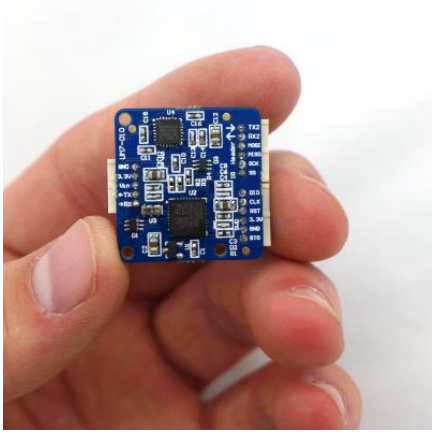
UM7 DATASHEET

Rev. 1.3 – Released 10/27/2014

INTRODUCTION



The UM7 is a 3rd-generation Attitude and Heading Reference System (AHRS) that takes advantage of state-of-the-art MEMS teschnology to improve performance and reduce costs. Like its predecessors, the UM7 combines triaxial accelerometer, rate gyro, and magnetometer data using a sophisticated Extended Kalman Filter to produce attitude and heading estimates.

FEATURES

High performance

- Excellent gyro bias stability over temperature
- Adjustable low-pass filter and EKF settings provide customizable performance for various applications.
- States and sensor data synchronized to GPS position and velocity using optional external GPS module.
- Supports alignment calibration and third-order bias and scale factor temperature compensation for accels, gyros, and magnetometer (optionally performed in-factory or by the end-user).
- Magnetometer soft and hard-iron calibration

Lower Cost

- State-of-the-art MEMS devices designed for mass-market consumer applications drastically lowers cost
- OEM version reduces overall cost, size, and weight

Ease of use

- Transmits data using human-readable NMEA strings, binary packets for higher efficiency, or a combination of both.
- Flexible communication architecture allows UM7 to transmit any combination of data at individually adjustable rates.
- Connects to the CHR Serial Interface software to allow for real-time plotting of sensor data, logging, device configuration, and magnetometer calibration.

UM7 DATASHEET

Rev. 1.3 – Released 10/27/2014

TABLE OF CONTENTS

DATASHEET REVISION HISTORY

Rev. 1.0 – Initial Release

Rev. 1.1 – Updated SPI bus information to more accurately describe minimum delay between SPI bytes. Updated absolute maximum rating information on header pins to identify 5V tolerant pins.

Rev 1.2 – Fixed some register definition problems. Changed the minimum period on the SPI clock to reflect actual limits.

Rev 1.3 – Added table-of-contents references for PDF export

SPECIFICATIONS

ATTITUDE AND HEADING SPECIFICATIONS

EKF Estimation Rate	500 Hz
Static Accuracy – Pitch and Roll	+/- 1 degree, typical*
Dynamic Accuracy – Pitch and Roll	+/- 3 degrees, typical*
Static Accuracy – Yaw	+/-3 degrees, typical*
Dynamic Accuracy – Yaw	+/- 5 degrees, typical*
Repeatability	0.5 degrees
Resolution	< 0.01 degrees

* Accuracy specifications depend on a variety of factors including the presence or lack of optional calibration and properties of the physical system being measured.

GYRO SPECIFICATIONS

Sensitivity change vs. temperature*	+/- 0.04% / deg C
Bias change vs. temperature*	+/- 20 deg/s from -40 C to +85 C
Rate noise density	0.005 deg/s/rtHz
Total RMS noise	0.06 deg/s-rms
Non-linearity	0.2 % FS
Dynamic range	+/- 2000 deg/s
Cross-axis sensitivity	+/- 2%
Nonlinearity	0.2%
Mechanical frequency – x-axis	33 kHz nominal
Mechanical frequency – y-axis	30 kHz nominal
Mechanical frequency – z-axis	27 kHz nominal

* Maximum change without calibration. Improved performance can be obtained with calibration.

ACCELEROMETER SPECIFICATIONS

Sensitivity change vs. temperature*	+/- 0.02% / deg C
Bias change vs. temperature (X,Y)*	+/- 0.75 mg/deg C
Bias change vs. temperature (Z)*	+/- 1.50 mg/deg C
Rate noise density	400 ug / rtHz
Dynamic Range	+/- 8 g

* Maximum change without calibration. Improved performance can be obtained with calibration.

MAGNETOMETER SPECIFICATIONS

Dynamic range	+/- 1200 uT
Initial scale factor tolerance*	+/- 4%
Initial bias tolerance*	+/- 300 uT
Dynamic range	+/- 1200 uT

* Initial bias and scale factor errors are removed via soft and hard-iron calibration. Temperature-dependent bias and scale factor errors removable with optional calibration.

OTHER SPECIFICATIONS

Vin	5.0V nominal
Communication	3.3V TTL UART, 3.3V SPI bus
Baud Rates Supported	9600, 14400, 19200, 38400, 57600, 115200, 128000, 153600, 230400, 256000, 460800, 921600. Default 115200
Power Consumption	~ 50 mA at 5.0V
Operating Temperature	-40C to +85C
Dimensions	1.06" x 1.02" x 0.26" (27mm x 26mm x 6.5mm)

Weight	0.4 oz (11 grams)
Data Output Rate	1 Hz to 255 Hz, binary packets 1 Hz to 100 Hz, NMEA packets
Output Data	Attitude, Heading (Euler Angles) Attitude quaternion GPS altitude, position, velocity (w/external GPS) GPS position in meters from home configurable home position. Raw mag, accel, gyro data Calibrated mag, accel, gyro, temperature data

ABSOLUTE MAXIMUM RATINGS

ABSOLUTE MAXIMUM MECHANICAL RATINGS

Max Acceleration	3000g for 0.5 ms 10000 g for 0.1ms
Operating Temperature Range	-40C to +85 C
Storage Temperature Range	-40C to +125 C

ABSOLUTE MAXIMUM ELECTRICAL RATINGS

Supply Voltage (Vin)	-0.3 V to +6.5 V
Maximum voltage on any input (except Vin, TX, and RX)	3.5V
Maximum voltage on TX and RX pins on main IO header	5.5V
Minimum voltage on any pin	-0.2V

*Operating at or beyond maximum ratings for extended periods will damage the device.

ESD CHARACTERISTICS

TVS ESD Withstand Voltage IEC61000-4-2 (Contact)*	+/- 15kV
TVS ESD Withstand Voltage IEC61000-4-2 (Air)*	+/- 30kV
TVS Peak ESD discharge current	2 A
Electrostatic discharge voltage (Class 2, human body model)*	2000 V
Electrostatic discharge voltage (Class II, charge device model)*	500 V

* ESD ratings for all external connector pins. ESD performance for contact with electronics parts on OEM device not characterized.

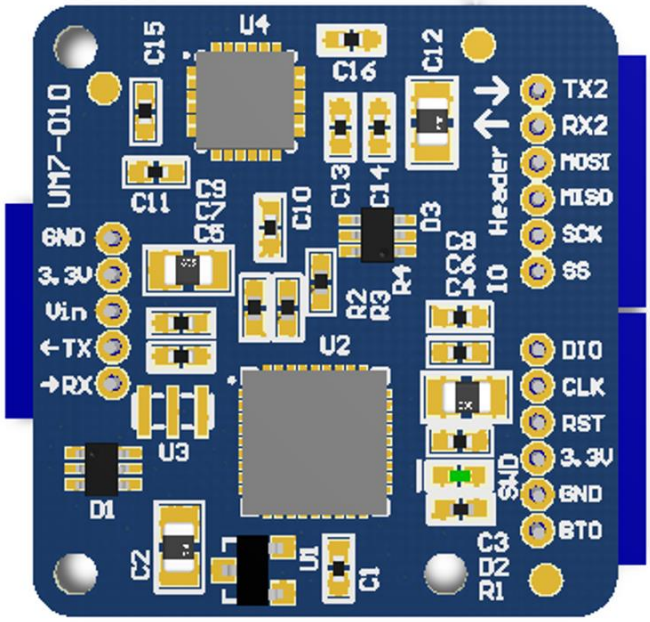
ELECTRICAL CHARACTERISTICS

OPERATING CHARACTERISTICS

Supply Voltage	+4.0 V to +5.5V
Operating current	~ 50mA at 5.0V
IO Logic Level	+3.3V TTL
Input logic low threshold	1.16V Max
Input logic high threshold	2.15V Min
3.3V pin, max output current (continuous, Vin = 5.0V, Ta = 25 C)	180mA
3.3V pin, max output current (continuous, Vin = 5.0V, Ta = 50 C)	120mA
3.3V pin, max output current (continuous, Vin = 5.0V, Ta = 80 C)	45mA

PINOUT

Main IO Header



IO Expansion Header

Programming Header

MAIN IO HEADER

Mating connector: JST part number ZHR-5(P)

Pin Name	Description
GND	Supply ground
3.3V	3.3V output from onboard LDO. May also function as a 3.3V supply input if Vin is left unconnected.
Vin	Supply voltage input. 5.0V nominal.
TX	UART TX output (3.3V TTL)
RX	UART RX input (3.3V TTL)

* The Main IO Header contains all pins required to operate the UM7 normally. The IO Expansion and Programming headers are optional.

IO EXPANSION HEADER

Mating connector: JST part number ZHR-6(P)

Pin Name	Description
TX2	Secondary UART TX output (3.3V TTL). Can be used to interface with external GPS or Bluetooth module. If connected to GPS, this pin can serve as the PPS input to synchronize the system clock with UTC time (configured in the CREG_MISC_SETTINGS register). Leave unconnected if not used.
RX2	Secondary UART RX input (3.3V TTL). Can be used to interface with external GPS or Bluetooth module. Leave unconnected if not used.
MOSI	SPI interface MOSI pin (3.3V TTL). Leave unconnected if not used.
MISO	SPI interface MISO pin (3.3V TTL). Leave unconnected if not used.
SCK	SPI interface clock (3.3V TTL). Leave unconnected if not used.
SS	SPI interface slave-select pin (3.3V TTL). Leave unconnected if not used.

