Wave Glider Software Architecture

An Introduction for Developers

The Wave Glider ™ is an autonomous ocean-going sensor platform that depends on software to direct its operation and its payloads' operation. You'll find its hardware architecture described in a companion white paper "Wave Glider Hardware Architecture." This white paper describes the processes, communication channels, and interfaces that define Wave Glider software architecture.

Wave Glider software contains both onboard and shore-side components. The onboard components run on each Wave Glider where they control its operation, communicate among themselves, and communicate with shore-side components.

Shore-side components run on shore or on a ship where they communicate with Wave Gliders and handle Wave Glider data. Human operators use shore-side operations components to command Wave Gliders, examine incoming data, and send files to Wave Glider components. Automated processes running shore-side may also command Wave Gliders and work with data going to and from the Wave Gliders.

Liquid Robotics (LRI) supplies a complete supporting system of onboard and shore-side software components. Mission operators can use that system to control general Wave Glider operation; developers can use the system as a transparent data tunnel that connects their onboard processes with their shore-side processes. The architecture is customizable and allows developers to replace LRI components with their own components at almost all levels for maximum flexibility.

Communication Channels

A Wave Glider's hardware determines both onboard and shore-side communication channels. You can read about the physical aspects of these channels in the hardware overview. Because they play such an important part in the Wave Glider software architecture, we discuss them here as well.

Onboard Channels

A Wave Glider may have up to three kinds of onboard communication channels. Two of them are standard in hardware but optional in use; the third is completely optional.

LRI # 070-01528
Version 1.0, March 23 2011

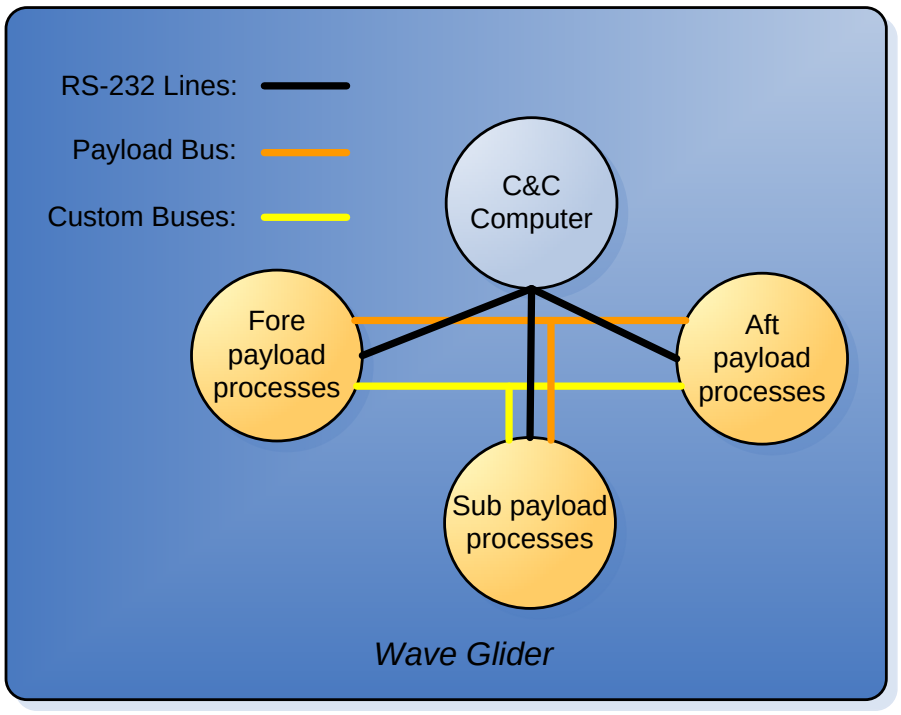

Figure 1: A Wave Glider can support three types of onboard communication channels. Developer processes appear in orange.

The RS-232 Lines

LRI supplies RS-232 lines for connections between the C&C module and float payload dry boxes. Within each payload, the line may extend to components there attached to payload integration boards (PIBs) or to custom payload boards. The RS-232 lines do not extend down the tether to the sub payload.

