NASA Contractor Report **182006**

IAPSA II Small-Scale System Specification

G. C. Cohen
Boeing Advanced Systems
Seattle, Washington

T. C. Torkelson
Boeing Advanced Systems
Seattle, Washington

NASA Contract NAS1-18099
March 1990

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23665

(NASA-CR-182006) IAPSA 2 SMALL-SCALE SYSTEM SPECIFICATION Final Report (Boeing Advanced Systems Co.) 316 p CSCL 12B

N90-24105

Unclas
G3/39 0287444

PREFACE

This report describes the IAPSA II Small-Scale System Specification. This work was supported under NASA contract NAS1-10899, Integrated Airframe/Propulsion Control System Architecture (IAPSA II).

The NASA technical monitor for this work is Daniel L. Palumbo, of the NASA Langley Research Center, Hampton, Virginia.

The work was accomplished by the Flight Controls Technology organization of Boeing Advanced Systems in Seattle, Washington. Personnel responsible for the work performed include:

D. Gangsaas	Responsible Manager
T. M. Richardson	Program Manager
G. C. Cohen	Principal Investigator
T. C. Torkelson	Flight Controls Technology

TABLE OF CONTENTS

		PAGE
1.0	SUMMARY	1
2.0	INTRODUCTION	3
3.0	DISCUSSION	9
3.1	General Simulation Host Hardware Description	9
3.1.1	VMEbus CPU Card	9
3.1.2	Bulk VMEbus Memory	9
3.1.3	VME-MicroVAX Interface	9
3.1.4	DIU Simulators	13
3.1.5	I/O Network Fault Insertion	15
3.1.6	Experiment Control and Experiment Bus	15
3.1.7	VMEbus Experiment Time and Fault Insertion Delay	16
3.2	Development Support Hardware	16
3.2.1	Development Host Computer	16
3.2.2	Terminal Server	16
3.2.3	Line Printer	17
3.2.4	Logic Analyzer	17
3.2.5	Microprocessor Emulators	17
3.3	General Software Description	17
3.3.1	Experiment Software Interaction	17
3.3.2	Experiment Host Software	19
3.3.3	Simulation Host Software	20
3.3.4	DIU Simulator Software	21
3.3.5	AIPS FTP Software	21
3.3.6	Development Host Support Software	21
3.4	Hardware Modifications to AIPS Building Blocks	22

PRECEDING PAGE BLANK NOT FILMED PAGE ii INTENTIONALLY BLANK

TABLE OF CONTENTS (Continued)

		PAGE
3.5	Simulation Host Hardware Details	22
3.5.1	VMEbus System Configuration	22
3.5.2	Experiment Host - Simulation Host Interface	24
3.5.3	VMEbus Backplane Modifications	33
3.5.4	DIU Simulator	35
3.5.5	I/O Network Fault Inserter	48
3.5.6	Miscellaneous Wire Wrap Board Functions	52
3.5.7	VMEbus Computer Timekeeping	54
3.6	Software Details	54
3.6.1	AIPS FTP System Services	54
3.6.2	AIPS FTP Pseudoapplications	55
3.6.3	VME System Kernel and Utilities	55
3.6.4	DIU Kernel	56
3.6.5	DIU Simulator	56
3.6.6	I/O Network Probe	58
3.6.7	DIU Data Formatting	59
3.6.8	Fault Insertion Control	59
3.6.9	VME Experiment Control	59
3.6.10	MicroVAX Interface Software	60
3.6.11	Experiment Control Command Files	60
REFERENCES		62
APPENDIX A:	SMALL-SCALE SYSTEM EXPERIMENT SYNCHRONIZATION	A-1
APPENDIX B:	AIPS I/O NETWORK INTERFACE REQUIREMENTS	B-1
APPENDIX C:	SMALL-SCALE SYSTEM DIU SIMULATOR	C-1

TABLE OF CONTENTS (Continued)

		PAGE
APPENDIX D:	SMALL-SCALE SYSTEM NETWORK FAULT INSERTION REQUIREMENTS	D-1
APPENDIX E:	SMALL-SCALE SYSTEM NETWORK/DIU CONFIGURATION	E-1
APPENDIX F:	SMALL-SCALE SYSTEM I/O NETWORK TRANSACTIONS	F-1
APPENDIX G:	EXPERIMENT BUS DESCRIPTION	G-1
APPENDIX H:	VULTURE PROGRAM DETAILS	H-1
APPENDIX I:	SOFTWARE TAPE LISTING	I-1
APPENDIX J:	DOCUMENTATION PACKAGES	J-1
Documentation Package A:	VMEbus Simulation Computer Configuration	J-1
Documentation Package B:	OPIO-1 Parallel Interface Modification	J-9
Documentation Package C:	VMEbus-MicroVAX Parallel Interface Adapter	J-23
Documentation Package D:	ISIO-2/DIU Simulator Daughter Board	J-29
Documentation Package E:	Fault Insertion and Control Wire Wrap Board	J-97

LIST OF FIGURES

FIGURE		PAGE
2.0-1	Experiment Test Configuration	4
2.0-2	Small-Scale System Block Diagram	5
3.1-1	CPU-29 and Memory Block Diagram	10
3.1-2	OPIO-1 and MicroVAX Interface Block Diagram	11
3.1-3	Wire Wrap Board-Network Fault Insertion/ Experiment Control Block Diagram	12
3.1-4	DIU Simulator Block Diagram	14
3.3-1	Small-Scale System Software Block Diagram	18
3.5-1	VMEbus System Overview, Small-Scale System	23
3.5-2	Experiment Host-Simulation Host Interface Detailed Block Diagram	25
3.5-3	VMEbus and MicroVAX Byte Stacking Order	28
3.5-4	Data Transfer Without Hardware Byte Swapping	30
3.5-5	Data Transfer With Hardware Byte Swapping	30
3.5-6	Experiment Host to Simulation Host Transfer	31
3.5-7	Simulation Host to Experiment Host Transfer	34
3.5-8	DIU Simulator Detailed Block Diagram	36
3.5-9	Block Diagram RX Clock EPLD	38
3.5-10	HDLC Input Edge Detect and Deglitch State Machine	40
3.5-11	RX Clock EPLD Timing Diagram	41
3.5-12	Sync Enable State Machine	42
3.5-13	RX Clock Generator State Machine	43
3.5-14	Flag Shutdown State Machine	45
3.5-15	Fault Insertion, Control Interface Detailed Block Diagram	49
3.5-16	Network Fault EPLD Block Diagram	50

1.0 SUMMARY

This document presents the specifications used to implement hardware and software for those portions of the IAPSA II small-scale system supplied by Boeing. Portions of the system provided by the Charles Stark Draper Laboratory (CSDL) are not included.

