Clearpath Communication Protocol

For use with the Clearpath Robotics research platforms

Version: 1.1
Date: 2 September 2010

Revision History

Version	*Date*	*Description*
1.0	26 March 2010	Release
1.1	2 September	Title change, name changes to differential commands, adding persistence, some factory-only settings, raw data, magnetometer information

Contents

Overview

This protocol is meant as a simple way for users of the various Clearpath Robotics research offerings to interface with the Clearpath Robotics hardware. It includes several features intended to increase communication reliability, while keeping message overhead and protocol complexity low. For the sake of rapid prototyping, it is not intended for multiple devices to be simultaneously connected to each communication line, removing the need for addressing or negotiation.

Additionally, no setup commands are necessary. Each platform's default firmware is set up to allow for movement out-of-the-box – no characterization or controls development necessary. However, low-level servo access is still available for those who wish to develop their own controllers.

Please consult the specific documentation for your platform for any safety considerations, terrain limits, or any other operating requirements.

Terms

Platform: The Clearpath Robotics hardware being used, including the microcontroller (one end of the serial line).

Mobile Platform: A platform which is not fixed to the ground and is capable of controlled movement.

Control computer: The computer issuing commands to the platform (the other end of the serial line).

Little-endian byte order: The first byte received of a multi-byte field is the least significant byte.

DOF: Degree of freedom, referring to an independent joint in a manipulator arm or actuated mount (such as a camera gimbal)

Differential-drive steering: Also referred to as "skid-steer", where turning is achieved by varying left and right wheel speeds independently.

Ackermann steering: The steering geometry used in cars and trucks whereby the front wheels pivot for turning.

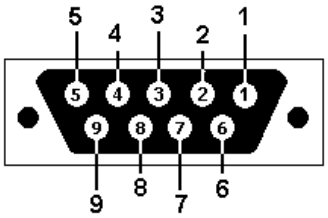
Hardware

The protocol is a bidirectional binary serial protocol intended for use over RS-232 or similar hardware. Settings are **115200 baud, 8 data bits, no parity, 1 stop bit**. If higher bandwidth communication is required, the protocol may also be packaged verbatim within TCP/IP or UDP packets if desired, with one protocol message per packet. More details will be forthcoming in a future document.

Connection Reference

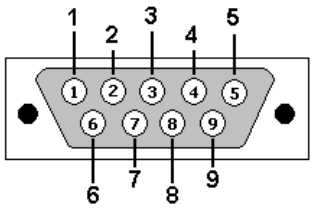
If your platform is equipped with a serial connection, the following information is provided for your reference. No null modem cable is necessary – the cable can be directly plugged into a hardware serial port or any USB to serial converter.

Female DB9 Connection (platform):



Pin	Name	Direction	Description
1	DCD	OUT	Unused (N/C)
2	TX	OUT	Data from platform
3	RX	IN	Commands to platform
4	DTR	IN	Unused (N/C)
5	SGND	N/A	Ground
6	DSR	OUT	Unused (N/C)
7	RTS	IN	Unused (N/C)
8	CTS	OUT	Unused (N/C)
9	RI	OUT	Unused (N/C)

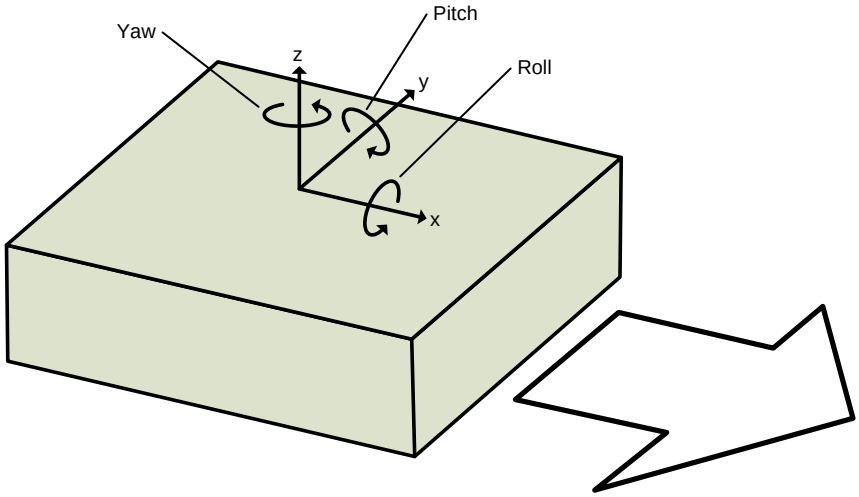
Male DB9 Connection (control computer):



Pin	Name	Direction	Description
1	DCD	IN	Unused
2	RX	IN	Data from platform
3	TX	OUT	Commands to platform
4	DTR	OUT	Unused
5	SGND	N/A	Ground
6	DSR	IN	Unused
7	RTS	OUT	Unused
8	CTS	IN	Unused
9	RI	IN	Unused

Mobile Platform Frame Of Reference

The reference frame fixed to mobile platforms is based on **ISO 8855** ("Vehicle Dynamics and Road-Holding Ability – Vocabulary"). The direction of the axes differ from those used for roll, pitch, and yaw in aircraft, and care should be taken to ensure that the data is being interpreted correctly.



Data Representations

All decimal quantities are represented by fixed-point signed numbers with prespecified scales. For example, a two byte number with a scale of **1000** will represent numbers in the range **[-32.768, 32.767]**, while the same number with a scale of **1** represents **[-32768, 32767]**.

$$RAW\ DATA = \frac{REAL\ DATA}{SCALE}$$

For the purposes of this document, a **char/byte** is an 8 bit field, a **short** is a 16 bit field, and an **int** is a 32 bit field. Two's complement is used for signed fields. **byte** implies an **unsigned char**.

Data Format

Package

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SOH								LENGTH																VERSION							
TIMESTAMP																															
FLAGS								MESSAGE TYPE																STX							
PAYLOAD (VARIABLE LENGTH)																															
																CHECKSUM															

Field Description

Field	Size
SOH	1 byte
Length	2 bytes
Version	1 byte
Timestamp	4 bytes
Flags	1 byte
Message Type	2 bytes
STX	1 byte
Payload	*n*-bytes
Checksum	2 bytes

SOH

0xAA, indicates start of data transmission.

Length

First byte in the field: Number of bytes following the field (but not including). Second byte: Compliment of the first.
Example:
Number of bytes following: 0x1D

Field	Value
Length[0]	0x1D
Length[1]	0xE2

Version

Version of protocol – currently 0x1.

Timestamp

4 bytes representing the time of message transmission, in milliseconds. Little-endian byte order. When data is returned from the platform, the timestamp is the platform clock, which resets to 0 when the platform is reset. However, the platform clock can be changed via **0x0005 – "Set Platform Time".**

As the timestamp can represent at most 50 days, it is recommended that the program transmits "milliseconds since program start" instead of using a measure based off the Unix epoch (unless the top bits are to be truncated).

For ease of processing any returned acknowledgements, no two sent messages should have the same timestamp. To avoid this, refrain from sending messages at a rate faster than 1 KHz. If serial communication hardware is in use, this will be enforced by the 115200 maximum baud rate.

Flags

One byte, as described below.

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	Suppress ACK

[0] Suppress ACK
Platform is not to acknowledge this command or request. Not recommended for use in the field, but may be of assistance in the initial development of custom implementations.

Message Type

The type of data contained in the payload.

STX

0x55, indicates start of data payload.

Payload

Payload contents. Varies for each message type.

Checksum

16 bit CRC-CCITT, encompassing the entire message. See Appendix A for further details about the checksum. Checksum is transmitted in a little-endian byte order.

Acknowledgements

The platform will acknowledge each command or data request it receives immediately upon message processing, unless the header has the flag set to suppress acknowledgements. An acknowledgement from the platform has the same format as the incoming message. The timestamp is the same as the message being acknowledged, and the payload is a two byte result code. If no issues were found in the command or request, the result code will be two empty bytes (no bits set). Otherwise, the result code will be as described below.

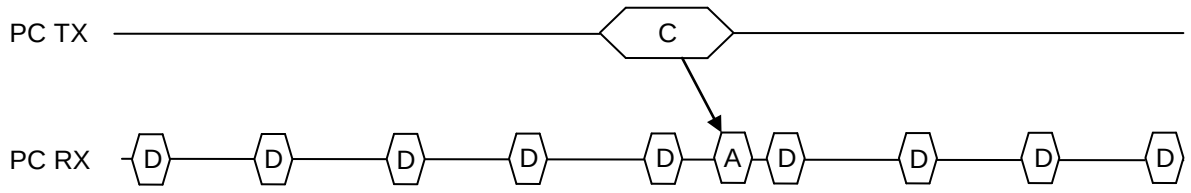
7	6	5	4	3	2	1	0
-	Too many message types	Frequency too high	No bandwidth	Out of range	Bad format	Type not supported	Bad checksum
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-	-	-	-	-	-	-	-

[0] Bad checksum
The command or request was received, but the checksum is incorrect.
[1] Type not supported
The platform configuration being used does not support this command or request.
[2] Bad format
(commands only) The command or request is in an incorrect format.
[3] Out of range
One or more of the parameters in the payload are out of range of the acceptable values for the command.
[4] No bandwidth
(requests only) There is not enough bandwidth remaining on the communication line to add this subscription.
[5] Frequency too high
(requests only) The desired subscription frequency is too high.
[6] Too many message types
(requests only) Too many different message types are being subscribed to.

The second result code byte is currently unused. Length & checksum are recalculated to match the new two byte payload.