The RS-232 lines transmit serial data at 115,200 bits per second. Although using the lines is optional, they are almost always used between payloads and the C&C. Without the RS-232 lines, a payload can't communicate with the C&C computer to issue commands and communicate with shore-side operations using the C&C module's communication channels. The RS-232 lines also support inter-payload communications.

The Payload Bus

LRI supplies a pair of payload bus lines that connects payload dry boxes, but does not connect to the C&C computer. The payload bus provides physical transport that is completely customizable: developers can implement a communication channel there using any physical or software protocol they'd like. The payload bus may be useful for custom communications among payload components. Because it extends down to a sub payload, using it is necessary if a sub payload wishes to communicate with a float payload.

Processes can't use the payload bus to communicate with the C&C computer; they must go through the RS-232 lines.

Custom Buses

Developers can build and install their own connections among payloads using any hardware or software standards they care to use. Developers must handle all communications on a custom bus using their own software. A custom bus can't communicate directly with the C&C computer.

Shore-Side Channels

A Wave Glider supplies two external communication channels for data exchange with shore-side components. Developers can use either of them and can add a custom communication channel. On shore (and on ship), standard network connections provide communication among components.

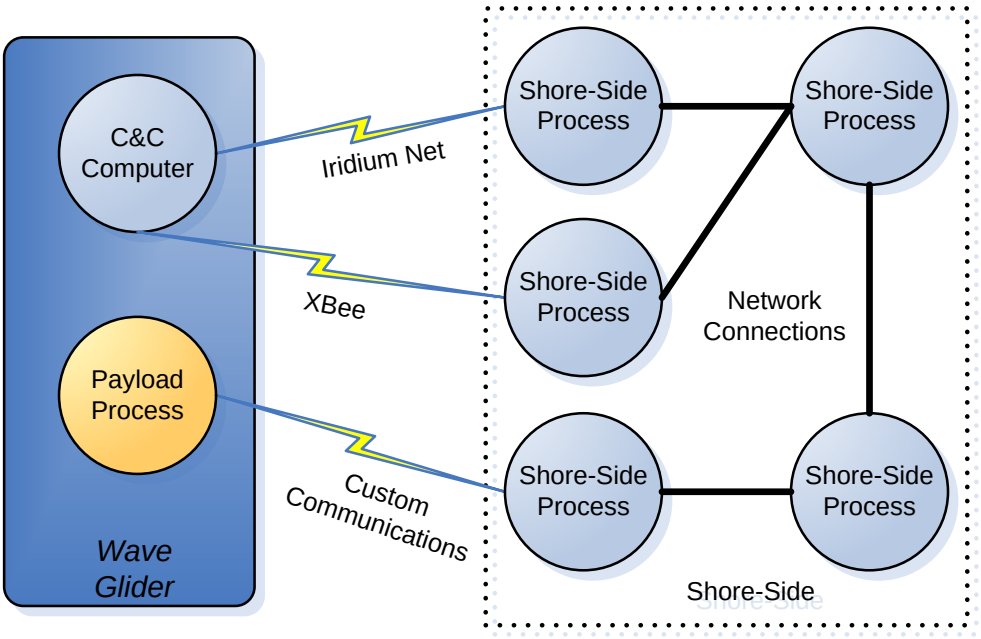

Figure 2: A Wave Glider may support three shore-side communication channels. Shore-side components may use standard network communications to communicate. Developer processes appear in orange.

The Iridium Satellite Network

A Wave Glider's Iridium modem can send and receive data through the Iridium satellite network. Onshore, the Iridium company's computers can send and receive Wave Glider data over the Internet to exchange data between a Wave Glider and shore-side processes. A shore-side computer without Internet access may have its own Iridium modem so that it can communicate directly over the satellite network with a Wave Glider. This paper describes both communication methods later.

Iridium data transmission is relatively slow and expensive, but provides global access for communication almost anywhere in the world.

