IMU383 SERIES USER MANUAL

Document Part Number: 7430-1398-01

WARNING

This product has been developed by ACEINNA exclusively for commercial applications. It has not been tested for, and ACEINNA 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 ACEINNA and without obtaining appropriate US export license(s), when required by US law. Diversion contrary US law is prohibited.

ACEINNA, Inc.

email: info@aceinna.com, website: www.aceinna.com

Revision History

Date	Document Revision	Firmware Applicability	Description	Author(s)
Aug 22, 2019	Rev. A	v1.1.x	Draft release of IMU383 manual	Joseph Motyka
Dec. 9, 2019	Rev. B	v1.1.3	Updates to manual to describe fault detection and settings	Joseph Motyka
Dec. 18, 2019	Rev. C	v1.1.4	Fixed links to and formatting of tables and figures.	Joseph Motyka

Table of Contents

About this Manual

The following symbols are used to provide additional information.

◀ NOTE

Note provides additional information about the topic.

☑ EXAMPLE

This symbol indicates an example that will 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.

This manual uses the following paragraph heading formats:

1 Heading 1

1.1 Heading 2

1.1.1 Heading 3

1.1.1.1 Heading 4

Normal

1 Introduction

1.1 Manual Overview

This manual describes ACEINNA's IMU383 IMU Inertial Measurement Unit. The IMU383 will be referred to as "the unit" frequently in this document.

For users wishing to get started quickly, please refer to the two-page quick start guide included with each evaluation kit shipment. Table 1 highlights the content in each section and suggests how to use this manual.

Table 1 Manual Content

Manual Section	Who Should Read?
Section 1: Manual Overview	All customers should read sections 1.1 and 1.2.
Section 2: Interface	Customers designing the electrical and mechanical interface to the IMU383 series products should read Section 2.
Section 3: Theory of Operation	All customers should read Section 3.
Section 4: SPI Port Interface	Customers designing the software interface to the IMU383 series products SPI Port should review Section 4.
Section 5-8: UART Port Interface	Customers designing the software interface to the IMU383 series products UART Port should review Sections 5-8.
Section 9: IMU383 Bootloader	Customers that intend to update Aceinna firmware in the unit should read Section 9.
Section 10: Warranty and Support Information	All customers should read Section 10.
Appendix A: Installation and Operation of NAV-VIEW	All customers should read Appendix A.
Appendix B: Sample Packet-Parser Code	All customers that intend to create scripts to interface with serial messages should read Appendix B
Appendix C: Sample Packet Decoding	All customers that intend to create scripts to interface with serial messages should read Appendix C

1.2 Overview of the IMU383 Series Inertial Measurement Unit

This manual describes the use of ACEINNA's IMU383 and is intended to be used as a detailed technical reference and operating guide. ACEINNA's IMU383 Series products combine the latest in high-performance commercial MEMS (Micro-electromechanical Systems) sensors and digital signal processing techniques to provide a small, cost-effective alternative to existing IMU systems.

The IMU383 Series is ACEINNA's fifth generation of MEMS-based Inertial Systems, building on over a decade of field experience, and encompassing thousands of deployed units and millions of operational hours in a wide range of land, marine, airborne, and instrumentation applications. It is designed for OEM applications.

At the core of the IMU383 Series is a trio of rugged 6-DOF (Degrees of Freedom) MEMS inertial sensor clusters. Each 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 for temperature and non-linearity effects during ACEINNA's manufacturing and test process using automated thermal chambers and rate tables.

The differentiating feature of the IMU383 Series is the trio of redundant 6-DOF MEMS sensor clusters. This redundancy has two direct benefits:

1) Combining multiple sensor reading improves the noise characteristics of the output signal
2) Using more than one sensor enables the unit to operate through a single sensor-chip failure by detecting and voting out the failed part. Failures include stuck or railed readings as well as sustained inconsistency between the three sensor sets.

Another unique feature of the IMU383 Series is the extensive field configurability of the units. This field configurability allows the IMU383 Series of Inertial Systems to satisfy a wide range of applications and performance requirements with a single mass produced hardware platform. The basic configurability includes parameters such as baud rate (UART), clock speed (SPI), packet type, and update rate, and the advanced configurability includes the defining of custom axes.

The IMU383 Series module is packaged in a lightweight, rugged, unsealed metal enclosure that is designed for cost-sensitive commercial and OEM applications. The unit 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. ACEINNA's NAV-VIEW 3.X Windows application supports using the unit low level UART data port NAV-VIEW 3.X is a powerful Windows-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 unit main connector is a SAMTEC FTM-110-02-F-DV-P defined in Figure 1. The SAMTEC CLM-110-02 mating connector is compatible with the IMU383unit main connector.

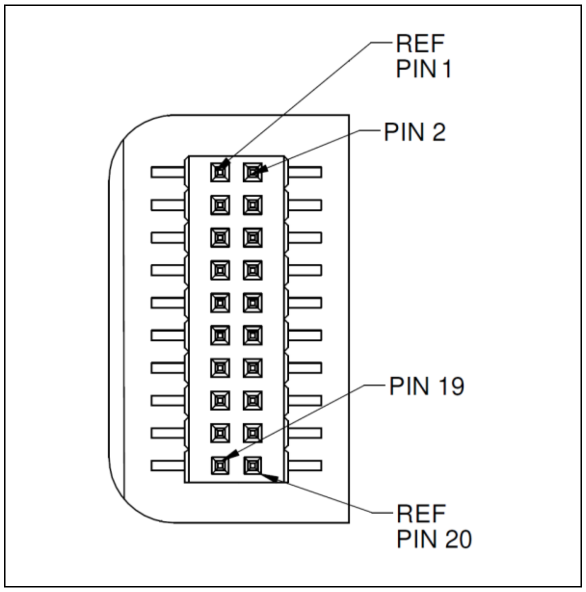


Figure 1 IMU383 Interface Connector

Table 2 Interface Connector Pin Definition IMU383

Pin	**Pin Description (IMU383)**
1	Inertial-Sensor Sampling Indicator (sampling upon falling edge)
2	Synchronization Input (1 kHz Pulse used to synchronize SPI Comm). If used, this clock should be 1kHz +/- 0.1%
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	External Reset (NRST)
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
18	Reserved – factory use only
19	Reserved
20	Reserved – factory use only

2.1.2 Power Input and Power Input Ground

Power is applied to the unit 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. Do not add greater than 5.5 volts on the power pins or damage may occur. This system has no reverse voltage or over-voltage protection.

2.1.3 Serial Data Interface

The user can select the serial interface used with the unit by controlling the logic level on connector pin 7 at system startup. If pin 7 is left floating then the unit is configured for SPI communications on pins 3-6. Pin 7 is set as an output and used as the DATA READY discrete for SPI communications. Additionally, the user can 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 unit 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 5-8. 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.

2.1.3.1 SPI Com Synchronization Input

Pin 2 can be used as a sync pulse to force synchronization of sensor data collection to a 1 kHz rising-edge signal for output over the SPI port. See Section 4.10 for a more complete description.

2.1.4 *Reserved – Factory Use Only*

During normal operation of the unit 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 unit to operate properly.

2.2 Mechanical Interface

The IMU383 mechanical interface is defined by the outline drawing in Figure 2.

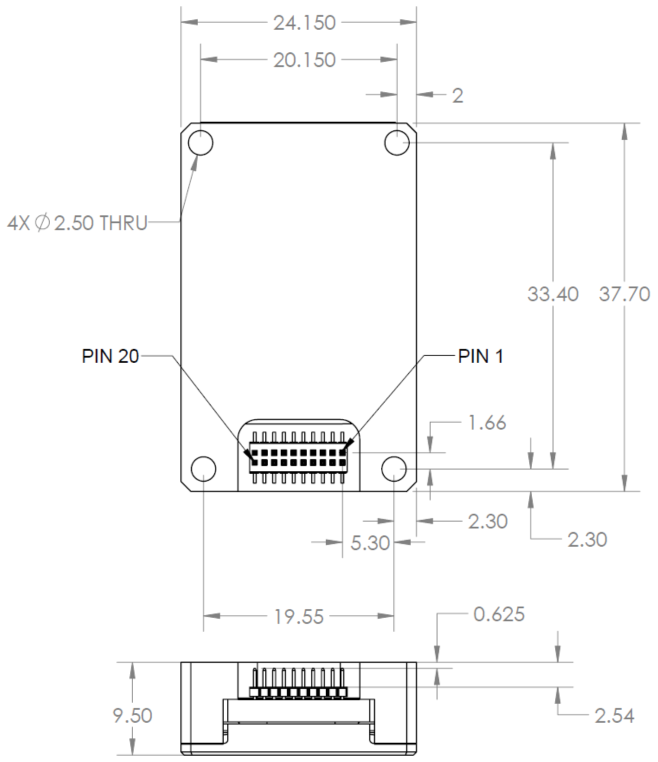


Figure 2 IMU383 Outline Drawing

NOTES UNLESS OTHERWISE STATED:

1) MATING CONNECTOR SAMTEC CLM-110-02
2) DIMENSION TO CENTROID OF PIN ONE

3 Theory of Operation

This section of the manual covers detailed theory of operation for the IMU383

Figure 3 shows the IMU383 Series hardware block diagram. At the core of the IMU383 Series is a rugged 6-DOF (Degrees of Freedom) MEMS inertial sensor cluster. 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 ACEINNA'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 rate gyro sensor bias shifts over temperature (-40 ^{0}C to +71 ^{0}C) are compensated and verified using calibrated thermal chambers and rate tables.

The 6-DOF inertial sensor cluster data is fed into a high speed signal processing chain, which provides the sensor compensation, digital filtering, and sensor fault detection. 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 5-8.

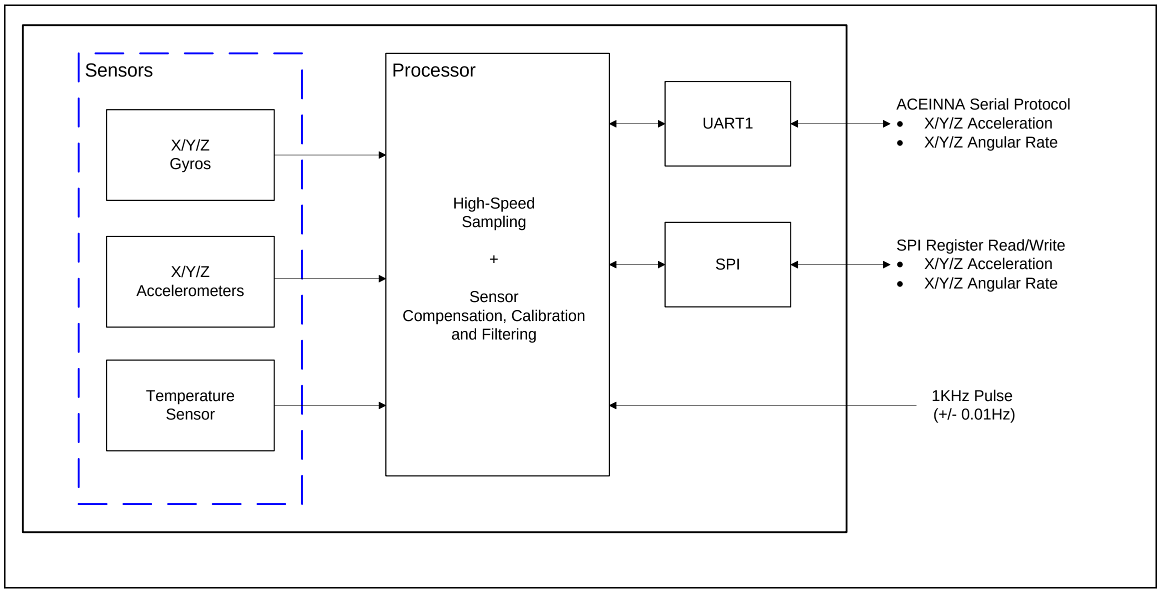


Figure 3 IMU383 Series Hardware Block Diagram

Figure 4 shows the software block diagram. The 6-DOF inertial sensor cluster data is fed into a high speed 200Hz signal processing chain. These 6-DOF signals pass through one or more of the processing blocks and these signals are converted into output measurement data as shown. Measurement data packets are available at fixed continuous output rates or on a polled basis.

As shown in the software block diagram, the IMU383 Series has a unit setting and profile block which configures the algorithm to user and application specific needs. This feature is one of the more powerful features in the IMU383 Series architecture as it allows the IMU383 Series to work in a wide range of commercial applications by settings different modes of operation for the IMU383 Series.

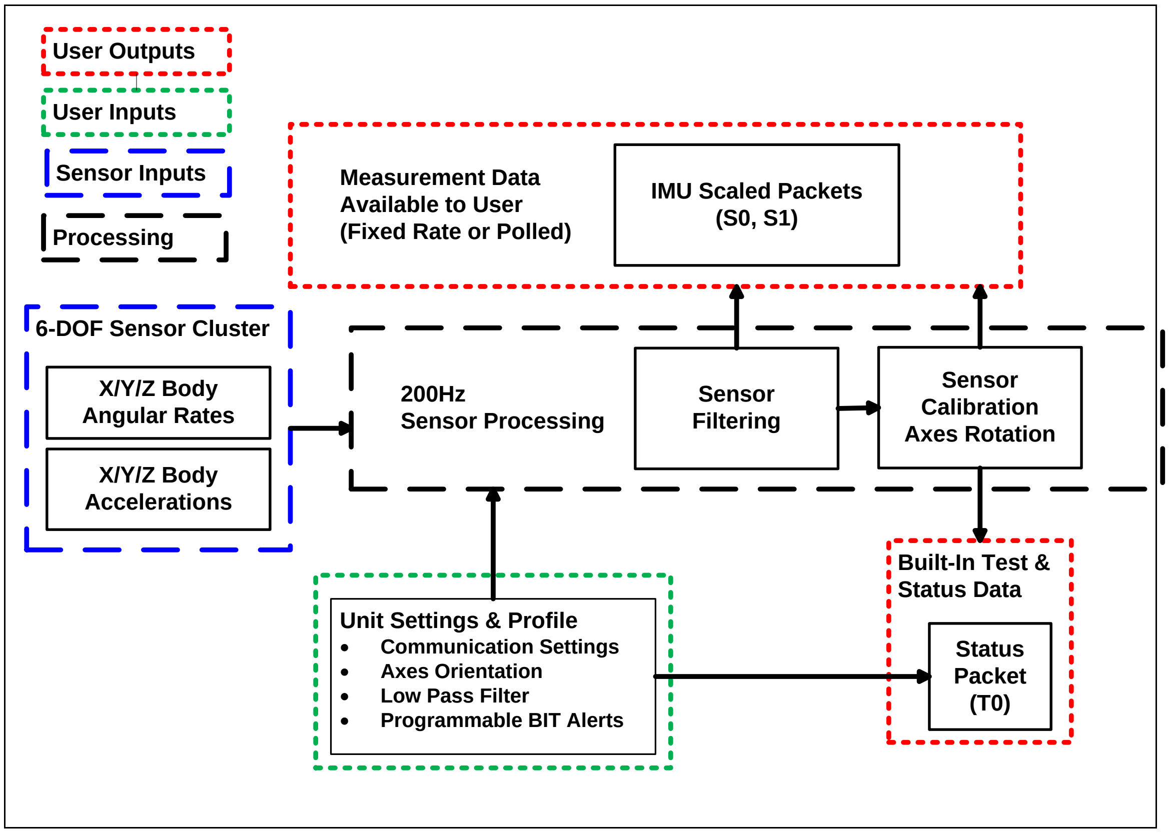


Figure 4 IMU383 Series Software Block Diagram

3.1 IMU383 Series Default Coordinate System

The IMU383 Series Inertial System default coordinate systems are shown in Figure 5. As with many elements of the IMU383 Series, the coordinate system is configurable with either NAV-VIEW or by sending the appropriate serial commands over the UART port. These configurable elements are known as ***Advanced Settings***. This section of the manual describes the default coordinate system settings of the IMU383 Series when it leaves the factory.

