NASA Technical Memorandum 86740

Design and Initial Application of the Extended Aircraft Interrogation and Display System

Multiprocessing Ground Support Equipment for Digital Flight Systems

Richard D. Glover

January 1987

National Aeronautics and
Space Administration

NASA Technical Memorandum 86740

Design and Initial Application of the Extended Aircraft Interrogation and Display System

Multiprocessing Ground Support Equipment for Digital Flight Systems

Richard D. Glover
Ames Research Center, Dryden Flight Research Facility, Edwards, California

1987

National Aeronautics and
Space Administration
Ames Research Center
Dryden Flight Research Facility
Edwards, California 93523-5000

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SCREEN DISPLAYS

SUMMARY

A pipelined, multiprocessor, general-purpose ground support equipment for digital flight systems has been developed and placed in service at the NASA Ames Research Center's Dryden Flight Research Facility. The design is an outgrowth of the earlier aircraft interrogation and display system (AIDS) used in support of several research projects to provide engineering-units display of internal control system parameters during development and qualification testing activities. The new system, incorporating multiple 16-bit processors, is called extended AIDS (XAIDS) and is now supporting the X-29A forward-swept-wing aircraft project. This report describes the design and mechanization of XAIDS and shows the steps whereby a typical user may take advantage of its high throughput and flexible features.

INTRODUCTION

Several years ago, the NASA Ames Research Center's Dryden Flight Research Facility (Ames-Dryden) undertook the development of a microprocessor-based, user-programmable, general-purpose ground support equipment (GSE) called the aircraft interrogation and display system (AIDS) (ref. 1). This development was spurred by the need for a research tool to support digital flight systems integration, software verification and validation, pre- and postflight testing, and system maintenance. Prior to that time, special-purpose GSE was procured for each project, resulting in a multiplicity of different types of equipment having various capabilities.

The original AIDS design was an 8-bit unit used to support F-8 digital fly-by-wire (F-8 DFBW) flight software verification and validation (V/V). This was followed by a second unit built to support highly maneuverable aircraft technology (HiMAT) flight control computer testing and systems integration. Success with these early units led to the construction of more units for application to other projects (ref. 2). Although users universally praised the utility of the AIDS, many suggestions for improving system capabilities were offered, leading ultimately to the development of an extended AIDS (XAIDS). The overall design objectives for the new system forced this development effort to take an entirely new approach, which produced a system mechanization believed to be the first of its kind. In addition, an auxiliary system called the remote data acquisition subsystem (RDAS) was designed to permit the user to extend the capability of a basic XAIDS as the application may require.

The XAIDS and RDAS are both designed to be high-bandwidth real-time systems and can support both open- and closed-loop applications. As an engineering-units data display system, the user taps into the target system at the desired data flow point, and the XAIDS passively acquires data and drives operator-specified displays. Another open-loop application could be as an excitation generator to support response testing of the target system. Closed-loop applications include real-time simulation support with interfaces to either an iron bird or an actual aircraft. Such simulations could, for example, generate faulty dynamic sensor signals to allow testing of redundancy management in multichannel systems. Another area of possible closed-loop application is the interactive control of other GSE, permitting centralization of control to a single workstation.

This report will discuss first the generic XAIDS system hardware and software, pointing out features intended to simplify user interfaces. Then the X-29A forward-swept-wing aircraft testing application will be discussed from the standpoint of the user's dual responsibility for engineering compatible add-on hardware and developing necessary software extensions. Examples of typical operator displays will be shown, and an overview of basic procedures will be presented. A description of the prototype RDAS will be presented in appendix A.

NOMENCLATURE

ADC	analog-to-digital converter
AIDS	aircraft interrogation and display system
ARINC	Aeronautical Radio Incorporated
BIOS	basic input-output system
CENPRO	central processor
CLP	control law processor
CPU	central processing unit
CRT	cathode ray tube display
DAC	digital-to-analog converter
DK	diskette
EEPROM	electrically erasable programmable read-only memory
FF	free form
GSE	ground support equipment
HDLC	high-level data link control
IEEE	Institute of Electrical and Electronic Engineers
INTR	interrupt pushbutton
I/O	input-output
IOP	input-output processor
ISO	International Standards Organization (U.N.)
LAN	local area network
LED	light-emitting diode

MAINT	maintenance processor
MDS	microcomputer development system
MP	make page
NDP	numeric data processor
NMI	nonmaskable interrupt
PERPRO	peripheral processor
PROM	programmable read-only memory
PTCH	patch manager
RAM	random access memory
RDAS	remote data acquisition subsystem
RETMA	Radio-Electronics-Television Manufacturers Association
RTPRO	real-time processor
SCP	status and control panel
SDLC	synchronous data link control
SEU	system evaluation unit
TDU	time and date unit
UDI	universal development interface
USART	universal synchronous-asynchronous receiver-transmitter
V/V	verification and validation
XAIDS	extended aircraft interrogation and display system

DESIGN OBJECTIVES

The fundamental objective of the AIDS-XAIDS family of general-purpose GSE was to provide an aircraft control system research tool that could support a variety of projects with minimum reconfiguration of hardware and software. The approach taken was to provide a baseline host system to which could be added application-dependent hardware and software to meet specific needs. The success of this approach hinged on the baseline hardware providing a minimum of constraints on hardware additions and the baseline software providing a user-friendly environment within which software extensions could operate. These goals were not well met in the AIDS mechanization, and the second-generation XAIDS effort was aimed at broad improvements in both hardware and software flexibility.

In general, the design objectives for the XAIDS closely paralleled those for the AIDS. These included mobility, flexible input-output (I/O), a common core of generic support software, user-oriented displays, commercial components used wherever possible, and integral maintenance support. Several years of operational experience with AIDS showed that numerous major improvements were desirable, leading to the formulation of the following XAIDS design objectives:

User hardware extensions
- —More cardcage slots available for user boards
- —Multiple bus master capability
- —Both 8- and 16-bit data bus operations allowed
- —Larger memory and I/O mapping space
- —More user interrupt lines available
- —I/O extension via local area network

User software extensions
- —Simplified interfaces to baseline software
- —Structured extensions to ease customizing
- —Larger support libraries
- —Resident software development capability
- —High-order languages available

Baseline environment
- —Improved operating system
- —Higher system throughput
- —Symbolic parameter referencing
- —Mass data storage and retrieval
- —Faster printer with upper- and lowercase
- —Dedicated analog recorder digital-to-analog converters (DACs)
- —Operator control of screen refresh rates
- —Status and control panel for process management
- —Automatic date and time tagging

Maintenance and troubleshooting
- —Full-time operating system debugger
- —Full-time maintenance monitor

SUMMARY OF REQUIREMENTS

It was decided at the outset of the XAIDS effort that the listed design objectives could best be met by a distributed system having multiple processors permitting pipelining of concurrent operations. The choice of system bus was somewhat arbitrary since several bus architectures would have served, but the Institute of Electrical and Electronic Engineers (IEEE) Standard 796 Bus Specification was chosen since it offered the widest choice of compatible hardware and software components on the open market. An analysis of the throughput of such a system showed that 16-bit processors would be adequate, assuming that hardware floating-point processing were included. The choice of the 8086 microprocessor with the companion 8087 numeric data processor (NDP) gave the desired throughput. The remaining objectives were easily met within this basic environment.

One important carryover from AIDS was its central processor board, for which a considerable amount of operator I/O software had been developed. This 8-bit board, with the AIDS operator I/O software placed in programmable read-only memory (PROM), was to become the intelligent I/O channel for XAIDS operator peripherals. The final detailed requirements are summarized as follows:

IEEE-796 system bus
- —20-bit memory mapping (1 megabyte)
- —16-bit I/O mapping
- —16-bit-wide data path
- —Parallel priority resolution
- —Eight bus interrupts
- —21-slot cardcage; 3/4-in spacing
- —System reset pushbutton
- —Pushbutton for interrupt INTO/
- —Switching power supply (45 A at 5 V)

Multiple processors
- —Central processor (8086 plus 8087 NDP)
- —Real-time processor (8086 plus 8087 NDP)
- —Peripheral processor (8085); carryover from AIDS
- —Maintenance processor (8086); no NDP required

Peripheral I/O devices
- —19-megabyte hard disk drive
- —Four double-density floppy drives (8 in)
- —19,200-baud operator terminal; 24 lines by 80 columns
- —Line printer; upper- and lowercase; paper width up to 15 in
- —16 DACs dedicated to analog recorder outputs
- —Status and control panel (SCP)
  - —Eight light-emitting diode (LED) indicators
  - —Eight debounced high-low switches
  - —00 to FF hexadecimal thumbwheels
  - —Manual interrupt pushbutton

Local area network (LAN) controller
- —International Standards Organization (ISO) X.25 high-level data link control (HDLC) and synchronous data link control (SDLC) protocols
- —1-megabit/sec data rate
- —Phase-locked loop clock recovery
- —Modem capable of driving 5000 ft of bus cable

Operating system
- —Complete 8086-based multitasking system
- —Custom device drivers for printer and operator terminal
- —Utilities for boot loading satellite processors
  - —IEEE-796 bus slave loader
  - —LAN remote slave loader
- —Full-time debugger
- —Must support software development tools

Time and date unit (TDU)
- —Julian calendar
- —Self-contained rechargeable battery
- —Initializes operating system at power-up
- —Peripheral processor (PERPRO) interface for data time tagging
- —Utility program for resetting time and date

Resident software development support
- —Text editor
- —Languages: Assembly, FORTRAN, Pascal, PL/M
- —Utilities for linking, locating, and library functions

SYSTEM HARDWARE OVERVIEW

Figure 1 shows the general appearance of the two-bay XAIDS console. It is mounted on wheels for mobility and requires 120-V single-phase power from a standard wall outlet. Power consumption is approximately 1200 W, and numerous blowers within the console provide ample cooling even in non-air-conditioned hangar and ramp areas.

Figure 2 shows an overview of the XAIDS bus architecture; each solid-outline box represents a board plugged into the 21-slot large cardcage carrying the system bus. Four processors form the baseline complement: the central processor (CENPRO), the real-time processor (RTPRO), the peripheral processor (PERPRO), and the maintenance processor (MAINT). Mass storage is provided by a 19-megabyte hard disk drive plus four floppy diskette drives each providing 0.5 megabyte on single-sided double-density diskettes. A local area network (LAN) controller provides a 1-MHz serial addressable interface using synchronous data link control (SDLC) protocol. This LAN is currently configured to interface with the prototype RDAS described in appendix A. The prototype RDAS bears address 01; additional RDAS units may be added as user requirements dictate. A time and date unit (TDU) board provides clock and calendar readouts to the operating system and contains batteries to sustain time-keeping even when the XAIDS is powered down. The line printer and operator terminal are permanent peripherals and provide the basic operator interfaces.

An additional permanent feature of the baseline XAIDS is the status and control panel (SCP), shown in figure 3. This unit is interfaced to the RTPRO and provides a general-purpose operator interface. Eight light-emitting diode (LED) indicators are provided for visual readout; each LED has a monitoring jack suitable for oscilloscope monitoring of its drive signal. Eight switches with debounced contacts are provided for generating high-low input discretes, and two hexadecimal thumbwheels provide another 8-bit input register. The interrupt (INTR) pushbutton is debounced and generates one RTPRO interrupt each time it is actuated. Since the thumbwheels are not debounced, it is usually desirable to use the INTR pushbutton to generate a software snapshot of the setting of the thumbwheels.

The analog recorder outputs from the XAIDS are driven by 12-bit digital-to-analog converter (DAC) units on RTPRO. There are 16 channels of -5 to +5 V accessed through two connectors on the XAIDS I/O panel, shown in figure 4. This panel, located at the lower right rear of the console, provides the mounting for the LAN interface connector.

Table 1 shows the board complement for the baseline XAIDS. The CENPRO serves as the host for the RMX86 operating system, which resides partly in PROM and partly in random access memory (RAM). The two auxiliary memory boards (the auxiliary RAM board and the auxiliary PROM board) serve to augment, respectively, the RAM and PROM found on CENPRO proper. The nine empty slots are reserved for user-selected I/O channel boards that provide the hardware interface to the user's target system. The RTPRO and the LAN channel board both require two slots because of the aboveboard height of the piggyback modules they carry. Appendix B details the assignment of interrupts throughout the system, including those on the bus and on RTPRO reserved for the user.

FUNCTIONAL DESCRIPTION

At system power-on, the XAIDS begins an automatic bootstrap load operation that ends with the operating system (described later) in control of CENPRO. The operating system is configured for a single resident user (assigned number 1) who is given access to the operating system through the human interface by means of the operator terminal. This allows the XAIDS operator to access all the resources of the operating system, including editors, language processors, and software development utilities, by entering commands on the terminal keyboard. It also allows the operator to load and execute custom utilities and other jobs (called I/O jobs) that have been created (by NASA) to make the hardware perform desired user functions. One of these custom I/O jobs is the XAIDS executive, which has been previously configured to service a particular XAIDS application (although a non-user-specific "default" version also exists for test purposes). It is loaded by entering the command XAIDS, at which time it takes control of the hardware and remains in control until the operator forces an exit back to the RMX86 operating system.

During the time the XAIDS I/O job is running, the system takes on the character of the pipelined multiprocessor architecture shown in figure 5. The arrows show the direction of information flow during a typical data acquisition and display cycle. Data flow in general from left to right in the figure, with the vertical dashed lines demarcating the five stages of the pipeline. The symbol table shown at the bottom of the figure is the one element of the system that is not directly part of the pipeline but is rather a data base for XAIDS supervisory software.

The I/O channel boards shown at the left of the figure constitute the first stage of the pipeline. The TDU and LAN are considered baseline system channels. The system, when configured for a specific user, will normally have one or more user I/O channel boards as well. The essential function of this I/O complement is to interface external equipment to the IEEE-796 bus, with the one exception of the TDU, which is self-contained. The I/O channels may fall into one of three categories: nonintelligent slaves (such as the TDU), intelligent channels executing onboard firmware (such as the LAN), and intelligent channels executing software that must first be loaded into onboard RAM. A bootstrap loader utility has been created that can load any channel having in firmware the required startup routine.

The next stage is RTPRO, which handles any and all time-critical functions associated with the user application. It is loaded using the bootstrap loader utility mentioned previously and is responsible for I/O channel management, raw data preprocessing, and analog recorder servicing, in addition to any user-supplied real-time

routines. The channel handlers provide a standard interface to the data acquisition executive and may be executed either polled or interrupt-driven. Notice that the TDU does not have a channel handler, because it is polled from CENPRO. The data acquisition executive is responsible for all data flow within RTPRO and services two separate structures, one for the analog recorder interface (high speed) and the other for the operator display (on demand). It is supported by a set of raw data type handlers that perform parameter type conversion to NDP-compatible formats. Parameter mapping and type conversion specifications are fetched from the symbol table as required.

The third pipeline stage is CENPRO, which under the XAIDS I/O job is given the main task of interpreting and carrying out operator commands. This requires it to exercise control over both RTPRO and PERPRO while carrying out baseline tasks as well as servicing whatever user-defined commands are present. For data acquisition and display, the operator will normally load a previously created display page from floppy diskette. The display page handler will then perform page setup: fetch information as required from the symbol table, create a display control structure, set up the RTPRO data acquisition executive, and initialize PERPRO to drive the display. It then serves as the intermediary between RTPRO and PERPRO, fetching data as required and formatting each parameter in turn using the display format handler routines. In addition, CENPRO has the ongoing task of periodically polling the TDU and sending updated time and date information to PERPRO.

The PERPRO, the fourth stage in the XAIDS pipeline, uses firmware to manage the digital peripherals of the fifth stage, namely, the operator keyboard, cathode ray tube (CRT) display, and the line printer. The PERPRO provides line-edited operator input strings from the keyboard in several formats, checking for error locally before sending the string on to CENPRO for interpreting. The PERPRO also drives the CRT in several output modes, including the cursor-vectored high-speed refresh mode normally used for data displays. It responds to operator keyboard command sequences to select the desired screen refresh period from 0.1 to 1.5 sec. It also responds to a special keyboard command to snapshot the screen buffer and send it to the line printer buffer for editing and dumping to printer. Both the screen displays and the printer dumps are tagged with the current date and time of day to the nearest second.

SOFTWARE OVERVIEW

The XAIDS software suite may be broken into three major categories. The largest body of software is the commercial operating system, which was purchased from the Intel Corporation and configured for the XAIDS architecture. The second category is the XAIDS baseline software complement (generic, user independent), which includes a mix of firmware (PROM-resident) and RAM-resident modules for the four system processors. The third category contains the user-dependent software modules, including two RAM-resident packages, one for the CENPRO and one for the RTPRO, plus whatever firmware or software is required to support the user I/O boards. All processors in the system (including user I/O boards) have PROMs containing firmware that begins executing when power is applied. The CENPRO and RTPRO (and usually the user I/O processors as well) are then automatically loaded with software from the hard disk drive.

Table 2 shows the mapping of the various XAIDS system elements within the memory space accessible through the IEEE-796 bus using 20-bit addressing. Three-fourths of the available 1 megabyte of memory domain are occupied by the CENPRO RAM, the auxiliary RAM board, and the RTPRO RAM in three contiguous 256K blocks. The remaining space is allocated as shown to the LAN controller, PERPRO RAM, user I/O, and auxiliary PROM. The uppermost 64K block is not used. Table 3 shows the mapping of system elements within the I/O address space, which is a separate 64K domain paralleling the memory-mapped domain. Notice that the TDU decodes only the lower 8 bits of the I/O address and thus is aliased into 256 separate 8-byte blocks within the domain. Such a mix of address decoding is acceptable only because none of the 16-bit I/O-mapped devices impinge on these multiple blocks.

The operating system software currently used in the XAIDS is the Intel iRMX86 release 5, to which has been added custom device drivers for the TDU, line printer, and operator terminal. The operating system is given exclusive access to the entire CENPRO RAM as well as the entire auxiliary RAM board. A portion of the total operating system is contained in PROMs on CENPRO proper, a portion is resident on the auxiliary PROM board, and the remainder is loaded from the hard disk drive into CENPRO and auxiliary RAM during the bootstrap load operation. The Intel-supplied bootstrap loader firmware was slightly modified to permit switch selection of either the normal hard disk boot mode or a backup bootstrap mode using one of the four floppy diskette drives.

The total operating system is a full-up configuration incorporating all the Intel-provided release 5 options, including the universal development interface (UDI). The UDI serves as the host for the software development tools available to the operator through the human interface command interpreter. Another permanent feature of the operating system is a full-time RMX86 debugger, which may be accessed while the system is executing without disturbing current job execution. To use the debugger, a separate terminal set for 9600 baud must be mated to the CENPRO connector on the system interface panel inside the left rear of the XAIDS cabinet. The debugger, activated with the single keystroke command <cntl D>, permits examination of the entire spectrum of operating system objects. This tool is invaluable for assessing system allocation of resources (especially memory) and is the only means whereby such system problems as task deadlock can be diagnosed. The debugger is deactivated by entering Q <return>.

When the operating system begins to execute following completion of the bootstrap load process, the human interface submits the log-on file (pathname :prog:r?logon), causing the operating system to perform the steps specified. This text file is created using the system text editor and can perform any system functions desired by the user. Currently this file is configured to first load the RDAS through the LAN using a system utility program called RBOOT. The final step is the loading of the XAIDS executive program (called XAIDS), which in turn handles the loading of user boards and RTPRO. This logon file thus completes the automatic bootstrap process, which begins when the XAIDS is powered up, and takes the user directly to the XAIDS executive command list interpreter.

The operator is given the freedom to return to the RMX86 human interface whenever desired by entering the XAIDS command EXIT. While running under the human interface, several XAIDS utilities are available to the operator: the RDAS loader (RBOOT), the slave loader (BOOT), and the TDU setup program (TCU). Program RBOOT

permits the operator to reload the RDAS as required. If the RDAS has previously been loaded, the RBOOT commands RDAS to dump its load and the new program is then loaded. The utility BOOT allows the operator to "manually" load a slave processor, rather than have the XAIDS executive do it. If the slave has already been loaded, BOOT aborts the load process with an appropriate error message. The INT0 pushbutton can be used to send an interrupt, forcing all slaves to purge previously loaded programs so that BOOT can be used to reload one or more of them. The utility TCU permits the operator to reset the time and date registers of the TDU as required to correct for drift or to permit daylight saving time or standard time adjustments. The operator may reload the XAIDS executive at any time by entering the human interface command XAIDS.

MAINTENANCE PROCESSOR SOFTWARE

The maintenance processor (MAINT) is a 16-bit single-board computer configured as a bus master that is designed to access the entire IEEE-796 bus including both the memory-mapped and the I/O-mapped domains. The current MAINT program is resident in onboard PROMs and provides most of the features usually found in a monitor program, including memory read-write and I/O read-write. A separate terminal, which is connected to the system interface panel inside the XAIDS rack (accessible through the left rear door), is required to use MAINT. This terminal may employ any baud rate in the range 110 to 19,200. When the XAIDS is powered up, the MAINT firmware enters a lockon loop, which senses one or two capital letter U keystrokes to determine the baud rate. The program then displays a command menu including an H (help) command, whereupon the operator may perform any desired system troubleshooting operations. Until the program is called upon to examine or alter system-mapped hardware or software, no bus operations are involved, and there is thus no loading of the system. A purely passive function performed by MAINT starting at power-up is the monitoring of the eight bus interrupts; when sensed, each causes a counter to be incremented. These eight counters may be displayed on the maintenance terminal whenever desired.

PERIPHERAL PROCESSOR SOFTWARE

The peripheral processor (PERPRO) is an 8-bit single-board computer configured as a bus slave that services the operator CRT and keyboard, the line printer, and the bus timeout light on the panel below the operator terminal. The PERPRO software is resident in onboard PROM and executes in three separate modes: reset, RMX86 I/O, and XAIDS. Communication with PERPRO is accomplished through PERPRO RAM control structures that are accessible to any bus master through the system bus (memory-mapped domain). When the system is powered up, PERPRO firmware begins executing immediately, performing an initialization sequence, entering reset mode, and displaying a sign-on message on the CRT. Whenever the human interface is running in CENPRO, the PERPRO switches to RMX86 mode, in which the CRT, keyboard, and line printer become I/O devices belonging to the operating system. If the XAIDS executive is loaded, the PERPRO mode switches to XAIDS, during which the firmware responds to numerous specialized commands for keyboard input sequences, screen display functions, and line printer operations (including screen snapshot) that are

not available in RMX86 mode. These special functions are tailored for high-speed support of the XAIDS page displays at refresh rates up to 10 Hz.

REAL-TIME PROCESSOR SOFTWARE

The real-time processor (RTPRO) is a 16-bit single-board computer configured as a bus master that directly services the analog recorder DAC outputs, the SCP, and an RS-232 serial port. The RTPRO software consists of two elements: a startup package resident in onboard PROM and a RAM-resident load module that is downloaded either by the BOOT utility or by the XAIDS executive. When the system is powered up, the firmware begins executing immediately and waits until a RAM-resident module download has been completed. At that time, the firmware passes control to the load module, which will remain in control unless the INT0 pushbutton forces a return to firmware, which permits a reload.

The RTPRO firmware, while waiting for download, performs an SCP test routine permitting the operator to perform confidence tests on the switches, thumbwheels, and INTR pushbutton. If the control switches are all LO, the LEDs are flashed on in slow succession; any control switches placed in the HI position turn on the corresponding LED. When the INTR pushbutton is pressed, the mode thumbwheels setting is logically ORed with the switches and the combination appears on the LEDs.

The RTPRO load module consists of a baseline set of routines linked to the user's real-time routines to form a single entity loaded into the lower portion of RTPRO RAM. The baseline routines provide the following services: LAN servicing, channel management, DAC and CRT data acquisition, and programmable function generation. Supervision of RTPRO by CENPRO is accomplished through control structures in a portion of RTPRO RAM that is reserved as a common data area. The interfaces to the user's routines provide a variety of options that can accomodate a broad spectrum of user requirements including interrupt linkages, timed polling, and background looping.

