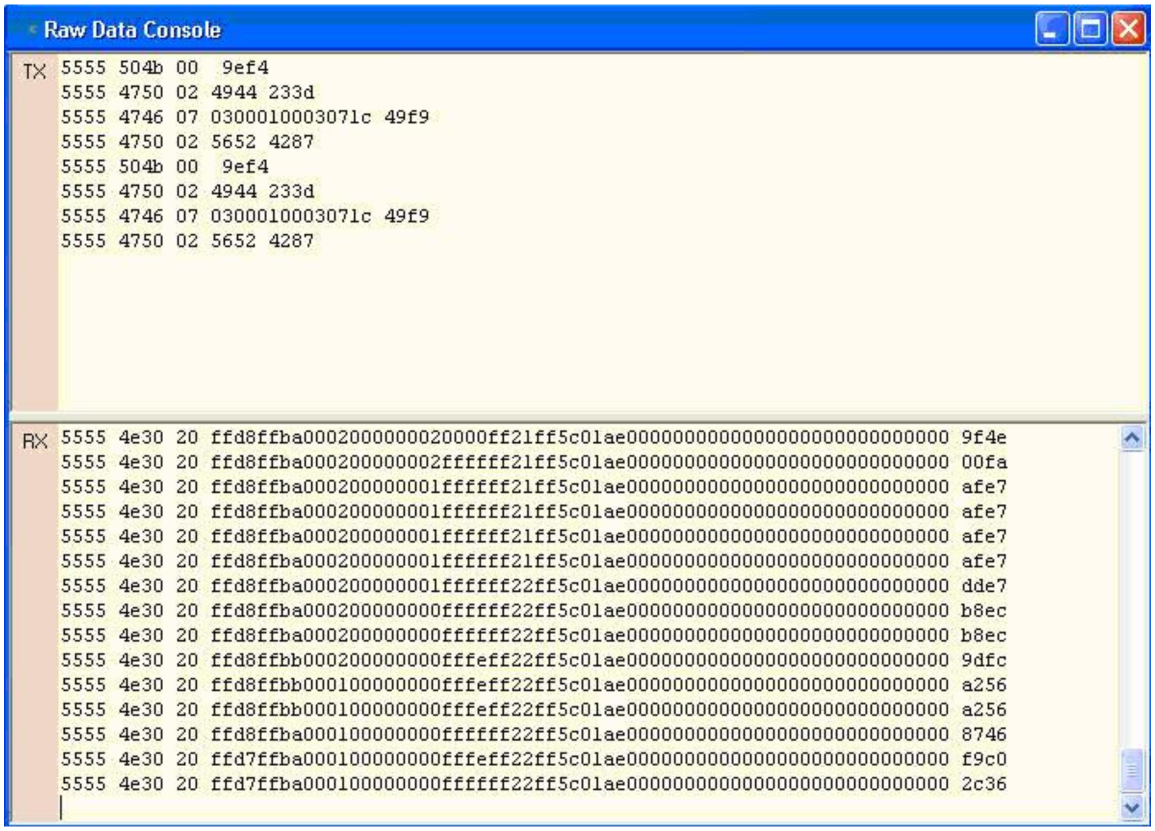


2. Selecting Playback mode will open a text prompt which will allow users to specify the location of the file they wish to play back. All three file formats are supported (Engineering, Hex, and Raw) for playback. In addition, each time recording is stopped/started a new section is created. These sections can be individually played back by using the drop down menu and associated VCR controls.
3. Once the file is selected, users can utilize the VCR style controls at the top of the page to start, stop, and pause the playback of the data.

4. NAV-VIEW 3.X also provides users with the ability to alter the start time for data playback. Using the slidebar at the top of the page users can adjust the starting time.

8.6 Raw Data Console

NAV-VIEW 3.X offers some unique debugging tools that may assist programmers in the development process. One such tool is the Raw Data Console. From the "View" drop down menu, simply select the "Raw Data Console". This console provides users with a simple display of the packets that have been transmitted to the unit (Tx) and the messages received (Rx). An example is provided below in Figure 14.