A small-scale system was implemented to embody the essential characteristics of a flight-critical system modeled earlier in the IAPSA-II contract. It was used to investigate the critical issues identified by those efforts in both normal and faulted operation.

The system under test was composed of existing proof-of-concept AIPS building-block hardware and software plus simulated device interface units (DIU). The entire system was controlled from a MicroVAX II experiment host computer.

Commercially available VMEbus building-block hardware and software were used to create the simulation host and DIUs. Hardware used to inject faults into the system I/O networks was built on a VMEbus wire wrap card and controlled by an off-the-shelf VMEbus parallel I/O card.

Pseudoapplication Ada software was used to simulate the computational loading of the FTP processors. Dummy data representing the total volume of flight control traffic was sent over the I/O network. DIU dummy response data were sent over the I/O network to answer dummy command data.

The revision level of these specifications reflects the system delivered to NASA Langley Airlab for testing.

2.0 INTRODUCTION

This document presents the details necessary to implement the small-scale system experiment test configuration shown in figure 2.0-1. Off-the-shelf components were used wherever possible to minimize development cost. All discussions pertain to the small-scale system used for experimentation at NASA Langley facilities.

A brief description of off-the-shelf components is provided to help understand system operation. More detailed information on building blocks may be found in manufacturers' specifications and operation manuals, referenced at the end of this document.

The discussion section that follows focuses on the details of modifications to the standard building blocks and on implementation of custom interfaces.

Specifications in the appendixes were derived from meetings and telephone conversations with CSDL personnel, Advanced Information Processing System (AIPS) schematics, Ada software templates supplied by CSDL, preliminary AIPS documentation, and discoveries made during integration testing at CSDL.

System Under Test Description. The FTP shown in figures 2.0-1 and 2.0-2 is a triplex, bit synchronous 68010-based AIPS building block. Each of its three channels has a computational processor (CP), an input/output processor (IOP), local memory, shared memory (SM), one or more input/output sequencers (IOS) to connect the fault-tolerant processor (FTP) to I/O networks, and a test port to allow control of the FTP by the experiment host computer.

The I/O network connected the FTP to DIUs via a circuit-switched network rich in redundant interconnections, enabling reconfiguration around network faults.

