1 Introduction

1.1 Overview

The creation of the RobotOpen Protocol is an effort to design an open, standardized protocol for communication and control of hobbyist robots over an IP based network. It is designed for two-way communication and allows any arbitrary values to be bundled and transmitted.

2 Protocol Structure

2.1 Data Bundles

Data bundles are the primary method of grouping data together and transmitting it over the RobotOpen protocol. The payload of each packet is nothing more than each data bundle one after another. Data bundles are identified by a single character or ASCII byte. This is typically 0-9, a-z, or A-Z. By default, the Driver Station sends each joystick as its own data bundle. The first joystick has the bundle ID of '0', the second is '1' and so on. Each byte in these bundles corresponds to a joystick component (analog axis, button, etc...).

2.2 Networking

RobotOpen packets are typically transmitted as UDP over IP based networks, however there are no limitations on this. The RobotOpen Arduino library implements a UDP server listening on port 22211. The driver station or UDP client that wishes to connect to the robot sends RobotOpen packets to this port. The RobotOpen Arduino library will send all response packets to the high level port that is automatically assigned when the client connects to the server. Note that only one device or driver station should send packets to a robot controller at a time. This is not always enforced, so ensure that you have properly secured the network that your robot controller is connected to.

2.3 Required Fields and Length

Every RobotOpen packet must be at least 5 bytes long. This includes the message type, protocol version, device ID, and 16 bit CRC checksum. In this initial release, RobotOpen packets may not be longer than 256 bytes. This may change in future protocol revisions.

2.4 Checksumming

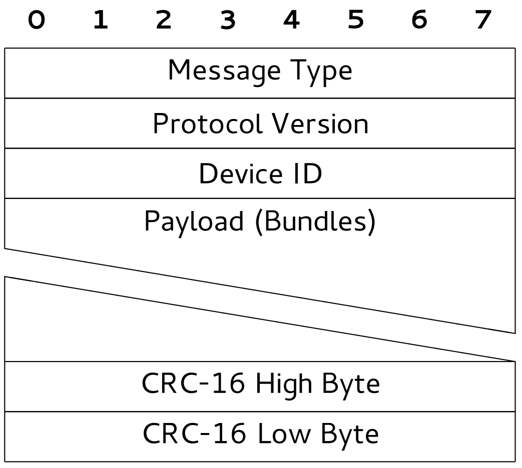
Because every RobotOpen packet has a 16 bit CRC, it is safe to assume that the data transmitted will not become corrupted in transit. Keep in mind that this does not guarantee the ordering of packets or even if the packet will ever be delivered.

3 Packet Types

As stated earlier, every RobotOpen packet is required to have a packet type associated with it. This is always the first byte of each packet. There are currently 3 packet types, each listed below. Packet structures have been provided. The width of each box is assumed to be 8 bits or 1 byte.

3.1 Control Packet

Control packets are used for sending control data to a robot from a driver station. If a robot controller does not receive a control packet within 250ms of the last, it should automatically disable itself.



Message Type: Always 0x01 for control packets.

Protocol Version: The version of RobotOpen protocol. This documentation is for version 2. This is transmitted as 0x02. If a controller detects a version mismatch it should throw out the packet.

Device ID: Used to identify what device is sending control packets (Driver Station app, Android, iPhone, etc...). See the Device ID section at the end of this document.

Payload: The actual control data to be sent. This will be in the form of bundles. See the Payload section later in this document. This portion of the packet is variable length.

CRC–16 High/Low Byte: A CRC-16 checksum of the entire packet, excluding these CRC-16 bytes. This 16 bit value should be transmitted as 2 bytes. The first byte in the packet is always the high byte, followed by the low byte. If a packet becomes corrupt it should be thrown out.

3.2 Query Packet

Query packets are sent from a driver station to a robot controller to request a feedback packet. Think of this packet type like a control packet with no payload. No control data is being sent but we'd like to receive data from the robot and notify it of our IP and port.

0 1 2 3 4 5 6 7
Message Type
Protocol Version
Device ID
CRC-16 High Byte
CRC-16 Low Byte

Message Type: Always 0x03 for query packets.

