Coral AHRS^{TM} User's Manual

Version 1.5

Autonomous Reconnaissance Systems, Inc.™

http://www.autoreconsystems.com/

No guarantees are made towards the accuracy of the information in this manual. If any errors are found, please contact Autonomous Reconnaissance Systems, Inc. Customer Support.

Contents

Chapter 1

Introduction

Autonomous Reconnaissance Systems, Inc. thanks you for purchasing the Coral AHRS™ module or starter kit. We are committed to delivering dependable products and quality customer service. Occasionally, we will be releasing software updates in order to improve the quality of our products. If you would like to receive email notifications about any released updates and new product releases, please send an email to `news@autoreconsystems.com`.

Nicholas Holifield
President

Chapter 2

Quick Start

This guide will list and describe all the parts of your Coral AHRS™ Starter Kit, give you an overview of what your CD contains, provide step-by-step instructions on how to install our utility software and a brief guide on its usage.

While this section provides a useful guide for your new Coral AHRS™ module up and running, it is recommended that you read the rest of the manual, particularly the "System Integration" section, before attempting to use the AHRS in your own system.

2.1 Coral AHRS™ Starter Kit Contents

If you placed a custom order or bought a Coral AHRS™ module separately, the items included may differ. Please consult your packing slip.

1. **Coral AHRS™ Module**
   Containing the sensors and processing necessary to provide real-time attitude information, the Coral AHRS™ module is enclosed in a durable hard-anodized 7075-T6 Aluminum alloy case. For simplicity, power input and data output are on the same DB-9 connector.
2. **Data/Power Cable**
   Allows you to communicate with the Coral AHRS™ module across a standard RS-232 serial connection and to power the unit with the included power supply.
3. **AC Power Adapter**
   9 VDC power supply that has the proper connector to mate with the supplied data/power cable.
4. **Starter Kit Storage Case**
   This foam-filled, hard plastic case is provided to protect your Coral AHRS™

while in storage or transit. It is recommended that you keep the Coral AHRS™ module in this container whenever it is not in use or installed in your system.

5. **Software/Documentation CD**
   Contains a utility program to configure your Coral AHRS™ module and view its output and the LibCoral™ Software Development Kit for writing programs that make use of your module. Versions of this software are provided for Windows, Linux, and Mac OS X. An electronic copy of this manual is also provided on the CD.
6. **User's Manual**
   Provides information on configuring and using your Coral AHRS™ module.

2.2 Software Installation

2.2.1 Windows

1. Insert the Software CD into your CD-ROM drive. If the installation program does not start automatically, open the CD in Windows Explorer and open the `windows` folder. From this folder you can open the file `setup.exe` to begin execution of the installation program.
2. Follow the instructions in the installation program to select where you would like the utility program to be installed.
3. Unless you choose otherwise, an icon will be added to your desktop. Double-click the icon to start the utility program.

2.2.2 Linux

1. Insert the CD and mount it into your filesystem (the actual procedure for mounting a CD may differ amongst Linux distributions).
2. From a shell session or command line, change to the `linux` directory on the CD.
3. If you desire, you may copy the `coralutil` program file to another directory on your system.
4. To run the program, type `./coralutil` in the directory in which the program is installed.
5. Note that the Coral AHRS™ Utility program requires a version of the X11 Windowing System and version 3.3.2 or later of the Qt Application Development Toolkit.

2.2.3 Mac OS X

1. Insert the CD into your CD-ROM drive and select the CD icon from your Desktop.

2. Open the `mac` folder.
3. If you wish, you may copy the '`CoralUtility`' program icon to a different directory.
4. Double-click the '`CoralUtility`' program icon to run the utility.

2.3 Coral AHRS™ Utility Usage

1. Connect your Coral AHRS™ module to an available RS-232 serial port on your computer and connect the AC Power Adapter.
2. Start the Coral Utility program.
3. Select '`Connect...`' from the '`File`' menu.
4. Choose the correct RS-232 serial port device from the drop-down list. If the correct device is not listed, you may type it in.
5. Make sure '`Automatic`' is the selected option for Baud Rate, and that the '`Set Output Mode to Full Data?`' checkbox is checked, then click '`Ok`'.
6. The 3D display will now reflect the orientation of your unit and the text displays will read out the correct data.

Chapter 3

Principles of Operation

3.1 Coral AHRS™ Basics

The observation of orientation by a strapdown AHRS (which is rigidly installed, unlike gimballed models) cannot be performed directly. A combination of complementary sensor data must be used to estimate orientation. Gyroscopes, accelerometers, and magnetometers are installed in sets of three with their axes arranged orthogonally. These sensors are periodically polled by a microcontroller. The data are corrected for slight mechanical misalignments and other manufacturing differences. Then, a Kalman filter algorithm combines the sensor data iteratively into a statistically optimal estimate of attitude. The result is supplied over the unit's serial port.

3.2 Gyroscopes

Gyroscopes sense angular rate. The Coral AHRS™ uses MEMS (micro-electro-mechanical systems) angular rate sensors with a range of $\pm 300 degrees/sec$ (models with up to four times that maximum rate are available). These measurements are mathematically integrated to attitude. However, the predominant noise mode of gyroscopes is bias, a long time constant offset in the reading that is mostly related to temperature, but also drifts randomly by small amounts, making static calibration impossible. When constant offsets are integrated, they produce rapidly increasing error in the result. Large, expensive, navigation grade gyroscopes have stable enough bias to be used for mission durations of several weeks without recalibration. Gyroscopes small enough for the Coral AHRS™ do not provide meaningful attitude solutions by themselves after several minutes. Producing a usable attitude solution with small, inexpensive components requires a complementary approach, where gyroscopes are augmented with other sensors.

3.3 Accelerometers

The accelerometers in the Coral AHRS^{TM} are MEMS integrated circuits that sense the capacitative changes when forces bend a silicon structure towards and away from another silicon structure. The standard Coral AHRS^{TM} configuration has $\pm 1.7g$ ($1g = 9.8m/s^2$) accelerometers, allowing for sensing of Earth's gravity, and moderate dynamic movement. Accelerometers by themselves are sufficient for sensing pitch and roll in relatively static environments. In dynamic situations, linear acceleration pollutes the gravity measurement. However, there is no unbounded drift in the attitude estimate produced by accelerometers, because no integration is required. Because gyroscopes are relatively unaffected by linear accelerations but suffer from unbounded integration error, the two types of sensors are complementary. The accelerometers in the Coral AHRS^{TM} are factory calibrated for scale factor, misalignment, and bias errors. Accelerometers are primarily susceptible to high amplitude vibrations as a performance degrading factor.

3.4 Magnetometers

The Earth's magnetic field consists of field lines that, for most locations on the planet, point to magnetic north at some angle to vertical. These two quantities are referred to as magnetic declination and magnetic inclination, respectively. Magnetometers sense the Earth's magnetic field, allowing heading to be determined. Magnetic inclination can be accounted for with readings from the accelerometers, but the user must set the relevant declination value for the unit's area of use. The Coral AHRS^{TM} ships with magnetic declination initialized for the user's desired zip code.

- "Hard iron" magnetic effects are caused by magnetic objects located nearby the Coral AHRS^{TM} unit. If the orientation of these objects remains constant relative to the AHRS, the object will add its own constant magnetic field to the Earth's. The Coral AHRS^{TM} Utility provides a simple interface to remove this field from the unit's readings.
- "Soft iron" magnetic effects are caused by objects which distort the Earth's magnetic field. These produce changes in the magnetometer readings that are dependent on the AHRS's orientation. While no utility is currently provided to correct for soft iron effects, users are able to adjust the magnetometer calibration settings manually or programmatically to account for such effects.

