ARTIC: A Communication Cornucopia

By Eddie Ho, David Carew, and Corradino Jones

The ARTIC family of adapters is widely used for serial communication in commercial DOS and OS/2 systems. The same programming environment and hardware is now available in AIX Version 3.2. This new addition can speed the proliferation of commercial and industrial AIX in a multi-user distributed environment.

The ARTIC (A Real-time Interface Coprocessor) family of programmable communication adapters can communicate with both standard and non-standard communication devices (such as ATM, point-of-sale terminals, and so on) via several electrical interfaces. The adapter has its own processor, memory, and communications hardware. With the RISC System/6000, ARTIC provides a loosely coupled multiprocessing system in which the adapter can handle all communications to a network.

The ARTIC support for AIX is a package of programming services that allows AIX processes to interface with tasks running on the adapters. Typical applications are shown in Figure 1.

Figure 2 shows a typical business application in which each regional office has a RISC System/6000 with an ARTIC adapter. The offices are connected through routers to a Wide Area Network (WAN) backbone.

Terminal Controller Area	Small Manufacturing Area	Networking Application
Point-Of-Sale Terminal	Food/Meat Processing Plant	SNA Network
Credit Card Authorization	Material Management	X.25 Communication
Reservation/Ticketing	Distribution Management	Bisynchronous Protocol
Restaurant Management	Inventory Control	Remote Job Entry

Figure 1. Typical applications of the ARTIC adapter

ARTIC Hardware

Three members of the ARTIC family are available in the RISC System/6000:

- Portmaster Adapter/A
- Multiport/2
- X.25 Interface Coprocessor/2

The Portmaster and the Multiport adapters each have a collection of daughter cards, known as Electrical Interface Boards (EIBs). Each EIB provides a variety of interfaces needed by the application that implements the networking layer. The protocol can be synchronous or asynchronous with some port restrictions on the synchronous usage. The adapter contains a multitasking operating system that can support up to 255 concurrent user-written tasks downloaded from the AIX application. Figure 3 shows the types of EIB for each adapter. The maximum speed shown is full duplex and available on all ports.

Programming in the ARTIC Environment

Since ARTIC adapters are intelligent, more extensive software is needed to control them as compared to traditional adapters. The programming environment is flexible enough to meet general-purpose needs as well as to support legacy or non-standard devices. Reference information is available in *Real-time Interface Coprocessor AIX Support for the RISC System/6000 User's Guide* (SC28-9596). The programming efforts (shown in Figure 4) concentrate on two major areas:

- **Programming the AIX application**. The application usually implements the upper networking layers. The network protocol can be