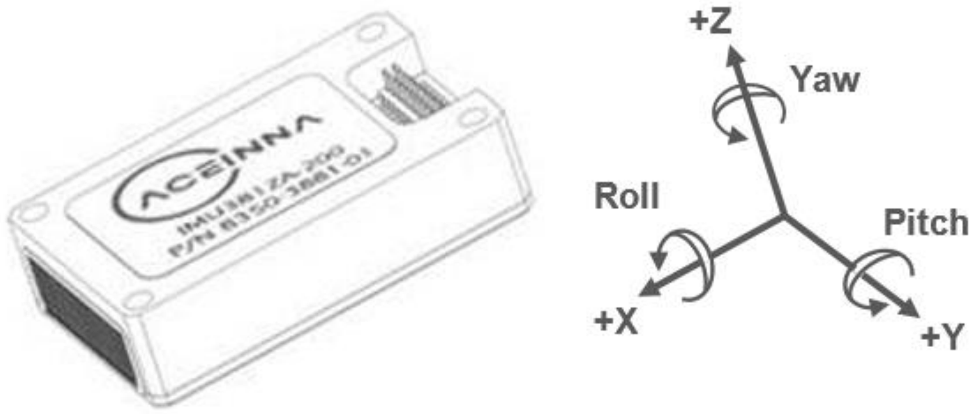


Figure 5 IMU383 Default Coordinate Frame

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, with a IMU383 Series product sitting on a level table, it 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 IMU383 Series 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 IMU383 Series product is sitting on a level surface and you rotate it clockwise on that surface, this will be a positive rotation around the z-axis. The x- and y-axis rate sensors would measure zero angular rates, and the z-axis sensor would measure a positive angular rate.

3.1.1 Advanced Settings

The IMU383 Series Inertial Navigation Units have a number of advanced settings that can be changed. All units support baud rate[1], power-up output packet type, output rate, sensor low pass

[1] Note: certain combinations of baud-rate, packet-type, and output data rate are invalid because the time to transmit the data exceeds a limit on the permissible message length. The IMU383 limits the output packet width to 80% of the time between data packets. For instance, if the packet is output every 10 milliseconds (100 Hz) then the packet width must be less than 8 milliseconds or the combination is not allowed. This prevents messages from overlapping and causing communication problems. For this reason, 57.6 kbps and higher baud-rates are suggested.

filtering, and custom axes configuration. The units can be configured using NAV-VIEW, as described in Appendix A, or directly with serial commands as described in Sections 5-8.

3.2 IMU383 Theory of Operation

The unit provides inertial rate and acceleration data in 6-DOF (six degrees of freedom). The unit signal processing chain consists of the 6-DOF sensor cluster, programmable low-pass filters, and a signal processing component for sensor error compensation. The unit has a UART input/output port and a SPI input/output port.

After passing through a digitally controlled programmable low-pass filter, the rate and acceleration sensor signals are obtained at 200 Hz. The factory calibration data, stored in EEPROM, is used by the processor to remove temperature bias, misalignment, scale factor errors, and non-linearities from the sensor data. Additionally any advanced user settings such as axes rotation are applied to the IMU data. Finally, sensor fault detection is performed on the sensor signals and the individual sensor signals are combined to form a signal with reduced noise characteristics.

The 200Hz IMU data is continuously maintained inside the unit, and is available at 200Hz on the SPI output port registers. Digital IMU data is output over the UART port at a selectable fixed rate (100, 50, 25, 20, 10, 5 or 2 Hz) or on as-requested basis using the GP, 'Get Packet' command. The digital IMU data is available in several measurement packet formats including Scaled Sensor Data ('S1' Packet). In the Scaled Sensor Data ('S1' Packet), data is output in scaled engineering units. See Section 6 of the manual for full packet descriptions.

3.2.1 Sensor Fault Detection:

New for the IMU383 is the incorporation of triple-redundant accelerometer and gyro sensors, which enables sensor fault detection. The fault detection routine incorporated into the firmware continually monitors the output of the three sensor chips. If an outlier is detected in the output, the routine continues to monitor the sensors until the part is deemed failed. The Fault Tolerant Time Interval (FTTI) for such a failure is set to 300 msec, however the actual detection time could be changed if needed. Contact the factory for more information.

Configuration Field Settings for Sensor Enable and Sensor Select:

The same mechanism that enables the firmware to vote out a failed or underperforming sensor also enables the user to turn on or off a sensor chip or exclude an enabled sensor from the output, if desired (Section 7.1). When communicating with the device using UART, field 0x42 controls the sensor enable function while field 0x43 controls which sensors are included in the combined sensor output (SPI communication protocol uses SPI registers 0x1A, 0x1B, and 0x1C to define which sensors are active, as described in Sections 4.1 and 4.8.7). . Nominally both fields are set to 0x7, representing the bit pattern *111* (a one indicates *enabled* while a zero indicates *disabled*). In this bit pattern, the least-significant bit (the bit on the far right) controls sensor chip 1, the most-significant bit (on the far left) controls sensor chip 3, and the middle bit controls sensor 2. Both fields tells the firmware which sensors to include in the combined sensor output. The combination is done by averaging the readings from the selected sensors.

Field 0x42 and 0x43 are set via the set-field (SF) or write-field (WF) commands (described in sections 7.7). In the case of the Sensor Enable Setting, the write-field command must be used as the part must be reset (either via power-cycling the unit of issuing a software-reset command) in order for the setting to take effect.

When a fault is detected, the system BIT (Section 8.5) will indicate that a fault related to the sensor fault detection has occurred. This field will not indicate which sensor was removed from the triad; Fields 0x42 and 0x43 must be read to determine which of the three sensors was removed from the combination.

Field 0x42: Sensor Enable Setting

Value	Sensor Enable Flag			Description
	Sensor #3	Sensor #2	Sensor #1	
0x0	0	0	0	All sensors disabled
0x1	0	0	1	Only sensor #1 enabled
0x2	0	1	0	Only sensor #2 enabled
0x3	0	1	1	Sensors #1 and #2 enabled
0x4	1	0	0	Only sensor #3 enabled
0x5	1	0	1	Sensors #1 and #3 enabled
0x6	1	1	0	Sensors #2 and #3 enabled
0x7	1	1	1	All sensors enabled

Field 0x43: Sensor Included in Output

Value	Sensor Inclusion Flag			Description
	Sensor #3	Sensor #2	Sensor #1	
0x0	0	0	0	No sensors included (output 0.0)
0x1	0	0	1	Only sensor #1 included
0x2	0	1	0	Only sensor #2 included
0x3	0	1	1	Sensors #1 and #2 included
0x4	1	0	0	Only sensor #3 included
0x5	1	0	1	Sensors #1 and #3 included
0x6	1	1	0	Sensors #2 and #3 included
0x7	1	1	1	All sensors included in output

User Settings:

Selecting which sensors are included in the output can be accomplished using either the write-field or set-field command (WF or SF) over UART. Using the SF command will cause the setting to take effect immediately; the WF command will require a unit-reset before the output is modified. Several example commands follow:

To get the values for fields 0x42 and 0x43 stored in RAM, use the Get-Field (GF) command:

```
5555 4746 05 02 0042 0043 a0d0
```

To set the value for field 0x43 stored in RAM to 0x1 (output sensor #1 only), use the Set-Field (SF) command:

```
5555 5346 05 01 0043 0001 236d
```

To change the value for field 0x42 in EEPROM to 0x1 (enable sensor #1 only), use the Write-Field (WF) command:

5555 5746 05 01 0042 0001 1b30

Section 4 describes how to set these fields via the SPI interface.

3.2.2 IMU383 Advanced Settings

The unit advanced settings are described in Table 3. All of the advanced settings are accessible through NAV-VIEW under the Configuration Menu, Unit Configuration settings. For a full definition of the SPI port please see section 4.

Table 3 IMU383 Advanced Settings

Setting	Default	Comments
Baud Rate	38,400 baud	57600, 115200, and 230400 also available
Packet Type	S0	S1 also available
Packet Rate	100Hz	This setting sets the rate at which selected Packet Type, packets are output. If polled mode is desired, then select Quiet. If Quiet is selected, the unit 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 will show the corresponding image of the unit, so it easy to visualize the mode of operation. Refer to Section 7.4 Orientation Field settings for the twenty four possible orientation settings.
Filter Settings (unfiltered, 2, 5, 10, 20, 25, 40 50 Hz).	20 Hz	The low pass filters are set to a default of 20 Hz for the accelerometers, and 20 Hz for the angular rate sensors. There is one filter setting for all three angular rate sensors. There is one filter setting for all three accelerometers. Setting either to zero disables the low-pass filter.
BIT		See section 8.1

NOTE on Filter Settings

Generally, there is no reason to change the low-pass filter settings on the unit or other IMU383 Series Inertial Systems. However, when an IMU383 Series product is installed in an environment with a lot of vibration, it can be helpful to reduce the vibration-based signal energy and noise prior to further processing on the signal. Installing the unit in the target environment and reviewing the data with NAV-VIEW can be helpful to determine if changing the filter settings would be helpful. Although the filter settings can be helpful in reducing vibration based noise in the signal, low filter settings (e.g., 5Hz) also reduce the bandwidth of the signal, which can wash out the signals containing the dynamics of a target. Therefore, caution should be used when changing the filter settings.

3.2.3 IMU383 Built-In Test

The unit Built-In Test capability allows users of the unit to monitor health, diagnostic, and system status information of the unit in real-time. The Built-In Test information consists of a BIT word (2 bytes) transmitted in every measurement packet. In addition, there is a diagnostic packet 'T0' that can be requested via the Get Packet 'GP' command which contains a complete set of status for each hardware and software subsystem in the unit. See Sections 5-8 for details on the 'T0' packet.

The BIT word, which is contained within each measurement packet, is detailed below. The LSB (Least Significant Bit) is the Error byte, and the MSB (Most Significant Bit) is a Status byte with programmable alerts. Internal health and status are monitored and communicated in both hardware and software. The ultimate indication of a fatal problem is the masterFail flag.

The masterStatus flag is a configurable indication that can be modified by the user. This flag is asserted as a result of any asserted alert signals which have been enabled. See Advanced BIT (Section 8) for details regarding the configuration of the masterStatus flags. Table 4 shows the BIT definition and default settings for BIT programmable alerts in the unit.

Table 4 IMU383 Default BIT Status Definition

BITstatus Field	***Bits***	***Meaning***	***Category***
masterFail	0	0 = normal, 1 = fatal error has occurred	BIT
HardwareError	1	0 = normal, 1= internal hardware error	BIT
comError	2	0 = normal, 1 = communication error	BIT
softwareError	3	0 = normal, 1 = internal software error	BIT
Reserved	4:7	N/A	N/A
masterStatus	8	0 = nominal, 1 = Alert, Sensor Over Range	Status
hardwareStatus	9	Disabled	Status
comStatus	10	Disabled	Status
softwareStatus	11	Disabled	Status
sensorStatus	12	0 = nominal, 1 = Sensor Over Range	Status
Reserved	13:15	N/A	N/A

The unit also allows a user to configure the Status byte within the BIT message. To configure the word, select the BIT Configuration tab from the Unit Configuration menu. The dialog box allows selection of which status types to enable (hardware, software, sensor, and comm). In the case of the unit, which has fewer features and options than other Acienna measurement products, the only meaningful parameter is sensor over-range. It is recommended that users leave the default configuration, which is sensorStatus enabled and flag on sensor over-range. The over-range only applies to the rotational rate sensors. Because instantaneous acceleration levels due to vibration can exceed the accelerometer sensor range in many applications, none of the IMU383 Series products trigger over-range on accelerometer readings.

4 IMU383 SPI Port Interface Definition

The IMU383 provides a SPI interface for data communications. This section of the user's manual defines the unit register map, register control capabilities, and the data register reading and writing methodologies.

The unit operates as a slave device. The master device must be configured to communicate with the unit using the following settings:

- Data transferred in 16-bit word-length and MSB-first
- $f_{CLK} \leq 1.2$ MHz
- CPOL = 1 (clock polarity) and CPHA = 1 (clock phase)

Additional operational requirements are described in Section 4.9.

4.1 IMU383 Register Map

Table 5 describes the IMU383 register map. In the following tables the phrasing "<address1> to <address2>" means all addresses in the inclusive range from <address1> through <address2> and the phrasing "<read-address>/<write-address>" means that the register is read from the <read-address> and written to the <write-address>.

NOTE: Sensor data scaling is specific to the message or register being read. Please see scaling information for each sensor data read operation. Specifically, the sensor data in the SPI S0_BURST, the SPI S1_BURST, the UART S0 message and the UART S1 message are scaled differently than the scaling for reading the individual SPI sensor data registers.

Table 5 IMU383 Register Map[2]

Name	Read/Write	Address	Default	Function
Reserved	N/A	0x00	N/A	N/A
Fault detection enable		0x01		Enabling of fault detection mechanism functions. See section 4.8.8
Fault detection disable		0x02		Disabling of fault detection mechanism functions. See section 4.8.8
Fault detect key		0x03		Key to enable changing of fault detect algorithm settings. See section 4.8.8
Reserved	N/A	0x04 to 0x19	N/A	N/A
Sensor control, chip 1	R/W	0x1A	0xFF (All ON)	Control of active sensors for Sensor Chip 1
Sensor control, chip2	R/W	0x1B		Control of active sensors for Sensor Chip 2
Sensor control, chip3	R/W	0x1C		Control of active sensors for Sensor Chip 3
Sensors status, chip1	R	0x1D	0x00 (All OK)	Status of sensors on chip 1
Sensors status, chip2	R	0x1E		Status of sensors on chip 2

[2] Register and data-packet availability is based on the features of the DMU383ZA..

Name	Read/Write	Address	Default	Function
Sensors status, chip3	R	0x1F		Status of sensors on chip 3
Reserved	N/A	0x20 to 0x33	N/A	
DATA_READY	R/W	0x34	0x04	See Table 14
OUTPUT_DATA_RATE	R/W	0x37	0x01 (200Hz)	See Table 15
LOW_PASS_FILTER	R/W	0x38	0x06 (5Hz)	
RS_DYNAMIC_RANGE	R/W	0x39	0x02 (125)	
Reserved	N/A	0x3A to 0x3C	N/A	
STATUS	R	0x3C		
STNDRD_BURST	R	0x3E		Command to perform a burst-read of the standard data-packet
EXTENDED_BURST	R	0x3F		Command to perform a burst-read of the extended data-packet
Reserved	R	0x40 to 0x51		
MANUF_CODE	R	0x52	Varies	Manufacturing code indicating year and location
MMANUF_LOCATION	R	0x53	Varies	Manufacturing code indicating location
UNIT_CODE	R	0x54	0x0000	Unit information code
PRODUCT_ID	R	0x56	0x38	Product identification code (High byte)
PRODUCT_ID	R	0x57	0x13	Product identification code (Low byte)
SERIAL_NUMBER	R	0x58	Varies	Serial number
Reserved	N/A	0x59 – 0x73	N/A	
ORIENTATION_MSB	R/W	0x74	0x00	See Table 30 for valid orientation settings. The orientation register must be written in order (MSB followed by LSB) for write to take effect.
ORIENTATION_LSB	R/W	0x75	0x00	
EEPROM_WRITE	W	0x76	N/A	See Section 4.8.5
BOOT_STATUS	R	0x78	0x06AB or 0x46AB	Normal mode Boot Mode
HW_VERSION	R	0x7E	0x00	See Section 4.8.7
SW_VERSION	R	0x7F	0x00	See Section 4.8.7

4.2 IMU383 SPI Register Read Methodology

The IMU383 SPI port uses registers to store information such as:

- Sensor data
- Configuration/Status information

A SPI master accesses information via the SPI bus in one of two ways:

- Polled-Mode
- Burst-Mode

In polled-mode, the IMU383 transfers information from any register back to the master in two (or more) SPI cycles[3]. In Burst-Mode, the IMU383 transfers predefined blocks of data in one contiguous group of nine to twenty SPI cycles.