The Coral AHRS^{TM} is enclosed by grounded metal 1/16th inch thick, so the magnetometers (which are packaged at the component level with some shielding, as well) are mostly unaffected by high frequency sources of EMI. Three magnetometers can give a complete reading of heading at all times, when coupled with pitch and roll information. Like accelerometers, they work best when augmented with a low noise gyroscope.

3.5 Attitude Filtering

A Kalman filter is used to combine the complementary sensor data into one attitude estimate. A Kalman filter is a statistically optimal iterative estimator. The attitude estimate is updated at a high rate by the gyroscopes. If this were to continue indefinitely, the estimate would degrade. However, many times a second, a corrective measurement of attitude derived from the accelerometers and magnetometers is applied to the attitude estimate. This also allows an estimate of gyro bias to be made, enhancing the estimates in between corrective measurements. The end result is that output from the Coral AHRS™ quickly responds to changes in attitude with a bounded error in its estimate.

Chapter 4

System Integration

4.1 Achieving Optimal Performance

The sensors used by the Coral AHRS^{TM} have well-defined limits. Additionally, the physical phenomena they use to provide information are prone to pollution by certain factors. It is important to consider that while the attitude estimate is highly accurate in ideal conditions, there are many ways this performance can be degraded. This guide will help you avoid those circumstances and get the most out of your Coral AHRS^{TM}.

4.1.1 Mechanical Considerations

The gravity reference provided by the accelerometers, which directly contributes to the pitch and roll estimates of the AHRS, is susceptible to interference from movements of all kind. While the gyroscope stabilization in the AHRS provides substantially improved performance over a simple static inclinometer, it has limits.

While short or slow accelerations will have negligible effect on the attitude solution, high amplitude vibrations or shocks can degrade performance. In addition, constant acceleration over a long period of time, such as a long, banking turn in an aircraft, will result in a slight roll offset. As long as this is planned for in applicable flight control algorithms, this does not pose a problem.

Vibration damping must be planned for as part of the user's system design. Many factors, such as the spectrum of the expected vibrations, must be known in order to design a vibration damper. In many designs, it will not be an issue, while in others, such as UAVs with two cycle engines, it is the main source of noise for the AHRS. Very high frequency vibrations may not be a concern due to the 40Hz analog filter employed for the accelerometers and gyros. However, user testing with the setup in question is required for assured functionality.

Even if these sources of noise are not entirely eliminated, the AHRS should only experience momentary degradation of performance, and return to stable operation after the noise is gone without user intervention.

4.1.2 Electrical Considerations

A reliable power supply is important to ensure reliable operation. You must be sure that the supply voltage to the AHRS will remain above the minimum level, even during peak loads by other parts of the system. The design of the Coral AHRS™ power supply is relatively immune to faults, noise, and ripples on the power input, but large amounts of power supply noise may degrade performance.

With regard to power supply efficiency, the switching regulator employed by the Coral AHRS is most efficient at lower input voltages. While using a 28VDC input will not result in the huge waste of power it would with a linear regulator, there is some reduction in efficiency – roughly 65% efficiency, as opposed to 80% efficiency at 12VDC.

4.1.3 Magnetic Considerations

Magnetometers sense the Earth's magnetic field in addition to any other magnetic field passing through the AHRS. Due to component level shielding, circuit board ground planes, and the grounded metal case of the AHRS, high frequency electromagnetic interference should be substantially attenuated.

DC magnetic fields like Earth's pass unobstructed to the magnetometers. Magnets, magnetized iron, and any inductive component can generate such a field. Efforts should be made to minimize these influences. Eliminating them entirely is very difficult (for example, the serial connector on the AHRS has a small magnetic signature). Wires carrying large amounts of current, electric motors, etc. should be isolated from the AHRS.

Magnetic fields with constant properties, such as those produced by magnets and ferrous mechanical components, are referred to as "hard iron". "Soft iron" interference is produced by materials which react to external magnetic fields, such as the Earth's, so that their effects are dependent on orientation.

At present, hard iron effects can be calibrated out. Soft iron effects are more rare, and are difficult to calibrate out. However, distorted heading readings only slightly affect roll or pitch estimates, and if special allowances must be made in your design for magnetic effects, soft iron effects can be accounted for in the output from the AHRS.

As with mechanical noise, transitory magnetic distortions will only temporarily affect the data produced by the AHRS. For information on hard iron calibration, see documentation on the Coral Utility program.

4.2 Calibration

4.2.1 Factory Calibration

Your Coral AHRS^{TM} module has been factory calibrated in order to correct scale factor, bias, and misalignment errors in the unit's gyroscopes, magnetometers, and accelerometers. These corrections are represented in the form of a vector that is subtracted from each group of sensor readings and a transformation matrix that is then applied. These calibration settings are completely user modifiable; the factory settings are stored on the unit and can be restored at any time. While most of these errors remain relatively constant, the gyros' biases are subject to drift. This drift is actively tracked by the Coral AHRS^{TM}'s filtering algorithm in order to provide a better estimate of gyro bias.

4.2.2 Magnetometer Calibration

The Coral AHRS^{TM}'s magnetometers are subject to errors arising from nearby objects' effects on the magnetic field. Two specific types of effects, hard iron and soft iron, can be compensated for by the Coral AHRS^{TM}. Hard iron effects are caused by magnetic objects which are placed at a constant orientation relative to the AHRS unit. These objects produce a constant magnetic field which appears as a bias to the magnetometers. Soft iron effects are caused by objects that modulate the external magnetic field. These errors appear as a distortion of the scale of the magnetic field in a given direction. Hard iron calibration is available via the Coral AHRS^{TM} Utility Software. Known hard iron effects can also be compensated for via vadjustment of the magnetometer bias calibration vector. Compensation for soft iron effects can be accomplished via modification of the magnetometer calibration matrix.

The local magnetic field in any region does not point to true north. Complex models are maintained by the United States National Oceanographic and Atmospheric Administration's Geomagnetism department. These models allow you to calculate the magnetic field's deviation from true north for a given latitude and longitude. This deviation is specified in the form of two angles, inclination and declination. Magnetic inclination, which represents the vertical deviation from true north, is automatically compensated for by the Coral AHRS^{TM}'s filter. Magnetic declination, which represents the horizontal deviation is corrected for once supplied to the Coral AHRS^{TM} unit. To find magnetic declination for your area, visit
`http://www.ngdc.noaa.gov/seg/geomag/jsp/Declination.jsp`.

4.3 Software

The LibCoral™ SDK is provided to accelerate integration of the Coral AHRS™ module into your systems' software. The SDK is provided for the Windows®, Mac OS X®, and Linux™operating systems. In the event that your system does not use one of these operating systems, you may also communicate with the Coral AHRS™ via its RS-232 based binary protocol. Full documentation of the Coral AHRS™ binary protocol is available in the appendices. The SDK is licensed under the BSD License. In short, this means that you may create and redistribute your own software using the SDK free of charge, in either a proprietary or "open source" fashion, as long as you maintain the appropriate copyright notice. See the license included with the SDK for full details.

4.4 I/O Formats

The Coral AHRS™ unit communicates via a RS-232 based binary protocol. The unit can communicate at baud rates of 4800, 9600, 19200, 38400, 57600, 115200, 230400, and 460800 bps at settings of 8 data bits, 1 stop bit, no parity, and no handshaking. While the default output rate of the unit is 100 Hz, the system features a configurable output divisor. Setting this output divisor causes the unit to output at any integer divisor rate between 1 and 255 of the original 100 Hz output rate. In this mode, the filter still runs at 100 Hz, providing the same level of accuracy.