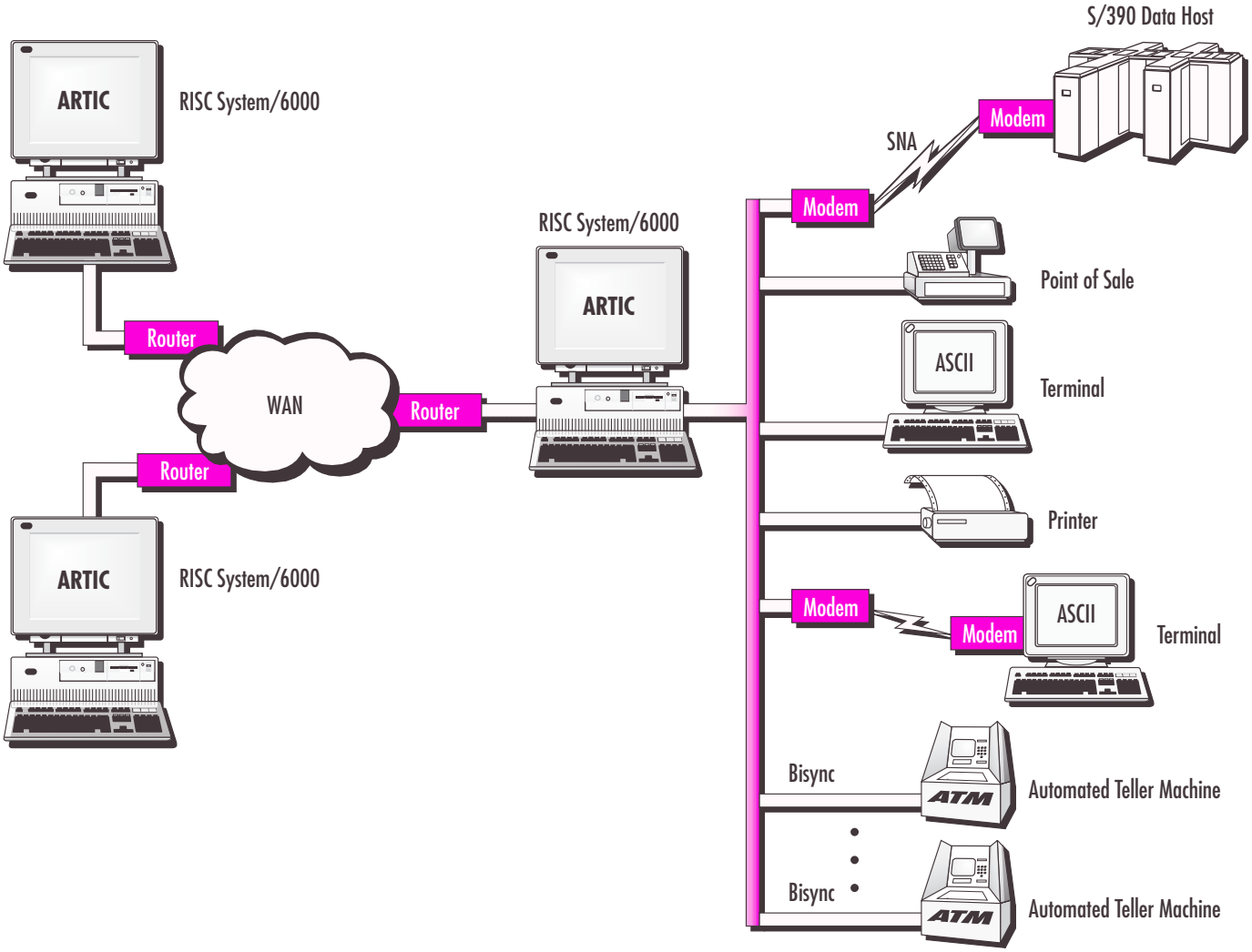

Figure 2. ARTIC in a branch office environment

Systems Network Architecture (SNA), bisynchronous, or some device-specific control protocol.

- **Programming the adapter tasks**. Adapter tasks can be loaded onto the adapter by the AIX application. These tasks are usually designed to handle input/output, data control, and processing.

Five major sections describe the AIX ARTIC programming toolkit: application loader, device driver, online dump facility, dump formatter facility, and C language interface routines.

Application Loader

The application loader allows application developers to download the ARTIC operating system and user-written applications onto the adapter. The loader can be invoked from the command line, from a shell script, or directly from the application program. The loader has several options:

- Loading a task but not executing it until further notice
- Loading a task at a specific memory location
- Resetting the adapter before loading the task

Device Driver

The device driver is the interface between AIX applications and adapter tasks. The device driver provides the following services to applications:

- Interrupting real-time tasks for asynchronous requests on the adapter

Family	Characteristics	EIB types	Interfaces	Max. Speed[1]
Portmaster	80C186 CPU 12.5 MHz RAM (.5,1.0, 2.0 MB) DMA (2 per port)	4-port selectable	RS-232 RS-422 X.21 X.21 bis V.35	64 KB
		8-port RS-232	RS-232 X.21 V.24	38.4 KB
		6-port RS-422	RS-422 X.21	64 KB
		6-port X.21	X.21	256 KB
		6-port V.35	V.35	256 KB
Multiport/2	80186 CPU 7.37 MHz RAM (.5, 1 MB) DMA (2 per port)	4-port RS-232	RS-232 X.21 bis V.24	9.6 KB
		8-port RS-232	RS-232 X.21 bis V.24	9.6 KB
		8-port RS-422	RS-422 X.21	9.6 KB
		6-port RS-232 synchronous	RS-232 X.32 bis X.21 V.24	9.6 KB
		4-port RS-232 and 4-port RS-422	RS-232 RS-422 X.21 V.24	9.6 KB
X.25 Interface Coprocessor/2	80186 CPU 7.37 MHz RAM (.5 MB)		X.21 bis X.21 V.35	19.2 KB for X.21 64 KB for V.35

