A MODULAR RANGE INTERFACE FOR ACQUISITION AND DISTRIBUTION OF TEST RANGE DATA

Thomas M. Marler
TYBRIN Corporation
Fort Walton Beach, Florida
under contract to
Freeman Computer Sciences Center
Eglin AFB, Florida

D. Kelly Cooper
TYBRIN Corporation
Fort Walton Beach, Florida
under contract to
Freeman Computer Sciences Center
Eglin AFB, Florida

William F. Lake
Freeman Computer Sciences Center
Test and Analysis Division
Eglin AFB, Florida

ABSTRACT

A flexible, modular method is needed to connect test range data systems to central real-time computer networks. This is achieved by the development of a real-time, networked, VME-based range interface system. Flexibility is achieved by a modular hardware and software design. The modular hardware consists of standard network interfaces, COTS VME interfaces, and a VME single board computer (with an onboard PCI bus). The modular software is implemented in C++ using the VxWorks real-time operating system. This paper describes the conceptual design and development of the Modular Range Interface (ModRI).

KEY WORDS

TSPI, real-time, networks, C++, VME

INTRODUCTION

One of the primary missions of Eglin AFB, Florida, is the testing of combat systems for multi-service and international customers. To accomplish this testing, Eglin explores ranges instrumented with data gathering systems that interface with centralized real-time computer networks. Traditional methods were used over the last 25 years to connect these range data systems to real-time computer networks. Since no interface standards existed, each equipment manufacturer developed proprietary input/output (I/O) interfaces. This resulted in one-of-a-kind interface systems. The unique systems that evolved through the course of time are now difficult to use and expensive to maintain. In order to provide customers with competitive and flexible solutions in an ever-changing testing environment, the Freeman Computer Sciences Center located at Eglin AFB is currently upgrading these legacy systems to modern software and hardware technologies.

Here at the Freeman Computer Sciences Center, customers can perform mission control and real-time data analysis during a test mission. The Center maintains the interface between the test range and real-time computer networks that support the customer. Currently these legacy systems consist of 1970's minicomputers coded in assembly language, using customized communication hardware and proprietary network protocols. These systems are both costly and difficult to maintain. Consequently, a flexible, modular method is needed to connect test range data systems to central real-time computer networks. These data systems provide the central real-time computer networks with Time-Space Position Information (TSPI) and Global Positioning System (GPS) data. This paper describes the design and development of a Modular Range Interface (ModRI) that can be easily configured to support connection of range data systems to centralized computers at any real-time test range.

A standard method for connecting data systems with centralized real-time networks has many advantages. ModRI is implemented with modular hardware and software components to maximize flexibility and portability. All of the hardware and software modules supporting a specific range data system are interchangeable between ModRI boxes. Furthermore, the entire ModRI box is easily transported from one range to another and simply plugged into a suitable network connection at the new location. An additional benefit is the reduction in application program changes as range data systems are added or removed or when real-time networks are upgraded.

Figure 1 depicts ModRI's implementation of range data system connections. All of the range-specific interface connections are localized to ModRI. ModRI receives and buffers incoming data, performs any needed data conversion or formatting, and sends the range data (in a standard format) to any real-time computer that requested the data. Data output to the range (slaving, control, closed loop, etc.) is the reverse of this process. ModRI can also transfer real-time data between ranges via standard Internet Protocol (IP) routers and wide area network (WAN) links. Figure 1 shows a phased-array radar tracking an aircraft, a GPS downlink system receiving position information from an aircraft, ship, and ground vehicle, and a multilateration system tracking the ground vehicle.