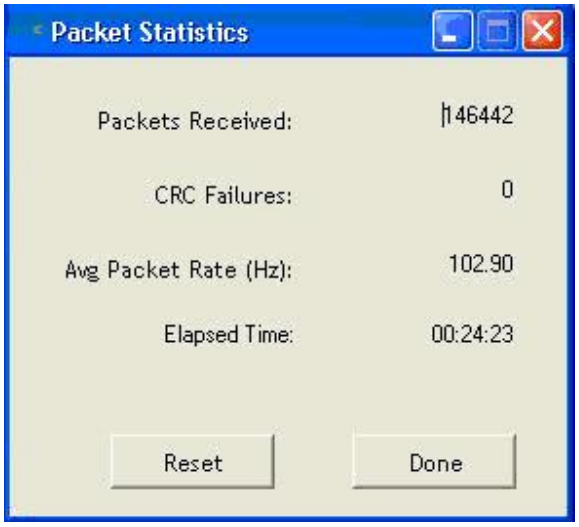
```
TX  5555 504b 00  9ef4
    5555 4750 02 4944 233d
    5555 4746 07 030001000307lc 49f9
    5555 4750 02 5652 4287
    5555 504b 00  9ef4
    5555 4750 02 4944 233d
    5555 4746 07 030001000307lc 49f9
    5555 4750 02 5652 4287
```

```
RX  5555 4e30 20 ffd8ffba0002000000020000ff21ff5c01ae0000000000000000000000000000 9f4e
    5555 4e30 20 ffd8ffba000200000002ffffff21ff5c01ae0000000000000000000000000000 00fa
    5555 4e30 20 ffd8ffba000200000001ffffff21ff5c01ae0000000000000000000000000000 afe7
    5555 4e30 20 ffd8ffba000200000001ffffff21ff5c01ae0000000000000000000000000000 afe7
    5555 4e30 20 ffd8ffba000200000001ffffff21ff5c01ae0000000000000000000000000000 afe7
    5555 4e30 20 ffd8ffba000200000001ffffff21ff5c01ae0000000000000000000000000000 afe7
    5555 4e30 20 ffd8ffba000200000001ffffff22ff5c01ae0000000000000000000000000000 dde7
    5555 4e30 20 ffd8ffba000200000000ffffff22ff5c01ae0000000000000000000000000000 b8ec
    5555 4e30 20 ffd8ffba000200000000ffffff22ff5c01ae0000000000000000000000000000 b8ec
    5555 4e30 20 ffd8ffbb000200000000fffeff22ff5c01ae0000000000000000000000000000 9dfc
    5555 4e30 20 ffd8ffbb000100000000fffeff22ff5c01ae0000000000000000000000000000 a256
    5555 4e30 20 ffd8ffbb000100000000fffeff22ff5c01ae0000000000000000000000000000 a256
    5555 4e30 20 ffd8ffba000100000000ffffff22ff5c01ae0000000000000000000000000000 8746
    5555 4e30 20 ffd7ffba000100000000fffeff22ff5c01ae0000000000000000000000000000 f9c0
    5555 4e30 20 ffd7ffba000100000000ffffff22ff5c01ae0000000000000000000000000000 2c36
```

Figure 14: Raw Data Console

8.7 Packet Statistics View

Packet statistics can be obtained from the "View" menu by selecting the "Packet Statistics" option as shown in Figure 15. This view simply provides the user with a short list of vital statistics (including Packet Rate, CRC Failures, and overall Elapsed Time) that are calculated over a one second window. This tool should be used to gather information regarding the overall health of the user configuration. Incorrectly configured communication settings can result in a large number of CRC Failures and poor data transfer.



