Product Information

Messenger

Advanced messaging for distributed and high availability architectures

LYNUXWORKS™

Messenger implements a message-based communication protocol for intelligent devices to cooperatively process information. The architecture is designed to facilitate intelligent I/O processor cards (IOP), and increase system scalability and availability. Peer-to-peer communications allow developers to flexibly integrate different I/O resources, enable direct communications between these I/O devices, and deploy split device drivers running in a High Availability system. Applications and split device drivers communicate to each other over the underlying I/O bus using the services defined by Messenger's APIs.

Messenger is an essential part of systems that are distributed across processors and must interact with one another in a High Availability distributed software/hardware architecture with N+1 redundancy.

By abstracting the system architecture and bus topology, Messenger allows applications to view their distributed environment as a single "virtual computer." High Availability processes execute without regard to processor location and do not need to address other processes/CPUs with reference to platform specifics. Such abstraction fosters the generality needed to deploy applications in a High Availability CompactPCI environment. Moreover, the abstraction eases prototyping while facilitating system monitoring and dynamic fault management.

Messenger is designed to work in a heterogeneous computing and operating systems environment. A distributed High Availability system may be made up of different kinds of computing nodes running different operating systems. LynxOS may run on nodes with real-time requirements, while other nodes run on Linux applications. The OS Abstraction Layer of Messenger is portable and can be ported to run on different operating systems.

Toolkit Capabilities

Messenger includes object forms of the following components:

- Messenger—Message communications facilities and software
- I/O Processor Manager—A software component that manages subordinate intelligent devices in a PCI domain. IOP Manager provides an interface between its clients and the host OS Hot Swap/High Availability Support subsystem. IOP management functions include:
  - —Configuration of intelligent devices on the PCI bus
  - —Boot and reset of intelligent devices
  - —Detection and handling of failures that occur at the intelligent devices
- MINET network driver—Messenger Interface Network, or MINET, is a data-link driver, implementing TCP/IP connectivity across the Messenger domain. MINET supports the familiar LynxOS TCP/IP APIs to implement communications between intelligent devices in a distributed IP environment
- Utilities, debug facilities, and management statistics

Enables Node-to-Node Communications and Full Mesh Connectivity

Messenger implements peer-to-peer and host-to-IOP models of communication. The concepts of Communications Paths and Messages allow a full mesh connectivity of Logical Devices: One to Many: Many to One: from logical module to logical module.

Integrated with LynxOS Hot Swap and High Availability

Messenger provides the most flexible approach to fault-tolerant real-time distributed systems in the industry. Split processing applications and IOPs can be mixed and matched for specific configuration requirements. In a High Availability and Hot Swap environment, the system's capabilities can be re-provisioned dynamically and expanded without interruption, or downtime.

In addition to loading LynxOS onto subordinate IOPs at the system initialization time, the IOP Manager is responsible for recovering IOPs from failures, including the Hot Swap Extraction/Insertion events.

Scalable and Dynamic Applications

Messenger supports the dynamic configuration of distributed applications and IOPs. The Logical Device Module naming scheme uses C-style character string object handles that allow InterProcess communications to be based on object types and properties, rather than on static addresses.

Designed for Performance

Support for message-passing among multiple independent processors greatly improves I/O performance in high-bandwidth applications such as layered communications protocol processing, VoIP, routing, and networked video applications by off-loading I/O intensive tasks from a single host processor onto specialized IOPs. Messenger implements a fast and slim asynchronous data-link messaging model.

For more robust communications and distributed applications, MINET interfaces directly with the LynxOS TCP/IP stack for IP network addressing and routing. MINET makes Messenger nodes appear like nodes on an Ethernet network, allowing programmers to use standard, off-the-shelf network development tools and applications.

Designed for Portability

The communications model for Messenger architecture is a message-passing system. The communication model is analogous to a connection-oriented networking protocol or the OSI layered model, in which two entities exchange messages by using the message layer to set up a connection and exchange data and control.

The Messenger architecture is written specifically to be I/O processor-independent, allowing maximum flexibility in choosing an IOP that best suits a particular performance and cost objective. Messenger imposes no restriction on where layered modules execute, providing support for multi-processor and clustered systems. Messenger is designed to support different types of transport media, including: serial, TDM I/O buses, intelligent switch fabric chip interfaces, and network-facilitated clustered systems.

Messenger APIs

Messenger includes a well-defined set of APIs to support all aspects of split application, or driver development. The Messenger APIs for distributed InterProcess Communications (IPC) complements a rich selection of POSIX APIs available in the LynxOS real-time operating system.

Message passing is one type of InterProcess Communication used in distributed, parallel systems. The messages consist of a very small header and the information to be transferred (payload). When a process is ready to send a message, it calls the Messenger Logical Device Module by submitting the name of the destination logical device and the payload information.

A message passing protocol is the logical model for building high-performance, fault-tolerant distributed systems.

- Object Hierarchy
- Message Instance
- Logical Device
- Path Pattern
- Communication Path
- Communication Path State
- Message Sending
- Message Receiving
- Message Frame
- List of Waitable Objects
- Static Message
- Scatter Gather Lists
- Service Status

Additional Messenger Services

Messenger includes several additional facilities and utilities that simplify debugging, testing and managing distributed applications, these include:

- Messenger Debug Services
- Messenger Statistics
- IOP Configuration and Booting
- IOP Manager Control Utility
- IOP Health Heartbeat

1.800.255.5969

LynuxWorks, Inc.
855 Branham Lane East
San Jose, CA 95138-1018
408·979·3900
408·979·3920 fax
www.lynuxworks.com

LynuxWorks Europe
2 Allee de la Fresnerie
78330 Fontenay Le Fleury
France
+33 1 30 85 06 00
+33 1 30 85 06 06 fax