The XBee Wireless Network

An XBee modem in a Wave Glider can communicate with one or more nearby XBee modems (within 100 meters) in other computers to create an ad hoc wireless network. XBee communication is much faster and cheaper than Iridium communication, and is ideal for testing in a laboratory. Operators may also use it between a ship and a nearby Wave Glider.

Custom Communications

Developers can create their own communication channel by installing custom communications gear on a Wave Glider and managing it as part of a payload. A custom communications channel can use alternate satellite networks or other connections to exchange data with shore-side components. Developers are responsible for routing and dealing with data on both sides of a customs communication channel.

Standard Network Connections: TCP/IP, HTTP, SSL and Email Connections

Shore-side components off the Wave Glider often use standard Internet connection protocols to exchange data. They use TCP/IP, for example, to convey data to and from Iridium or XBee channels to shore-side operations components that manage a Wave Glider. HTTP and SSL connections let human operators and automated processes control operations components. And shore-side components can send email to human operators to notify them of Wave Glider events.

These network connections may take place over the Internet or, if all computers are local, within a local network. They can also take place within a single computer if the entire shore-side software installation resides on a single computer.

Components

Wave Glider software components are processes that run both onboard and shore-side.

Onboard Components

Processes running onboard a Wave Glider are either LRI processes that run in the C&C module or developer processes that run in the payloads.

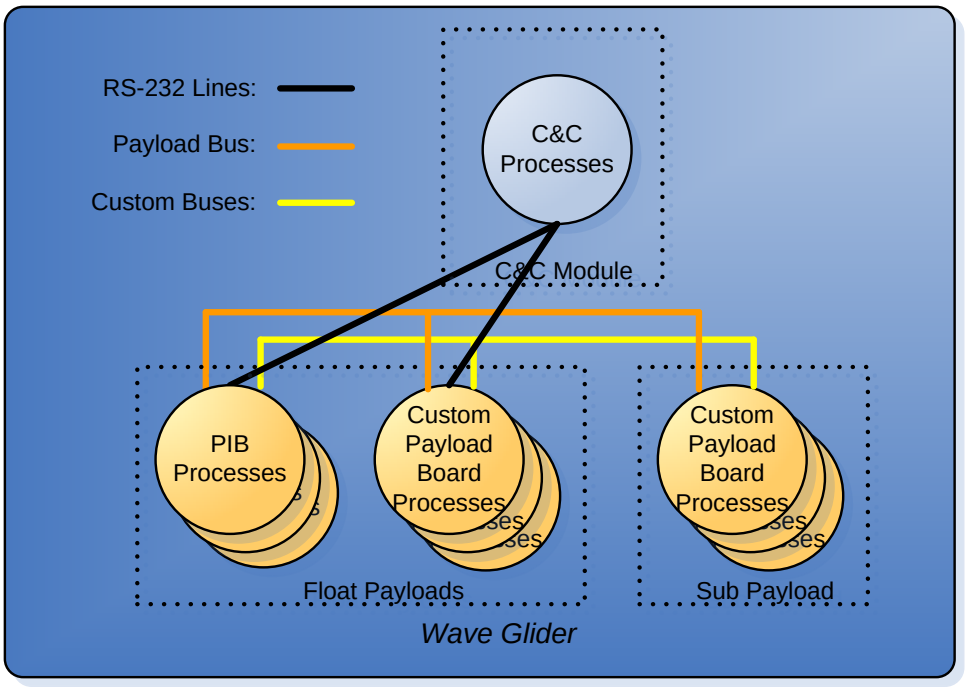

Figure 3: Processes that may run onboard a Wave Glider. Developer processes appear in orange.

Command and Control (C&C) Processes

C&C processes run on the C&C computer. They control vehicle operation such as navigation, sending regular telemetry, and reporting alarms. They read and report on C&C internal sensors that include GPS location and temperature. They also manage communication, sending and routing messages among both on-board and shore-side processes.

C&C processes execute commands coming from shore-side operators or from onboard payloads. Those commands can determine a course, set telemetry intervals, turn components on and off, transfer files to payloads, and perform other tasks. Developers can control C&C processes, but they can't customize the processes.