Packets Received: 146442

CRC Failures: 0

Avg Packet Rate (Hz): 102.90

Elapsed Time: 00:24:23

Figure 15: Packet Statistics

8.8 Unit Configuration

The Unit Configuration window gives the user the ability to view and alter the system settings (see Figure 16). This window is accessed through the "Unit Configuration" menu item under the configuration menu. Under the "General" tab, users have the ability to verify the current configuration by selecting the "Get All Values" button. This button simply provides users with the currently set configuration of the unit and displays the values in the left column of boxes.

There are three tabs within the "Unit Configuration" menu; only the General tab applies for the IMU380ZA-200. The General tab displays some of the most commonly used settings.

To alter a setting, simply select the check box on the left of the value that you wish to modify and then select the value using the drop down menu on the right side. Once you have selected the appropriate value, these settings can be set temporarily or permanently (a software reset or power cycle is required for the changes to take affect) by selecting from the choices at the bottom of the dialog box. Once the settings have been altered a "Success" box will appear at the bottom of the page.

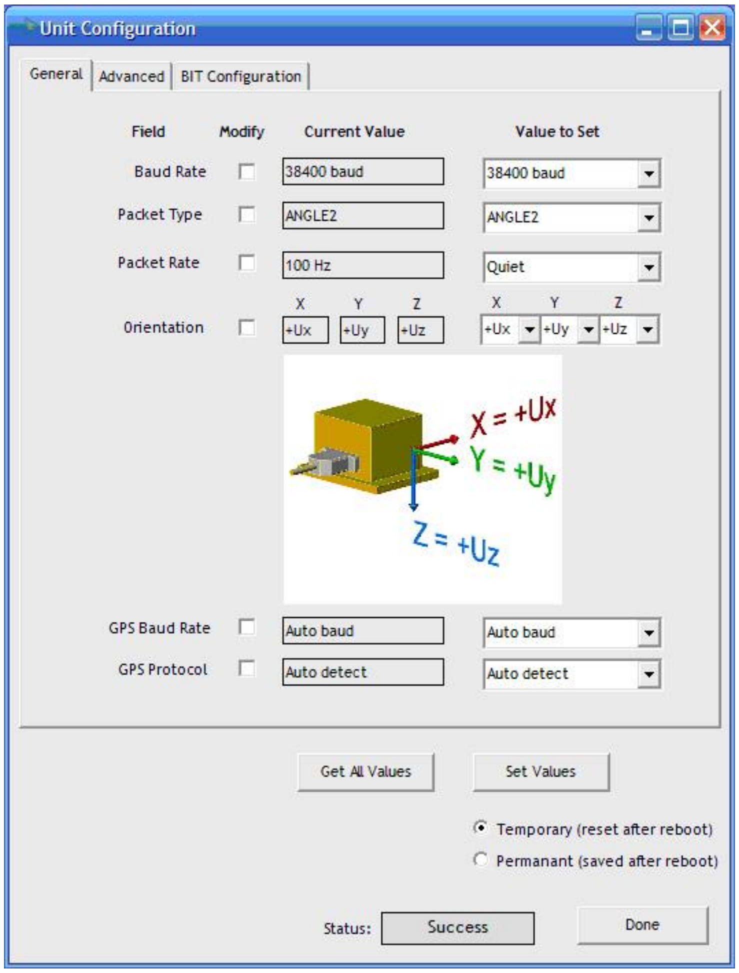
IMPORTANT

Caution must be taken to ensure that the settings selected are compatible with the system that is being configured. In most cases a "FAIL" message will appear if incompatible selections are made by the user, however it is the users responsibility to ensure proper configuration of the unit. Refer to Section 5 for valid IMU380ZA-200 configuration settings.

IMPORTANT

Unit orientation selections must conform to the right hand coordinate system as noted in Section 7.3 of this user manual. Selecting orientations that do not conform to these criteria are not allowed.

