Autonomous Reconnaissance Systems, Inc.
Coral AHRS™ LibCoral™ Reference

Contents

1 Installation

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.

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.

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`

2 General Usage

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.

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.

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.

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`

3 API Reference

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™ 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™ module failed to respond.

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.

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.

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^{TM} 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^{TM} will block indefinitely. The default timeout value is 100 ms.

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

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`.

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.

3.7 SetCoralOutputMode

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

This function sets the type of data that the Coral AHRS^{TM} 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^{TM} module.

`mode` This argument specifies the data output mode of the Coral AHRS^{TM}. 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^{TM} module timed out.

`-3` The output mode specified was invalid.

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.

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.

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^{TM} 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^{TM} module timed out.

`-3` An invalid sensor group was specified.

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^{TM} 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^{TM} module timed out.

`-3` An invalid sensor group was specified.

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^{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.

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.

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.

3.15 SetCoralSerialSpeed

`int SetCoralSerialSpeed(CoralSerialPort port, int serial_baud_rate)`

This function changes the baud rate of the Coral AHRS^{TM} module's serial port. Note that using this function will likely cause the Coral AHRS^{TM} 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^{TM} module.

`serial_baud_rate` This argument specifies the new serial speed at which the Coral AHRS^{TM} 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^{TM} 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^{TM} unit timed out.

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

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.

3.17 CoralResetSettings

```
int CoralResetSettings(CoralSerialPort port, int settings)
```

This function resets the Coral AHRS^{TM} 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^{TM} 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^{TM} 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^{TM} unit timed out.

3.18 CoralPing

`int CoralPing(CoralSerialPort port)`

This function verifies that the Coral AHRS^{TM} 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^{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.

3.19 CoralQuatToEuler

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

This function converts the quaternion output from the Coral AHRS^{TM} 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.

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.

4 License