PRECEDING PAGE BLANK NOT FILMED

PAGE 2 INTENTIONALLY BLANK

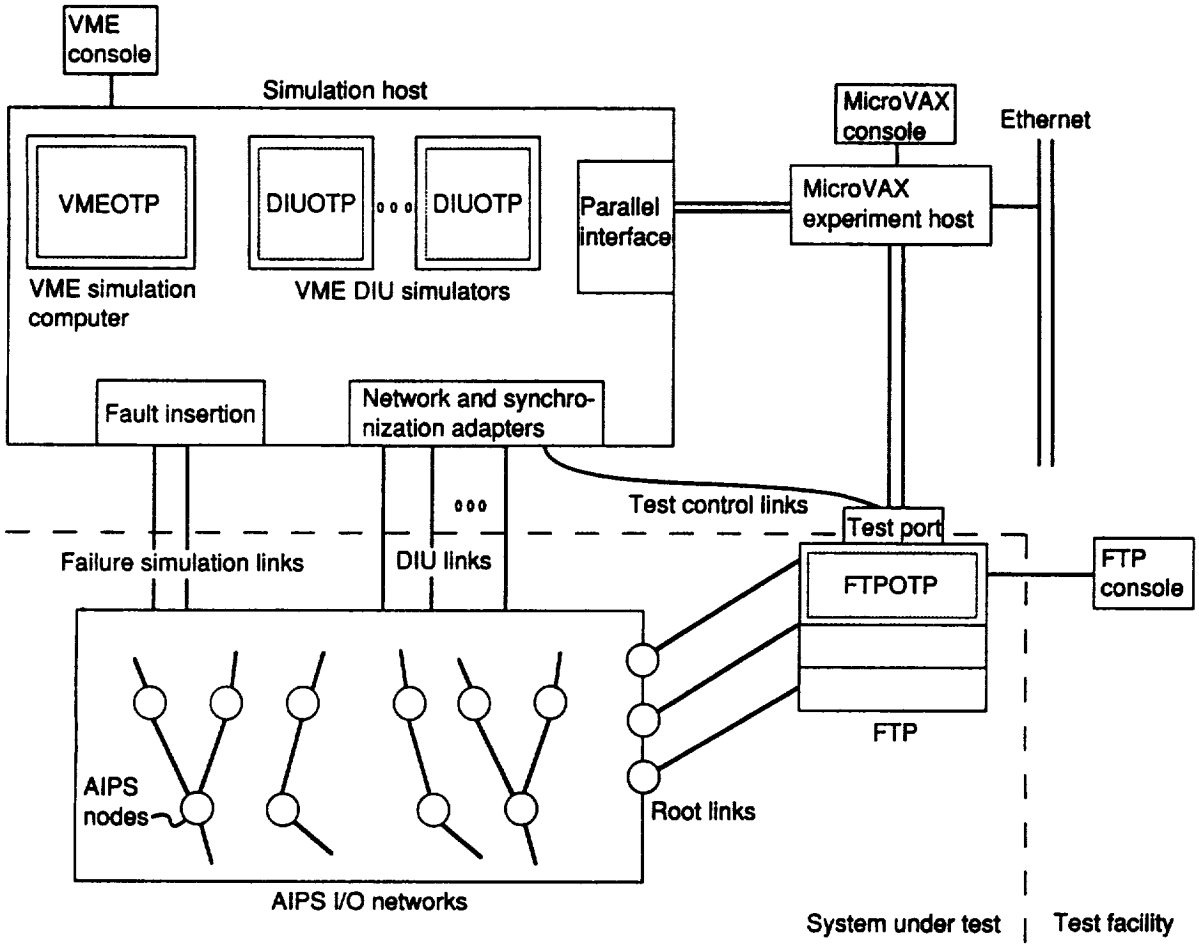


Figure 2.0-1. Experiment Test Configuration

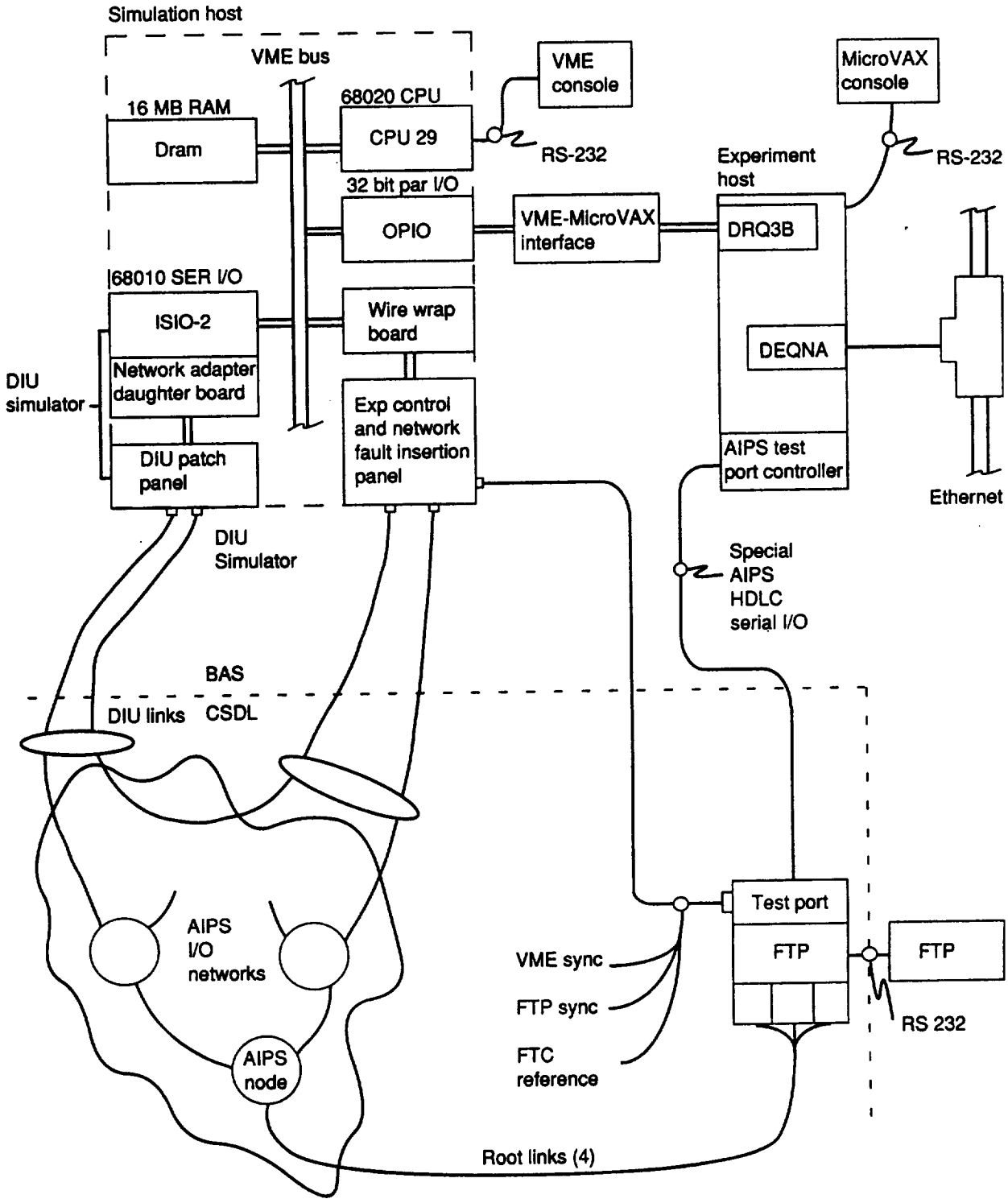


Figure 2.0-2. Small-Scale System Block Diagram

Ada pseudoapplication software was run on the FTP under the control of AIPS system services.

The AIPS system building blocks used in the small-scale system were proof-of-concept components that were still in development and for which no firm specifications existed. Without CSDL's close assistance and cooperation, integration of the small-scale system would not have been possible.

Simulation Host Description. The test facility depicted in figures 2.0-1 and 2.0-2 was designed to support generic, general-purpose test systems that require special interfaces in hardware-in-the-loop simulation environments. It is VMEbus-based and uses standard VMEbus boards obtained from Force Computers, Inc. The base-level test system configuration includes a CPU card, bulk memory, seven intelligent serial interface cards, and a high-speed parallel interface between the simulation host and the experiment host.

Figure 2.0-2 shows the base-level test system after the addition of modifications to support small-scale system hardware-in-the-loop requirements. The upper half of the simulation host block is the base configuration of the general-purpose test system, including the bulk memory, CPU, and interface to the experiment host. The lower half of the simulation host block represents the additions and modifications required to create DIU simulators and special interfaces for the small-scale system. Daughter boards were designed and built for the serial interface cards; a custom wire wrap board was built; and modifications were made to the parallel interface board.

The configurations of each of the main components of the test system are described in greater detail in the discussion section that follows.

Experiment Host Description. The experiment host shown in figure 2.0-2 was a Digital Equipment Corporation (DEC) MicroVAX II computer with 10 MB

of memory, two RD-53 70-MB hard disks, a TK-50 93-MB cassette tape, a DEQNA Ethernet controller, and a DRQ3B parallel, DMA interface card. An AIPS test port controller card was installed to control the AIPS FTP from the experiment host.

Small-Scale System Software Operation. Figure 2.0-1 shows major software elements: VME Operational Test Program (VMEOTP) and DIU Operational Test Program (DIUOTP) in the simulation host, and FTP Operational Test Program (FTPOTP) in the AIPS FTP. The interaction of these software elements is described later in this document.

Section 6.0 of reference 1 is a discussion of small-scale system testing that presents additional information concerning the configuration of the system and the use of the the experiment test configuration.

3.0 DISCUSSION

3.1 GENERAL SIMULATION HOST HARDWARE DESCRIPTION

3.1.1 VMEbus CPU Card

As shown in figure 3.1-1, the CPU-29 contains 128K x 32 bit (1 MB) high-speed static ram, 128K x 32 bits of EPROM, a real-time clock, and two serial I/O ports controlled over a local bus by a Motorola 68020 microprocessor/68881 math coprocessor combination operating at 16.7 MHz. The VMEbus interface is controlled by the 68020 when it gains access to the VMEbus as a bus master. None of the resources on the CPU-29 are accessible to other VMEbus masters.