The LAN servicing provides pipeline management for all data flowing to and from the RDAS. At present, only a single secondary station (01) is tied to the bus, although up to 255 addressable stations may be serviced. As discussed in appendix A, the maximum servicing rate is approximately 96 Hz, although degradation can be expected with high RTPRO loading. During each cycle, all 48 discretes are passed (both ways), 16 analog-to-digital converter (ADC) values are fetched, and all 28 DACs are updated. The data are passed to and from buffers in RTPRO RAM; if a routine requires access to the data stream for either input or output, this is accomplished synchronously using handshake semaphores. In addition to analogs and discretes, serial character streams to and from the RDAS peripherals can be passed using packets attached to the LAN messages. These packets are handled as low-priority traffic to maximize the bandwidth of the analogs and discretes.

Channel management involves the declaration and subsequent supervision of the data acquisition channels, which may number up to 32. Each is identified by a two-character mnemonic and has associated with it a control structure as shown in table 4. The handler for each channel consists of a set of six subroutines whose addresses are provided when the channel is declared. The baseline RTPRO software declares four channels: PP for access to PERPRO RAM, RP for access to RTPRO RAM,

RA for access to RDAS analog inputs, and RD for access to RDAS discrete inputs. All channels, including any the user requires, are declared during RTPRO initialization following download. The baseline software provides routines for the timing, synchronization, and status monitoring of each channel.

The DAC and CRT data acquisition is managed using two large control structures that contain all the information necessary to fetch the data items called for and perform raw data preprocessing. Up to 16 recorder output DAC parameters may be handled, and up to 255 CRT display parameters may be handled. The DAC control structure is scanned at a maximum rate of 100 Hz, although rate degradation is normal depending on RTPRO loading. The CRT control structure is scanned at the screen refresh rate selected by the operator (0.67 Hz to 10.0 Hz). The control structures contain a block of the type shown in table 5 for each parameter. These blocks provide registers for data buffering, constants used in preprocessing, and pointers to the necessary supporting subroutines. As soon as the DAC control structure scan is complete, the DAC values are copied from the structure to the DAC buffer, and the 16 DAC outputs are immediately updated. For the CRT control structure, CENPRO is notified when scan is complete, and CENPRO then fetches from the structure whatever data elements are required to support its screen display formatting.

For both DAC and CRT data acquisition, the subroutines called to perform the data snapshot (copying data from the channel handler) and subsequent processing are embodied in software modules called raw data type handlers. Each data type is identified by a four-character mnemonic and has an associated declaration structure as shown in table 6. The two setup routines are used in the configuration of the CRT and DAC control structures. These routines are responsible for generating the constants embedded in each control block and also for supplying the pointers for the snapshot and processing routines. Up to 64 total raw data types may be declared at initialization time; currently the baseline software declares the 7 types shown in footnote a of table 6. The algorithms employed in the processing routines for these 7 data types are shown in appendix C.

The programmable function generator contained within the baseline RTPRO software permits up to 100 time-varying functions to be generated. The outputs are used to selectively drive RDAS analog and discrete outputs to provide forcing functions to a system under test. The generator is initialized by CENPRO and is clocked by eight separate software timers triggered by the operator using the SCP. Up to 32 function types may be declared at initialization time; currently 10 functions are declared by the baseline software. These are sinusoidal, step, ramp, doublet, pulse, rectangular periodic, triangular periodic, random discrete, shaped white noise, and exponential. The generator is controlled by a large data structure containing for each function a block such as that shown in table 7. When each function is declared, a pointer is supplied for a setup routine that is responsible for initializing the block constants and pointers. The structure is scanned at a maximum rate of 100 Hz, with slower rates to be expected for larger numbers of functions as RTPRO loading increases. However, the timers maintain a uniform time scale even though servicing rates may slow.

CENTRAL PROCESSOR XAIDS JOB SOFTWARE

The XAIDS executive load module consists of a baseline set of routines linked to the user's extension modules to form a single load-time locatable RMX86 I/O job.

This module is loaded by the logon file during system power-up initialization; it may be later reloaded manually using the operator human interface command XAIDS. At its core is a looping command interpreter that accepts command mnemonics of one to four characters in length and then branches to the corresponding servicing routine, called a command handler. The menu is the combination of the XAIDS baseline command set plus whatever user-defined commands have been declared. The XAIDS executive provides an interface for the overall software suite to the RMX86 operating system for such operations as file reading and writing. In addition to RMX86 I/O, the executive baseline manages the display scratch diskette, several data display formats, a symbol table, two display page handlers, a function generator, and several test modules.

All XAIDS commands with the exception of EXIT are declared by tables consisting of concatenated sets of command declaration blocks such as shown in table 8. A concatenated set of such blocks may be of any length and is delimited by a final block containing blanks in the mnemonic field. Associated with each declared command mnemonic are two ASCII text strings that are used in the menu presentation process. The first is a 20-character explanation of the mnemonic itself while the second is a 45-character description of its function. Also contained in each block is the entry address of the command handler, the address of the scratch file save template (if any), and the length of said template in bytes.

A template is defined as any data structure that is created by a command handler and that can be preserved on diskette and retrieved for later reuse. All templates contain the same initial block of 54 ASCII bytes, the first 4 bytes containing the mnemonic for the associated command handler and the next 50 bytes containing the identifying name for the template. Diskette drive :F3: is reserved for the scratch diskette, which may contain up to 100 numbered files each containing one template. An additional file on the diskette is the so-called scratch diskette directory (not to be confused with the diskette's operating system directory), structured as shown in table 9. The scratch diskette manager has routines for viewing the directory, loading a file, dumping of file groups to printer, deleting a file, copying a file to a backup scratch diskette, and initializing a new scratch diskette. When a fresh diskette is initialized to become an XAIDS scratch diskette, its newly created (empty) directory is given a name, and the date and time of creation is permanently recorded. This directory contains 100 entries for the status information corresponding to diskette files given pathnames 1 to 100. Each entry contains the empty-occupied status flag, the command mnemonic of the handler associated with the template, the date and time of the save operation, and the title for the template.

Two major PUBLIC routines within the scratch diskette manager handle the saving of templates and the subsequent reloading of templates. The routine that saves a template is invoked from within a command handler and either overwrites a specified file number (1 to 100) or writes to the first available empty file if 0 is the file number. For a typical save operation, the current scratch diskette directory is loaded, the elected file is written, the directory is updated, and the directory is then rewritten. The loading of a template (always specified by file number) invokes the routine that reads the directory from the scratch diskette, determines which handler is involved, retrieves the template, copies it to the handler's buffer, and invokes the handler's servicing routine.

Since the main uses of XAIDS in one way or another involve the display of data, a wide choice of screen display formats has been made available. The executive

manages a set of baseline and user-defined display formatting routines that are declared during the initialization of the executive immediately after load. The formats are declared by blocks such as that shown in table 10. Each format is identified by a single letter that has associated with it the address of the corresponding servicing routine. The eight format types declared by the baseline software are shown. With the exception of M, the letters assigned are identical to those used in FORTRAN format statements. The M (for "message") format causes the display of one of two alternative ASCII strings, each from 1 to 10 characters in length. This format is widely used for displaying the status of a single discrete or a set of discretes within a given parameter.

A central feature of the XAIDS is the use of symbolic parameter referencing through a large structure called the symbol table. This table occupies 64 kbytes in the uppermost portion of RTPRO RAM at bus memory domain address range 0B0000H to 0BFFFFH. The table is alphabetized and can hold a maximum of 510 parameter entries described by blocks such as that shown in table 11. This information is the data base used by both RTPRO and CENPRO in the management of data flow from the channel handlers to the ultimate destination, whether RTPRO DACs or CRT display. The symbol table editor is a routine that manages the symbol table; it can load a new table if required, insert, modify, delete, or clone individual entries in the currently loaded table, attach an identifying name to the table, and store the resulting table on diskette (usually using the default pathname :F3:SYMBOL.TAB). It also controls the printer listing of the table in one of two formats and generates scans of the table on the CRT.

Screen display of data is managed within CENPRO using a table of item blocks such as that shown in table 12. Each parameter to be displayed on the screen requires such a block; each has an index number identifying which element of the RTPRO CRT data acquisition structure is used as the data source. Also specified in each block is the display format, the address of the corresponding formatting routine, the position on the screen of the left edge of the data field, and a copy of the zero-one messages from the symbol table entry for the parameter. The zero-one messages are used only by the M format; they are copied from the symbol table to the structure to reduce bus traffic levels.

Two data display page handlers are provided by the baseline software. The simplest of these is a tabular form called a make page having the template structure shown in table 13. The make page handler displays a maximum of 20 data items, 1 per line, each specified by data source channel code, parameter name, and display format. The format for a line item is usually the default format read from the symbol table but may be a different format in effect for this line item only. The handler accepts whatever title is desired for the page template and can save the template on scratch diskette for later retrieval. Besides being used for data display, this page is used to set up the recorder analog outputs. A subroutine controls the mapping of the first 16 entries of any make page to the RTPRO DACs on a one-to-one basis whenever the page is "latched" to the recorder outputs. Even though the display is exited, the latched configuration will remain in effect until either unlatched or relatched.

The other baseline display page handler, called the free form, has the template structure shown in table 14. The handler for this style page permits unlimited freedom to create a display of whatever form desired occupying screen lines 3 to 23. The template contains an image of the static information or "background" to be

displayed plus data entry blocks for up to 255 entries. Notice that the data item entry structure is slightly different from that used for the make page template since each free form data item must be explicitly positioned. Normally the template would contain a mix of background plus data but may contain only background (thus generating a static display) or may contain only data items (without background). Data items are specified by channel code plus parameter name; the format for each data item is configurable, the title for the page can be whatever desired, and the template can be saved on scratch diskette. One unique feature of this handler is the means to list the data item blocks from the template on either CRT or line printer.

Control of the programmable RDAS function generator within the RTPRO baseline software (discussed previously) is exercised by a command handler having the template structure shown in table 15. Each template defines a "program" specifying up to 100 different functions capable of driving either an RDAS DAC or an RDAS discrete output "channel." Functions may individually drive a channel, may be summed to a channel, or may be time-division multiplexed to a channel. Each function entry specifies the desired output channel, the function name, the timer specification, and the three elapsed times at which the function is to be enabled, triggered, and disabled. The function name correlates to an entry in a function definition table, a tabular data structure loaded from diskette, which specifies the function type (for example, sine) and the arguments characterizing the function. The timer specification for each function determines which of eight timers (controlled by the SCP) is to be used as its time base. This handler can edit the program template, give it a title, save the program template on diskette, define new functions in the function definition table, and cause the current program template to initialize the RTPRO function generator control structure. Once the RTPRO generator is initialized to run mode, it is independent of CENPRO, and the handler may be exited.

The function definition table is an example of a category of data structures called ASCII editor tables. As the name implies, these tables contain only ASCII (that is, text) information and may be employed by any command handler requiring an auxiliary data base in addition to its template. These tables, like symbol tables, exist as operating system files (not numbered scratch files) and temporarily overlay the memory block normally occupied by the symbol table. To distinguish the different table types, all are tagged with the "owner" mnemonic. Therefore all symbol tables are tagged SYM, and a function definition table will always be tagged RDFN. For each type of table, there must be a default pathname permitting the automatic reload of the correct table type as required. The generic ASCII editor routine may be called by any command handler that "owns" an ASCII editor table to permit loading, viewing, listing, modifying, naming, and storing of its table.

The structure of an ASCII editor table entry is shown in table 16. The assignment of fields is determined by the handler owning a particular table. In the case of RDFN's function definition table, the entry name field contains the name of the function, information field 1 contains the function type mnemonic, and information fields 2 to 9 are assigned to various function arguments. Any application that requires a field to contain a numeric constant must encode it in ASCII string form when the entry is created and later decode it when the constant is fetched. This general-purpose structure is suitable for any auxiliary ASCII data base application of up to 510 total entries, provided each entry can be encoded into a name and description field of 32 characters plus 9 information fields of 10 characters each.

The baseline CENPRO software currently contains four test routines. An RDAS test routine generates two-way traffic to the RDAS unit through the LAN, including analog I/O, discretes I/O, and peripherals I/O. There is a routine used primarily in software V/V that permits the generation of numeric constants in various formats anywhere within the memory-mapped domain. Another routine writes to the analog recorder outputs using various periodic waveforms as well as individually controlled calibration test voltage levels. Finally, there is a general-purpose system monitor routine that is similar to the MAINT firmware described earlier except that it cannot tally the IEEE-796 bus interrupts.

XAIDS-USER INTERFACES OVERVIEW

The central philosophy adopted in the design of the XAIDS was to provide a flexible environment that would permit a prospective user to easily configure the system in ways that could not be anticipated. To make this possible, the interfaces of the baseline hardware and software to the user-supplied extensions have been structured so as to be limited, clearly defined, and easily understood. This approach has been so successful that the design of any XAIDS application is now a straightforward engineering task.

The interfaces that a prospective XAIDS user must deal with can be partitioned into the following categories: cabinet-mounted hardware, cardcage components, cabling, and software. As might be expected, the software area involves the most extensive interfaces and as such usually demands the largest part of the engineering effort. These four areas will be discussed here in generic terms with emphasis on the options open to a user.

The XAIDS cabinet hardware employs standard RETMA (Radio-Electronics-Television Manufacturers Association) compatible mounting rails and accepts standard 19-in-wide rack-mounted subsystems. Within the cabinet, three spaces have been reserved for the user: a 3.5-in-high panel space above the status and control panel in the upper left front bay, a 7.0-in-high panel space above the blower unit in the lower right front bay, and a 8.75-in-high panel space at the bottom of the left rear bay. The 3.5-in space above the SCP will accept a unit not exceeding 12 in. in depth; the 7-in space above the blower unit will accept a unit not exceeding 8 in. in depth. These two spaces may be utilized for user equipment or panels; spare power outlets are provided supplying up to 360 W total. Equipment that dissipates appreciable power should contain integral ventilation blowers. The rear panel space is reserved for user I/O connectors, which provide a mating interface for the cabling connected to the user's target system or systems.

The XAIDS cardcage is a 21-slot (0.75-in spacing) IEEE-796 compatible unit having integral parallel priority resolution. Auxiliary (P2) connectors are not installed, and if possible, the user should avoid selecting boards that require them. Cardcage backplane power for +5, +12, and -12 V is provided, and the user is permitted to draw up to 16 A from the +5-V feed. Nine slots are reserved for the user board complement, six of which are wired for parallel bus priority resolution and may thus contain bus masters. All boards must decode 20 address bits, and memory-mapped domain bus requirements must not total more than 96 kbytes. The I/O-mapped domain access may be 8 or more address bits, and a minimum of 198 I/O addresses are available. Three backplane interrupts are reserved for the user,

one of which is currently configured to be generated by RTPRO, while the other two are sensed by RTPRO. Within the stated guidelines, the user may select or build whatever board complement is required to provide target system interfaces.

The user must provide cabling within the XAIDS cabinet to connect the user's I/O connector panel to rack-mounted equipment (if any) as well as to cardcage boards. While there does not need to be an interface of such cabling to the cabinet, there does exist a convenient interconnect point that the user may employ if desired. The cardcage interface connector panel is mounted inside the cabinet behind the cardcage and has 8 empty mounting holes for 25-pin connectors. This is a convenient spot to transition from ribbon cable (coming from the cardcage) to the pigtails going to the user I/O connector panel.

User-supplied software may fall into one of four categories: RMX86 utilities, I/O board programming, RTPRO resident extensions, and XAIDS executive extensions. The user has the option of creating utility programs, which are loaded and executed through the RMX86 human interface by simply entering the name of the file containing the load-time locatable code. Such utilities would necessarily be stand-alone RMX86 I/O jobs totally independent of the XAIDS executive (which in itself is a stand-alone I/O job). An example of such a utility would be a file processor that analyzes data previously written while running under the XAIDS executive. Creating such utilities requires knowledge of RMX86 system calls and file management techniques. The advantage of a stand-alone utility is that it may be quite large and has unrestricted access to system resources. An alternative option is to make the utility a routine that runs under XAIDS and is accessed with a menu command. This has the advantage of convenience by being accessible without exiting XAIDS.

With regard to user I/O board programs, the type of board selected in each case determines what programming options exist. If an I/O board does not have a large dual-port RAM, its program must be PROM resident and cannot be bootstrapped. If the board has a large dual-port RAM and employs an 8086 or 8088 central processing unit chip, the user can bootstrap its program using the XAIDS utility BOOT. If this option is selected, the necessary cooperative firmware module checks the byte flag at onboard address 4, which if set causes an interrupt 0 to occur. The BOOT utility first loads the program into the board's RAM, places the starting address in the interrupt 0 vector location (addresses 0 to 3), and sets the flag. A user I/O board that does not employ either the 8086 or the 8088 can still be bootstrapped if the user is willing to write a custom loader utility that transfers code from a non-RMX86 diskette to the I/O board. In this case, the load module would need to be on 8-in diskette and would necessarily be generated on a user-supplied development system. In general, the advantage of bootstrapping an I/O board is that its program may be changed more easily. The advantage of PROM-resident I/O programs is that the lengthy bootstrapping process at system power-up is eliminated.

All baseline software used in the XAIDS is created using Intel iAPX 86/88 Family Utilities. The user-supplied software that links to the RTPRO and XAIDS executive baseline software must be compatible. The XAIDS system has resident software development support for PL/M, FORTRAN, Pascal, and assembly language, thus giving the user a choice of source language. The compiler for the high-order language "C" is also available but is not currently installed. All baseline software is written in PLM86 and compiled using LARGE model. When a user elects to use one of the other languages, the compilation model must be compatible with PLM86 LARGE, and the source code must be rigidly disciplined in subroutine parameter passing. The use of PLM86

LARGE is encouraged since it guarantees problem-free integration with the baseline code.

After linking with the user's extensions, the RTPRO baseline software expects to find the following routines declared PUBLIC: USER$INIT, USER$TIMER$0, and USER$BACKGROUND. The first is called only once during the RTPRO initialization sequence, the second is called every 2 msec, and the third is called at a variable rate that depends on RTPRO loading. All three of these routines have access to the full set of RTPRO support routines shown in appendix D. Some of these are subroutines, and others are typed procedures (functions) that return a value. All are written in PLM86; non-PLM86 calls to these routines are permitted, provided that proper parameter passing is observed. The user has access to a large block of RAM for common data area and is expected to design protocol for handshaking with user CENPRO routines where required.

The USER$INIT routine in RTPRO provides the user an opportunity to perform initialization steps related to load module identification, raw data type handlers, and channel handlers. The user must select a four-character ASCII identifier and write it into the PUBLIC variable RTPRO$LOAD$MODULE$ID. This allows the user's CENPRO software to later verify that the proper RTPRO load module has been booted. For each raw data type that falls outside the baseline set, the user must supply a raw data handler and declare it from USER$INIT. For each separate data input path (usually originating from an I/O board), the user must supply a channel handler, declare it, and decide what mode to employ in servicing it. The user has the option of declaring a channel from USER$INIT or delaying until later and declaring it from USER$BACKGROUND. Once declared, the servicing modes available are interrupt driven, timed polling, and background polling. If interrupt driven, the linkage must be set up and the interrupt level enabled.

The USER$TIMER$0 routine in RTPRO provides a means of performing timed polling of channel handlers or of performing other operations at user-selected intervals of 2-msec granularity. The USER$TIMER$0 routine is called with interrupts disabled from the timer 0 interrupt routine and is restricted to servicing that does not require the NDP. This routine should be as short as possible so as not to miss the next timer 0 interrupt and thus cause timing distortions.

The USER$BACKGROUND routine in RTPRO is called from the looping main RTPRO program at intervals that vary widely with RTPRO loading. It provides a means for the user to service low-priority tasks that are triggered by handshaking flags from the user CENPRO software. This could even include the declaration of user channel handlers, the vectoring of interrupts, and the enabling or disabling of interrupts. It would likely control channel handler status and guarantee that proper synchronization with baseline channel handlers was maintained.

After linking with the user's extensions, the XAIDS executive baseline software expects to find the following declared PUBLIC: an ASCII string called USER$NAME, two pointers called USERCMDLIST$PTR and USER$TEST$LIST$PTR, and five subroutines called USER$INIT, USER$CRT$REQUEST$ACTIVATE, USERCRTREQUEST$RESET, USER$DAC$REQUEST$ACTIVATE, and USERDACREQUEST$RESET. The USER$NAME string must be 16 bytes long, the 16th byte must be null, and the remaining bytes should contain the identifying name that will be placed in the upper left corner of all displays. The pointers provide the user a means to specify the location of the two tables of

commands to be placed at the top of the main menu and the test menu, respectively. The USER$INIT routine is called once when the XAIDS executive is initialized following load. The remaining four routines provide the user with the means to perform special processing if required at the beginning and end of display operation and at the latching and unlatching of recorder outputs. All user routines have access to the full set of CENPRO XAIDS operator interface support routines shown in appendix E. The names for these routines all begin with xq to distinguish them from RMX86 routines, which begin with rq.

The USER$INIT routine in XAIDS allows the user to perform steps related to bootstrap loading of RTPRO and the I/O channel boards, performing RTPRO handshaking setup, declaring special display formats, initializing command handlers, and loading the pointers specifying the location of the two command tables. When USER$INIT is called, it must first determine whether RTPRO is loaded with the proper load module. If not, the user purges the incorrect one (if required) and calls a bootstrap loader routine that is an embedded version of the utility BOOT. Like BOOT, OFFSET$LOADER requires two arguments: a pointer to the filename for the RTPRO load module and the offset in 16-byte segments of the RTPRO RAM (which is 8000H). If any I/O boards need booting, OFFSET$LOADER must be called for each. After successful I/O board boot, the RTPRO must be informed that the related channel handler or handlers may now be declared. If any special I/O board initialization following boot is required, that must be done here as well. If the user has any special display formats, they must be declared by calling a routine called DECLARE$FORMAT$TYPE with two arguments: the ASCII letter code and the address of the formatting routine associated with it. All user command handlers must be initialized, and their command declaration blocks in the appropriate table must be set up. The final step is to load the table pointers with the correct addresses.

XAIDS CONFIGURATION FOR X-29A PROJECT SUPPORT

The X-29A forward-swept-wing aircraft employs a triplex digital flight control system using dual computers in each channel. At an early date, NASA stated a requirement for an XAIDS unit that would be used by X-29A project personnel both in the simulation laboratory and in the hangar. It was intended to support control system testing during software V/V, simulation exercises, systems integration, and pre- and postflight testing. The desired level of support required three different kinds of interfaces: monitoring three telemetry streams simultaneously, controlling two pieces of GSE, and providing I/O tie-in to the Ames-Dryden simulation laboratory. This section will deal with the configuration of the X-29A project's XAIDS to meet these requirements.

Figure 6 shows an overview of the XAIDS interfaces that had to be engineered for the X-29A project. Each channel of the aircraft's triplex flight control system consists of two processors called the control law processor (CLP) and the I/O processor (IOP). The three telemetry streams are Aeronautical Radio Incorporated (ARINC) 429B format running at 100 kbits/sec, with each IOP sending 64 32-bit "words" every 25 msec. The GSE are called system evaluation units (SEUs), with two being required, one to access the CLPs and the other to access the IOPs. The interface to the SEUs is through full duplex RS-232C serial trunks, which communicate with the resident firmware internal to each SEU. An SEU can perform various CLP-IOP

system interface bus operations, such as selectively halting or running a processor, reading or modifying RAM, reprogramming electrically erasable programmable read-only memory (EEPROM) chips, recomputing checksums, and examining registers. The simulation laboratory interface required both analogs and discretes to be bidirectionally transmitted through the RDAS for closed-loop simulated fault testing.

Table 17 shows the I/O board complement required to support these interfaces. Each I/O board handles one telemetry stream through a piggyback ARINC 429B transceiver module that is designed to mount on one of its so-called I/O expansion bus connectors. Two of the three boards also have a second piggyback module that provides one RS-232C interface to supplement the one on the main board. The RDAS interface through the LAN is part of the baseline XAIDS configuration and as such did not require any user-supplied components. Figure 7 shows the final configuration of the XAIDS cardcage with the I/O board complement installed.

