ARCHITECTURE OF A MULTIPROCESSOR-BASED HIGH SPEED COMMUNICATIONS PROCESSOR

M. Sreetharan PhD
Senior Consultant, Syntek Systems Corporation

R. Stubblefield
Program Manager

S. Subramaniam MS
Senior Software Engineer
Aeronautical Radio Incorporated, Annapolis, MD

ABSTRACT

This paper details the system architecture of a communications processor that uses an existing distributed message switching data communications network as the transport medium for the Arinc Transaction Terminal Service (ATTS) application. Salient features of a powerful communications controller board are described and the organization of the software is outlined to show how the hardware/operating system dependence is minimized to enhance software portability.

INTRODUCTION

The availability of powerful computing devices and VLSI chips that integrate more and more higher level protocol layer functions in hardware have enabled development of cost effective alternate solutions to many existing data communication environments (Ref: 1-3). In particular, microcomputer-based communication devices are finding applications in computer networks which are being increasingly required to provide additional specialized user services to sustain their economic viability (Ref: 4-6).

The Arinc Data Network Service (ADNS) is a distributed message switching data communications network that supports a variety of communication protocols. For users mainly from the air transport industry, the network provides the following capabilities:

- Inquiry/response message routing
- Protected transmission of conventional messages according to industry standard format and practices
- Specialized message routing functions and information services

The ADNS consists of six Arinc Packet Switches (APS) which are streamlined for fast, high volume switching of message traffic between other network nodes. The APS has no user-line connections or user protocol software and no disk storage. It performs its message routing functions according to the destination node of each message and a knowledge of transmission delay times. A second type of node, the Arinc Message Processor (AMP), implements all message switching functions with protocol software appropriate for each of the connected user lines.

The airline reservation systems currently use a significant amount of long-distance leased line facilities. The Reservation Computer (RC) is connected to terminals that are distributed over the entire service area for the airline (in many cases these are world wide communications networks). These terminals are used for booking seats, printing tickets and boarding passes, checking passengers in and many other support services. The availability of these reservation system services is critical to the airline in the generation of revenue and maintenance of operational schedules.

The Arinc Transaction Terminal Service (ATTS) is the interface for airline
reservation systems into the ADNS. The spare capacity within the ADNS will be utilized to reduce the number of long distance leased line facilities required to service many of the remote terminal locations. This will provide a significant cost reduction while maintaining the high level of service required by the airlines.

The communications processor used for the ATTS service is a Protocol Adapter/Concentrator (PAC). This unit will provide the interface to the high level network without user modification for the routing and packetizing normally associated with a high level network (transparent interface). In this paper we will focus on the functional requirements and the architecture of the above communications processor.

**DESIGN REQUIREMENTS**

**Functionality:** Protocol Adaption, Concentration, Routing

In its role as a protocol adapter, multiple protocols have to be supported on the user side and at least one protocol has to be supported on the network side. In the current implementation the user protocol is an IBM 6-bit BCD (or PARS) and the network protocol is Arinc Network Link Protocol (ANLP) which is a variant of ADCCP. When the PAC is located between a terminal subsystem and the network it will function as a Host emulator; and when the PAC is located between a host and the network the PAC will emulate the remote terminal subsystem, as illustrated in Fig 1.

Due to the large number of terminals used in an airline reservation system, the ability to concentrate multiple terminals is important. Although, a line can support up to 30 cluster controllers (multidropped) and a cluster controller can address up to 44 terminals, in practical situations only a fraction of the maximum capacity of terminals is physically supported by a line. As a concentrator the PAC can be located near a high concentration of terminals reducing the need for MULTIPLE long haul communication facilities.

The concentration of data and the routing of packets within a PAC will be controlled by configuration tables which may be updated in service by operator commands from the network control center. These tables will be created from a high level definition of the connectivity of the terminal-host subsystem.