The Coral AHRS™ provides attitude estimates in quaternion form (`http://mathworld.wolfram.com/EulerParameters.html`), Euler angles (`http://mathworld.wolfram.com/EulerAngles.html`), and rotation matrices (`http://mathworld.wolfram.com/RotationMatrix.html`). Quaternions are a simple mathematical means of representating arbitrary rotations. Unlike Euler angles, quaternions do not suffer from gimbal-lock issues or singularities. The LibCoral™ SDK has functions to convert Coral AHRS™ output from quaternions to either Euler angles or rotation matrices, and vice versa. The Coral AHRS™ can also transmit accelerometer, gyro, and magnetometer readings in either uncalibrated, raw format or calibrated format. The filter runs and outputs at the same speed regardless of what output mode is selected.
The Euler angle format the Coral AHRS uses has been modified somewhat to make it more suitable for real world purposes. Due to the singularity in the formulas for converting from a quaternion to Euler angles when pitch is ±90 degrees (straight up or down), the outputs become noisy and nearly useless. The modification results in a redefinition of the Euler angle formulas for values of pitch very close to ±90 degrees. In that case, roll is simply set to 0 degrees. This resolves the ambiguity resulting from the fact that heading and roll are the same axis in this orientation, and therefore heading will express all movement in this axis in vertical orientations. In this case, heading will be computed from the positive z vector (out of the bottom of the unit) when pitch is 90 degrees, and the negative z vector when pitch is -90 degrees. Therefore, if a level AHRS with a heading of 30 degrees pitches up 90 degrees, the heading will still read 30 degrees, and rotating it in the body roll axis will result in a change in heading.

4.5 I/O Setup

All input and output settings can be set with the Coral AHRS^{TM} Utility program, the LibCoral^{TM} SDK, or by directly communicating the settings using the Coral AHRS^{TM}'s native binary protocol. Consult the appendices for more detailed information on how to configure the unit using each method. All settings can be saved to the unit's EEPROM, allowing them to be loaded automatically upon unit startup.

Chapter 5

Maintenance and Service

5.1 Storage and Transport

The Coral AHRS^{TM} unit is installed in a machined aluminum enclosure. While this enclosure is designed to be both rugged and resilient, when the module is not installed in your system, we recommend that it be stored and transported in the carrying case supplied with the starter kit, or some similar case. It is also recommended that you ship the AHRS in this case if it ever requires factory service.

5.2 Troubleshooting

The Coral AHRS^{TM} is not designed to be internally user-serviceable. If you encounter any problems with your Coral AHRS^{TM} module not covered by the troubleshooting guide, contact Autonomous Reconnaissance Systems, Inc. Customer Service.

5.3 Customer Service

If you have any questions, problems, or suggestions, please contact Autonomous Reconnaissance Systems, Inc. Customer Service. We may be contacted by email at `support@autoreconsystems.com` and the contact phone number and postal address may be found at our website, `http://www.autoreconsystems.com`.

Chapter 6

Warranty

Except as specificially stated, Autonomous Reconnaissance Systems, Inc. makes no express warranties and no implied warranty of merchantability or fitness for a customer's particular purpose. The customer assumes full responsibility for the Coral AHRS^{TM} module's (the Product) operating conditions and for the installation of the Product supplied by Autonomous Reconnaissance System, Inc., a Delaware Corporation.

Autonomous Reconnaissance Systems, Inc. warrants for a period of one hundred eighty (180) calendar days from the shipment date of the Product to the original customer that the product is free from manufacturing defects. This warranty is void if the Product has been subjected to improper use or the enclosure of the Product has been opened by the customer. In the event of a manufacturing defect within the warranted period, the customer must return the product and a copy of the original invoice to Autonomous Reconnaissance Systems, Inc., who will repair or replace the unit at their discretion. All support for this product shall be provided at the discretion of Autonomous Reconnaissance Systems, Inc.

Except as provided herein, Autonomous Reconnaissance Systems, Inc. shall have no liability or responsibility to the customer or any other person or entity for loss or damages, caused or alleged to be caused, directly or indirectly by the Product or any software contained within. Autonomous Reconnaissance Systems, Inc. shall not be liable for any damages, direct, indirect, or consequential, arising from any breach of this warranty or connected with the sale, lease, and any other use or anticipated use of the Product or software contained within. Not withstanding the above limitations and warranties, Autonomous Reconnaissance Systems, Inc.'s liability shall be limited to the original amount paid by the customer for the Product. No action arising out of any claimed breach of this warranty or transactions under this warranty may be brought later than one (1) year after the cause of action has accrued or more than two (2) years after the original shipment date of the Product, whichever occurs earlier.

The terms and conditions of this warranty shall apply to Autonomous Reconnaissance

Systems, Inc. and the customer upon such time as a sale or lease of the Product to the customer occurs.

Any portion of this warranty that is deemed to be invalid or unenforcable does not affect the validity and enforcability of any other portion of this warranty.

Chapter 7

Contact Information

Autonomous Reconnaissance Systems, Inc. sales and support staff may be contacted via phone at (919) 309-4859. Our mailing address is:

Autonomous Reconnaissance Systems, Inc.
104 White Pine Dr.
Durham, NC 27705-7507

The support staff may be contacted via email at `support@autoreconsystems.com`. The sales staff is available at `sales@autoreconsystems.com`. For general information contact `info@autoreconsystems.com`. For the most recent contact information, visit our website at `http://www.autoreconsystems.com`.

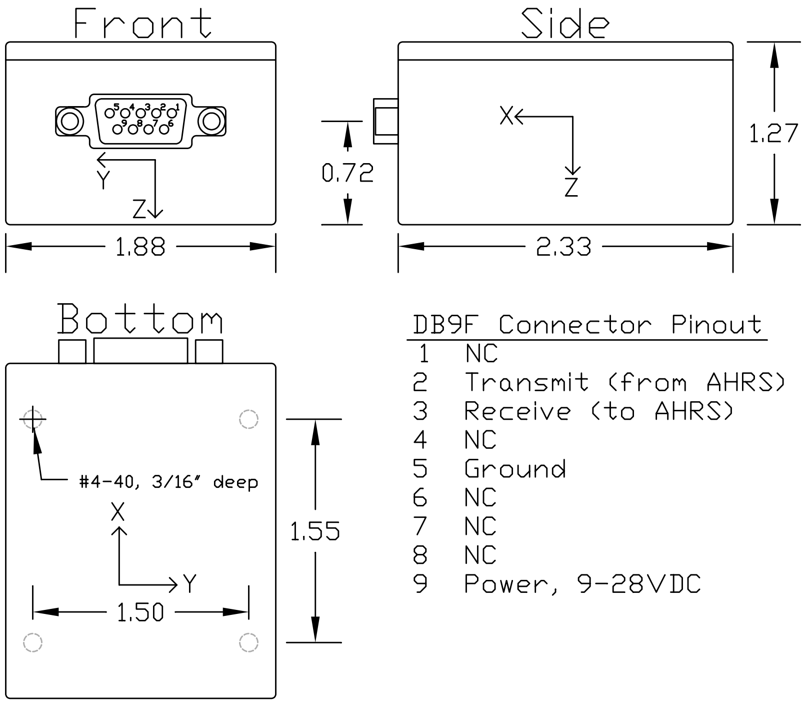
Appendix A

Technical Specifications

Output	
Output Data Mode	One filtered output (Euler Angles, Quaternion or DCM) and one sensor values output (Calibrated Values or Raw Values)
Output Data Rate	100 Hz, software configurable to lower rates with integer divisors
Data Format	Proprietary binary with checksum, see Appendix for more information
Turn-on Time	$<$2.5 seconds
Serial Protocol	RS-232
Serial Baud Rate	4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, configurable
Serial Port Settings	8 data bits, no parity, 1 stop bit, no handshaking
Orientation	
Range	Roll $\pm$180, Pitch $\pm$90, Heading 0-360
Static Accuracy, Roll and Pitch	1 degree
Static Accuracy, Heading	2 degrees
Dynamic Accuracy, Roll and Pitch	2 degrees rms
Dynamic Accuracy, Heading	3 degrees rms
Repeatability	0.2 degrees
Resolution	0.1 degrees
Sensors	
Gyroscopes	$\pm$ 300 degrees/sec, other configurations available up to $\pm$1200 degrees/sec
Accelerometers	$\pm$1.7g standard, $\pm$10g configuration available
Magnetometers	$\pm$10 Gauss
ADC Resolution	12 bits
ADC Sample Rate	100 Hz