Data Flow

The below figure shows a typical sequence of commands. The platform is sending data **(D)** with a regular frequency when the control computer sends a command **(C)**. The platform acknowledges the command immediately **(A)** and continues sending data.



Commands

Overview

These messages issue control commands to the platform. Message types range from **0x0001** to **0x3FFF**.

Messages

Message Type	*Name*	*Hardware Requirements*
0x0002	Set platform name	-
0x0005	Set platform time	-
0x0010	Set safety system	-
0x0200	Set differential speeds	Drivetrain feedback, differential drive platform
0x0201	Set differential control constants	Drivetrain feedback, differential drive platform
0x0202	Set differential output	Differential drive platform
0x0203	Set Ackermann output	Ackermann platform
0x0204	Set velocity	Mobile platform
0x0205	Set turn	Mobile platform
0x0206	Set Ackermann control constants	Drivetrain feedback, Ackermann platform
0x0210	Set max speed	Mobile platform
0x0211	Set max accel	Mobile platform
0x0212	Set gear setpoint	Gearbox platform
0x0300	Set GPADC output	-
0x0301	Set GPIO direction	-
0x0302	Set GPIO output	-
0x0400	Set pan/tilt/zoom position	Pan/tilt mount
0x0802	Configure encoders	Drivetrain feedback
0x1010	Set absolute joint position setpoints	Manipulator hardware
0x1011	Set relative joint position setpoints	Manipulator hardware
0x1012	Set joint control constants	Manipulator hardware
0x1013	Run joint homing sequence	Manipulator hardware
0x1020	Set end effector position setpoint	Manipulator hardware
0x1021	Set end effector pose setpoint	Manipulator hardware
0x2000	Reset processor	-
0x2001	Restore system settings	-
0x2002	Store system settings	-

Set Platform Name

Description: Sets a human-readable name for the platform.
Message Type: 0x0002
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Length	byte	1 byte	-	-	-
Name	ASCII	*n*-bytes	-	-	-

Length: Total number of characters in ***Name***. The maximum length is 64 characters.
Name: ASCII string representing the platform name

Set Platform Time

Description: Sets the platform internal clock to the provided value.
Message Type: 0x0005
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Time	unsigned int	4 bytes	-	-	ms

Time: The new value for the platform internal clock (little-endian byte order).

Set Safety System

Description: Controls the software-controlled aspects of the platform safety system.
Message Type: 0x0010
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Flags	unsigned short	2 bytes	-	-	-

Flags: The new settings for the onboard safety system. Refer to the platform documentation for the specific meaning.

Set Differential Speeds

Description: Sets independent wheel speeds for a differential-drive platform.
Message Type: 0x0200
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Left speed	signed short	2 bytes	100	[-320,320]	m/s
Right speed	signed short	2 bytes	100	[-320,320]	m/s
Left accel	signed short	2 bytes	100	[0, 320]	m/s^2
Right accel	signed short	2 bytes	100	[0, 320]	m/s^2

Left/right speed: The desired speed of the respective side's wheels or tracks
Left/right accel: The desired magnitude of the respective side's wheel/track acceleration

Set Differential Control Constants

Description: Sets the control constants for the left and right sides of a differential drive platform. Input into each controller is the speed error (in m/s), and the output is a percentage of the corresponding motor's nominal input voltage.
Message Type: 0x0201
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Left P	signed short	2 bytes	100	[-320,320]	-
Left I	signed short	2 bytes	100	[-320,320]	-
Left D	signed short	2 bytes	100	[-320,320]	-
Left feed-forward	signed short	2 bytes	100	[-320,320]	-
Left stiction compensation	signed short	2 bytes	100	[0,100]	%
Left integral limit	signed short	2 bytes	100	[0,100]	%
Right P	signed short	2 bytes	100	[-320,320]	-
Right I	signed short	2 bytes	100	[-320,320]	-
Right D	signed short	2 bytes	100	[-320,320]	-
Right feed-forward	signed short	2 bytes	100	[-320,320]	-
Right stiction compensation	signed short	2 bytes	100	[0,100]	%
Right integral limit	signed short	2 bytes	100	[0,100]	%

Left/right P: The proportional constant for the respective side's control loop
Left/right I: The integral constant for the respective side's control loop
Left/right D: The derivative constant for the respective side's control loop
Left/right feed-forward: The feed-forward constant for the respective side's control loop
Left/right stiction compensation: An offset (in % of motor voltage) for the respective side's control output.
Left/right integral limit: A limit (in % of motor voltage) to the amount that the integral term can contribute to the output of the respective side's controller.

Set Differential Output

Description: Sets the raw motor outputs on a platform with a differential configuration, as a percentage of the maximum possible output. The maximum possible output is platform dependent. If the platform is battery powered, the maximum output voltage will be lower than that achievable on a full charge to ensure consistent performance as the batteries are drained.
Message Type: 0x0202
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Left	signed short	2 bytes	100	[-100,100]	%
Right	signed short	2 bytes	100	[-100,100]	%

Left: Set the left differential output to this percentage of the maximum, where 0 is off, 100 is full forward, and -100 is full reverse.
Right: Set the right differential output to this percentage of the maximum, where 0 is off, 100 is full forward, and -100 is full reverse.

Set Ackermann Output

Description: Sets servo positioning for an Ackermann-based platform. If the vehicle is not equipped with a brake, the brake setpoint will be ignored (but should still be sent).
Message Type: 0x0203
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Steering	signed short	2 bytes	100	[-100,100]	%
Throttle	signed short	2 bytes	100	[-100,100]	%
Brake	signed short	2 bytes	100	[0,100]	%

Steering: Set the steering to this percentage of the maximum, where 0 is straight ahead. A positive value corresponds to a left turn.
Throttle: Set the throttle to this percentage of the maximum, where 0 is closed, 100 is full forward, and -100 is full reverse. If the platform is battery powered, the maximum throttle may be kept lower than the possible maximum at full battery charge, to ensure consistent performance as the batteries are drained. Depending on platform configuration, the vehicle may not support reversing through a negative throttle value, but may instead require a gear change via command **0x0212 – "Change Gear".**
Brake: Set the brake to this percentage of the maximum, where 0 is completely released and 100 is completely engaged.

Set Velocity Setpoint

Description: Sets desired vehicle velocity. Platform will use the applicable kinematic model to the best of its ability to maintain these setpoints.
Message Type: 0x0204
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Translational velocity	signed short	2 bytes	100	[-320,320]	m/s
Rotational velocity	signed short	2 bytes	100	[-320,320]	rad/s
Translational acceleration	signed short	2 bytes	100	[0, 320]	m/s^2

Translational velocity: The desired translational velocity of the vehicle.
Rotational velocity: The desired rotational velocity of the vehicle. A positive value corresponds to a left turn.
Translational acceleration: The desired magnitude of the translational acceleration of the vehicle

Set Turn

Description: Sets a desired vehicle turn, defined by a velocity and turn radius. Platform will use the applicable kinematic model to the best of its ability to maintain these parameters.
Message Type: 0x0205
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Translational velocity	signed short	2 bytes	100	[-320,320]	m/s
Turn radius	signed short	2 bytes	100	[-320, 320]	m
Translational acceleration	signed short	2 bytes	100	[0, 320]	m/s^2

Translational velocity: The desired translational velocity of the vehicle.
Turn radius: The desired turn radius of the vehicle. A positive value corresponds to a left turn.
Translational acceleration: The desired magnitude of the translational acceleration of the vehicle

Set Ackermann Control Constants

Description: Sets the control constants for the speed and heading control of an Ackermann-steered platform. Input into the speed controller is the speed error (in m/s), input into the heading control is the heading error (in rad/s) and the output is a percentage of the corresponding motor's nominal input voltage. If the platform is not equipped with a speed or heading feedback, the corresponding control constants will be ignored (though the feed-forward term may still be used for open-loop output).
Message Type: 0x0206
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Speed P	signed short	2 bytes	100	[-320,320]	-
Speed I	signed short	2 bytes	100	[-320,320]	-
Speed D	signed short	2 bytes	100	[-320,320]	-
Speed feed-forward	signed short	2 bytes	100	[-320,320]	-
Speed stiction compensation	signed short	2 bytes	100	[0,100]	%
Speed integral limit	signed short	2 bytes	100	[0,100]	%
Heading P	signed short	2 bytes	100	[-320,320]	-
Heading I	signed short	2 bytes	100	[-320,320]	-
Heading D	signed short	2 bytes	100	[-320,320]	-
Heading feed-forward	signed short	2 bytes	100	[-320,320]	-
Heading stiction compensation	signed short	2 bytes	100	[0,100]	%
Heading integral limit	signed short	2 bytes	100	[0,100]	%

Speed/heading P: The proportional constant for the respective control loop
Speed/heading I: The integral constant for the respective control loop
Speed/heading D: The derivative constant for the respective control loop
Speed/heading feed-forward: The feed-forward constant for the respective control loop
Speed/heading stiction compensation: An offset (in % of motor voltage) for the respective control output.
Speed/heading integral limit: A limit (in % of motor voltage) to the amount that the integral term can contribute to the output of the respective controller.

Set Max Speed

Description: Sets the maximum translational speed for the platform.
Message Type: 0x0210
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Max forward speed	signed short	2 bytes	100	[0,320]	m/s
Max reverse speed	signed short	2 bytes	100	[0,320]	m/s

Max forward speed: The maximum forward translational speed.
Max reverse speed: The maximum reverse translational speed

Set Max Accel

Description: Sets the maximum translational acceleration for the platform.
Message Type: 0x0211
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Max forward accel	signed short	2 bytes	100	[0,320]	m/s^2
Max reverse accel	signed short	2 bytes	100	[0,320]	m/s^2

Max forward accel: The maximum forward translational acceleration
Max reverse accel: The maximum reverse translational acceleration

Change Gear

Description: Sets the desired gear.
Message Type: 0x0212
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Gear	signed char	1 byte	-	-	-

Gear: The desired gear for the vehicle's transmission. 0 corresponds to neutral, -1 to parked. Gears > 0 are forward gears in order of their gear ratio. Gears < -1 are reverse gears in order of their gear ratio.