Figure 16: Unit Configuration



Field	Modify	Current Value	Value to Set
Baud Rate	☐	38400 baud	38400 baud
Packet Type	☐	ANGLE2	ANGLE2
Packet Rate	☐	100 Hz	Quiet
Orientation	☐	X +Ux, Y +Uy, Z +Uz	X +Ux, Y +Uy, Z +Uz
GPS Baud Rate	☐	Auto baud	Auto baud
GPS Protocol	☐	Auto detect	Auto detect

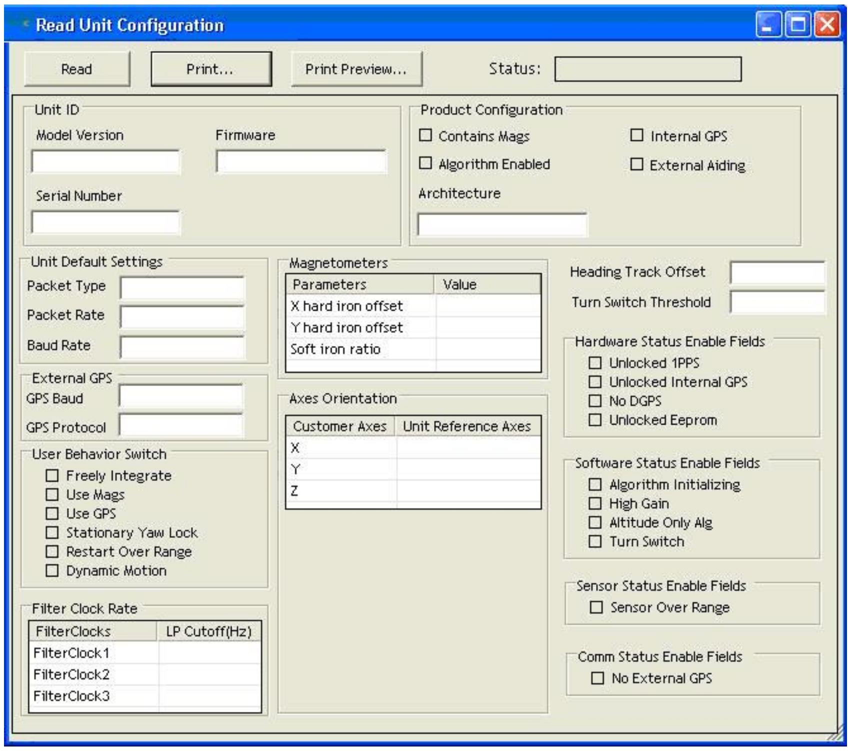
(● Temporary (reset after reboot)) ○ Permanant (saved after reboot)

Status: Success

8.9 Read Unit Configuration

NAV-VIEW 3.X allows users to view the current settings and calibration information for a given IMU380ZA-200 unit by accessing the "Read Configuration" selection from the "Configuration" drop down menu as shown in Figure 17. From this dialog, users can print a copy of the unit's current configuration and calibration values with the click of a button. Simply select the "Read" button at the top of the dialog box and upon completion select the "Print" or "Print Preview" buttons to print a copy to your local network printer. This information can be helpful when storing hard copies of unit configuration, replicating the original data sheet and for troubleshooting if you need to contact MEMSIC's Support Staff. Note that several of the Read Configuration functions and fields will not be applicable to the IMU380ZA-200.

Figure 17: Read Configuration



Unit ID
Model Version
Firmware
Serial Number

Product Configuration
☐ Contains Mags
☐ Internal GPS
☐ Algorithm Enabled
☐ External Aiding
Architecture

Unit Default Settings
Packet Type
Packet Rate
Baud Rate

Magnetometers

Parameters	Value
X hard iron offset	
Y hard iron offset	
Soft iron ratio	

Heading Track Offset
Turn Switch Threshold

External GPS
GPS Baud
GPS Protocol

Axes Orientation

Customer Axes	Unit Reference Axes
X	
Y	
Z	

Hardware Status Enable Fields
☐ Unlocked 1PPS
☐ Unlocked Internal GPS
☐ No DGPS
☐ Unlocked Eeprom