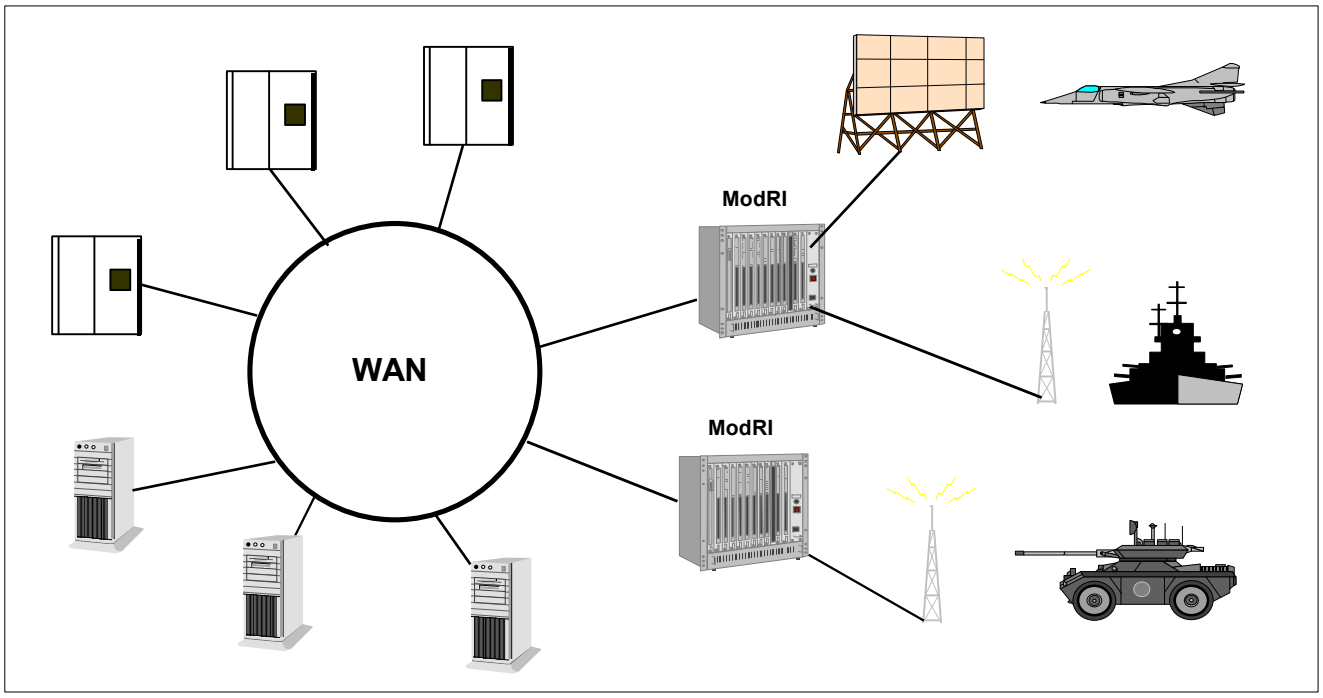

Figure 1. ModRI Range Connections

The goal of this project is to design, develop, demonstrate, and operationally use ModRI for real-time mission support. ModRI will accomplish this goal with modular hardware and software components, maximizing flexibility and portability while minimizing complexity and cost. The modular hardware design integrates modular hardware components into a commercial off-the-shelf (COTS) enclosure. Standard modular I/O hardware connects to the range data systems, and standard network interfaces connect to the real-time computer networks. All the modular hardware is controlled by a single board computer (SBC) supporting daughter board add-ons. The modular software design uses object-oriented design techniques and is implemented in an object-oriented language. The software development environment consists of a standard real-time operating system (or RTOS, which assures deterministic behavior) adhering to POSIX standards (ensuring platform independence). Many suitable interface standards have been exploited to build ModRI. These standards facilitate asset sharing between test ranges, lower the cost of testing, increase dual-use potential, and allow easy integration of new range data systems in a cost-effective manner.

CONCEPTUAL DESIGN

Hardware Design. Technology has advanced to the point where it is possible to build a complete interface system entirely from modular COTS hardware components. The key technologies exploited are VME enclosures with standard I/O hardware, fast SBCs, and standard network hardware.