[1] Kilobytes per second

Figure 3. EIB types for each class of adapters

- Reading and writing to the adapter's memory
- Serializing the execution of calls made from multiple applications
- Handling interrupts from tasks on the adapter and notifying applications

The device driver supports standard system calls, such as `open()`, `close()`, `select()`, and `ioctl()`. The `ioctl()` calls handle most adapter interaction. Figure 5 shows the types of `ioctl` calls supported.

Online Dump Facility

The dump facility assists developers in debugging. It allows the application to record and display information in the adapter memory, I/O port, and adapter registers. The recording can be immediate or can run as a background process triggered from a non-maskable interrupt.

Dump Formatter Facility

The dump formatter converts data generated by the dump facility into ASCII format that is usable by an AIX output device. The formatter organizes the dump data into an easy-to-read format using headers and blocks to group related information.

C Language Interface Routines

The C language programming interface enables AIX applications to issue commands to the ARTIC adapter. The application can link with the library routines, issue commands, and access adapter memory and task parameters.

Application Development for AIX

Because applications using the ARTIC adapter must communicate with tasks running on the adapter, they are tightly coupled and should be designed together. The design usually includes a definition of the commands and their data structure, as well as the definition of the data associated with interrupts from adapter tasks to the AIX application. Although the tasks on the adapter implement most functions, the AIX application can monitor tasks. There are several considerations in writing AIX applications:

- **Command interface**: The AIX application uses this interface to communicate with tasks on the adapter. It allows the application to pass commands and data to the adapter task. Decoding commands is dependent on the adapter task. For example, the command `0x01` can be interpreted by the adapter task as opening a port with the data containing the line characteristics.
- **Receiving interrupt**: The AIX application can receive interrupts with associated data from the adapter tasks. This process allows the adapter task to notify the AIX application asynchronously when attention is required. For example, a task can receive bisynchronous data, but it can interrupt the AIX application whenever it receives an entire frame.
- **Data alignment**: Data is represented differently in AIX and the adapter, so it must be normalized to the adapter data format before being passed to the adapter. The adapter data representation is in little-endian format, while AIX uses big-endian format. Figure 6 illustrates the differences for a double word with value `0xABCD` stored at address 4.

Task Development for the Adapter

The adapter tasks manage the communication ports, interface to the AIX application, and interface to other adapter tasks. Adapter tasks are primarily event driven. Each event originates either from the AIX application or from the adapter interrupt handler. The adapter task must be designed to handle all possible events and interrupts.

Each task can have three buffers to interface with the adapter operating system. The conventions for the buffer usage are as follows:

- **Output buffer**: Used by the AIX application to pass data to adapter tasks. Data must be loaded before a command is issued.
- **Input buffer**: Used by the adapter task to pass data to AIX. Data must be loaded before the task generates an interrupt to AIX.
- **Secondary Status buffer**: Used by the adapter task to report error information to the AIX application.