4.2.1 IMU383 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, comprised of the register-address byte and a zero-byte, across the MOSI line. For example, to request the unit's serial number, stored in register 0x58, the master sends the command 0x5800. The unit returns information from this address across the MISO line during the following 16 clock-cycles.

Subsequent SPI-master commands sent to the unit consist of either:

- Sixteen zero-bits (0x0000) to complete the read of a single register.
- The address of another register followed by a zero-byte. This permits back-to-back reads of data-registers.

Single-Register Polled-Read

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.

In this example, 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[4]. This is followed by 16 zero-bits to complete the SPI data-transfer cycle.

As the master transmits the read command over MOSI, the unit transmits information back over MISO. In this transmission, the first data-word sent by the unit (as the read-command is sent) consists of 16-bits of non-applicable data. The subsequent 16-bit message contains the x-axis rate-sensor information (most significant byte followed by least-significant byte).

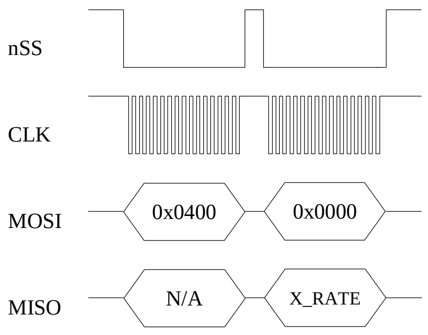


Figure 6 Single Register Read via Polled-Mode

4.2.2 IMU383 SPI Port Burst-Mode Read

In burst-mode, the unit returns predefined blocks of data in single groups, referred to as data-packets, without the need to send multiple read commands. These groups vary from eight to ten

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

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

words in length, depending on the packet selected. Table 6 lists the data-packets available for the unit. The data packets are described in more detail, including data-ordering and conversion factor information, in Section 6.4. Burst-reads and polled-reads should be initiated when data-ready signal indicates that new data is available. Reading data asynchronously to the data-ready signal can result in system instability. Application of a 1 kHz (+/- 0.1%) on pin 2 can be used to achieve more stable burst-read results.

Table 6 IMU383 Burst-Mode Data-Packets

Data-Packet	Register Address	Number of 16-bit Words	Pertinent Section	Description
Standard	0x3E	8	4.2.2	Rates, Accelerations, and Temperature
Extended	0x3F	10	4.2.2	Standard plus Timestamps

Burst-Read of Standard Data-Packet

The standard data-packet comprises data from eight predefined registers. Table 7 lists the data contained in a standard packet.

Table 7 IMU383 Burst-Mode Output Data Packet

Register Name	Description
STATUS	System Status Word (same as in register 0x3C)
X_RATE	Rate Sensor Output (X-Axis)
Y_RATE	Rate Sensor Output (Y-Axis)
Z_RATE	Rate Sensor Output (Z-Axis)
X_ACCEL	Accelerometer Output (X-Axis)
Y_ACCEL	Accelerometer Output (Y-Axis)
Z_ACCEL	Accelerometer Output (Z-Axis)
BOARD_TEMP	System Temperature

Burst-mode begins when the master requests a read from a burst-mode data-packet (i.e. 0x3E). Eight additional SPI cycles complete the read (one for each word in the standard data-packet). Figure 7 and Figure 8 illustrate the burst-mode sequence. Note: if the incorrect number of SPI cycles follow the burst-mode command, the SPI transfer will either complete early or remain in burst-mode; subsequent reads/writes will be out of sync with the SPI transfer cycle of the unit.

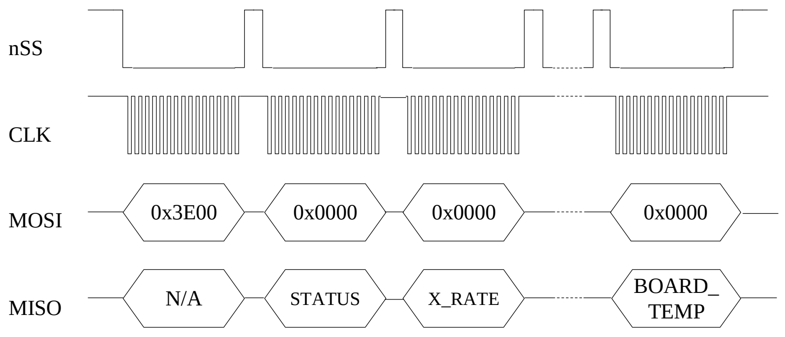


Figure 7 Multiple Register Read via Burst-Mode

Burst-Read of Extended Data-Packet

The extended data-packet comprises data from multiple predefined registers. Table 8 lists the data contained in an extended data packet. The registers are listed in the order in which they are sent during extended burst-mode read. The extended data packet includes the same 8 data words from the standard data packet, plus 2 additional data words which aid in the synchronization of the DMU383ZA to an external 1kHz clock signal.

TIMESTAMP1 is a 16-bit value that represents the time (in µs) between the sensor sampling point and the active edge of the external clock; TIMESTAMP2 is a 16-bit value representing the time (in µs) between the active edge of the external clock and the DRDY signal. Note that these TIMESTAMP values are only available as part of the extended data packet (cannot be read individually in polled reads), since they are relative measurements and are linked directly to the sensor values in the extended data packet.

Table 8 Extended Burst Mode Output Data Packet

Register Name	Description
STATUS	System Status Word (same as in register 0x3C)
X_RATE	Rate Sensor Output (X-Axis)
Y_RATE	Rate Sensor Output (Y-Axis)
Z_RATE	Rate Sensor Output (Z-Axis)
X_ACCEL	Accelerometer Output (X-Axis)
Y_ACCEL	Accelerometer Output (Y-Axis)
Z_ACCEL	Accelerometer Output (Z-Axis)
BOARD_TEMP	System Temperature
TIMESTAMP1	Time in µs from sensor sampling to active edge of synchronization clock
TIMESTAMP2	Time in µs from active edge of synchronization clock to data ready signal

Extended Burst-mode begins when the master requests a read of burst-mode extended data-packet by sending the 16-bit word 0x3F00. Ten additional 16-bit SPI cycles complete the read (one for each word in the extended data-packet). Figure 8 illustrates the burst-mode sequence. Note: if the incorrect number of SPI cycles follow the burst-mode command, the SPI transfer will either complete early or remain in burst-mode; subsequent reads/writes will be out of sync with the SPI transfer cycle of the DMU383ZA; the 1 kHz sync signal should be accurate to +/-0.1%, or it will not be recognized as a valid synchronization signal.

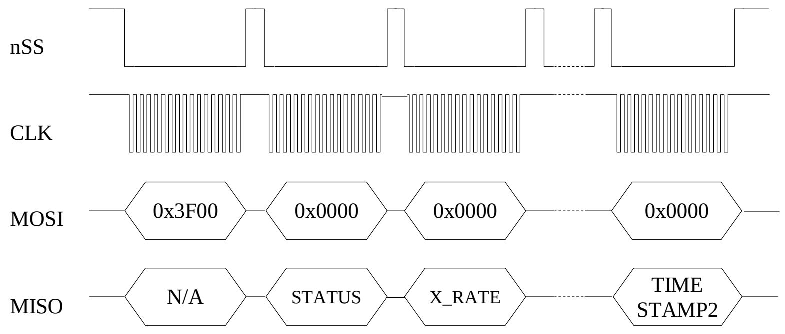


Figure 8 Extended Burst Mode Timing

Operational notes:

1. Care must be taken when switching between data-packets as values returned during the first burst-read of a new packet are invalid. A single burst read-cycle is needed to populate the internal burst-mode register, Then, subsequent reads from the same packet type contain valid information.
2. When combining polled and burst reads, use only single-register polled-reads.

4.3 Output Data Scaling

Table 9 lists each parameter and its conversion factor. Note: the scale-factors described below only applies to the values in and standard and extended burst-mode packets.

Table 9 IMU383 Output Data Scaling

Name	Function
X_RATE	X, Y, Z-axis rate-sensor information, twos complement format, conversion factor: 200 LSB/[°/sec] (default); changes with selected dynamic range ()
Y_RATE	
Z_RATE	
X_ ACCEL	X, Y, Z-axis accelerometer information, twos complement format, conversion factor: 4000 LSB/g (default) ; changes with selected dynamic range (Table 16)
Y_ ACCEL	
Z_ACCEL	
BOARD_TEMP	System temperature information, twos complement format, conversion: Tout [°C] = Vout · 0.07311 [°C/LSB] + 31.0 [°C]

4.4 System Registers

In addition to the output data registers, there are other read-only registers that provide IMU383 system information to the SPI master. Table 10 provides a description of each along with their read-addresses.

Table 10 IMU383 System Registers

Name	Read Address	Function
DIAGNOSTIC_STATUS	0x3C	Sensor self-test and over-range information (See Section 4.5)
MANUF_CODE	0x52	Product manufacturing code
UNIT_CODE	0x54	Additional product manufacturing information
PRODUCT_ID	0x56	Product ID 0x3830
SERIAL_NUMBER	0x58	Unique unit identification number
SW_VERSION	0x7E	Software Version
HW_VERSION	0x7F	Hardware Version

4.5 Diagnostic Status Register

The diagnostic status register contains information describing the results of the self-test as well as sensor over-range information. It is defined in Table 11.

Table 11 Diagnostic Status Register

	(Base Address: 0x3C), Read Only
Bits	**Description (Default: 0x0000)**
15	Failure of the accelerometers on Sensor chip 3 (1 - failure)
14	Failure of the accelerometers on Sensor chip 2 (1 - failure)
13	Failure of the accelerometers on Sensor chip 1 (1 - failure)
12	Failure of the rate sensors on Sensor chip 3 (1 - failure)
11	Failure of the rate sensors on Sensor chip 2 (1 - failure)
10	Failure of the rate sensors on Sensor chip 1 (1 - failure)
[9:6]	Reserved
5	Self-Test Success/Failure bit 0: Success, 1: Failure (Legacy)
4	Rate Sensor Over-range
3	Accelerometer Over-range
[2:0]	Unused

4.6 Sensors status registers

For the IMU383, triple sensor redundancy (along each axis) has been introduced. Each axes of every sensor is individually calibrated. The output signal generated for each sensor type is a composite of the three unique sensors. For example, output value for acceleration on axis X in normal conditions is combination of all three individual X-axis accelerometer readings.

The output from each sensor is constantly monitored and, if an anomaly is detected, the failing sensor is removed from the triad. This is done on a sensor-type basis. For example, if a failure of an accelerometer is detected, all three axes from that accelerometer will be disabled and the resulting acceleration output will consist of the combined signal from the two remaining good accelerometers. However, the angular-rate output will be a combination of all three rate-sensors.

Additional sensor status registers contain more detailed information about sensors status for each redundant sensor chip (read-only):

Table 12 Sensors Status Registers

Sensor Chip	SPI Register #	Bit 0 (0x01)	Bit 1 (0x02)	Bit 2(0x04)	Bit 3(0x08)	Bit 4 (0x0010)	Bit 5(0x0020)
1	0x1D	AccelX	AccelY	AccelZ	RateX	RateY	RateZ
2	0x1E	AccelX	AccelY	AccelZ	RateX	RateY	RateZ
3	0x1F	AccelX	AccelY	AccelZ	RateX	RateY	RateZ

If a specific bit is HIGH – that sensor axis is considered faulty and excluded from the output.

4.7 IMU383 SPI Register Write Methodology

The SPI master configures the unit by writing to specific registers. However, unlike reads, writes are performed *one byte at a time*. The specific registers that affect system configuration are listed in Table 13 along with their write-addresses.

Table 13 IMU383 Configuration Registers

Name	Write Address	Function
DATA_READY	0x34	See Table 14
OUTPUT_DATA_RATE	0x37	See Table 15: Sets Output Data Rate (ODR) of the unit
RS_DYNAMIC_RANGE LOW_PASS_FILTER	0x39 0x38	See Table 16.: Set the rate-sensor dynamic range and the digital filter
ORIENTATION_MSB ORIENTATION_LSB	0x74 0x75	See Sections 4.8.4and 7.4: Sets the orientation (x, y, and z-axes) of the unit

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. 0x34 for data ready options
- Change the most-significant bit of the address to 1 (the write-bit), e.g. 0x34 becomes 0xB4
- Create the write command by appending the write-bit/address combination with the value to be written to the register, e.g. 0xB404 (see Table 14 for a description of the self-test register)

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

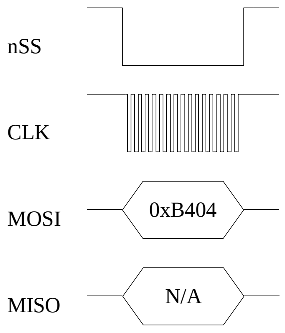


Figure 9 Single Register Write to Initiate Self-Test

Note: as described in the Register Reads (Section 4.2), a register read returns two bytes, in this case a read from register 0x34 returns data from registers 0x34 and 0x35.

4.8 Configuration Registers

4.8.1 Data-Ready

Data-ready register has address 0x34; individual bits are assigned according to Table 14.

Table 14 Data-Ready Register

(Base Address: 0x34 (52)), Read/Write	
Bits	**Description (Default: 0x0004)**
[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 data-ready bits enable the master to enable or disable the data-ready signal provided on pin 7 of the unit and to set the data-ready signal polarity (high or low). To enable data-ready with a high signal, the master sends 0xB406.

4.8.2 Output Data Rate

Output data rate (ODR) is contained in register 0x36; individual bits are assigned according to Table 15. Note: these settings apply only to data output via the unit SPI port and do not affect the low-level UART output port.

Table 15 Output Data Rate/Clock Configuration Register

(Base Address: 0x37 (55)), Read/Write	
Bits	**Description (Default: 0x0101)**
[7:0]	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

The ODR enables the master to specify the output rate of data provided by the unit. 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.

4.8.3 Rate-Sensor Scaling and Low-Pass Filter

The rate-sensor scaling configuration register has address 0x39 and described in Table 16. Note: these settings apply only to data output via the unit SPI port and do not affect the low-level UART output port.

Table 16 Sensor Scaling/Digital Low-Pass Filter Register

(Base Address: 0x39), Read/Write	
Bits	**Description (Default: 0x0206)**
[7:0]	Rate-Sensor Scaling/Dynamic Range Selector 0x01 (1): +/-62.5°/sec 0x02 (2): +/-125.0°/sec (default) 0x04 (4): +/-250.0°/sec 0x08 (8): +/-500.0°/sec 0x10 (16): +/-1000.0°/sec

The rate-sensor scaling selector adjusts the output scaling applied to the rate-sensor values[5] in the standard and extended data-packets. Additionally, this setting affects the limits that control the sensor over-range bit in the diagnostic status register (Table 11); 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 17.

[5] Limits will affect the signal output only if the system is capable of generating a signal of that level. For instance, for an IMU383ZA-200, a 220 °/sec limit will apply however the 440 °/sec limit will not, as the sensor is incapable of outputting signals greater than 220 °/sec.

Table 17 Rate-Sensor Scaling Factor

Dynamic Range	Scale Factor	Signal Limit	Over-Range Limit
+/-62.5°/sec	400 LSB/(°/sec)	+/-80.0 °/sec	+/-62.5 °/sec
+/-125.0°/sec	200 LSB/(°/sec)	+/-160.0 °/sec	+/-125.0 °/sec
+/-250.0°/sec	100 LSB/(°/sec)	+/-200.0 °/sec	+/-220.0 °/sec
+/-500.0°/sec	50 LSB/(°/sec)	+/-400.0 °/sec	+/-440.0 °/sec
+/-1000.0°/sec	25 LSB/(°/sec)	+/-600.0 °/sec	+/-660.0 °/sec

The digital low-pass filter configuration register has address 0x38 and described in Table 18. Note: these settings apply only to data output via the unit SPI port and do not affect the low-level UART output port.