The cabling for the ARINC busses and RS-232C ports uses flat ribbon cables from the I/O boards as far as the system interface panel behind the cardcage. Figure 8 shows the location of the X-29A connectors mounted in the right half of the system interface panel with the mated pigtails connecting to the X-29A aircraft interface panel at the rear of the console. Figure 9 shows the layout chosen for the aircraft interface panel; the three ARINC busses share a single connector at the bottom.

The PL/M program written for the I/O boards was designed to permit the boards to operate as slaves to RTPRO rather than as bus masters. Each board has 32 kbytes of dual-port RAM, all of which is mapped to the IEEE-796 bus memory domain creating three contiguous blocks. Each board has the same small startup routine in PROM, and the same program (IOP.X29A) is loaded by the bootstrap loader into the three RAMs. This program contains servicing routines that respond to local interrupts from the ARINC transceiver and RS-232C modules and handle the resulting data flow solely under the direction of the RTPRO. Each board signals status to RTPRO through a combination of semaphores and state variables, and thus no bus interrupts are required.

The RTPRO software extensions required for the X-29A system include six channel handlers, six raw data type handlers, and one rather large background task called X29A$SIM, which locally computes anticipated control law gains. The status and control panel is not used, and no interrupts are serviced. The software was written using FORTRAN for the X29A$SIM package and PL/M for everything else. A large block of RTPRO RAM is devoted to hard-mapping the FORTRAN COMMON blocks for X29A$SIM so that CENPRO can display the common variables. The USER$INIT routine declares three channels (H1, H2, and H3), declares six raw data types (HSP, HDP, HFLT, HHEX, HINT, and HBOO), and loads the identifier X29A into RTPRO$LOAD$MODULE$NAME. The three channels access buffers within RTPRO RAM that contain control system data fetched through the SEUs. The six raw data type handlers provide conversion of the processor data types, single precision, double precision, floating point, hexadecimal, integer, and boolean, to NDP-compatible types. The algorithms employed in these data handlers are summarized in appendix C. The USER$BACKGROUND routine checks on I/O board status and waits until bootstrap load is complete before declaring the corresponding ARINC channel handler (A1, A2, or A3). It also invokes the execution of X29A$SIM when so directed by CENPRO and manages the status keeping for any ARINC channels that happen to be idle. The USER$TIMER$0 routine polls the three I/O boards (every 2 msec) to see if a fresh ARINC data buffer load is ready. As the active channels show ready, an elapsed timer is checked so that only data frames

coincident within an 8-msec window are accepted. If proper synchronization is sensed, it calls the routine named SYNCH, which in turn triggers the copying of whatever data are required from the I/O board buffers to the data acquisition control structures.

The CENPRO software extensions required for the X-29A application include the USER$INIT routine, seven main menu command handlers, two test menu command handlers, and a USERCRTREQUEST$ACTIVATE routine. No custom data display formats are declared. The USER$INIT routine loads the I/O boards with the file IOP.X29A, sets the default baud rate (2400) on the four SEU trunks, and loads the RTPRO with the file RTPRO.X29A. It then initializes the nine command handlers and sets up the command list pointers. The seven main menu commands are QT, RU, SEU1, SEU2, PTCH, PMSW, and GAIN. The two test menu commands are ARNC and BAUD.

OPERATION OF THE X-29A XAIDS

The XAIDS unit constructed for the X-29A project went into service in January 1984 and by October 1986 had accumulated over 3600 hr of operation in the Ames-Dryden real-time simulation laboratory. During this period the unit was used for diverse support including software V/V, closed-loop simulation testing, and pilot training. This section gives an overview of XAIDS operator procedures and presents sample printer hard copies of displays made during actual operations.

The XAIDS firmware and software load modules were configured to totally eliminate any need for operator interaction during bootstrap and initialization following power-up. The operator need only turn on the RDAS power switch, then turn on the XAIDS power switch, and wait approximately 1 min until the XAIDS main menu appears on the CRT screen. During this waiting period, several messages appear on the screen that give the operator the status of the ongoing bootstrap and initialization stages. The first message to appear is the RESET mode sign-on generated by PERPRO to signal its readiness to accept communications from CENPRO. Meanwhile the RMX86 operating system is being loaded, and approximately 30 sec later the RMX86 basic I/O system (BIOS) switches PERPRO to RMX86 mode, thus telling the operator that the operating system is running. Following this, the operating system submits the logon file for execution. The next message to appear is generated by the utility RBOOT as it performs the RDAS bootstrap; if RDAS is not connected or not powered up, RBOOT aborts with an appropriate message. The final stage of the initialization process is when the logon file invokes the command XAIDS, which begins the loading of the XAIDS executive.

The XAIDS executive loaded into CENPRO by the human interface is classified as an RMX86 "I/O job" and consists of the executive baseline linked to the X-29A extensions. When the load cycle is complete, the executive takes control of the system, switches PERPRO to XAIDS mode, and activates the operator's keyboard. From this point on the operator has the option of aborting the XAIDS executive initialization and returning to RMX86 mode by pressing the <esc> key. The executive next calls the X-29A USER$INIT routine, which controls the boot of the I/O boards and RTPRO. The following messages appear on the screen (in bottom to top scroll fashion) during the execution of USER$INIT:

```
SLAVE PROCESSOR LOADER ROUTINE
PATHNAME : :sd:user/1/iop.x29a
SEGMENT = C800
LHEADER : X29A_IOP_MAIN_P86
MODEND TYPE 3 : MAIN MODULE WITH START ADDRESS = 0040 : 0006
TOTALS :  IGNORED = 108    PEDATA = 82    PIDATA = 0


SETTING UP DEFAULT SEU #1 BAUD RATES
TERMINAL BAUD RATE = 2400        MODEM BAUD RATE = 2400


SLAVE PROCESSOR LOADER ROUTINE
PATHNAME : :sd:user/1/iop.x29a
SEGMENT = D000
LHEADER : X29A_IOP_MAIN_P86
MODEND TYPE 3 : MAIN MODULE WITH START ADDRESS = 0040 : 0006
TOTALS :  IGNORED = 108    PEDATA = 82    PIDATA = 0

SETTING UP DEFAULT SEU #2 BAUD RATES
TERMINAL BAUD RATE = 2400        MODEM BAUD RATE = 2400


SLAVE PROCESSOR LOADER ROUTINE
PATHNAME : :sd:user/1/iop.x29a
SEGMENT = D800
LHEADER : X29A_IOP_MAIN_P86
MODEND TYPE 3 : MAIN MODULE WITH START ADDRESS = 0040 : 0006
TOTALS :  IGNORED = 108    PEDATA = 82    PIDATA = 0


SLAVE PROCESSOR LOADER ROUTINE
PATHNAME : :sd:user/1/rtpro.x29a
SEGMENT = 8000
LHEADER : RTPRO_MAIN_P86
MODEND TYPE 3 : MAIN MODULE WITH START ADDRESS = 0040 : 0006
TOTALS :  IGNORED = 155    PEDATA = 494    PIDATA = 1098
```

This sequence loads and initializes the four slave processors in the following order: IOP A, IOP B, IOP C, and finally RTPRO. The executive then loads the symbol table from the scratch diskette in drive :F3: and finally enters the command interpreter.

Beginning at the time the XAIDS executive first enters the command interpreter, the operator may at any time request PERPRO to perform a snapshot of the screen display and provide a hard copy on the line printer. These hard copies are true single-frame snapshots and are requested using the keyboard entry <cntl P>. Another keyboard command recognized by PERPRO is <cntl U>, which signals a request to change the screen refresh update period (initially defaulted to 0.5 sec). The next keystroke is then interpreted as the desired period in deciseconds and must be one of the set 1,2,3,4,5,6,7,8,9,A,B,C,D,E,F covering the range from 0.1 to 1.5 sec. These commands are recognized by PERPRO only while in XAIDS mode; in RMX86 mode all keystrokes are passed to the RMX86 BIOS without prior PERPRO interpretation.

The command interpreter main menu display shown in display 1 (screen displays follow the figures at the end of this report) is the first formatted page display presented by the executive and is the last one seen when returning to RMX86 using the command EXIT. If the command TEST is entered, the subsidiary menu shown in display 2 is displayed. On the main menu, the first seven commands are X-29A extensions; on the test menu, the first two commands are X-29A extensions. These two command menus provide first-level access to the complete spectrum of XAIDS executive functions.

The first step normally required of the operator is to command the three control system computers to RUN state using command RU. Display 3 shows the display generated by this package as it verifies the links to the SEUs, selects all three channels, displays the checksums, and finally sends the SEU command RU to place the three channels in RUN mode. The sequence is performed twice since separate SEUs control the CLPs and IOPs, and both sets must be placed in RUN mode. A companion XAIDS command is QT, which performs the shutdown sequence shown in display 4. Notice the difference in the order of the operations performed since the computers must first be halted using the SEU command QT before the checksums can be fetched. If either package cannot communicate with either SEU, the software hangs while waiting for SEU response, and the operator must enter <esc> to terminate. The <esc> key is universally recognized by all XAIDS command handlers as an abort request; pressing the <esc> key enough times will return the operator from even the deepest level of a command handler to the main menu display.

The main menu commands SEU1 and SEU2 invoke handlers that allow the operator access to the respective SEU using the XAIDS terminal in emulator mode. In this mode, keystrokes are echoed to the CRT and sent to the SEU one at a time, and the response character stream received back from the SEU is written to the CRT. XAIDS responds to <esc> by terminating the emulation and returning to the main menu. While an SEU terminal emulator is being used, the <cntl P> command is functional, and hard copies may be made at any time.

During X-29A control system testing, it is frequently necessary to make minor program changes to set up special test conditions. One of the most common types of change is to the tables controlling the parameters being transmitted by the IOPs through the ARINC trunks, so that parameters may be viewed that are not normally accessible. Such program changes are called patches, and the patch manager (PTCH) permits creation, storage, retrieval, modification, and transmission of patch tables. Display 5 shows the PTCH handler menu and the status of the table currently occupying the PTCH template. A patch table may be up to 100 entries long and is edited using a display such as that shown in display 6. Each entry specifies the address and new contents for a single 16-bit memory word that may map to either RAM or EEPROM. When the SEND command is entered, the operator is queried as to which computers (either IOP or CLP) and what subset (any desired set of 1, 2, or 3) are to receive the patch table. The appropriate SEU is selected, the specified set or subset of processors is selected, those processors are halted, the table is transmitted, and new checksums are generated and displayed.

The main menu command PMSW (postmortem switch) permits the operator to control whether stale data from the ARINC trunks can be displayed. Normally, the XAIDS places blanks in the data field of any parameter whose source channel is not being serviced for any reason. If one channel of the X-29A control system fails, the ARINC stream from its IOP is no longer transmitted, and the fail-safe timer in the respective channel handler in RTPRO times out and declares the channel failed. However, there are tests where it is necessary to override this blanking feature so that data may be displayed from the last complete frame received before the channel went dead. If the postmortem switch is set, stale data will be displayed on free form and make page displays, but the routine USERCRTREQUEST$ACTIVATE will write a cautionary message on the bottom line of the CRT.

The X-29A aircraft employs a control system that dynamically adjusts flight control system gains as a function of flight condition. The X-29A-configured RTPRO

load module contains a set of equations that for comparison can compute the gains anticipated for any flight condition. The command GAIN accesses the display shown in display 7, which presents the contents of the current GAIN template. Display 8 shows the page displayed if H (help) is entered, summarizing the single-keystroke commands available. The operator may specify the value for each of 16 flight condition parameters using M (modify), give the table a name using T (title), and save the resulting table using S (save). When C (compute) is entered, the initial values are sent to RTPRO and the X29A$SIM package is iterated once, computing a full set of predicted gains. If the flight control system processors have previously been properly patched, these same initial conditions may then be overlayed to the processors using the I (initialize) command. The operator must then manually cycle the computers to RUN state for approximately 5 sec to allow the flight program to compute its gains. The operator then enters W (wrap up) to synchronously halt the processors and read the gains back into XAIDS through the SEUs. The XAIDS-computed and X-29A-computed gains are compared, and the results may be displayed using an operator-created make-page (MP) or free-form (FF) format.

The main menu load display command LD followed by a file number loads the template from the designated scratch file on drive :F3: and starts execution of the command handler associated with it. Display 9 shows the screen display resulting from the command entry LD 45 <return> . File 45 happened to contain the template for an MP display used for failure modes and effects testing of the angle-of-attack vane. Following the LD operation, this display will remain on the screen in refreshed mode until the operator enters either <esc> or <return>. If <esc> is entered, the display terminates, and control reverts back to the main menu. If <return> is pressed, the MP setup mode is entered, giving the operator access to a command list with means to modify, name, and save the page on scratch file. If H (help) is entered, display 10 appears giving an explanation of the MP command list. Note that MP not only displays data on the screen but is the means whereby recorder analog outputs are controlled. The MP template (like all templates) is retained indefinitely either until modified by its command handler or until overlaid by an LD operation that fetches another like template. Whenever the command MP is entered from the main menu, the MP command handler activates a display based on the current contents of the MP template.

Display 11 shows the page displayed by the command sequence LD 56 <return>. File 56 happened to contain a template for a free-form page created to display the strake positions and strake actuator commands from all three channels. Like MP, the display remains live until either <esc> returns control to the main menu or <return> forces the handler into the setup mode. Display 12 shows the FF setup page, which displays the static background plus the fields occupied by data items as bracketed zones. Eight single-keystroke commands are available; if H (help) is pressed, the page shown in display 13 is displayed. Since data items are not explicitly identified on the FF page except in background, the operator requires means to verify which data items are being requested. The L (list) command permits the operator to display the data item blocks from the current FF template on either the CRT screen or the line printer. Display 14 shows the data item list on CRT for the FF display loaded from file 56. If FF is entered from the main menu, the FF handler creates a live display based on the current contents of the FF template.

When the main menu command DK is entered, the page shown in display 15 appears with the DK command list and the floppy diskette drive or drives accessible by each. The diskette manager permits the operator to initialize a scratch diskette, delete a

file, view the directory on CRT, dump the directory to line printer, copy files from one scratch diskette to another, and sequentially load a group of files and produce a hard copy of the resulting displays on the line printer. An XAIDS scratch diskette is an 8-in single-sided double-density diskette that has been first formatted as a NAMED volume using the RMX86 human interface command FORMAT and then initialized using the DK command INIT. A scratch diskette is used to save MP templates, FF templates, PTCH tables, GAIN tables, and RDFN programs as numbered files using SAVE commands from within the respective command handlers. All scratch diskettes also have an additional file named DISPLAYFILES (created by the DK command INIT), which serves as the directory used by the DK command handler and by the LD command to access the numbered files. When viewing the directory for a scratch diskette on CRT, a display such as display 16 is created. For each of 100 numbered files, the directory contains an entry showing the time and date of creation, the owner mnemonic, and the title. The operator is able to scroll through the directory viewing the entries (in groups of 10) for all nonempty files until the desired information is found.

When the main menu command SYM is entered, the symbol table manager display shown in display 17 appears. This page displays the identifying information taken from the header of the currently loaded table and presents the SYM command list with a description of the function of each command. To make any changes to the individual symbol entries in the table, the operator must enter EDIT to access the page shown in display 18. This page allows the operator to find a specific entry or to scroll through the entire table alphabetically, and it permits the following types of changes to be made: addition of a new entry, deletion of an existing entry, modification to an existing entry, and cloning a new entry from an existing entry. After the needed changes are made, the operator must press <esc> to return to the SYM command list to save the new or revised table on diskette.

The main menu command RDFN accesses the handler for the RDAS function generator, which in turn controls the companion software embedded in RTPRO. This package is designed to permit both analog and discrete excitation functions to be fed through RDAS to the simulation laboratory interface with the X-29A control system. This mode of operation of the XAIDS has not yet been exercised, and results will be presented in a separate report.

When TEST is entered from the main menu, the additional five commands on the test menu (display 2) are made available to the operator. If BAUD is then entered, the display shown in display 19 appears. This handler provides direct access to the RS-232C transceivers on the three IOP boards and permits baud rate changes and the sending and receiving of test messages. Display 19 shows the messages generated as the operator changes the baud rate of IOP A port 1 from its default value of 2400 to 4800 (the snapshot was taken before the final <return>). Such a baud rate change would be made only if the terminal port on SEU 1 had also been changed to 4800 baud. The commands S (send) and R (receive) are hardware troubleshooting commands used only with the SEU trunks disconnected and wrap-back jumpers installed; in this configuration the transmitter outputs back to receiver inputs.

The ARNC command on the test menu provides hardware troubleshooting functions for the ARINC transceivers on the three IOPs. Such testing first requires disconnecting the trunks coming from the X-29A control system so that wrap-back jumpers may be installed. The handler allows test messages to be transmitted from one board and received and verified on another.

The RDAS command on the test menu permits hardware diagnostic and calibration tests on the RDAS rack through the LAN, and it indirectly provides confidence testing of the LAN itself. Input-output tests may be performed on the analogs, discretes, and the two serial ports in either open-loop or closed-loop mode. Test messages may be sent to the line printer port, and the status of the LAN activity counters may be displayed.

The CON command on the test menu permits the operator to generate numeric constants in various formats at any memory domain address desired. This command is used only during XAIDS software V/V.

When CAL is entered from the test menu the display shown in display 20 appears. This handler allows the operator to perform various tests on the analog recorder output DACs on RTPRO. The AUTO and SAW commands generate continuous waveforms on all 16 DAC channels, while MAN generates a manually stepped multilevel output on all 16 DACs. The SEL command permits the operator to select one of the 16 DAC channels and set it to any desired level for calibration. Display 20 shows the response if the operator wishes to set "pen" 1 to 2.5 V (the hard copy was taken before the final <return>). Entering TEST commands RTPRO to trigger the confidence test firmware in the DAC modules and report the status, thus verifying that the DAC modules are responding properly.

GUIDE FOR THE PROSPECTIVE XAIDS OWNER

A decision to commit to the use of an XAIDS in support of a project requires analysis of the cost of ownership and the expected benefits. This analysis must consider both the initial cost and the long-term overhead of supporting such a system. Based on experience to date with the XAIDS at Ames-Dryden, the following guidelines are offered to help the prospective user more accurately predict costs.

Initial hardware cost (excluding software) for an XAIDS console configured for the X-29A application is estimated at $100,000, which includes the cost of assembly. For the RDAS, the figure is approximately $25,000. The cost of spare parts is indeterminate since each user must decide what level of parts backup is required based on project tolerance to outages. In general, failure rates have been extremely low, with most maintenance problems relating to dirty connectors and conducted electromagnetic interference. Only one major failure has been experienced to date; the failure occurred in the XAIDS cardcage power supply and was caused by an inadvertant momentary short circuit on one of its outputs.

Hardware cost for the XAIDS console can be reduced slightly without impact on the baseline software. If the user has no requirement for an RDAS and no other LAN interfaces are involved, the LAN controller board can be deleted from the baseline board complement. All related software has been designed to be tolerant of LAN failures, and thus the deletion has no impact other than loss of LAN communications. Another optional deletion is the MAINT board, although this step should not be taken lightly. The maintenance processor has proved invaluable for hardware integration and XAIDS software V/V and is considered a wise investment. Any other departures from baseline will impact either the configuration of the operating system or the XAIDS baseline software, or both. In this case, the additional cost of software reengineering will probably exceed any reduction in hardware investment.

A hardware cost that is more difficult to estimate is the infrastructure required to support software and firmware. There are several strategies available, and much depends on whether the user already has on hand equipment that could host Intel support software. Both 8-bit and 16-bit support are required, and hardware must be available for programming a wide variety of PROMs. At Ames-Dryden an Intel Series III microcomputer development system (MDS) with 512K of RAM was already on hand, carried over from the earlier AIDS development. To this MDS was added a new PROM programmer and a 35-megabyte hard disk subsystem. In addition, it was decided to interface the MDS to an XAIDS brassboard configured to the baseline board complement, costing about $70,000. The interface being used is the Intel SBC957B, which provides file format conversion from MDS ISIS-II format to brassboard RMX86 format. This assemblage of support equipment is intended to be shared among all XAIDS users at Ames-Dryden and is highly cost effective.

The combination of a brassboard interfaced to a MDS has been highly productive and is strongly recommended as the best approach for software and firmware support. It provides a total support environment including 8-bit support for the PERPRO and RDAS in a totally independent facility, freeing the XAIDS console for full-time project support. However, for users who envision no need for 8-bit support, there is a minimum-cost strategy that would permit the XAIDS console itself to be used in the support role on a part-time basis. This would require the addition of an SBX351 multimodule to CENPRO to provide a serial RS232 interface to an external user-supplied PROM programmer. The programmer selected must have the necessary serial port and be able to accept PROM files in hexadecimal format through a user-written RMX86 utility. The additional hardware cost for the SBX351 and the PROM programmer should be under $4000.

The cost of purchased software depends upon the user's final choice of support strategy. For the Ames-Dryden environment, a wide spectrum of available support software has been incorporated. For the XAIDS brassboard, the cost is less then $20,000, including the RMX86 operating system, utilities, assembler, screen editor, and compilers for Pascal, FORTRAN, and PL/M. No additional cost is incurred for the RMX86 incorporation in the XAIDS console. For the MDS support of both 8-bit and 16-bit microprocessors, the cost is also less than $20,000, including screen editor, utilities, assemblers, and compilers for FORTRAN and PL/M.

Software engineering costs for user-specific modules is very much a function of the application; for the X-29A-configured XAIDS, approximately two man-years total development effort was expended. A facet of software engineering cost that needs to be taken into account is the V/V and certification of user modules as they are added to the system. At Ames-Dryden each increment of user software is incorporated into separate test modules kept apart from the normally booted system. When time is available, the XAIDS is rebooted with the new modules, and V/V testing is performed to a test plan developed jointly by the project engineer (the user) and the software engineer who prepared the update. If all tests are successfully passed, the updated modules are made available to the project team on a trial basis, in that either the old or the new versions may be used. Following a minimum of 10 hr operational use on the new version without errors, the new version is upgraded to normal boot. At least one previous version is kept on the hard disk drive in case it should ever be necessary to step back to an earlier version.

The cost effectiveness of the XAIDS, like that of its predecessor AIDS, has been amply demonstrated in spite of the hardware and software costs discussed above.

One component of the payback is the project time saved through productivity gains afforded by its high throughput, simplification of operator procedures, and potential for automatic testing. Another cost saving results from the elimination of extra equipment, made possible by its flexibility in assuming multiple roles. An indirect payback is the lowering of operator stress resulting from centralization of diverse functions at a user-friendly workstation. Many past and present users have declared this type of support system to be indispensable.

FUTURE XAIDS DEVELOPMENTS

As the X-29A XAIDS system loading has increased with the steady addition of more user software to provide new services, it was inevitable that a need for increased bandwidth would arise. Two goals for increased performance are being approached: (1) reducing bus loading to lower waiting times and (2) increasing processor speeds. Accomplishing the first goal will require combining the functions of the central processor, auxiliary RAM board, and auxiliary PROM board into a single SBC86/35 processor board having 512K of RAM and 128K of PROM. This will provide a self-contained environment for the RMX86 operating system without the need for bus cycles for instruction fetches. The second goal will be easily accomplished for the CENPRO and RTPRO by upgrading the current 5-MHz 8087 NDP chips to the 8-MHz version, thus immediately achieving a 60-percent increase in throughput. No speed improvement for the PERPRO or MAINT are judged as necessary.

A major upgrade to the XAIDS is planned for the next unit to be built. Two areas of enhancement are now being engineered: (1) addition of a second cardcage and (2) redesign of the hard disk and floppy diskette subsystems. To give future users additional board slots for user I/O channel boards, a second 21-slot cardcage is planned, with a much larger power supply. The two cardcages will be tied together with a pair of commercially available bus extender boards, one board plugged into each cardcage and linked by ribbon cable. This will result in one cardcage being available for slave boards only (not bus masters), while the other cardcage will be configured basically as shown in this report.