VME enclosures are very durable and reliable under extreme conditions making them ideal for field use. As an industry standard for control and communications equipment, VME has over 5000 different interface cards supplied by more than 500 different manufacturers. Types of cards needed

for connection to range data systems include serial data (asynchronous, synchronous) supporting various communication protocols (RS-232, RS-422, SDLC, BISYNC, PCM), IRIG-B time code readers, precision interrupt timers, digital parallel (PIO, DMA transfers), Ethernet, and replicated shared memory.

Extremely fast 32-bit SBCs using Intel Pentium III chip technology are now available. These SBCs plug into a VME bus backplanes (ANSI/IEEE 1014-1987), support VME 64-bit data transfers, contain up to 256 MB of memory, have built-in Ethernet (IEEE 802.3) interfaces, and include a PCI bus (132 MB/sec). The onboard PCI bus accepts a PCI mezzanine card (PMC, IEEE P1386.1) that can support FDDI interfaces, ATM interfaces, or a bridge to a PMC expansion module. Various timers and a SCSI-2 bus are also implemented on the SBCs.

The most common standard network interfaces are Ethernet and FDDI. Ethernet bandwidth is 10/100 Mbit/sec while FDDI operates at 100 Mbit/sec. Ethernet is a CSMA/CD (carrier-sense multiple access with collision detection) network and can be used in real-time networks with a small number of nodes. Use of Ethernet in real-time with large numbers of nodes causes numerous collisions that introduce random latencies during data packet transmission. FDDI is a token ring-passing network well suited for real-time use. FDDI data transmission latencies are more accurately predicted, and FDDI networks are easily extended to multiple FDDI rings by using high-speed network switches to control latencies. ATM or Gigabit Ethernet networks implemented with high bandwidth switches will also work well in the real-time environment. Figure 2 depicts the system configuration for ModRI and the actual hardware that ModRI will use to implement its modular design.

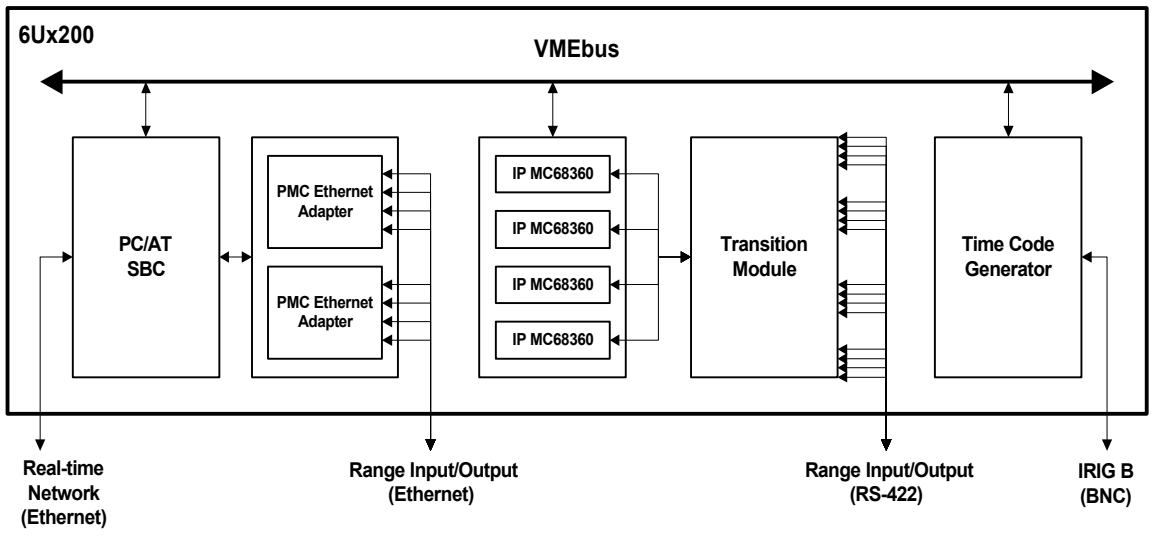

Figure 2. ModRI System Configuration

Another value-added aspect of ModRI concept is that current technology enables increasing the data rates from the test range. A simple way to enhance real-time TSPI processing is to increase the sample rate. Eglin's radars currently send data at 10 samples/sec but could easily operate at 40 samples/sec using ModRI. New real-time computers are now being installed that have the I/O bandwidth and power necessary to process TSPI data at these higher rates.