Table 18 Sensor Scaling/Digital Low-Pass Filter Register

(Base Address: 0x38), Read/Write	
Bits	**Description (Default: 0x0206)**
[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 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 10 shows the output response of the different Bartlett filter settings.

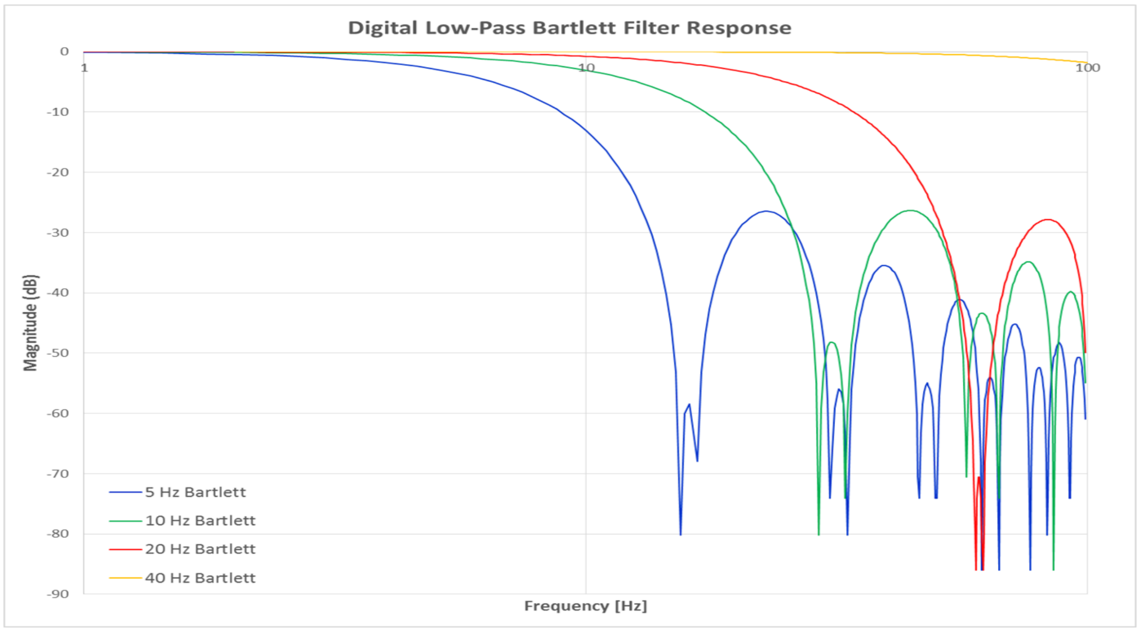


Figure 10 IMU383 Bartlett Filter Response

4.8.4 Axis Orientation Settings

The unit gives users the ability to set the axes orientation by selecting which axis aligns with the base axes as well as the sign. The only constraint is the axes must conform to a right-hand definition. The available settings are described in Section 7.4. The specific selections are provided in Table 30. The default setting is (-Uy, -Ux, -Uz).

To specify the orientation over SPI requires the user to write to two SPI registers (0x74 and 0x75) in succession. Writing to register 0x75 prior to 0x74 will have no effect. Additionally, reading the current orientation from register 0x74 will reset the write and require the user to rewrite the two bytes again (if done before both bytes are written).

To write the orientation, the user must first select the orientation and corresponding value from Table 30. Then the value must be split into most-significant and least-significant bytes. The most-significant byte is then written to register 0x74. This is followed by writing the least-significant byte to 0x75. Only by writing the two bytes back-to-back will the selection take effect.

For example, to select an orientation of (-Ux, +Uz, +Uy) the user must write 0x01 to 0x74 followed by 0x11 to 0x75. Note: this register does not require the user to swap bytes for the write to load the bytes properly, unlike other registers.

4.8.5 Saving the Configuration to EEPROM

The IMU383 enables the user to save certain settings to the EEPROM so they are set automatically the next time the system is started. This is accomplished by writing to register 0x76, as shown in Table 19.

To save the value written to a register either write the address of the register to 0x76 (to save an individual configuration setting) or a zero to save all settings.

Table 19 Saving Configuration to EEPROM

Data Written to Register 0x76	Function
0x00	Permanently saves orientation, dynamic range, data ready signal configuration, SPI packet rate, LPF type
0x01	Permanently saves fault detection settings
0x74	Permanently saves orientation
0x38	Permanently saves dynamic range
0x35	Permanently saves data ready signal configuration
0x36	Permanently saves SPI packet rate
0x39	Permanently saves LPF type

4.8.6 Hardware and Software Version

SPI register 0x7E contains information about the hardware and software of the IMU383, as listed in Table 20. The software version is contains both the major and minor version numbers concatenated. For example, a value of 0x7F = 127, refers to a major version of 12 and a minor version of 7.

Table 20 Hardware and Software Version

(Base Address: 0x7E), Read Only	
Bits	**Description (Default: 0x0000)**
[15:8]	Hardware Version
[7:0]	Software Version (Major and Minor versions concatenated)

4.8.7 Sensor Control registers

IMU383 has six inertial sensor chips. The three sensor control registers described in the Table 21 provide the ability to control the output of the composite sensor data by describing how data from individual sensor chips are combined.

Table 21 Sensor control registers

Sensor Chip	SPI Register #	Bit 0 (0x01)	Bit 1 (0x02)	Bit 2(0x04)	Bit 3(0x08)	Bit 4 (0x0010)	Bit 5(0x0020)
1	0x1A	AccelX	AccelY	AccelZ	RateX	RateY	RateZ
2	0x1B	AccelX	AccelY	AccelZ	RateX	RateY	RateZ
3	0x1C	AccelX	AccelY	AccelZ	RateX	RateY	RateZ

If a specific bit is written HIGH, the corresponding sensor axis is activated and included into combined output value as long as data from that sensor is deemed healthy (no fault is detected by the internal fault detection mechanism). If a sensor fault occurs, the corresponding sensor control bit is ignored.

If a specific bit is set LOW, data from the corresponding sensor axis is excluded from the combined output value. Note: if a specific sensor axis is excluded from the combined output but internally deemed good by the fault detection algorithm, as good one – it will continue to participate in the fault detection mechanism.

All bits in registers 0x1A to 0x1C are HIGH by default.

4.8.8 Fault detection Control registers

Inside the IMU383 fault detection, up to five different fault detection algorithms may be enabled at the same time. The registers in Table 22 control the configuration of fault detection mechanism. Note: any bitmask written to the control register only affects the functioning of the fault detection functions related to that bitmask. For instance, writing 0x08 to register 0x01enables the accelerometer consistency check but does not affect the status of any other fault detection function. Alternately, writing 0x08 to register 0x02 disables the accelerometer consistency check without affecting any other check.

Table 22 Fault Detection Control Registers

Registers	Function	Default value
0x01	Enable fault detection functions (bitmask), see Table 23	0x18
0x02	Disable fault detection functions (bitmask) see Table 23	0
0x03	Fault detection settings unlock key	0

The current status of fault detection can be checked by reading from register 0x01. The value in this register provides an indication of what checks are enable or disabled. Register 0x02 is read-only and will not provide any information about the status of fault detection.

Before writing to either register 0x01 or 0x02, the user must first unlock the fault detection interface. This is done by writing 0x96 and 0x78 sequentially into register 0x03. If the unlock was successful, BIT 1 in the status register (0x3C) will be set. After a successful unlock it is possible to change the fault detection settings until the part is repowered. The fault detection setting can be saved to memory by writing address 0x01 to register 0x76 (described in Section 4.8.5).

Bitmasks used to enable or disable the fault detection checks as described in Table 23. The specific bitmask written into registers 0x01 or 0x02 will affect the operation of the fault detection. Writing a bit value of 0x00 to either register will not affect the checks that are enabled or disabled.

Table 23 Enable/Disable Bitmasks used with Registers 0x01 and 0x02

Bitmask	Function
0x08	Accelerometer Data Consistency Fault Detection
0x10	Rate sensors Data Consistency Fault Detection

4.9 Suggested Operation

The following operational procedure and timing specifications should be adhered to while communicating with the IMU383unit via SPI to ensure proper system operation. These points are highlighted later in this section.

4.9.1 Startup Timing

The following timing applies at system startup (see Figure 11):

- During system setup, the unit should be held in reset (nRST line held low) until the SPI master is configured and the system is ready to begin communications with the unit.

- After releasing the reset line, the unit requires 300 msec ($t_{System\ Delay}$) before the system is ready for use
- Data should be read from the unit after the data-ready line is set active (low) (see Section 4.8.1)

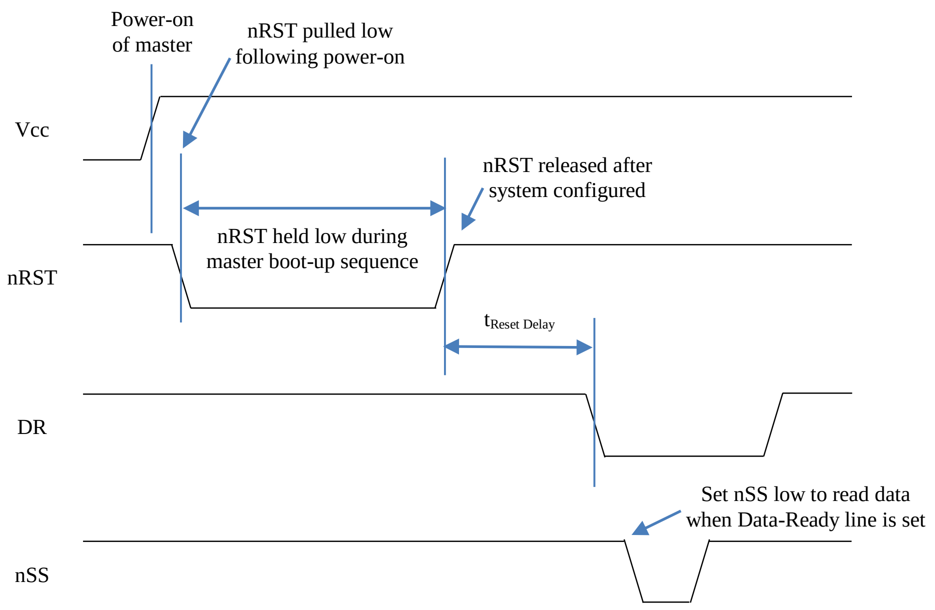

Power-on of master
nRST pulled low following power-on
Vcc
nRST released after system configured
nRST held low during master boot-up sequence
nRST
tReset Delay
DR
Set nSS low to read data when Data-Ready line is set
nSS

Figure 11 Startup Timing

4.9.2 SPI Timing

The timing requirements for the SPI are listed in Table 24 and illustrated in Figure 12 and Figure 13. In addition, the following operational constraints apply to the SPI communications:

- The unit operates with CPOL = 1 (polarity) and CPHA = 1 (phase)
- Data is transmitted 16-bits words, Most Significant Bit (MSB) first

Table 24 SPI Timing Requirements

Parameter	***Description***	***Value***	***Units***
f_{CL}	SPI clock frequency	1.2 (max)	MHz
t_{DELAY}	Time between 16-bit bus cycles	15 (min)	usec
$t_{SU,NSS}$	nSS setup time prior to clocking data	133	nsec
$t_{h,NSS}$	nSS hold time following clock signal	133	nsec
$t_{V,MISO}$	Time after falling edge of previous clock-edge that MISO data-bit is invalid	25	nsec
$t_{SU,MOSI}$	Data input setup time prior to rising edge of clock	5	nsec
$t_{h,MOSI}$	Data input hold time following rising edge of clock	4	nsec

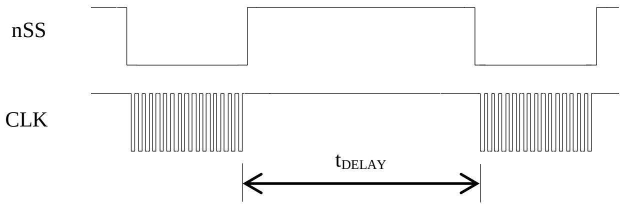


Figure 12 Delay Time

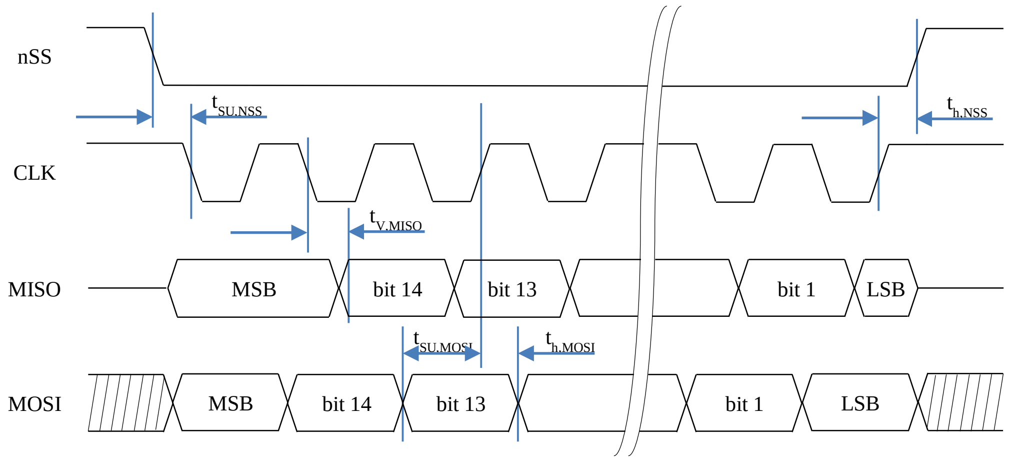


Figure 13 SPI Timing Diagram

4.10 Signal Synchronization

The IMU383 is capable of synchronizing its output with a 1 kHz external clock signal, in the form of a square wave, applied to Pin 2. When detected, the IMU383 ignores its internal timer, replacing it with the external clock. Care must be taken to ensure the signal is a true 1 kHz clock (1kHz +/-0.1%), as the firmware will assume all signals on the line have a 1 kHz frequency. Also, once an external sync pulse is applied, the signal must remain or the unit will cease its sampling and processing functions; the system cannot return to internal timing without resetting the system and removing the sync signal. Note that since the output data rate of the IMU383 is always less than the 1kHz external clock, there is a potential for ambiguity in which edge of the 1kHz clock is active (used for synchronization). Note that since the delay from the 1kHz edge to the data ready output is ~300us, the active edge of the 1kHz clock is the one immediately preceding the data ready pulse.

If it is desirable to determine the sensor sampling points more precisely, there are two methods of accomplishing that. In the first method, the Sensor Sampling Indicator (pin 1) is used to detect the sampling points (see "Method 1) below). In the second method, the Timestamp registers (read via the Extended Burst Mode command) are used to determine the sampling points (see Method 2 below).

Note that the inertial sensors are sampled at a nominal rate of 800Hz. The sensor readings are calibrated (at 800Hz), and decimated down to the selected Output Data Rate (ODR). The data ready signal indicates when the latest sample has been process and is available via the SPI interface.

Method 1- Using the Sensor Sampling Indicator (Pin 1)

When the user requires finer knowledge of the instant that data is sampled, the IMU383 provides the ability to determine when the sensor read is performed. A falling edge of the signal provided on pin 1 indicates when the inertial sensors are sampled. The rising edge of the 1kHz clock immediately preceding the data ready signal is the active edge of the clock.

Method 2- Using the Timestamp Registers in the Extended Burst Message

The extended burst message described on page 15 contains two values called TIMESTAMP1 AND TIMESTAMP2. These values contain the time offset in microseconds relative to the active edge of the 1kHz clock used to generate the most recent data ready signal. TIMESTAMP1 is the delay from the sensor sampling point to the active edge of the 1kHz clock; TIMESTAMP2 is the delay from the active edge of the 1kHz clock to the data ready signal. Note that the active edge of the 1kHz signal is the rising edge immediately preceding the data ready signal.