PROGRAMMING HEADER

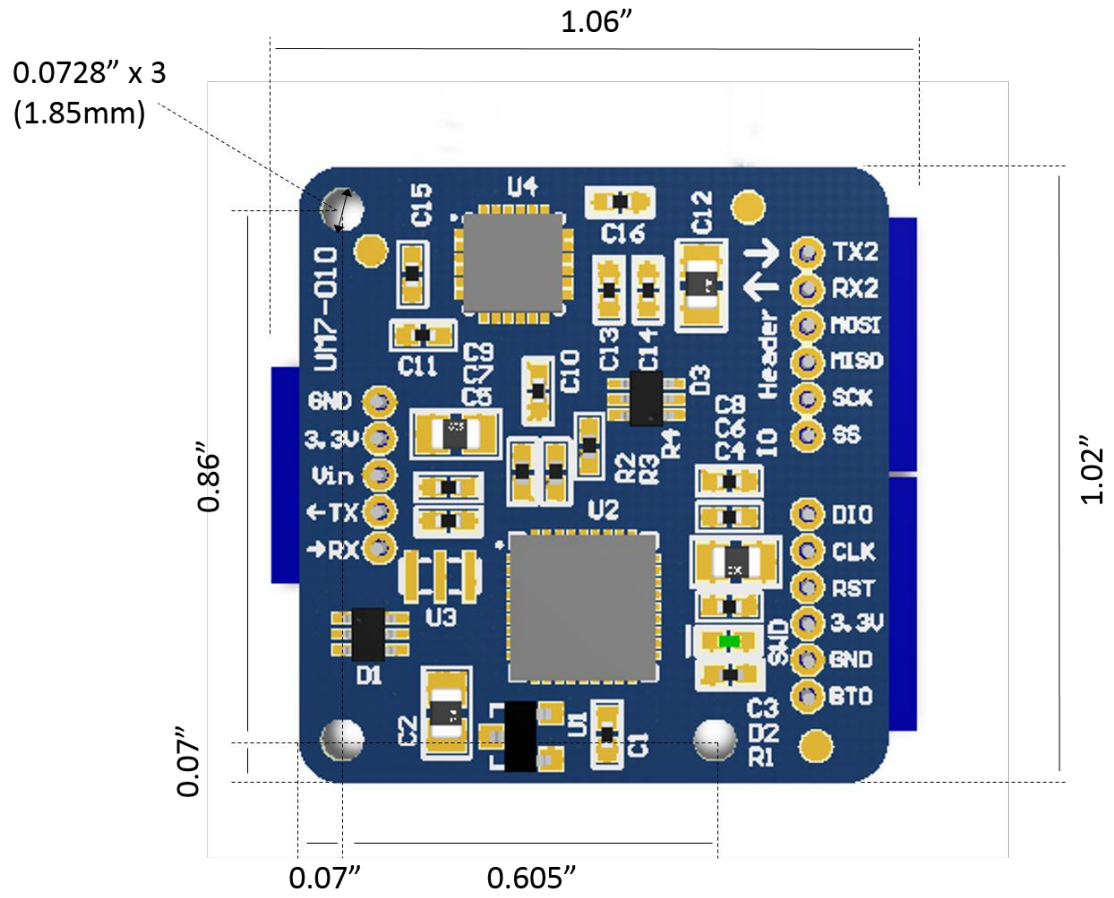
Mating connector: JST part number ZHR-6(P)

Pin Name	Description
DIO	Pin for factory programming. Leave unconnected if not used.
CLK	Pin for factory programming. Leave unconnected.
RST	Pin for factory programming. Leave unconnected.
3.3V	3.3V out
GND	Supply GND
BT0	Boot0 pin. Short this pin to +3.3V pin before powering the device to activate bootloader for firmware programming.

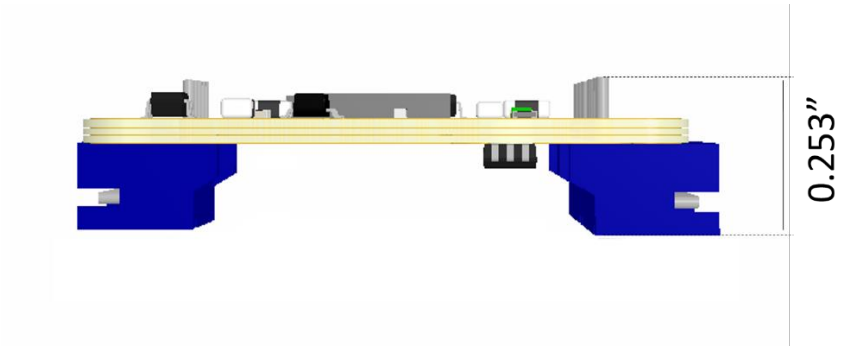
MECHANICAL DRAWING

Dimensions are in inches. Unless otherwise noted, dimensional tolerances are within +/- 0.002".

TOP VIEW



SIDE VIEW



SPI COMMUNICATION

The UM7 SPI bus operates at a +3.3V logic level. The UM7 is a slave on the bus, remaining inactive unless queried by a master device. Since the SPI bus will not always be used, bus inputs (MOSI, SCK, SS) are pulled to +3.3V internally. This prevents noise from being registered by the UM7 as attempts to communicate with the sensor.

The UM7 SPI clock (SCK) is active high, with data clocked in on the first rising edge (usually labeled SPI Mode 0 on microcontrollers and other devices). The master should place its data on the MOSI line on the clock falling edge.

The maximum SPI clock rate is 10 MHz. However the UM7 needs at least 5 microseconds between bytes to copy the next byte into the SPI transmit register. For high clock rates, this means that a delay must be added between consecutive bytes for correct operation.

All SPI operations begin when the master writes two control bytes to the bus. The first byte indicates whether the operation is a register read (0x00) or a write (0x01). The second byte is the address of the register being accessed.

A read operation is performed by writing the control byte 0x00 to the MOSI line, followed by the address of the register to be read. During the next four transfers, the UM7 will write the contents of the register to the MISO line starting with the most-significant byte in the register as shown in Figure 2 - Single Register Read Operation. The master should pull the MOSI line low during the remainder of the read.

A read operation can be extended to read more than one register at a time as shown in Figure 3 - Multiple Register Read Operation to initiate the batch read, the master should write the address of the next desired register to the MOSI line while last byte of the previous register is being transmitted by the UM7.

A write operation is performed by writing the control byte 0x01 to the MOSI line, followed by the address of the register to modify. During the next four transfers, the UM7 will read the data from the MOSI line and write it to the specified register. During a write operation, the UM7 will pull the MISO line low to indicate that it is receiving data. There is no batch write operation. The structure of a write operation is illustrated in Figure 4 - Single Register Write Operation.

Commands are initiated over the SPI bus by sending a **write** operation to the command address, with all four data bytes at zero. The UM7 will not report that a command was executed

over the SPI bus except for during a GET_FW_REVISION command, which causes the firmware revision to be sent over the MISO line during the next four byte transfers on the bus.

For example, to execute a zero rate gyros command over the SPI bus, the following data bytes should be sent over the bus: 0x01 0xAC 0x00 0x00 0x00 0x00.

Similarly, to query the UM7 to get the firmware revision over the SPI bus, the following sequence should be used: 0x01 0xAA 0x00 0x00 0x00 0x00. The UM7 will send the firmware revision over the MISO line during the last four byte transfers.

Figure 1 - SPI Bus Timing

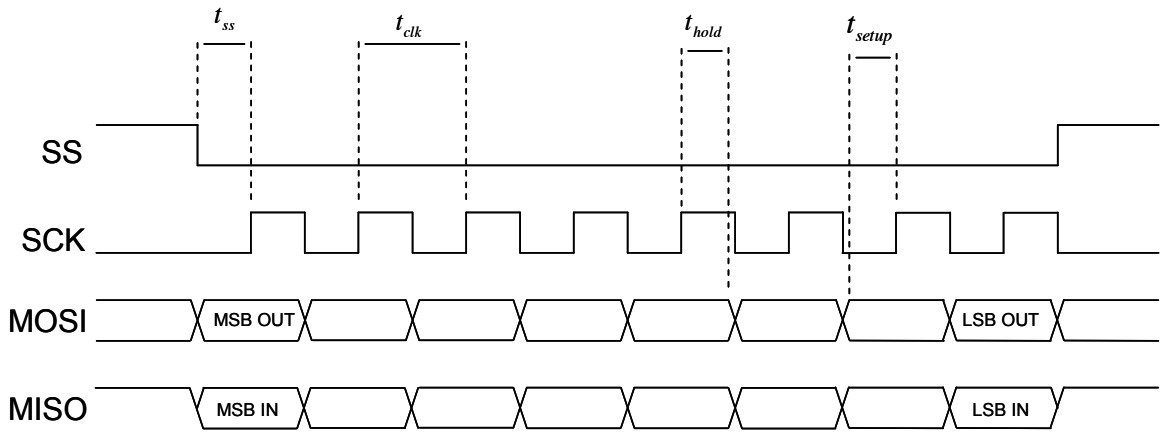


Table 1 - SPI Bus Timing

Name	Description	Min	Max
t_{ss}	Slave-select setup time	1 us	NA
t_{clk}	Clock period	0.1 us	NA
f_{clk}	Clock frequency	NA	10 Mhz
t_{hold}	Data hold time	10 ns	NA
t_{setup}	Data setup time	10 ns	NA

Figure 2 - Single Register Read Operation

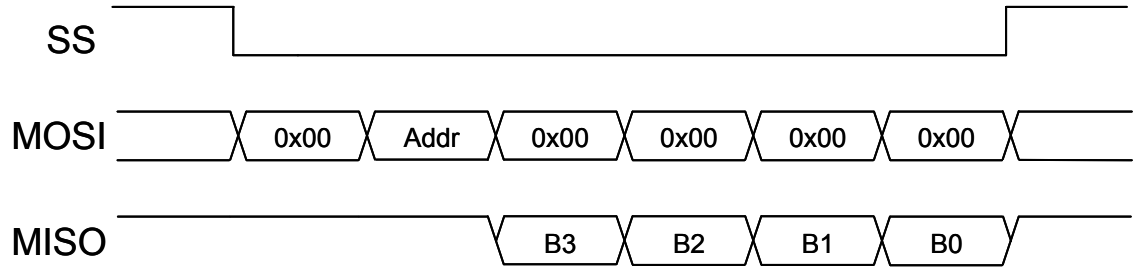


Figure 3 - Multiple Register Read Operation

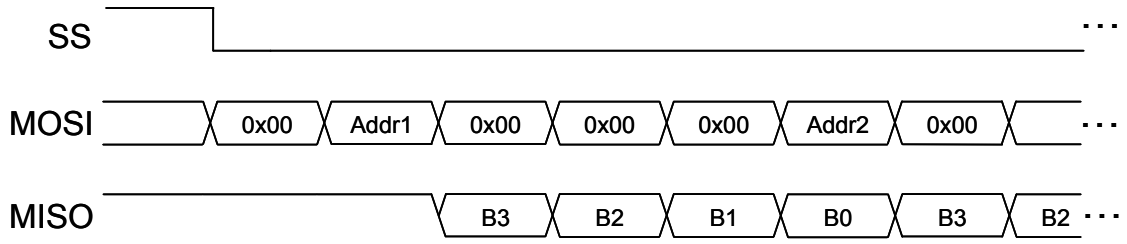
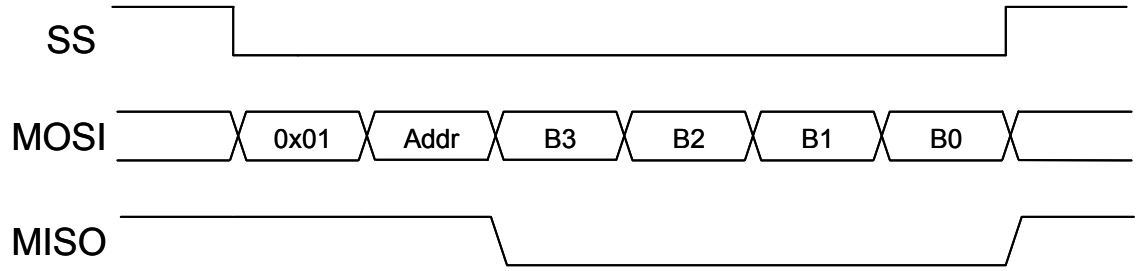


Figure 4 - Single Register Write Operation



SERIAL COMMUNICATION

The UM7 communicates using a 3.3V TTL UART at user-configurable baud rates ranging from 9600 baud to 921600 baud. The default baud rate is 115200.