The hard disk and floppy diskette subsystems will be consolidated into a single chassis controlled by a single controller board. The current single 19-megabyte hard disk drive will be replaced by two 25-megabyte hard disk drives, and the 8-in floppy diskette drives will be replaced by 5.25-in drives. In addition, a 0.25-in streaming tape (cartridge) drive will be added to provide a backup capability for saving the contents of the hard disk drives. The entire complement will be interfaced to a single SBC214 controller board.

With regard to the RDAS, several enhancements are being considered for incorporation into future units. One is to use a larger cardcage and power supply to permit the user to install more I/O boards. A second improvement would be to use a 16-bit processor board as local controller instead of the present 8-bit board. A third major change would be to upgrade the RDAS-XAIDS link to the higher speed 10-MHz Ethernet (XEROX Corporation). This link would provide a 10-fold increase in bandwidth and would dramatically lower transport lag. In addition, having Ethernet capability on XAIDS would permit it to be interfaced to a wide range of networks now in operation.

With regard to future XAIDS users, personnel representing several projects have expressed interest in funding the construction of a unit. The engineering of interfaces and development of user software will be handled on a case-by-case basis as requirements are documented.

CONCLUDING REMARKS

A second-generation, general-purpose, user-programmable ground support equipment has been developed and placed in service in support of the X-29A forward-swept-wing aircraft project. The XAIDS design provides many enhancements over the earlier AIDS mechanization, including multiple 16-bit processors, pipeline data flow architecture, advanced operating system, resident software development tools, and remote data I/O capability. The baseline system software suite includes a large selection of data type handlers and display formats. User-defined extensions to this basic library are easily incorporated. Hardware and software interfaces to user-dependent subsystems have been tailored for flexibility in configuration to meet user requirements.

The major contribution of the XAIDS to the X-29A project has been the centralization of operator activites at a single workstation providing both telemetry data analysis and flight computer test set control. Software has been developed to permit operators to overlay flight computer memory with test code or data tables, or both, to permit alteration of test conditions and remapping of the telemetry stream. In addition, several large software modules have been incorporated to provide flight control system performance evaluation.

As with the earlier design, the use of off-the-shelf commercial hardware and operating system software greatly reduced the development burden and cost of ownership. The increasingly wide selection of directly compatible commercial hardware now available has made the design of user interfaces a straightforward matter of selecting the proper components.

The experience to date with both the XAIDS laboratory brassboard system and the X-29A unit has been excellent over many thousand hours of operating time. Reliability has been high, and users universally proclaim the system to be indispensable in support of the project. Several enhancements are now being engineered aimed at further increasing system bandwidth, and upgrades are being planned for the next XAIDS to be constructed.

National Aeronautics and Space Administration
Ames Research Center
Dryden Flight Research Facility
Edwards, California, October 7, 1986

APPENDIX A — PROTOTYPE REMOTE DATA ACQUISITION SUBSYSTEM

The remote data acquisition subsystem (RDAS) (fig. 10) was engineered to be an extension of the XAIDS that could meet two needs. Its primary intended use is to provide a remote I/O interface to a user target system in applications where direct cabling to the XAIDS is undesirable for reasons of wiring complexity or electromagnetic interference. An example of wiring complexity is a simulation interface to an actual aircraft, iron bird, or simulation laboratory involving many discretes and many analog parameters requiring rather large wiring bundles. Long distance to a target system could require use of an RDAS to eliminate interference in noisy environments.

An RDAS may be located up to 5000 ft from the XAIDS, with control and data flowing over a LAN employing a single RG-59/U coaxial cable of 3/16-in diameter. The XAIDS-RDAS protocol is SDLC at a 1.0-MHz rate using phase shift keying and automatic phase-locked-loop clock extraction.

The block diagram in figure 11 shows the configuration of the prototype RDAS constructed to support the X-29A project. The requirement was to provide a simulation I/O interface with 16 ADC inputs, 28 DAC outputs, 48 input discretes, and 48 output discretes. The solid-outline boxes in the diagram represent boards plugged into the 12-slot cardcage, shown in figure 12. The board complement includes an 8-bit processor, the LAN channel, seven DAC boards, one multiplexed ADC board, and two I/O expansion boards providing discretes I/O as well as two RS-232 serial ports. The dashed-outline boxes in the diagram represent optional peripherals that are not currently part of X-29A project requirements. The monitor terminal port on the processor board interfaces to software providing maintenance and troubleshooting utilities. The printer port is configured for Centronix parallel interface protocol and may be used, for example, to provide remote duplicate hard copy of the traffic going to the XAIDS printer. The serial ports could provide remote operator terminal function or data I/O. All connections to the RDAS are through the user I/O connector panel shown in figure 13.

Figure 14 shows the timing of the XAIDS-RDAS data flow, which iterates at a maximum rate of 96 Hz. The XAIDS processing is handled by RTPRO and provides data handling services to user modules as well as to the RA and RD channel handlers. Each cycle, RTPRO sends a message that contains as a minimum the 48 output discretes and the 28 DAC values. When RDAS receives this message, the discrete outputs and DACs are serviced, and any message extensions are passed to a background servicing task. Approximately 3 msec later, XAIDS commands RDAS to perform the data acquisition cycle for the 16 analog input values, fetch the 48 input discretes, snapshot certain status counters, and build the reply message. If the background task has generated any response packets, these are appended to the reply. A delay in the XAIDS LAN controller program allows time for the foregoing to complete and then triggers the transmission of the reply. When XAIDS receives this reply, the RTPRO interfaces are serviced, and a new message is generated.

Table 18 shows the structure of the XAIDS to RDAS data message. The command byte flags the message as one of two types: a data output message or a command to perform input data acquisition. In the latter case the command is a very short transmission consisting of just the command byte. Within a data message, a link word of 0FFFFH marks the end of the message while any other value marks the start of a packet and points to the next link.

Table 19 shows the format for the RDAS to XAIDS reply message, which, in addition to the input discretes and ADC values, routinely sends the length for five queues related to peripherals. The state flag byte will be zero if the RDAS background servicing task is idle; otherwise it is still servicing the previous set of packets and cannot accept more. The link word in the reply has the same meaning as above.

Servicing of the RDAS peripherals is handled in background as a low-priority task to provide maximum bandwidth for the discretes and analogs, which must be serviced at the full 96-Hz rate. Table 20 shows the currently implemented set of packet types that provide the services shown. When XAIDS sends one or more packets attached to its outgoing message, it must wait until RDAS sends back the response packets and clears the state flag; this is to prevent overrun of the background task. The X-29A project has not as yet generated a requirement for use of the peripheral ports, and therefore there are normally no packets transmitted in either direction.

APPENDIX B — XAIDS SYSTEM INTERRUPT UTILIZATION

The XAIDS system makes extensive use of interrupts to communicate between various elements of the system. Some interrupts are generated by one board and sent to another board through the IEEE-796 bus, which has eight paths set aside for such signalings. Other interrupts are generated by a chip on a given board and are simply routed to the CPU chip on that same board for processing. The processors CENPRO, RTPRO, PERPRO, and MAINT each have an interrupt controller chip that prioritizes eight input interrupts and outputs a single interrupt to the CPU chip. Software then determines which of the eight inputs caused the CPU interrupt and takes appropriate action. In addition, each CPU chip has one or more "hardware interrupts," which are direct inputs bypassing the interrupt controller. For PERPRO, there are four hardware interrupts called TRAP, A, B, and C; the other three processors have a single hardware interrupt called NMI (nonmaskable interrupt). The eight interrupt controller chip inputs are always called IR0 to IR7.

The design of an interrupt management system for XAIDS involved the allocation of scarce resources to meet baseline system and user-defined requirements. Table 21 shows how the eight bus interrupt lines were assigned for interboard signaling. INT3 is generated by RTPRO under user software control and may be used to signal one or more user-supplied I/O channel boards. INT1 and INT2 are reserved for intercommunication between user boards as required. The five remaining bus interrupt lines carry baseline system interrupts.

Tables 22 to 25 show how the interrupt servicing facilities are allocated on CENPRO, RTPRO, PERPRO, and MAINT, respectively. With the exception of PERPRO, the interrupts serviced are a mix of those generated onboard and those coming from one or more of the bus interrupt lines specified in table 22. Although PERPRO does generate a bus interrupt (INT5/), it does not receive any bus interrupts; it merely services interrupts generated on board.

APPENDIX C — RTPRO RAW DATA PROCESSING ALGORITHMS

One of the main tasks of RTPRO is the preprocessing of raw data under the control of two data acquisition control structures, each composed of blocks of the type shown in table 5. Each block is 64 bytes long and consists of 16 fields of various lengths containing an assortment of housekeeping information, data registers, pointers to servicing routines, constants, and flags. The CRT data acquisition structure may contain up to 255 such blocks, one for each data item displayed on the CRT screen. The DAC data acquisition structure contains up to a maximum of 16 such blocks, one for each DAC feeding the outputs to the recorder channels. There is one difference in register assignment in the two applications: the final register is used as a flag in the CRT case, while it is used as a data register in the DAC case.

Data registers within an acquisition structure are serviced by routines accessed using two pointers embedded in each block. The first to be called is a raw data snapshot routine, which copies 1 to 8 bytes (based on the number of raw data fetches) from the I/O channel handler's buffer to the raw data register. After all blocks in a given structure have had their data snapshots performed, a second pass is made, and a raw data processing routine for each block is called. These routines perform functions that vary widely, depending on the raw data type and whether the block is part of a CRT or DAC structure. In general, the raw data undergo three transformations: a fixed-point derivative is placed in the adjusted data register, a floating-point derivative is placed in the processed data register, and either a CRT zero flag or a DAC offset binary value is placed in the last data register. The constants K0 and K1 are derived from parameters in the symbol table and are computed only once during structure initialization.

The following algorithms are used by the seven system raw data processing routines for the CRT data acquisition structure. The steps shown for each algorithm are listed in the order taken. Each step represents a mathematical operation performed by one or more PLM86 language statements.

```
Raw data type 'ASC ' - ASCII character string
    adjusted$data  = raw$data
    processed$data = 0.0
    zero$flag      = true

Raw data type 'DISC' - Discrete
    adjusted$data  = (raw$data EXCL-OR xor$mask) AND and$mask
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'MBIN' - Masked binary
    adjusted$data  = (raw$data EXCL-OR xor$mask) AND and$mask
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'UBIN' - Unsigned binary
    adjusted$data  = raw$data
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false
```

```
Raw data type 'SBIN' - Signed binary
    adjusted$data  = raw$data with sign extended
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'SFBN' - Signed fractional binary
    K0             = display$zero
    K1             = 128.0 * (display$max-display$zero) / (256.0**nbytes)
    adjusted$data  = raw$data with sign extended
    processed$data = K0 + K1 * float(adjusted$data)
    adjusted$data  = fix(processed$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'IFLT' - Intel floating point
    processed$data = raw$data
    adjusted$data  = fix(processed$data)
    zero$flag      = true if processed$data = 0.0 else false
```

The following algorithms are used by the seven system raw data processing routines for the DAC data acquisition structure.

```
Raw data type 'ASC ' - ASCII character string
    dac$value      = 8000H       /* -5.000 volts */

Raw data type 'DISC' - Discrete
    adjusted$data    = (raw$data EXCL-OR xor$mask) AND and$mask
    dac$value        = 0000H if adjusted$data = 0   /*  0.000 volts */
    else dac$value   = 7FF0H                        /* +4.995 volts */

Raw data type 'MBIN' - Masked binary
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    adjusted$data  = (raw$data EXCL-OR xor$mask) AND and$mask
    processed$data = K0 + K1 * float(adjusted$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'UBIN' - Unsigned binary
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    processed$data = K0 + K1 * float(raw$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'SBIN' - Signed binary
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    adjusted$data  = raw$data with sign extended
    processed$data = K0 + K1 * float(adjusted$data)
    dac$value      = fix(processed$data)            /* see note */
```

```
Raw data type 'SFBN' - Signed fractional binary
    K0             = 32768.0 * (display$zero-recorder$bias) / recorder$scale
    K1             = 128.0 * (display$max-display$zero) / (256.0**nbytes)
                           * 32768.0 / recorder$scale
    adjusted$data  = raw$data with sign extended
    processed$data = K0 + K1 * float(adjusted$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'IFLT' - Intel floating point
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    processed$data = K0 + K1 * raw$data
    dac$value      = fix(processed$data)            /* see note */

Note:  Prior to conversion to fixed point, processed data are range
       limited to -32768.0 ≤ processed$data ≤ +32767.0.  This
       range limits the DAC value to -5.000 V to +4.995 V.
```

The following algorithms are used by the six X-29A raw data processing routines for the CRT data acquisition structure.

```
Raw data type 'HBOO' - HDP-5301 Boolean (16 to 64 bits)
    adjusted$data  = (raw$data EXCL-OR xor$mask) AND and$mask
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'HHEX' - HDP-5301 Hexadecimal (16 to 64 bits)
    adjusted$data  = raw$data
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'HINT' - HDP-5301 Integer (16 to 64 bits)
    adjusted$data  = raw$data with sign extended
    processed$data = float(adjusted$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'HSP ' - HDP-5301 Single Precision (16 bits)
    K0             = display$zero
    K1             = (display$max-display$zero) / 32768.0
    processed$data = K0 + K1 * float(raw$data)
    adjusted$data  = fix(processed$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'HDP ' - HDP-5301 Double Precision (32 bits)
    K0             = display$zero
    K1             = (display$max-display$zero) / 2147483648.0
    processed$data = K0 + K1 * float(raw$data)
    adjusted$data  = fix(processed$data)
    zero$flag      = true if processed$data = 0.0 else false

Raw data type 'HFLT' - HDP-5301 Floating Point (48 bit)
    processed$data = raw$data converted to Intel floating point format
    adjusted$data  = fix(processed$data)
    zero$flag      = true if processed$data = 0.0 else false
```

The following algorithms are used by the six X-29A raw data processing routines for the DAC data acquisitions structure.

```
Raw data type 'HBOO' - HDP-5301 Boolean (16 to 64 bits)
    adjusted$data    = (raw$data EXCL-OR xor$mask) AND and$mask
    dac$value        = 0000H if adjusted$data = 0   /*  0.000 volts */
    else dac$value   = 7FF0H                        /* +4.995 volts */

Raw data type 'HHEX' - HDP-5301 Hexadecimal (16 to 64 bits)
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    processed$data = K0 + K1 * float(raw$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'HINT' - HDP-5301 Integer (16 to 64 bits)
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    raw$data       = raw$data with sign extended
    processed$data = K0 + K1 * float(raw$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'HSP ' - HDP-5301 Single Precision (16 bits)
    K0             = 32768.0 * (display$zero-recorder$bias) / recorder$scale
    K1             = (display$max-display$zero) / recorder$scale
    processed$data = K0 + K1 * float(raw$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'HDP ' - HDP-5301 Double Precision (32 bits)
    K0             = 32768.0 * (display$zero-recorder$bias) / recorder$scale
    K1             = (display$max-display$zero) / 65536.0 / recorder$scale
    processed$data = K0 + K1 * float(raw$data)
    dac$value      = fix(processed$data)            /* see note */

Raw data type 'HFLT' - HDP-5301 Floating Point (48 bits)
    K0             = - 32768.0 * recorder$bias / recorder$scale
    K1             = + 32768.0 / recorder$scale
    processed$data = K0 + K1 * (raw$data converted to Intel floating pt.)
    dac$value      = fix(processed$data)            /* see note */
```

Note: Prior to conversion to fixed point, processed data are range limited to $-32768.0 \leq$ processed$data $\leq +32767.0$. This range limits the DAC value to -5.000 V to +4.995 V.

```
$save nolist

/* Module RTPRO.EXT  R. Glover  16 Oct 1984 */

/*
   This module contains the PLM86 external declarations for user-
   accessable RTPRO procedures.  It is assumed that the user is writing
   all RTPRO routines in PLM86 (large model).  The user must insert the
   following statement :
          $INCLUDE (RTPRO.EXT)
   where RTPRO.EXT is a copy of this module in the user's directory.  */

/* Following five subroutines provide a means for the user to link
   servicing routines to the indicated interrupts, where "ptr" is a
   pointer to the entry point of the servicing routine.  */

vector$timer$1: procedure (ptr) external; declare ptr pointer; end;
vector$user$A:  procedure (ptr) external; declare ptr pointer; end;
vector$user$B:  procedure (ptr) external; declare ptr pointer; end;
vector$user$C:  procedure (ptr) external; declare ptr pointer; end;
vector$intr$pushbutton: procedure (ptr) external;
                        declare ptr pointer; end;

/* Following 10 subroutines provide the user with means to enable or
   disable the specified RTPRO interrupt (all initially disabled). */

enable$timer$1:           procedure external; end;
disable$timer$1:          procedure external; end;
enable$user$A:            procedure external; end;
disable$user$A:           procedure external; end;
enable$user$B:            procedure external; end;
disable$user$B:           procedure external; end;
enable$user$C:            procedure external; end;
disable$user$C:           procedure external; end;
enable$intr$pushbutton:   procedure external; end;
disable$intr$pushbutton:  procedure external; end;

/* Following routine allows user to control the rate at which timer 1
   interrupts occur.  Period is milliseconds in the range 1 to 426.  */

set$timer$1$period: procedure (period) external;
                    declare period word; end;

/* Following routine allows user to set baud rate of RTPRO on-board
   USART.  "baud$rate" must be in the range 75 to 19200.  Note that
   user must poll USART since no interrupts are available. */

set$baud$rate: procedure (baud$rate) external;
               declare baud$rate word; end;
```

```
/* Following two functions return byte values for the specified
   Status/Control Panel registers.  */

mode$thumbwheels: procedure byte external; end;
control$switches: procedure byte external; end;

/* Following three routines provide the user with means to control the
   status LEDs on the Status/Control Panel.  The first routine updates
   all eight simultaneously where "pattern" is a byte value with the
   MSB corresponding to LED #7, 1=on, and 0=off.  The other two routines
   allow the user to turn on or off a single LED where "led" is a byte
   in the range 0 to 7 corresponding to the numbering on the panel.  */

set$status$leds:     procedure (pattern) external;
                     declare pattern byte; end;
turn$on$status$led:  procedure (led) external; declare led byte; end;
turn$off$status$led: procedure (led) external; declare led byte; end;

/* Following three routines allow user to declare and susequently service
   a user-created channel handler where "ptr" is a pointer to the
   channel declaration structure.  Note : user can allow "synch" to
   perform "channel$ready$crt" calls automatically (see below).  However,
   "channel$ready$dac" calls must be made individually and are usually
   made as soon as each user channel is serviced.  */

declare$channel:   procedure (ptr) external; declare ptr pointer; end;
channel$ready$crt: procedure (ptr) external; declare ptr pointer; end;
channel$ready$dac: procedure (ptr) external; declare ptr pointer; end;

/* Following routine allows user to inform RTPRO executive that all data
   input channels are ready and CRT control structure may be serviced.
   Note : "synch" calls "channel$ready$crt" routine for each active
   channel.  Only those channels which have not already been serviced by
   user-generated "channel$ready$crt" calls are processed.  */

synch: procedure external; end;

/* Following routine allows user to declare a user-created raw data
   type handler where "ptr" is a pointer to the data type declaration
   structure.  */

declare$data$type: procedure (ptr) external; declare ptr pointer; end;

/* Following function returns a DWORD value equal to the current master
   timer interrupt counter.  Can only be called from background since
   this routine momentarily disables interrupts during snapshot.   */

master$timer$cycle: procedure dword external; end;

$restore
```

```
$save nolist

/* Module XQSUB.EXT   R. Glover   15 October 1984  */

/* This module contains the PLM86 external declarations for CENPRO XAIDS
   support subroutines.  It is assumed that the user is writing CENPRO
   routines in PLM86 (large model).  The user must insert the following:
         $INCLUDE (XQSUB.EXT)
   where XQSUB.EXT is a copy of this module in the user's directory.  */

/* Following three routines are called by XAIDS executive only.  */

xq$init:      procedure (task$ptr) external;
              declare task$ptr pointer; end;
xq$set$time:  procedure byte external; end;
xq$exit:      procedure external; end;

/* Following three routines interface to the bus time-out (BTO) system
   tied to the non-maskable (NMI) interrupt.  The first two are proced-
   ures which activate or deactivate the NMI rupt, while the third is a
   function which returns the state of the BTO flag (clearing it after
   reading it).  */

xq$bto$enable:  procedure external; end;
xq$bto$disable: procedure external; end;
xq$bto$flag:    procedure byte external; end;

/* Following seven routines perform operator keyboard input editing and
   syntax checking for six types of input: any string, binary, octal,
   signed decimal, hexadecimal, signed floating point, and signed float-
   ing point or null string (return key only)  respectively. "result$ptr"
   is a pointer to a structure of the form:
   declare result structure
                    (last$key byte, nchar byte, chars (80) byte);  */

xq$keysin:     procedure (result$ptr) external;
               declare result$ptr pointer; end;
xq$binkey:     procedure (result$ptr) external;
               declare result$ptr pointer; end;
xq$octkey:     procedure (result$ptr) external;
               declare result$ptr pointer; end;
xq$deckey:     procedure (result$ptr) external;
               declare result$ptr pointer; end;
xq$hexkey:     procedure (result$ptr) external;
               declare result$ptr pointer; end;
xq$fpkey:      procedure (result$ptr) external;
               declare result$ptr pointer; end;
xq$fpkey$ret:  procedure (result$ptr) external;
               declare result$ptr pointer; end;
```

```
/* The following true-false function specifies whether ASCII string
   (delimited by blank) represents a valid floating-point constant.  */

xq$valid$fp$string: procedure (string$ptr,length) byte external;
                    declare string$ptr pointer, length byte; end;

/* The following routine clears the last keystroke buffer to null.  */

xq$clear$key: procedure external; end;


/* Following five functions return the ascii byte value equivalent of a
   single keystroke. The first waits for 'Y', 'y', 'N', 'n' , or <esc>
   and returns 'Y', 'N', or <esc>. The second waits for 'T', 't', 'F',
   'f' , or <esc> and returns 'T', 'F', or <esc>. The third waits for
   any keystroke but does not attempt to echo the keyin. The fourth
   returns a snapshot of the last keystroke buffer without waiting. The
   last accepts a byte argument and capitalizes if lower case letter. */

xq$yesno:         procedure byte external; end;
xq$true$false:    procedure byte external; end;
xq$anykey:        procedure byte external; end;
xq$key$snapshot:  procedure byte external; end;
xq$capitalize:    procedure (key) byte external; declare key byte; end;

/* Following two functions return true or false depending on last
   operator key pressed.  The first checks for a match to the key passed
   as argument while the second checks specifically for <ESC> .   */

xq$keystroke: procedure (key) byte external; declare key byte; end;
xq$escape:    procedure byte external; end;

/* Following two functions accept a single operator keystroke only if it
   is a member of a set of allowable characters contained in an array
   pointed to by the argument.  The first function returns the keystroke
   itself after echoing it.  The second function does not echo the
   keystroke and returns the index to the character based on its position
   in the array.  */

xq$command$key: procedure (array$ptr) byte external;
   declare array$ptr pointer; end;
xq$command$key$case: procedure (array$ptr) byte external;
   declare array$ptr pointer; end;

/* Following two routines wait for the operator to enter an acknowledge
   keystroke.  The first is usually used following an error condition;
   a beep is first sounded and then either a <rub> or <esc> must be
   entered.  The second first prints the message 'WAITING FOR <esc> > '
   and then will accept only an escape keystroke.  */

xq$rubesc:          procedure external; end;
xq$waiting$for$esc: procedure external; end;
```

```
/* Following five routines provide screen blanking operations.  */

xq$erase: procedure external; end;    /* Erase entire screen (1-24). */
xq$purge: procedure external; end;    /* Erase bottom line (24). */
xq$scrub: procedure external; end;    /* Erase last write string. */
xq$flush: procedure external; end;    /* Erase last operator entry. */
xq$blank$line: procedure (line$no) external; declare line$no byte; end;
                                      /* Erase specified line (3-23). */

/* Following moves cursor to specified row (1-24) & column (1-80). */

xq$movcur: procedure (row,col) external; declare (row,col) byte; end;

/* Following routine echoes most recent operator keystroke.  */

xq$echo: procedure external; end;

/* Following routine sounds a beep on the operator terminal.  */

xq$beep: procedure external; end;

/* Following two routines write a message string on the screen.  The
   first begins writing at current cursor position and the second begins
   at specified row and column.  In both cases, string$ptr must point to
   an ASCII byte string terminated by a zero byte (ASCII null).  */

xq$write:     procedure (string$ptr) external;
              declare string$ptr pointer; end;
xq$vec$write: procedure (row,col,string$ptr) external;
              declare (row,col) byte, string$ptr pointer; end;

/* Following two routines write an ASCII message of fixed length on the
   screen at current cursor position.  The first writes a single char-
   acter while the second permits writing a message up to 131 char.  */

xq$show$char: procedure (char) external; declare char byte; end;
xq$show$msg:  procedure (msg$ptr,length) external;
              declare msg$ptr pointer, length byte; end;

/* Following three routines are line printer interface routines providing
   string output, form feed, and multiple line feed respectively.  The
   first requires a pointer to a string as defined above.  */

xq$lp$out: procedure (string$ptr) external;
           declare string$ptr pointer; end;
xq$lp$ff:  procedure external; end;
xq$lp$lf:  procedure (nlines) external; declare nlines byte; end;


/* Following three routines return an ASCII string representation of the
   current time and/or date.  "dest$ptr" is a pointer to the destination
   buffer which must be of length 20, 8, and 9 bytes respectively.  */
```

```
xq$time$date: procedure (dest$ptr) external;
              declare dest$ptr pointer; end;
xq$time:      procedure (dest$ptr) external;
              declare dest$ptr pointer; end;
xq$date:      procedure (dest$ptr) external;
              declare dest$ptr pointer; end;

/* Following three functions return numeric value of operator keyin.  */

xq$get$hex$word:    procedure (max, last$key$ptr) word external;
                    declare max word, last$key$ptr pointer; end;
xq$get$hex$dword:   procedure (max, last$key$ptr) dword external;
                    declare max dword, last$key$ptr pointer; end;
xq$get$dec$integer: procedure (min, max, last$key$ptr) integer external;
                  declare (min, max) integer, last$key$ptr pointer; end;

/* Following four routines involve PLM86 REAL floating point quantities.

   In all cases "fp$ptr" is a pointer to a structure of the form:
    declare fp based fp$ptr structure
                             (value real, min real, max real);  */

xq$get$real:    procedure (fp$ptr, last$key$ptr) external;
                declare (fp$ptr, last$key$ptr) pointer; end;
xq$encode$real: procedure (fp$ptr, output$buffer$ptr) external;
                declare (fp$ptr, output$buffer$ptr) pointer; end;
xq$decode$real: procedure (fp$ptr, input$buffer$ptr) external;
                declare (fp$ptr, input$buffer$ptr) pointer; end;
xq$write$real:  procedure (fp$ptr) external; declare fp$ptr pointer; end;

/* Following two routines get an ASCII string from the operator keyboard
   and place it in the buffer designated by "dest$ptr".  The first is a
   general-purpose routine while the second gets a string of max length
   48 bytes.  */

xq$get$info:  procedure (dest$ptr, nchar, last$key$ptr) external;
              declare nchar byte, (dest$ptr, last$key$ptr) pointer; end;
xq$get$title: procedure (dest$ptr, last$key$ptr) external;
              declare (dest$ptr, last$key$ptr) pointer; end;

/* Following function returns a PLM86 DWORD which represents a 1-4 byte
   command in packed form.  The bytes are packed such that the command
   may be tested by a PLM86 instruction of the form:
      IF CMD = 'DIR ' THEN CALL DIR$CMD$SERVICE ;          */

xq$get$command: procedure (last$key$ptr) dword external;
                declare last$key$ptr pointer; end;


/* Following sends CR plus specified number of LF to screen cursor.  */

xq$cr$lf: procedure (nlf) external; declare nlf byte; end;
```

```
/* Following three routines write specified quantity to the CRT in
   hexadecimal or signed decimal format preceded and followed by a
   single blank.  WORD and DWORD quantities are written in hex and
   integer quantities are written in signed decimal.   */

xq$write$word:    procedure (val) external; declare val word; end;
xq$write$dword:   procedure (val) external; declare val dword; end;
xq$write$integer: procedure (val) external; declare val integer; end;

/* Following two routines provide error message display.  The first
   prints a diagnostic message on the same line as the operator entry;
   a beep is sounded and following either <rub> or <esc> the message
   plus the faulty operator entry is erased.  The second erases the
   bottom line (24) on the screen and flashes the message until <esc>
   is entered.  */

xq$diagnostic:    procedure (msg$ptr) external;
                  declare msg$ptr pointer; end;
xq$error$message: procedure (msg$ptr) external;
                  declare msg$ptr pointer; end;

/* Following six routines provide for display code conversions.  */

xq$encode$word:    procedure (val,output$ptr) external;
                   declare val word, output$ptr pointer; end;
xq$encode$dword:   procedure (val,output$ptr) external;
                   declare val dword, output$ptr pointer; end;
xq$encode$integer: procedure (val,output$ptr) external;
                   declare val integer, output$ptr pointer; end;
xq$decode$word:    procedure (input$ptr) word external;
                   declare input$ptr pointer; end;
xq$decode$dword:   procedure (input$ptr) dword external;
                   declare input$ptr pointer; end;
xq$decode$integer: procedure (input$ptr) integer external;
                   declare input$ptr pointer; end;

/* Following five routines invoke screen image handler processes.  */

xq$set$image$pointer:   procedure (image$ptr) external;
                        declare image$ptr pointer; end;
xq$fetch$image$pointer: procedure (row,col) pointer external;
                        declare (row,col) byte; end;
xq$image$insert:        procedure (nbytes,string$ptr,row,col) external;
                        declare (nbytes,row,col) byte, string$ptr pointer; end;
xq$image$extract:       procedure (row,col,nbytes,string$ptr) external;
                        declare (nbytes,row,col) byte, string$ptr pointer; end;
xq$create$static:       procedure external; end;
```

```
/* Following six routines invoke page display processes.  */

xq$screen$static:  procedure external; end;
xq$screen$buffer:  procedure word external; end;
xq$screen$show:    procedure (seg$token,keep$flag) external;
                   declare seg$token word, keep$flag byte; end;
xq$screen$refresh: procedure (seg$token,keep$flag) external;
                   declare seg$token word, keep$flag byte; end;
xq$screen$clean:   procedure (seg$token,keep$flag) external;
                   declare seg$token word, keep$flag byte; end;
xq$screen$finish:  procedure external; end;

/* Following three routines control the annotation of display pages
   in the upper left corner. */

xq$show$user$name:   procedure external; end;        /* row 1, col 1 */
xq$show$cmd$name:    procedure external; end;        /* row 2, col 1 */
xq$declare$cmd$name: procedure (cmd$ptr) external;
                     declare cmd$ptr pointer; end;

$restore
```