Electrical	
Input Voltage	9-28 VDC
Input Current	72.5mA at 12 VDC
Power Consumption	870mW typical at 12 VDC
Physical	
Size (L x W x H)	2.33" (2.5" incl. connector) x 1.88" x 1.27"
	59mm (63.5mm) x 48mm x 32mm
Mounting Hole Thread	#4-40
Mount Hole Depth	3/16", 4.8mm
Mounting Hole Pattern	1.55" x 1.50"
Weight	3.74 oz, 106 grams
Case Material	7075 Aluminum, hard anodized
Connector	9 pin "D" sub-miniature female (DB9F)
Environmental	
Operating Temperature	-20 to +70 °C
Non-operating Temperature	-40 to +85 °C

Coral AHRS™ Mechanical Drawing
All quantities specified to ±0.005” tolerances.



Appendix B

Coral AHRS^{TM} Protocol Reference

B.1 Introduction

The Coral AHRS^{TM} module communicates with a simple protocol over its RS-232 serial port device. Each packet follows a standard format: each packet begins with a single byte of value 255 (FF in hexadecimal). The next byte represents the type of packet being sent. The next byte represents the length of the data section of the packet. The data section then follows. An 8-bit checksum (the 8-bit arithmetic sum of each of the preceding bytes in the packet) follows the data section. The same packet format is used for all communications to the Coral AHRS^{TM} module as well.

FF
Packet Type
Length
Data ⋮
8-bit Checksum

Figure B.1: Coral AHRS^{TM} Packet Format

B.2 Data Types

Packets sent to and from the Coral AHRS™ module contain data in several different formats. In all cases, data types which are longer than a single byte are sent with the most significant bytes first. The data types used by the Coral AHRS™ module are as follows:

u8 An unsigned 8-bit integer.

u16 An unsigned 16-bit integer.

f16 A 16-bit signed (two's complement) fixed point number. To convert to the proper scale, divide the number by 4096.

quat A vector of four **f16** values, representing the w, x, y, and z components of a quaternion.

matrix A 3x3 matrix of **f16** values, in column-major format.

vec3 A vector of three **f16** values, representing x, y, and z components or *roll*, *pitch*, and *heading* components.

string A null-terminated string of up to 60 ASCII characters.

The unit each data type is sent in is as follows:

Euler Angles Radians

Calibrated Gyros Radians/Sec

Calibrated Accelerometers G ($9.81m/s^2$)

Calibrated Magnetometers Local field units

Uncalibrated Sensors Analog-digital converter counts

B.3 Packets Sent from Coral AHRS™

B.3.1 CORAL_DATA_QUAT

Packet Type: 21h
Length: 10 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `quat Orientation`
  A quaternion describing the orientation of the Coral AHRS™ module.

This packet is sent when the system output mode has been set to `CORAL_QUAT`. It contains system time and quaternion orientation data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.2 CORAL_DATA_EULER

Packet Type: 22h
Length: 8 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `vec3 Euler Angles`
  A fixed-point vector describing the roll, pitch, and heading of the Coral AHRS™.

This packet is sent when the system output mode has been set to `CORAL_EULER`. It contains system time and euler angle orientation data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.3 CORAL_DATA_MATRIX

Packet Type: 23h
Length: 20 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `matrix Orientation`
  A 3x3 matrix describing the Coral AHRS™ module's orientation.

This packet is sent when the system output mode has been set to `CORAL_MATRIX`. It contains system time and matrix orientation data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.4 CORAL_DATA_SENSORS

Packet Type: 24h
Length: 20 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `vec3 Gyros`
  A vector with the calibrated output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the calibrated output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the calibrated output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_SENSORS`. It contains system time and calibrated sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.5 CORAL_DATA_QUAT_AND_SENSORS

Packet Type: 25h
Length: 28 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `quat Orientation`
  A quaternion describing the orientation of the Coral AHRS™ module.
- `vec3 Gyros`
  A vector with the calibrated output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the calibrated output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the calibrated output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_QUAT | CORAL_SENSORS`. It contains system time, quaternion orientation data, and calibrated sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.6 CORAL_DATA_EULER_AND_SENSORS

Packet Type: 26h
Length: 26 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `vec3 Euler Angles`
  A vector describing the roll, pitch, and heading of the Coral AHRS™ module.
- `vec3 Gyros`
  A vector with the calibrated output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the calibrated output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the calibrated output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_EULER | CORAL_SENSORS`. It contains system time, euler angle orientation data, and calibrated sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.7 CORAL_DATA_MATRIX_AND_SENSORS

Packet Type: 27h
Length: 38 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `matrix Orientation Matrix`
  A matrix describing the orientation of the Coral AHRS™ module.
- `vec3 Gyros`
  A vector with the calibrated output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the calibrated output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the calibrated output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_MATRIX | CORAL_SENSORS`. It contains system time, matrix orientation data, and calibrated sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.8 CORAL_DATA_RAW_SENSORS

Packet Type: 28h
Length: 20 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `vec3 Gyros`
  A vector with the raw output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the raw output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the raw output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_RAW_SENSORS`. It contains system time and raw sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.9 CORAL_DATA_QUAT_AND_RAW_SENSORS

Packet Type: 29h
Length: 28 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `quat Orientation`
  A quaternion describing the orientation of the Coral AHRS™ module.
- `vec3 Gyros`
  A vector with the raw output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the raw output of the X, Y, and Z accelerometers.

- `vec3 Magnetometers`
  A vector with the raw output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_QUAT | CORAL_RAW_SENSORS`. It contains system time, quaternion orientation data, and raw sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.10 CORAL_DATA_EULER_AND_RAW_SENSORS

Packet Type: 2Ah
Length: 26 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.
- `vec3 Euler Angles`
  A vector describing the roll, pitch, and heading of the Coral AHRS™ module.
- `vec3 Gyros`
  A vector with the raw output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the raw output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the raw output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_EULER | CORAL_RAW_SENSORS`. It contains system time, euler angle orientation data, and raw sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.11 CORAL_DATA_MATRIX_AND_RAW_SENSORS

Packet Type: 2Bh
Length: 38 bytes

Data:

- `u16 System Time`
  The value of the Coral AHRS™ module's system timer, in units of milliseconds.

- `matrix Orientation Matrix`
  A matrix describing the orientation of the Coral AHRS™ module.
- `vec3 Gyros`
  A vector with the raw output of the roll, pitch, and heading gyros.
- `vec3 Accelerometers`
  A vector with the raw output of the X, Y, and Z accelerometers.
- `vec3 Magnetometers`
  A vector with the raw output of the X, Y, and Z magnetometers.

This packet is sent when the system output mode has been set to `CORAL_MATRIX | CORAL_RAW_SENSORS`. It contains system time, matrix orientation data, and raw sensor data. These packets are sent at a constant rate equal to the standard system output rate divided by the output rate divisor.

B.3.12 CORAL_ID_STRING

Packet Type: 15h
Length: 60 bytes

Data:

- `string System ID`
  A string indicating the system type, firmware revision, and serial number of a Coral AHRS™ module.

This packet is sent in reponse to a `CORAL_REQUEST_ID` packet. The string returned contains the system firmware revision and serial number.