Depending on how it is configured, the UM7 can transmit either binary packets for efficiency, NMEA-style ASCII packets for easy readability, or a combination of both. NMEA packets can be

transmitted by the UM7 at rates of up to 100 Hz. Binary packets can be transmitted at rates of up to 255 Hz.

The communication architecture of the UM7 allows the user to select both what data is transmitted, and at what rates. Each packet type can be configured to transmit at its own unique rate. For example, the UM7 might be configured to transmit a health packet once a second, a position packet 10 times per second, and attitude packets 250 times per second. Alternatively, the user can poll the sensor to read any subset of its data registers at any time. This flexible communication architecture allows the user to take full advantage of the limited bandwidth available on the UART.

If the UM7 is configured to transmit more information than can fit on the serial bus, a COM_OVERFLOW flag will be set in the device's health register.

To configure the UM7's transmission settings and other settings, binary packets must be sent to change the settings in a variety of configuration registers, described later in this document. Settings can be changed manually by constructing and sending the appropriate packets, or more easily by using the CHR Serial Interface software. Configuration settings can be saved to device FLASH so that they persist when power is removed.

Packet transmission rates are configured using registers CREG_COM_RATES1 through CREG_COM_RATES7 (See the Configuration Registers section for more details about configuration registers).

The serial baud rate can be configured by writing to the CREG_COM_SETTINGS register.

For more details about the register architecture on the UM7, see the Register Overview section. For specific details about how to write to and read from registers, see the Binary Packet Structure section, or refer to the UM7 User's Guide, available at www.chrobotics.com, to learn to use the CHR Serial Interface software.

NMEA PACKETS

NMEA packets provide data in human-readable, comma-separated messages over the serial port. These packets are easily interpreted by human operators and, depending on the context, by computers as well. The downside is that they are less efficient than binary packets, taking more time to transmit the same amount of information.

Each NMEA packet begins with a $ symbol and ends with a carriage-return, linefeed pair ('\r\n', simply a new line when viewed on a serial terminal). Between the start and end symbols, the packet consists of a predefined comma-separated list containing sensor data.

NMEA packets can be automatically sent by the UM7 at between 1 Hz and 100 Hz. The actual amount of data that can be transmitted depends on the baud rate, and some NMEA packets (like the sensor data packet, for example) can quickly overwhelm the serial bus, preventing the data from being transmitted as often as requested.

While NMEA messages can be transmitted by the UM7, the UM7 can only be configured using binary packets – the UM7 transmits NMEA packets, but there are no NMEA packets defined to change configuration settings.

Each NMEA sentence transmitted by the UM7 begins with the text sequence $PCHRx. The character 'P' indicates that the packet to follow is a proprietary packet (i.e. it isn't a standard NMEA packet). The sequence 'CHR' indicates that it is a CH Robotics packet. Finally, the character 'x' varies depending on the exact packet being transmitted.

Health Packet - $PCHRH

DESCRIPTION

The NMEA health packet contains a summary of health-related information, including basic GPS information and sensor status information.

PACKET FORMAT

$PCHRH,time,sats_used,sats_in_view,HDOP,mode,COM,accel,gyro,mag,GPS,res,res,res,*checksum

e.g. $PCHRH,105.015,05,11,1.5,0,0,0,0,0,0,0,0,0,*70

PACKET FIELD DESCRIPTION

Field	Description
$PCHRH	Header field, always precedes the health packet
time	e.g. 105.015 This field can be up to 13 digits long. There will always be three decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
sats_used	e.g. 05

	Represents the number of satellites used in the GPS position fix, if GPS is connected. This field is always 2 digits long.
sats_in_view	e.g. 11 Represents the number of satellites being tracked by the GPS, if GPS is connected. This field is always 2 digits long.
HDOP	e.g. 1.5 Represents the HDOP reported by the GPS module, if one is connected.
mode	e.g. 0 Indicates the UM7 mode of operation. '0' indicates Euler Angle mode. '1' indicates quaternion mode.
COM	e.g. 0 This flag is 1 if the UM7 is configured to transmit more data than it is able on the serial port. If COM overflow ever happens, this bit is set and remains set until power is cycled.
accel	e.g. 0 This flag goes from 0 to 1 if there is an accelerometer fault. This flag will remain high until accelerometer data is read properly
gyro	e.g. 0 This flag goes from 0 to 1 if there is a rate gyro fault. This flag will remain high until gyro data is read properly.
mag	e.g. 0 This flag goes from 0 to 1 if there is a magnetometer fault. This flag will remain high until mag data is read properly.
GPS	e.g. 0 This flag goes from 0 to 1 if the UM7 goes for 2 seconds or longer without receiving a packet from GPS. This flag will go low if a valid GPS packet is received.
res	e.g. 0 These 3 fields are reserved and will always output 0 on the UM7.
*checksum	e.g. *70 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

Pose packet - $PCHRP

DESCRIPTION

The NMEA pose packet contains sensor position and Euler Angle attitude. The position information in this packet is given in meters away from the sensor's configurable home lat/lon and altitude (the GPS Pose packet provides position in terms of latitude/longitude instead).

PACKET FORMAT

$PCHRP,time,pn,pe,alt,roll,pitch,yaw,heading,*checksum

e.g. $PCHRP,105.015,-501.234,-501.234,15.521,20.32,20.32,20.32,20.32,*47

PACKET FIELD DESCRIPTION

Field	Description
$PCHRP	Header field, always precedes the pose packet
time	e.g. 105.015 This field can be up to 13 digits long. There will always be 3 decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
pn	e.g. -501.234 This field can be up to 10 digits long. There will always be 3 decimal digits. If GPS is connected, this represents the sensor's position in meters north of the configurable home position. If the sensor is more than 99 km away from its home position, the fractional digits are truncated.
pe	e.g. -501.234 This field can be up to 10 digits long. There will always be 3 decimal digits. If GPS is connected, this represents the sensor's position in meters east of the configurable home position. If the sensor is more than 99 km away from its home position, the fractional digits are truncated.
alt	e.g. 15.521 This field can be up to 10 digits long. There will always be 3 decimal digits.

	If GPS is connected, this represents the sensor's altitude in meters from the configurable home position. If the sensor is more than 99 km away from its home position, the fractional digits are truncated.
roll	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the roll angle of the sensor in degrees.
pitch	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the pitch angle of the sensor in degrees.
yaw	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the yaw angle of the sensor in degrees.
heading	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the heading angle of the sensor in degrees, as reported by the GPS module if connected.
*checksum	e.g. *47 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

Attitude Packet - $PCHRA

DESCRIPTION

The NMEA attitude packet contains sensor Euler Angle attitude and GPS heading if GPS is connected.

PACKET FORMAT

$PCHRA,time,roll,pitch,yaw,heading,*checksum

e.g. $PCHRA,105.015,20.32,20.32,20.32,20.32,*66

PACKET FIELD DESCRIPTION

Field	Description

$PCHRA	Header field, always precedes the attitude packet
time	e.g. 105.015 This field can be up to 13 digits long. There will always be 3 decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
roll	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the roll angle of the sensor in degrees.
pitch	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the pitch angle of the sensor in degrees.
yaw	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the yaw angle of the sensor in degrees.
heading	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the heading angle of the sensor in degrees, as reported by the GPS module if connected.
*checksum	e.g. *66 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

Sensor Packet - $PCHRS

DESCRIPTION

The NMEA sensor packet contains gyro, accelerometer, and magnetometer data measured by the sensor. Because all the needed data will not fit into a single packet, this packet will be transmitted three times, once for each sensor.

PACKET FORMAT

$PCHRS,count,time,sensor_x,sensor_y,sensor_z,*checksum

e.g. $PCHRS,1,105.015,-0.9987,-0.9987,-0.9987,*79

PACKET FIELD DESCRIPTION

Field	Description
$PCHRS	Header field, always precedes the attitude packet
count	e.g. 1 This flag can be 0, 1, or 2. If 0, then the packet contains gyro data. If 1, then the packet contains accelerometer data. If 3, then the packet contains magnetometer data.
time	e.g. 105.015 This field can be up to 13 digits long. There will always be 3 decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
sensor_x	e.g. -0.9987 This field can be up to 11 digits long. On accel and mag data, there will always be 4 decimal digits. On gyro data, there will always be 2 decimal digits. This represents sensor data as specified in the 'count' field above. Gyro data is in degrees per second. Accelerometer data is in gravities. Magnetometer data is unit-norm (unitless).
sensor_x	e.g. -0.9987 This field can be up to 11 digits long. On accel and mag data, there will always be 4 decimal digits. On gyro data, there will always be 2 decimal digits. This represents sensor data as specified in the 'count' field above. Gyro data is in degrees per second. Accelerometer data is in gravities. Magnetometer data is unit-norm (unitless).
sensor_x	e.g. -0.9987 This field can be up to 11 digits long. On accel and mag data, there will always be 4 decimal digits. On gyro data, there will always be 2 decimal digits. This represents sensor data as specified in the 'count' field above. Gyro data is in degrees per second. Accelerometer data is in gravities. Magnetometer data is unit-norm (unitless).
*checksum	e.g. *79 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except

	the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

Rate Packet - $PCHRR

DESCRIPTION

The NMEA rate packet contains angular rates and GPS velocities measured by the sensor, if GPS is present.

PACKET FORMAT

$PCHRR,time,vn,ve,vup,roll_rate,pitch_rate,yaw_rate,*checksum

e.g. $PCHRR,105.015,15.23,15.23,15.23,-450.26,-450.26,-450.26,*68

PACKET FIELD DESCRIPTION

Field	Description
$PCHRR	Header field, always precedes the rate packet
time	e.g. 105.015 This field can be up to 13 digits long. There will always be 3 decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
vn	e.g. 15.23 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the north velocity of the sensor in m/s as reported by GPS, if it is connected.
ve	e.g. 15.23 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the east velocity of the sensor in m/s as reported by GPS, if it is connected.
vu	e.g. 15.23 This field can be up to 7 digits long. There will always be 2 decimal digits. This field is provided for compatibility and is not used on the UM7, as GPS does not provide an upward velocity component. Will always be 0.00 on the UM7.
Roll rate	e.g. -450.26

	This field can be up to 8 digits long. There will always be 2 decimal digits. This field provides the sensor's measured roll rate in degrees per second.
Pitch rate	e.g. -450.26 This field can be up to 8 digits long. There will always be 2 decimal digits. This field provides the sensor's measured pitch rate in degrees per second.
Yaw rate	e.g. -450.26 This field can be up to 8 digits long. There will always be 2 decimal digits. This field provides the sensor's measured yaw rate in degrees per second.
*checksum	e.g. *68 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

GPS Pose Packet - $PCHRG

DESCRIPTION

The NMEA GPS pose packet contains GPS latitude, longitude, and altitude in addition to Euler Angle attitude and GPS heading.

PACKET FORMAT

$PCHRG,time,latitude,longitude,altitude,roll,pitch,yaw,heading,*checksum

e.g. $PCHRG,105.015,40.047706,-111.742072,15.230,20.32,20.32,20.32,20.32,*49

PACKET FIELD DESCRIPTION

Field	Description
$PCHRG	Header field, always precedes the rate packet
time	e.g. 105.015 This field can be up to 13 digits long. There will always be 3 decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
latitude	e.g. 40.047706 This field can be up to 11 digits long. There will always be 6 decimal digits.