Set GPADC Output

Description: Sets the value of specified generic analog output channels. The number of output channels and maximum voltage for each is platform dependent. The amount of bytes in the payload is equal to *n*3 + 1*, where *n* is equal to the amount of channels being controlled by the payload.
Message Type: 0x0300
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of channels	byte	1 byte	-	-	-
Channel ID 1	byte	1 byte	-	-	-
Channel value 1	unsigned short	2 bytes	-	-	-
Continue for the remaining channels					
Channel ID *n*	byte	1 byte	-	-	-
Channel value *n*	unsigned short	2 bytes	-	-	-

 # channels: The amount of channels to modify
Channel ID 1: The identifier of the first channel to be controlled. Identifiers are dependent on the specific platform configuration.
Channel value 1: The value to output to the channel identified by the preceding field (scaled such that the maximum (0xFFFF) corresponds to the maximum voltage for the specified analog output).
Channel ID n: The identifier of the last channel to be controlled. Identifiers are dependent on the specific platform configuration.
Channel value n: The value to output to the channel identified by the preceding field (scaled such that the maximum (0xFFFF) corresponds to the maximum voltage for the specified analog output).

Set GPIO Direction

Description: Sets the direction of the generic digital I/O channels. The number of I/O channels is platform dependent.
Message Type: 0x0301
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Bitmask	unsigned int	4 bytes	-	-	-
Direction	unsigned int	4 bytes	-	-	-

Bitmask: Set bits high to configure the corresponding channels. A platform may not necessarily have 32 GPIO channels available.
Direction: Write a 1 to configure channel as an output, 0 to configure as an input. Will only be written if the corresponding bit in the bitmask is high.

Set GPIO Output

Description: Sets the value of the generic digital output channels. The number of I/O channels is platform dependent.
Message Type: 0x0302
Message Fields:

Field	Type	Size	Range	Units
Bitmask	unsigned int	4 bytes	-	-
Output value	unsigned int	4 bytes	-	-

Bitmask: Set bits high to write to these channels. A platform may not necessarily have 32 GPIO channels available.
Output value: Write a 1 for a HIGH output, 0 for a LOW output. Will only be written if the corresponding bit in the bitmask is high.

Set Pan/Tilt/Zoom Position

Description: Sets pan, tilt, and zoom settings on a specified camera mount. If mount does not have a pan, tilt, or zoom axis, send 0 as the appropriate setpoint. Pan, tilt, and zoom limits are dependent on the camera and camera mount. Zero position and direction of rotation for the pan and tilt axes are dependent on the camera mount.
Message Type: 0x0400
Message Fields:

Field	Type	Size	Scale	Range	Units
Mount	byte	1 byte	-	-	-
Pan	signed short	2 bytes	100	[-180,180]	deg
Tilt	signed short	2 bytes	100	[-180, 180]	deg
Zoom	signed short	2 bytes	100	[1, 320]	-

Mount: The desired camera mount to move
Pan: The desired pan position of the camera mount
Tilt: The desired pan position of the camera mount
Zoom: The desired zoom level of the camera

Configure Encoders

Description: Sets PPR and scale factors for the platform's encoders. The position of each encoder is platform-dependent. If the encoder is a quadrature unit, PPR = CPR * 4. The PPR refers to the specific PPR rating of the encoder itself, while the scale factor is the relationship between the encoder and the final output. For translational actuators or drive wheels, this unit is in m/rev. For rotational actuators (such as pan/tilt or arm servos), this unit is in rev/rev.
Message Type: 0x0802
Message Fields:

Field	Type	Size	Scale	Range	Units
PPR of encoder 1	signed short	2 bytes	1	[0, $32x10^3$]	PPR
Scale factor of encoder 1	signed short	2 bytes	1000	[-32,32]	m/rev *or* rev/rev
Continue for the remaining encoders					
PPR of encoder *n*	signed short	2 bytes	1	[0, $32x10^3$]	PPR
Scale factor of encoder *n*	signed short	2 bytes	1000	[-32,32]	m *or* rev

PPR of encoder 1: The amount of pulses per revolution of the first encoder.
Scale factor of encoder 1: The amount the final output moves per revolution of the encoder (units dependent on output type).
PPR of encoder n: The amount of total pulses per revolution of the last encoder***.***
Scale factor of encoder n: The amount the final output moves per revolution of the encoder (units dependent on output type).

Set Absolute Joint Position

Description: Sets a set of desired joint positions. Platform will use its onboard control to the best of its ability to maintain these positions.
Message Type: 0x1010
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# joints	byte	1 byte	-	-	-
Joint ID 1	byte	1 byte	-	-	-
Joint angle 1	signed short	2 bytes	10000	[-π, π]	rad
Continue for the remaining joints					
Joint ID *n*	byte	1 byte	-	-	-
Joint angle *n*	signed short	2 bytes	10000	[-π, π]	rad

 # joints: The amount of joints being controlled by this command.
Joint ID 1: The identifier of the first joint to be controlled. Identifiers are dependent on the specific manipulator hardware.
Joint angle 1: The angle to move the first joint to, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware.
Joint ID n: The identifier of the last joint to be controlled. Identifiers are dependent on the specific manipulator hardware.
Joint angle n: The angle to move the last joint to, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware.

Set Relative Joint Position

Description: Sets a set of relative joint positions. Platform will use its onboard control to the best of its ability to maintain these positions.
Message Type: 0x1011
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# joints	byte	1 byte	-	-	-
Joint ID 1	byte	1 byte	-	-	-
Joint angle 1	signed short	2 bytes	10000	[-3.1416, 3.1416]	rad
Continue for the remaining joints					
Joint ID *n*	byte	1 byte	-	-	-
Joint angle *n*	signed short	2 bytes	10000	[-3.1416, 3.1416]	rad

 # joints: The amount of joints being controlled by this command.
Joint ID 1: The identifier of the first joint to be controlled. Identifiers are dependent on the specific manipulator hardware.
Joint angle 1: The relative angle to move the first joint to, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware. The base angle is the angle of the joint before reception of the command.
Joint ID n: The identifier of the last joint to be controlled. Identifiers are dependent on the specific manipulator hardware.
Joint angle n: The relative angle to move the last joint to, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware. The base angle is the angle of the joint before reception of the command.

Set Joint Control Constants

Description: Sets the control constants for the specified joint on a manipulator. Input into each controller is the position error (in radians), and the output is the torque the corresponding motor's should exert.
Message Type: 0x1012

Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Joint ID	byte	byte	-	-	-
P	signed short	2 bytes	100	[-320,320]	-
I	signed short	2 bytes	100	[-320,320]	-
D	signed short	2 bytes	100	[-320,320]	-
Feed-forward	signed short	2 bytes	100	[-320,320]	-
Stiction compensation	signed short	2 bytes	100	[0,100]	N-m
Integral limit	signed short	2 bytes	100	[0,100]	N-m

Joint ID: The joint being controlled. Identifiers are dependent on the specific manipulator hardware.
P: The proportional constant for the control loop
I: The integral constant for the control loop
D: The derivative constant for the control loop
Feed-forward: The feed-forward constant for the respective side's control loop
Stiction compensation: An offset (in N-m) for the respective side's control output used to compensate for static friction.
Integral limit: A limit (in N-m) to the amount that the integral term can contribute to the output of the controller.

Run Joint Homing Sequence

Description: Homes a specified joint. Uses position feedback and limit sensors as available. When homing is complete, the joint will be moved to its zero angle.
Message Type: 0x1013
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Joint ID	byte	byte	-	-	-

Joint ID: The joint being homed. Identifiers are dependent on the specific manipulator hardware.

Set End Effector Position

Description: Moves the end effector origin to a specified position in the robot frame, without placing any constraints on its orientation. The robot frame definition is dependent on the specific manipulator hardware.
Message Type: 0x1020
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
x	signed short	2 bytes	1000	[-32,32]	m
y	signed short	2 bytes	1000	[-32,32]	m
z	signed short	2 bytes	1000	[-32,32]	m

x: The desired x position in the robot frame of the end effector origin
y: The desired y position in the robot frame of the end effector origin
z: The desired z position in the robot frame of the end effector origin

Set End Effector Pose

Description: Moves the end effector origin to a specified pose in the robot frame. The robot frame definition is dependent on the specific manipulator hardware.
Message Type: 0x1021
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
x	signed short	2 bytes	1000	[-32,32]	m
y	signed short	2 bytes	1000	[-32,32]	m
z	signed short	2 bytes	1000	[-32,32]	m
Roll	signed short	2 bytes	1000	[-π, π]	rad

Pitch	signed short	2 bytes	1000	[-π, π]	rad
Yaw	signed short	2 bytes	1000	[-π, π]	rad

x: The desired x position in the robot frame of the end effector origin
y: The desired y position in the robot frame of the end effector origin
z: The desired z position in the robot frame of the end effector origin
Roll: The desired roll of the end effector in the robot frame
Pitch: The desired pitch of the end effector in the robot frame
Yaw: The desired yaw of the end effector in the robot frame

Reset Processor

Description: Resets the control processor.
Message Type: 0x2000
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Passcode	unsigned short	2 bytes	-	-	-

Passcode: Fixed value of 0x3A18.