4.11 Bootloader

The IMU383 possesses the capability to upgrade the firmware via an on-board bootloader. See Section 9 for a description of the process.

5 IMU383 UART Port Interface Definition

This section of the manual explains the IMU383 packet formats as well as the supported commands. NAV-VIEW 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 input data packets, please see refer to Appendix B.

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

5.1 General Settings

Standard serial port settings are used with 1 start bit, 8 data bits, no parity bit, 1 stop bit, and no flow control. Standard baud rates supported are: 38400, 57600, 115200, and 230400. The default baud rate is 115200.

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, which corresponds to a sequence of two ASCII 'U' characters.
- 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 IMU383 Series inertial system within a 4 second period.

5.2 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 25 defines number formats:

Table 25 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	

5.3 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 a IMU383 Series unit can be pinged from a terminal emulator. To Ping a IMU383 Series unit, type the ASCII sequence 'UUPK'. If properly connected, the IMU383 Series unit will respond with 'PK'. All other communications with the IMU383 Series unit require the 2-byte CRC. {Note: A IMU383 Series unit will also respond to a ping command using the full packet formation with payload 0 and correctly calculated CRC. Example: 0x5555504B009ef4 }.

5.3.1 Packet Header

The packet header is always the bit pattern 0x5555.

5.3.2 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.

5.3.3 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.

5.3.4 Payload

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

5.3.5 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 B for sample code that implements the 16-bit CRC algorithm.

5.3.6 Messaging Overview

Table 26 summarizes the messages available by IMU383 Series model. 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 the table below. The referenced variables are defined in the detailed sections following. Output messages are sent from the IMU383 Series 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 IMU383 Series 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 26 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
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
T0	0x5430	28	Test 0 (Detailed BIT and Status)	Output Message
Output Messages: Measurement Data (Continuous or Polled)				
S0	0x5330	30	Scaled Sensor 0 Data	Output Message
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 IMU383 Standard UART Port Commands and Messages

This section describes the functionality and operation of the UART port.

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))

6.1.2 Ping Response

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.

6.1.3 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.

6.1.4 Echo Response

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 IMU383 Series, and to calibrate or reset the IMU383 Series.

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 'T0', '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

6.2.9 *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 *faiiledInputPacketType* 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

6.3.3 Test 0 (Detailed BIT and Status) Packet

Test ('T0' = 0x5430)				
Preamble	Packet Type	Length	Payload	Termination
03.3x5555	0x5430	0x1C	<T0 payload>	<CRC (U2)>

This packet contains detailed BIT and status information. The full BIT Status details are described in Section 8 of this manual.

T0 Payload Contents					
Byte Offset	Name	Format	Scaling	Units	Description
0	BITstatus	U2	-	-	Master BIT and Status Field
2	hardwareBIT	U2	-	-	Hardware BIT Field
4	hardwarePowerBIT	U2	-	-	Hardware Power BIT Field
6	hardwareEnvironmentalBIT	U2	-	-	Hardware Environmental BIT Field
8	comBIT	U2	-	-	communication BIT Field

10	comSerialABIT	U2	-	-	Communication Serial A BIT Field
12	comSerialBBIT	U2	-	-	Communication Serial B BIT Field
14	softwareBIT	U2	-	-	Software BIT Field
16	softwareAlgorithmBIT	U2	-	-	Software Algorithm BIT Field
18	softwareDataBIT	U2	-	-	Software Data BIT Field
20	hardwareStatus	U2	-	-	Hardware Status Field
22	comStatus	U2	-	-	Communication Status Field
24	softwareStatus	U2	-	-	Software Status Field
26	sensorStatus	U2	-	-	Sensor Status Field

6.4 Output Packets (Polled or Continuous)

NOTE: Sensor data scaling is specific to the message or register being read. Please see scaling information for each sensor data read operation. Specifically, the sensor data in the SPI S0_BURST, the SPI S1_BURST, the UART S0 message and the UART S1 message are scaled differently than the scaling for reading the individual SPI sensor data registers.

6.4.1 Scaled Sensor Data Packet 0

Scaled Sensor Data ('S0' = 0x5330)				
Preamble	Packet Type	Length	Payload	Termination
0x5555	0x5330	0x1E	<S0 payload>	<CRC (U2)>

This packet contains scaled sensor data. The scaled sensor data is fixed point, 2 bytes per sensor, MSB first, for 13 sensors in the following order: accels(x,y,z); gyros(x,y,z); mags(x,y,z); temps(x,y,z,board). Data involving angular measurements include the factor pi in the scaling and can be interpreted in either radians or degrees. Note the timer value can be used for synchronization and computation of DeltaT. It may appear in NAV-VIEW log files under another column heading.

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

S0 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	Reserved	I2			
14	Reserved	I2			
16	Reserved	I2			
18	xRateTemp	I2	200/2^16	deg. C	X rate temperature
20	yRateTemp	I2	200/2^16	deg. C	Y rate temperature
22	zRateTemp	I2	200/2^16	deg. C	Z rate temperature
24	boardTemp	I2	200/2^16	deg. C	CPU board temperature
26	timer	U2	15.259022	uS	Free running fast counter 1s= 65535, captured at sampling
28	BITstatus	U2	-	-	Master BIT and Status

6.4.2 Scaled Sensor Data Packet 1 (Default IMU Data)

NOTE: Sensor data scaling is specific to the message or register being read. Please see scaling information for each sensor data read operation. Specifically, the sensor data in the SPI S0_BURST, the SPI S1_BURST, the UART S0 message and the UART S1 message are scaled differently than the scaling for reading the individual SPI sensor data registers.

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. Note the timer value can be used for synchronization and computation of DeltaT. It may appear in NAV-VIEW log files under another column heading.

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	boardTemp	I2	200/2^16	deg. C	CPU board temperature
20	timer	U2	15.259022	uS	Free running fast counter 1s= 65535, captured at sampling
22	BITstatus	U2	-	-	Master BIT and Status

7 IMU383 Advanced UART Port Commands

The advanced commands allow users to programmatically change the IMU383 Series settings. This section of the manual documents all of the settings and options contained under the Unit Configuration tab within NAV-VIEW. Using these advanced commands, a user's system can change or modify the settings without the need for NAV-VIEW.

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 type, algorithm type, etc. 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 Table 27 below.

Table 27 Configuration Fields

configuration fields	***field ID***	***Valid Values***	***Description***
Packet rate divider	0x0001	0,1,2,4,5,10, 20, 25, 50	quiet, 100Hz, 50Hz, 25Hz, 20Hz, 10Hz, 5Hz, 4Hz, 2Hz
Unit BAUD rate	0x0002	2,3,5,6	38400, 57600, 115200 (default), 230400
Continuous packet type	0x0003	Any output packet type	Not all output packets available for all products. See detailed product descriptions.
Unused	0x0004		
Gyro Filter Setting	0x0005	18750-65535 [2Hz] 8035-18749 [5Hz] 4018-8034 [10Hz] 2411-4017 [20Hz] 1741-2410 [25Hz] 1205-1740 [40Hz] 1-1204 [50 Hz] 0 [Unfiltered]	Sets low pass cutoff for rate sensors. Cutoff Frequency choices are 2, 5, 10, 20, 25, 40, and 50Hz
Accelerometer Filter Setting	0x0006	18750-65535 [2Hz] 8035-18749 [5Hz] 4018-8034 [10Hz] 2411-4017 [20Hz] 1741-2410 [25Hz] 1205-1740 [40Hz] 1-1204 [50 Hz] 0 [Unfiltered]	Sets low pass cutoff for accelerometers. Cutoff Frequency choices are 2, 5, 10, 20, 25, 40, and 50Hz
Orientation	0x0007	See below	Determine forward, rightward, and downward facing sides
Reserved	0x0009		
Reserved	0x000A		
Sensor Enable Setting	0x0042	0x0 to 0x7	Enables sensors activation (requires restart for mode to take effect). Used to disable a sensor chip in case it introduces extra noise or crosstalk.
Output Select Settings	0x0043	0x0 to 0x7	Selects sensors for combined data output.
Fault detection fault cause, sensor chip #1	0x004C	0x0 to 0x5	Cause of the fault was for sensor chips 1 to 3:

Fault detection fault cause, sensor chip #2	0x004D	0x0 to 0x5	0x0: No fault 0x4: Accelerometer Consistency Fault 0x5: Rate-Sensor Consistency Fault
Fault detection fault cause, sensor chip #3	0x004E	0x0 to 0x5	
Accelerometer consistency check enable	0x0061	1 – enabled, 0 - disabled	See Section 4.8.8 (Default: 1)
Rate-Sensor consistency check enable	0x0062	1 – enabled, 0 - disabled	See Section 4.8.8 (Default: 1)

Note: BAUD rate can be changed by command WF and will take effect after unit reset or power cycle.

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 6.4 for a complete list of the available packet types.

7.3 Digital Filter Settings

These two fields set the digital low pass filter cutoff frequencies (See Table 28). Each sensor listed is defined in the default factory orientation. Users must consider any additional rotation to their intended orientation.

Table 28 Digital Filter Settings

Filter Setting	Sensor
FilterGyro	Ux,Uy,Uz Rate
FilterAccel	Ux,Uy,Uz Accel

7.4 Orientation Field

This field defines the rotation from the factory to user axis sets. This rotation is relative to the default factory orientation for the appropriate IMU383 family model. The default factory axis setting for the IMU383-200 orientation field is (-Uy, -Ux, -Uz) which defines the connector pointing in the +Z direction and the +X direction going from the connector through the serial number label at the end of the IMU383. The user axis set (X, Y, Z) as defined by this field setting is depicted in Figure 14 below:

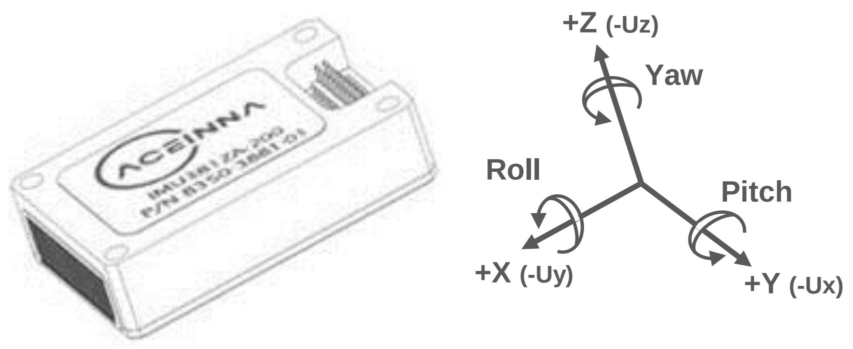


Figure 14 IMU383-200 Default Orientation Field (0x006B)

Table 29 IMU383 Orientation Definitions

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 (See Table 30). Setting/Writing the field to anything else generates a NAK and has no effect.

Table 30 IMU383 Orientation Fields

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

7.5 Fault-Detection Fault-Causation Fields

Fields 0x4C to 0x4E provide the cause of any faults flagged by the fault detection check. These fields are read-only with the faults indicated by the values in Table 31.

Table 31 Fault-Detection Fault-Causation Flags

Field Value	Corresponding Fault
0x0	No fault
0x4	Accelerometer Consistency Fault
0x5	Rate-Sensor Consistency Fault

When a fault occurs, the master BIT (Section 8.2) will be set to indicate a Hardware Status error (the sensor status BIT is also set but the sensor-status field does not indicate an error – this will be implemented in the next code release). By reading the Test 0 message (Section 6.3.3) and parsing the message, the hardware status (Table 33 in Section 8.3) indicates either an accelerometer or a rate-sensor fault via bits 4 or 5 respectively. Once it has been determined that a Fault-Detection error occurred, reading Fields 0x4C, 0x4D, and 0x4E will result in the failure mechanism and sensor-chip that triggered the fault.

7.6 Fault-Detection Enable/Disable

Fields 0x61 to 0x62 enables or disables the Fault-Detection checks. Table 27 lists the checks enabled and disabled at startup.

7.7 Commands to Program Configuration

7.7.1 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

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

7.7.2 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

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 set 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

7.8 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

7.9 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

7.10 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

7.11 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 IMU383 Advanced UART Port BIT

8.1 Built In Test (BIT) and Status Fields

Internal health and status are monitored and communicated in both hardware and software. The ultimate indication of a fatal problem is a hardware BIT signal on the user connector which is mirrored in the software BIT field as the masterFail flag. This flag is thrown as a result of a number of instantly fatal conditions (known as a "hard" failure) or a persistent serious problem (known as a "soft" failure). Soft errors are those which must be triggered multiple times within a specified time window to be considered fatal. Soft errors are managed using a digital high-pass error counter with a trigger threshold.

The masterStatus flag is a configurable indication as determined by the user. This flag is asserted as a result of any asserted alert signals which the user has enabled.

The hierarchy of BIT and Status *fields* and signals is depicted here:

- *BITstatus Field*
  - masterFail
    - hardwareError
      - *hardwareBIT Field*
        - HwTestError
        - AccelerometerError
        - RateSensorError
    - SoftwareError
      - *softwareBIT Field*
        - calibrationCRCError
  - masterStatus
    - hardwareStatus
      - *hardwareStatus Field*
        - unlockedEEPROM
        - AccelerometerFaultDetected
        - RateSensorFaultDetected
    - softwareStatus
      - *softwareStatus Field*
        - Self Test Mode
    - sensorStatus
      - *sensorStatus Field*
        - overRange (enabled by default)

8.2 Master BIT and Status (BITstatus) Field

The BITstatus field is the global indication of health and status of the IMU383 Series product (See Table 32). The LSB contains BIT information and the MSB contains status information.

There are four intermediate signals that are used to determine when masterFail and the hardware BIT signal are asserted. These signals are controlled by various systems checks in software that are classified into three categories: hardware, communication, and software. Instantaneous soft failures in each of these four categories will trigger these intermediate signals, but will not trigger the masterFail until the persistency conditions are met.

There are four intermediate signals that are used to determine when the masterStatus flag is asserted: hardwareStatus, sensorStatus, comStatus, and softwareStatus. masterStatus is the logical OR of these intermediate signals. Each of these intermediate signals has a separate field with individual indication flags. Each of these indication flags can be enabled or disabled by the user. Any enabled indication flag will trigger the associated intermediate signal and masterStatus flag.

Table 32 IMU383 BIT Status Field

BITstatus Field	***Bits***	***Meaning***	***Category***
masterFail	0	0 = normal, 1 = fatal error has occurred	BIT
HardwareError	1	0 = normal, 1= internal hardware error	BIT
Reserved	2		BIT
softwareError	3	0 = normal, 1 = internal software error	BIT
Reserved	4:7	N/A	
masterStatus	8	0 = nominal, 1 = hardware, sensor, com, or software alert	Status
hardwareStatus	9	0 = nominal, 1 = programmable alert	Status
Reserved	10	N/A	Status
softwareStatus	11	0 = nominal, 1 = programmable alert	Status
sensorStatus	12	0 = nominal, 1 = programmable alert	Status
Reserved	13:15	N/A	

8.3 hardwareBIT Field

The hardwareBIT field contains flags that indicate various types of internal hardware errors (See Table 33). Each of these types has an associated message with low level error signals. The hardwareError flag in the BITstatus field is the bit-wise OR of this hardwareBIT field.

Table 33 IMU383 Hardware BIT Field

hardwareBIT Field	***Bits***	***Meaning***	***Category***
reserved	0:3	N/A	
accelerometerError	4	0 = normal, 1 = error	Hard
gyroError	5	0 = normal, 1 = error	Hard
selfTestError1	6		Hard
selfTestError2	7		Hard

reserved	8:15	N/A	

8.4 softwareDataBIT Field

The softwareDataBIT field contains flags that indicate low level software data errors (See Table 34). The dataError flag in the softwareBIT field is the bit-wise OR of this softwareDataBIT field.