	This represents the latitude as reported by GPS, in degrees.
longitude	e.g. -111.742072 This field can be up to 11 digits long. There will always be 6 decimal digits. This represents the longitude as reported by GPS, in degrees.
altitude	e.g. 15.230 This field can be up to 11 digits long. There will always be 3 decimal digits. This represents the altitude as reported by GPS
roll	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the roll angle of the sensor in degrees.
pitch	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the pitch angle of the sensor in degrees.
yaw	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the yaw angle of the sensor in degrees.
heading	e.g. 20.32 This field can be up to 7 digits long. There will always be 2 decimal digits. This represents the heading angle of the sensor in degrees, as reported by the GPS module if connected.
*checksum	e.g. *49 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

Quaternion Packet - $PCHRQ

DESCRIPTION

The NMEA quaternion packet contains the attitude quaternion (valid when the sensor is in quaternion mode).

PACKET FORMAT

$PCHRG,time,a,b,c,d,*checksum

e.g. $PCHRG,105.015,0.76592,0.76592,0.76592,0.76592,*60

PACKET FIELD DESCRIPTION

Field	Description
$PCHRQ	Header field, always precedes the quaternion packet
time	e.g. 105.015 This field can be up to 13 digits long. There will always be 3 decimal digits. If no GPS is connected, this represents the amount of time in seconds since the sensor was last turned on. If GPS with PPS is properly connected, this is synchronized to UTC time of day.
a	e.g. 0.76592 This field can be up to 8 digits long. There will always be 5 decimal digits. This represents element a of the attitude quaternion.
b	e.g. 0.76592 This field can be up to 8 digits long. There will always be 5 decimal digits. This represents element b of the attitude quaternion.
c	e.g. 0.76592 This field can be up to 8 digits long. There will always be 5 decimal digits. This represents element c of the attitude quaternion.
d	e.g. 0.76592 This field can be up to 8 digits long. There will always be 5 decimal digits. This represents element d of the attitude quaternion.
*checksum	e.g. *60 This is the hex representation of the single-byte checksum of the packet. The checksum is computed using the exclusive-or of every byte in the packet except the * character preceding the checksum and the $ starting the packet. The checksum computation does include commas in the packet.

BINARY PACKET STRUCTURE

Data transmitted and received by the UM7 is formatted into packets containing:

1. The three character start sequence 's', 'n', 'p' to indicate the start of a new packet (i.e. **s**tart **n**ew **p**acket)
2. A "packet type" (PT) byte describing the function and length of the packet
3. An address byte indicating the address of the register or command
4. A sequence of data bytes, the length of which is specified in the PT byte
5. A two-byte checksum for error-detection

Table 2 - UART Serial Packet Structure

's'	'n'	'p'	packet type (PT)	Address	Data Bytes (D0...DN-1)	Checksum 1	Checksum 0

All binary packets sent and received by the UM7 must conform to the format given above.

The PT byte specifies whether the packet is a read or a write operation, whether it is a batch operation, and the length of the batch operation (when applicable). The PT byte is also used by the UM7 to respond to commands. The specific meaning of each bit in the PT byte is given below.

Table 3 - Packet Type (PT) byte

7	6	5	4	3	2	1	0
Has Data	Is Batch	BL3	BL2	BL1	BL0	Hidden	CF

Table 4 - Packet Type (PT) Bit Descriptions

Bit(s)	Description
7	Has Data: If the packet contains data, this bit is set (1). If not, this bit is cleared (0).
6	Is Batch: If the packet is a batch operation, this bit is set (1). If not, this bit is cleared (0)
5:2	Batch Length (BL): Four bits specifying the length of the batch operation. Unused if bit 7 is cleared. The maximum batch length is therefore 2^4 = 16
1	Hidden: If set, then the packet address specified in the “Address” field is a “hidden” address. Hidden registers are used to store factory calibration and filter

	tuning coefficients that do not typically need to be viewed or modified by the user. This bit should always be set to 0 to avoid altering factory configuration.
0	Command Failed (CF): Used by the autopilot to report when a command has failed. Must be set to zero for all packets written to the UM7.

The address byte specifies which register will be involved in the operation. During a read operation (Has Data = 0), the address specifies which register to read. During a write operation (Has Data = 1), the address specifies where to place the data contained in the data section of the packet. For a batch read/write operation, the address byte specifies the starting address of the operation.

The "Data Bytes" section of the packet contains data to be written to one or more registers. There is no byte in the packet that explicitly states how many bytes are in this section because it is possible to determine the number of data bytes that should be in the packet by evaluating the PT byte.

If the Has Data bit in the PT byte is cleared (Has Data = 0), then there are no data bytes in the packet and the Checksum immediately follows the address. If, on the other hand, the Has Data bit is set (Has Data = 1) then the number of bytes in the data section depends on the value of the Is Batch and Batch Length portions of the PT byte.

For a batch operation (Is Batch = 1), the length of the packet data section is equal to 4*(Batch Length). Note that the batch length refers to the number of registers in the batch, NOT the number of bytes. Registers are 4 bytes long.

For a non-batch operation (Is Batch = 0), the length of the data section is equal to 4 bytes (one register). The data section lengths and total packet lengths for different PT configurations are shown below.

Table 5 - Packet Length Summary

Has Data	Is Batch	Data Section Length (bytes)	Total Packet Length (bytes)
0	NA	0	7
1	0	4	11
1	1	4*(Batch Length)	7 + 4*(Batch Length)

Note that if a packet is a batch operation, the batch length must be greater than zero.

The two checksum bytes consist of the unsigned 16-bit sum of all preceding bytes in the packet, including the packet header.

Read Operations

To initiate a serial read of one or more registers aboard the sensor, a packet should be sent to the UM7 with the "Has Data" bit cleared. This tells the device that this will be a read operation from the address specified in the packet's "Address" byte. If the "Is Batch" bit is set, then the packet will trigger a batch read in which the "Address" byte specifies the address of the first register to be read.

In response to a read packet, the UM7 will send a packet in which the "Has Data" bit is set, and the "Is Batch" and "Batch Length" bits are equivalent to those of the packet that triggered the read operation. The register data will be contained in the "Data Bytes" section of the packet.

Write Operations

To initiate a serial write into one or more registers aboard the sensor, a packet should be sent to the UM7 with the "Has Data" bit set. This tells the device that the incoming packet contains data that should be written to the register specified by the packet's "Address" byte. If a batch write operation is to be performed, the "Is Batch" bit should be set, and the "Batch Length" bits should indicate the number of registers that are to be written to.

In response to a write packet, the UM7 will update the contents of the specified register(s) with the contents of the data section of the packet. It will then transmit a COMMAND_COMPLETE packet to indicate that the write operation succeeded. A COMMAND_COMPLETE packet is a packet with PT = 0 (no data, no batch) and with an address matching the address of the register to which the write operation was made, or the start address of the write operation if this was a batch write.

Note that the COMMAND_COMPLETE packet is equivalent to a packet that would cause the autopilot to initiate a read operation on the address to which data was just written. Since the packet is going from the sensor to the host, however, its meaning is different (it would not make sense for the autopilot to request the contents of one of its registers from an external host).

Command Operations

There are a variety of register address that do not correspond with actual physical registers aboard the UM7. These "command" addresses are used to cause the sensor to execute specific

commands (there are commands for executing calibration operations, resetting the onboard filters, etc. See the Register Overview in this document for more details).

To initiate a command, simply send a packet to the autopilot with the command's address in the packet "Address" byte. The PT byte should be set to zero for a command operation.

If the UM7 successfully completes the specified command, then a COMMAND_COMPLETE packet is returned with the command address in the "Address" byte of the response packet. If the command fails, the device responds by sending a COMMAND_FAILED packet. The COMMAND_FAILED packet is equivalent to the COMMAND_COMPLETE packet except that the "Command Failed" bit in the PT byte is set (CF = 1).

In some cases, a command will cause specific packets to be sent other than the COMMAND_COMPLETE packet. A GET_FW_VERSION command will, for example, return a packet containing the version of the firmware installed on the UM7. In this and similar cases, the COMMAND_COMPLETE packet is not sent.

Example Binary Communication Code

RECEIVING DATA FROM THE UM7

There are a lot of ways to parse the incoming data from the UM7. Often, it is easiest to write a generalized parser that takes all incoming data and extracts the data, address, and packet type information and then makes it easily accessible to the user program. The following code shows an example of a good general parser that can be used to extract packet data.

```
// Structure for holding received packet information
typedef struct UM7_packet_struct
{
    uint8_t Address;
    uint8_t PT;
    uint16_t Checksum;

    uint8_t data_length;
    uint8_t data[30];
} UM7_packet;
```

```
// parse_serial_data
// This function parses the data in 'rx_data'  with length 'rx_length' and attempts to find a packet
// in the data.  If a packet is found, the structure 'packet' is filled with the packet data.
// If there is not enough data for a full packet in the provided array, parse_serial_data returns 1.
// If there is enough data, but no packet header was found, parse_serial_data returns 2.
// If a packet header was found, but there was insufficient data to parse the whole packet,
// then parse_serial_data returns 3.  This could happen if not all of the serial data has been
// received when parse_serial_data is called.
// If a packet was received, but the checksum was bad, parse_serial_data returns 4.
// If a good packet was received, parse_serial_data fills the UM7_packet structure and returns 0.
uint8_t parse_serial_data( uint8_t* rx_data, uint8_t rx_length, UM7_packet* packet )
{
  uint8_t index;

  // Make sure that the data buffer provided is long enough to contain a full packet
  // The minimum packet length is 7 bytes
  if( rx_length < 7 )
  {
    return 1;
  }

  // Try to find the 'snp' start sequence for the packet
  for( index = 0; index < (rx_length – 2); index++ )
  {
    // Check for 'snp'.  If found, immediately exit the loop
    if( rx_data[index] == 's' && rx_data[index+1] == 'n' && rx_data[index+2] == 'p' )
    {
      break;
    }
  }

  uint8_t packet_index = index;

  // Check to see if the variable 'packet_index' is equal to (rx_length - 2).  If it is, then the above
  // loop executed to completion and never found a packet header.
```

```
if( packet_index == (rx_length – 2) )
{
    return 2;
}

// If we get here, a packet header was found.  Now check to see if we have enough room
// left in the buffer to contain a full packet.  Note that at this point, the variable 'packet_index'
// contains the location of the 's' character in the buffer (the first byte in the header)
if( (rx_length – packet_index) < 7 )
{
    return 3;
}

// We've found a packet header, and there is enough space left in the buffer for at least
// the smallest allowable packet length (7 bytes).  Pull out the packet type byte to determine
// the actual length of this packet
uint8_t PT = rx_data[packet_index + 3];

// Do some bit-level manipulation to determine if the packet contains data and if it is a batch
// We have to do this because the individual bits in the PT byte specify the contents of the
// packet.
uint8_t packet_has_data = (PT >> 7) & 0x01;     // Check bit 7 (HAS_DATA)
uint8_t packet_is_batch = (PT >> 6) & 0x01;     // Check bit 6 (IS_BATCH)
uint8_t batch_length = (PT >> 2) & 0x0F;        // Extract the batch length (bits 2 through 5)

// Now finally figure out the actual packet length
uint8_t data_length = 0;
if( packet_has_data )
{
    if( packet_is_batch )
    {
        // Packet has data and is a batch.  This means it contains 'batch_length' registers, each
        // of which has a length of 4 bytes
        data_length = 4*batch_length;
    }
    else    // Packet has data but is not a batch.  This means it contains one register (4 bytes)
```

```
    {
        data_length = 4;
    }
}
else    // Packet has no data
{
    data_length = 0;
}

// At this point, we know exactly how long the packet is.  Now we can check to make sure
// we have enough data for the full packet.
if( (rx_length – packet_index) < (data_length + 5) )
{
    return 3;
}

// If we get here, we know that we have a full packet in the buffer.  All that remains is to pull
// out the data and make sure the checksum is good.
// Start by extracting all the data
packet->Address = rx_data[packet_index + 4];
packet->PT = PT;

// Get the data bytes and compute the checksum all in one step
packet->data_length = data_length;
uint16_t computed_checksum = 's' + 'n' + 'p' + packet_data->PT + packet_data->Address;
for( index = 0; index < data_length; index++ )
{
    // Copy the data into the packet structure's data array
    packet->data[index] = rx_data[packet_index + 5 + index];
    // Add the new byte to the checksum
    computed_checksum += packet->data[index];
}

// Now see if our computed checksum matches the received checksum
// First extract the checksum from the packet
uint16_t received_checksum = (rx_data[packet_index + 5 + data_length] << 8);
```

```
    received_checksum |= rx_data[packet_index + 6 + data_length];

    // Now check to see if they don't match
    if( received_checksum != computed_checksum )
    {
        return 4;
    }

    // At this point, we've received a full packet with a good checksum.  It is already
    // fully parsed and copied to the 'packet' structure, so return 0 to indicate that a packet was
    // processed.
    return 0;
}
```

