Remote I/O Protocol Client Library Alire Crate for GNAT Ada

Revision 1
5 August 2021

by Philip Munts
President, Munts Technologies
http://tech.munts.com

Remote I/O Protocol

The ***Remote I/O Protocol*** is a lightweight message protocol for performing remote I/O operations. The protocol is implemented using a request/reply pattern, where the master device (*e.g.* a Linux computer) transmits an I/O request in a 64-byte message to the slave device (*e.g.* a single chip microcontroller). The slave device performs the requested I/O operation and returns an I/O response in a 64-byte message back to the master device.

The protocol is kept as simple as possible (exactly one 64-byte request message and one 64-byte response message) to allow using low end single chip microcontrollers such as the **PIC16F1455** for the slave device. Although particularly suited for USB raw HID devices, this protocol can use any transport mechanism that can reliably transmit and receive 64-byte messages.

See the ***Remote I/O Protocol Specification*** document for more details.

About this Crate

This crate contains a subset of the ***Linux Simple I/O Library*** Ada packages that are relevant for building ***Remote I/O Protocol*** client programs.

This crate can be built directly or indirectly (as a dependency) on Linux and Microsoft Windows 10 64-bit.

This crate includes a project for building four test programs that display the capabilities of the connected ***Remote I/O Protocol*** server, each using one of:

- raw HID (Human Interface Device) via **HIDAPI**,
- raw HID via **libusb**
- the **Stream Framing Protocol** over an asynchronous serial port
- UDP (User Datagram Protocol) over a TCP/IP network

You can study these four test programs to learn how to write client programs (in **`src/programs/`** in the library crate project directory) using each of the four transport methods.

Cross-Compiling

You can cross-compile this crate or a program using this crate for a Linux target computer by just copying the target configuration project file (**`.cgpr`**) from the cross-toolchain to **`default.cgpr`** in the Alire project directory. For example, to cross-compile for a Raspberry Pi 1 microcomputer running ***MuntsOS Embedded Linux***, using the MuntsOS cross-toolchain for the Raspberry Pi 1, just copy **`RaspberryPi1.cgpr`** to **`default.cgpr`**.

Web Links

Remote I/O Protocol Specification:
http://git.munts.com/libsimpleio/doc/RemoteIOProtocol.pdf

Remote I/O Protocol Ada example programs:
http://git.munts.com/libsimpleio/ada/programs/remoteio

Linux Simple I/O Library:
https://github.com/pmunts/libsimpleio

Stream Framing Protocol Specification:
http://git.munts.com/libsimpleio/doc/StreamFramingProtocol.pdf

MuntsOS Embedded Linux:
https://github.com/pmunts/muntsos

MuntsOS Embedded Linux Thin Servers (the GPIO servers include Remote I/O protocol):
http://repo.munts.com/muntsos/thinservers

Buy a dedicated Remote I/O Server:
https://www.tindie.com/products/pmunts/usb-flexible-io-adapter
https://www.tindie.com/products/pmunts/usb-grove-adapter

HIDAPI library for HID (Human Interface Device) device access:
https://github.com/libusb/hidapi

libusb library for USB device access:
https://github.com/libusb/libusb