3.1.2 Bulk VMEbus Memory

The VMEbus DRAM-Exxx bulk memory system shown in figure 3.1-2 has 16 MB of 32-bit-wide dynamic memory. The dynamic ram is slower than the CPU-29 local static RAM but provides relatively low-cost, fast bulk storage for experiment programs and data. The DRAM storage system consists of two cards; the master DRAM controller card has 4 MB of memory and the VMEbus interface; and the slave card, holding 12 MB of memory, connects to the master over a private intercard bus.

3.1.3 VME-MicroVAX Interface

The OPIO-1 card is used for several purposes, as shown in figures 3.1-2 and 3.1-3. This card has four Motorola 68230 programmable interface/timer (PI/T) chips and a Hitachi 68450 four-channel direct memory access (DMA) controller. The PI/T chips provide a total of 32 bits of input, 32 bits of output, 16 handshake lines, and four 24-bit timers. The DMA controller was not used in the small-scale system.

PRECEDING PAGE BLANK NOT FILMED

PAGE 8 INTENTIONALLY BLANK

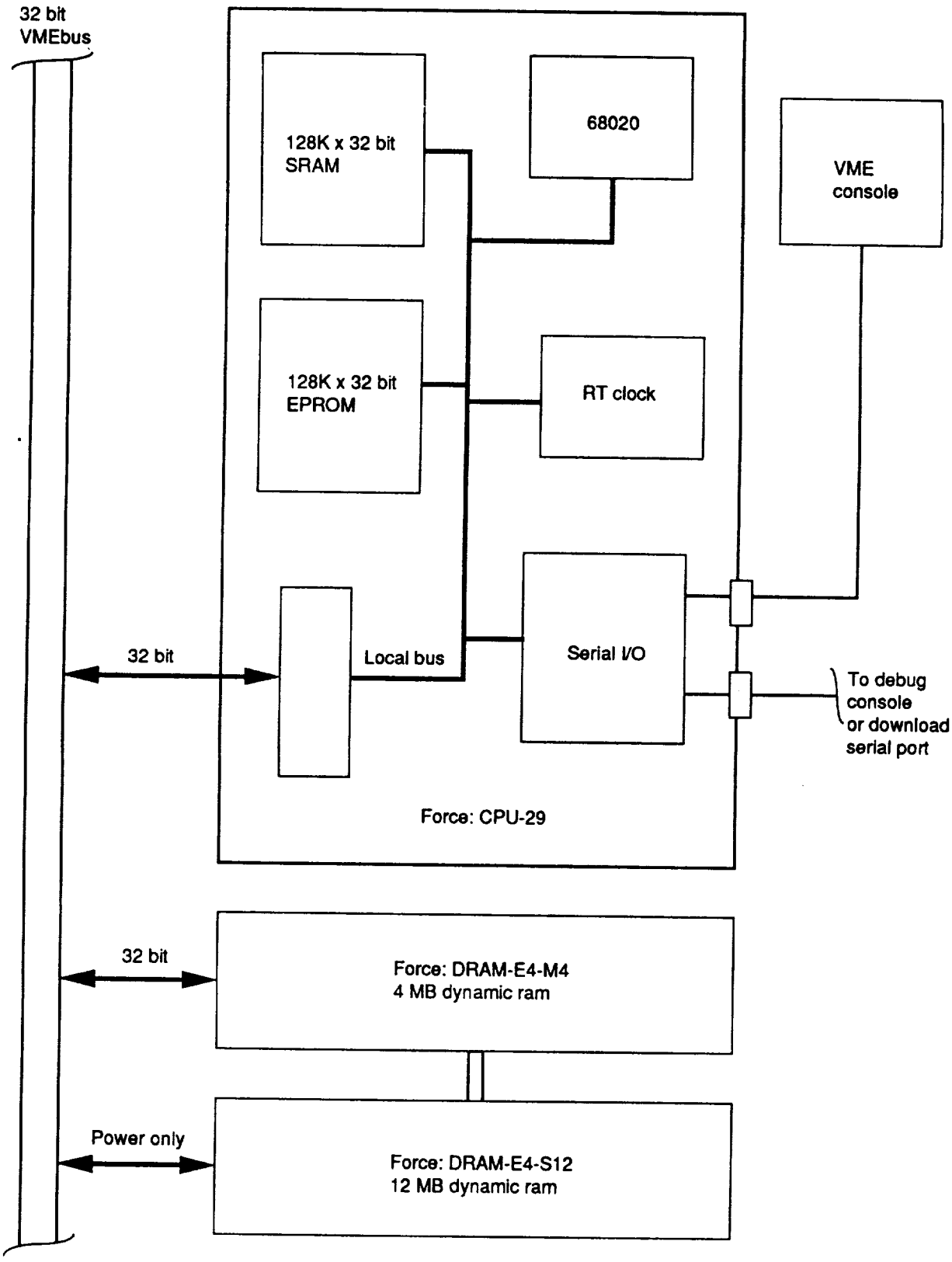


Figure 3.1-1. CPU-29 and Memory Block Diagram

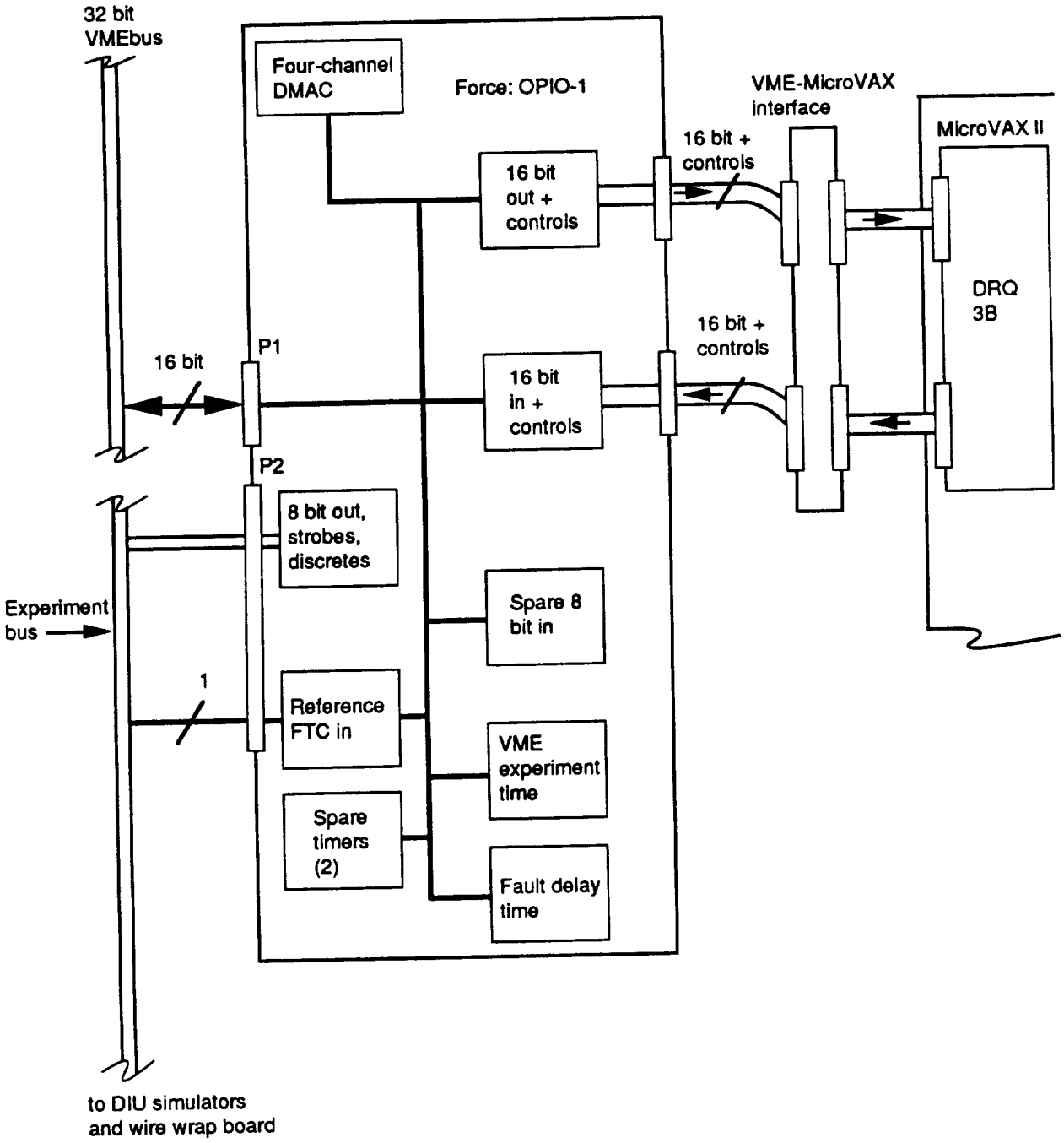


Figure 3.1-2. OPIO-1 and MicroVAX Interface Block Diagram

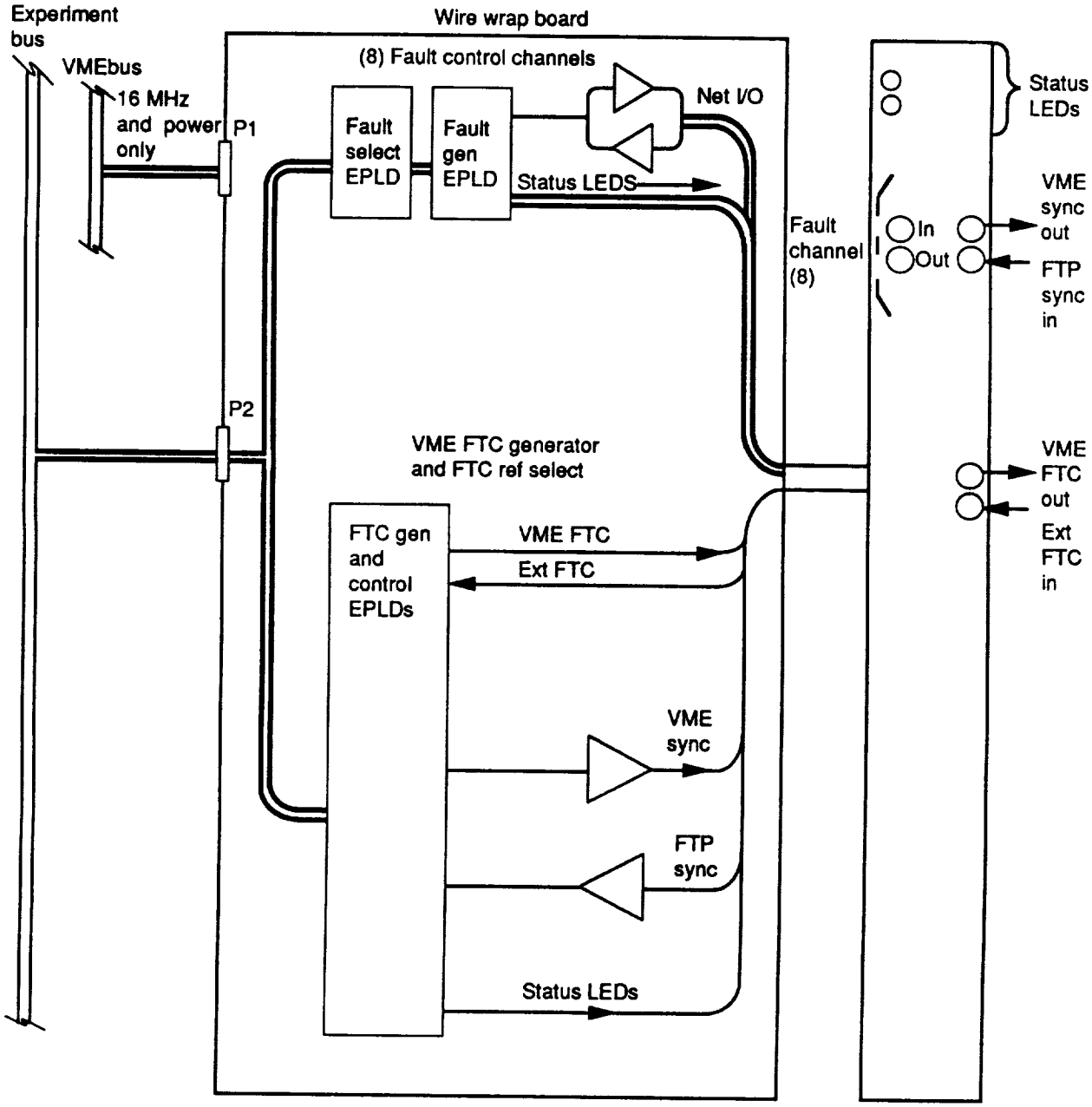


Figure 3.1-3. Wire Wrap Board—Network Fault Insertion/Experiment Control Block Diagram

As illustrated in the upper half of figure 3.1-2, half of the OPIO-1 I/O capability is used for a high-speed 16-bit parallel communications link between the simulation host and the experiment host. The interface is used to download VMEbus programs, to control the simulation host from the experiment host during experiments, and to upload VMEbus experiment data after experiments. Data transfer rates are on the order of 500 KB per second.