Once the packet has been parsed and copied into the packet structure, accessing the desired data is as simple as watching for packets with the desired address, checking the data length to make sure it is as long as you expect, and then pulling the data out of the packet's data array.

GETTING THE FIRMWARE REVISION

To read the firmware revision from the UM7, a GET_FW_REVISION command should be sent to the over the serial port. Recall that to initiate a command on the UM7, a read packet should be sent using the command's address in the 'Address' byte of the packet. For a GET_FW_REVISION command, the address is 170 (0xAA).

C-code for constructing and sending the command is shown below:

```
uint8_t tx_data[20];

tx_data[0] = 's';
tx_data[1] = 'n';
tx_data[2] = 'p';
tx_data[3] = 0x00;      // Packet Type byte
tx_data[4] = 0xAA;      // Address of GET_FW_REVISION register
tx_data[5] = 0x01;      // Checksum high byte
tx_data[6] = 0xFB;      // Checksum low byte
USART_transmit( tx_data, 7 );
```

The preceding code assumes that a function called USART_transmit(uint8_t* data, uint8_t length) exists that transmits 'length' characters from the provided buffer over the UART.

Once the UM7 receives the above packet, it will respond with a packet containing the firmware revision. Example code for receiving the firmware revision packet is given below. Note that this code assumes that the serial data is being received and transferred to a buffer before the example code is executed.

```
UM7_packet new_packet;
char FW_revision[5];

// Call the parse_serial_data function to handle the incoming serial data. The serial data should
// be placed in 'rx_data' and the length in 'rx_data_length' before this function is called.
if( !parse_serial_data( rx_data, rx_data_length, &new_packet )
{
   // We got a good packet! Check to see if it is the firmware revision
   if( packet.Address == 0xAA )
   {
      // Extract the firmware revision
      FW_revision[0] = packet.data[0];
      FW_revision[1] = packet.data[1];
      FW_revision[2] = packet.data[2];
      FW_revision[3] = packet.data[3];
      FW_revision[4] = '\0';    // Null-terminate the FW revision so we can use it like a string

      // Print the firmware revision to the terminal (or do whatever else you want…)
      printf("Got the firmware revision: %s\r\n", FW_revision);
   }
   // TODO: Check to see if any of the other packets that we care about have been found.
   // If so, do stuff with them.
}
```

Note that it is not always sufficient to simply check the address of the data register that you want to read. In almost all cases, data automatically transmitted by the UM7 is sent in batch operations to improve efficiency. For example, when processed rate gyro is transmitted, it is sent in one batch packet containing registers 97 (gyro x), 98 (gyro y), 99 (gyro z), and 100 (gyro time). Thus, the address of the packet is 97, but it is a batch packet with batch length 4.

REGISTER OVERVIEW

There are three types of registers onboard the UM7: configuration registers, data registers, and command registers.

Configuration registers begin at address 0x00 and are used to configure UM7's filter settings and communication behavior. Configuration register contents can be written to onboard flash to allow settings to be maintained when the device is powered down.

Data registers begin at address 0x55 (85), and store raw and processed data from the sensors along with estimated states. Unlike configuration registers, data register contents cannot be written to flash.

Command registers technically aren't registers at all, but they provide a convenient way to send commands to the UM7 when those commands do not require additional data beyond the command itself. For example, a command to run an onboard gyro bias calibration routine is triggered by querying the ZERO_GYROS command register. By using a unique register address for each command, the same communication architecture used to read from and write to data and configuration registers can be used to send commands to the autopilot. Command registers begin at address 0xAA.

Configuration Registers

Address	Register Name	Register Description
0x00 (0)	CREG_COM_SETTINGS	General communication settings
0x01 (1)	CREG_COM_RATES1	Broadcast rate settings
0x02 (2)	CREG_COM_RATES2	Broadcast rate settings
0x03 (3)	CREG_COM_RATES3	Broadcast rate settings
0x04 (4)	CREG_COM_RATES4	Broadcast rate settings
0x05 (5)	CREG_COM_RATES5	Broadcast rate settings
0x06 (6)	CREG_COM_RATES6	Broadcast rate settings
0x07 (7)	CREG_COM_RATES7	Broadcast rate settings
0x08 (8)	CREG_MISC_SETTINGS	Misc. settings
0x09 (9)	CREG_HOME_NORTH	GPS north position to consider position 0
0x0A (10)	CREG_HOME_EAST	GPS east position to consider position 0
0x0B (11)	CREG_HOME_UP	GPS altitude to consider position 0
0x0C (12)	CREG_GYRO_TRIM_X	Bias trim for x-axis rate gyro

0x0D (13)	CREG_GYRO_TRIM_Y	Bias trim for y-axis rate gyro
0x0E (14)	CREG_GYRO_TRIM_Z	Bias trim for z-axis rate gyro
0x0F (15)	CREG_MAG_CAL1_1	Row 1, Column 1 of magnetometer calibration matrix
0x10 (16)	CREG_MAG_CAL1_2	Row 1, Column 2 of magnetometer calibration matrix
0x11 (17)	CREG_MAG_CAL1_3	Row 1, Column 3 of magnetometer calibration matrix
0x12 (18)	CREG_MAG_CAL2_1	Row 2, Column 1 of magnetometer calibration matrix
0x13 (19)	CREG_MAG_CAL2_2	Row 2, Column 2 of magnetometer calibration matrix
0x14 (20)	CREG_MAG_CAL2_3	Row 2, Column 3 of magnetometer calibration matrix
0x15 (21)	CREG_MAG_CAL3_1	Row 3, Column 1 of magnetometer calibration matrix
0x16 (22)	CREG_MAG_CAL3_2	Row 3, Column 2 of magnetometer calibration matrix
0x17 (23)	CREG_MAG_CAL3_3	Row 3, Column 3 of magnetometer calibration matrix
0x18 (24)	CREG_MAG_BIAS_X	Magnetometer X-axis bias
0x19 (25)	CREG_MAG_BIAS_Y	Magnetometer Y-axis bias
0x1A (26)	CREG_MAG_BIAS_Z	Magnetometer Z-axis bias

Data Registers

Address	Register Name	Register Description
0x55 (85)	DREG_HEALTH	Contains information about the health and status of the UM7
0x56 (86)	DREG_GYRO_RAW_XY	Raw X and Y rate gyro data
0x57 (87)	DREG_GYRO_RAW_Z	Raw Z rate gyro data
0x58 (88)	DREG_GYRO_TIME	Time at which rate gyro data was acquired
0x59 (89)	DREG_ACCEL_RAW_XY	Raw X and Y accelerometer data
0x5A (90)	DREG_ACCEL_RAW_Z	Raw Z accelerometer data
0x5B (91)	DREG_ACCEL_TIME	Time at which accelerometer data was acquired
0x5C (92)	DREG_MAG_RAW_XY	Raw X and Y magnetometer data
0x5D (93)	DREG_MAG_RAW_Z	Raw Z magnetometer data
0x5E (94)	DREG_MAG_RAW_TIME	Time at which magnetometer data was acquired
0x5F (95)	DREG_TEMPERATURE	Temperature data
0x60 (96)	DREG_TEMPERATURE_TIME	Time at which temperature data was acquired
0x61 (97)	DREG_GYRO_PROC_X	Processed x-axis rate gyro data
0x62 (98)	DREG_GYRO_PROC_Y	Processed y-axis rate gyro data
0x63 (99)	DREG_GYRO_PROC_Z	Processed z-axis rate gyro data
0x64 (100)	DREG_GYRO_PROC_TIME	Time at which rate gyro data was acquired
0x65 (101)	DREG_ACCEL_PROC_X	Processed x-axis accel data
0x66 (102)	DREG_ACCEL_PROC_Y	Processed y-axis accel data
0x67 (103)	DREG_ACCEL_PROC_Z	Processed z-axis accel data
0x68 (104)	DREG_ACCEL_PROC_TIME	Time at which accelerometer data was acquired
0x69 (105)	DREG_MAG_PROC_X	Processed x-axis magnetometer data
0x6A (106)	DREG_MAG_PROC_Y	Processed y-axis magnetometer data
0x6B (107)	DREG_MAG_PROC_Z	Processed z-axis magnetometer data
0x6C (108)	DREG_MAG_PROC_TIME	Time at which magnetometer data was acquired

0x6D (109)	DREG_QUAT_AB	Quaternion elements A and B
0x6E (110)	DREG_QUAT_CD	Quaternion elements C and D
0x6F (111)	DREG_QUAT_TIME	Time at which the sensor was at the specified quaternion rotation
0x70 (112)	DREG_EULER_PHI_THETA	Roll and pitch angles
0x71 (113)	DREG_EULER_PSI	Yaw angle
0x72 (114)	DREG_EULER_PHI_THETA_DOT	Roll and pitch angle rates
0x73 (115)	DREG_EULER_PSI_DOT	Yaw rate
0x74 (116)	DREG_EULER_TIME	Time of computed Euler attitude and rates
0x75 (117)	DREG_POSITION_NORTH	North position in meters
0x76 (118)	DREG_POSITION_EAST	East position in meters
0x77 (119)	DREG_POSITION_UP	Altitude in meters
0x78 (120)	DREG_POSITION_TIME	Time of estimated position
0x79 (121)	DREG_VELOCITY_NORTH	North velocity
0x7A (122)	DREG_VELOCITY_EAST	East velocity
0x7B (123)	DREG_VELOCITY_UP	Altitude velocity
0x7C (124)	DREG_VELOCITY_TIME	Time of velocity estimate
0x7D (125)	DREG_GPS_LATITUDE	GPS latitude
0x7E (126)	DREG_GPS_LONGITUDE	GPS longitude
0x7F (127)	DREG_GPS_ALTITUDE	GPS altitude
0x80 (128)	DREG_GPS_COURSE	GPS course
0x81 (129)	DREG_GPS_SPEED	GPS speed
0x82 (130)	DREG_GPS_TIME	GPS time (UTC time of day in seconds)
0x83 (131)	DREG_GPS_SAT_1_2	GPS satellite information
0x84 (132)	DREG_GPS_SAT_3_4	GPS satellite information
0x85 (133)	DREG_GPS_SAT_5_6	GPS satellite information
0x86 (134)	DREG_GPS_SAT_7_8	GPS satellite information
0x87 (135)	DREG_GPS_SAT_9_10	GPS satellite information
0x88 (136)	DREG_GPS_SAT_11_12	GPS satellite information