Table 34 IMU383 Software Data BIT Field

SoftwareDataBIT Field	*Bits*	*Meaning*	*Category*
calibrationCRCError	0	0 = normal, 1 = incorrect CRC on calibration EEPROM data or data has been compromised by a WE command.	Hard
Reserved	2:15	N/A	

8.5 hardwareStatus Field

The hardwareStatus field contains flags that indicate various internal hardware conditions and alerts that are not errors or problems (See Table 35). The hardwareStatus flag in the BITstatus field is the bit-wise OR of the logical AND of the hardwareStatus field and the hardwareStatusEnable field. The hardwareStatusEnable field is a bit mask that allows the user to select items of interest that will logically flow up to the masterStatus flag.

Table 35 IMU383 Hardware Status BIT Field

hardwareStatus Field	*Bits*	*Meaning*
unlockedEEPROM	3	0=locked, WE disabled, 1=unlocked, WE enabled
accelFaultDetected	4	1 – accelerometer fault detected
gyroFaultDetected	5	1 – gyro fault detected
Reserved	4:15	N/A

8.6 sensorStatus Field

The sensorStatus field contains flags that indicate various internal sensor conditions and alerts that are not errors or problems (See Table 36).

Table 36 IMU383 Sensor Status Field

sensorStatus Field	*Bits*	*Meaning*
overRange	0	0 = not asserted, 1 = asserted
Reserved	1:15	N/A

9 IMU383 BOOTLOADER

A bootloader function is available to upgrade the IMU383 firmware. The firmware can be updated over either the UART or SPI interface, as described in the following sections.

9.1 Updating firmware over UART Interface

A user can initiate the bootloader at any time by sending 'JI' command (see below command's format) to application program. This command forces the unit to enter bootloader mode. The unit will communicate at 57.6Kbps baud rate regardless of the original baud rate the unit is configured to. The Bootloader always communicates at 57.6Kbps until the firmware upgrade is complete.

As an additional device recovery option immediately after powering up, every IMU383ZA will enter a recovery window of 100ms prior to application start. During this 100mS window, the user can send 'JI' command at 57.6Kbs to the Bootloader in order to force the unit to remain in Bootloader mode.

Once the device enters Bootloader mode via the 'JI' command either during recovery window or from normal operation, a user can send a sequence 'WA' commands to write a complete application image into the device's FLASH.

After loading the entire firmware image with successive 'WA' commands, a 'JA' command is sent to instruct the unit to exit Bootloader mode and begin application execution. At this point the device will return to its original baud rate.

Optionally, the system can be reboot by toggling power or toggling nRst (pull low and release) to restart the system.

Firmware upgrade is performed by a Write APP command through UART port, through Windows GUI, NAV-View, or a python program. See Appendix A.

The following commands allow users to install a pre-built binary into flash memory and force system enters either bootloader or application mode.

9.1.1 Jump to BootLoader Command

Jump To BootLoader ('JI' = 0x4A49)				
Preamble	Packet Type	Length	Payload	Termination
0x5555	0x4A49	0x00		CRC (U2)

The command allows system to enter bootloader mode.

9.1.2 Write APP Command

Write APP ("WA" = 0x5741)				
Preamble	Packet Type	Length	Payload	Termination
0x5555	0x5741	length+5		CRC (U2)

The command allows users to write binary sequentially to flash memory in bootloader mode. The total length is the sum of payload's length and 4-byte address followed by 1-byte data length. See the following table of the payload's format.

WA Payload Contents					
Byte Offset	Name	Format	Scaling	Units	Description
0	startingAddress	U4	-	bytes	The FLASH word offset to begin writing data
4	byteLength	U1	-	bytes	The word length of the data to write
5	dataByte0	U1	-	-	FLASH data
6	dataByte1	U1	-	-	FLASH data
…	…				
4+byteLength	dataByte	U1	-	-	FLASH data

Payload starts from 4-byte address of flash memory where the binary is located. The fifth byte is the number of bytes of *dataBytes*, but less than 240 bytes. User must truncate the binary to less than 240-byte blocks and fill each of blocks into payload starting from the sixth-byte.

9.1.3 Jump to Application command

Jump To Application ('JA' = 0x4A41)				
Preamble	Packet Type	Length	Payload	Termination
0x5555	0x4A41	0x00		CRC (U2)

The command allows system directly to enter application mode.

9.2 Updating firmware using the SPI Interface

9.2.1 Boot Mode Registers

Boot Mode registers are used to control IMU383ZA in-system or in-field application upgrade process via SPI interface. The following table provides description of each register along with their addresses.

Table 37. IMU383ZA Boot Mode Registers

Name	**Read Address**	**Function**
BOOT_STATUS	0x78	Unit Boot Status register in BOOT and NORMAL mode
BOOT_COMMAND	0x7A	Unit Boot Command register in BOOT and NORMAL mode
BOOT_DATA	0x7C	Unit Boot Data transfer register in BOOT mode

9.2.2 Boot Mode Status Register (0x78)

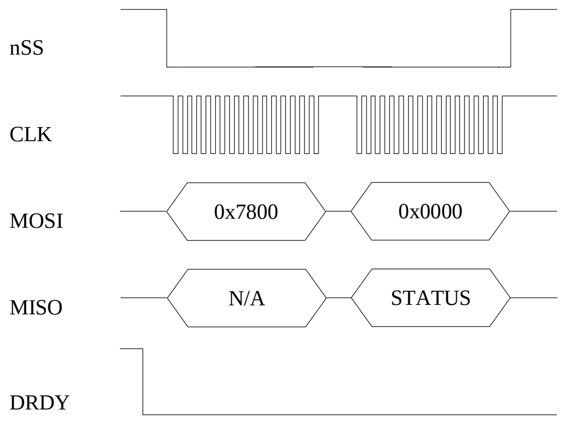
The Boot Status register contains information about current unit status in Boot Mode.

Table 38 Boot Mode Status Register

Bits	Description
15	System Busy 0: Ready, 1: Busy
14	System Mode 0: Normal Mode, 1: Boot Mode
13	Last Operation Status 0: Pass, 1: Fail (sticky, will stay 1 until reset)
12	CRC Check Status: 0: Pass, 1: Fail
11	Application image compatibility: 0 – Pass, 1 - Fail
10	1
9	1
8	Programming Status: 0 – Pass, 1 - Fail
7	1
6	0
5	1
4	0
3	1
2	0
1	1
0	1

The following example highlights how to read the Boot Status register over SPI interface:

- Create 16-bit the Read Status command by placing address of Boot Status register into MSB and placing 0x00 into LSB
- Clock out 16-bit command followed by 16-bit word 0x0000
- 16-bit Status register contents will be placed by device onto MISO line



NOTE: Data Ready signal will be HIGH when device is BUSY and LOW when device available for new transaction over SPI interface.

9.2.3 Boot Command Register (0x7A)

The Boot Command register dedicated to receive Boot Mode related commands. The commands provided in Table 39.

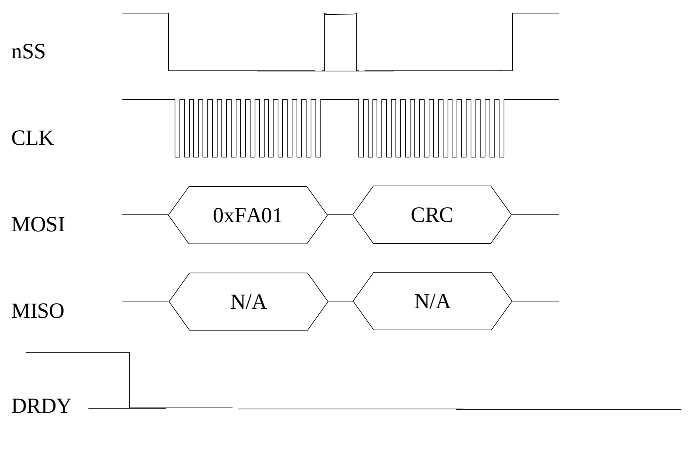
Table 39 Boot Commands

Command Code	Description	Action
0x01	Enter Boot Mode	Device will enter Boot mode from application (if not in boot mode already).
0x02	Program Block	Instructs unit to program previously uploaded data block
0x03	Reset status register	Resets (clears) status bits in status register.
0x05	Activate new application	Activate new application image
0x06	Reset device	Reset device

The following example highlights how to send Boot Commands to the unit:

- Create 16-bit command word: set bit 7 of Boot Command address register to 1 and place it to MSB, place command code from Table 39to LSB.

The next figure illustrates sending "Enter Boot Mode" command to the device over SPI interface:



NOTE: Data Ready signal will be HIGH when device is BUSY and LOW when device available for new transaction over SPI interface.

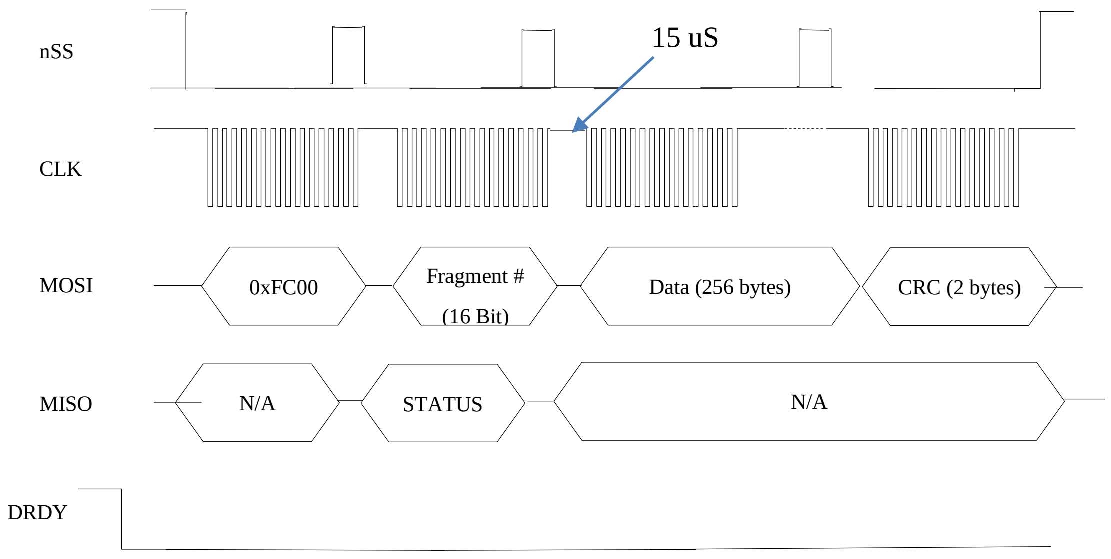
NOTE: Each command message includes 16-bit CRC which covers first 16 bit command word and 32-bit data word.

9.2.4 Boot Data Register (0x7C)

The Boot Data register (0x7C) is dedicated to receiving fragments of application image one at a time. The application image is split into N 256-byte fragments. Messages containing those fragments are sent to the device sequentially and must include fragment number and CRC.

Fragment N corresponds to byte offset N*256 from start of binary application image. Once received and verified the fragment gets buffered in CPU RAM. If required one can read BOOT_STATUS register after sending each fragment to ensure that the fragment was received successfully. This is done by checking bits "Last Operation status" and "CRC check status".

The next figure illustrates sending fragment of application image to the device over SPI interface:



NOTE: Data Ready signal will be HIGH when device is BUSY and LOW when device available for new transaction over SPI interface.

NOTE: Status word will be sent back in second 16-bit word. It will reflect status of previous SPI transaction execution.

NOTE: There should be delay at least 15 uS between fragment number word and data words.

NOTE: Application image will be padded for size to be even to 256 bytes.

9.2.5 Boot Loading Sequence

1. Reset unit using nRST pin or power cycle.
2. Wait about 1 second.
3. Read BOOT_STATUS register and check for boot mode compatibility (register should have pattern 0x06AB in NORMAL mode or 0x46AB in BOOT mode). If register has pattern 0x46AB that means unit already in boot mode. So skip steps 4,5,6.
4. Send “Enter Boot Mode” command to unit.
5. Wait about 1 second.
6. Read BOOT_STATUS status register to ensure, that device entered Boot Mode (register should have pattern 0x46AB)
7. Send application fragments to unit one by one (up to 16*256-bytes fragments – 4K). If desired check BOOT_STATUS register after each fragment to ensure it was successfully accepted (bits “Last Operation Status” and “CRC check status” should be 0). Note: if application size is not even to 256 bytes, the final application segment needs to be padded to 256 bytes by value 0xff
8. Issue “Program Block” command and wait about 300 – 500 ms or when DRDY becomes active again allowing command to complete (time includes erasing of FLASH pages). After that check BOOT_STATUS register (bits “Last Operation Status”, “CRC check status”, “Application image compatibility” and “Programming status” should be all 0).
9. Cycle through steps 7 and 8 until whole image is programmed.
10. Send “Activate New Application” command to unit.
11. Check BOOT_STATUS register to see if command succeeded.
12. Send Reset command or toggle “nRST” signal. Unit will reboot and start executing new application. Wait for about 1 second and then read BOOT_STATUS register. It should have pattern 0x06AB.

NOTE: Sending SPI messages to the unit allowed only when DRDY signal is active (low) except very first command after unit reset.

NOTE: After sending each data fragment to the unit it’s recommended to read back BOOT_STATUS register to check if programming procedure had some errors. If “CRC check status” bit is set to 1 – same fragment can be resent. If “Last Operation Status” bit set to 1 whole procedure needs to be repeated from start (as many times as needed).

10 Warranty and Support Information

10.1 Customer Service

As an ACEINNA customer you have access to product support services, which include:

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

10.2 Contact Directory

United States:	Email: techsupport@aceinna.com
Non-U.S.:	Refer to website www.aceinna.com

10.3 Return Procedure

10.3.1 Authorization

Before returning any equipment, please contact ACEINNA 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?

10.3.2 Identification and Protection

If the equipment is to be shipped to ACEINNA 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.

10.3.3 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.

10.3.4 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.

10.4 Warranty

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

Appendix A: Installation and Operation of NAV-VIEW

NAV-VIEW has been designed to allow users to control all aspects of the compatible Aceinna IMU product families. Note some of the functions described are not available on all of the product families (for example, the AHRS or INS functionality does not apply to an IMU). NAV-VIEW version 3.5.2 or higher is required.

NAV-VIEW Computer Requirements

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

- CPU: Pentium-class (1.5GHz minimum)
- RAM Memory: 500MB minimum, 1GB+ recommended
- Hard Drive Free Memory: 20MB
- Operating System: Windows 7 and 10
- Properly installed Microsoft .NET 2.0 or higher

Install NAV-VIEW

To install NAV-VIEW onto your computer:

1. Go to https://www.aceinna.com/manuals and locate the link to the Nav-View zip file. Click the link and download the latest version of the software.
2. Locate the downloaded version of Nav-View on your computer and unzip the file. In the extracted folder click the "setup.exe" file.
3. Follow the setup wizard instructions. You will install NAV-VIEW and .NET 2.0.

⚠ WARNING

Do not reverse the power leads! Reversing the power leads to the IMU383 Series can damage the unit. Although reverse power protection is provided, ACEINNA is not responsible for resulting damage to the unit should the reverse voltage protection electronics fail.

Setting up NAV-VIEW

With the IMU383 Series product powered up and connected to your PC serial port, open the NAV-VIEW software application.

1. NAV-VIEW should automatically detect the IMU383 Series product and display the serial number and firmware version if it is connected.
2. If NAV-VIEW 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.
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

IMU383 Series product and the computer, check the power supply, and verify that the COM port is not occupied by another device.

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 IMU383 Series data. The remaining choices will be discussed in the following pages.

Data Recording

NAV-VIEW 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 (See Figure 15):

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.