Additional functions monitored or controlled using the OPIO-1 card are discussed in sections 3.1.5, 3.1.6, and 3.1.7 below.

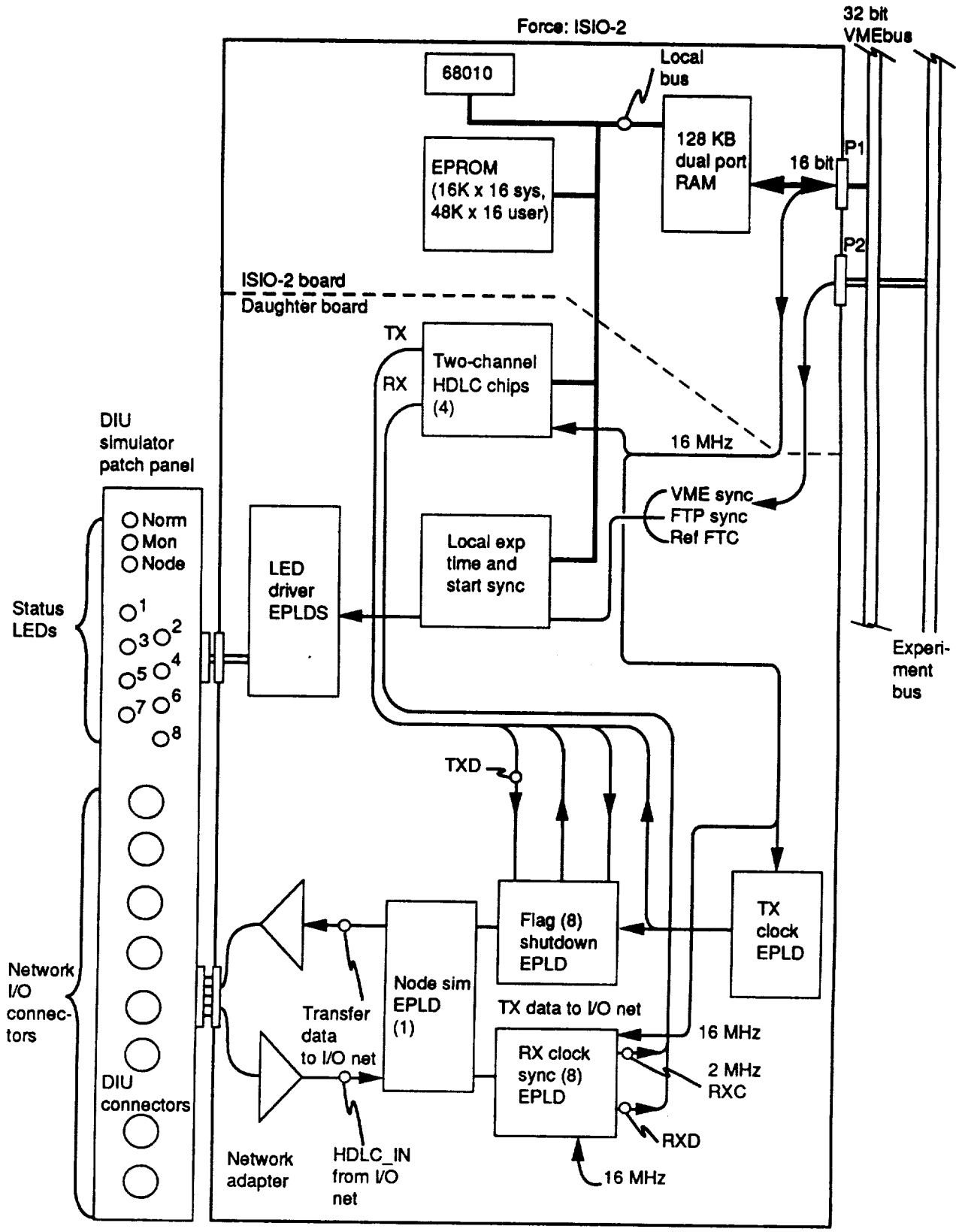
3.1.4 DIU Simulators

Figure 3.1-4 shows a block diagram of the AIPS-compatible DIU simulators based on modified Force Computer ISIO-2 boards from the base-level simulation host. These boards are intelligent peripheral boards: each board has a local 10 MHz Motorola 68010 microprocessor, 128 KB of local/VMEbus dual-ported high-speed static RAM and eight channels of high-speed serial interface capable of supporting the hierarchical data link control (HDLC) protocol at up to 4 MHz.

Local ISIO-2 resources are not directly available to the VMEbus, and the VMEbus is not directly accessible to the local ISIO-2 CPU; all communications in the small-scale system between the ISIO-2 and the VMEbus were via the dual-port ram.

A daughter board and AIPS I/O connectors were added to the existing boards to implement the special AIPS I/O requirements for clock synchronization and flag shutdown. An additional Motorola 68230 PI/T chip provided status, timekeeping, and synchronization functions. The experiment bus (see below) was routed to each of the DIU simulator cards to synchronize the simulators with the VMEbus and AIPS FTP.

The small-scale system used one complete I/O network and one partial network. The design of the ISIO-2 daughter boards enables each simulator



(n) indicates number of replications

Figure 3.1-4. DIU Simulator Block Diagram

board to act either as eight independent DIUs with individual I/O connectors or as eight independent DIUs sharing a single I/O connector. Each DIU simulator board can also be used as a network probe to view all activity on an I/O network for debugging purposes.

3.1.5 I/O Network Fault Insertion

Figure 3.1-3 is a block diagram of the wire wrap card. The top portion of the block diagram shows the network fault insertion channels. Each channel consists of an in and out connector, which are used to break a link in the I/O network. Eight physical fault channels are present on the card. Each of the channels can be mapped to a logical channel for fault control purposes. Failing a logical channel causes all physical channels mapped to that logical channel to fail simultaneously to the same fault condition.