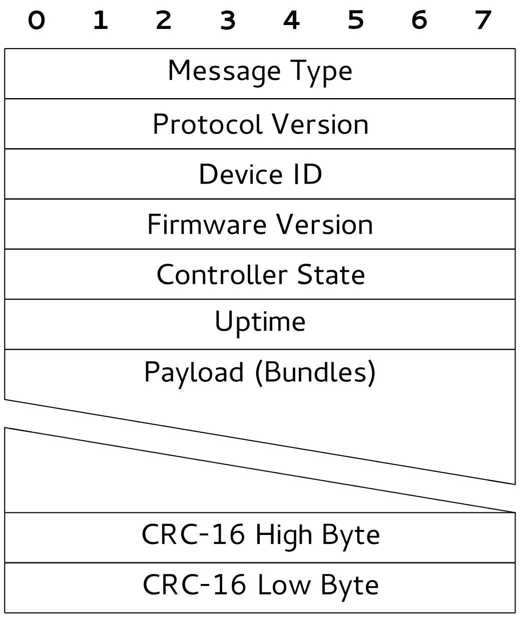
Protocol Version: The version of RobotOpen protocol. This documentation is for version 2. This is transmitted as 0x02. If a controller detects a version mismatch it should throw out the packet.

Device ID: Used to identify what device is sending query packets (Driver Station app, Android, iPhone, etc...). See the Device ID section at the end of this document.

CRC–16 High/Low Byte: A CRC-16 checksum of the entire packet, excluding these CRC-16 bytes. This 16 bit value should be transmitted as 2 bytes. The first byte in the packet is always the high byte, followed by the low byte. If a packet becomes corrupt it should be thrown out.

3.3 Feedback Packet

Feedback packets are sent from a robot controller to a driver station. They contain robot state information and any data that should be sent back for display or processing on the DS. When a driver station sends a packet to a robot controller it should save the IP and port that sent the packet. All feedback packets are then returned to this IP/port. For every control and query packet, a feedback packet should be sent back.



Message Type: Always 0x02 for feedback packets.

Protocol Version: The version of RobotOpen protocol. This documentation is for version 2. This is transmitted as 0x02. If a controller detects a version mismatch it should throw out the packet.

Device ID: Used to identify what device is sending the feedback packet (Arduino, Computer, etc...). See the Device ID section at the end of this document.

Firmware Version: Used for identifying the version of your control code loaded onto a robot. This will typically be user provided. Values can range from 0x00 to 0xFF.

Controller State: The state of your robot (enabled or disabled). If the robot is disabled it should return 0x00. If it is enabled it should return 0xFF.

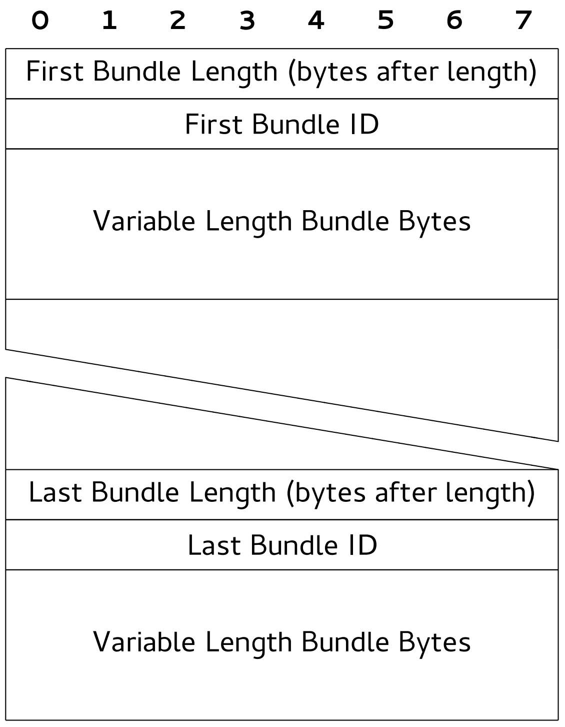
Uptime: The time (in minutes) that the controller has been powered on for. Values can range from 0x00 to 0xFF. After 255 minutes, the controller can either keep this value at 255 or loop around. The Arduino library currently caps this at 255.