B.3.13 CORAL_CONFIGURATION

Packet Type: 1Ah
Length: 3 bytes

Data:

- `u8 Serial Speed`
  A single byte indicating the system's current serial speed setting.
- `u8 Output Mode`
  A single byte indicating the system's current output mode.
- `u8 Output Rate Divisor`
  A single byte indicating the systme's current output rate divisor.

This packet is sent in reponse to a `CORAL_REQUEST_CONFIGURATION` packet. It returns the current serial speed setting, output mode, and output rate divisor. The output mode setting is `1` for `CORAL_QUAT` output mode, `2` for `CORAL_SENSORS` output mode, `3` for `(CORAL_SENSORS | CORAL_QUAT)` output mode, `4` for `CORAL_RAW_SENSORS`, and `0` for no output. The serial speed setting corresponds to an actual baud rate according to the following chart:

- `0` - 4800 bps
- `1` - 9600 bps
- `2` - 19200 bps
- `3` - 38400 bps
- `4` - 57600 bps
- `5` - 115200 bps
- `6` - 230400 bps
- `7` - 460800 bps

B.3.14 CORAL_CALIBRATION

Packet Type: 1Ch
Length: 24 bytes

Data:

- `matrix Calibration`
  A matrix containing the scale and misalignment transformation for a given sensor group.
- `vec3 Bias`
  A vector containing the bias data for a given sensor group.

This packet is sent in response to a `CORAL_REQUEST_CALIBRATION` packet. It contains the calibration data for a given sensor group (gyros, accelerometers, magnetometers, or declination). Calibration adjustments for the accelerometers and magnetometers are performed by taking each vector of sensor data, subtracting the bias vector from it, and then multiplying it by the calibration matrix. This produces a vector of calibrated data. Unlike the accelerometers and magnetometers, the vector of gyro data is multiplied by the calibration matrix, then the bias vector is subtracted out. Unlike the other matrices, only the first two components of the declination matrix are used. The first component represents the cosine of half the declination angle and the second component represents the sine of the declination angle. The declination bias vector is ignored

B.3.15 CORAL_PONG

Packet Type: EEh
Length: 0 bytes

This packet is sent as a response to a `CORAL_PING` packet. It is used as a means to verify that the Coral AHRS™ unit is responding to messages.

B.4 Packets Sent to Coral AHRS™

B.4.1 CORAL_SET_OUTPUT_MODE

Packet Type: 01h
Length: 1 byte

Data:

- `u8 Output Mode`
  The desired output mode for the Coral AHRS™ module.

This packet changes the output mode on the Coral AHRS™ module. The output mode is a 4-bit field in which bits 0 and 1 choose an orientation data type (00 = None, 01 = Quaternion, 10 = Euler Angles, 11 = Orientation Matrix), and bits 2 and 3 choose a sensor data type (00 = None, 01 = Calibrated, 10 = Raw). If an invalid mode is selected, no data will be output until a proper mode is selected.

B.4.2 CORAL_SET_CALIBRATION

Packet Type: 02h
Length: 25 bytes

Data:

- `u8 Sensor Group`
  A byte selecting which group of sensors to adjust calibration settings for. A value of `0` corresponds to gyros. A value of `1` corresponds to accelerometers. A value of `2` corresponds to magnetometers. A value of `3` corresponds to declination information. Unlike the other matrices, only the first two components of the declination matrix are used. The first component represents the cosine of half the

declination angle and the second component represents the sine of the declination angle. The declination bias vector is ignored

- `matrix Calibration Matrix`
  A matrix with the desired calibration data for the sensor group.
- `vec3 Bias Data`
  A vector with bias data for the sensor group.

This packet adjusts calibration data for a given sensor group. Any changes made are lost when the power is cycled, unless a `CORAL_SAVE_SETTINGS` packet is sent. Calibration adjustments for the accelerometers and magnetometers are performed by taking each vector of sensor data, subtracting the bias vector from it, and then multiplying it by the calibration matrix. This produces a vector of calibrated data. Unlike the accelerometers and magnetometers, the vector of gyro data is multiplied by the calibration matrix, then the bias vector is subtracted out.

B.4.3 CORAL_CAPTURE_GYRO_BIAS

Packet Type: 03h
Length: 0 bytes

This packet requests that the Coral AHRS™ module use the current gyro readings as an estimate for its gyro biases.

B.4.4 CORAL_REQUEST_ID

Packet Type: 05h
Length: 0 bytes

This packet requests that the Coral AHRS™ module send its system identification string. The Coral AHRS™ module sends a `CORAL_ID_STRING` packet in response.

B.4.5 CORAL_RESTORE_USER_SETTINGS

Packet Type: 06h
Length: 0 bytes

This packet requests that the Coral AHRS™ module restore its calibration, serial baud rate, output mode, and output divisor settings to the settings saved in the user settings section of EEPROM.

B.4.6 CORAL_RESTORE_FACTORY_SETTINGS

Packet Type: 07h
Length: 0 bytes

This packet requests that the Coral AHRS™ module restore its calibration, serial baud rate, output mode, and output divisor settings to the settings set by the factory. CORAL_SAVE_SETTINGS must still be sent to retain the change after power cycling the module, however.

B.4.7 CORAL_SET_OUTPUT_RATE_DIVISOR

Packet Type: 08h
Length: 1 byte

Data:

- `u8 Output Rate Divisor`
  A single byte specifying the new output rate divisor.

This packet sets the output rate divisor. This specifies that the Coral AHRS™ is to only output orientation or sensor information at the standard system rate divided by the output rate divisor. This does not affect the accuracy of the output, as all calculations are still done at the standard system rate. If the serial speed is currently set at 4800 bps, the minimum output rate divisor allowed is 8. If the serial speed is currently set at 9600, the minimum output rate divisor allowed is 4. If the serial speed is currently set at 19200 bps, the minimum output rate divisor allowed is 2. Otherwise, the output rate divisor must be between 1 and 255.

B.4.8 CORAL_SET_SERIAL_RATE

Packet Type: 09h
Length: 1 byte

Data:

- `u8 Serial Rate Specifier`
  A single byte specifying the new serial baud rate.

This packet requests that the Coral AHRS™ module change the baud rate of its serial port. The `Serial Rate Specifier` argument reflects an actual baud rate as shown on the following chart:

- `0` - 4800 bps
- `1` - 9600 bps
- `2` - 19200 bps
- `3` - 38400 bps
- `4` - 57600 bps
- `5` - 115200 bps
- `6` - 230400 bps
- `7` - 460800 bps

B.4.9 CORAL_REQUEST_CONFIGURATION

Packet Type: 0Ah
Length: 0 bytes

This packet requests that the Coral AHRS™ module send its current output mode, serial baud rate, and output rate divisor. The module sends a `CORAL_CONFIGURATION` packet in response.

B.4.10 CORAL_REQUEST_CALIBRATION

Packet Type: 0Ch
Length: 1 byte

Data:

- `u8 Sensor Group`
  The group of sensors for which calibration data is requested. A value of `0` represents gyros. A value of `1` represents accelerometers. A value of `2` represents magnetometers. A value of `3` represents declination information.

This packet requests that calibration data for a given sensor group be returned. The Coral AHRS™ module responds with a `CORAL_CALIBRATION` packet.

B.4.11 CORAL_SAVE_SETTINGS

Packet Type: 0Fh
Length: 0 bytes

This packet requests that the Coral AHRS™ module save its current calibration, output mode, serial baud rate, and output rate divisor settings into the user settings section of EEPROM. These settings are loaded during the module's power-on cycle, and can be restored at any time by sending a `CORAL_RESTORE_USER_SETTINGS` packet.

B.4.12 CORAL_PING

Packet Type: DDh
Length: 0 bytes

This packet requests that the Coral AHRS™ module responds with a `CORAL_PONG` packet. This is used to verify that the module is active and responding.