The fault channels support three different modes: normal, in which inputs are routed directly to outputs; passive, in which outputs are failed to a low state; and active, in which outputs are failed to a high state. The state of each fault channel in and out connector output is indicated by a bi-color LED: green for normal; off for passive; and red for active. Small-scale system faults were identically applied to both in and out connectors.

3.1.6 Experiment Control and Experiment Bus

The wire wrap card shown in figure 3.1-3 was also used to interface the simulation host to the AIPS FTP to control experiment synchronization and timing. The VME sync output was used to signal the FTP when the simulation host was ready to proceed; the FTP sync input was used to synchronize all experiment time keeping functions in the simulation host. The state of the the sync signals is indicated by bi-color LEDs: red for stop and green for run.

A common timebase is used for all experiment timekeeping in the simulation host. Two timebase options are provided: either the FTC generated by the VMEbus wire wrap card or an external FTC can be used. The small-scale system used an external FTC provided by the FTP so that experiment timekeeping in the FTP and the simulation host would correlate.

Timebase, synchronization, and control signals are routed to each DIU simulator and the wire wrap board from the OPIO-1 card via spare pins in the VMEbus P2 connector, which form the experiment bus.

3.1.7 VMEbus Experiment Time and Fault Insertion Delay

The OPIO-1 card shown in figure 3.1-2 was modified to add a clock control daughter board that synchronizes VMEbus timing signals with all DIU simulators and the external system under test. One of the OPIO-1 PI/T timers is the source of VMEbus experiment timekeeping; another is used as a delay timer for controlling fault insertion timing. Two spare timers remain on the OPIO-1.

3.2 DEVELOPMENT SUPPORT HARDWARE

3.2.1 Development Host Computer

All software and hardware were designed using software tools installed on a DEC VAXstation 2000. The VAXstation was equipped with 6 MB of RAM, an internal RD-53 70-MB hard disk, an external RD-54 150-MB hard disk, and an external TK-50 tape drive.

3.2.2 Terminal Server

A DECserver 200 was included in the development support hardware to allow flexible access to serial devices. Using the terminal server in the development system allows serial port access to the simulation host computer from local CRTs or the VAXstation; use of a single serial

printer for both experiment host and development host; access to emulators from CRTs or development host; and access to either the development or experiment host computer from local CRTs.

3.2.3 Line Printer

A Mannesmann Tally MT660 line printer provided both text and graphics output for the VAXstation and the experiment host computer.

3.2.4 Logic Analyzer

A Tektronix DAS 9200 logic analyzer with two 92A90 modules, a 92A16 module, a 68010 PGA adapter pod plus software, a 68020 adapter pod plus software, and a parallel graphics printer were used to aid in hardware and software debugging and performance evaluation. The logic analyzer was also used to view activity on the AIPS I/O network.

3.2.5 Microprocessor Emulators

Applied Microsystems emulators equipped with C source level debugging software were available for debugging hardware and software problems in both the Motorola 68010-based ISIO card and the Motorola 68020-based CPU-29 card.

3.3 GENERAL SOFTWARE DESCRIPTION

The major software development efforts were the implementation of DIU simulator software, creation of FTP pseudoapplications, and production of experiment control programs.

3.3.1 Experiment Software Interaction

Figure 3.3-1 is a block diagram of small-scale system software as it was used during an experiment run.