Software Design. To complement the modular hardware design, ModRI needs a modular software architecture that adheres to industry standards. ModRI utilizes software technologies that include a POSIX-compliant RTOS, standard networking protocols, and an object-oriented design methodology and programming language.

A widely used operating system for developing real-time applications is VxWorks software from Wind River Systems, Inc. VxWorks features a priority-based, preemptive multitasking microkernel executive suitable for applications requiring high-rate performance and deterministic behavior. VxWorks supports key industry standards including real-time POSIX (IEEE 1003.1), C/C++, TCP/IP, and includes a feature rich Integrated Development Environment (IDE).

ModRI takes advantage of the client/server architecture implemented on the real-time computer networks. As the interface between the test range and the real-time computer network, ModRI acts as a server to multiple real-time clients located on the network. These clients communicate with ModRI through Real-time Network Control (RTNC) software running on each client. RTNC handles data system selection/deselection and bi-directional data transfers between ModRI and the clients. Client applications call a Real-time Application Program Interface (RTAPI) to request data from any range data source connected to ModRI. The RTAPI supports one or more user processes running on a client. These clients connect to one or more ModRIs via multiple network links.

Currently, ModRI and RTNC communicate via raw sockets and a proprietary protocol; however, use of any OSI transport layer protocol suitable for real-time communications is acceptable. One suggestion is UDP/IP. The worldwide acceptance of this standard makes inter-range data communication possible (i.e., displaying local data on at a remote site), assuming there is sufficient bandwidth available on the network.

A modular software design is required under the ModRI philosophy. Modular software allows easy configuration of ModRI for different data system and network interfaces at the various test ranges. The software development approach that most closely fits ModRI's philosophy is an object-oriented analysis and design (OOAD). Achieving the best results requires good communication throughout the design process. To facilitate good communication, we propose using Unified Modeling Language (UML) diagrams and notation. Because of UML's wide acceptance, there is greater opportunity for productive design reviews, and professionals outside the ModRI project can easily understand the design. UML also allows electronic design verification using UML design software. Where possible, design patterns are used for maintainability. Design patterns are well documented, readily available, and highly optimal and modular. Figure 3 shows the design pattern solution to the modular hardware/software requirement. Using good OOAD practices allows the software to be as easily interchangeable as the hardware; thus keeping with the ModRI design philosophy.

