redrockmicro

ECLIPSE

API

ECLIPSE API

Version 1.0

Revision F | 2.20.2020

Overview

The ECLIPSE API is a USB protocol interface that allows query, interaction, and control of compatible Redrock Eclipse products.

Redrock Eclipse products natively use the CAN bus to communicate and coordinate between each other. The Eclipse API enables third party app and product developers access to the CAN bus to interact with Redrock products to enhance or extend their own applications.

Redrock Eclipse products currently compatible with the Eclipse API include:

- ATLAS lens motors
- NAVIGATOR 7-in-1 controller
- SATELLITE (V2 and later)
- COMMANDER Handheld FIZ controller

Eclipse API Example Applications

The Eclipse API offers unprecedented flexibility to add lens controls and UI components to custom and commercial applications. Examples of these applications include:

- Lens controls (focus, iris, and/or zoom) for robotic arms and motion control (moco), including keyframing and interactive setups
- Lens controls for custom applications such as light field cameras
- UI controls for Real-time production performance capture (e.g., capturing focus pulls and camera control for realistic camera playback)
- UI controls for game engines such as Unity and Unreal (including virtual camera controls)

Advantages of Eclipse API

Eclipse products communicate using state-of-the-art CAN bus, the same core technology that is used in autonomous cars and real-time sensing applications. Eclipse API uses a broadcast messaging style Instead of point-to-point or round robin approaches to maximize performance and flexibility. A single message format contains everything needed to query, interact with, and control Redrock products on the CAN bus.

Advantages of the Eclipse API and communications framework include:

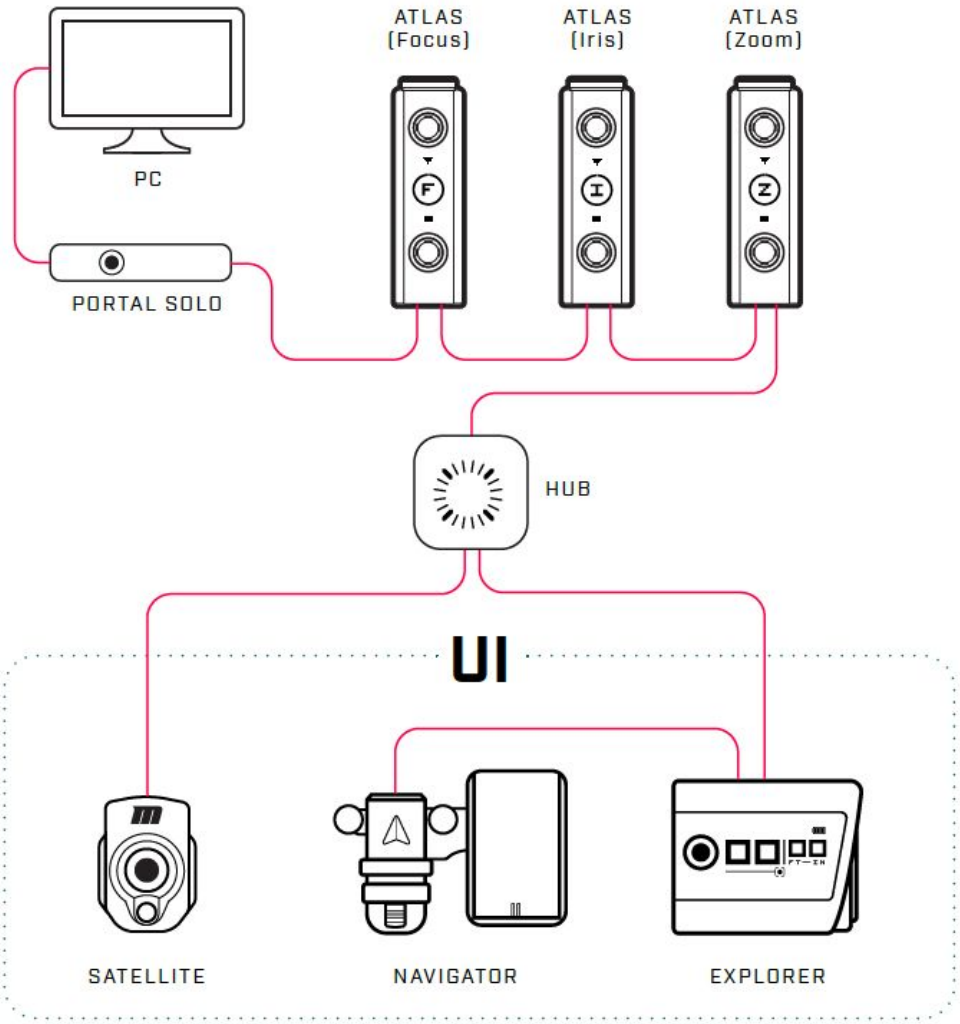
- High performance, real-time environment that is faster and more reliable.
- Easy to integrate with using simple message formats in a broadcast style messaging, and can be embedded with other motion such as robotic arm.
- Technology designed from the ground up for robotics and remote-use applications
- Modern technology - meet your needs now, with room to grow using additional hardware and advancements that can be simply plugged in to the CAN bus
- Features and hardware to make setup and keyframing easy
- Extensible - easy to add additional motors or controls just by plugging them in