Restore System Settings

Description: Restores system settings from user storage or factory defaults.
Message Type: 0x2001
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Passcode	unsigned short	2 bytes	-	-	-
Flags	byte	1 byte	-	-	-

Passcode: Fixed value of 0x3A18.
Flags: Set to 0x1 to restore system settings from user storage, 0x2 to restore to factory defaults

Store System Settings

Description: Stores current configuration into flash memory.
Message Type: 0x2002
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Passcode	unsigned short	2 bytes	-	-	-

Passcode: Fixed value of 0x3A18.

Requests

Overview

These messages are requests for data from the platform. The control computer can also subscribe to repeated data requests at specified frequencies. The maximum per-message frequency is defined on a per-platform basis. To ensure low-latency replies, the platform will limit the total amount of messages per second it sends. Additionally, each platform has a limit on the maximum different message types that can be subscribed to at any given time. As before, these limits are defined on a per-platform basis.

When new subscriptions are received which would overwhelm the communication bandwidth, an error message is returned in the result code. The platform will always acknowledge a request for subscription. Request types range from **0x4000** to **0x7FFF**. Each request has a corresponding data message with a message type of **Message Type + 0x4000**.

Subscriptions

For all messages there is a two-byte ***subscription*** field. Setting this to **0** is a request for a single immediate response (without affecting the subscription setting, if it exists), setting it to **0xFFFF** is a request to cancel the subscription to this message (if it exists), and any other value corresponds to a desired frequency, in Hz. Too many subscriptions, frequencies that are too high, and requests which would saturate the bandwidth of the communication systems will result in the system setting the ***Too Many Message Types*** bit, the ***Frequency Too High*** bit, and the ***No Bandwidth*** bit respectively in the result code.

The platform will reply with the data message of the specified type as soon as the request is processed. The transmission times of subsequent messages of the same type are based off the sending time of this message. Subsequent requests for different frequencies will likewise cause immediate data transmission.

Messages

Message Type	*Name*	*Hardware Requirements*
0x4000	Request echo	-
0x4001	Request platform info	-
0x4002	Request platform name	-
0x4003	Request firmware info	-
0x4004	Request system status	-
0x4005	Request power system status	-
0x4006	Request processor status	-
0x4010	Request safety system status	-
0x4200	Request differential speeds	Drivetrain feedback, differential drive platform
0x4201	Request differential control constants	Drivetrain feedback, differential drive platform
0x4202	Request differential output	Differential drive platform
0x4203	Request Ackermann output	Ackermann platform
0x4204	Request velocity setpoint	Mobile platform
0x4205	Request turn setpoint	Mobile platform
0x4206	Request Ackermann control constants	Drivetrain feedback, Ackermann platform
0x4210	Request max speed	Mobile platform
0x4211	Request max accel	Mobile platform
0x4212	Request gear setpoint	Gearbox platform
0x4300	Request GPADC output setpoints	-

0x4301	Request GPIO status	-
0x4303	Request GPADC inputs	-
0x4400	Request pan/tilt/zoom position	Pan/tilt mount
0x4500	Request rangefinder data	Sonar/IR interface package
0x4501	Request rangefinder data & timing	Sonar/IR interface package
0x4600	Request platform orientation	Tilt/enhanced IMU package
0x4601	Request platform rotational rates	Gyro/IMU/enhanced IMU package
0x4602	Request platform acceleration	Accel/IMU/enhanced IMU package
0x4603	Request 6-axis data	IMU/enhanced IMU package
0x4604	Request 6-axis & orientation data	IMU/enhanced IMU package
0x4606	Request platform magnetometers	Compass/enhanced IMU package
0x4800	Request encoder data	Drivetrain feedback
0x4801	Request raw encoder data	Drivetrain feedback
0x4802	Request encoder config	Drivetrain feedback
0x5010	Request absolute joint position setpoints	Manipulator hardware
0x5011	Request relative joint position setpoints	Manipulator hardware
0x5012	Request joint control constants	Manipulator hardware
0x5013	Request joint homing sequence status	Manipulator hardware
0x5014	Request joint torques	Manipulator hardware
0x5020	Request end effector position setpoint	Manipulator hardware
0x5021	Request end effector pose setpoint	Manipulator hardware
0x5022	Request calculated end effector pose	Manipulator hardware

Request Echo

Description: Requests the vehicle acknowledge with an empty transmission. Effectively a “ping.”
Message Type: 0x4000
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Platform Info

Description: Requests platform configuration information.
Message Type: 0x4001
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Platform Name

Description: Requests the current platform name (default or as set by Set Platform Name).
Message Type: 0x4002
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Firmware Information

Description: Requests firmware version information.
Message Type: 0x4003
Message Fields:

Field	Type	Size
Subscription	unsigned short	2 bytes

Request System Status

Description: Requests current system status – uptime, voltages, temperature, etc.
Message Type: 0x4004
Message Fields:

Field	Type	Size
Subscription	unsigned short	2 bytes

Request Power System Status

Description: Requests current power system status - estimated battery percentages, batteries in use, types of batteries attached and their capacities.
Message Type: 0x4005
Message Fields:

Field	Type	Size
Subscription	unsigned short	2 bytes

Request Processor Status

Description: Requests the processor load information.
Message Type: 0x4006
Message Fields:

Field	Type	Size
Subscription	unsigned short	2 bytes

Request Safety System Status

Description: Requests the current status of any onboard safety systems
Message Type: 0x4010
Message Fields:

Field	Type	Size
Subscription	unsigned short	2 bytes

Request Differential Speeds

Description: Requests the current desired wheel speeds for the left and right sides of a differential-drive platform.
Message Type: 0x4200
Message Fields:

Field	Type	Size
Subscription	unsigned short	2 bytes

Request Differential Control Constants

Description: Requests the current control constants for the motor control on a differential drive platform.
Message Type: 0x4201
Message Fields:

Field	Type	Size

Subscription	unsigned short	2 bytes

Request Differential Output

Description: Requests the current raw differential output (left and right actuators).
Message Type: 0x4202
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Ackermann Output

Description: Requests the current raw output for an Ackermann-drive platform.
Message Type: 0x4203
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Velocity Setpoint

Description: Requests the current general velocity setpoint (translational, rotational velocity, translational acceleration).
Message Type: 0x4204
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Turn Setpoint

Description: Requests the current turn setpoint (translational velocity, turn radius, translational acceleration)
Message Type: 0x4205
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Ackermann Control Constants

Description: Requests the current control constants for the motor control on an Ackermann drive platform.
Message Type: 0x4206
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Max Speed

Description: Requests the current maximum speed settings (forward and reverse).
Message Type: 0x4210
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Max Accel

Description: Requests the current maximum acceleration settings (forward and reverse).
Message Type: 0x4211
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Gear Setpoint

Description: Requests the current desired gear.
Message Type: 0x4212
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request GPADC Output Setpoints

Description: Requests the values for the general-purpose ADC output channels.
Message Type: 0x4300
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request GPIO Status

Description: Requests the values and direction settings for the general-purpose IO lines.
Message Type: 0x4301
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request GPADC Inputs

Description: Requests the values for the general-purpose ADC input channels.
Message Type: 0x4303
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Pan/Tilt/Zoom Position

Description: Requests the position and zoom setting for a specified camera mount.
Message Type: 0x4400
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes
Mount	byte	1 byte

Mount: The mount being requested. The number of mounts available is dependent on the platform hardware.

Request Rangefinder Data

Description: Requests the values for the onboard rangefinders.
Message Type: 0x4500
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Rangefinder Data & Timing

Description: Requests the values for the onboard rangefinders, along with timestamps for when the data was received for each sensor.
Message Type: 0x4501
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Platform Orientation

Description: Requests the vehicle's best estimate of its orientation.
Message Type: 0x4600
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Platform Angular Rate

Description: Requests the vehicle's best estimate of its angular rate
Message Type: 0x4601
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Platform Acceleration

Description: Requests the vehicle's best estimate of its translational acceleration.
Message Type: 0x4602
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request 6-Axis Data

Description: Requests the vehicle's best estimate of its angular rate & translational acceleration.
Message Type: 0x4603
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request 6-Axis & Orientation Data

Description: Requests the vehicle's best estimate of its angular rate, translational acceleration, and orientation.

Message Type: 0x4604
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Platform Magnetometers

Description: Requests the vehicle's measurement of the surrounding magnetic fields.
Message Type: 0x4606
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Encoders

Description: Requests position and speed estimates of each platform encoder.
Message Type: 0x4800
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Raw Encoders

Description: Requests raw tick count of each platform encoder.
Message Type: 0x4801
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Encoder Config

Description: Requests the PPR and scale factors for the platform's encoders.
Message Type: 0x4802
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Absolute Joint Position Setpoint

Description: Requests the current desired absolute joint positions.
Message Type: 0x5010
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Relative Joint Position Setpoint

Description: Requests the current desired relative joint positions. As these positions are relative to the last angle, this information is not necessarily helpful after the movement has started.
Message Type: 0x5011
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Joint Control Constants

Description: Requests the control constants for a specified joint.
Message Type: 0x5012
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes
Joint	byte	1 byte

Joint: The joint for which the constants are being requested. Joints are defined on a per-manipulator basis.