ARTIC Programming Environment

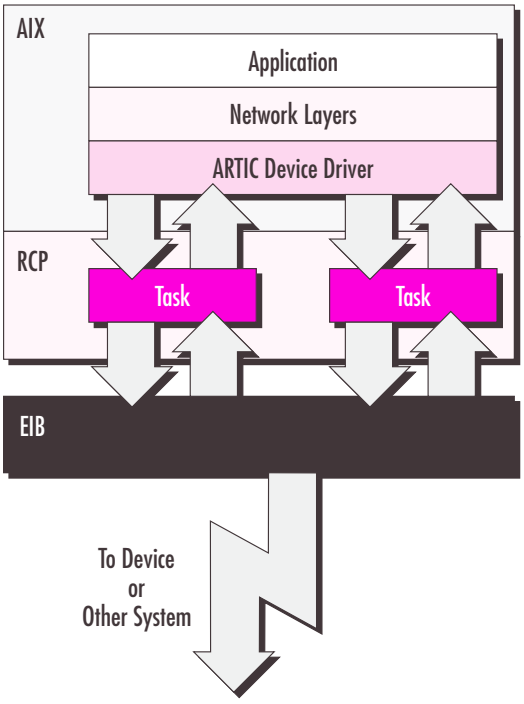

Figure 4. Structure of the ARTIC programming environment

Reset	Interrupt Wait	Get Primary Stat
Read Memory	Interrupt Deregister	Special Events Register
Write Memory	Get Parameters	Special Events Wait
Issue Command	Get Buffer Addresses	Special Events Deregister

Figure 5. The ioctl calls supported by the ARTIC adapter device driver

AIX and ARTIC Memory

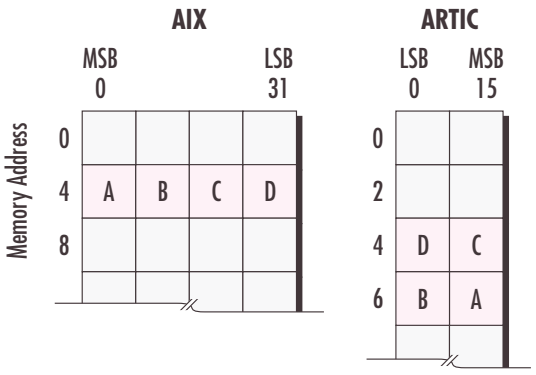

Figure 6. Data representation in AIX and ARTIC memory

Task Development Workbench

Developing an application for the adapter requires special tools. These tools are available in the AIX, DOS, or OS/2 environments:

- **Real-time Interface Coprocessor C Support Version 1.03**: Contains a set of libraries and "include" files for task development.
- **Microsoft C Optimizing Compiler 6.0**: Available in DOS, OS/2, or the PC Simulator on AIX.
- **Real-time Interface Coprocessor Services Version 1.02**: Provides a variety of services to assist in task development. The two major components are Communication Services, which provides the programming interface with the hardware, and Programming Services, which provides intertask communication for the adapter.

Performance Considerations

The ARTIC family is designed to relieve CPU workloads, which allows for greater throughput. The following performance considerations are helpful when writing applications:

- Off-load networking layers as much as possible to the adapter to minimize the AIX workload.
- Use Direct Memory Access (DMA) as much as possible for transmitting and receiving data.
- Use the execution priority scheme of the adapter to the best advantage. Try to assign priorities to tasks based on their predicted CPU load.
- Maximize the use of block transfer between RISC System/6000 and adapter tasks. Block transfers can eliminate byte swapping caused by different data format.
- Use kernel processes to handle as many real-time interrupts from the adapter as possible.

Conclusion

The ARTIC adapter delivers multitasking, real-time performance, and connectivity to the diverse equipment controllers and devices in a business. It is an intelligent, flexible adapter that can grow as business communications needs increase.

Eddie Ho, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a senior programmer in AIX Communications. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.*

David Carew, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Carew is a programmer in WAN Development Engineering. He has a BS and an MS in Computer Science from Northwestern University.*

Corradino Jones, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Jones is a senior associate programmer in AIX Communications. He has a BS in Computer Science from Grambling State University in Louisiana.*

COMMON Spring 1994 Conference

The annual COMMON Spring Conference will be held March 21–25 at the Anaheim Hilton and Towers (not affected by the recent earthquake) in Anaheim, California. The early registration deadline has been extended to March 4.

For more information, call:

Monika Paus
COMMON Headquarters
401 N. Michigan Avenue
Chicago, IL 60611-4267
Phone: (312) 644-6610
Fax: (312) 527-6657