Payload: Any additional feedback data to be sent. This will be in the form of bundles. See the Payload section later in this document. This portion of the packet is variable length.

CRC–16 High/Low Byte: A CRC-16 checksum of the entire packet, excluding these CRC-16 bytes. This 16 bit value should be transmitted as 2 bytes. The first byte in the packet is always the high byte, followed by the low byte. If a packet becomes corrupt it should be thrown out.

4 Payload

The payload of a packet is nothing more than each data bundle one after another. Below is an example of how this data is structured. Data bundles are open ended and can used in any way you choose. Typically each joystick is sent as its own bundle in a control packet and each sensor or reading is sent back as a bundle in a feedback packet. When using the publish functions in the Arduino library, every value passed is placed into its own bundle.


0 1 2 3 4 5 6 7
First Bundle Length (bytes after length)
First Bundle ID
Variable Length Bundle Bytes
Last Bundle Length (bytes after length)
Last Bundle ID
Variable Length Bundle Bytes

Bundle Length: This byte specifies the length of the bundle excluding the length byte. A bundle that has 2 bytes of payload will have a length of 0x03 due to the bundle ID byte.

Bundle ID: A single ASCII character that identifies this bundle. Although this can be any value between 0x00 and 0xFF, it is encouraged to use printable characters such as 0-9, a-z, or A-Z.

Variable Length Bundle Bytes: The actual data. This can be in whatever format is most convenient to you. For joysticks, each byte is a component value (y axis, dpad, button), however the use of this is open ended. ASCII strings could even be sent in a bundle using this portion.

As seen in the diagram, each bundle follows one after another. Both the driver station and robot controller will parse out each of these bundles individually.

5 Joystick Bundles

5.1 Joystick (Fixed Format) Payload

Version 2 of the RobotOpen protocol introduced a fixed format for all joystick devices. This was an effort to standardize the control scheme across the Android, iOS, and desktop driver stations. Starting with version 2 of the protocol, all RobotOpen controllers will expect control data in this format.

0 1 2 3 4 5 6 7
First Bundle Length - Always 0x12
First Bundle ID - 0x30 for Joystick 1
Analog Left X Axis (0x00-0xFF)
Analog Left Y Axis (0x00-0xFF)
Analog Right X Axis (0x00-0xFF)
Analog Right Y Axis (0x00-0xFF)
Analog Left Button (0x00 or 0xFF)
Analog Right Button (0x00 or 0xFF)
D-Pad (0x00-0xFF)
Button 1 (0x00 or 0xFF)
Button 2 (0x00 or 0xFF)
Button 3 (0x00 or 0xFF)
Button 4 (0x00 or 0xFF)
Button 5 (0x00 or 0xFF)
Button 6 (0x00 or 0xFF)
Button 7 (0x00 or 0xFF)
Button 8 (0x00 or 0xFF)
Button 9 (0x00 or 0xFF)
Button 10 (0x00 or 0xFF)
More joystick data...

5.2 D-Pad Values

0x3F	Up
0x1F	Up Left
0x5F	Up Right
0xBF	Down
0xDF	Down Left
0x9F	Down Right
0xFF	Left
0x7F	Right

5.3 Multiple Joysticks

Note that more joystick data can be added to the packet by simply repeating the same pattern for the next joystick. The next byte would be the bundle length (again - 0x12), followed by the bundle ID. This follows the ASCII format, giving joystick 2 a bundle ID of 0x31.

6 Device IDs

The following device IDs have been assigned thus far. Note that driver stations begin at 0x01 counting up and robot controllers begin at 0xFF counting down.

- OFFICIAL_JAVA_DS - 0x01
- OFFICIAL_ANDROID_APP - 0x02
- OFFICIAL_IPHONE_APP - 0x03
- ROBOTOPEN_V1 - 0xFF