User Behavior Switch
☐ Freely Integrate
☐ Use Mags
☐ Use GPS
☐ Stationary Yaw Lock
☐ Restart Over Range
☐ Dynamic Motion

Software Status Enable Fields
☐ Algorithm Initializing
☐ High Gain
☐ Altitude Only Alg
☐ Turn Switch

Sensor Status Enable Fields
☐ Sensor Over Range

Filter Clock Rate

FilterClocks	LP Cutoff(Hz)
FilterClock1	
FilterClock2	
FilterClock3	

Comm Status Enable Fields
☐ No External GPS

Appendix B: UART Port Sample Packet Parser Code

8.10 Overview

This appendix includes sample code written in ANSI C for parsing packets from data sent by the IMU380ZA-200 over the UART port. This code can be used by a user application reading data directly from the IMU380ZA-200 unit, or perhaps from a log file.

The sample code contains the actual parser, but also several support functions for CRC calculation and circular queue access:

- **process_xbow_packet** – for parsing out packets from a queue. Returns these fields in structure XBOW_PACKET (see below). Checks for CRC errors
- **calcCRC** – for calculating CRC on packets.
- **Initialize** - initialize the queue
- **AddQueue** - add item in front of queue
- **DeleteQueue** - return an item from the queue
- **peekWord** - for retrieving 2-bytes from the queue, without popping
- **peekByte** – for retrieving a byte from the queue without popping
- **Pop** - discard item(s) from queue
- **Size** – returns number of items in queue
- **Empty** – return 1 if queue is empty, 0 if not
- **Full** - return 1 if full, 0 if not full

The parser will parse the queue looking for packets. Once a packet is found and the CRC checks out, the packet's fields are placed in the XBOW_PACKET structure. The parser will then return to the caller. When no packets are found the parser will simply return to the caller with return value 0.

The XBOW_PACKET structure is defined as follows:

```
typedef struct xbow_packet
{
    unsigned short packet_type;
    char           length;
    unsigned short crc;
    char           data[256];
} XBOW_PACKET;
```