**Requirements for a Controllable Network Component:**

Technical control of the network is effected from a Network Control Center which provides status reporting, trouble alarms annunciation, network configuration and control command services. As an integral element of the ADNS network the PAC will have the capability for alarms and command processing and will provide billing and other line/terminal statistics.

**Performance in speed and I/O support**

The minimum I/O requirements for a communications processor for use in
the airline reservation environment have been derived from analysing typical operating environments and found to be as follows:

- a minimum of 16 I/O ports
- two ports have to be V.35 compatible to support a data rate of 56 Kbits/sec for linking to the network side.
- other 14 ports must be RS232 compatible and be able to support a data rate of 9.6 Kbits/sec with about 75% utilization.

Requirements for Fault Tolerance

Since the PAC is in the critical communications stream of the airlines, any downtime will have a severe economic and scheduling impact on the operations of the airlines. The nodal elements of the ADNS already have identically configured redundant machines and achieve a high MTBF. The PAC will also have to employ redundant techniques to have high availability.

System Expandability

A primary design requirement is that the system architecture must allow easy integration of additional user protocols as they are developed. Also, changes in the high level transport network interface must be easy to incorporate. To accommodate the increased processing load with additional protocols, a multiprocessor architecture supported by a standard bus structure such as the multibus will be employed. It should be possible to plug-in additional CPU boards with minimal impact on the existing operational system. Software design has to be based on highly modular software structure with standardized interfaces, making the hardware environment, specifically the multiple CPU boards, transparent to the application software.

HARDWARE ARCHITECTURE

The overall hardware architecture of a non-redundant PAC is shown in Fig 2. The primary interconnection backbone for components within the PAC is an 8-slot Multibus I (IEEE-796) standard card cage. The Multibus backplane contains parallel priority resolution hardware that can support up to eight Multibus masters. The principal processing component of the PAC is a 68000 processor-based CPU board. All serial communications circuits are controlled through an Intelligent Communications Controller (ICC) board. The ICC board functions as a slave board on the multibus; although the design allows the use of multiple ICC boards only one board per system will be initially used. External serial device interfaces to the ICC board are provided by 'paddle cards', each containing a single 8530 dual USART.

![Fig 2: Hardware Architecture of the PAC](image)

The ICC board supports a proprietary bus implemented from a VME-style card cage and can control up to eight paddle cards providing a maximum of sixteen serial ports. Furthermore, in a redundant configuration, a custom redundancy card selects one of two identical multibus/ICC board subsystems to drive the 'VME' bus supporting the paddle cards. The following sections describe in more detail the salient hardware features of the PAC.

Host Processor Subsystem

Each CPU board contains a Motorola 68000 processor and either one or two megabytes of dynamic RAM operating...
under a 10 MHz clock. Access to the on-board memory and the memory of all other boards on the multibus is controlled by a Memory Management Unit (MMU). When part of the on-board dualported RAM is configured for read/write access by other multibus devices, the high 256-byte addresses are automatically allocated for the 'mailbox' facility. When a multibus master performs a write to the mailbox address space a mailbox interrupt is generated and the written value stored in RAM. This feature provides the mechanism for implementing interprocessor communication among the host CPUs. In addition, the MMU allows mapping of part of the virtual address space (processor output on the address lines to the MMU) into multibus address space.

The CPU board contains four I/O ports, two serial and two parallel. One of the serial ports is used for a local console interface and the other is used for a processor to processor link in a redundant configuration. The parallel ports are also used in the redundant configuration for control and monitoring of the redundancy hardware. When multiple CPU boards are used, the I/O ports on the additional boards are not used. Five independently programmable timers are available which can be used for hardware refresh timing of the dynamic RAM, to provide baud rate clocks to the I/O hardware and to generate clock interrupts for the operating system.