REFERENCES

1. Glover, Richard D.: Aircraft Interrogation and Display System: A Ground Support Equipment for Digital Flight Systems. NASA TM-81370, 1982.

2. Glover, Richard D.: Application Experience With the NASA Aircraft Interrogation and Display System: A Ground Support Equipment for Digital Flight Systems. Proc. IEEE/AIAA 5th Digital Avionics Systems Conference, pp. 17.3.1 to 17.3.10, Oct. 31 - Nov. 3, 1983, Seattle, Washington.

TABLE 1. — XAIDS BASELINE BOARD COMPLEMENT

Slots	Assignment	Manufacturer	Board type	Piggyback modules
1	Floppy controller	Intel Corp.	SBC 208	
2	Hard disk controller	SMS[a]	FWD8006	
3	Maintenance processor	Intel Corp.	SBC 86/05	
4,5	LAN I/O channel	Computrol	30-0090	30-0078 (1-MHz modem)
6,7	Real-time processor	Intel Corp.	SBC 86/30	SBX 328 (DAC, 2 each) SBX 337 (8087 NDP) SBC 304 (128K RAM)
8,9	(Empty)			
10	Auxiliary 256K RAM	Intel Corp.	SBC 056A	
11	Central processor	Intel Corp.	SBC 86/30	SBX 337 (8087 NDP) SBC 304 (128K RAM)
12	Auxiliary 64K PROM	Intel Corp.	SBC 464	
13 to 19	(Empty)			
20	Peripheral processor	Intel Corp.	SBC 80/30	
21	Clock and Calendar	DP[b]	TCU-410	

[a]Scientific Micro Systems, Inc.

[b]Digital Pathways, Inc.

TABLE 2. — XAIDS BUS MEMORY MAPPING

Memory address (hexadecimal)	Block size, bytes	Board
F0000 to FFFFF	64K	Not mapped to bus
E0000 to EFFFF	64K	Auxiliary PROM
C8000 to DFFFF	96K	User I/O boards
C4000 to C7FFF	16K	PERPRO RAM
C0800 to C3FFF	14K	System I/O spare
C0000 to C07FF	2K	LAN controller RAM
80000 to BFFFF	256K	RTPRO RAM
40000 to 7FFFF	256K	Auxiliary RAM
00000 to 3FFFF	256K	CENPRO RAM

TABLE 3. — XAIDS BUS I/O MAPPING

I/O address (hexadecimal)	Block size, bytes	Assignment
xxF0 to xxF7[a]	8	Time and date unit
0180 to 01AF	48	Floppy diskette controller
0100	1	Hard disk controller
000F	1	Auxiliary RAM parity register

[a]xx signifies that the digits are not decoded.

TABLE 4. — RTPRO CHANNEL DECLARATION STRUCTURE

Number of bytes	Field type	Assignment
2	ASCII	Channel mnemonic[a]
1	Integer	Channel status[b]
1	Integer	Channel number (0 to 31)
4	Hexadecimal	Assigned channel mask[c]
4	Pointer	Address of CRT setup routine[d]
4	Pointer	Address of CRT begin routine[e]
4	Pointer	Address of CRT halt routine[f]
4	Pointer	Address of DAC setup routine[d]
4	Pointer	Address of DAC begin routine[e]
4	Pointer	Address of DAC halt routine[f]
32	total bytes per channel declaration structure	

[a]Four system channels are declared by RTPRO:

'PP'	PERPRO	RAM (address range 04000H to 07FFFH)
'RP'	RTPRO	RAM (address range 00000H to 3FFFFH)
'RA'	RDAS	analog inputs (ADC 0 to 15)
'RD'	RDAS	discrete inputs (byte registers 0 to 5)

The user may declare as many additional channels as desired, up to a maximum of 32 total channels.

[b]Channel status is used to denote condition of data acquisition flow paths:

0 channel fail
1 CRT data flow OK
2 DAC data flow OK
3 both CRT and DAC data flow OK

[c]Channel mask is equal to 2**<channel number>.

[d]Setup routine is called once for each data item block requesting channel in corresponding acquisition structure.

[e]Begin routine is called only once to initiate channel data flow to corresponding acquisition structure.

[f]Halt routine is called only once to terminate channel data flow to corresponding acquisition structure.

TABLE 5. — RTPRO DATA ACQUISITION BLOCK STRUCTURE

Number of bytes	Field type	Assignment
1	Integer	Sequence number[a]
1	Integer	Channel number (0 to 31)
1	Integer	Raw data type (0 to 63)
2	Integer	Symbol table entry number (1 to 510)
1	Integer	Number of raw data fetches[b]
4	Pointer	Address of raw data snapshot routine
4	Pointer	Address of raw data in channel buffer
8	Unstructured	Raw data register[c]
4	Pointer	Address of raw data processing routine
8	Hexadecimal	Exclusive-OR mask[d]
8	Hexadecimal	AND mask[d]
8	Unstructured	Adjusted data register[c]
4	Floating point	Constant K0 (intercept)[e]
4	Floating point	Constant K1 (slope)[e]
4	Floating point	Processed data register
2	Integer or flag	DAC value or zero flag register[f]
64	total bytes per data acquisition block (either CRT or DAC)	

[a]For DAC acquisition, sequence number must be 0 to 15, representing which DAC receives the output. For CRT acquisition, it must be 0 to 254, representing the index number of the data item in the page display handler. In either case, a value of 0FFH (255) indicates the end of data acquisition table.

[b]The type of fetch performed depends on the raw data snapshot routine. The total data fetch must not exceed 64 bits in length.

[c]Organization of these registers is under the control of the raw data snapshot and raw data processing routines.

[d]Used by masked binary (MBIN) and discrete (DISC) raw data handlers. XOR mask used for inversion; AND mask provides selection.

[e]Used for both DAC and CRT acquisition processing for parameters that can be approximated by a first-order (linear) equation. Any parameter needing higher order processing (including piecewise linear) must have its own dedicated raw data processing routine.

[f]For DAC acquisition, this register contains DAC output value in left-justified 12-bit signed twos complement binary. For CRT acquisition, it contains zero flag indicating state of processed data register.

TABLE 6. — RTPRO RAW DATA TYPE DECLARATION STRUCTURE

Number of bytes	Field type	Assignment
4	ASCII	Raw data type mnemonic[a]
4	Pointer	Address of CRT setup routine[b]
4	Pointer	Address of DAC setup routine[c]
12	total bytes per raw data type declaration structure	

[a]Seven system raw data types are declared by RTPRO:

DISC	discrete
UBIN	unsigned binary
SBIN	signed binary
SFBN	signed fractional binary
ASC	ASCII character string
IFLT	Intel floating point
MBIN	masked binary

The user may declare as many additional raw data types as desired, up to a maximum of 64 total raw data types.

[b]CRT setup routine is called once for each data item block having corresponding raw data type in CRT acquisition structure.

[c]DAC setup routine is called once for each data item block having corresponding raw data type in DAC acquisition structure.

TABLE 7. — RTPRO FUNCTION GENERATOR BLOCK STRUCTURE

Number of bytes	Field type	Assignment
1	Integer	Sequence number (0 to 99)
1	Integer	Timer number (0 to 7)
1	Integer	Timer number link
1	---	(Fill)
2	Integer	Enable time link
2	Integer	Trigger time link
2	Integer	Disable time link
1	Hexadecimal	Dynamically active flag
1	Integer	Dynamic backward link
1	Integer	Dynamic forward link
1	Integer	Output destination code[a]
1	Integer	Function type number (0 to 31)
1	Integer	Function state variable[b]
4	Pointer	Address of state change routine
4	Pointer	Address of dynamic routine
4	Floating point	Enable time
4	Floating point	Trigger time
4	Floating point	Disable time
4	Floating point	Register 0[c]
4	Floating point	Register 1[c]
4	Floating point	Register 2[c]
4	Floating point	Register 3[c]
4	Floating point	Register 4[c]
4	Floating point	Register 5[c]
4	Floating point	Register 6[c]
64	total bytes per function generator block	

[a]Output destination code values:

0 to 27 indicates RDAS DAC number
64 to 111 indicates RDAS discrete output bit 0 to 47

[b]Function states:

0 idle
1 enabled
2 triggered
3 disabled

[c]General-purpose registers whose assignment is under the control of the individual function type software modules.

TABLE 8. — CENPRO COMMAND DECLARATION STRUCTURE

Number of bytes	Field type	Assignment
4	ASCII	Command mnemonic[a]
4	Pointer	Address of ASCII name string[b]
4	Pointer	Address of ASCII description string[c]
4	Pointer	Address of command servicing routine
4	Pointer	Address of template (if any)[d]
2	Integer	Length in bytes of template
22	total bytes per command declaration structure	

[a]Eight system commands are declared by CENPRO:

Mnemonic	Name	Description
'MP '	Make page	Tabular data display format
'FF '	Free form	Unstructured data display format
'LD '	Load display	Display scratch file loader
'DK '	Diskette	Display scratch diskette manager
'SYM '	Symbols	Symbol table manager
'RDFN'	RDAS f(t) generator	RDAS DAC and discrete function generator
'TEST'	Test routines	User and system maintenance packages
'EXIT'	Exit XAIDS	Return to RMX86 executive

User may declare as many additional commands as desired.

[b]Name may be up to 20 characters in length, delimited by zero byte.

[c]Description may be up to 45 characters in length plus delimiter.

[d]"Template" refers to a data structure that can be saved on scratch diskette as a numbered file and subsequently retrieved using 'LD'.

TABLE 9. — CENPRO SCRATCH DISKETTE DIRECTORY

Number of bytes	Field type	Assignment
5	---	(Not used)
21	ASCII	Time and date of scratch diskette initialization
49	ASCII	Name of scratch diskette
7500	Structured	Scratch file information entries[a]
7575	total bytes in scratch diskette directory file 'displayfiles'	

[a]Up to 100 scratch files may be created bearing RMX86 filenames '1' to '100'. For each file created by a SAVE command within an XAIDS servicing routine, the corresponding entry in the scratch diskette directory file is updated with the following information:

Number of bytes	Subfield type	Assignment
1	Integer	Flag byte (0 denotes empty, 1 denotes occupied)
4	ASCII	Command mnemonic of saving routine
21	ASCII	Time and date save is performed
49	ASCII	Title generated by saving routine
75	bytes per directory entry	

TABLE 10. — CENPRO DISPLAY FORMAT DECLARATION STRUCTURE

Number of bytes	Field type	Assignment
1	ASCII	Display format type code letter[a]
4	Pointer	Address of format servicing routine
5	total bytes per display format declaration structure	

[a]Eight display formats are declared by CENPRO:

Code	Invocation	Description	Register displayed
'A'	Aw	Alphanumeric (ASCII)	Raw data
'B'	Bw	Binary	Raw data
'L'	Lw	Logical ('T' or 'F')	Raw data (least significant bit only)
'Z'	Zw	Hexadecimal	Raw data
'I'	Iw	Signed decimal integer	Adjusted data
'F'	Fw.d	Signed fixed point	Processed data
'E'	Ew.d	Signed exponential	Processed data
'M'	Mw	Message (one of two)	Zero flag

w is the field width (1 to 80) and d is number of decimal places (0 to 9). User may declare as many as 18 additional formats for customized data display using the remaining letters of the alphabet.

TABLE 11. — SYMBOL TABLE ENTRY BLOCK STRUCTURE

Number of bytes	Field type	Assignment
2	Integer	Entry number index (1 to 510)
2	Integer	Link up index[a]
2	Integer	Link down index[a]
14	ASCII	Allowable channel codes[b]
32	ASCII	Symbol name + description field[c]
4	ASCII	Raw data type designator
1	---	(Spare)
1	Integer	Number of locations to be fetched
8	Hexadecimal	Exclusive-OR mask[d]
8	Hexadecimal	AND mask[d]
4	Floating point	Display zero
4	Floating point	Display max
4	Floating point	Recorder bias
4	Floating point	Recorder scale
4	Hexadecimal	Address of raw data[e]
1	ASCII	Display format type code (letter)[f]
1	Integer	Display format field length (1 to 80)[f]
1	Integer	Display format number of decimal places (0 to 9)[f]
1	---	(Spare)
10	ASCII	Zero message[g]
10	ASCII	One message[g]
10	ASCII	Units
128	total bytes per symbol table entry block	

[a]Links provide alphabetized access.

[b]One to five allowable channels may be specified, two alphanumeric characters each separated by a single blank.

[c]Name subfield is delimited by first blank encountered. Remainder of field is considered descriptive information for display only.

[d]These are used by the masked binary (MBIN) and discrete (DISC) raw data handlers. XOR mask used for inversion; AND mask provides selection.

[e]Mapping scheme depends on channel handler.

[f]These three fields combined provide FORTRAN-like format specification.

[g]Alternative messages selected by message (M) display format handler.

TABLE 12. — CENPRO DATA DISPLAY ITEM BLOCK STRUCTURE

Number of bytes	Field type	Assignment
1	Integer	Request number (0 to 254)[a]
1	ASCII	Display format type code
1	Integer	Display format field length (1 to 80)
1	Integer	Display format number of decimal places (0 to 9)
4	Pointer	Address of format servicing routine[b]
1	Integer	Row number (1 to 24)
1	Integer	Column number (1 to 80)
10	ASCII	Zero message[c]
10	ASCII	One message[c]
30	total bytes per	data display item block

[a]This number is keyed to the RTPRO CRT data acquisition block sequence number corresponding to data item. This allows format servicing routine to map RTPRO memory registers containing parameter data.

[b]This address is obtained from display format declaration structure.

[c]These messages are copied from symbol table entry block structure. They are used only by type M (message) display format routine.

TABLE 13. — CENPRO MAKE PAGE DISPLAY TEMPLATE

Number of bytes	Field type	Assignment
4	ASCII	Command mnemonic is 'MP '
50	ASCII	Page title[a]
760	Structured	Data definition entries[b]
814 total bytes per MP template		

[a]Operator chooses desired page title (appears on line 2).

[b]Operator may enter up to 20 data items. The data items each occupy a single line (4 to 23) and are accessed by item number (1 to 20). Each item is defined by a structure as follows:

Number of bytes	Subfield type	Assignment
2	ASCII	Channel code
32	ASCII	Parameter name
1	ASCII	Display format type code
1	Integer	Display format field length
1	Integer	Display format number of decimal places
1	---	(Spare)
38 bytes per line item entry		

A vacant entry is denoted by two blanks in channel code subfield.

TABLE 14. — CENPRO FREE FORM DISPLAY TEMPLATE

Number of bytes	Field type	Assignment
4	ASCII	Command mnemonic is 'FF '
50	ASCII	Page title[a]
1680	ASCII	Background image[b]
10200	Structured	Data definition entries[c]
11934 total bytes per FF template		

[a]Operator chooses desired page title (appears on line 2).

[b]Operator may create any static background desired within lines 3 to 23.

[c]Operator may create from 0 to 255 data items positioned anywhere within lines 3 to 23. If desired, the display may be entirely static information (background only). Entries are structured as follows:

Number of bytes	Subfield type	Assignment
2	ASCII	Channel code
32	ASCII	Parameter name
1	ASCII	Display format type code
1	Integer	Display format field length
1	Integer	Display format number of decimal places
1	---	(Spare)
1	Integer	Row location (3 to 23)
1	Integer	Column location (1 to 80)
40	bytes per data item entry	

A vacant entry is denoted by two blanks in channel code subfield.

TABLE 15. — CENPRO RDAS FUNCTION GENERATOR TEMPLATE

Number of bytes	Field type	Assignment
4	ASCII	Command mnemonic is 'RDFN'
49	ASCII	Program title[a]
21	ASCII	Time and date of creation[b]
1	Integer	Number of template entries
29	---	(Unused)
5200	Structured	Program function control entries[c]
5304 total		

[a]Operator chooses desired page title (appears on line 2).

[b]This records time and date when save operation is performed. This field is filled with blanks if template modified and save not yet performed.

[c]Operator may create from 1 to 100 function control entries positioned anywhere within the table. Entries are structured as follows:

Number of bytes	Subfield type	Assignment
4	ASCII	Output channel code
32	ASCII	Function type specification
4	ASCII	Timer control specification
4	Real	Enable time
4	Real	Trigger time
4	Real	Disable time
52	bytes per function control entry	

A vacant entry has four blanks in output channel code subfield.

TABLE 16. — ASCII EDITOR ENTRY BLOCK STRUCTURE

Number of bytes	Field type	Assignment
2	Integer	Entry number index (1 to 510)
2	Integer	Link up index[a]
2	Integer	Link down index[a]
32	ASCII	Entry name and description field[b]
10	ASCII	Information field 1[c]
10	ASCII	Information field 2[c]
10	ASCII	Information field 3[c]
10	ASCII	Information field 4[c]
10	ASCII	Information field 5[c]
10	ASCII	Information field 6[c]
10	ASCII	Information field 7[c]
10	ASCII	Information field 8[c]
10	ASCII	Information field 9[c]
128	total bytes per ASCII editor entry block	

[a]Links provide alphabetized access.

[b]Name subfield is delimited by first blank encountered. Remainder of field is considered descriptive information for display only.

[c]These fields are assigned meanings by the command handler invoking the ASCII editor routine.