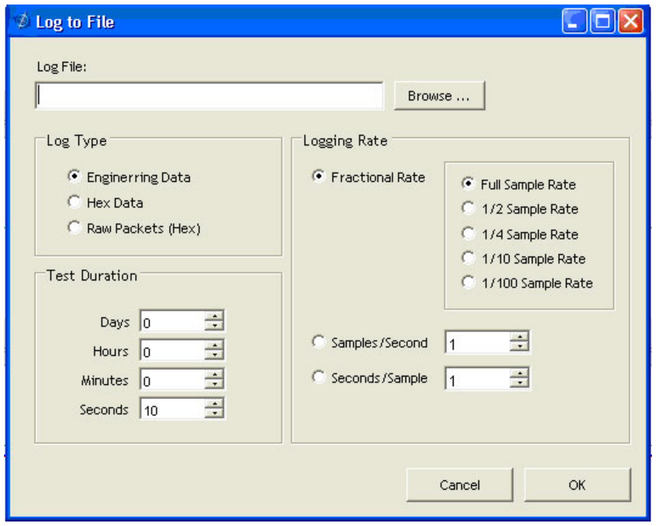
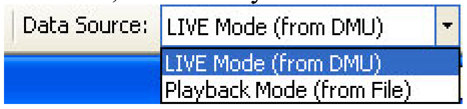


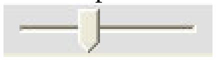
Figure 15. Log to File Dialog Screen

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.

Data Playback

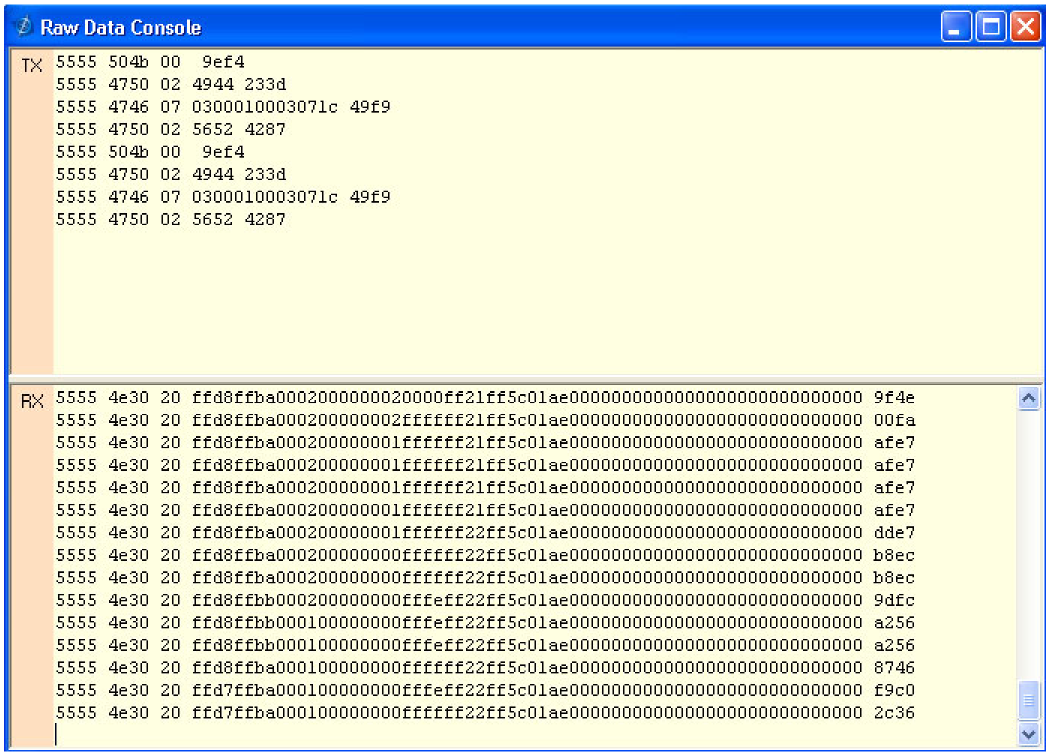
In addition to data recording, NAV-VIEW allows the user to replay saved data that has been stored in a log file.

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 also provides users with the ability to alter the start time for data playback. Using the  slide bar at the top of the page users can adjust the starting time.

Raw Data Console

NAV-VIEW 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 in Figure 16.