Appendix C

LibCoral™ SDK Reference

C.1 Installation

C.1.1 Windows®

To install the LibCoral™ SDK on Windows, perform the following steps:

1. Copy the `libcoral.dll` file to your system's DLL directory (usually `C:\Windows\System`).
2. Copy the `libcoral.lib` file to your compiler's library directory.
3. Copy the `coral.h`, `serialport.h`, and `arpacket.h` header files to your compiler's include directory.

C.1.2 Linux™

To install the LibCoral™ SDK on Linux, perform the following steps:

1. Copy the `libcoral.a` and `libcoral.so.1.0` files to `/usr/lib` (or another directory of your choosing)
2. Run `ldconfig -n /usr/lib`
3. Create a symbolic link to the .so file using `ln -sf /usr/lib/libcoral.so.1 /usr/lib/libcoral.so`
4. Copy the `coral.h`, `serialport.h`, and `arpacket.h` header files to `/usr/include` or another location of your choosing.

C.1.3 Mac OS X®

To install the LibCoral™ SDK on Mac OS X, perform the following steps:

1. Copy the `LibCoral.framework` directory to `/Library/Frameworks`

C.2 General Usage

C.2.1 Visual Studio® 6.0/.NET

To use the LibCoral™ SDK in a Visual Studio project, add `libcoral.lib` to the list of Object/Library Modules in the Link tab under Settings in the Project menu. Include the `coral.h` header file, and you will be able to use the LibCoral™ functions in your program.

C.2.2 GCC on Linux™

To use the LibCoral™ SDK in a GCC project under Linux, add `-lcoral` to the compiler options either in your makefile or on the command line and include the `coral.h` header file.

C.2.3 GCC on Mac OS X

To use the LibCoral™ SDK in a GCC project under Mac OS X, add `-framework LibCoral -I/Library/Frameworks/LibCoral.framework/Headers` to the command line and include the `coral.h` header file.

C.2.4 XCode

To use the LibCoral™ SDK in an XCode project, choose `Add Frameworks` from the `Project` menu. Navigate to `/Library/Frameworks` and choose the `LibCoral.framework`

C.3 API Reference

C.3.1 CoralOpen

`int CoralOpen(const char *device, CoralSerialPort *port, int baud_rate)`

This function opens an RS-232 serial port device at the specified baud rate and tests the connection to make sure communication with the Coral AHRS^{TM} is possible.

Arguments:

`device` This argument represents the name of an RS-232 serial port device on the system. In Windows, these are of the form `COM1`, `COM2`, etc. In Linux, they are of the form `/dev/ttyS0`, `/dev/ttyS1`, etc. or `/dev/tts/0`, `/dev/tts/1` depending on your distribution. Serial port devices begin with `/dev/cu.` in Mac OS X.

`port` This argument is a pointer to a variable which will contain the resulting serial port device.

`baud_rate` This argument specifies the baud rate at which the serial port should be open. Valid rates are `4800`, `9600`, `19200`, `38400`, `57600`, `115200`, `230400`, and `460800`.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port was unable to be opened.

`-2` The specified baud rate was invalid.

`-3` The serial port was opened at the specified baud rate, but the Coral AHRS^{TM} module failed to respond.

C.3.2 CoralOpenAuto

`int CoralOpenAuto(const char *device, CoralSerialPort *port)`

This function opens an RS-232 serial port device using an autobauding procedure to determine at which baud rate communication with the Coral AHRS^{TM} is possible.

Arguments:

`device` This argument represents the name of an RS-232 serial port device on the system. In Windows, these are of the form `COM1`, `COM2`, etc. In Linux, they are of the form `/dev/ttyS0`, `/dev/ttyS1`, etc. or `/dev/tts/0`, `/dev/tts/1` depending on your distribution. Serial port devices begin with `/dev/cu.` in Mac OS X.

`port` This argument is a pointer to a variable which will contain the resulting serial port device.

Return Values:
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port was unable to be opened.

`-2` No acceptable baud rate was found.

C.3.3 CoralClose

```
int CoralClose(CoralSerialPort port)
```

This function closes a serial port device that was previously opened by the `CoralOpen` or `CoralOpenAuto` function. Once the port is closed, no more data can be read from or sent to the port and any data sent from the Coral AHRS™ module will be lost until the port is reopened.

Arguments:

`port` This argument is the serial port device you wish to close.

Return Values
This function returns `0` if the function succeeds and `-1` otherwise.

C.3.4 SetCoralTimeout

```
int SetCoralTimeout(CoralSerialPort port, int timeout)
```

This function sets the time limit for any functions which read data from the Coral AHRS™ module to complete.

Arguments:

`timeout` This argument is the desired timeout value, in units of 10 ms. If this value is less than `0`, the unit will have no timeout value set, and any calls to retrieve data from the Coral AHRS™ will block indefinitely. The default timeout value is 100 ms.

Return Values:
This function returns `0` if the function succeeds and `-1` otherwise.

C.3.5 GetCoralTimeout

```
int GetCoralTimeout(CoralSerialPort port)
```

This function returns the global timeout value.

Return Values:
This function returns the global timeout value, in units of 10 ms, if successful. If unsuccessful, it returns `-1`.

C.3.6 GetCoralBaudRate

`int GetCoralBaudRate(CoralSerialPort port)`

This function returns the baud rate at which the specified serial port device is opened.

Arguments:

`port` This argument is the serial port device on which to query the baud rate.

Return Values:
This function returns the baud rate at which the serial port device is open if successful or `-1` otherwise.

C.3.7 SetCoralOutputMode

```
int SetCoralOutputMode(CoralSerialPort port, int mode)
```

This function sets the type of data that the Coral AHRS™ module sends and that is retrieved via the GetCoralData function.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

`mode` This argument specifies the data output mode of the Coral AHRS™. Valid values are one of the following orientation data constants, one of the following sensor data constants, or a bitwise OR of one of each:

Orientation Data CORAL_QUAT for quaternion output, CORAL_EULER for euler output, or CORAL_MATRIX for matrix output.

Sensor Data CORAL_SENSORS for calibrated sensor values or CORAL_RAW_SENSORS for uncalibrated sensor values.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ module timed out.

`-3` The output mode specified was invalid.

C.3.8 GetCoralID

```
int GetCoralID(CoralSerialPort port, char *buffer)
```

This function retrieves the system ID string from the Coral AHRS^{TM} module.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS^{TM} module.

`buffer` This argument specifies a buffer to hold the retrieved string. It must be at least 65 characters long.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS^{TM} module timed out.

C.3.9 GetCoralData

`int GetCoralData(CoralSerialPort port, CoralData *data)`

This function retrieves orientation and sensor information from the Coral AHRS^{TM} module. The data retrieved depends on the output mode of the module, as set by the `SetCoralOutputMode` function.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS^{TM} module.

`data` This argument specifies a pointer to a CoralData structure to hold the retrieved data.

The CoralData structure holds both orientation and sensor information retrieved from the Coral AHRS^{TM} module. The structure has the following format:

```
struct CoralData
{
unsigned char datatype;

unsigned short systime;

double att[9];

double gyro[3];
double accel[3];
double mag[3];
}
```

`datatype` This data member specifies which data is actually contained in the structure. This should reflect the current output mode as specified with the `SetCoralOutputMode` function.

`systime` This member specifies the system time of the Coral AHRS^{TM} module when the packet was sent, in units of milliseconds.

`att` This member contains the attitude data from the Coral AHRS. If the output mode contains quaternions, the quaternion will be stored in values att[0]..att[3]. If the output contains euler angles, roll will be stored in att[0], pitch in att[1], and heading in att[2]. If the output mode contains an output matrix, the first row will be stored in att[0]..att[2], the second row in att[3]..att[5], and the third row in att[6]..att[8].