TABLE 17. — X-29A INTERFACE CHANNEL BOARD COMPLEMENT

Slots	Assignment	Manufacturer	Board type	Piggyback modules
13,14	X-29A I/O channel A	Intel Corp.	SBC 86/14	ARX-429B[a] SBX 351[b]
15,16	X-29A I/O channel B	Intel Corp.	SBC 86/14	ARX-429B[a] SBX 351[b]
17,18	X-29A I/O channel C	Intel Corp.	SBC 86/14	ARX-429B[a]

[a]ARINC 429B transceiver is manufactured by Procise Corporation.

[b]RS-232 serial interface module is manufactured by Intel Corporation.

TABLE 18. — XAIDS TO RDAS MESSAGE STRUCTURE

Number of bytes	Field type	Assignment
1	Control	Command byte[a]
1	---	(Spare)
1	Packed binary	Discretes out 07 to 00
1	Packed binary	Discretes out 15 to 08
1	Packed binary	Discretes out 23 to 16
1	Packed binary	Discretes out 31 to 24
1	Packed binary	Discretes out 39 to 32
1	Packed binary	Discretes out 47 to 40
56	Offset binary	DAC 00 to 27[b]
8	---	(Spare)
2	Hexadecimal	Packet stream link[c]
74	total bytes per XAIDS message, assuming no packets appended	

[a]This message format is for message carrying discretes out and DACs. Two different commands are recognized by RDAS:

0 perform input data acquisition cycle
2 update output discretes and DACs

For the first of these, message length is 1 (command alone).

[b]DAC values are 12-bit left-justified offset binary with the following ranges:

7FFxH	+9.995 V
000xH	0 V
800xH	-10.000 V

Least significant four bits are not used.

[c]Link is 0FFFFH if no packets are appended.

TABLE 19. — RDAS TO XAIDS REPLY STRUCTURE

Number of bytes	Field type	Assignment
1	Control	Background task status byte[a]
3	---	(Spare)
2	Integer	Port A transmit queue length[b]
2	Integer	Port B transmit queue length[b]
2	Integer	Port A receive queue length[b]
2	Integer	Port B receive queue length[b]
2	Integer	Printer queue length[b]
1	Packed binary	Discretes in 07 to 00
1	Packed binary	Discretes in 15 to 08
1	Packed binary	Discretes in 23 to 16
1	Packed binary	Discretes in 31 to 24
1	Packed binary	Discretes in 39 to 32
1	Packed binary	Discretes in 47 to 40
32	Offset binary	ADC 00 to 15[c]
2	Hexadecimal	Packet stream link[d]
54	total bytes per RDAS reply, assuming no packets appended	

[a]If status byte is not equal to 0, then RDAS background task is busy processing previous packet stream and cannot accept more. When processing is complete, response packets are appended as required to next outgoing reply and status byte is reset to zero.

[b]RDAS reports these each cycle to give XAIDS means of managing circular queues that buffer I/O flow to and from peripherals.

[c]ADC values are 12-bit left-justified offset binary with the following ranges:

7FF0H	+9.995 V
0000H	0 V
8000H	-10.000 V

Least significant four bits are always zero.

[d]Link is 0FFFFH if no packets are appended.

TABLE 20. — XAIDS-RDAS PACKET FORMATS

Code	Length	Structure	Description
		XAIDS to RDAS message packet formats	
10	5 + n	code, address, count, [n bytes]	Write block to memory
11	5	code, address, count	Return block from memory
12	5 + n	code, port, count, [n bytes]	Write to I/O ports
13	5	code, port, count	Read I/O ports
14	3	code, address	Execute subroutine
20	3 + n	code, count, [n bytes]	Output to printer
21	3 + n	code, count, [n bytes]	Output to port A
22	1	code	Return port A receiver queue
23	3 + n	code, count, [n bytes]	Output to port B
24	1	code	Return port B receiver queue
25	1	code	Activate port A
26	1	code	Deactivate port A
27	1	code	Activate port B
28	1	code	Deactivate port B
		RDAS to XAIDS reply packet formats	
10	2	code, status	Memory write complete
11	6 + n	code, status, address, count, [n bytes]	Memory read block
12	2	code, status	I/O write complete
13	6 + n	code, status, port, count, [n bytes]	I/O read block
14	2	code, status	Subroutine executed
20	2	code, status	Printer output complete
21	2	code, status	Port A output complete
22	6 + n	code, status, count, [n bytes]	Port A input queue
23	2	code, status	Port B output complete
24	6 + n	code, status, [n bytes]	Port B input queue
25	2	code, status	Port A activated
26	2	code, status	Port A deactivated
27	2	code, status	Port B activated
28	2	code, status	Port B deactivated

TABLE 21. — XAIDS BUS INTERRUPT ALLOCATION

Interrupt	Generated by	Signals	Monitored by[a]
INT0/[b]	INT0 pushbutton	Software reset	RTPRO and user I/O
INT1/	LAN controller	I/O complete	RTPRO
INT2/[c]	User I/O channel	User defined	RTPRO
INT3/[c]	User I/O channel	User defined	RTPRO
INT4/[c]	RTPRO	User defined	User I/O channels
INT5/[d]	PERPRO	I/O acknowledge	CENPRO
INT6/[d]	Hard disk controller	I/O complete	CENPRO
INT7/[d]	Floppy controller	I/O complete	CENPRO

[a]The maintenance processor also monitors all eight interrupts.

[b]Used to reenter bootstrap firmware prior to rebooting.

[c]Reserved for user.

[d]Used by operating system basic I/O.

TABLE 22. — XAIDS CENPRO INTERRUPT ASSIGNMENTS

Interrupt	Generated by	Signals
NMI[a]	Fail-safe timer	Bus timeout
IR0	NDP	Floating-point error
IR1	(Spare)	
IR2[b]	Timer 0	10-msec clock
IR3[c]	PERPRO	I/O acknowledge
IR4[c]	Hard disk controller	I/O complete
IR5[c]	Floppy controller	I/O complete
IR6[d]	USART	Receiver ready
IR7[d]	USART	Transmitter ready

[a]Can be activated and deactivated using XAIDS system calls; flashes red bus timeout light and sounds beep at operator's terminal.

[b]Used by operating system as master timer.

[c]Routed to CENPRO through XAIDS bus interrupt lines.

[d]Used by operating system debugger terminal handler.

TABLE 23. — XAIDS RTPRO INTERRUPT ASSIGNMENTS

Interrupt	Generated by	Signals
NMI[a]	INT0 Pushbutton	Software reset
IR0	NDP	Floating-point error
IR1[b]	Fail-safe timer	Bus timeout
IR2[a]	LAN controller	I/O complete
IR3[a,c]	User I/O channel	User defined
IR4[a,c]	User I/O channel	User defined
IR5	Timer 0	2-msec clock
IR6[c]	Timer 1	User clock
IR7[c,d]	SCP INTR	User manual interrupt

[a]Routed to RTPRO through XAIDS bus interrupt lines.

[b]Flashes red bus timeout light and sounds beep at operator's terminal.

[c]Setup and servicing under control of user software modules.

[d]Generated by status and control panel INTR push-button; this switch is filtered and generates only one interrupt per operation.

TABLE 24. — XAIDS PERPRO INTERRUPT ASSIGNMENTS

Interrupt	Generated by	Signals
TRAP	(Not used)	
A[a]	Fail-safe timer	Bus timeout
B	(Not used)	
C	(Not used)	
IR0	(Not used)	
IR1	(Not used)	
IR2	(Not used)	
IR3[b]	Timer 1	10-msec clock
IR4	(Not used)	
IR5[c]	USART	Transmitter ready
IR6[c]	USART	Receiver ready
IR7	(Not used)	

[a]Flashes red bus timeout light and sounds beep at operator's terminal.

[b]Used as master clock for screen refresh timing.

[c]Used in servicing of main operator terminal.

TABLE 25. — XAIDS MAINTENANCE PROCESSOR INTERRUPT ASSIGNMENTS

Interrupt	Generated by	Triggers
NMI[a]	Fail-safe timer	Bus timeout
IR0	Bus INT0/	Tally counter 0
IR1	Bus INT1/	Tally counter 1
IR2	Bus INT2/	Tally counter 2
IR3	Bus INT3/	Tally counter 3
IR4	Bus INT4/	Tally counter 4
IR5	Bus INT5/	Tally counter 5
IR6	Bus INT6/	Tally counter 6
IR7	Bus INT7/	Tally counter 7

[a]Flashes red bus timeout light and sounds beep at operator's terminal.

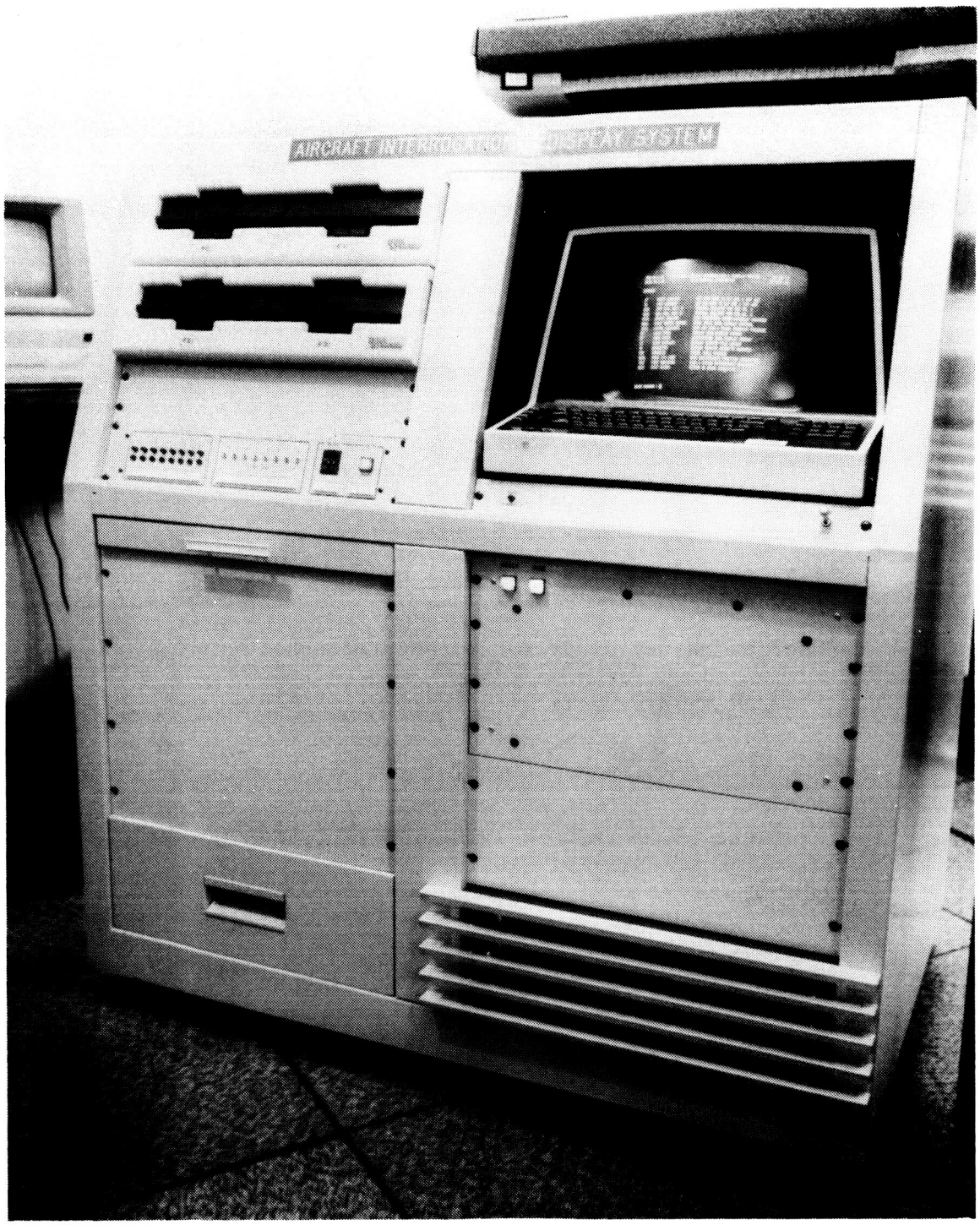


Figure 1. Extended aircraft interrogation and display system.

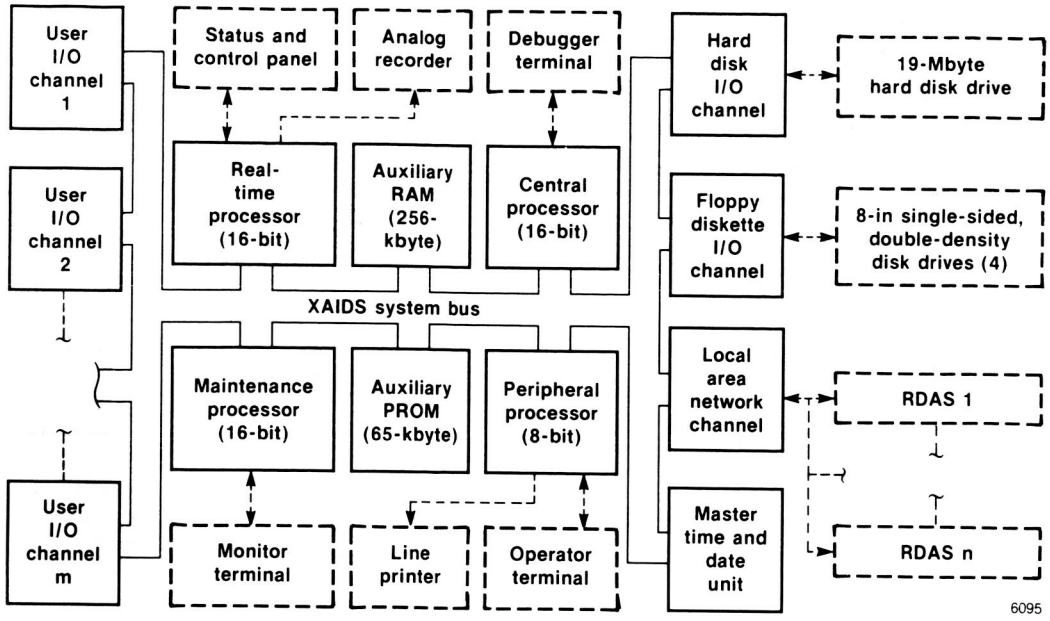


Figure 2. XAIDS system overview.

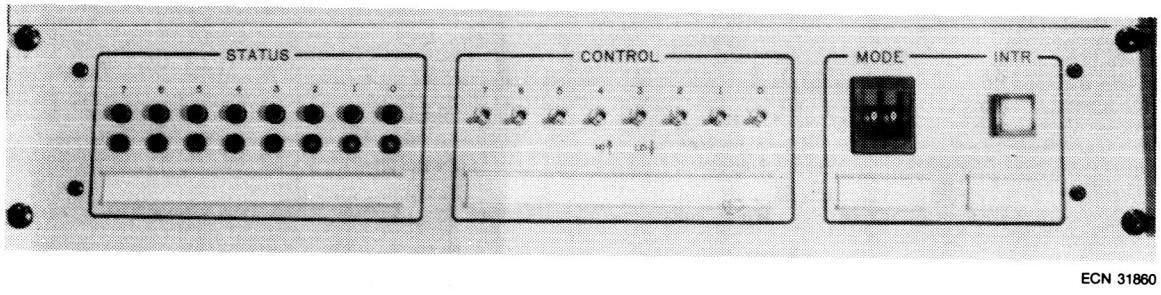


Figure 3. XAIDS status and control panel.

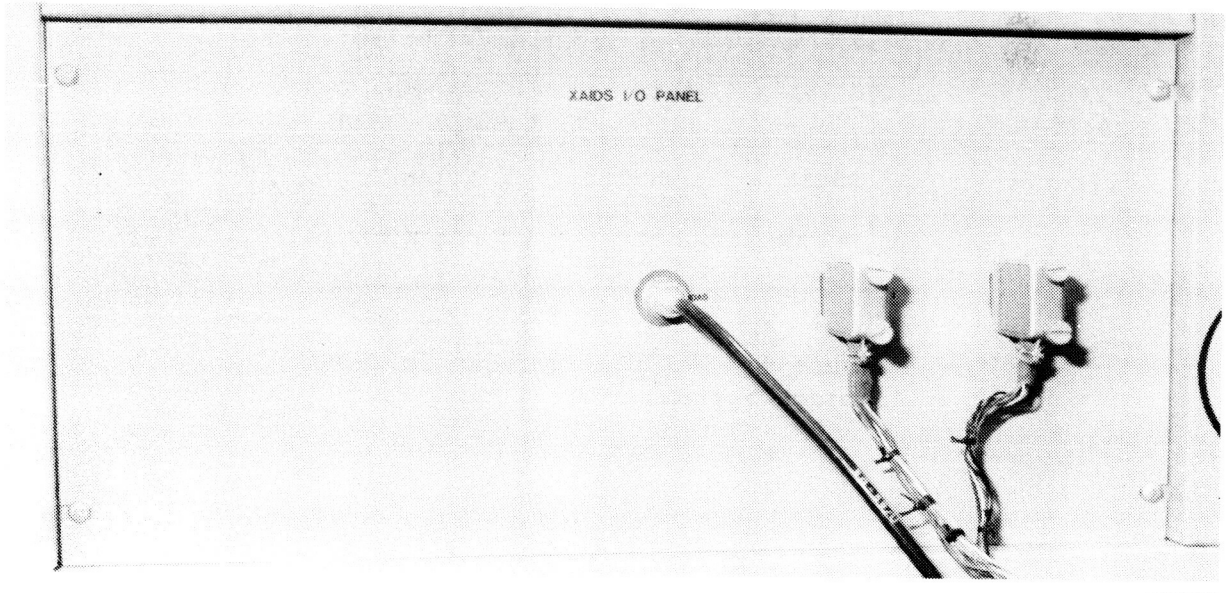


Figure 4. XAIDS input-output panel.

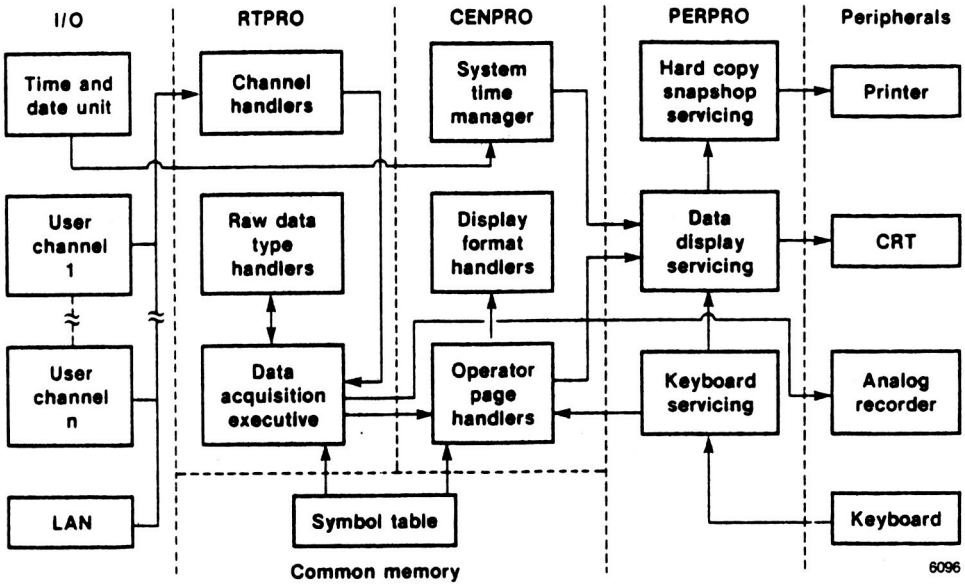


Figure 5. XAIDS functional interfaces.

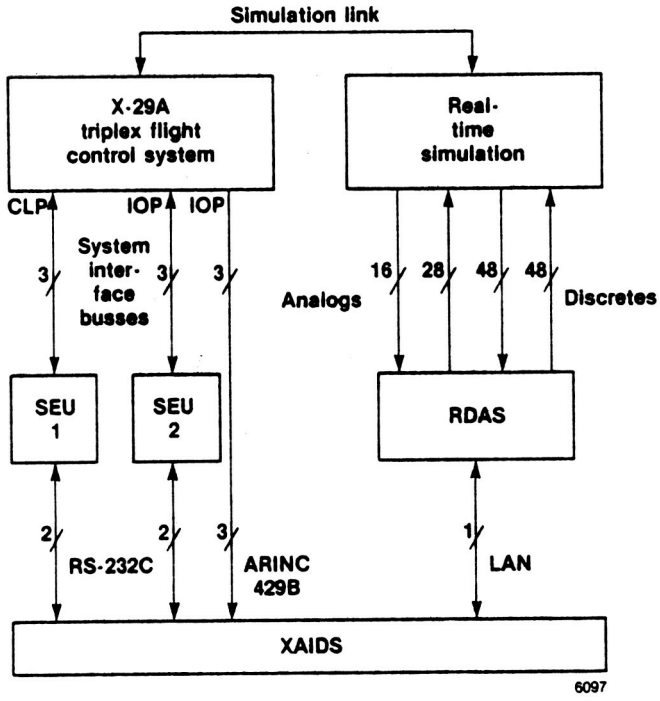
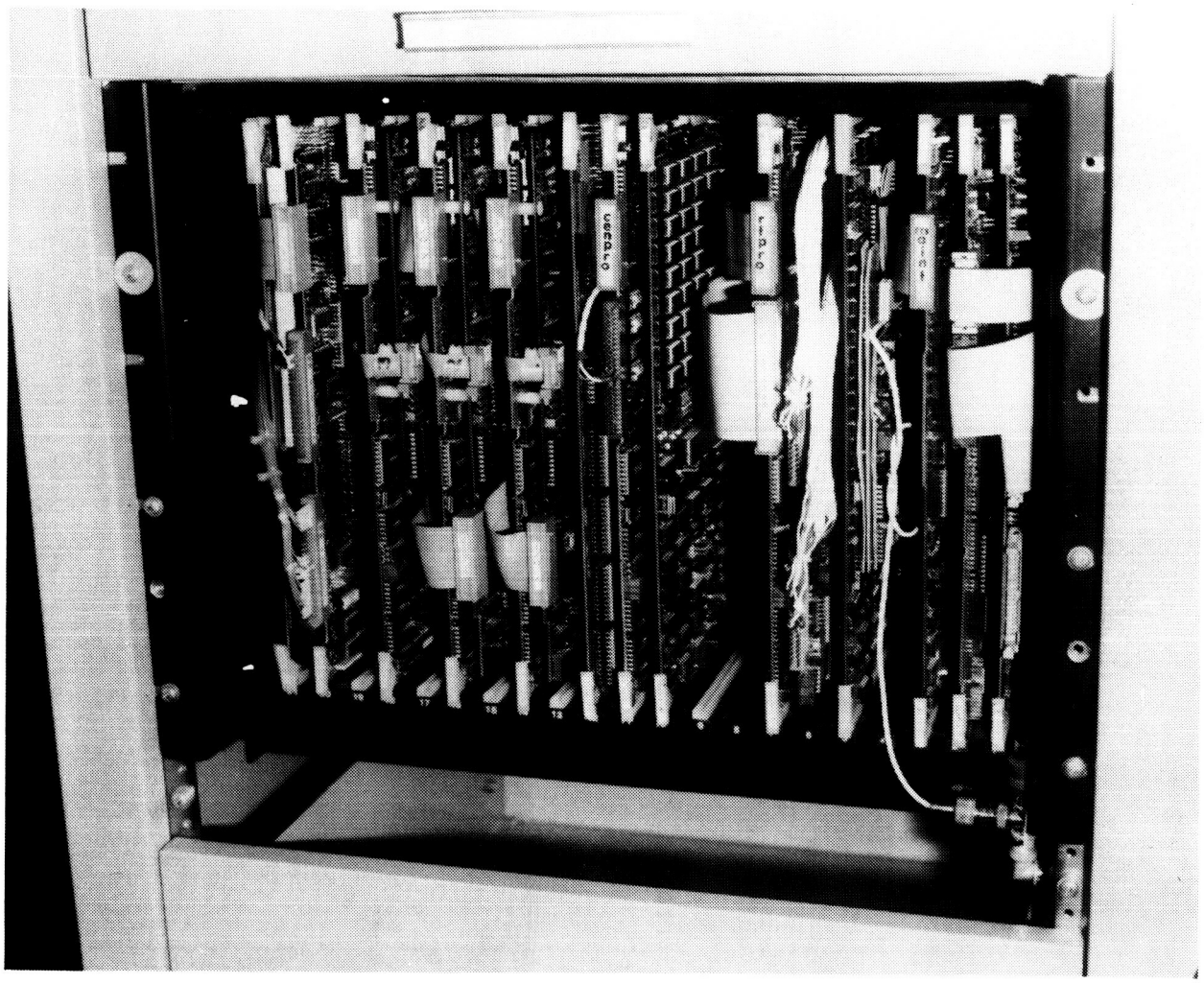


Figure 6. XAIDS interfaces to X-29A project systems.