Payload Integration Board (PIB) Processes

PIB processes are developer-written and run on one or more PIBs that may be located in each float payload dry box. Each PIB runs MicroC/OS, an embedded multitasking operating system, and communicates with the C&C computer and other payload processes via the RS-232 lines.

Developer PIB processes are typically written in C. They can use an LRI-supplied communications library to exchange data and commands with other payloads, the C&C computer, and shore-side processes. They can also sidestep the communications library to handle the details of data and command exchange themselves.

PIB processes can communicate with and control hardware components attached to the PIB via personality modules.

Custom Payload Board Processes

Developers may create their own intelligent payload board to stand in place of a PIB or to connect to a PIB through a personality module. The developer determines the CPU and operating system the board will use, and writes supporting software. The board may connect directly to the RS-232 lines if it's in a float payload, in which case processes running on the board can use the LRI communications library to communicate via the RS-232 lines, or they can communicate using their own functions.

A custom payload board may also connect to other payload components using the payload bus instead of the RS-232 lines. If so, processes running on the board may communicate with other payload components using whatever communication protocol is set up on the bus.

Sub payloads *must* use the payload bus to communicate with float payloads since there are no RS-232 lines going to the sub. To communicate with the C&C or use C&C external communication channels, a sub process sends data to a float payload component that passes on data via RS-232 lines to the C&C.

If a float payload has a custom on-board communications channel available (created by the developer), custom payload processes may use that custom channel. Developers using a custom channel create all their own supporting software for the channel.

Shore-Side Components

Wave Gliders are autonomous, but they may be controlled remotely and they regularly exchange data with operations on shore or ship. Liquid Robotics supplies an extensive and flexible system of shore-side software components that support Wave Glider control and data exchange. Developers typically use this system, but may in some cases create their own operations management system or adapt an existing system.

Wave Glider shore-side components may be divided into two main types:

- *Communication channel relays* sit at the shore-side end of external communication channels and move data packets between the channels and other shore-side processes.
- *Operations processes* handle data coming from and going to relays. They allow human and automated operators to control Wave Glider operation and to exchange data with Wave Gliders.

Communication Channel Relays

There's a relay for each type of possible external communication channel available to a Wave Glider.

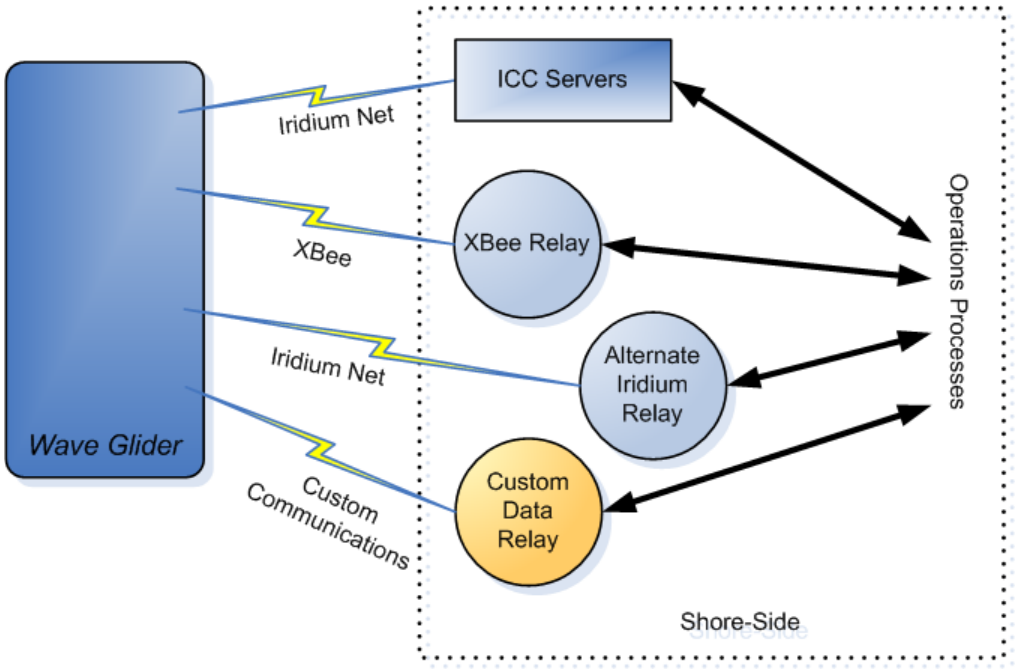

Figure 4: Possible communication channel relays in the Wave Glider software architecture. Developer processes appear in orange.