`gyro` This member contains the values read from the roll, pitch, and heading gyros respectively. These values are in units of radians/second.

`accel` This member contains the values read from the X, Y, and Z accelerometers respectively. These values are in units of g.

`mag` This member contains the values read from the X, Y, and Z magnetometers respectively. These values are in units of 10 microtesla. These values represent a field vector pointing towards geographic north, as opposed to a measurement of the actual local magnetic field.

All sensor values have been adjusted for misalignment errors, scale factor errors, bias errors, and magnetic declination and inclination effects.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS^{TM} module timed out.

C.3.10 GetCoralCalibration

```
int GetCoralCalibration(CoralSerialPort port, int sensor_group,
double matrix[3][3], double bias[3])
```

This function retrieves the calibration matrix and bias information associated with a particular sensor group.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

`sensor_group` This argument specifies which group of sensors to retrieve information for. Valid values are `0` for gyros, `1` for accelerometers, `2` for magnetometers, and `3` for the declination settings. Unlike the other matrices, only the first two components of the declination matrix are used. The first value represents the cosine of half the declination angle, while the second value represents the sine of half the declination angle. The bias vector for declination is ignored.

`matrix` This argument specifies a 3 x 3 matrix to receive the calibration matrix information.

`bias` This argument specifies an array to receive the bias information.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ module timed out.

`-3` An invalid sensor group was specified.

C.3.11 SetCoralCalibration

```
int SetCoralCalibration(CoralSerialPort port, int sensor_group,
double matrix[3][3], double bias[3])
```

This function sets the calibration matrix and bias information associated with a particular sensor group. These settings are stored in a separate area of the system's EEPROM from the factory calibration settings. Adjusting the calibration settings allows the application arbitrary linear transformations to the sensor data in order to account for system specific considerations. Adjusting the bias data of the magnetometers also allows the correction of hard iron effects.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

`sensor_group` This argument specifies which group of sensors to retrieve information for. Valid values are `0` for gyros, `1` for accelerometers, `2` for magnetometers, and `3` for declination data. Unlike the other matrices, only the first two components of the declination matrix are used. The first value represents the cosine of half the declination angle, while the second value represents the sine of half the declination angle. The bias vector for declination is ignored.

`matrix` This argument specifies a 3 x 3 matrix containing the calibration matrix information.

`bias` This argument specifies an array containing the bias information.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ module timed out.

`-3` An invalid sensor group was specified.

C.3.12 CoralCaptureGyroBias

`int CoralCaptureGyroBias(CoralSerialPort port)`

This function uses the current readings for the gyros as the estimated values for gyro bias. This function should be called when the unit is stationary.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ unit timed out.

C.3.13 GetCoralConfig

```
int GetCoralConfig(CoralSerialPort port, int *serial_baud_rate,
int *output_rate_divisor, int *output_mode)
```

This function retrieves configuration data from the specified Coral AHRS^{TM} unit.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS^{TM} module.

`serial_baud_rate` This argument specifies a pointer to a variable that will receive an integer representing the baud rate at which the Coral AHRS^{TM} module opens its serial port.

`output_rate_divisor` This argument specifies a pointer to a variable that will receive an integer representing the system's output rate divisor.

`output_mode` This argument specifies a pointer to a variable that will receive an integer representing the system's current output mode.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS^{TM} unit timed out.

C.3.14 SetCoralOutputDivisor

```
int SetCoralOutputDivisor(CoralSerialPort port, int output_divisor)
```

This function changes sets the output rate of the Coral AHRS^{TM} module to a fraction of its standard output rate.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS^{TM} module.

`output_divisor` The divisor of the standard output rate at which to output packets. This value must be greater than zero.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS^{TM} unit timed out.

`-3` An invalid output divisor was selected.

C.3.15 SetCoralSerialSpeed

`int SetCoralSerialSpeed(CoralSerialPort port, int serial_baud_rate)`

This function changes the baud rate of the Coral AHRS™ module's serial port. Note that using this function will likely cause the Coral AHRS™ module and the specified serial port device to be communicating at different baud rates. To correct this, close and reopen the serial port device with the new baud rate.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

`serial_baud_rate` This argument specifies the new serial speed at which the Coral AHRS™ module should communicate.

The `serial_baud_rate` argument will hold one of the following values representing one of the modes at which the Coral AHRS™ module is capable of operating:

- `0` - 4800 bps
- `1` - 9600 bps
- `2` - 19200 bps
- `3` - 38400 bps
- `4` - 57600 bps
- `5` - 115200 bps
- `6` - 230400 bps
- `7` - 460800 bps

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ unit timed out.

`-3` An invalid value for the `serial_baud_rate` argument was supplied.

C.3.16 CoralSaveSettings

```
int CoralSaveSettings(CoralSerialPort port)
```

This function saves the current configuration, calibration, and bias settings to the user settings section of EEPROM. These settings are loaded automatically on unit startup, and can be retrieved at a later time via the `CoralResetSettings` function. The user settings section of EEPROM is separated from the factory settings section. Factory settings can always be retrieved via the
`CoralResetSettings`.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS^{TM} module.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS^{TM} unit timed out.

C.3.17 CoralResetSettings

`int CoralResetSettings(CoralSerialPort port, int settings)`

This function resets the Coral AHRS™ module settings to the settings stored either in the user setttings section of EEPROM or the factory settings section of EEPROM. When changing settings it is possible that the Coral AHRS™ serial port may be reopened at a different baud rate than is currently being used by the specified serial port device, causing the two devices to be communicating at a different baud rate. To correct this, close the serial port device and reopen it at the correct baud rate.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

`settings` This argument specifies which group of settings to load. Specify a `0` here for user settings, or a non-zero value for factory settings.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ unit timed out.

C.3.18 CoralPing

`int CoralPing(CoralSerialPort port)`

This function verifies that the Coral AHRS™ unit is responding by sending a ping packet and waiting for a pong response packet.

Arguments:

`port` This argument specifies the serial port device that is connected to the Coral AHRS™ module.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` The serial port specified was invalid.

`-2` Communication with the Coral AHRS™ unit timed out.

C.3.19 CoralQuatToEuler

```
int CoralQuatToEuler(double quat[4], double euler[3])
```

This function converts the quaternion output from the Coral AHRS™ module into a roll, pitch, and heading measurement.

Arguments:

`quat` This argument specifies the quaternion to convert.

`euler` This argument specifies an array of double values in which to store the Euler angle values. Roll is stored in `euler[0]`, pitch is stored in `euler[1]`, and heading is stored in `euler[2]`.

Return Values

If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` An invalid quaternion value was supplied.

C.3.20 CoralQuatToMatrix

```
int CoralQuatToMatrix(double quat[4], double matrix[3][3])
```

This function converts the quaternion output from the Coral AHRS^{TM} module into a rotation matrix.

Arguments:

`quat` This argument specifies the quaternion to convert.

`matrix` This argument specifies a 3x3 matrix of double values in which to store the rotation matrix. The values are stored in a row-major format.

Return Values
If the function succeeds, the return value is 0. Otherwise, the return value is a negative number, indicating one of the following errors:

`-1` An invalid quaternion value was supplied.

C.4 License

The LibCoral™ Software Development Kit is Copyright ©2004, Autonomous Reconnaissance Systems, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Autonomous Reconnaissance Systems, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix D

Coral AHRS™ Utility Reference

D.1 Connecting to the Coral AHRS™

To connect to the Coral AHRS™ unit, select '`Connect`' from the '`File`' menu. Either select your serial port device from the drop-down list presented, or type it in manually. Select the device's baud rate. If you choose '`Automatic`', the Coral AHRS™ Utility will attempt to determine the correct baud rate for the module. This process may take several seconds. An option is provided to automatically switch the AHRS into full output mode. If this option is deselected, the Coral AHRS™ will remain in its current output mode. If that mode does not include orientation information, the display will remain static even though the utility will still be receiving data. Once you click '`Ok`', the utility will attempt to connect to the unit. Once the unit is connected, the display should update to reflect the data received from the unit.