Commands

Address	Name	Description
0xAA (170)	GET_FW_REVISION	Causes the autopilot to respond with a packet containing the current firmware revision.
0xAB (171)	FLASH_COMMIT	Writes all current configuration settings to flash
0xAC (172)	RESET_TO_FACTORY	Reset all settings to factory defaults
0xAD (173)	ZERO_GYROS	Causes the rate gyro biases to be calibrated.
0xAE (174)	SET_HOME_POSITION	Sets the current GPS location as position (0,0)
0xAF (175)	RESERVED	RESERVED
0xB0 (176)	SET_MAG_REFERENCE	Sets the magnetometer reference vector
0xB1 (177)	RESERVED	RESERVED
0xB2 (178)	RESERVED	RESERVED
0xB3 (179)	RESET_EKF	Resets the EKF

CONFIGURATION REGISTERS

A set of 32-bit configuration registers allows the UM7's behavior to be customized for specific applications. In general, settings are most easily configured using the CHR Serial Interface, which allows the contents of each configuration register to be set without understanding the register contents at the bit/byte level.

This section outlines in detail the contents and functionality of each register.

CREG_COM_SETTINGS – 0x00 (0)

SUMMARY

The CREG_COM_SETTINGS register is used to set the autopilot's serial port baud rate and to enable or disable the automatic transmission of sensor data and estimated states (telemetry).

REGISTER CONTENTS

B3								B2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BAUD_RATE				GPS_BAUD				Reserved							

B1								B0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							GPS	Reserved			SAT	Reserved			

DESCRIPTION

Bits	Name	Description
31:28	BAUD_RATE	Sets the baud rate of the UM7 main serial port. *0 = 9600* *1 = 14400* *2 = 19200* *3 = 38400*

		4 = 57600 *5 = 115200* *6 = 128000** *7 = 153600** *8 = 230400** *9 = 256000** *10 = 460800** *11 = 921600** *12:15 = reserved* ** Most PC serial ports do not support baud-rates above 115200*
27:24	GPS_BAUD	Sets the baud rate of the UM7 auxiliary serial port. *0 = 9600* *1 = 14400* *2 = 19200* *3 = 38400* *4 = 57600* *5 = 115200*
23:9	Reserved	These bits are reserved for future use
8	GPS	If set, this bit causes GPS data to be transmitted automatically whenever new GPS data is received. GPS data is stored in registers 125 to 130. These registers will be transmitted in a batch packet of length 6 starting at address 125.
7:5	Reserved	These bits are reserved for future use
4	SAT	If set, this bit causes satellite details to be transmitted whenever they are provided by the GPS. Satellite information is stored in registers 131 to 136. These registers will be transmitted in a batch packet of length 6 beginning at address 131.
3:0	Reserved	These bits are reserved for future use

CREG_COM_RATES1 – 0x01 (1)

SUMMARY

The CREG_COM_RATES1 register sets desired telemetry transmission rates in Hz for raw accelerometer, gyro, and magnetometer data. If the specified rate is 0, then no data is transmitted.

REGISTER CONTENTS

B3	B2	B1	B0
RAW_ACCEL_RATE	RAW_GYRO_RATE	RAW_MAG_RATE	Reserved

DESCRIPTION

Bits	Name	Description
31:24	RAW_ACCEL_RATE	Specifies the desired raw accelerometer data broadcast rate in Hz. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
23:16	RAW_GYRO_RATE	Specifies the desired raw gyro data broadcast rate in Hz. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
15:8	RAW_MAG_RATE	Specifies the desired raw magnetometer data broadcast rate in Hz. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
7:0	Reserved	These bits are reserved.

Raw accelerometer data is stored in registers 89 to 91. When the raw accel rate is greater than 0, the accelerometer data is transmitted in a batch packet of length 3 with start address 89.

Raw rate gyro data is stored in registers 86 to 88. When the raw gyro rate is greater than 0, the rate gyro data is transmitted in a batch packet of length 3 with start address 86.

Raw magnetometer data is stored in registers 92 to 94. When the raw magnetometer rate is greater than 0, the magnetometer data is transmitted in a batch packet of length 3 with start address 92.

If the "all raw data rate" in CREG_COM_RATES2 is greater than 0, then all gyro, accelerometer, magnetometer, and pressure data will be transmitted together. The rates in CREG_COM_RATES1 are then not used.

CREG_COM_RATES2 – 0x02 (2)

SUMMARY

The CREG_COM_RATES2 register sets desired telemetry transmission rates for all raw data and temperature. If the specified rate is 0, then no data is transmitted.

REGISTER CONTENTS

B3	B2	B1	B0
TEMP_RATE	RES	RES	ALL_RAW_RATE

DESCRIPTION

Bits	Name	Description
31:24	TEMP_RATE	Specifies the desired broadcast rate for temperature data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
23:16	RES	These bits are reserved for future use
15:8	RES	These bits are reserved for future use
7:0	ALL_RAW_RATE	Specifies the desired broadcast rate for all raw sensor data. If set, this overrides the broadcast rate setting for individual raw data broadcast rates. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.

Temperature data is stored in registers 95 and 96. If the temperature broadcast rate is greater than 0, then temperature data will be sent in a batch packet of length 2 with start address 95. If all raw data is being transmitted (as specified by byte 3 of this register), then the temperature data will be transmitted as part of the raw batch packet at “all raw rate” instead of the raw temperature rate.

Raw sensor/temperature data occupies registers 86 through 96. If the raw data broadcast rate is greater than 0, then all raw data and temperature data is sent in one batch packet of length 11, with start address 86.

CREG_COM_RATES3 – 0x03 (3)

SUMMARY

The CREG_COM_RATES3 register sets desired telemetry transmission rates for processed sensor data. If the specified rate is 0, then no data is transmitted.

REGISTER CONTENTS

B3	B2	B1	B0
PROC_ACCEL_RATE	PROC_GYRO_RATE	PROC_MAG_RATE	Reserved

DESCRIPTION

Bits	Name	Description
31:24	PROC_ACCEL_RATE	Specifies the desired broadcast rate for processed accelerometer data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
23:16	PROC_GYRO_RATE	Specifies the desired broadcast rate for processed rate gyro data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
15:8	PROC_MAG_RATE	Specifies the desired broadcast rate for processed magnetometer data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
7:0	Reserved	These bits are reserved.

Processed accelerometer data is stored in registers 101 to 104. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet of length 4 and start address 104.

Processed rate gyro data is stored in registers 97 to 100. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet of length 4 and start address 100.

Processed magnetometer data is stored in registers 105 to 108. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet of length 4 and start address 108.

If the “all processed data broadcast rate” setting in register CREG_COM_RATES4 is not zero, then the rates specified in the CREG_COM_RATES3 register are overridden.

CREG_COM_RATES4 – 0x04 (4)

SUMMARY

The CREG_COM_RATES4 register sets desired telemetry transmission rates for all processed dat. If the specified rate is 0, then no data is transmitted.

REGISTER CONTENTS

B3	B2	B1	B0
RES	RES	RES	ALL_PROC_RATE

DESCRIPTION

Bits	Name	Description
31:24	RES	These bits are reserved for future use
23:16	RES	These bits are reserved for future use
15:8	RES	These bits are reserved for future use
7:0	ALL_PROC_RATE	Specifies the desired broadcast rate for raw all processed data. If set, this overrides the broadcast rate setting for individual processed data broadcast rates. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.

All processed data comprises registers 97 through 108 (a total of 12 registers). If ALL_PROC_RATE is non-zero, the processed data will be transmitted as a single packet of batch length 12, starting at address 97.

CREG_COM_RATES5 – 0x05 (5)

SUMMARY

The CREG_COM_RATES5 register sets desired telemetry transmission rates for quaternions, Euler Angles, position, and velocity estimates. If the specified rate is 0, then no data is transmitted.

REGISTER CONTENTS

B3	B2	B1	B0
QUAT_RATE	EULER_RATE	POSITION_RATE	VELOCITY_RATE

DESCRIPTION

Bits	Name	Description
31:24	QUAT_RATE	Specifies the desired broadcast rate for quaternion data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
23:16	EULER_RATE	Specifies the desired broadcast rate for Euler Angle data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
15:8	POSITION_RATE	Specifies the desired broadcast rate position. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
7:0	VELOCITY_RATE	Specifies the desired broadcast rate for velocity. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.

Quaternion data is stored in registers 109 to 111. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet with length 3 and start address 109.

Euler Angle data is stored in registers 112 to 116. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet of length 5 and start address 112.

Position data is stored in registers 117 to 120. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet of length 4 and start address 117.

Velocity data is stored in registers 121 to 124. If the specified broadcast rate is greater than 0, then the data will be transmitted in a batch packet of length 4 and start address 121.

If the “pose broadcast rate” setting in register CREG_COM_RATES6 is not zero, then the rates specified by EULER_RATE and POSITION_RATE are overridden.

CREG_COM_RATES6 – 0x06 (6)

SUMMARY

The CREG_COM_RATES6 register sets desired telemetry transmission rates for pose (Euler/position packet) and health. If the specified rate is 0, then no data is transmitted.

REGISTER CONTENTS

B3								B2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSE_RATE								RESERVED				HEALTH_RATE			

B1								B0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

DESCRIPTION

Bits	Name	Description
31:24	POSE_RATE	Specifies the desired broadcast rate for pose (Euler Angle and position) data. The data is stored as an unsigned 8-bit integer, yielding a maximum rate of 255 Hz.
23:20	RESERVED	These bits are reserved for future use.
19:16	HEALTH_RATE	Specifies the desired broadcast rate for the sensor health packet. 0 = off 1 = 0.125 Hz 2 = 0.25 Hz 3 = 0.5 Hz 4 = 1 Hz 5 = 2 Hz 6 = 4 Hz 7:15 = Unused* * Will default to 1Hz
15:0	RESERVED	These bits are reserved for future use.

Pose data (Euler Angles and position) is stored in registers 112 to 120. If the pose rate is greater than 0, then pose data will be transmitted in a batch packet with length 9 and start address 112.

Health data is stored in register address 85. If the health rate is not 0, then health data will be transmitted as a non-batch packet with address 85.

CREG_COM_RATES7 – 0x07 (7)

SUMMARY

The CREG_COM_RATES7 register sets desired transmission rates for CHR NMEA-style packets.