Request Joint Homing Status

Description: Requests the homing status for each joint – if they have been homed and if they have been moved from this position.
Message Type: 0x5013
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Joint Torques

Description: Requests the torques each joint is experiencing
Message Type: 0x5014
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request End Effector Position Setpoint

Description: Requests the current desired end effector position (no orientation)
Message Type: 0x5020
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request End Effector Pose Setpoint

Description: Requests the current desired end effector pose
Message Type: 0x5021
Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Request Calculated End Effector Pose

Description: Requests the current end effector pose, calculated based on the available feedback sensors
Message Type: 0x5022

Message Fields:

Field	*Type*	*Size*
Subscription	unsigned short	2 bytes

Data

Overview

This section describes the messages which the platform returns to the control computer when requested or subscribed to. There is no need for the control computer to acknowledge them. Message types range from **0x8000** to **0xBFFF**. Each data message corresponds to a request with a message type of **Message Type – 0x4000**.

Messages

Message Type	*Name*	*Hardware Requirements*
0x8000	Echo data	-
0x8001	Platform info	-
0x8002	Platform name	-
0x8003	Firmware info	-
0x8004	System status data	-
0x8005	Power system status data	-
0x8006	Processor status data	-
0x8010	Safety system status data	-
0x8200	Differential speed data	Drivetrain feedback, differential drive platform
0x8201	Differential control constants	Drivetrain feedback, differential drive platform
0x8202	Differential output data	Differential drive platform
0x8203	Ackermann output data	Ackermann platform
0x8204	Velocity setpoint data	Mobile platform
0x8205	Turn setpoint data	Mobile platform
0x8206	Ackermann control constants	Drivetrain feedback, Ackermann platform
0x8210	Max speed data	Mobile platform
0x8211	Max accel data	Mobile platform
0x8212	Gear setpoint data	Gearbox platform
0x8300	GPADC output setpoint data	-
0x8301	GPIO data	-
0x8303	GPADC input data	-
0x8400	Pan/tilt/zoom position data	Pan/tilt mount
0x8500	Rangefinder data	Sonar/IR interface package
0x8501	Rangefinder data & timing	Sonar/IR interface package
0x8600	Platform orientation data	Tilt/enhanced IMU package
0x8601	Platform rotational rate data	Gyro/IMU/enhanced IMU package
0x8602	Platform acceleration data	Accel/IMU/enhanced IMU package
0x8603	6-axis data	IMU/enhanced IMU package
0x8604	6-axis & orientation data	IMU/enhanced IMU package
0x8606	Platform magnetometer data	Compass/enhanced IMU package
0x8800	Encoder data	Drivetrain feedback
0x8801	Raw encoder data	Drivetrain feedback
0x8802	Encoder config data	Drivetrain feedback
0x9010	Absolute joint position data	Manipulator hardware
0x9011	Relative joint position data	Manipulator hardware
0x9012	Joint control constant data	Manipulator hardware
0x9013	Joint homing status	Manipulator hardware
0x9014	Joint torque data	Manipulator hardware
0x9020	End effector position data	Manipulator hardware

0x9021	End effector pose data	Manipulator hardware
0x9022	End effector pose data	Manipulator hardware

Echo Data

Description: Blank echo message.
Message Type: 0x8000
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
-	-	-	-	-	-

Platform Info

Description: Information about the specific platform configuration.
Message Type: 0x8001
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Model Length	byte	1 byte	-	-	-
Model	ASCII	*n*-bytes	-	-	-
Revision	byte	1 byte	-	-	-
Serial	unsigned int	4 bytes	-	-	-

Model Length: Total number of characters in the **model** string.
Model: ASCII string representing the platform model, where the number of bytes is equal to the preceding **length** field. Not null-terminated.
Revision: The platform model's revision number.
Serial: The platform specific, unique serial number. Obtained at the time of purchase.

Platform Name

Description: A human readable name of the platform. Default is "Clearpath1".
Message Type: 0x8002

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Length	byte	1 byte	-	-	-
Name	ASCII	*n*-bytes	-	-	-

Length: Total number of characters in the name.
Name: ASCII string representing the platform name, where the number of bytes is equal to the preceding **length** field. Not null-terminated.

Firmware Info

Description: Information about the version of firmware the vehicle is running.
Message Type: 0x8003
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Major firmware version	byte	1 byte	-	-	-
Minor firmware version	byte	1 byte	-	-	-
Major protocol version	byte	1 byte	-	-	-
Minor protocol version	byte	1 byte	-	-	-
Firmware write time	unsigned int	4 bytes	-	-	-

Major firmware version: The major version number in the vehicle's control firmware
Minor firmware version: The minor version number in the vehicle's control firmware

Major protocol version: The major version number of the protocol library in use
Minor protocol version: The minor version number of the protocol library in use
Write time: 4 bytes representing the last time the firmware was programmed. Little-endian byte order.
Bits 0-5: Minute (0-59)
Bits 6-10: Hour (0-23)
Bits 11-16: Day (1-31)
Bits 17-20: Month(1-12)
Bits 21-27: Year(2000-2127)

System Status Data

Description: Voltage, current, and temperature at varying points within the platform. The location and number of each internal sensor is platform dependent. The total size of the payload is equal to *# of voltage sensors * 2 + # of current sensors * 2 + # of temperature sensors * 2 + 3*.
Message Type: 0x8004

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Uptime	unsigned int	4 bytes	-	-	ms
# of voltage measurements	byte	1 byte	-	-	-
Voltage measurement 1	signed short	2 bytes	100	[-320,320]	V
Continue for the remaining sensors					
Voltage measurement *n*	signed short	2 bytes	100	[-320,320]	V
# of current measurements	byte	1 byte	-	-	-
Current measurement 1	signed short	2 bytes	100	[-320,320]	A
Continue for the remaining sensors					
Current measurement *n*	signed short	2 bytes	100	[-320,320]	A
# of temperature measurements	byte	1 byte	-	-	-
Temperature measurement 1	signed short	2 bytes	100	[-320,320]	°C
Continue for the remaining sensors					
Temperature measurement *n*	signed short	2 bytes	100	[-320,320]	°C

Uptime: The amount of time since the controller was turned on (in ms)
# of voltage measurements: The number of voltage measurements
Voltage measurement 1: The voltage measurement from the first sensor.
Voltage measurement n: The voltage measurement from the last sensor. In total, there are *2n* bytes returned for *n* integrated voltage sensors.
# of current measurements: The number of current measurements
Current measurement 1: The current measurement from the first sensor.
Current measurement n: The current measurement from the last sensor. In total, there are *2n* bytes returned for *n* integrated current sensors.
# of temperature measurements: The number of temperature measurements
Temperature measurement 1: The temperature measurement from the first sensor.
Temperature measurement n: The temperature measurement from the last sensor. In total, there are *2n* bytes returned for *n* integrated temperature sensors.

Power System Status Data

Description: Returns estimates of each power source's current charge state, indications of which power sources are in use, and the chemistry and capacity of each attached power source (if applicable). In some cases, units will have multiple power sources, of which only a few will be used (in the case of platforms with hot-swap capability). In other cases, multiple power sources may be in use simultaneously (in the case of platforms with dedicated drive and control power). Power source locations are platform-dependent. The total size of the payload is equal to *# sources * 5 + 1*
Message Type: 0x8005

Field	*Type*	*Size*	*Scale*	*Range*	*Units*

# of batteries	byte	1 byte	-	-	-
Charge state estimate 1	signed short	2 bytes	100	[0,100]	%
Continue for the remaining batteries					
Charge state estimate *n*	signed short	2 bytes	100	[0,100]	%
Capacity estimate 1	signed short	2 bytes	1	[0,32000]	W-Hr
Continue for the remaining sensors					
Capacity estimate *n*	signed short	2 bytes	1	[0,32000]	W-Hr
Battery description 1	byte	1 byte	-	-	-
Continue for the remaining sensors					
Battery description *n*	byte	1 byte	-	-	-

# of batteries: The number of batteries the platform supports
Charge state estimate 1: An estimate of the first battery's state of charge.
Charge state estimate n: An estimate of the last battery's state of charge.
Capacity estimate 1: An estimate of the first battery's total energy capacity.
Charge state estimate n: An estimate of the last battery's total energy capacity.
Battery description:

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
PRESENT	IN_USE	-	-	TYPE_3	TYPE_2	TYPE_1	TYPE_0

PRESENT: The battery in this position is attached. If a platform is not capable of detecting this, it will return 1.
IN_USE: The power source is in use
TYPE_3:TYPE_0: Power system type.
0x0: External supply
0x1: Lead-acid battery
0x2: Ni-Mh battery
0x8: Gas engine (for this case, the capacity estimate is in units of liters)

Processor Status Data

Description: Returns a count of the amount of time each process has failed to execute due to an error or a lack of available processor time. Process types, purposes, and desired periods are platform-dependent. The total size of the payload is equal to *# processes * 2 + 1*
Message Type: 0x8006

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of processes	byte	1 byte	-	-	-
Errors in process 1	signed short	2 bytes	1	[0,32000]	-
Continue for the remaining processes					
Errors in process *n*	signed short	2 bytes	1	[0,32000]	-

# of processes: The number of processes executing on the platform
Errors in process 1: The amount of times the first process has experienced an error
Errors in process n: The amount of times the last process has experienced an error

Safety System Status

Description: The status of the platform safety system.
Message Type: 0x8010
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Flags	unsigned short	2 bytes	-	-	-

Flags: The status for onboard safety system. Refer to the platform documentation for the specific meaning of the flags.