Iridium Communications Corporation (ICC) Servers

The Iridium Communications Corporation (ICC) maintains a set of company servers that accept Short Burst Data (SBD) packets from Iridium modems on Wave Gliders. The ICC servers extract data from the SBD packets and then send that data via Internet connections (using TCP/IP protocols) to Wave Glider operations management processes.

The ICC controls its own servers—developers can't alter them. Each Wave Glider Iridium modem is mapped to a URL, though, so incoming data goes to that URL where developers can work with the data as they see fit. Outgoing data specifies a Wave Glider modem; the ICC servers pass on the data via SBD to that Wave Glider.

The XBee Relay

The XBee relay is an LRI process that runs on an XBee-enabled computer. The XBee relay exchanges packets via XBee with a Wave Glider. The relay can use the Internet to pass packets on to operations management processes just as the ICC servers do. Because the relay runs on a computer under LRI or a developer's control, though, the relay also has the option of exchanging packets via an internal network instead of the Internet. If operations processes run on the same computer as the XBee relay, the relay can communicate directly with the processes without using a network.

The Alternate Iridium Relay

Operations management processes don't always have Internet connections so they can't receive Iridium packets from the ICC servers. A ship at sea, for example, may not have Internet access. If so, operations processes can bypass the ICC servers using a local

Iridium modem that exchanges data directly with a Wave Glider's Iridium modem—an alternate Iridium connection.

LRI supplies an alternate Iridium relay to handle data exchange through an Iridium modem. The relay runs on the computer attached to the modem and, like the XBee relay, conveys packets between operations processes and the Wave Glider.

Custom Data Relay

When a developer creates a custom communication channel, the developer is responsible for exchanging data with the Wave Glider and passing data back and forth between the Wave Glider and operations processes.

Operations Processes

Operations processes accept, store, and act on data coming from Wave Gliders and their payloads. They also let human operators and automated processes issue commands and send data to Wave Gliders and their payloads. Wave Gliders may work with any of three possible operations systems: the Wave Glider Management System, the XBee CLI, or a custom operations system.

The Wave Glider Management System (WGMS)

Most Wave Glider missions use the Wave Glider Management System (WGMS), a full-featured operations system that runs on LRI-hosted computers maintained 24 hours a day, 7 days a week. WGMS may also run on a developer's own computers, and can even run on a single laptop if desired for working with Wave Gliders in the field.