Intelligent Communications Controller (ICC)

The ICC board is a novel and unique component of the PAC architecture. The functions provided by the ICC board are implemented around two processors, a Motorola 68010 and a Signetics 8X305 with application independent on-board firmware. Furthermore, a triple ported 512 Kbyte dynamic RAM, that allows read and write access to the 68010, the 8X305 and any master device on the multibus serves as an elegant medium for inter-processor communications and data transfers. This offers a high-level interface for the application layers that reside on multibus host processors, to perform I/O operations on the synchronous/asynchronous ports supported by the paddle cards. As described below and shown in Fig 3, within the ICC board, the processors perform entirely different but complementary functions.

8X305 Functions

The 8X305 processor essentially performs the functions of an intelligent Direct Memory Access (DMA) controller servicing the character input/output to and from the 8530 multi protocol serial communication chips. The 8X305 is architecturally a unique processor, providing features that allow a system designer to implement application specific processing capabilities directly in hardware. This is achieved by using a concept called 'extended microcode', whereby an additional 16 bits of extended information are obtained by the processor with every instruction access, effectively widening the instruction length to 32 bits. These extended 16 bits are decoded and executed by external hardware specially designed to enable fast execution of communications related functions.

The following hardware components are used with the extended microcode to provide extra processing power to the 8X305.

PAGE REGISTERS:
Two five bit wide registers are used to setup addresses of one of possible 32 I/O units.

SCRATCH RAM:
The local RAM for the 8X305 is organized as 32 pages of 64 bytes. The page register selects the relevant page for the current I/O unit and the least 6 bits of the extended microcode selects the byte within the page.

AUTO INCREMENT LOGIC:
The nineteen address bits required to address the 512 Kbyte resource memory is accomplished within the 8-bit 8X305 with this group of logic components. It includes 32 pages of 8 24-bit registers, a 24 bit counter, a pair of 24-bit registers and a programmable logic sequencer (PLS) that executes macro operations as requested by the 8X305.

COMPTROLLER:
A PLS receives high level commands from the 8X305 and performs the I/O operations on the 8530a more efficiently.

11A.2.4.
TASK FIFOs:
Two 32-byte deep FIFOs enable the 8X305 processor to aid in 'task' scheduling. The firmware is grouped into critical and non-critical modules, which are queued by appropriate interrupt processing routines. Context switching time and interrupt latency are minimized using this technique.

68010 Functions
The main function of the 68010 processor is to manage the command and data pipeline between the host processors on the multibus and the 8X305 processor. I/O initiation commands and port configuration commands from the host processors and command completion commands from the 8X305 are queued in the 64-byte deep hardware FIFO located at the base address of the resource memory. These entries generate interrupts to the 68010 as they become the next entry to be serviced. The acknowledgements back to the multibus masters are queued in resource RAM in a FIFO setup implemented as a 1 Kbyte circular buffer. Fig 4 illustrates the command sequence flow among the three processors during the READ and WRITE operations.

Paddle Cards
Each paddle card drives two line interface blocks. As the paddle cards are not redundant and are single point of failure components, the circuitry on the cards is held to a minimum. In addition to the 8530 USART, the board contains TTL to line drivers and jumpers that can be configured for various synchronous/asynchronous protocol requirements.

11A.2.5.
For standard bit-synchronous protocols, the 8530 is capable of performing part of the link level functions such as Cyclic Redundancy Check (CRC) character generation/check and handling the flag byte (01111110). However, for the 6-bit, byte synchronous protocol PARS, the 8530, although capable of handling the sync1 sync2 characters (12 bits: 3F 3E) cannot generate or check the 6-bit CRC character which uses the generator polynomial $X^6 + X^5 + 1$. The CRC handling for transmit and receive has been implemented as part of the 81305 firmware with minimal penalty on execution time (overhead approximately 1 microsec per character).

There are two different paddle card types available for use with the PAC. An Rs232 version will support line speeds up to 19.2 Kbits/sec and a V.35 version that will support speeds up to 56 Kbits/sec.