Differential Speed Data

Description: The independent wheel speed setpoints for a differential-drive platform.
Message Type: 0x8200
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Left speed	signed short	2 bytes	100	[-320,320]	m/s
Right speed	signed short	2 bytes	100	[-320,320]	m/s
Left accel	signed short	2 bytes	100	[0, 320]	m/s^2
Right accel	signed short	2 bytes	100	[0, 320]	m/s^2

Left/right speed: The desired speed of the respective side's wheels or tracks
Left/right accel: The desired magnitude of the respective side's wheel/track acceleration

Differential Control Constant Data

Description: The control constants for the left and right sides of a differential drive platform.
Message Type: 0x8201
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Left P	signed short	2 bytes	100	[-320,320]	-
Left I	signed short	2 bytes	100	[-320,320]	-
Left D	signed short	2 bytes	100	[-320,320]	-
Left feed-forward	signed short	2 bytes	100	[-320,320]	-
Left stiction compensation	signed short	2 bytes	100	[0,100]	%
Left integral limit	signed short	2 bytes	100	[0,100]	%
Right P	signed short	2 bytes	100	[-320,320]	-
Right I	signed short	2 bytes	100	[-320,320]	-
Right D	signed short	2 bytes	100	[-320,320]	-
Right feed-forward	signed short	2 bytes	100	[-320,320]	-
Right stiction compensation	signed short	2 bytes	100	[0,100]	%
Right integral limit	signed short	2 bytes	100	[0,100]	%

Left/right P: The proportional constant for the respective side's control loop
Left/right I: The integral constant for the respective side's control loop
Left/right D: The derivative constant for the respective side's control loop
Left/right feed-forward: The feed-forward constant for the respective side's control loop
Left/right stiction compensation: An offset (in % of motor voltage) for the respective side's control output.
Left/right integral limit: A limit (in % of motor voltage) to the amount that the integral term can contribute to the output of the respective side's controller.

Differential Output Data

Description: The raw motor outputs for a differential-drive platform. .
Message Type: 0x8202
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Left	signed short	2 bytes	100	[-100,100]	%
Right	signed short	2 bytes	100	[-100,100]	%

Left: The current left differential output as a percentage of the maximum, where 0 is off, 100 is full forward, and -100 is full reverse.
Right: The current left differential output as a percentage of the maximum, where 0 is off, 100 is full forward, and -100 is full reverse.

Ackermann Output Data

Description: The servo positioning setpoints for an Ackermann-based platform. .
Message Type: 0x8203
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Steering	signed short	2 bytes	100	[-100,100]	%
Throttle	signed short	2 bytes	100	[0,100]	%
Brake	signed short	2 bytes	100	[0,100]	%

Steering: The current steering servo position setpoint as a percentage of the maximum travel, where 0 is straight ahead. A positive value corresponds to a left turn.
Throttle: The current throttle servo position setpoint as a percentage of the maximum travel, where 0 is closed, 100 is full forward and -100 is full reverse.
Brake: The current brake servo position as a percentage of the maximum travel, where 0 is completely released and 100 is completely engaged.

Velocity Setpoint Data

Description: The desired vehicle velocity.
Message Type: 0x8204
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Translational velocity	signed short	2 bytes	100	[-320,320]	m/s
Rotational velocity	signed short	2 bytes	100	[-320,320]	rad/s
Translational acceleration	signed short	2 bytes	100	[0, 320]	m/s^2

Translational velocity: The current desired translational velocity of the vehicle.
Rotational velocity: The current desired rotational velocity of the vehicle. A positive value corresponds to a left turn.
Translational acceleration: The current desired magnitude of the translational acceleration of the vehicle

Turn Setpoint Data

Description: The desired vehicle turn.
Message Type: 0x8205
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Translational velocity	signed short	2 bytes	100	[-320,320]	m/s
Turn radius	signed short	2 bytes	100	[-320,320]	m
Translational acceleration	signed short	2 bytes	100	[0, 320]	m/s^2

Translational velocity: The current desired translational velocity of the vehicle.
Turn radius: The current desired turn radius of the vehicle. A positive value corresponds to a left turn.
Translational acceleration: The current desired magnitude of the translational acceleration of the vehicle

Ackermann Control Constants

Description: The control constants for the speed and heading control of an Ackermann-steered platform
Message Type: 0x8206
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Speed P	signed short	2 bytes	100	[-320,320]	-
Speed I	signed short	2 bytes	100	[-320,320]	-
Speed D	signed short	2 bytes	100	[-320,320]	-

Speed feed-forward	signed short	2 bytes	100	[-320,320]	-
Speed stiction compensation	signed short	2 bytes	100	[0,100]	%
Speed integral limit	signed short	2 bytes	100	[0,100]	%
Heading P	signed short	2 bytes	100	[-320,320]	-
Heading I	signed short	2 bytes	100	[-320,320]	-
Heading D	signed short	2 bytes	100	[-320,320]	-
Heading feed-forward	signed short	2 bytes	100	[-320,320]	-
Heading stiction compensation	signed short	2 bytes	100	[0,100]	%
Heading integral limit	signed short	2 bytes	100	[0,100]	%

Speed/heading P: The proportional constant for the respective control loop
Speed/heading I: The integral constant for the respective control loop
Speed/heading D: The derivative constant for the respective control loop
Speed/heading feed-forward: The feed-forward constant for the respective control loop
Speed/heading stiction compensation: An offset (in % of motor voltage) for the respective control output.
Speed/heading integral limit: A limit (in % of motor voltage) to the amount that the integral term can contribute to the output of the respective controller.

Max Speed Data

Description: The maximum translational speed for the platform.
Message Type: 0x8210
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Max forward speed	signed short	2 bytes	100	[0,320]	m/s
Max reverse speed	signed short	2 bytes	100	[0,320]	m/s

Max forward speed: The maximum forward translational speed
Max reverse speed: The maximum reverse translational speed

Max Accel Data

Description: The maximum translational acceleration for the platform.
Message Type: 0x8211
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Max forward accel	signed short	2 bytes	100	[0,320]	m/s
Max reverse accel	signed short	2 bytes	100	[0,320]	m/s

Max forward accel: The maximum forward translational acceleration
Max reverse accel: The maximum reverse translational acceleration

Gear Setpoint Data

Description: The current desired gear.
Message Type: 0x8212
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Flags	byte	1 byte	-	-	-
Gear	signed char	1 byte	-	-	-

Flags:

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
-	-	-	-	-	-	UPSHIFT	DOWNSHIFT

UPSHIFT: Vehicle is currently in the process of shifting up.

DOWNSHIFT: Vehicle is currently in the process of shifting down.
Gear: The desired gear for the vehicle's transmission. 0 corresponds to neutral, -1 to parked. Gears > 0 are forward gears in order of their gear ratio. Gears < -1 are reverse gears in order of their gear ratio.

GPADC Output Setpoint Data

Description: The setpoints for each of the *n* generic analog output channels. The payload will be *2n + 1* bytes in size. The amount of analog output channels is platform dependent.
Message Type: 0x8300
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of channels	byte	1 byte	-	-	-
Channel value 1	unsigned short	2 bytes	-	-	-
Continue for the remaining channels					
Channel value *n*	unsigned short	2 bytes	-	-	-

# of channels: The amount of channel setpoint values being returned.
Channel value 1: The setpoint of channel #1, scaled such that the maximum (0xFFFF) corresponds to the maximum voltage that channel can output.
Channel value n: The setpoint of the last channel, scaled such that the maximum (0xFFFF) corresponds to the maximum voltage that channel can output.

GPIO Data

Description: The value of the generic digital channels.
Message Type: 0x8301
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Direction	unsigned int	4 bytes	-	-	-
Value	unsigned int	4 bytes	-	-	-

Direction: If a bit is high, the channel is currently an output. If low, is an input.
Value: 1 for HIGH output, 0 for LOW.

GPADC Input Data

Description: The value of the *n* generic analog input channels. The payload will be *2n + 1* bytes in size. The amount of analog input channels is platform dependent.
Message Type: 0x8303
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of channels	byte	1 byte	-	-	-
Channel value 1	unsigned short	2 bytes	-	-	-
Continue for the remaining channels					
Channel value *n*	unsigned short	2 bytes	-	-	-

# of channels: The amount of channel values being returned. This number is platform-dependent.
Channel value 1: The value of channel #1, scaled such that the maximum (0xFFFF) corresponds to that channel's reference voltage.
Channel value n: The value of the last channel, scaled such that the maximum (0xFFFF) corresponds to that channel's reference voltage.

Pan/Tilt/Zoom Position Data

Description: The pan, tilt, and zoom settings on a specified camera mount. If mount does not have a pan, tilt, or zoom axis, 0 will be returned. Zero position and direction of rotation for the pan and tilt axes are dependent on the camera mount.
Message Type: 0x8400
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Mount	byte	1 byte	-	-	-
Pan	signed short	2 bytes	100	[-180,180]	deg
Tilt	signed short	2 bytes	100	[-180, 180]	deg
Zoom	signed short	2 bytes	100	[1, 320]	-

Mount: The camera mount corresponding to the data.
Pan: The current pan position of the camera mount
Tilt: The current tilt position of the camera mount
Zoom: The current zoom level of the camera

Rangefinder Data

Description: The range values for the *n* equipped rangefinder. The position and type of each rangefinder is platform-dependent. The payload will be *2n + 1* bytes in size.
Message Type: 0x8500
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of distance measurements	byte	1 byte	-	-	-
Distance measurement 1	signed short	*2* bytes	1000	[0, 32]	m
Continue for the remaining sensors					
Distance measurement *n*	signed short	*2* bytes	1000	[0, 32]	m

# of distance measurements: The number of distance measurements being returned.
Distance measurement 1: The distance value from the first distance sensor, in meters.
Distance measurement n: The distance value from the last distance sensor, in meters.