ECN 31859

Figure 7. XAIDS cardcage.

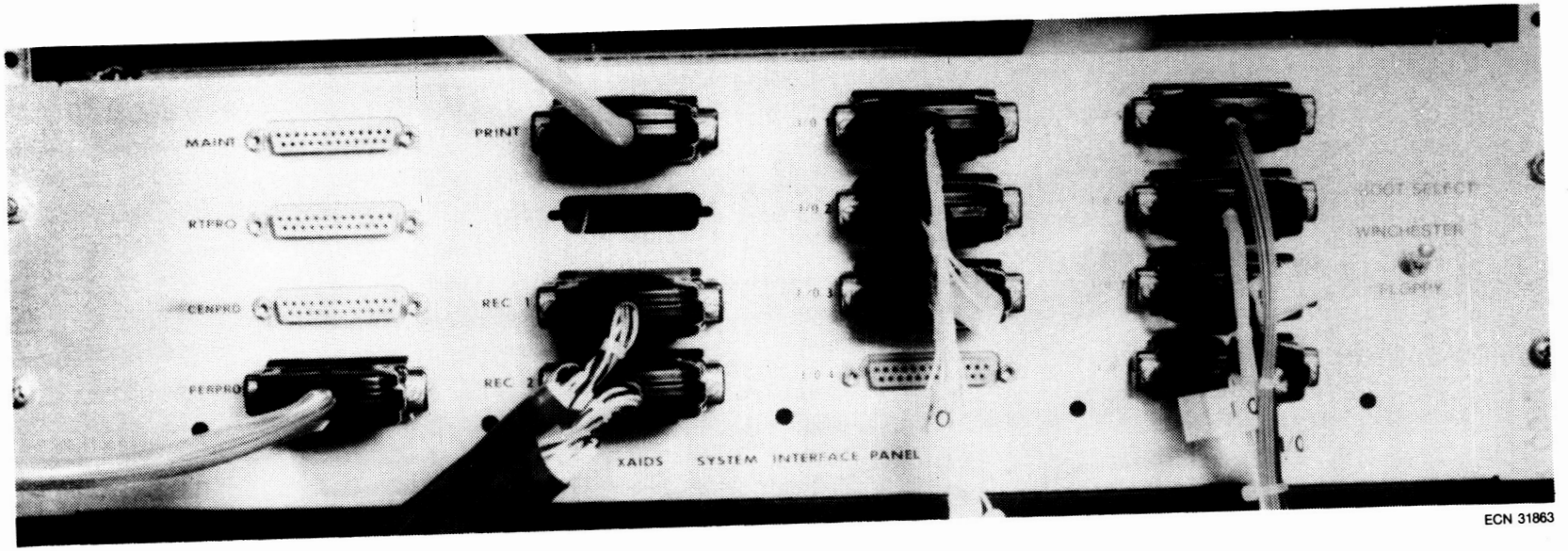


Figure 8. XAIDS system interface panel.

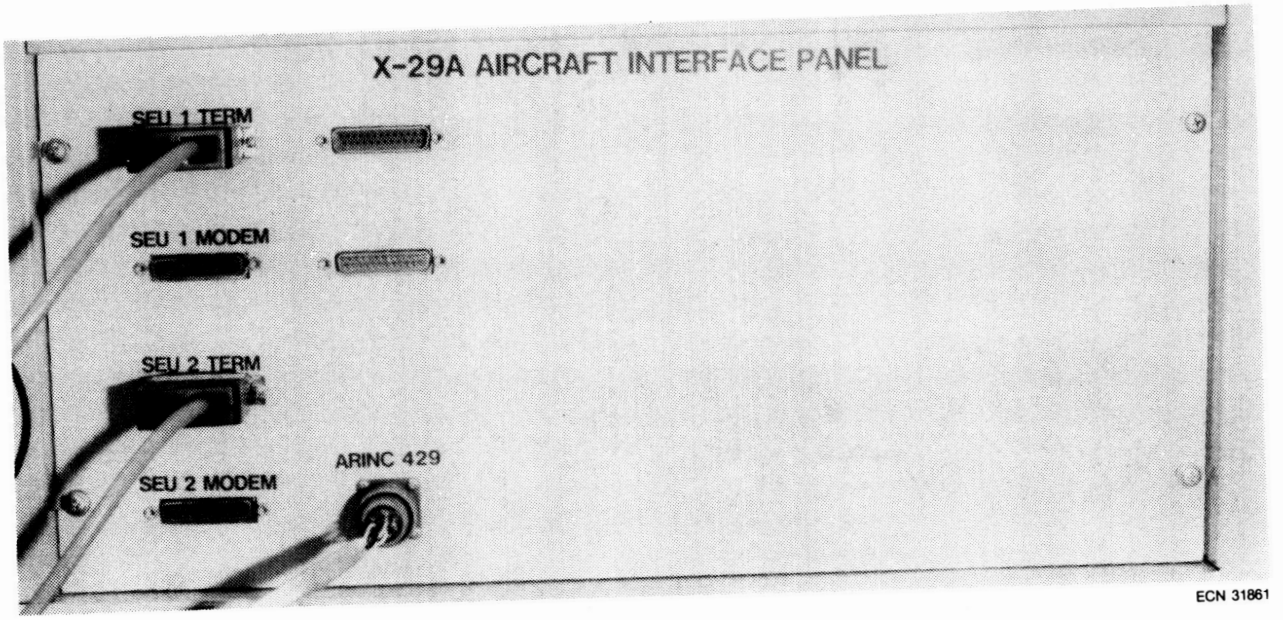
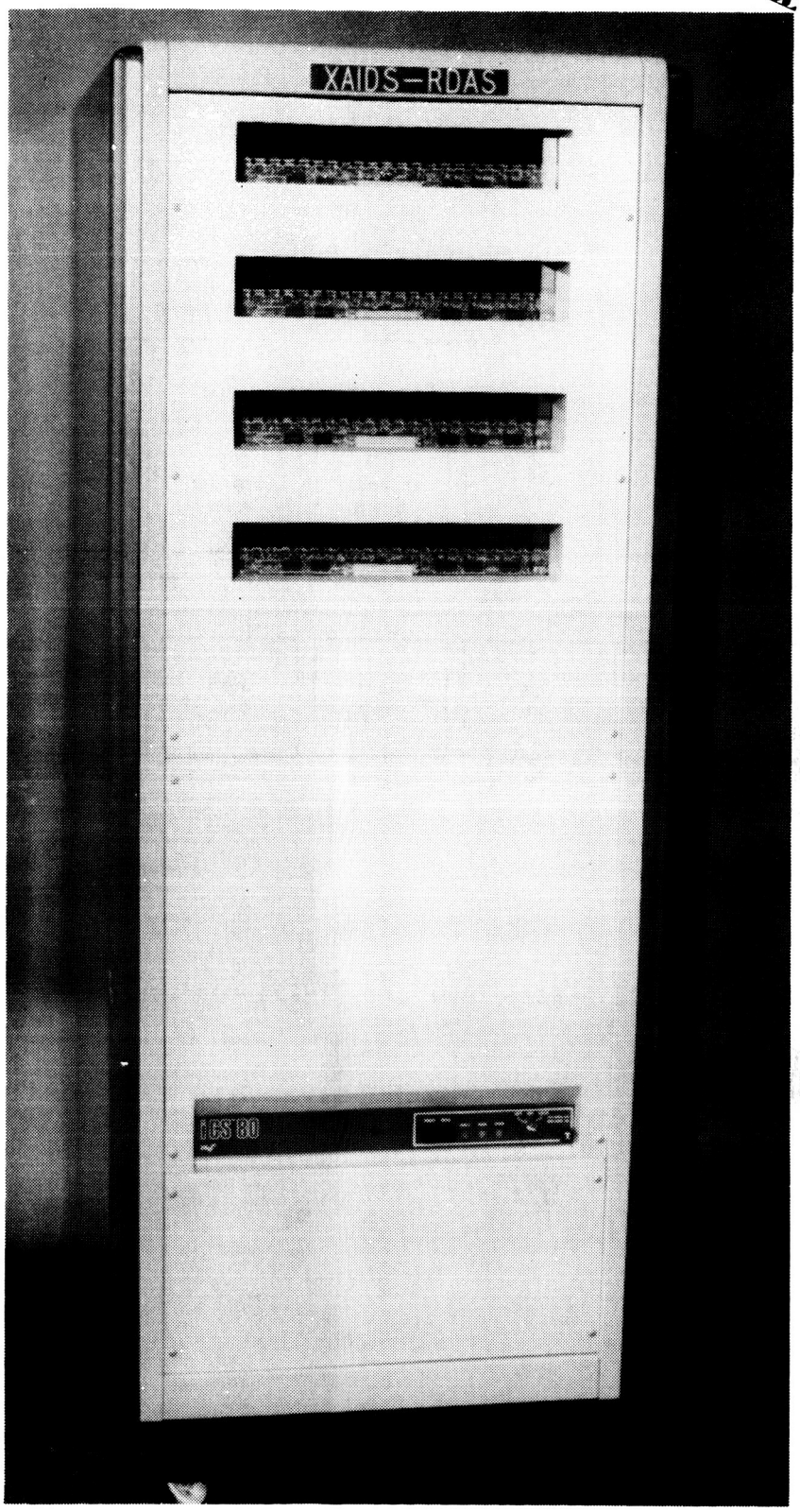


Figure 9. XAIDS X-29A aircraft interface panel.



ECN 31856

Figure 10. Remote data acquisition subsystem.

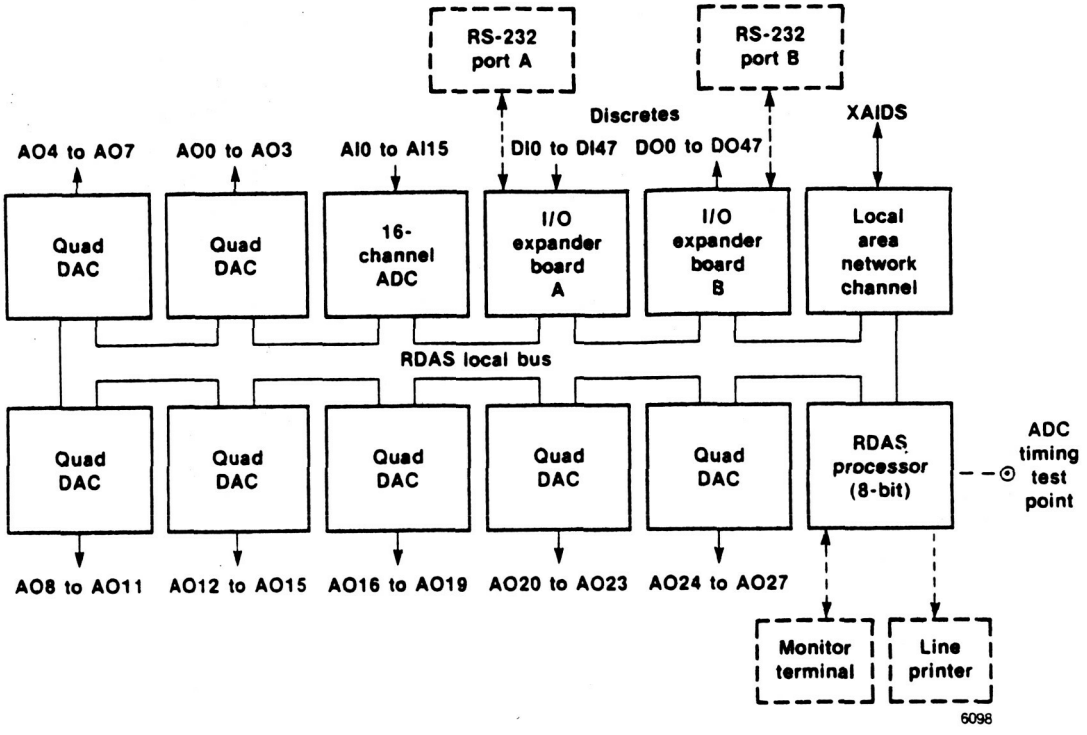
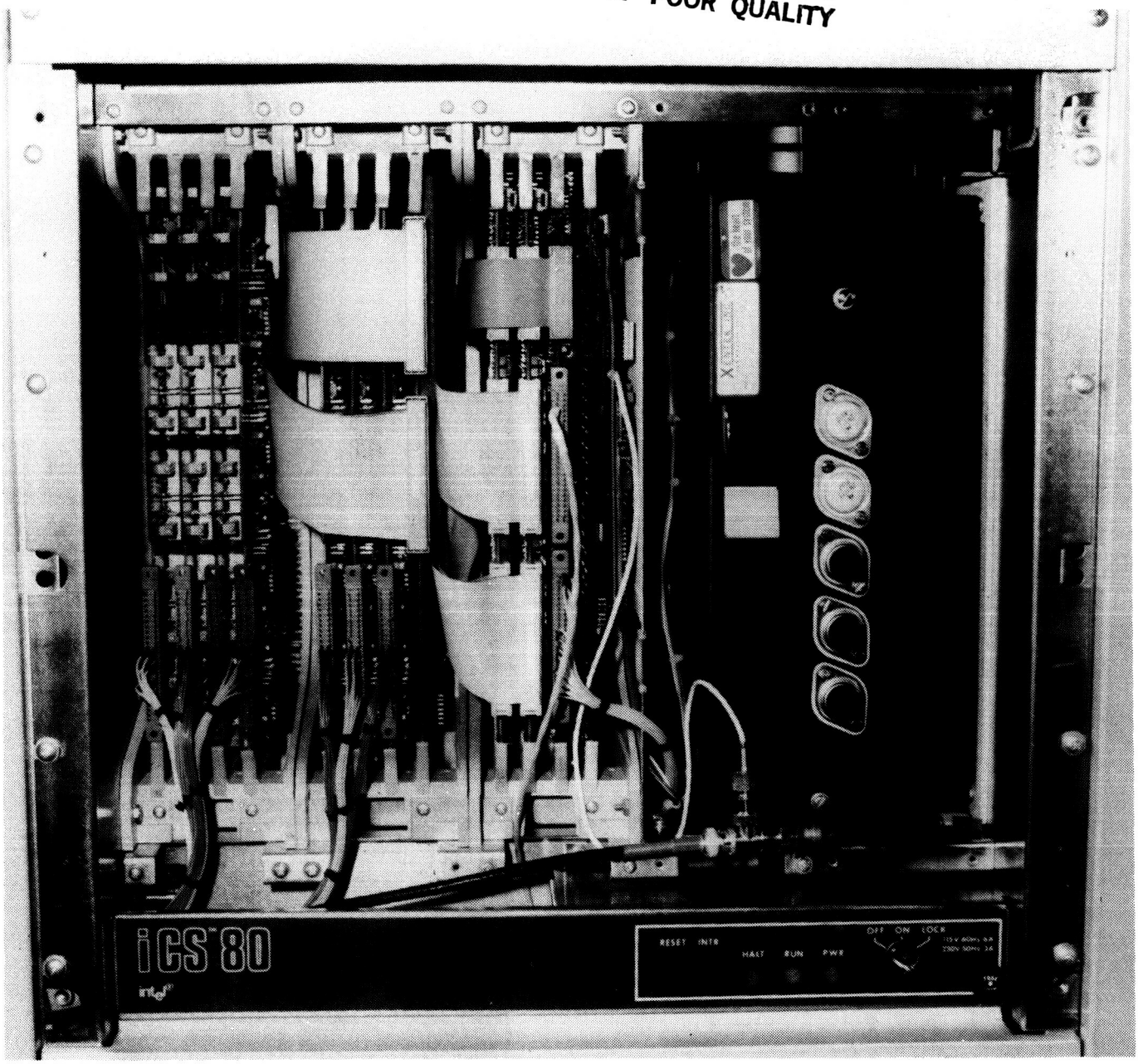


Figure 11. RDAS overview, X-29A configuration.



ECN 31857

Figure 12. RDAS cardcage.

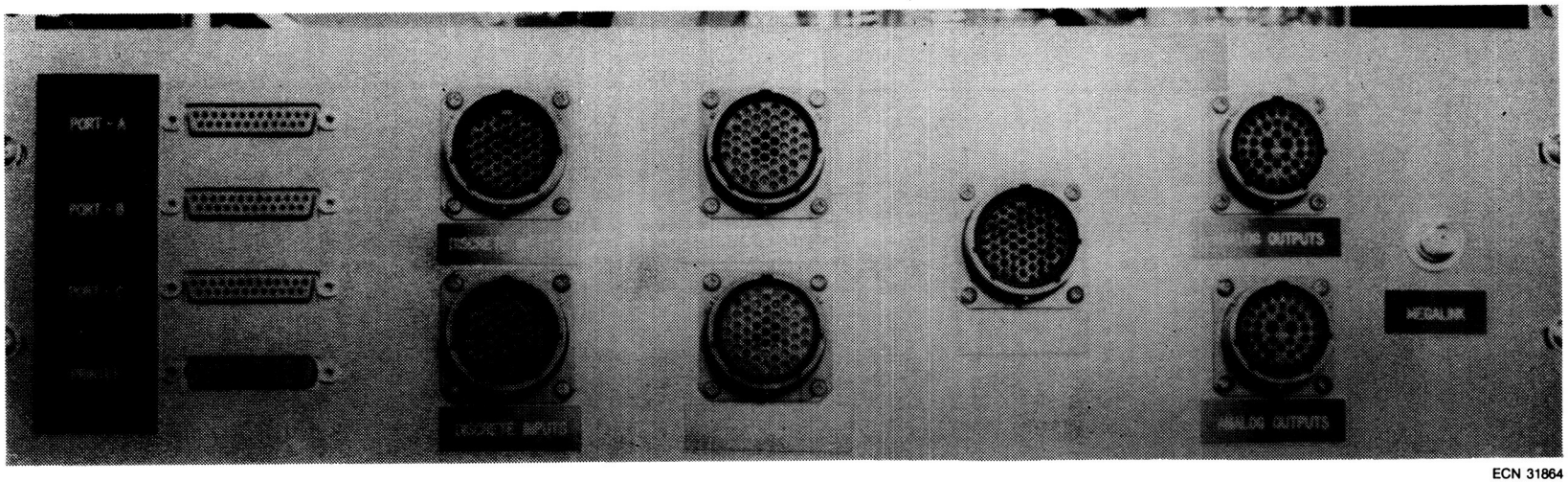


Figure 13. RDAS user input-output panel.

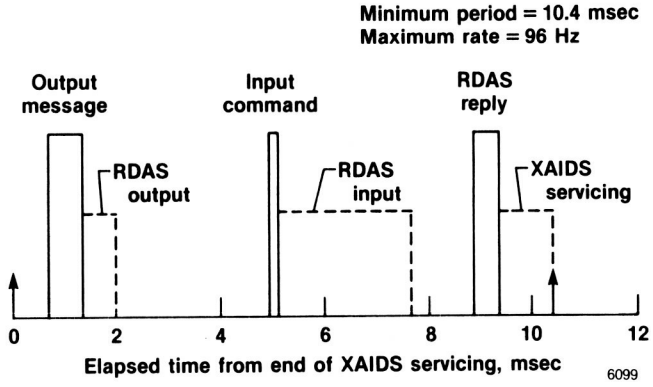


Figure 14. XAIDS-RDAS data exchange.

```
X29A 84.3.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:08:57
XAIDS 84.8.20                  XAIDS COMMAND INTERPRETER                29 NOV 84

COMMANDS

  QT      (QUIT COMMAND)          SET STOP MODE IN CLP ABC & IOP ABC
  RU      (RUN COMMAND)           SET RUN MODE IN CLP ABC & IOP ABC
  SEU1    (SYS EVAL UNIT 1)       TERMINAL EMULATOR FOR SEU NO. 1
  SEU2    (SYS EVAL UNIT 2)       TERMINAL EMULATOR FOR SEU NO. 2
  PTCH    (PATCH MANAGER)         X29A PROCESSORS MEMORY OVERLAY MANAGER
  PMSW    (POST MORTEM SWITCH)    ARINC STALE DATA DISPLAY CONTROL
  GAIN    (GAINS CHECKOUT)        X29A AIR DATA & GAINS CHECKOUT PACKAGE
  MP      (MAKE PAGE)             TABULAR DATA DISPLAY FORMAT
  FF      (FREE FORM)             UNSTRUCTURED DATA DISPLAY FORMAT
  LD      (LOAD DISPLAY)          DISPLAY SCRATCH FILE LOADER
  DK      (DISKETTE)              DISPLAY SCRATCH DISKETTE MANAGER
  SYM     (SYMBOLS)               SYMBOL TABLE MANAGER
  RDFN    (RDAS f(t) GEN)         RDAS DAC/DISCRETE FUNCTION GENERATOR
  TEST    (TEST ROUTINES)         USER & SYSTEM MAINTENANCE PACKAGES
  EXIT    (EXIT XAIDS)            RETURN TO RMX86 EXECUTIVE


ENTER COMMAND )
```

Display 1. Main command menu.

```
X29A 84.3.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:09:04
TEST ROUTINES            USER & SYSTEM MAINTENANCE PACKAGES             29 NOV 84

 TEST

 BAUD    (BAUD RATE)             XAIDS I/O CHANNELS RS-232 CONTROL/CHECKOUT
 ARNC    (ARINC TEST)            XAIDS I/O CHANNELS ARINC CHECKOUT PACKAGE
 RDAS    (REMOTE DATA ACQ SYS)   RDAS FUNCTIONAL TEST ROUTINES
 CON     (CONSTANTS ENCODER)     PLM86 NUMERICAL VALUE GENERATOR
 CAL     (CALIBRATE)             ANALOG RECORDER DAC CALIBRATION
 (esc)   (EXIT TEST)             RETURN TO XAIDS COMMAND INTERPRETER









ENTER TEST )
```

Display 2. TEST command menu.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:01:03
RUN COMMAND                ALL CHANNELS RUN ROUTINE                    29 NOV 84

VERIFYING SEU 1 LINK - - WAIT
/
? CHABC
? BC00,C01
0C00  BC8C  BC8C  BC8C
0C01  45F1  45F1  45F1
? RU
?

VERIFYING SEU 2 LINK - - WAIT
/
? CHABC
? BC00,C01
0C00  F035  F035  F035
0C01  B045  B045  B045
? RU
?