Redundant Organization

The redundant PAC is made up of two 8-slot multibus-I card cages, with a separate battery backed power supply feeding each cage. The redundancy control card selects which of the host-ICC subsystem is to control the paddle cards. The first host CPU in each multibus cage is also connected to the redundancy card through a parallel port. A watchdog timer, which must be triggered every 100 mS, provides the basic mechanism for switch over. Health monitor software running on the host processors continually checks for software and/or hardware failures and controls the watchdog triggering activity. The redundancy board, on detecting watchdog trigger failure will switch the paddle card control to the other unit provided this subsystem is healthy.
SOFTWARE ARCHITECTURE

Overall Structure

A high level representation of the PAC software structure is shown in Fig 5. The software components of the PAC fall into three major categories: Application Protocol Virtual Machines which constitute the environment independent component of the software; Virtual Operating System (VOS) which provides interface to the hardware and operating system dependent services; and totally environment dependent components which include the operating system.

The allocation of software functions to physical hardware is shown in Fig 6. Multiprocessing capability is implemented as tightly coupled multiple uniprocessors. Each CPU board contains a private copy of an operating system that is responsible for controlling all on-board processing. The exchange of units of work between application modules, on-board and off-board, is performed by the Virtual Message Exchange (VMX) portion of the VOS services. An application is notified of a unit of work (event) through event flags and message buffers. When this notification is across CPU boards, a hardware mailbox is used prior to the application notification. A request to pass a unit of work contains the identification (ID) of the destination task. This information and the configuration tables are used by VMX to determine whether the request comes from an on-board or an off-board task.

Real-Time Multitasking Executive

The operating system (OS) used in the host CPU boards is an event driven multitasking executive. Each CPU board operates under the control of its own OS. Various services related to task management, timer processing, event flags, message buffers, mailboxes and memory management are provided by the operating system. Although, these typical services provided by the off-the-shelf OS are adequate for developing real-time embedded systems, additional communication specific services had to be added to aid application development.

![Diagram of Software Layers in the PAC Architecture](image-url)

Virtual Operating System

The Virtual Operating System (VOS) enhances the portability of the communications application software by providing a host independent set of functions not normally found in higher level languages, but very necessary to communications control. To this end it contains all of the environment specific, non-portable code and uses system capabilities when they are provided and emulating them if they are not. The services provided by VOS fall into three different categories as detailed below:

(a) Memory Management

In the multiple processor architecture, the VOS memory management function maintains memory pools within global memory donated by the various processor boards. Tasks can use the VOS services to acquire and release buffers dynamically thereby optimizing the use of memory. The free memory area within the 68000 CPU boards are maintained as a single memory pool and the unused memory in the ICC board is maintained as a
(b) Operating System Services

A set of VOS services maps generalized operating system service requests to MTOS-specific service calls. This feature provides operating system independence to the application software and also allows communications-related enhancements to be added as system services. For example, communication protocols typically involve several timers, and a set of comprehensive timer service routines have been implemented and incorporated as VOS services greatly simplifying application development.

(c) External Device Interfaces

The most difficult interface to define between the communications application and the hardware/OS environment are the characteristics of each device line which can be manipulated and the mechanism for I/O requests. VOS addresses this issue by directing the operation of the ICC board and controlling access to MTOS services. Each protocol operates transparent to the existence of the ICC board and operating system, with the actual interface buried in the lower level set of VOS functions.

In actual implementation, the architecture of the ICC board dictates a need for two separate pieces of device driver software to accommodate the presence of several 68000 processor boards. The ICC board functions as a slave on the multibus and is only capable of asserting two multibus interrupts to signal when an I/O request has been completed. Within the multibus environment only one master processor can be configured to acknowledge a particular ICC interrupt. Obviously, the interrupt mechanism alone does not provide a sufficiently discrete mechanism for the ICC to signal completion when more than one master processor can initiate an I/O request.