REGISTER CONTENTS

B3								B2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HEALTH_RATE				POSE_RATE				ATTITUDE_RATE				SENSOR_RATE			

B1								B0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RATES_RATE				GPS_POSE_RATE				QUAT_RATE				RESERVED			

DESCRIPTION

Bits	Name	Description
31:28	HEALTH_RATE	Specifies the desired broadcast rate for CHR NMEA-style health packet. 0 = off 1 = 1 Hz 2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz

		8 = 30 Hz 9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz 15 = 100 Hz
27:24	POSE_RATE	Specifies the desired broadcast rate for CHR NMEA-style pose (Euler Angle/position) packet. 0 = off 1 = 1 Hz 2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz 8 = 30 Hz 9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz 15 = 100 Hz
23:20	ATTITUDE_RATE	Specifies the desired broadcast rate for CHR NMEA-style attitude packet. 0 = off 1 = 1 Hz

		2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz 8 = 30 Hz 9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz 15 = 100 Hz
19:16	SENSOR_RATE	Specifies the desired broadcast rate for CHR NMEA-style sensor data packet 0 = off 1 = 1 Hz 2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz 8 = 30 Hz 9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz

		15 = 100 Hz
15:12	RATES_RATE	Specifies the desired broadcast rate for CHR NMEA-style rate data packet. 0 = off 1 = 1 Hz 2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz 8 = 30 Hz 9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz 15 = 100 Hz
11:8	GPS_POSE_RATE	Specifies the desired broadcast rate for CHR NMEA-style GPS pose packet. 0 = off 1 = 1 Hz 2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz 8 = 30 Hz

		9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz 15 = 100 Hz
7:4	QUAT_RATE	Specifies the desired broadcast rate for CHR NMEA-style quaternion packet. 0 = off 1 = 1 Hz 2 = 2 Hz 3 = 4 Hz 4 = 5 Hz 5 = 10 Hz 6 = 15 Hz 7 = 20 Hz 8 = 30 Hz 9 = 40 Hz 10 = 50 Hz 11 = 60 Hz 12 = 70 Hz 13 = 80 Hz 14 = 90 Hz 15 = 100 Hz
3:0	RES	These bits are reserved.

CREG_MISC_SETTINGS – 0x08 (8)

SUMMARY

This register contains miscellaneous filter and sensor control options.

REGISTER CONTENTS

B3								B2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															

B1								B0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							PPS	RESERVED					ZG	Q	MAG

DESCRIPTION

Bits	Name	Description
31:9	RESERVED	These bits are reserved for future use
8	PPS	If set, this bit causes the TX2 pin on the IO Expansion header to be used as the PPS input from an external GPS module. PPS pulses will then be used to synchronize the system clock to UTC time of day.
7:3	RESERVED	These bits are reserved for future use
2	ZG	If set, this bit causes the UM7 to attempt to measure the rate gyro bias on startup. The sensor must be stationary on startup for this feature to work properly.
1	Q	If this bit is set, the sensor will run in quaternion mode instead of Euler Angle mode.
0	MAG	If set, the magnetometer will be used in state updates.

CREG_HOME_NORTH – 0x09 (9)

SUMMARY

This register sets the north home latitude in degrees, used to convert GPS coordinates to position in meters from home.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_HOME_EAST – 0x0A (10)

SUMMARY

This register sets the east home longitude in degrees, used to convert GPS coordinates to position in meters from home.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_HOME_UP – 0x0B (11)

SUMMARY

This register sets the home altitude in meters. Used to convert GPS coordinates to position in meters from home.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_GYRO_TRIM_X – 0x0C (12)

SUMMARY

This register sets the x-axis rate gyro trim, which is used to add additional bias compensation for the rate gyros during calls to the ZERO_GYRO_BIAS command.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_GYRO_TRIM_Y – 0x0D (13)

SUMMARY

This register sets the y-axis rate gyro trim, which is used to add additional bias compensation for the rate gyros during calls to the ZERO_GYRO_BIAS command.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_GYRO_TRIM_Z – 0x0E (14)

SUMMARY

This register sets the z-axis rate gyro trim, which is used to add additional bias compensation for the rate gyros during calls to the ZERO_GYRO_BIAS command.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_MAG_CAL1_1 to CREG_MAG_CAL3_3 – 0x0F (15) to 0x17 (23)

SUMMARY

These registers store the 9 entries into a 3x3 matrix that is used to perform soft-iron calibration of the magnetometer on the device. These terms can be computed by performing magnetometer calibration with the CHR Serial Interface.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_MAG_BIAS_X – 0x18 (24)

SUMMARY

This registers stores a bias term for the magnetometer x-axis for hard-iron calibration. This term can be computed by performing magnetometer calibration with the CHR Serial Interface.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_MAG_BIAS_Y – 0x19 (25)

SUMMARY

This registers stores a bias term for the magnetometer y-axis for hard-iron calibration. This term can be computed by performing magnetometer calibration with the CHR Serial Interface.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

CREG_MAG_BIAS_Z – 0x1A (26)

SUMMARY

This registers stores a bias term for the magnetometer z-axis for hard-iron calibration. This term can be computed by performing magnetometer calibration with the CHR Serial Interface.

REGISTER CONTENTS

B3	B2	B1	B0
32-bit IEEE Floating Point Value			

DATA REGISTERS

DREG_HEALTH – 0x55 (85)

SUMMARY

The health register reports the current status of the GPS module and the other sensors on the UM7. Monitoring the health register is the easiest way to monitor the quality of the GPS lock and to watch for other problems that could affect the behavior of the UM7.

REGISTER CONTENTS

B3								B2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SATS_USED						HDOP									

B1								B0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SATS_IN_VIEW						RES	OVF	RES		MG_N	ACC_N	ACCEL	GYRO	MAG	GPS

DESCRIPTION

Bits	Name	Description
31:26	SATS_USED	Reports the number of satellites used in the position solution.
25:16	HDOP	Reports the horizontal dilution of precision (HDOP) reported by the GPS. The actual HDOP value is equal to the contents of the HDOP bits divided by 10.
15:10	SATS_IN_VIEW	Reports the number of satellites in view.
9	RES	This bit is reserved.
8	OVF	Overflow bit. This bit is set if the UM7 is attempting to transmit data over the serial port faster than is allowed given the baud-rate. If this bit is set, reduce broadcast rates in the COM_RATES registers.
7:6	RES	These bits are reserved.
5	MG_N	This bit is set if the sensor detects that the norm of the magnetometer measurement is too far away from 1.0 to be

		trusted. Usually indicates bad calibration, local field distortions, or both.
4	ACC_N	This bit is set if the sensor detects that the norm of the accelerometer measurement is too far away from 1G to be used (i.e. during aggressive acceleration or high vibration).
3	ACCEL	This bit will be set if the accelerometer fails to initialize on startup.
2	GYRO	This bit will be set if the rate gyro fails to initialize on startup.
1	MAG	This bit will be set if the magnetometer fails to initialize on startup.
0	GPS	This bit is set if the GPS fails to send a packet for more than two seconds. If a GPS packet is ever received, this bit is cleared.

DREG_GYRO_RAW_XY – 0x56 (86)

SUMMARY

Contains raw X and Y axis rate gyro data

REGISTER CONTENTS

B3	B2	B1	B0
Gyro X (2's complement 16-bit integer)		Gyro Y (2's complement 16-bit integer)	

DREG_GYRO_RAW_Z – 0x57 (87)

SUMMARY

Contains raw Z axis rate gyro data

REGISTER CONTENTS

B3	B2	B1	B0
Gyro Z (2's complement 16-bit integer)		RES	

DREG_GYRO_RAW_TIME – 0x58 (88)

SUMMARY

Contains time at which the last rate gyro data was acquired.

REGISTER CONTENTS

B3	B2	B1	B0
Gyro Time (IEEE Floating Point)			

DREG_ACCEL_RAW_XY – 0x59 (89)

SUMMARY

Contains raw X and Y axis accelerometer data.

REGISTER CONTENTS

B3	B2	B1	B0
Accel X (2's complement 16-bit integer)		Accel Y (2's complement 16-bit integer)	

DREG_ACCEL_RAW_Z – 0x5A (90)

SUMMARY

Contains raw Z axis accelerometer data

REGISTER CONTENTS

B3	B2	B1	B0
Accel Z (2's complement 16-bit integer)		RES	

DREG_ACCEL_RAW_TIME – 0x5B (91)

SUMMARY

Contains time at which the last accelerometer data was acquired.

REGISTER CONTENTS

B3	B2	B1	B0
Accel Time (IEEE Floating Point)			

DREG_MAG_RAW_XY – 0x5C (92)

SUMMARY

Contains raw X and Y axis magnetometer data.

REGISTER CONTENTS

B3	B2	B1	B0
Mag X (2's complement 16-bit integer)		Mag Y (2's complement 16-bit integer)	

DREG_MAG_RAW_Z – 0x5D (93)

SUMMARY

Contains raw Z axis magnetometer data

REGISTER CONTENTS

B3	B2	B1	B0
Mag Z (2's complement 16-bit integer)		RES	

DREG_MAG_RAW_TIME – 0x5E (94)

SUMMARY

Contains time at which the last magnetometer data was acquired.

REGISTER CONTENTS

B3	B2	B1	B0
Mag Time (IEEE Floating Point)			

DREG_TEMPERATURE – 0x5F (95)

SUMMARY

Contains the temperature output of the onboard temperature sensor.

REGISTER CONTENTS

B3	B2	B1	B0
Temperature in degrees C (IEEE Floating Point)			

DREG_TEMPERATURE_TIME – 0x60 (96)

SUMMARY

Contains time at which the last temperature was acquired.

REGISTER CONTENTS

B3	B2	B1	B0
Temperature time (IEEE Floating Point)			

DREG_GYRO_PROC_X – 0x61 (97)

SUMMARY

Contains the actual measured angular rate in degrees/s after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Gyro X (IEEE Floating Point)			

DREG_GYRO_PROC_Y – 0x62 (98)

SUMMARY

Contains the actual measured angular rate in degrees/s after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Gyro Y (IEEE Floating Point)			

DREG_GYRO_PROC_Z – 0x63 (99)

SUMMARY

Contains the actual measured angular rate in degrees/s after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Gyro Z (IEEE Floating Point)			

DREG_GYRO_PROC_TIME – 0x64 (100)

SUMMARY

Contains the time at which the last rate gyro data was measured.

REGISTER CONTENTS

B3	B2	B1	B0
Gyro Time (IEEE Floating Point)			

DREG_ACCEL_PROC_X – 0x65 (101)

SUMMARY

Contains the actual measured acceleration in m/s/s after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Accel X (IEEE Floating Point)			

DREG_ACCEL_PROC_Y – 0x66 (102)

SUMMARY

Contains the actual measured acceleration in m/s/s after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Accel Y (IEEE Floating Point)			

DREG_ACCEL_PROC_Z – 0x67 (103)

SUMMARY

Contains the actual measured acceleration in m/s/s after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Accel Z (IEEE Floating Point)			

DREG_ACCEL_PROC_TIME – 0x68 (104)

SUMMARY

Contains the time at which the acceleration was measured.

REGISTER CONTENTS

B3	B2	B1	B0
Accel Time (IEEE Floating Point)			

DREG_MAG_PROC_X – 0x69 (105)

SUMMARY

Contains the actual measured magnetic field after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Mag X (IEEE Floating Point)			

DREG_MAG_PROC_Y – 0x6A (106)

SUMMARY

Contains the actual measured magnetic field after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Mag Y (IEEE Floating Point)			

DREG_MAG_PROC_Z – 0x6B (107)

SUMMARY

Contains the actual measured magnetic field after calibration has been applied.