COMPLETE.   WAITING FOR (esc)  )
```

Display 3. RU command display.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:02:36
QUIT COMMAND               ALL CHANNELS STOP ROUTINE                   29 NOV 84

VERIFYING SEU 1 LINK - - WAIT
/
? CHABC
? QT
? BC00,C01
0C00  8C8C  8C8C  8C8C
0C01  45F1  45F1  45F1
?

VERIFYING SEU 2 LINK - - WAIT
/
? CHABC
? QT
? BC00,C01
0C00  F035  F035  F035
0C01  B045  B045  B045
?

COMPLETE.   WAITING FOR (esc)  )
```


Display 4. QT command display.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:23:31
PATCH MANAGER          X29A PROCESSORS MEMORY OVERLAY MANAGER          29 NOV 84


TABLE NAME : IOPARINC - BLK-III-AH-IOP
SAVED :      11:49:03   16 NOV 84
CONTAINS :   64 ENTRIES



COMMAND LIST :

SEND  = WRITE CURRENT PATCH TABLE TO FLIGHT COMPUTER(S)
EDIT  = EXAMINE/MODIFY CURRENT PATCH TABLE
SAVE  = STORE CURRENT PATCH TABLE ON DISKETTE
WIPE  = ERASE CURRENT PATCH TABLE
NAME  = REDEFINE TITLE OF CURRENT PATCH TABLE
LIST  = COPY CURRENT PATCH TABLE TO LINE PRINTER
(ESC) = RETURN TO XAIDS COMMAND INTERPRETER



ENTER COMMAND )
```

Display 5. PTCH command menu.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:24:34
PATCH MANAGER                IOPARINC - BLK-III-AH-IOP                 29 NOV 84
ITEM  ADDR  DATA ------------------------ COMMENTS -------------------------
  21  0CC6  4003 Q
  22  0CC7  4008 NZ
  23  0CC8  4006 NX
  24  0CC9  401C PS
  25  0CCA  4021 PT
  26  0CCB  400D TTI
  27  0CCC  4009 ALPHA
  28  0CCD  4211 ALPHAI
  29  0CCE  0903 ALPHAIR
  30  0CCF  0902 ALPHAIL
  31  0CD0  403A SELSI6
  32  0CD1  0A55 FTDISC2
  33  0CD2  0A56 FTDISC3
  34
  35  0CDC  4218 COMCYCLE
  36  0CDD  400E DCL
  37  0CDE  400F DCR
  38  0CDF  4010 FLLI
  39  0CE0  4012 FLLO
  40  0CE1  4011 FLRI
COMMANDS :  Insert   Modify   Remove   Up   Down   (esc)=Exit   )
```

Display 6. PTCH command edit display.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:33:11
GAINS CHECKOUT        DEFAULT TEST POINT  0.6 / 30K  NORMAL MODE        29 NOV 84
 ITEM  CHAN  VARIABLE NAME & DESCRIPTION     FORMAT          VALUE  UNITS
   1    RP   QCI IMPACT PRESSURE              I6              2415  MIL MG
   2    RP   PSI STATIC PRESSURE              I6              8885  MIL MG
   3    RP   AOAI MEASURED ANGLE OF ATTACK    F6.3           5.180  DEGREES
   4    RP   PF ROLL RATE                     F7.3           0.000  DEG/SEC
   5    RP   UFO WING FLAP                    F5.2            0.00  DEGREES
   6    RP   YCI RUDDER PEDAL INPUT           F5.3           0.000  PER CENT
   7    RP   RFAT TOTAL TEMPERATURE           F6.2          441.27  DEGREES R
   8
   9
  10
  11    RP   GAINDSL NORMAL MODE              L1                T
  12    RP   DRSEL DIGITAL REVSN MODE STATUS  L1                F
  13    RP   ARSEL ANALOG REVSN MODE STATUS   L1                F
  14    RP   DELSEL DIRECT LINK               L1                F
  15    RP   GRHDL GEAR HANDLE                L1                F
  16    RP   ALPBIT AOA FAILURE BIT           L1                F
  17    RP   ABBIT AIR DATA FAILURE BIT       L1                F
  18    RP   FLPFLG FLAP FLAG FUNCTION        I2                4
  19    RP   HARDRG AR/DR GAIN TABLE SELECT   I1                2
  20
Compute  Help  Init  Modify  Save  Title  Wrapup  (esc)=Exit  )
```

Display 7. GAIN command display.

```
                EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:33:26
                   GAIN PAGE OPERATOR ASSISTANCE INFORMATION          29 NOV 84

CMD

 C  FORCES XAIDS RECOMPUTATION OF AIR DATA & GAINS (SINGLE ITERATION).

 H  DISPLAYS OPERATOR ASSISTANCE DISPLAY.

 I  INITIALIZES INPUT VALUES IN IOPs AND CLPs .

 M  PERMITS MODIFICATION OF INITIAL VALUE FOR A SINGLE LINE ITEM, AFTER
    WHICH NEW AIR DATA & GAIN VALUES ARE AUTOMATICALLY COMPUTED BY XAIDS.

 S  SAVES CURRENT INITIAL VALUES TABLE ON SCRATCH DISKETTE.

 T  PERMITS NEW TITLE TO BE ENTERED.

 W  SIGNALS BEGINNING OF WRAPUP OPERATIONS INCLUDING A PROGRAMMED STOP
    OF THE 3 CLPs FOLLOWED BY DUMP OF THE OUTPUT VALUES VIA THE SEUs.

(ESC) KEY CAUSES RETURN TO XAIDS COMMAND LINE INTERPRETER.

WAITING FOR (esc)  )
```

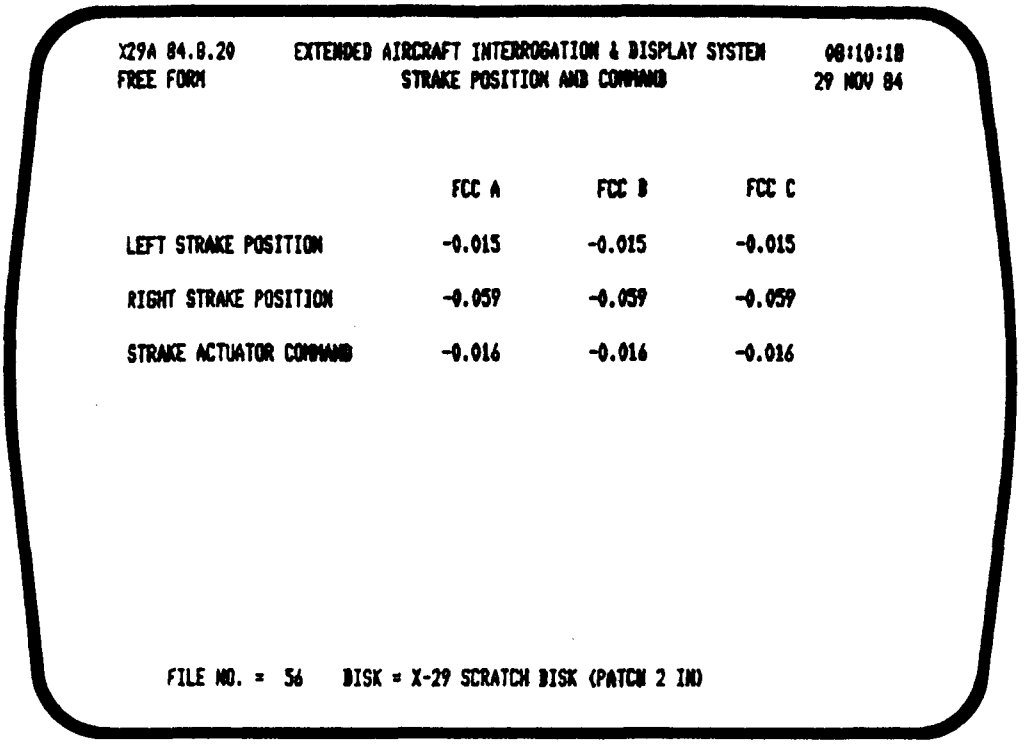
Display 8. GAIN command help page.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:18:17
MAKE PAGE                      FMET TEST ALPHA FAILURES                29 NOV 84
 ITEM  CHAN  VARIABLE NAME & DESCRIPTION    FORMAT          VALUE  UNITS
   1    A1   ALPHAI Loc Ind Angle of Attack F10.3          3.345  DEGREES
   2    A2   ALPHAI Loc Ind Angle of Attack F10.3         14.985  DEGREES
   3    A3   ALPHAI Loc Ind Angle of Attack F10.3          3.598  DEGREES
   4    A2   ALPHA True Angle of Attack     F10.3          0.000  DEGREES
   5
   6    A1   MACH Mach Number               F10.3          0.590
   7    A2   MACH Mach Number               F10.3          0.590
   8    A3   MACH Mach Number               F10.3          0.590
   9
  10    A1   ALT (Patch #2, was RAV A/I #1) F10.3       4831.558  FEET
  11    A2   ALT (Patch #2, was RAV A/I #1) F10.3       4831.558  FEET
  12    A3   ALT (Patch #2, was RAV A/I #1) F10.3       4831.558  FEET
  13
  14    A1   DW06_B07 Air Input Fail (local) M10        ......FAIL
  15    A2   DW06_B07 Air Input Fail (local) M10        ......FAIL
  16    A3   DW06_B07 Air Input Fail (local) M10        ......FAIL
  17
  18
  19
  20
     FILE NO. = 45    DISK = X-29 SCRATCH DISK (PATCH 2 IN)
```

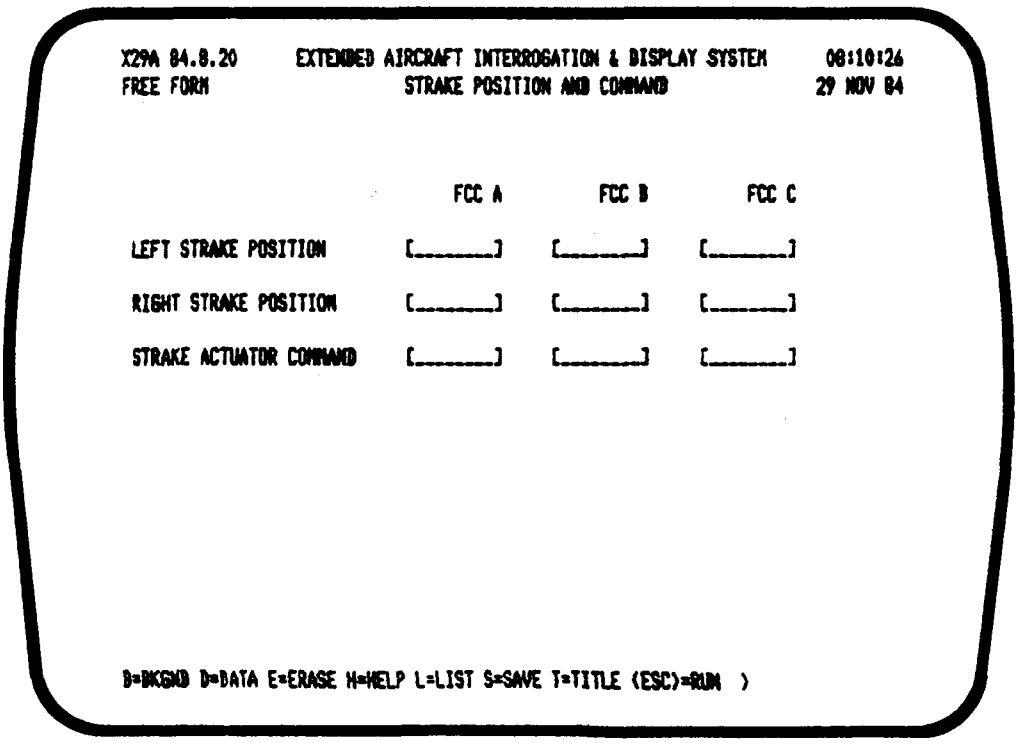
Display 9. Example MP-driven display.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:18:29
MAKE PAGE              MAKE PAGE OPERATOR ASSISTANCE INFORMATION         29 NOV 84

CMD
 E  ERASES ENTIRE TEMPLATE INCLUDING PAGE TITLE.
 H  DISPLAYS OPERATOR ASSISTANCE DISPLAY.
 I  PERMITS INSERTION, DELETION, OR ALTERATION OF A SINGLE LINE ITEM.
    CHAN ) TWO-LETTER CODE FOR SOURCE CHANNEL OR TWO BLANKS TO DELETE ITEM
    VARIABLE ) NAME OF PARAMETER TO BE DISPLAYED
    NOTE : USER IS GIVEN OPTION OF ACCEPTING THE DEFAULT DISPLAY FORMAT AS
           SPECIFIED IN THE SYMBOL TABLE OR SUPPLYING AN ALTERNATE FORMAT.
 L  LATCHES ITEMS 1 THRU 16 INTO ANALOG RECORDER OUTPUT DRIVERS.
 S  SAVES CURRENT TEMPLATE ON SCRATCH DISKETTE.  USER MAY SPECIFY WHICH FILE
    NUMBER IS WRITTEN TO OR MAY ALLOW SYSTEM TO SELECT EMPTY FILE.
 T  ALLOWS USER TO SPECIFY PAGE TITLE OF UP TO 48 CHARACTERS IN LENGTH.
 U  UNLATCHES ANALOG RECORDER OUTPUT DRIVERS AND FORCES PENS TO ZERO.
    NOTE : INDEPENDENT OF PAGE USED TO PERFORM ORIGINAL LATCH OPERATION.
(ESC) KEY CAUSES DISPLAY TO ENTER RUN MODE.  WHILE IN RUN MODE, OPTIONS ARE :
        HALT DISPLAY AND RE-ENTER SETUP MODE BY PRESSING (RETURN) KEY OR
        EXIT DISPLAY AND INVOKE XAIDS COMMAND MENU BY PRESSING (ESC) KEY.



PRESS (ESC) TO EXIT.
```

Display 10. MP command help page.



Display 11. Example FF-driven display.



Display 12. Example FF setup page.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:10:35
FREE FORM            FREE FORM OPERATOR ASSISTANCE INFORMATION         29 NOV 84
CMD
 B  PERMITS CREATION OF STATIC BACKGROUND IMAGE TO ACCOMPANY DATA.
    NOTE : ALL CHARACTERS ALLOWED INCLUDING SPECIAL CHARACTERS.
 D  PERMITS INSERTION, ALTERATION, OR DELETION OF DATA ITEM.  FOR INSERTIONS
    AND ALTERATIONS, CURSOR MUST BE POSITIONED AT LEFT END OF FIELD.
    FOR DELETIONS, CURSOR MAY BE POSITIONED ANYWHERE WITHIN BRACKETS.
    CHAN ) TWO-LETTER CODE FOR SOURCE CHANNEL OR TWO BLANKS TO DELETE ITEM.
    VARIABLE ) NAME OF PARAMETER TO BE DISPLAYED
    NOTE : USER IS GIVEN OPTION OF ACCEPTING THE DEFAULT DISPLAY FORMAT AS
           SPECIFIED IN THE SYMBOL TABLE OR SUPPLYING AN ALTERNATE FORMAT.
 E  ERASES ENTIRE TEMPLATE INCLUDING PAGE TITLE.
 H  DISPLAYS OPERATOR ASSISTANCE DISPLAY.
 L  LISTS DATA ITEMS IN CURRENT TEMPLATE TO EITHER CRT OR LINE PRINTER.
 S  SAVES CURRENT TEMPLATE ON SCRATCH DISKETTE.  USER MAY SPECIFY WHICH FILE
    NUMBER IS WRITTEN TO OR MAY ALLOW SYSTEM TO SELECT EMPTY FILE.
 T  ALLOWS USER TO SPECIFY PAGE TITLE OF UP TO 48 CHARACTERS IN LENGTH.
(ESC) KEY CAUSES DISPLAY TO ENTER RUN MODE.  WHILE IN RUN MODE, OPTIONS ARE :
        HALT DISPLAY AND RE-ENTER SETUP MODE BY PRESSING (RETURN) KEY OR
        EXIT DISPLAY AND INVOKE XAIDS COMMAND MENU BY PRESSING (ESC) KEY.
GENERAL NOTE : CURSOR POSITIONING USES FOUR ARROW KEYS PLUS (HOME), (TAB),
               AND (RETURN) KEYS.  RANGE RESTRICTED TO ROWS 3 THRU 23.
PRESS (ESC) TO EXIT.
```

Display 13. FF command help page.


```
                    EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:10:52
                                                                          29 NOV 84



LIST DATA TEMPLATE TO "CRT" OR "PTR" ? ) CRT

  SEQ NO.   ROW   COL   FORMAT   CHAN   NAME & DESCRIPTION
     0       8    30    F10.3    A1    LSS Lt Strake Position
     1       8    45    F10.3    A2    LSS Lt Strake Position
     2       8    60    F10.3    A3    LSS Lt Strake Position
     3      10    30    F10.3    A1    RSS Rt Strake Position
     4      10    45    F10.3    A2    RSS Rt Strake Position
     5      10    60    F10.3    A3    RSS Rt Strake Position
     6      12    30    F10.3    A1    PSTCMD+6 Strake Act Command
     7      12    45    F10.3    A2    PSTCMD+6 Strake Act Command
     8      12    60    F10.3    A3    PSTCMD+6 Strake Act Command


END OF TEMPLATE
```

Display 14. Example FF command list to CRT.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM     08:21:59
DISKETTE               XAIDS DISPLAYS SCRATCH DISKETTE MANAGER         29 NOV 84




CMDS   DISPLAY SCRATCH DISKETTE OPERATION                 DRIVE
V      VIEW DISPLAYS DIRECTORY                            2 or 3
LD     LOAD & ACTIVATE DISPLAY FILE                       3 only
DEL    DELETE DISPLAY FILE                                2 or 3
LIST   WRITE DIRECTORY TO PRINTER                         2 or 3
DUMP   SEQUENTIAL DUMP OF SCRATCH FILES TO PRINTER        3 only
INIT   INITIALIZE NEW SCRATCH DISKETTE                    2 or 3
COPY   COPY SELECTED DISPLAY FILE FROM DISK TO DISK       3 to 2
(ESC)  RETURN TO XAIDS COMMAND INTERPRETER


ENTER COMMAND ) V
```

Display 15. DK command menu.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM     08:22:09
DISKETTE               XAIDS DISPLAYS SCRATCH DISKETTE MANAGER         29 NOV 84


DIRECTORY NAME :  X-29 SCRATCH DISK (PATCH 2 IN)

CREATED :         09:40:08    8 JUN 84

NUMBER OF FILES : 64


 41  13:58:06  11 JUN 84  PTCH  REMOVE PATCH 84 - IOP
 42  14:02:14  11 JUN 84  PTCH  REMOVE PATCH 84 - CLP
 43  11:15:12  19 JUN 84  MP    LONGSSN INPUTS
 44  11:16:02  19 JUN 84  MP    LATSSN INPUTS
 45  09:50:53  12 JUL 84  MP    FMET TEST ALPHA FAILURES
 46  09:59:48  12 JUL 84  MP    FMET TESTS QC FAILURES
 47  10:02:40  12 JUL 84  MP    FMET TEST TOTAL TEMP FAILURES
 48  10:08:44  12 JUL 84  MP    FMET TESTS MON FAILURES(DEL MODE)
 49  10:11:38  12 JUL 84  MP    FMET TESTS AHRS FAILURES
 50  10:12:36  12 JUL 84  MP    FMET TESTS PS FAILURES


Scroll using U, D, or arrow keys.  (ESC) to exit.
```

Display 16. Example DK view display.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:18:47
SYMBOLS                        SYMBOL TABLE MANAGER                    29 NOV 84


TABLE NAME : X-29A ARINC Data Stream Symbol Definition (Patch 02 in)
FILE SAVED : 13:51:20   28 NOV 84
OWNED BY :   SYM
CONTAINS :   230 SYMBOLS


SYMBOL TABLE MANAGER COMMAND LIST :

SCAN  = VIEW CURRENT TABLE
EDIT  = EXAMINE/MODIFY TABLE ENTRIES
SAVE  = WRITE CURRENT TABLE TO DISKETTE
LOAD  = LOAD NEW TABLE FROM DISKETTE
WIPE  = ERASE ENTIRE TABLE
NAME  = REDEFINE SYMBOL TABLE NAME
LIST  = COPY TABLE TO LINE PRINTER
(ESC) = RETURN TO XAIDS COMMAND INTERPRETER



ENTER COMMAND )
```

Display 17. SYM command menu.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM      08:20:42
SYMBOLS                        SYMBOL TABLE MANAGER                    29 NOV 84

TABLE NAME : X-29A ARINC Data Stream Symbol Definition (Patch 02 in)

ITEM  ------- SUBFIELD --------  ---------- CONTENTS ----------
  1   SYMBOL NAME & DESCRIPTION  ALPHAI Loc Ind Angle of Attack
  2   DISPLAY FORMAT             F10.3
  3   UNITS                      DEGREES
  4   RELATED CHANNELS           A1 A2 A3
  5   RAW DATA TYPE              HSP
  6   NO. OF MEMORY LOCATIONS    1                              MODE :
  7   STARTING ADDRESS           0000005A                       COMMAND
  8   INVERTING CONTROL          0000000000000000
  9   MASKING CONTROL            FFFFFFFFFFFFFFFF               STATUS :
 10   DISPLAY ZERO                   0.000                      LINKED
 11   DISPLAY MAX                  180.000
 12   DISCRETE ZERO MESSAGE                                     LINKS :
 13   DISCRETE ONE MESSAGE                                      UP  00CE
 14   RECORDER BIAS                  0.000                      EN  00CF
 15   RECORDER SCALE               180.000                      DN  00C7

EDIT OPERATIONS : New  Modify  Clone  Find  Up  Down  Remove  (ESC)=EXIT
SELECT OPERATION )
```

Display 18. Example SYM edit page.

```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM     08:30:02
BAUD RATE              XAIDS X29A IOP RS-232 SERIAL PORT CHECKOUT      29 NOV 84

CURRENT XAIDS I/O PROCESSOR RS-232 PORT STATUS :

USART  ASSIGNMENT     DEVICE          BAUD RATE     XMT QUEUE     RCV QUEUE
  4    SPARE          IOP C PORT 1        0             0             0
  5    SEU #1 TERM    IOP A PORT 1     2400             0             0
  6    SEU #1 MODEM   IOP A PORT 2     2400             0             0
  7    SEU #2 TERM    IOP B PORT 1     2400             0             0
  8    SEU #2 MODEM   IOP B PORT 2     2400             0             0

B=BAUD RATE CHANGE  S=SEND MSG  R=DISPLAY RECEIVED MSGS  (ESC)=EXIT ) B

SELECT USART (4-8) ) 5
ENTER DESIRED BAUD RATE (75-19200) ) 4800
```

Display 19. BAUD command display.


```
X29A 84.8.20      EXTENDED AIRCRAFT INTERROGATION & DISPLAY SYSTEM     08:31:32
CALIBRATE              RTP80 RECORDER DAC CALIBRATE JOB                29 NOV 84

COMMAND LIST :

AUTO  = 0, -4, -5, +4, +5 AUTO STEP PATTERN
SAW   = ONE HERTZ SAWTOOTH PATTERN
MAN   = 0, -4, -5, +4, +5 MANUAL STEP PATTERN
SEL   = SELECT INDIVIDUAL DAC & VOLTS
TEST  = PERFORM TEST PROGRAM & RETURN STATUS
(ESC) = RETURN TO RMX86 EXECUTIVE

SELECTED PEN = 1
CURRENT PEN VALUE = 0.000
ENTER PEN OUTPUT (-5 TO +5 VOLTS) ) 2.5
```

Display 20. CAL command display.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86740		

4. Title and Subtitle	5. Report Date
Design and Initial Application of the Extended Aircraft Interrogation and Display System — Multiprocessing Ground Support Equipment for Digital Flight Systems	January 1987
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Richard D. Glover	H-1296
	10. Work Unit No.
9. Performing Organization Name and Address	RTOP 533-02-51
NASA Ames Research Center Dryden Flight Research Facility P.O. Box 273 Edwards, CA 93523-5000	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Contact author for further information on software.

16. Abstract

A pipelined, multiprocessor, general-purpose ground support equipment for digital flight systems has been developed and placed in service at the NASA Ames Research Center's Dryden Flight Research Facility. The design is an outgrowth of the earlier aircraft interrogation and display system (AIDS) used in support of several research projects to provide engineering-units display of internal control system parameters during development and qualification testing activities. The new system, incorporating multiple 16-bit processors, is called extended AIDS (XAIDS) and is now supporting the X-29A forward-swept-wing aircraft project. This report describes the design and mechanization of XAIDS and shows the steps whereby a typical user may take advantage of its high throughput and flexible features.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Aircraft simulation Aircraft test equipment Data acquisition Data display	Unclassified — Unlimited Subject category 05

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	96	A05

For sale by the National Technical Information Service, Springfield, Virginia 22161.