To overcome this limitation, the identity of the processor (and the VM) which initiates the request is
Included in the packet of information passed to the ICC board when the request is submitted. This information is later made available to the master processor during interrupt servicing. Only the first processor installed in the PAC is then jumpered to respond to those interrupts asserted by the ICC; the interrupt service routine on that board will examine the identification information in the data packet for the completed request and pass the information to the initiating board. The mailbox interrupt facility of the CPU boards provides the mechanism by which the main master will be able to pass data on to every other CPU board in the system.

**Application Layer**

Application software is being developed in the programming language 'C'. The portability of the software is further enhanced by the use of VOS which by hiding hardware and OS specific entities within itself has made application software independent of the execution environment. Fundamental to application software design is the concept of 'virtual machine' as described in the following sections.

**Virtual Machines**

A Virtual Machine is defined as set of tasks collectively performing a major function. The tasks which make up a VM are unique entities to MTOS, but share access to common data and control information (event flags etc.). Tasks which constitute each VM are under the control of a single MTOS environment on the same CPU board, but are coded with no intrinsic knowledge of tasks which belong to other VMs.

**Supervisory and Administrative VMs**

To provide appropriate data visibility for supervision of the network, each virtual machine will need to provide regular information regarding its status as an element of the system. This leads to the need for supervisory VMs which exist primarily to organize system status data and perform overhead functions needed by other VMs, without directly managing user traffic. ATTS implementation has only one such virtual machine, the Network Services VM (NSVM).

NSVM is responsible for monitoring the status of all hardware and software components within the PAC, reporting this information back to the Network Control Station (NCS), and managing the execution of commands and data requests from the NCS. During normal system operation, NSVM also controls the local diagnostic console.

**Protocol Application VMs**

Each user protocol to be managed within a particular PAC serving ATTS users will be implemented as a distinct virtual machine. A protocol VM is responsible for managing one or more user lines (through the ICC board and the paddle card) according to the specification of the protocol, and any line configuration parameters established during node generation at the NCS. The message data is extracted from each line, converted as necessary into appropriate transmission format, and handed off to the network transport protocol for delivery. Conversely, any data received from the network is packaged as necessary to fit the conventions of the protocol, queued for delivery to the proper line or station, and transmitted to the user line.

As seen in Fig 6, in the current implementation of the ATTS system, there are two protocol application VMs:

- **PARS**: This code segment services the reservation terminals when in host emulation mode, or services the reservation computer links if it is in terminal emulation mode. It is also possible for a single PAC to have both terminal emulator and host emulator software. Fourteen I/O ports are currently available for use by the PARS code.

- **ANLP**: The Arinc Network Link Protocol services the link between the PAC and an APS of the ADNS backbone network. Dual 56 Kbits/sec lines are expected to be supported by this protocol software.

**Virtual Message Exchange (VMX)**

The Virtual Machine Exchange (VMX) provides the mechanism for inter-PAC transfer of data between application VMs. Every VM is given a VM identifier
(VMID). In addition, all possible inter-VM data/control paths (logical connectivity) are labelled with a Logical Channel Number (LCN) (see Fig 7) which are defined at configuration time. Data structures defined as Inter VM Control Blocks (IVCBs) are created by the source VM and are forwarded to the VMX via VOS utilities. The VMX uses the source VM identifier and the LCN from the IVCB and the routing tables in the VMX to identify the destination VM. It then notifies the destination VM of the availability of an IVCB using the message buffer services of the OS that sets an event flag owned by the VM.

![Diagram of virtual machines and VMX interfaces]

CONCLUSIONS

The communications processor architecture outlined in the above paper is in the final phase of development. Easy expandability has been achieved by imposing a highly modular structure on hardware and software designs. Furthermore, the choice of a standard bus arrangement chosen for the host processor provides easy migration to other more powerful CPU boards, to a 32-bit CPU based host processor for example, if increased processing capacity is required in the future. The concept of virtual machines and the architecture that allows incremental addition of CPU boards also have provided a framework for implementing multiple related applications within a single PAC.

REFERENCES