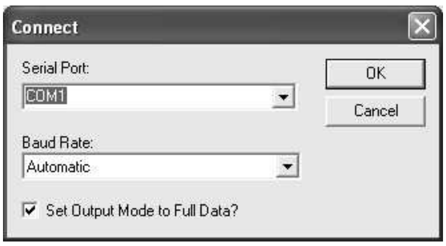


Figure D.1: Connection Dialog Box

D.2 Logging Data

The Coral AHRS™ Utility features basic data logging capability. The utility logs orientation and sensor data in a basic comma-separated values (CSV) format that is compatible with most major spreadsheet packages.

To begin logging, select '`Start`' from the '`Log`' menu. Enter the name of the file to which data should be saved and select which pieces of data you would like to log. Once you click '`Ok`', data will begin to be saved to that file. Choose '`Stop`' from the '`Log`' menu to end data logging.

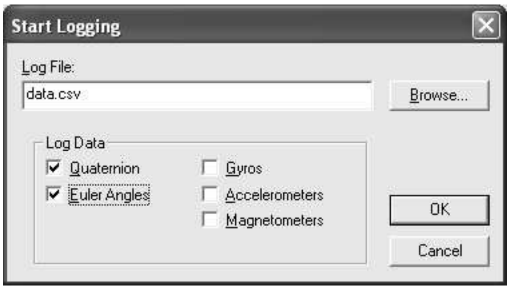


Figure D.2: Log Options Dialog Box

D.3 Coral AHRS™ Configuration

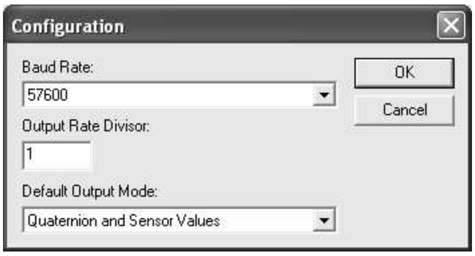


Figure D.3: Configuration Dialog Box

The Coral AHRS™ Utility allows you to adjust the module's baud rate, output mode, and output divisor. Choose '`Edit`' from the '`Configuration`' menu to modify these options. You can also save the current configuration and calibration settings to file by choosing '`Save to File`' from the '`Configuration`' menu. You can later load the settings by choosing '`Load from File`' from the '`Configuration`' menu. Choosing '`Save to EEPROM`' saves the current configuration and calibration settings to the user settings section in the unit's EEPROM. These settings will be loaded on unit startup.

You may also reset the unit's configuration and calibration settings to the factory defaults or the most recently saved user settings by choosing the appropriate option from the '`Reset`' submenu.

D.4 Adjusting Calibration

The Coral AHRS^{TM} Utility allows you to adjust the current calibration settings directly by choosing the '`Recalibrate`' option from the '`Tools`' menu. Any changes made are applied to the unit upon clicking '`Ok`'. Changes may be reversed by choosing the options from the '`Reset`' submenu in the '`Configuration`' menu.

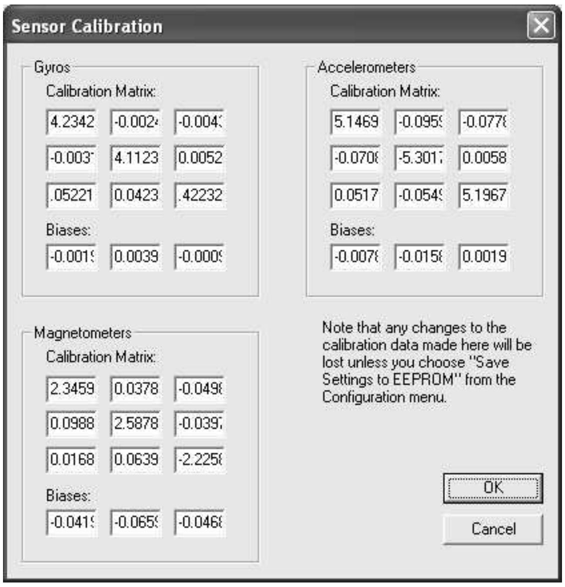


Figure D.4: Calibration Dialog Box

D.5 Hard Iron Calibration

The Coral AHRS^{TM} Utility provides a routine for calibrating out hard iron magnetic effects in your system. To use this routine, install the Coral AHRS^{TM} in your system, then choose '`Hard Iron Calibration`' from the '`Tools`' menu. Rotate your unit several times around all axes until the numbers displayed on the screen stop changing. Once you click '`Ok`', the updated information will be transmitted to the unit.

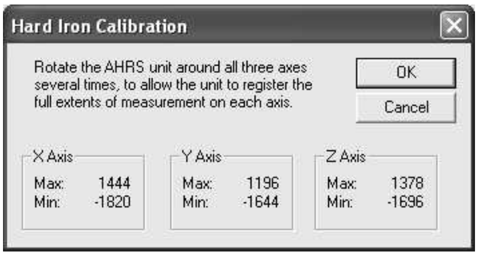


Figure D.5: Hard Iron Calibration Dialog Box

D.6 Gyro Bias Capturing

At any time when the unit is stationary, any non-zero reading on a gyro is due to bias errors. The Coral AHRS^{TM} filter actively tracks gyro bias but still benefits from an accurate measurement of bias taken when the unit is stationary. The Coral AHRS^{TM} Utility provides a simple interface for instructing the module to sample its gyros for a measurement of bias. To use this functionality, choose '`Gyro Bias Capture`' from the '`Tools`' menu. Make sure the unit is stationary, then select '`Ok`'. The unit will take a measurement of gyro bias.

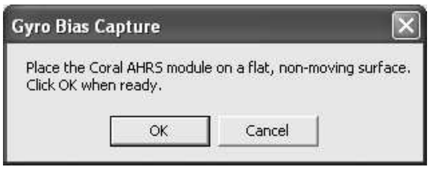


Figure D.6: Gyro Bias Capture Dialog Box

D.7 Magnetometer Declination Adjustment

In any given location, the Earth's magnetic field does not point directly north. The deviation in heading between true north and the direction of the magnetic field is known as declination. Many complex variables determine the declination at any given point, however, the United States National Oceanographic and Atmospheric Organization maintains a public geomagnetism model which can be used to determine magnetic declination for a given latitude, longitude, and elevation.

The Coral AHRS^{TM} Utility provides the capability to retrieve and set the declination setting for a Coral AHRS^{TM} unit. To access this utility, select '`Magnetometer Declination Adjustment`' from the '`Tools`' menu. Enter the desired declination value in the dialog box and choose '`Ok`' to set the value on the unit.

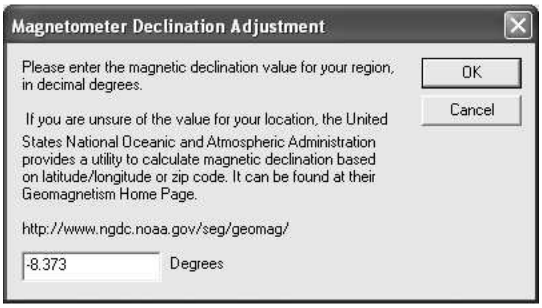


Figure D.7: Magnetometer Declination Dialog Box

D.8 License

The Coral AHRS^{TM} Utility is distributed under the terms of the GNU General Public License. The text of the license follows:

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of

warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations

under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.