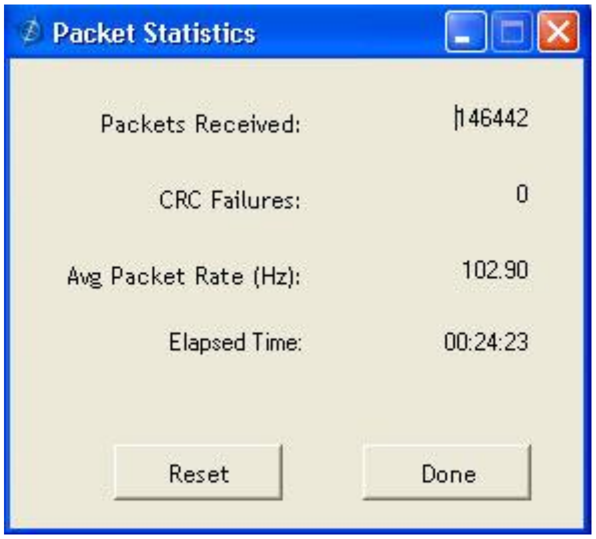
```
TX 5555 504b 00  9ef4
   5555 4750 02 4944 233d
   5555 4746 07 030001000307lc 49f9
   5555 4750 02 5652 4287
   5555 504b 00  9ef4
   5555 4750 02 4944 233d
   5555 4746 07 030001000307lc 49f9
   5555 4750 02 5652 4287

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 16 Raw Data Console

Packet Statistics View

Packet statistics can be obtained from the "View" menu by selecting the "Packet Statistics" option (See Figure 17). 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 17 Packet Statistics

Unit Configuration

The Unit Configuration window (See Figure 18) gives the user the ability to view and alter the system settings. 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; General, Advanced and BIT Configuration. The General tab displays some of the most commonly used settings. The Advanced and BIT Configuration menus provide users with more detailed setting information that they can tailor to meet their specific needs.

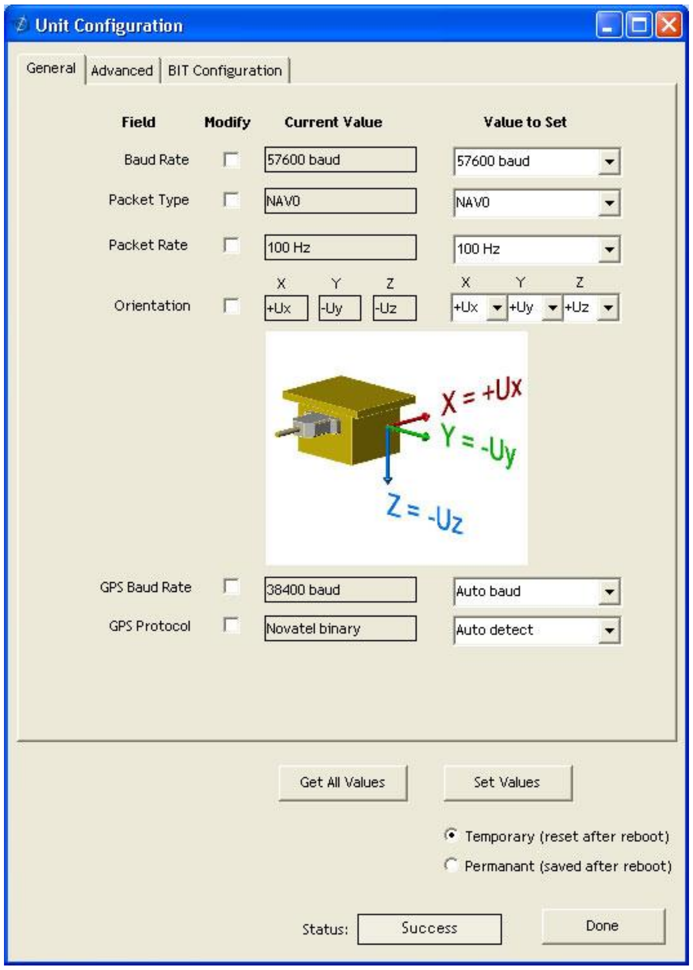
To alter a setting, simply select the check box on the left of the value that you wish to modify and then select the value using the drop down menu on the right side. Once you have selected the appropriate value, these settings can be set temporarily or permanently (a software reset or power cycle is 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.

IMPORTANT

Unit orientation selections must conform to the right hand coordinate system as noted in Section 3.1 of this user manual. Selecting orientations that do not conform to these criteria are not allowed.


Unit Configuration

General | Advanced | BIT Configuration

Field	Modify	Current Value	Value to Set
Baud Rate	☐	57600 baud	57600 baud
Packet Type	☐	NAV0	NAV0
Packet Rate	☐	100 Hz	100 Hz
Orientation	☐	X +Ux, Y -Uy, Z -Uz	X +Ux, Y +Uy, Z +Uz
GPS Baud Rate	☐	38400 baud	Auto baud
GPS Protocol	☐	Novatel binary	Auto detect

Get All Values | Set Values

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

Status: Success

Done

Figure 18 Unit Configuration

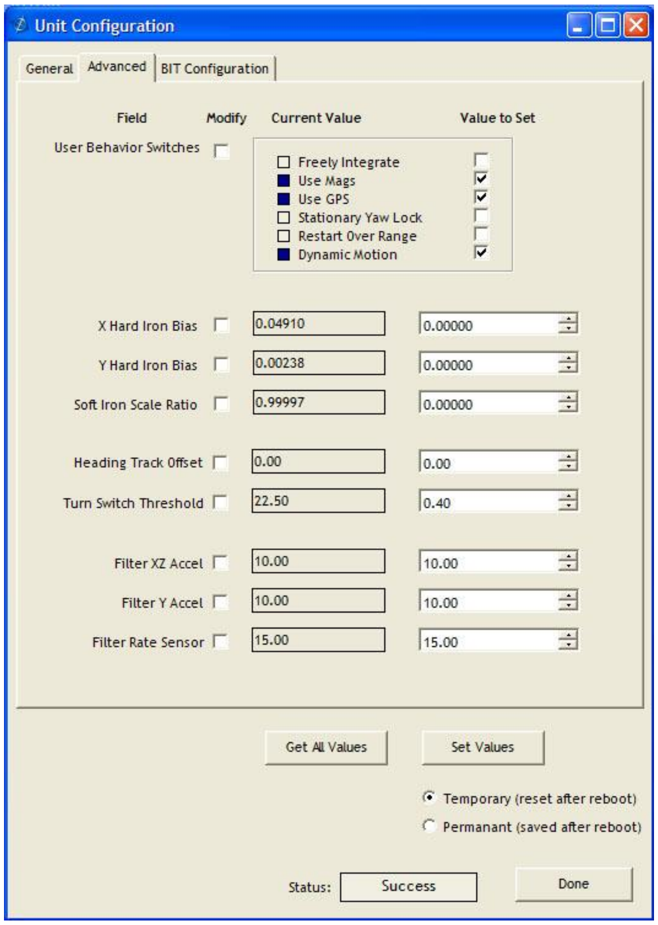
Advanced Configuration

Users who wish to access some of the more advanced features of NAV-VIEW and the IMU383 Series products can do so by selecting the "Advanced" tab at the top of the "Unit Configuration" window.

⚠ WARNING

Users are strongly encouraged to read and thoroughly understand the consequences of altering the settings in the "Advanced" tab before making changes to the unit configuration. These settings are discussed in detail in Chapter 4 below.

Behavior switches (unused with IMU) are identified at the top of the page with marked boxes. A blue box will appear if a switch has been enabled similar to Figure 19 below. The values can be set in the same manner as noted in the previous section. To set a value, users select the appropriate "Modify" checkbox on the left side of the menu and select or enable the appropriate value they wish to set. At the bottom of the page, users have the option of temporarily or permanently setting values. When all selections have been finalized, simply press the "Set Values" button to change the selected settings.


Unit Configuration

General | Advanced | BIT Configuration

Field	Modify	Current Value	Value to Set
User Behavior Switches	☐	☐ Freely Integrate	☐
		■ Use Mags	✓
		■ Use GPS	✓
		☐ Stationary Yaw Lock	☐
		☐ Restart Over Range	☐
		■ Dynamic Motion	✓
X Hard Iron Bias	☐	0.04910	0.00000
Y Hard Iron Bias	☐	0.00238	0.00000
Soft Iron Scale Ratio	☐	0.99997	0.00000
Heading Track Offset	☐	0.00	0.00
Turn Switch Threshold	☐	22.50	0.40
Filter XZ Accel	☐	10.00	10.00
Filter Y Accel	☐	10.00	10.00
Filter Rate Sensor	☐	15.00	15.00

Get All Values | Set Values

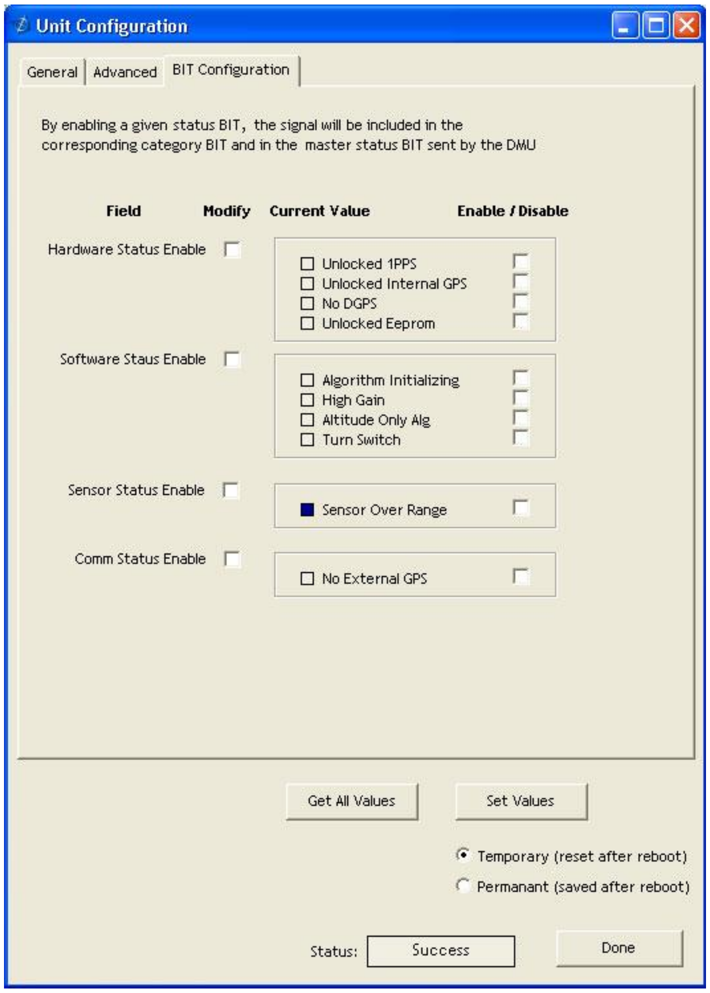
[✓ Temporary (reset after reboot)] Permanant (saved after reboot)

Status: Success

Figure 19 Advanced Settings

Bit Configuration

The third and final tab of the unit configuration window is "Bit Configuration" (See Figure 20). This tab allows the users to alter the logic of individual status flags that affect the masterStatus flag in the master BIT status field (available in most output packets). By enabling individual status flags users can determine which flags are logically OR'ed to generate the masterStatus flag. This gives the user the flexibility to listen to certain indications that affect their specific application. The masterFail and all error flags are not configurable. These flags represent serious errors and should never be ignored.


Unit Configuration

General | Advanced | BIT Configuration

By enabling a given status BIT, the signal will be included in the corresponding category BIT and in the master status BIT sent by the DMU

Field	Modify	Current Value	Enable / Disable
Hardware Status Enable	☐	☐ Unlocked 1PPS	☐
		☐ Unlocked Internal GPS	☐
		☐ No DGPS	☐
		☐ Unlocked Eeprom	☐
Software Staus Enable	☐	☐ Algorithm Initializing	☐
		☐ High Gain	☐
		☐ Altitude Only Alg	☐
		☐ Turn Switch	☐
Sensor Status Enable	☐	■ Sensor Over Range	☐
Comm Status Enable	☐	☐ No External GPS	☐

[✓ Temporary (reset after reboot)] Permanant (saved after reboot)

Status: Success

Figure 20 BIT Configuration

Read Unit Configuration

NAV-VIEW allows users to view the current settings and calibration data for a given IMU383 Series unit by accessing the "Read Configuration" selection from the "Configuration" drop down menu (See Figure 21). 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 ACEINNA's Support Staff.

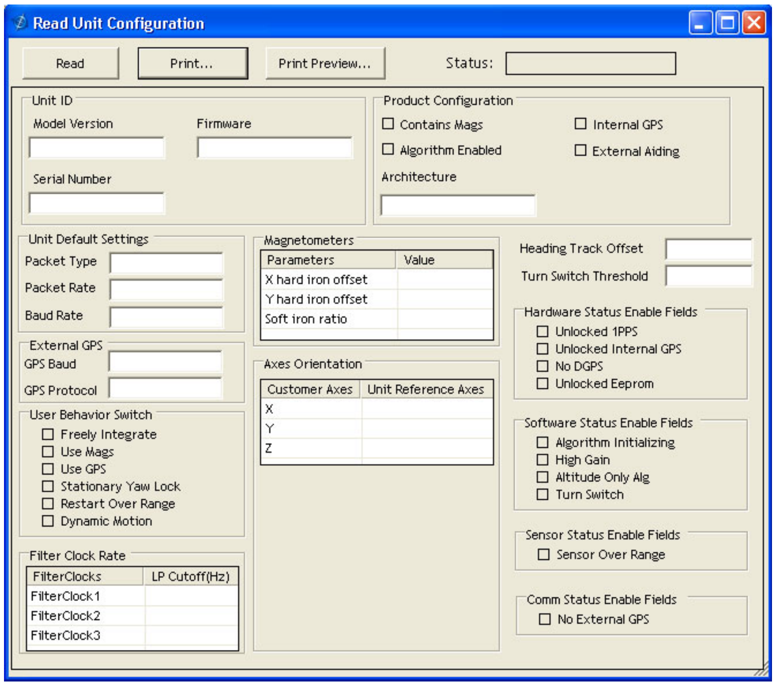
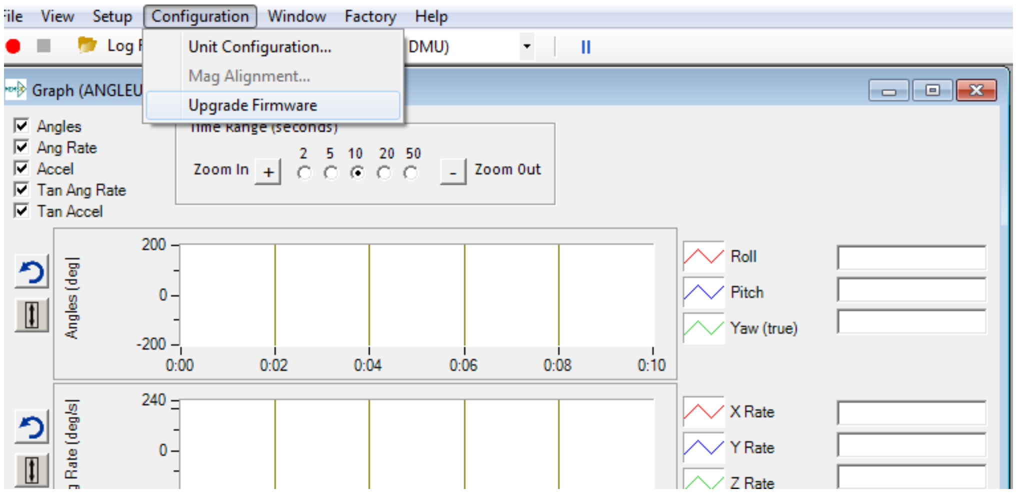


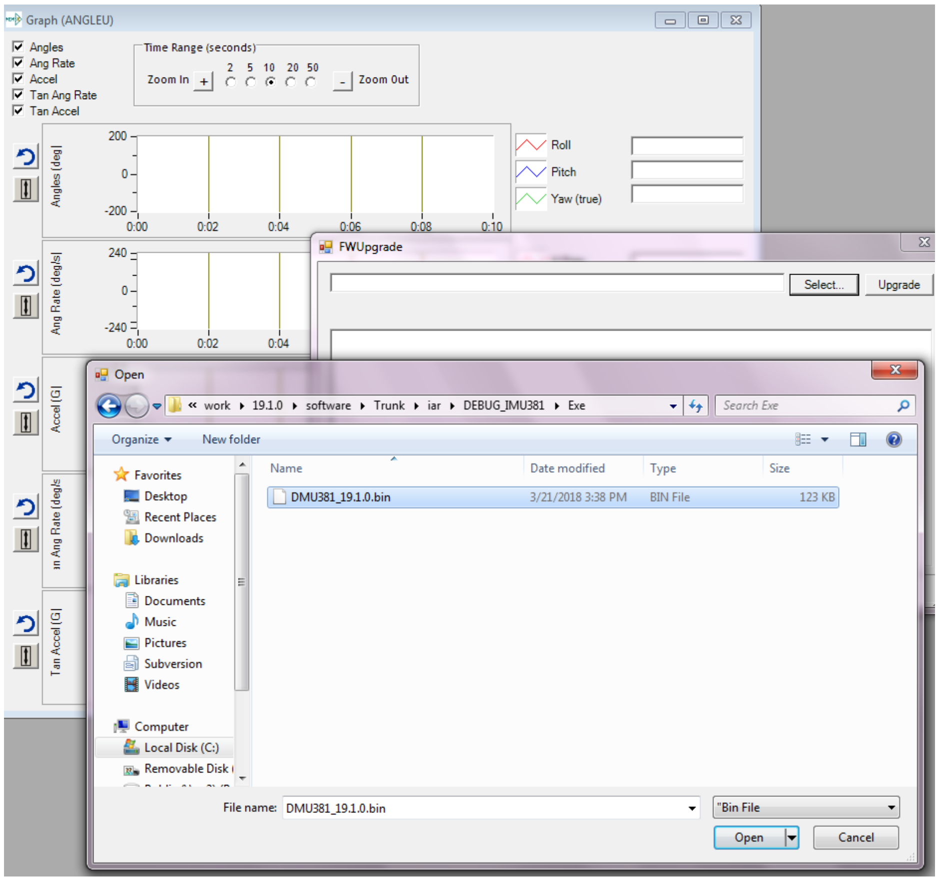
Figure 21 Read Configuration

Appendix A: Firmware Upgrade

Step 1, select Firmware upgrade from configuration menu.



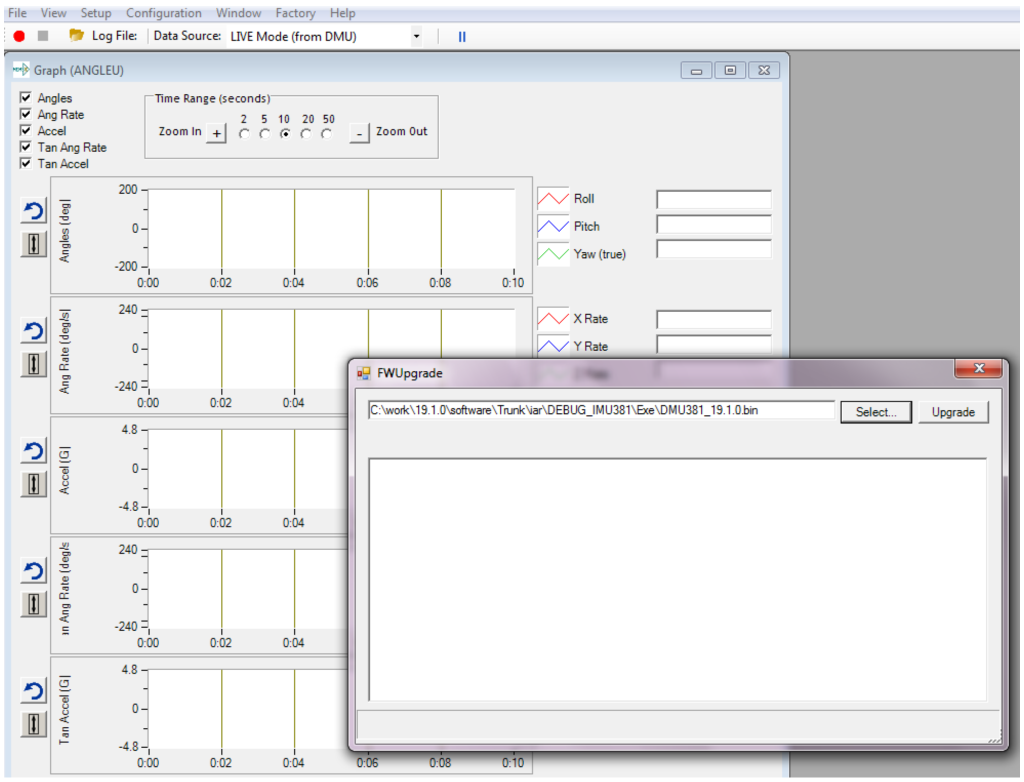
Step 2, On pop-up window, select a new version binary file by clicking SELECT button, then click Upgrade button.



Name	Date modified	Type	Size
DMU381_19.1.0.bin	3/21/2018 3:38 PM	BIN File	123 KB

File name: DMU381_19.1.0.bin

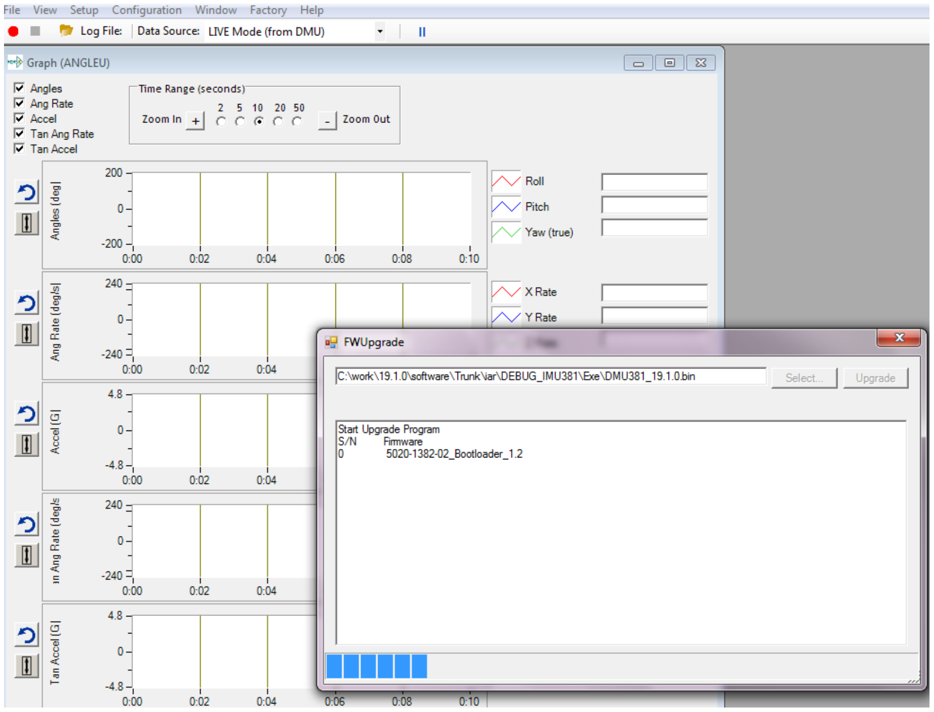
*Bin File



Data Source: LIVE Mode (from DMU)

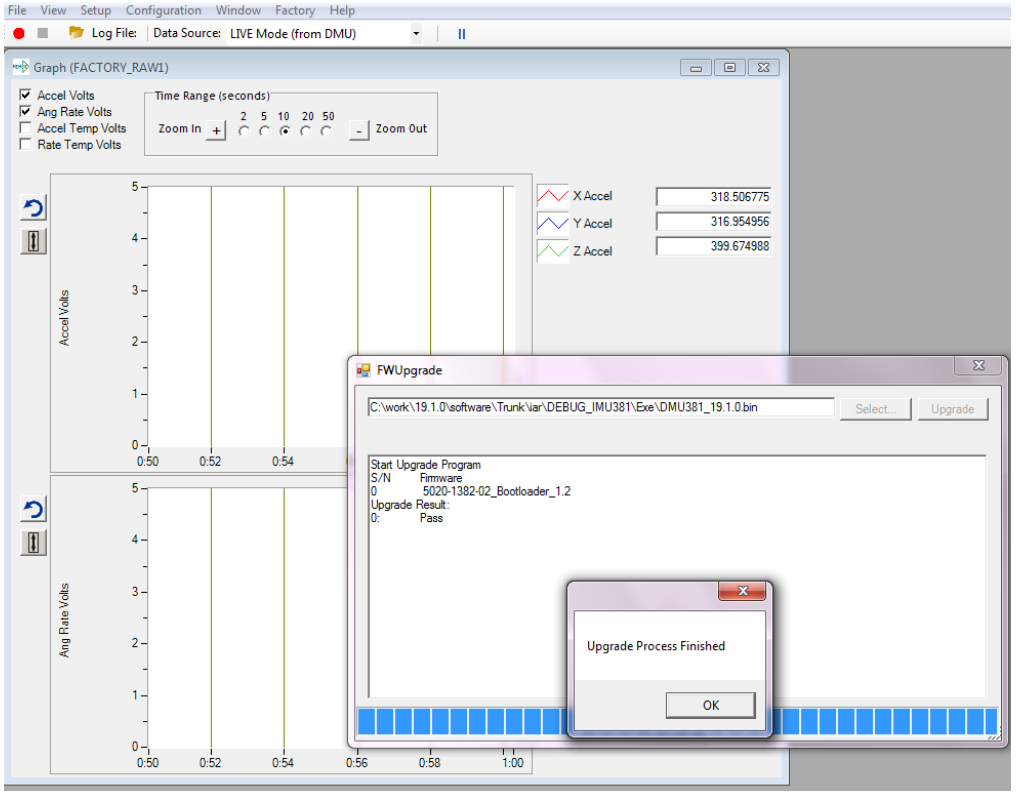
C:\work\19.1.0\software\Trunk\iar\DEBUG_IMU381\Exe\DMU381_19.1.0.bin

Step 3, wait for the process ongoing until a successful or failure message pops up.



```
C:\work\19.1.0\software\Trunk\iar\DEBUG_IMU381\Exe\DMU381_19.1.0.bin
```

```
Start Upgrade Program
S/N     Firmware
0       5020-1382-02_Bootloader_1.2
```



X Accel 318.506775
Y Accel 316.954956
Z Accel 399.674988

```
C:\work\19.1.0\software\Trunk\iar\DEBUG_IMU381\Exe\DMU381_19.1.0.bin
```

```
Start Upgrade Program
S/N     Firmware
0       5020-1382-02_Bootloader_1.2
Upgrade Result:
0:      Pass
```

Upgrade Process Finished

Appendix B: Sample Packet-Parser Code

Overview

This appendix includes sample code written in ANSI C for parsing packets from data sent by the IMU383 Series Inertial Systems. This code can be used by a user application reading data directly from the IMU383 Series product, or perhaps from a log file. Check at https://github.com/Aceinna for other reference code.

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.

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;

/*
 * ACEINNA 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 initial 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;
}
```