ECLIPSE Setup Overview

Eclipse CAN bus uses a high speed broadcast style network where all devices are listening for messages.

Below is an example of how components are connected:

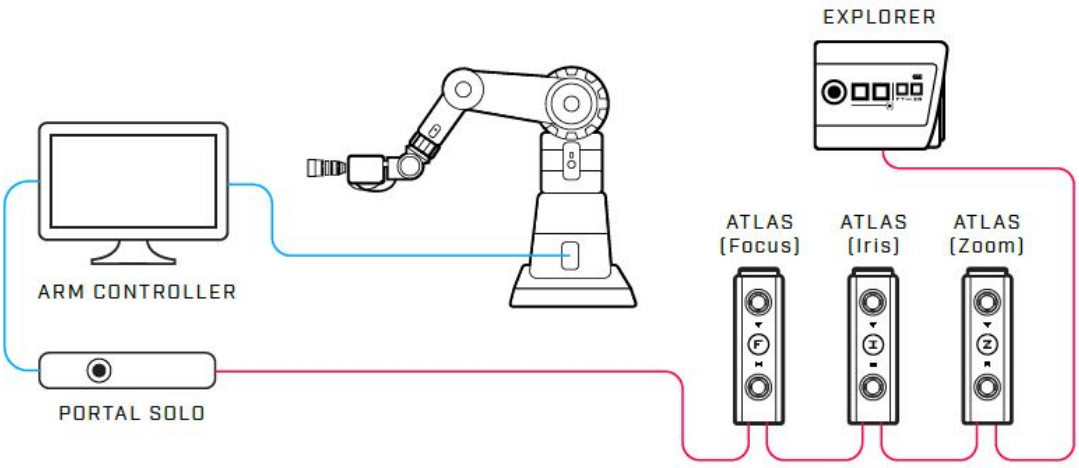
1. PC is connected via standard USB
2. USB is converted to CAN via Redrock Portal Solo
3. All Redrock products are connected via CAN bus, and share power and communications as soon as they are connected
4. UI components are also available to control or be queried
5. Advanced applications for autofocus and rangefinding are available with the Halo Explorer laser rangefinder


ATLAS
(Focus)
ATLAS
(Iris)
ATLAS
(Zoom)
PC
PORTAL SOLO
HUB
UI
SATELLITE
NAVIGATOR
EXPLORER

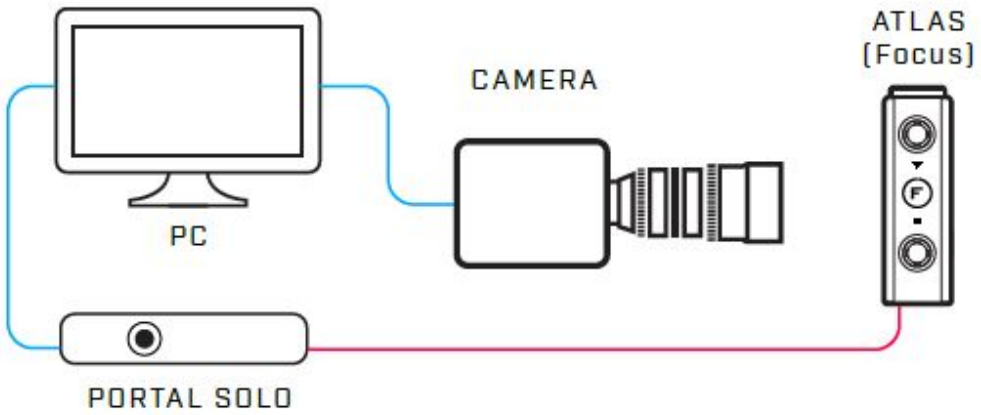
Example Applications

Eclipse API can be used in many applications to add lens and UI controls. Here are some typical examples:

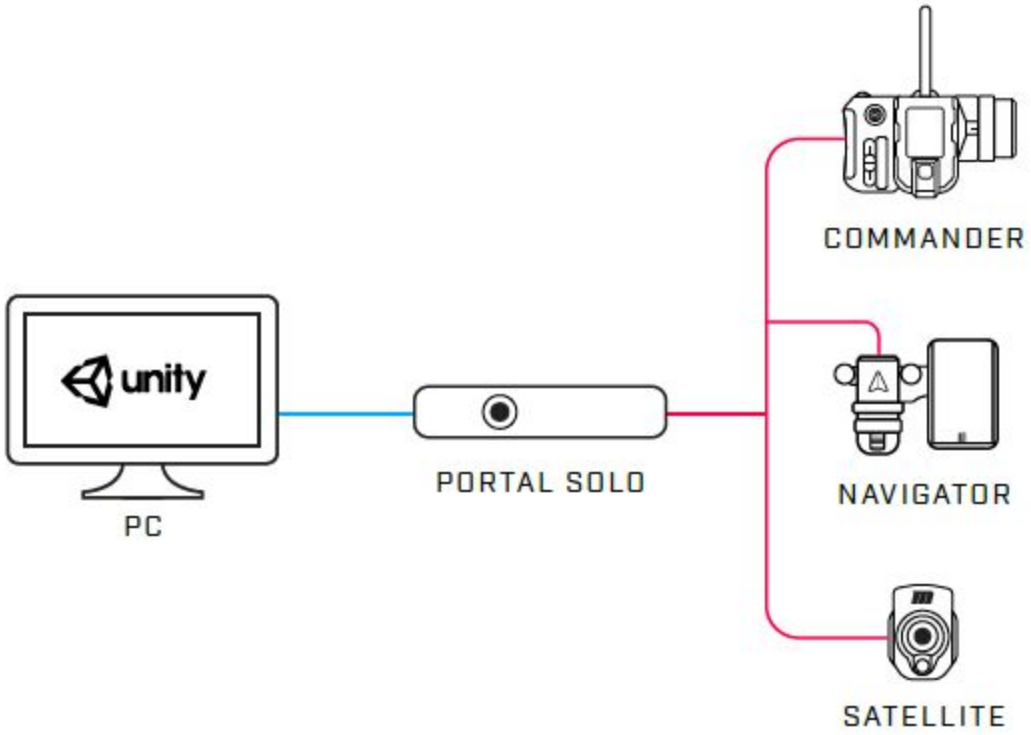
1. **Integrating lens control with Robotic arms/ Moco:** Eclipse API can be used in during setup/keyframe to interactively read and set the motor positions. During runtime the robot control or custom controller can be used to feed real-time motor position to Atlas motors.



2. **Custom camera applications** that are driven by a computer application (such as advanced 3D and light field cameras): Custom PC applications can directly drive lens control motors in real time



3. **Real-time performance capture of production crew** (such as CG movies where camera movement, lens control, etc. is all recorded for digital playback via Unreal or Unity game engines): in more advanced applications, Redrock UI controllers are used to stream real-time performance of the controller, such as pulling focus, changing zoom, adjusting camera pan/tilt, etc. these are addressable and behave nearly identical to lens control motors from an app development perspective



Connecting to Eclipse Devices

Connection to ECLIPSE Devices requires PORTAL SOLO. Portal Solo is Redrock's product for accessing the Eclipse CAN network via USB virtual comm port.

NOTE: When installing Portal Solo onto your computer, the FTDI (USB) Driver for Windows my require "Load VCP" to be enabled in the device's properties in Window's Device Manager.

ECLIPSE API v1.0 supported USB Serial Port Settings (115200 8-N-1):

Baud Rate	115200 bps
Data Bits	8
Parity	None

Stop Bits	1
Flow Control	Not used.

ECLIPSE Protocol Layout

The Eclipse API is essentially understanding and using our Eclipse universal message format.

There are two main uses of the message - Send (to set values) and Read (to receive values).

When using the Send message, that last field value determines if a receive message will be sent back. In real-time playback where status is not required, the return would be turned off to minimize network traffic and latency.