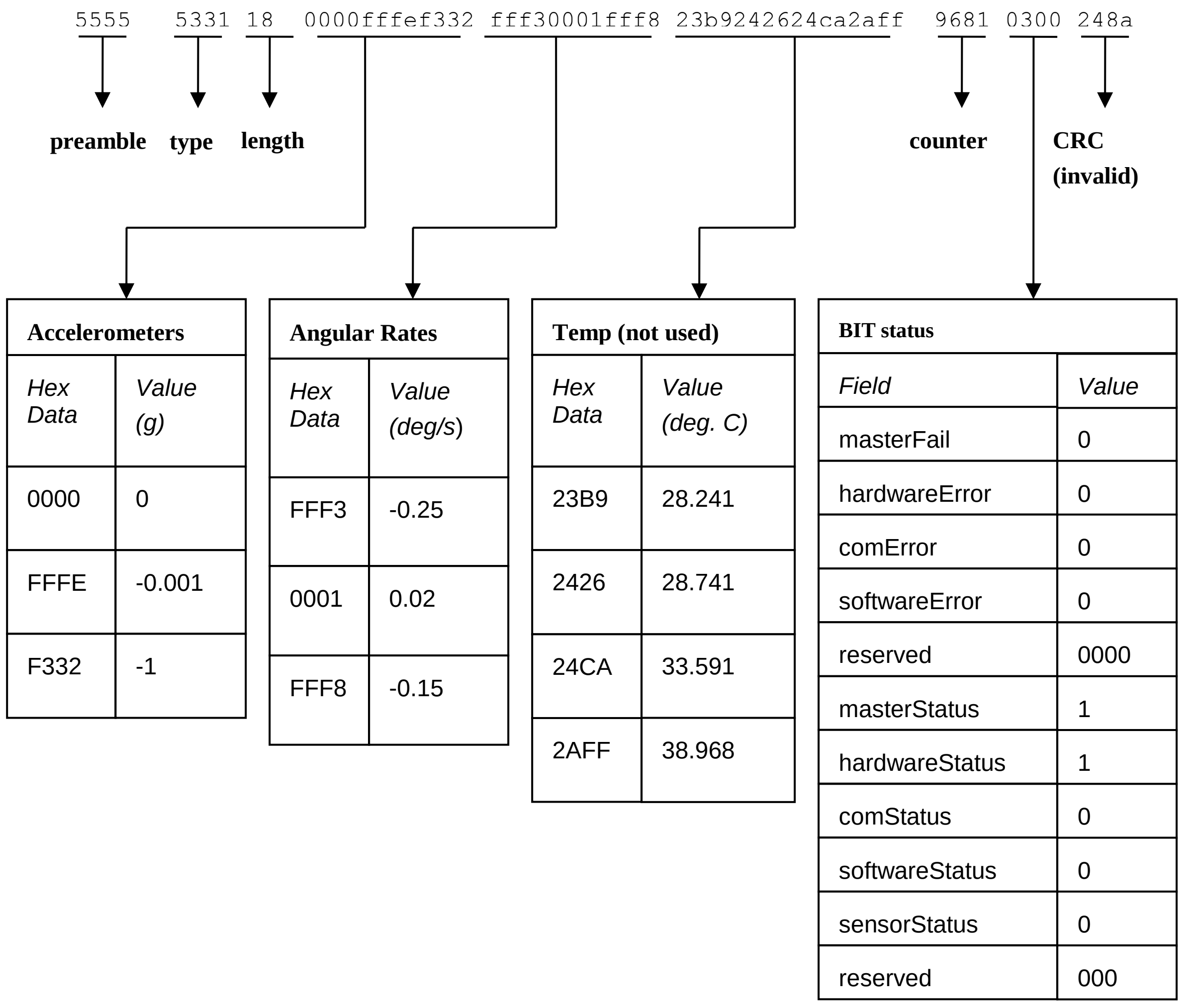
Typically, the parser would be called within a loop in a separate process, or in some time triggered environment, reading the queue looking for packets. A separate process might add data to this queue when it arrives. It is up to the user to ensure circular-queue integrity by using some sort of mutual exclusion mechanism within the queue access functions.