REGISTER CONTENTS

B3	B2	B1	B0
Mag Z (IEEE Floating Point)			

DREG_MAG_PROC_TIME – 0x6C (108)

SUMMARY

Contains the time at which magnetometer data was acquired.

REGISTER CONTENTS

B3	B2	B1	B0
Mag Time (IEEE Floating Point)			

DREG_QUAT_AB – 0x6D (109)

SUMMARY

Contains the first two components of the estimated quaternion attitude.

REGISTER CONTENTS

B3	B2	B1	B0
Quat A		Quat B	

DESCRIPTION

Bits	Name	Description
31:16	Quat A	First quaternion component. Stored as a 16-bit signed integer. To get the actual value, divide by 29789.09091.
15:0	Quat B	Second quaternion component. Stored as a 16-bit signed integer. To get the actual value, divide by 29789.09091.

DREG_QUAT_CD – 0x6E (110)

SUMMARY

Contains the second two components of the estimated quaternion attitude.

REGISTER CONTENTS

B3	B2	B1	B0
Quat C		Quat D	

DESCRIPTION

Bits	Name	Description
31:16	Quat C	Third quaternion component. Stored as a 16-bit signed integer. To get the actual value, divide by 29789.09091.
15:0	Quat D	Fourth quaternion component. Stored as a 16-bit signed integer. To get the actual value, divide by 29789.09091.

DREG_QUAT_TIME – 0x6F (111)

SUMMARY

Contains the time that the quaternion attitude was measured

REGISTER CONTENTS

B3	B2	B1	B0
Quaternion Time (IEEE Floating Point)			

DREG_EULER_PHI_THETA – 0x70 (112)

SUMMARY

Contains the pitch and roll angle estimates.

REGISTER CONTENTS

B3	B2	B1	B0
Phi (roll)		Theta (Pitch)	

DESCRIPTION

Bits	Name	Description
31:16	Phi (roll)	Roll angle. Stored as a 16-bit signed integer. To get the actual value, divide by 91.02222.
15:0	Theta (pitch)	Pitch angle. Stored as a 16-bit signed integer. To get the actual value, divide by 91.02222.

DREG_EULER_PSI – 0x71 (113)

SUMMARY

Contains the yaw angle estimate.

REGISTER CONTENTS

B3	B2	B1	B0
Psi (yaw)		Unused	

DESCRIPTION

Bits	Name	Description
31:16	Psi (yaw)	Yaw angle. Stored as a 16-bit signed integer. To get the actual value, divide by 91.02222.
15:0	Unused	These bits are unused

DREG_EULER_PHI_THETA_DOT – 0x72 (114)

SUMMARY

Contains the pitch and roll rate estimates.

REGISTER CONTENTS

B3	B2	B1	B0
Roll rate		Pitch rate	

DESCRIPTION

Bits	Name	Description
31:16	Roll rate	Roll rate. Stored as a 16-bit signed integer. To get the actual value in degrees per second, divide by 16.0.
15:0	Pitch rate	Pitch angle. Stored as a 16-bit signed integer. To get the actual value in degrees per second, divide by 16.0.

DREG_EULER_PSI_DOT – 0x73 (115)

SUMMARY

Contains the yaw rate estimate.

REGISTER CONTENTS

B3	B2	B1	B0
Yaw rate		Unused	

DESCRIPTION

Bits	Name	Description
31:16	Yaw rate	Yaw rate. Stored as a 16-bit signed integer. To get the actual value in degrees per second, divide by 16.0.

15:0	Unused	These bits are unused

DREG_EULER_TIME – 0x74 (116)

SUMMARY

Contains the time that the Euler Angles were measured.

REGISTER CONTENTS

B3	B2	B1	B0
Euler Time (IEEE Floating Point)			

DREG_POSITION_N – 0x75 (117)

SUMMARY

Contains the measured north position in meters from the latitude specified in CREG_HOME_NORTH.

REGISTER CONTENTS

B3	B2	B1	B0
North Position (IEEE Floating Point)			

DREG_POSITION_E – 0x76 (118)

SUMMARY

Contains the measured east position in meters from the longitude specified in CREG_HOME_EAST.

REGISTER CONTENTS

B3	B2	B1	B0
East Position (IEEE Floating Point)			

DREG_POSITION_UP – 0x77 (119)

SUMMARY

Contains the measured altitude in meters from the altitude specified in CREG_HOME_UP.

REGISTER CONTENTS

B3	B2	B1	B0
Altitude (IEEE Floating Point)			

DREG_POSITION_TIME – 0x78 (120)

SUMMARY

Contains the time at which the position was acquired.

REGISTER CONTENTS

B3	B2	B1	B0
Position Time (IEEE Floating Point)			

DREG_VELOCITY_N – 0x79 (121)

SUMMARY

Contains the measured north velocity in m/s.

REGISTER CONTENTS

B3	B2	B1	B0
North Velocity (IEEE Floating Point)			

DREG_VELOCITY_E – 0x7A (122)

SUMMARY

Contains the measured east velocity in m/s.

REGISTER CONTENTS

B3	B2	B1	B0
East Velocity (IEEE Floating Point)			

DREG_VELOCITY_UP – 0x7B (123)

SUMMARY

Contains the measured altitude velocity in m/s.

REGISTER CONTENTS

B3	B2	B1	B0
Altitude Velocity (IEEE Floating Point)			

DREG_VELOCITY_TIME – 0x7C (124)

SUMMARY

Contains the time at which the velocity was measured.

REGISTER CONTENTS

B3	B2	B1	B0
Velocity Time (IEEE Floating Point)			

DREG_GPS_LATITUDE – 0x7D (125)

SUMMARY

Contains the GPS-reported latitude in degrees.

REGISTER CONTENTS

B3	B2	B1	B0
GPS Latitude (IEEE Floating Point)			

DREG_GPS_LONGITUDE – 0x7E (126)

SUMMARY

Contains the GPS-reported longitude in degrees.

REGISTER CONTENTS

B3	B2	B1	B0
GPS Longitude (IEEE Floating Point)			

DREG_GPS_ALTITUDE – 0x7F (127)

SUMMARY

Contains the GPS-reported altitude in meters.

REGISTER CONTENTS

B3	B2	B1	B0
GPS Altitude (IEEE Floating Point)			

DREG_GPS_COURSE – 0x80 (128)

SUMMARY

Contains the GPS-reported course in degrees.

REGISTER CONTENTS

B3	B2	B1	B0
GPS Course (IEEE Floating Point)			

DREG_GPS_SPEED – 0x81 (129)

SUMMARY

Contains the GPS-reported speed in m/s.

REGISTER CONTENTS

B3	B2	B1	B0
GPS Speed (IEEE Floating Point)			

DREG_GPS_TIME – 0x82 (130)

SUMMARY

Contains the GPS-reported time in seconds from the last epoch.

REGISTER CONTENTS

B3	B2	B1	B0
GPS Time (IEEE Floating Point)			

DREG_GPS_SAT_1_2 – 0x83 (131)

SUMMARY

Contains satellite ID and SNR for satellites 1 and 2.

REGISTER CONTENTS

B3	B2	B1	B0
Sat1 ID	Sat1 SNR	Sat2 ID	Sat2 SNR

DESCRIPTION

Bits	Name	Description
31:24	Sat1 ID	ID of satellite
23:16	Sat1 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver
15:8	Sat2 ID	ID of satellite
7:0	Sat2 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver

DREG_GPS_SAT_3_4 – 0x84 (132)

SUMMARY

Contains satellite ID and SNR for satellites 3 and 4.

REGISTER CONTENTS

B3	B2	B1	B0
Sat3 ID	Sat3 SNR	Sat4 ID	Sat4 SNR

DESCRIPTION

Bits	Name	Description

31:24	Sat3 ID	ID of satellite
23:16	Sat3 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver
15:8	Sat4 ID	ID of satellite
7:0	Sat5 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver

DREG_GPS_SAT_5_6 – 0x85 (133)

SUMMARY

Contains satellite ID and SNR for satellites 5 and 6.

REGISTER CONTENTS

B3	B2	B1	B0
Sat5 ID	Sat5 SNR	Sat6 ID	Sat6 SNR

DESCRIPTION

Bits	Name	Description
31:24	Sat5 ID	ID of satellite
23:16	Sat5 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver
15:8	Sat6 ID	ID of satellite
7:0	Sat6 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver

DREG_GPS_SAT_7_8 – 0x86 (134)

SUMMARY

Contains satellite ID and SNR for satellites 7 and 8.

REGISTER CONTENTS

B3	B2	B1	B0
Sat7 ID	Sat7 SNR	Sat8 ID	Sat8 SNR

DESCRIPTION

Bits	Name	Description

31:24	Sat7 ID	ID of satellite
23:16	Sat7 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver
15:8	Sat8 ID	ID of satellite
7:0	Sat8 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver

DREG_GPS_SAT_9_10 – 0x87 (135)

SUMMARY

Contains satellite ID and SNR for satellites 9 and 10.

REGISTER CONTENTS

B3	B2	B1	B0
Sat9 ID	Sat9 SNR	Sat10 ID	Sat10 SNR

DESCRIPTION

Bits	Name	Description
31:24	Sat9 ID	ID of satellite
23:16	Sat9 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver
15:8	Sat10 ID	ID of satellite
7:0	Sat10 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver

DREG_GPS_SAT_11_12 – 0x88 (136)

SUMMARY

Contains satellite ID and SNR for satellites 11 and 12.

REGISTER CONTENTS

B3	B2	B1	B0
Sat11 ID	Sat11 SNR	Sat12 ID	Sat12 SNR

DESCRIPTION

Bits	Name	Description

31:24	Sat11 ID	ID of satellite
23:16	Sat11 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver
15:8	Sat12 ID	ID of satellite
7:0	Sat12 SNR	Signal-to-Noise Ratio of satellite as reported by GPS receiver

COMMANDS

GET_FW_REVISION – 0xAA (170)

Causes the UM7 to transmit a packet containing the firmware revision string. The firmware revision is a four-byte character sequence. The first firmware release version for the UM7, for example, was "OR1A".

The address of the packet will be 0xAA. The data section of the packet will contain four bytes.

FLASH_COMMIT – 0xAB (171)

Causes the UM7 to write all configuration settings to FLASH so that they will remain when the power is cycled.

RESET_TO_FACTORY – 0xAC (172)

Causes the UM7 to load default factory settings.

ZERO_GYROS – 0xAD (173)

Causes the UM7 to measure the gyro outputs and set the output trim registers to compensate for any non-zero bias. The UM7 should be kept stationary while the zero operation is underway.

SET_HOME_POSITION – 0xAE (174)

Sets the current GPS latitude, longitude, and altitude as the home position. All future positions will be referenced to the current GPS position.

SET_MAG_REFERENCE – 0xB0 (176)

Sets the current GPS latitude, longitude, and altitude as the home position. All future positions will be referenced to the current GPS position.

RESET_EKF – 0xB3 (179)

Resets the filter.

DISCLAIMER

This document is provided as reference only. Typical device specifications must be evaluated by the end-user. CH Robotics reserves the right to modify this document and the products it describes without notice.

CH Robotics products are not intended for use in weapons systems, aircraft, life-saving or life-sustaining systems, automobiles, or any other application where failure could result in injury, death, property damage, or environmental damage.