ATLAS Control (Send) Definitions

Field #	Field	Value	Length (bytes)	Range	Function
1	Header	String “RRM”	3	3	Header
2	Packet Type	String “G”	1	1	Packet Type
3	P_Length	8 bit Unsigned Integer	1	5-255	Length of final packet
4	Focus	16 bit Unsigned Integer	2	0-0xFFFF	Focus Value (Position)
5	Iris	16 bit Unsigned Integer	2	0-0xFFFF	Iris Value (Position)
6	Zoom	16 bit Unsigned Integer	2	0-0xFFFF	Zoom Value (Position)
7	AutoCal	8 bit Unsigned Integer	1	0-1	Start Auto Calibration
8	RTR_M	8 bit Unsigned Integer	1	0-2	Return Motor Packet 0=off 1=on with new position 2 = on no motor position sent

ATLAS Control (Read) Definitions

In order to receive this Field # 8 needs to be set to 1or 2 on the “Send” packet.

Field #	Field	Value	Length (Bytes)	Range	Function
1	Header	String “RRM”	3	3	Header
2	Packet Type	String “M”	1	1	Packet Type
3	P_Length	8 bit Unsigned Integer	1	5-255	Length of final packet
4	Focus	32 bit Unsigned Integer	4	0-0xFFFFFFFF	Focus Value (Position)
5	Focus CAL Status	8 bit Unsigned Integer	1	0-1	Focus Calibration Status (0 = Offline, 1 = Online, 3 = Calibrated)
6	Iris	32 bit Unsigned Integer	4	0-0xFFFFFFFF	Iris Value (Position)
7	Iris CAL Status	8 bit Unsigned Integer	1	0-1	Iris Calibration Status (0 = Offline, 1 = Online, 3 = Calibrated)
8	Zoom	32 bit Unsigned Integer	4	0-0xFFFFFFFF	Zoom Value (Position)
9	Zoom CAL Status	8 bit Unsigned Integer	1	0-1	Zoom Calibration Status (0 = Offline, 1 = Online, 3 = Calibrated)
10	Explorer Value	16 bit Unsigned Integer	2	0-0xFFFF	Distance to subject in inches

There are additional message formats for various setup and configuration tasks as well:

PORTAL SOLO Control Reboot Request

Field #	Field	Value	Length (bytes)	Range	Function
1	Header	String “RRM”	3	3	Header
2	Packet Type	String “R”	1	1	Packet Type
3	P_Length	8 bit Unsigned Integer	1	5-255	Length of final packet

PORTAL SOLO Control Status Request

Field #	Field	Value	Length (bytes)	Range	Function
1	Header	String “RRM”	3	3	Header
2	Packet Type	String “S”	1	1	Packet Type
3	P_Length	8 bit Unsigned Integer	1	5-255	Length of final packet

UI Control (Read) Definitions

In response to the Status Packet

Field #	Field	Value	Length(bytes)	Range	Function
1	Header	String “RRM”	3	3	Header
2	Packet Type	String “U”	1	1	Packet Type
3	P_Length	8 bit Unsigned Integer	1	5-255	Length of final packet
4	Knob 1	16 bit Unsigned Integer	2	0-0xFFFF	Focus Value (Position)
5	Knob 2	16 bit Unsigned Integer	2	0-0xFFFF	Iris Value (Position)
6	Zoom	16 bit Unsigned Integer	2	0-0xFFFF	Zoom Joystick Value (Position)
7	NAV Joystick X	16 bit Signed Integer	2	-0x800 - 0x800	Navigator Joystick X (left/right) Value (Position)
8	NAV Joystick Y	16 bit Signed Integer	2	-0x800 - 0x800	Navigator Joystick Y (up/down) Value (Position)
9	SAT Joystick X	16 bit Signed Integer	2	-0x800 - 0x800	Satellite Joystick X (left/right) Value (Position)
10	SAT Joystick Y	16 bit Signed Integer	2	-0x800 - 0x800	Satellite Joystick Y (up/down) Value (Position)
11	NAV Button	8 bit Unsigned Integer	1	0-0xFF	Navigator button bitmap
12	NAV Dial	8 bit Unsigned Integer	1	0-0xFF	Navigator dial bitmap
13	Battery	8 bit Unsigned Integer	1	0-0xFF	Battery voltage (123[0x7B] = 12.3 Volts) Implied Decimal
14	Record Enabled	8 bit Unsigned Integer	1	0-1	Recording Enabled