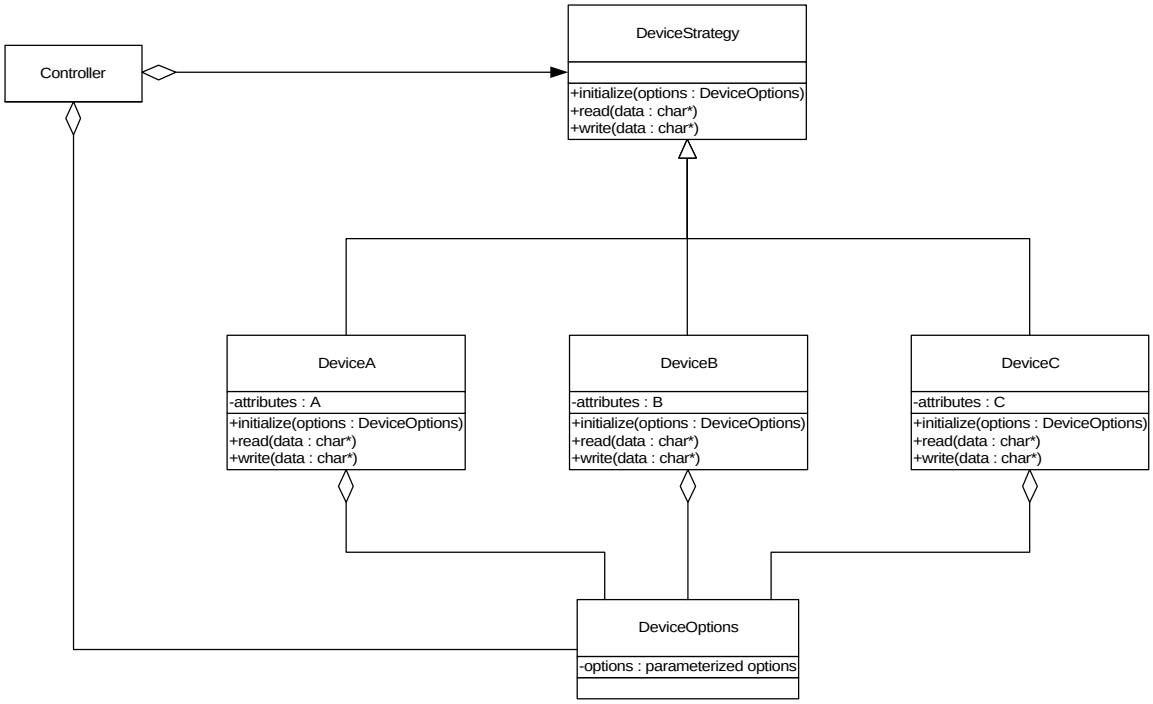

Figure 3. Virtual Hardware

Implementing a solid, modular software design requires a good OO language. ModRI needs a widely available and understood language. Additionally, ModRI needs the ability to control the low level devices while allowing flexibility for optimizations. C++ satisfies these requirements. A GNU C++ compiler is packaged with the Tornado II development environment for VxWorks. C++ offers good OO features with the flexibility to write hardware device drivers and to optimize algorithms for real-time applications.

CONCLUSION

The primary ModRI goal, as discussed above, is to support real-time range data transfers for all real-time mission activity at Eglin AFB, Florida. This goal is accomplished with a modular hardware and software design that results in the quick implementation of new range data systems, provides configuration flexibility as systems change, and replaces legacy equipment that is expensive to maintain (perhaps impossible to maintain in the near future). Additionally, the use of a modular hardware and software design provides portability—any range with existing data sources can use ModRI. As discussed, standardization keeps the cost and complexity low. Accordingly, ModRI utilizes a variety of VME COTS hardware, standard network protocols, and an OO paradigm. As a proof of concept, a ModRI prototype has been successfully developed and tested. Incremental development of the production model is currently in progress.

ACKNOWLEDGMENT

The 96th Communications Group, Test and Analysis Division, Eglin AFB, Florida funded this effort.

REFERENCES

1. Lake, William F., Jr., “Statement of Need (SON): Modular Range Interface,” February 18, 1999.

2. Lake, William F., Jr., “Modular Ranger Interface Briefing,” April 18, 1997.

3. Lake, William F., Jr., “Real-Time Communications Protocol for High-Speed Data Bus Project Design Plan,” March 1991.

4. Douglas, Bruce Powel, Doing Hard Time, Addison-Wesley Longman, Inc., 1999.

5. Fowler, Martin and Scott, Kendall, UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley Longman, Inc., 1997.

6. Schneider, Geri and Winters, Janson P., Applying Use Cases: A Practical Guide, Addison-Wesley Longman, Inc., 1998.

7. Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissides, John; Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman, Inc., 1995.

8. Stroustrup, Bjarne, The C++ Programming Language, Third Edition, Addison-Wesley Longman, Inc., 1997.

9. Wind River Systems, Tornado User Guide 2.0 (Window Version), Edition 1, Wind River Systems, Inc., 1999.

10. Wind River Systems, VxWorks Programmer Manual 5.4, Wind River Systems, Inc., 1999.

11. Wind River Systems, VxWorks Reference Manual 5.4, Wind River Systems, Inc., 1999.