8.11 Code listing

```
#include <stdio.h>

/* buffer size */
#define MAXQUEUE 500

/*
 * circular queue
 */
typedef struct queue_tag
{
    int count;
    int front;
    int rear;
    char entry[MAXQUEUE];
} QUEUE_TYPE;

/*
 * MEMSIC packet
 */
typedef struct xbow_packet
{
    unsigned short packet_type;
    char                        length;
    unsigned short crc;
    char                        data[256];
} XBOW_PACKET;

QUEUE_TYPE circ_buf;

/*******************************************************************************
 * FUNCTION:  process_xbow_packet looks for packets in a queue
 * ARGUMENTS: queue_ptr: is pointer to queue to process
 *                      result: will contain the parsed info when return value is 1
 * RETURNS:    0 when failed.
 *                      1 when successful
 *******************************************************************************/
int process_xbow_packet(QUEUE_TYPE *queue_ptr, XBOW_PACKET *result)
{
    unsigned short myCRC = 0, packetCRC = 0, packet_type = 0, numToPop=0, counter=0;
    char packet[100], tempchar, dataLength;

    if(Empty(queue_ptr))
    {
        return 0;   /* empty buffer */
    }

    /* find header */
    for(numToPop=0; numToPop+1<Size(queue_ptr) ;numToPop+=1)
    {
        if(0x5555==peekWord(queue_ptr, numToPop)) break;
    }

    Pop(queue_ptr, numToPop);
    if(Size(queue_ptr) <= 0)
    {
        /* header was not found */
        return 0;
    }

  /* make sure we can read through minimum length packet */
  if(Size(queue_ptr)<7)
  {
    return 0;
  }

  /* get data length (5th byte of packet) */
  dataLength = peekByte(queue_ptr, 4);

  /* make sure we can read through entire packet */
  if(Size(queue_ptr) < 7+dataLength)
  {
    return 0;
    }


    /* check CRC */
    myCRC = calcCRC(queue_ptr, 2,dataLength+3);
    packetCRC = peekWord(queue_ptr, dataLength+5);

    if(myCRC != packetCRC)
    {
        /* bad CRC on packet – remove the bad packet from the queue and return */
        Pop(queue_ptr, dataLength+7);
        return 0;
    }

    /* fill out result of parsing in structure */
    result->packet_type = peekWord(queue_ptr, 2);
    result->length      = peekByte(queue_ptr, 4);
    result->crc         = packetCRC;
    for(counter=0; counter < result->length; counter++)
    {
        result->data[counter] = peekByte(queue_ptr, 5+counter);
    }

    Pop(queue_ptr, dataLength+7);

    return 1;
}

/*******************************************************************************
 * FUNCTION:  calcCRC calculates a 2-byte CRC on serial data using
 *            CRC-CCITT 16-bit standard maintained by the ITU
 *                     (International Telecommunications Union).
 * ARGUMENTS: queue_ptr is pointer to queue holding area to be CRCed
 *                     startIndex is offset into buffer where to begin CRC calculation
 *                     num is offset into buffer where to stop CRC calculation
 * RETURNS:    2-byte CRC
 ******************************************************************************/
unsigned short calcCRC(QUEUE_TYPE *queue_ptr, unsigned int startIndex, unsigned int num) {
    unsigned int i=0, j=0;
    unsigned short crc=0x1D0F; //non-augmented inital value equivalent to augmented initial value 0xFFFF

    for (i=0; i<num; i+=1) {
        crc ^= peekByte(queue_ptr, startIndex+i) << 8;

        for(j=0;j<8;j+=1) {
            if(crc & 0x8000) crc = (crc << 1) ^ 0x1021;
            else crc = crc << 1;
        }
    }
    return crc;
}


/*******************************************************************************
 * FUNCTION:  Initialize - initialize the queue
 * ARGUMENTS: queue_ptr is pointer to the queue
 ******************************************************************************/
void Initialize(QUEUE_TYPE *queue_ptr)
{
    queue_ptr->count =  0;
    queue_ptr->front =  0;
    queue_ptr->rear  = -1;
}

/*******************************************************************************
 * FUNCTION:  AddQueue - add item in front of queue
 * ARGUMENTS: item holds item to be added to queue
 *                    queue_ptr is pointer to the queue
 * RETURNS:    returns 0 if queue is full. 1 if successful
 ******************************************************************************/
int AddQueue(char item, QUEUE_TYPE *queue_ptr)
{
    int retval = 0;
    if(queue_ptr->count >= MAXQUEUE)
    {
        retval = 0; /* queue is full */
    }
    else
    {
        queue_ptr->count++;
        queue_ptr->rear = (queue_ptr->rear + 1) % MAXQUEUE;
        queue_ptr->entry[queue_ptr->rear] = item;
        retval = 1;