Rangefinder Data & Timing

Description: The range values for the *n* equipped rangefinders, and the time at which each was last updated. The position and type of each rangefinder is platform-dependent. The payload will be 6*n + 1* bytes in size.
Message Type: 0x8501
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of distance measurements	byte	1 byte	-	-	-
Distance measurement 1	signed short	*2* bytes	1000	[0, 32]	m
Continue for the remaining sensors					
Distance measurement *n*	signed short	*2* bytes	1000	[0, 32]	m
Acquisition time 1	unsigned int	4 bytes	-	-	ms
Continue for the remaining sensors					
Acquisition time *n*	unsigned int	4 bytes	-	-	ms

# of distance measurements: The number of distance measurements being returned.
Distance measurement 1: The distance value from the first distance sensor, in meters.
Distance measurement n: The distance value from the last distance sensor, in meters.
Acquisition time 1: 4 bytes representing the time the first sensor acquired its data, according to the platform clock.
Acquisition time n: 4 bytes representing the time the last sensor acquired its data, according to the platform clock.

Platform Orientation Data

Description: The vehicle's best estimate of its orientation.
Message Type: 0x8600
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Roll	signed short	2 bytes	1000	[-π, π]	rad
Pitch	signed short	2 bytes	1000	[-π, π]	rad
Yaw	signed short	2 bytes	1000	[-π, π]	rad

Roll: Vehicle's angle about its roll axis, in a right-hand sense.
Pitch: Vehicle's angle about its pitch axis, in a right-hand sense.
Yaw: Vehicle's global heading with respect to magnetic north, about its yaw axis, in a left-hand sense. Note: This means that the vehicle's yaw increases as the vehicle turns towards the east (based on the traditional compass NED frame). If a vehicle is only equipped with tilt sensors, this field will default to 0.

Platform Rotational Rate Data

Description: The vehicle's best estimate of its various rotational rates.
Message Type: 0x8601
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Roll rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s
Pitch rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s
Yaw rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s

Roll rate: Vehicle's angular rate about its roll axis, in a right-hand sense.
Pitch rate: Vehicle's angular rate about its pitch axis, in a right-hand sense.
Yaw rate: Vehicle's angular rate about its yaw axis, in a left-hand sense. Note: Positive yaw rate corresponds to a right turn (based on the traditional compass NED frame).

Platform Acceleration Data

Description: The vehicle's best estimate of its translational acceleration.
Message Type: 0x8602
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
X	signed short	2 bytes	1000	[-32, 32]	m/s^2
Y	signed short	2 bytes	1000	[-32, 32]	m/s^2
Z	signed short	2 bytes	1000	[-32, 32]	m/s^2

X: Vehicle's acceleration along its x axis
Y: Vehicle's acceleration along its y axis
Z: Vehicle's acceleration along its z axis

Platform 6-Axis Data

Description: The vehicle's best estimates of its translational acceleration and rotational rates.
Message Type: 0x8603
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
X	signed short	2 bytes	1000	[-32, 32]	m/s^2
Y	signed short	2 bytes	1000	[-32, 32]	m/s^2
Z	signed short	2 bytes	1000	[-32, 32]	m/s^2
Roll rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s
Pitch rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s

Yaw rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s

X: Vehicle's acceleration along its x axis
Y: Vehicle's acceleration along its y axis
Z: Vehicle's acceleration along its z axis
Roll rate: Vehicle's angular rate about its roll axis, in a right-hand sense.
Pitch rate: Vehicle's angular rate about its pitch axis, in a right-hand sense.
Yaw rate: Vehicle's angular rate about its yaw axis, in a right-hand sense. Note: Positive yaw rate corresponds to a left turn. Compasses traditionally have heading increasing as one turns right.

Platform 6-Axis & Orientation Data

Description: The vehicle's best estimates of its translational acceleration and rotational rates.
Message Type: 0x8604
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Roll	signed short	2 bytes	1000	[-π, π]	rad
Pitch	signed short	2 bytes	1000	[-π, π]	rad
Yaw	signed short	2 bytes	1000	[-π, π]	rad
X	signed short	2 bytes	1000	[-32, 32]	m/s^2
Y	signed short	2 bytes	1000	[-32, 32]	m/s^2
Z	signed short	2 bytes	1000	[-32, 32]	m/s^2
Roll rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s
Pitch rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s
Yaw rate	signed short	2 bytes	1000	[-10π, 10π]	rad/s

Roll: Vehicle's angle about its roll axis, in a right-hand sense.
Pitch: Vehicle's angle about its pitch axis, in a right-hand sense.
Yaw: Vehicle's global heading with respect to magnetic north, about its yaw axis, in a right-hand sense. Note: This means that the vehicle's yaw increases as the vehicle turns towards the west. Compasses traditionally have heading increasing as one turns east. If a vehicle is only equipped with tilt sensors, this field will default to 0.
X: Vehicle's acceleration along its x axis
Y: Vehicle's acceleration along its y axis
Z: Vehicle's acceleration along its z axis
Roll rate: Vehicle's angular rate about its roll axis, in a right-hand sense.
Pitch rate: Vehicle's angular rate about its pitch axis, in a right-hand sense.
Yaw rate: Vehicle's angular rate about its yaw axis, in a right-hand sense. Note: Positive yaw rate corresponds to a left turn. Compasses traditionally have heading increasing as one turns right.

Platform Magnetometer Data

Description: The vehicle's best estimate of the surrounding magnetic fields.
Message Type: 0x8606
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
X	signed short	2 bytes	1000	[-32, 32]	G
Y	signed short	2 bytes	1000	[-32, 32]	G
Z	signed short	2 bytes	1000	[-32, 32]	G

X: Vehicle's magnetic fields on its x axis
Y: Vehicle's magnetic fields on its y axis
Z: Vehicle's magnetic fields on its z axis

Encoder Data

Description: Data from the vehicle's *n* encoders. The payload will be *6n + 1* bytes in size.
Message Type: 0x8800

Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of encoders	byte	1 byte	-	-	-
Travel of encoder 1	signed int	4 bytes	1000	[-2x10^6, 2x10^6]	m
Continue for the remaining encoders					
Travel of encoder *n*	signed int	4 bytes	1000	[-2x10^6, 2x10^6]	m
Speed of encoder 1	signed short	2 bytes	1000	[-32,32]	m/s
Continue for the remaining encoders					
Speed of encoder *n*	signed short	2 bytes	1000	[-32,32]	m/s

# of encoders: The number of encoder measurements being returned. The position of each encoder is platform-dependent.
Travel of encoder 1: The distance encoder #1 will have driven since startup under ideal non-slip conditions.
Travel of encoder n: The distance the last encoder will have driven since startup under ideal non-slip conditions.
Speed of encoder 1: The current instantaneous speed of the first encoder.
Speed of encoder n: The current instantaneous speed of the last encoder.

Raw Encoder Data

Description: Raw data from the vehicle's *n* encoders. The payload will be *4n + 1* bytes in size.
Message Type: 0x8801
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of encoders	byte	1 byte	-	-	-
Encoder 1 ticks	signed int	4 bytes	1	[-2^{31}, 2^{31}-1]	-
Continue for the remaining encoders					
Encoder *n* ticks	signed int	4 bytes	1	[-2^{31}, 2^{31}-1]	-

# of encoders: The number of encoder measurements being returned. The position of each encoder is platform-dependent.
Encoder 1 ticks: The amount of ticks accumulated by the first encoder. Forward travel increments this value, backwards travel decrements this value.
Encoder n ticks: The amount of ticks accumulated by the last encoder. Forward travel increments this value, backwards travel decrements this value.

Encoder Config Data

Description: The PPR and scale factors for the platform's *n* encoders, as preset by the factory or overridden via **0x0802 – "Configure Encoders".** The payload will be *4n + 1* bytes in size.
Message Type: 0x8802
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of encoders	byte	1 byte	-	-	-
PPR of encoder 1	signed short	2 bytes	1	[0, 32x10^3]	PPR
Scale factor of encoder 1	signed short	2 bytes	1000	[-32,32]	m/rev *or* rev/rev
Continue for the remaining encoders					
PPR of encoder *n*	signed short	2 bytes	1	[0, 32x10^3]	PPR
Scale factor of encoder *n*	signed short	2 bytes	1000	[-32,32]	m *or* rev

# of encoders: The number of encoder measurements being returned. The position of each encoder is platform-dependent.
PPR of encoder 1: The amount of pulses per revolution of the first encoder.
Scale factor of encoder 1: The amount the final output moves per revolution of the encoder (units dependent on output type).

PPR of encoder n: The amount of total pulses per revolution of the last encoder***.***
Scale factor of encoder n: The amount the final output moves per revolution of the encoder (units dependent on output type).

Absolute Joint Position Setpoint Data

Description: The current desired absolute joint positions. The payload will be *2n + 1* bytes in size, where *n* is the amount of joints on the manipulator.
Message Type: 0x9010
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of joints	byte	1 byte	-	-	-
Joint angle 1	signed short	2 bytes	10000	[-π, π]	rad
Continue for the remaining joints					
Joint angle *n*	signed short	2 bytes	10000	[-π, π]	rad

 # joints: The amount of joint angles being returned. The position of each joint is manipulator-dependent.
Joint angle 1: The angle the first joint is attempting to attain, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware.
Joint angle n: The angle the last joint is attempting to attain, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware.

Relative Joint Position Setpoint Data

Description: The current desired relative joint positions. As these positions are relative to the angle the joints were at when the relative motion command was received, this information is not necessarily helpful after the movement has started. The payload will be *2n + 1* bytes in size, where *n* is the amount of joints on the manipulator.
Message Type: 0x9011
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of joints	byte	1 byte	-	-	-
Joint angle 1	signed short	2 bytes	10000	[-π, π]	rad
Continue for the remaining joints					
Joint angle *n*	signed short	2 bytes	10000	[-π, π]	rad

 # joints: The amount of joint angles being returned. The position of each joint is manipulator-dependent.
Joint angle 1: The relative angle the first joint is attempting to attain, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware.
Joint angle n: The relative angle the last joint is attempting to attain, in radians. The definition of which direction is a positive turn is dependent on the specific manipulator hardware.