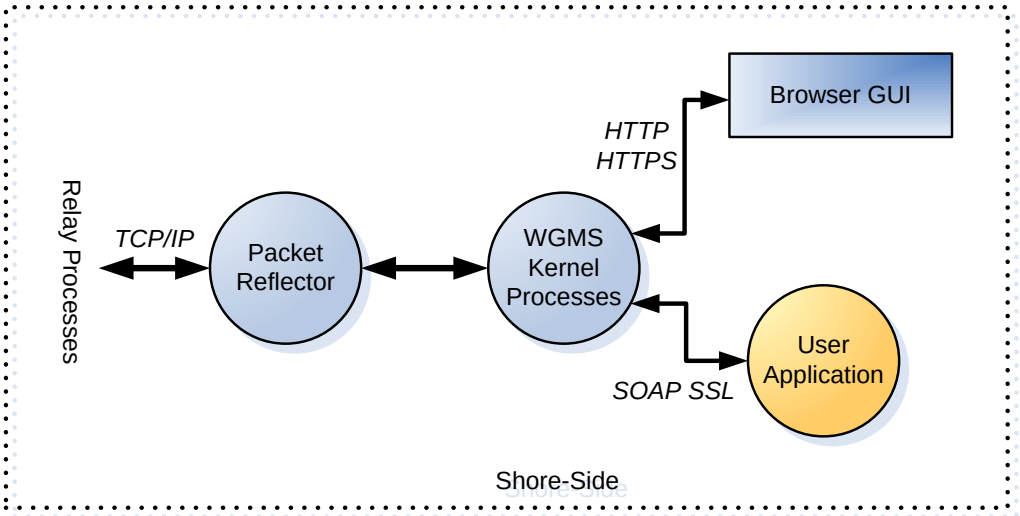

Figure 5: The Wave Glider Management System has four main components. Developer processes appear in orange.

WGMS has four main components:

- A *packet reflector* routes packets of data between communications relays (ICC, XBee, alternate Iridium, and custom) and one or more WGMS kernels. A packet reflector is connected to these components via TCP/IP. A WGMS system may have multiple packet reflectors, each connecting to one or more kernels.

- *Kernel processes* are the heart of WGMS and include a communications server, a relational database server, and the kernel itself. These processes receive packets from Wave Gliders, extract data from the packets, and store the data in a database. They also handle user authentication and authorization, present data to operators, accept commands from operators, and send commands and data to Wave Gliders after formatting the commands and data for Wave Glider reception.

  A system can have multiple kernels, each working with a single packet reflector.

- A *browser GUI (Graphical User Interface)* is a standard web browser that presents an easy-to-use Wave Glider console to human operators. Operators use the console for many activities that include tracking multiple Wave Glider locations and past courses, setting new courses, issuing other operating commands to Wave Gliders, exchanging files with payloads, and retrieving past transmissions to and from Wave Gliders.

  The browser GUI communicates with the kernel through an HTTP/HTTPS connection. The kernel presents displays on the browser GUI using HTML and JavaScript.

- A *user application* is a custom-developed process that communicates with the kernel using web services calls (SOAP 1.1 or 2.2) over an HTTPS connection. A user application can perform all the Wave Glider activities available through the browser GUI, but because it's a process running on its own can automate those activities. A user application in this case is an automated operator. A user application may also provide alternate controls, such as payload-specific controls, to human operators.

The XBee CLI

The XBee CLI (Command Line Interface) is a very simple operations system designed to test a Wave Glider's operation in the lab. To use the XBee CLI, a computer with an XBee transceiver and the XBee relay must connect to a Wave Glider. A terminal emulator program running on the computer then connects through the XBee relay to the Wave Glider's C&C computer. The C&C computer presents a text-based menu of possible commands in the emulator window.

An operator enters a numeral to specify a command; the Wave Glider executes the command and sends an acknowledgment that may contain data about the command's execution.

The XBee CLI command set is simpler than the full command set available through WGMS.

Custom Operations System

Developers typically use WGMS for Wave Glider operations management, but in rare cases they may create their own system from scratch. A custom operations system must

be able to send and receive packets through communications relays; it must also be able to format data in packets that the Wave Glider C&C can read and be able to retrieve data from C&C-created packets. What the system does with that data is up to the developer.

Wave Glider Software Interfaces

Wave Glider software architecture exposes four LRI-defined software interfaces to developers. Three of these interfaces are packet formats that convey data among the C&C computer, onboard payload processes, and shore-side operations processes. They are:

- *C&C messages*
- *Operations packets*
- *Vehicle packets*

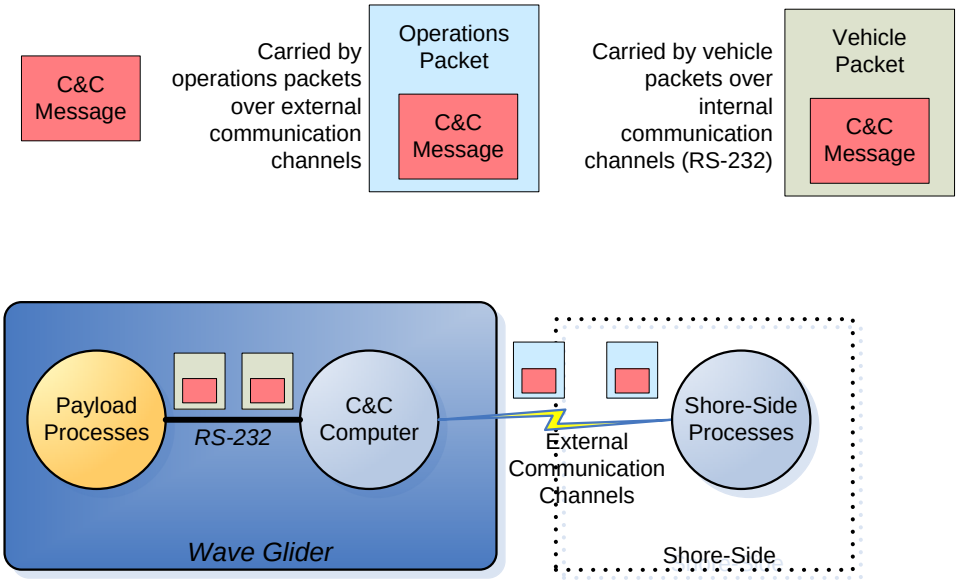

Figure 6: C&C messages pass between the C&C and payload and shore-side processes. Operations packets convey C&C messages and other data between shore-side processes and the C&C. Vehicle packets convey C&C messages and other data between onboard processes and the C&C.

Developers need not use any of these packet interfaces directly. On the operations side, WGMS makes takes care of formatting for C&C message and operations packets so they're completely transparent to operators and user applications. On the vehicle side, the LRI communications library makes C&C messages and vehicle packet formatting transparent to processes running in payloads.

The fourth LRI-defined interface is the WGMS SOAP API. A user application uses the API to post requests to and receive responses from the WGMS kernel. Developers use this interface if they decide to integrate a user application with WGMS.

C&C Messages

C&C messages travel between the C&C computer and processes running either onboard a Wave Glider or shore-side. The messages convey commands to the C&C that control Wave Glider operation. They also carry responses back to command senders, and they transmit telemetry from the C&C to onboard and shore-side processes. C&C messages travel over both the internal RS-232 line and the external communication channels.

The command set specified within C&C messages control the C&C computer's operation. They can:

- Control vehicle operation (moving the rudder or turning on a light, for example).
- Set system parameters (change the telemetry transmission interval, for example)
- Communicate with a payload process.
- Navigate by defining a course and asking that it be followed.

Operations Packets

Operations packets are a wrapper for data exchange over external communication channels. They convey C&C messages and other data. Their format differs slightly depending on the communication channel used and whether data's going to or from the Wave Glider. Information in their header defines, among other things, how the packet will be transmitted over the channel.

Vehicle Packets

Vehicle packets are a wrapper for data exchange over the Wave Glider's internal RS-232 line. They convey C&C messages and other data. Each vehicle packet contains the address of its intended recipient, which can be the C&C computer, a payload process, or a general broadcast to all onboard processes. The address may also specify an external communication channel such as Iridium or XBee.

Onboard processes that receive a vehicle packet must forward it to other processes if the receiver controls the connection to those processes. When the C&C receives a vehicle packet addressed to it, it extracts the packet data and, if it's a C&C message with a command, executes that command. If a message is addressed to an external communication channel, the C&C extracts the packet data, wraps it in an appropriate operations packet, and transmits the packet over the channel.

Payload processes don't need to deal with vehicle packets directly. They can use functions in LRI communications library that create, address, and handle vehicle packets. They also handle creating C&C messages.

The WGMS SOAP API

Developers may, as described earlier, create a user application that connects to the WGMS kernel and acts as an automated operator or provides additional controls to a human operator. That application communicates with the kernel using the LRI-supplied

WGMS SOAP API. The API provides commands that read, write, and revise data stored by the WGMS kernel and that issue commands and exchange data with Wave Gliders.