    }
    return retval;
}

/*******************************************************************************
 * FUNCTION:  DeleteQeue - return an item from the queue
 * ARGUMENTS: item will hold item popped from queue
 *                    queue_ptr is pointer to the queue
 * RETURNS:    returns 0 if queue is empty. 1 if successful
 *****************************************************************************/
int DeleteQueue(char *item, QUEUE_TYPE *queue_ptr)
{
    int retval = 0;
    if(queue_ptr->count <= 0)
    {
        retval = 0; /* queue is empty */
    }
    else
    {
        queue_ptr -> count--;
        *item = queue_ptr->entry[queue_ptr->front];
        queue_ptr->front = (queue_ptr->front+1) % MAXQUEUE;
        retval=1;
    }
    return retval;
}

/*****************************************************************************
 * FUNCTION:  peekByte returns 1 byte from buffer without popping
 * ARGUMENTS: queue_ptr is pointer to the queue to return byte from
 *                    index is offset into buffer to which byte to return
 * RETURNS:    1 byte
 * REMARKS:    does not do boundary checking. please do this first
 *****************************************************************************/
char peekByte(QUEUE_TYPE *queue_ptr, unsigned int index) {
    char byte;
    int firstIndex;

    firstIndex = (queue_ptr->front + index) % MAXQUEUE;

    byte = queue_ptr->entry[firstIndex];
    return byte;
}


/*****************************************************************************
 * FUNCTION:  peekWord returns 2-byte word from buffer without popping
 * ARGUMENTS: queue_ptr is pointer to the queue to return word from
 *                    index is offset into buffer to which word to return
 * RETURNS:    2-byte word
 * REMARKS:    does not do boundary checking. please do this first
 *****************************************************************************/
unsigned short peekWord(QUEUE_TYPE *queue_ptr, unsigned int index) {
    unsigned short word, firstIndex, secondIndex;

    firstIndex = (queue_ptr->front + index) % MAXQUEUE;
    secondIndex = (queue_ptr->front + index + 1) % MAXQUEUE;

    word = (queue_ptr->entry[firstIndex] << 8) & 0xFF00;
    word |= (0x00FF & queue_ptr->entry[secondIndex]);
    return word;
}

/*****************************************************************************
 * FUNCTION:  Pop - discard item(s) from queue
 * ARGUMENTS: queue_ptr is pointer to the queue
 *                     numToPop is number of items to discard
 * RETURNS:    return the number of items discarded
 ******************************************************************************/
int Pop(QUEUE_TYPE *queue_ptr, int numToPop)
{
    int i=0;
    char tempchar;
    for(i=0; i<numToPop; i++)
    {
        if(!DeleteQueue(&tempchar, queue_ptr))
        {
            break;
        }
    }
    return i;
}

/******************************************************************************
 * FUNCTION:  Size
 * ARGUMENTS: queue_ptr is pointer to the queue
 * RETURNS:    return the number of items in the queue
 ******************************************************************************/
int Size(QUEUE_TYPE *queue_ptr)
{
    return queue_ptr->count;
}

/******************************************************************************
 * FUNCTION:  Empty
 * ARGUMENTS: queue_ptr is pointer to the queue
 * RETURNS:    return 1 if empty, 0 if not
 ******************************************************************************/
int Empty(QUEUE_TYPE *queue_ptr)
{
    return queue_ptr->count <= 0;
}


/******************************************************************************
 * FUNCTION:  Full
 * ARGUMENTS: queue_ptr is pointer to the queue
 * RETURNS:    return 1 if full, 0 if not full
 ******************************************************************************/
int Full(QUEUE_TYPE *queue_ptr)
{
    return queue_ptr->count >= MAXQUEUE;
}
```

Appendix C: UART Port Sample Packet Decoding

Figure 18: Example payload from Scaled Sensor 1 data packet (S1)



Accelerometers

Hex Data	*Value (g)*
0000	0
FFFE	-0.001
F332	-1

Angular Rates

Hex Data	*Value (deg/s)*
FFF3	-0.25
0001	0.02
FFF8	-0.15

Temp (not used)

Hex Data	*Value (deg. C)*
23B9	28.241
2426	28.741
24CA	33.591
2AFF	38.968

BIT status

Field	*Value*
masterFail	0
hardwareError	0
comError	0
softwareError	0
reserved	0000
masterStatus	1
hardwareStatus	1
comStatus	0
softwareStatus	0
sensorStatus	0
reserved	000

1759 McCarthy Blvd.
Milpitas, CA 95035
Phone: 408.964.9700
Fax: 408.854.7702
Website: www.memsic.com
Email: infoca@memsic.com