Joint Control Constant Data

Description: The control constants for the specified joint on a manipulator. Input into each controller is the position error (in radians), and the output is the torque the corresponding motor's should exert.
Message Type: 0x9012
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
Joint ID	byte	byte	-	-	-
P	signed short	2 bytes	100	[-320,320]	-
I	signed short	2 bytes	100	[-320,320]	-
D	signed short	2 bytes	100	[-320,320]	-
Feed-forward	signed short	2 bytes	100	[-320,320]	-
Stiction compensation	signed short	2 bytes	100	[0,100]	N-m

Integral limit	signed short	2 bytes	100	[0,100]	N-m

Joint ID: The joint the data corresponds to. The position of each joint is manipulator-dependent..
P: The proportional constant for the control loop
I: The integral constant for the control loop
D: The derivative constant for the control loop
Feed-forward: The feed-forward constant for the respective side's control loop
Stiction compensation: An offset (in N-m) for the respective side's control output used to compensate for static friction.
Integral limit: A limit (in N-m) to the amount that the integral term can contribute to the output of the controller.

Joint Homing Status

Description: The homing status for each joint – if they have been homed and if they have been moved from this position. The payload will be *n + 1* bytes in size, where *n* is the amount of joints on the manipulator.
Message Type: 0x9013
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of joints	byte	byte	-	-	-
Joint 1 status	byte	byte	-	-	-
Continue for the remaining joints					
Joint *n* status	byte	byte	-	-	-

# of joints: The amount of joints. The position of each joint is manipulator-dependent.
Joint 1 status: The homing status for joint 1 (defined below).
Joint n status: The homing status for joint *n* (defined below).
Homing Status Values:
0x00: Unhomed
0x01: Homed, still at its home position
0x02: Homed, moved away from its home position

Joint Torque Data

Description: The torque being experienced by each joint. The payload will be *2n + 1* bytes in size, where *n* is the amount of integrated torque sensors.
Message Type: 0x9014
Message Fields:

Field	*Type*	*Size*	*Scale*	*Range*	*Units*
# of torque measurements	byte	1 byte	-	-	-
Torque 1	signed short	2 bytes	100	[-320,320]	N-m
Continue for the remaining joints					
Torque *n*	signed short	2 bytes	100	[-320,320]	N-m

# of torque measurements: The number of torque measurements being returned. The position of each torque sensor is manipulator-dependent.
Torque 1: The torque the first joint is experiencing. The direction of the torque is defined by the specific manipulator hardware.
Torque n: The torque the last joint is experiencing. The direction of the torque is defined by the specific manipulator hardware.

End Effector Position Setpoint Data

Description: The desired end effector position in the robot frame. The robot frame definition is dependent on the specific manipulator hardware.
Message Type: 0x9020
Message Fields:

Field	Type	Size	Scale	Range	Units
x	signed short	2 bytes	1000	[-32,32]	m
y	signed short	2 bytes	1000	[-32,32]	m
z	signed short	2 bytes	1000	[-32,32]	m

x: The desired x position in the robot frame of the end effector origin
y: The desired y position in the robot frame of the end effector origin
z: The desired z position in the robot frame of the end effector origin

End Effector Pose Setpoint Data

Description: The desired end effector position and orientation in the robot frame. The robot frame definition is dependent on the specific manipulator hardware.
Message Type: 0x9021
Message Fields:

Field	Type	Size	Scale	Range	Units
x	signed short	2 bytes	1000	[-32,32]	m
y	signed short	2 bytes	1000	[-32,32]	m
z	signed short	2 bytes	1000	[-32,32]	m
Roll	signed short	2 bytes	1000	[-π, π]	rad
Pitch	signed short	2 bytes	1000	[-π, π]	rad
Yaw	signed short	2 bytes	1000	[-π, π]	rad

x: The desired x position in the robot frame of the end effector origin
y: The desired y position in the robot frame of the end effector origin
z: The desired z position in the robot frame of the end effector origin
Roll: The desired roll of the end effector in the robot frame
Pitch: The desired pitch of the end effector in the robot frame
Yaw: The desired yaw of the end effector in the robot frame

End Effector Pose Data

Description: The current end effector position and orientation in the robot frame. The robot frame definition is dependent on the specific manipulator hardware.
Message Type: 0x9021
Message Fields:

Field	Type	Size	Scale	Range	Units
x	signed short	2 bytes	1000	[-32,32]	m
y	signed short	2 bytes	1000	[-32,32]	m
z	signed short	2 bytes	1000	[-32,32]	m
Roll	signed short	2 bytes	1000	[-π, π]	rad
Pitch	signed short	2 bytes	1000	[-π, π]	rad
Yaw	signed short	2 bytes	1000	[-π, π]	rad

x: The current x position in the robot frame of the end effector origin
y: The current y position in the robot frame of the end effector origin
z: The current z position in the robot frame of the end effector origin
Roll: The current roll of the end effector in the robot frame
Pitch: The current pitch of the end effector in the robot frame
Yaw: The current yaw of the end effector in the robot frame

Appendix A – CRC Generation

A 16-bit CRC using one of the CCITT polynomials is used to confirm message integrity.

Polynomial: x16+x12+x5+1 (0x1021)
Initial value: 0xFFFF
Check constant: 0x1D0F

The calculated CRC of the string '123456789' should be 0x29B1

To confirm CRC implementation, the following process can be used:

1) Calculate the CRC of any message
2) XOR it with 0xFFFF (bitwise inversion)
3) Append it to the original message
4) Perform the CRC calculation on the extended message
5) Confirm that the new CRC is equal to the check constant (0x1D0F)

Sample C Code for table-driven CRC computation:

```
/*
 *  crc.h
 */

#ifndef __CRC16_H
#define __CRC16_H

/***----------Table-driven crc function----------***/
/*Inputs: -size of the character array, the CRC of which is being computed   */
/*        - the initial value of the register to be used in the calculation  */
/*        - a pointer to the first element of said character array           */
/*Outputs: the crc as an unsigned short int                                  */
unsigned short int crc16(int size, int init_val, char *data);

#endif

/*
 *  crc.c
 */

#include "crc.h"

//CRC lookup table for polynomial 0x1021
const unsigned short int table[256] =
{0, 4129, 8258, 12387, 16516, 20645, 24774, 28903, 33032, 37161, 41290, 45419, 49548,
53677, 57806, 61935, 4657, 528, 12915, 8786, 21173, 17044, 29431, 25302, 37689, 33560,
45947, 41818, 54205, 50076, 62463, 58334, 9314, 13379, 1056, 5121, 25830, 29895, 17572,
21637, 42346, 46411, 34088, 38153, 58862, 62927, 50604, 54669, 13907, 9842, 5649, 1584,
30423, 26358, 22165, 18100, 46939, 42874, 38681, 34616, 63455, 59390, 55197, 51132,
18628, 22757, 26758, 30887, 2112, 6241, 10242, 14371, 51660, 55789, 59790, 63919,
35144, 39273, 43274, 47403, 23285, 19156, 31415, 27286, 6769, 2640, 14899, 10770,
56317, 52188, 64447, 60318, 39801, 35672, 47931, 43802, 27814, 31879, 19684, 23749,
11298, 15363, 3168, 7233, 60846, 64911, 52716, 56781, 44330, 48395, 36200, 40265,
32407, 28342, 24277, 20212, 15891, 11826, 7761, 3696, 65439, 61374, 57309, 53244,
48923, 44858, 40793, 36728, 37256, 33193, 45514, 41451, 53516, 49453, 61774, 57711,
```

```
4224, 161, 12482, 8419, 20484, 16421, 28742, 24679, 33721, 37784, 41979, 46042, 49981,
54044, 58239, 62302, 689, 4752, 8947, 13010, 16949, 21012, 25207, 29270, 46570, 42443,
38312, 34185, 62830, 58703, 54572, 50445, 13538, 9411, 5280, 1153, 29798, 25671, 21540,
17413, 42971, 47098, 34713, 38840, 59231, 63358, 50973, 55100, 9939, 14066, 1681, 5808,
26199, 30326, 17941, 22068, 55628, 51565, 63758, 59695, 39368, 35305, 47498, 43435, 22596,
18533, 30726, 26663, 6336, 2273, 14466, 10403, 52093, 56156, 60223, 64286, 35833, 39896,
43963, 48026, 19061, 23124, 27191, 31254, 2801, 6864, 10931, 14994, 64814, 60687, 56684,
52557, 48554, 44427, 40424, 36297, 31782, 27655, 23652, 19525, 15522, 11395, 7392, 3265,
61215, 65342, 53085, 57212, 44955, 49082, 36825, 40952, 28183, 32310, 20053, 24180, 11923,
16050, 3793, 7920};


/***----------Table-driven crc function----------***/
/*Inputs: -size of the character array, the CRC of which is being computed   */
/*        - the initial value of the register to be used in the calculation  */
/*        - a pointer to the first element of said character array */
/*Outputs: the crc as an unsigned short int    */
unsigned short int crc16(int size, int init_val, char *data)
{
          unsigned short int crc = (unsigned short int) init_val;
          while(size--) {
                    crc = (crc << 8) ^ table[((crc >> 8) ^ *data++) & 0xFFFF];
          }
          return crc;
}
```