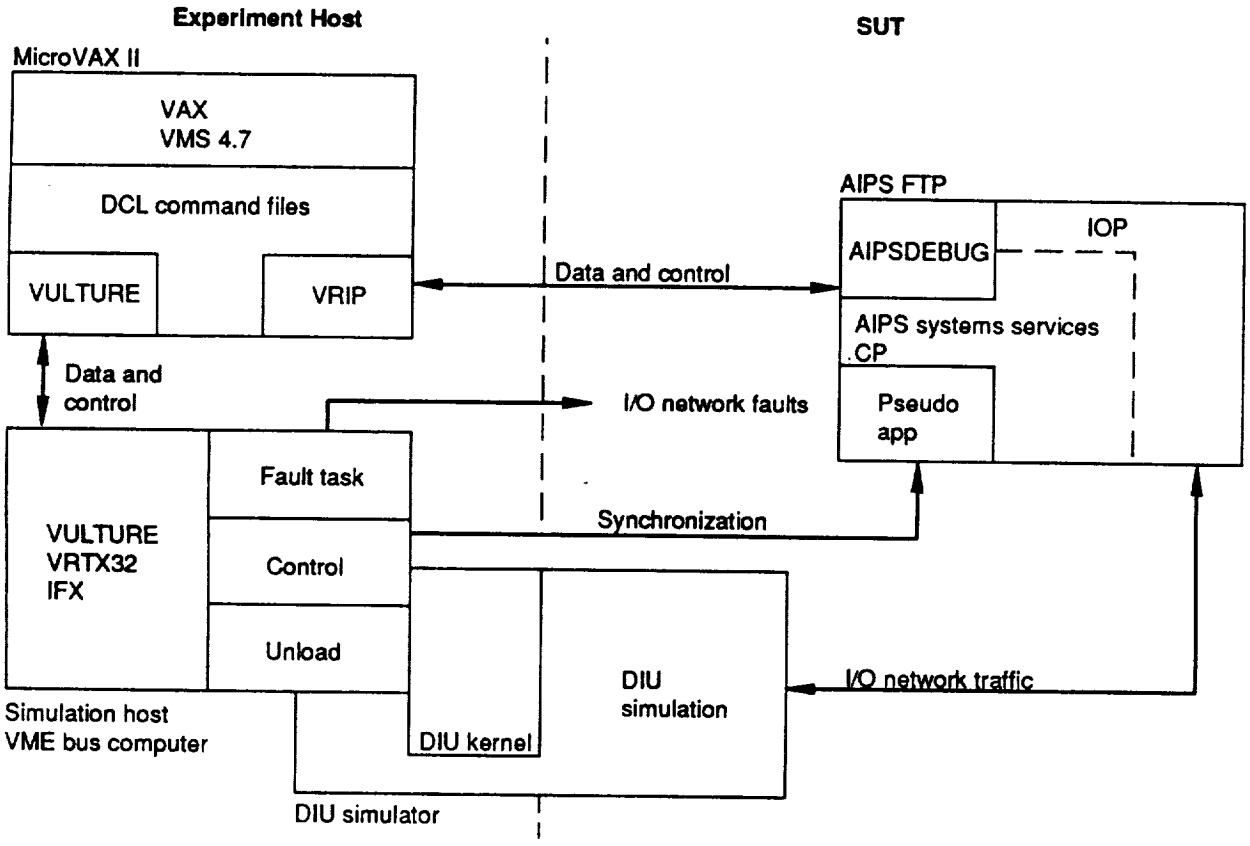


Figure 3.3-1. Small-Scale System Software Block Diagram

Experiments were controlled by DCL command files running in the experiment host. Experiment control commands were routed to the AIPS FTP using the VRIP/AIPSDEBUG interface and routed to the simulation host using the VME Ultimate User Environment (VULTURE) interface. The VRIP/AIPSDEBUG interface was used to start initialization programs in the ftp, activating aips systems services and pseudoapplication programs. VULTURE commands to the simulation host loaded DIU software in simulator boards and started the VMEbus computer control, fault task, and unload programs. The simulation host control program was used to synchronize operation of the FTP pseudoapplication, the fault task, and the DIU simulation program.

Upon completion of an experiment run, the unload program in the simulation host removed and reformatted data from the DIU simulator for uploading to the experiment host disk using VULTURE commands. VRIP/AIPSDEBUG commands were used to remove experiment application data from the FTP for storage on the experiment host disk.

3.3.2 Experiment Host Software

The experiment host MicroVAX II ran version 4.7 of the VAX VMS operating system.

Software in the experiment host was responsible for controlling the overall operation of both the simulation host and the FTP. The main experiment control programs were DEC DCL command files. The simulation host was controlled using the VULTURE commands; the FTP was controlled using VRIP/AIPSDEBUG commands.

All executable programs for the simulation host and the FTP were stored on disk in the experiment host. These programs were downloaded to their appropriate target machines using either VULTURE or VRIP/AIPSDEBUG.

Upon completion of experiment runs, data were uploaded from the simulation host and the FTP to disk files on the experiment host, where they were transferred to magnetic tape for archiving.

Experiment host resident analysis software was used to perform preliminary analysis on collected data. The direction of experimentation was guided by the ability to perform timely data analysis.

3.3.3 Simulation Host Software

A portion of the software in the simulation host is located in EPROM to control the operation of the simulation host at power up. The EPROMS contain Ready Systems VRTX-32 and IFX, a board support package to adapt VRTX-32 and IFX to the CPU-29; VMEPROM, which is shipped with the CPU-29, the VMEbus resident portion of the VULTURE program; and a sharable copy of Ready Systems Real Time C library.

A portion of simulation host RAM is configured as two RAM disks: DRAM: which is located in the DRAM-E4XXX boards, and SRAM:, which is in CPU-29 static RAM. All files in these two RAM disks obey MS-DOS file-naming conventions. Files in the RAM disks can be either contiguous or noncontiguous.

Programs downloaded from the experiment host to the simulation host can reside on either of the two RAM disks. Executable programs are required to be in contiguous files in RAM disk. Optimum performance of CPU-29 targetted programs is obtained when they reside on the SRAM: disk. CPU-29 programs will also run on DRAM: disk but with slightly reduced performance. When a CPU-29 program is started using the VULTURE VRUN command, the program executes the image of the program in the contiguous ram disk file.

Data collected by the simulation host during an experiment run were stored in noncontiguous files in the DRAM: disk and later uploaded to disk files in the MicroVAX II experiment host.

Repeated creation and deletion of RAM disk files may cause disk fragmentation; no attempt is made to repack memory. No fragmentation problems were encountered during operation with the small-scale system.

3.3.4 DIU Simulator Software

Executable versions of DIU simulator software were stored on disk in the experiment host. They were downloaded from the experiment host to contiguous RAM disk files in the DRAM: disk in the simulation host. At the start of an experiment run, the DIU simulator software was loaded from the DRAM: disk into the active DIU simulator boards.

The ISIO-2 DUSCC chip initialization portion of firmware, supplied by Force Computers in EPROM on the ISIO-2 boards, was modified to prevent a hardware conflict for the DUSCC chip serial clock source.

A DIU simulator control kernel was developed to supply synchronization and system-level functions for the DIU simulator software. The kernel software must be loaded before any DIU software can be run.

DIU simulator software was configured to respond to specific AIPS I/O network addresses. See appendix E and software tape for more details.

3.3.5 AIPS FTP Software

The pseudoapplication programs run in the AIPS FTP were based on Ada code templates provided by CSDL. The code templates were modified to meet small-scale system testing requirements of reference 1.

Some modifications of the AIPS runtime software were made to support the testing requirements of reference 1.

Pseudoapplication software is discussed in reference 1. Listings are found in the software tape.

3.3.6 Development Host Support Software

The VAXstation 2000 development host computer system was supplied with software packages to support both hardware and software design. All the software ran under version 4.7 of the VAX VMS operating system.

Software included DEC Ada to support the data analysis program, DEC C to support the VAX portion of the VULTURE interface, and DEC FORTRAN to support recompilation the VRIP interface for the MicroVAX II.

Third-party software included Autodesk AutoCAD for documentation and ISIO daughter board layout, Data I/O Abel to support the design and production of EPLDs for small-scale system (SSS) custom hardware, Verdix 68010 Ada for compilation of FTP pseudoapplications, and Microtec C and Assembler to support VMEbus CPU and DIU simulation software.

3.4 HARDWARE MODIFICATIONS TO AIPS BUILDING BLOCKS

To provide a common timebase in both the FTP and the simulation host, the test port controllers for all three channels of the SSS FTP were modified to provide a differential driver ftc output. Only the Channel A connector panel at the rear of the FTP was modified to connect the differential FTC output to one of the spare connectors. This signal was used by the simulation host for all of its local experiment timekeeping functions.

No other modifications were required to AIPS hardware building blocks.

3.5 SIMULATION HOST HARDWARE DETAILS

3.5.1 VMEbus System Configuration

Figures in documentation package A show the allocation of cards and special interfaces in the slots of the VMEbus card cage. All slots are used when all seven DIU simulator boards are installed. The small-scale system can operate with four DIU simulators boards when no simulator boards are required for network I/O debugging.

Figure 3.5-1 is a detailed block diagram of the VMEbus system simulation host configured for use in the small-scale system.

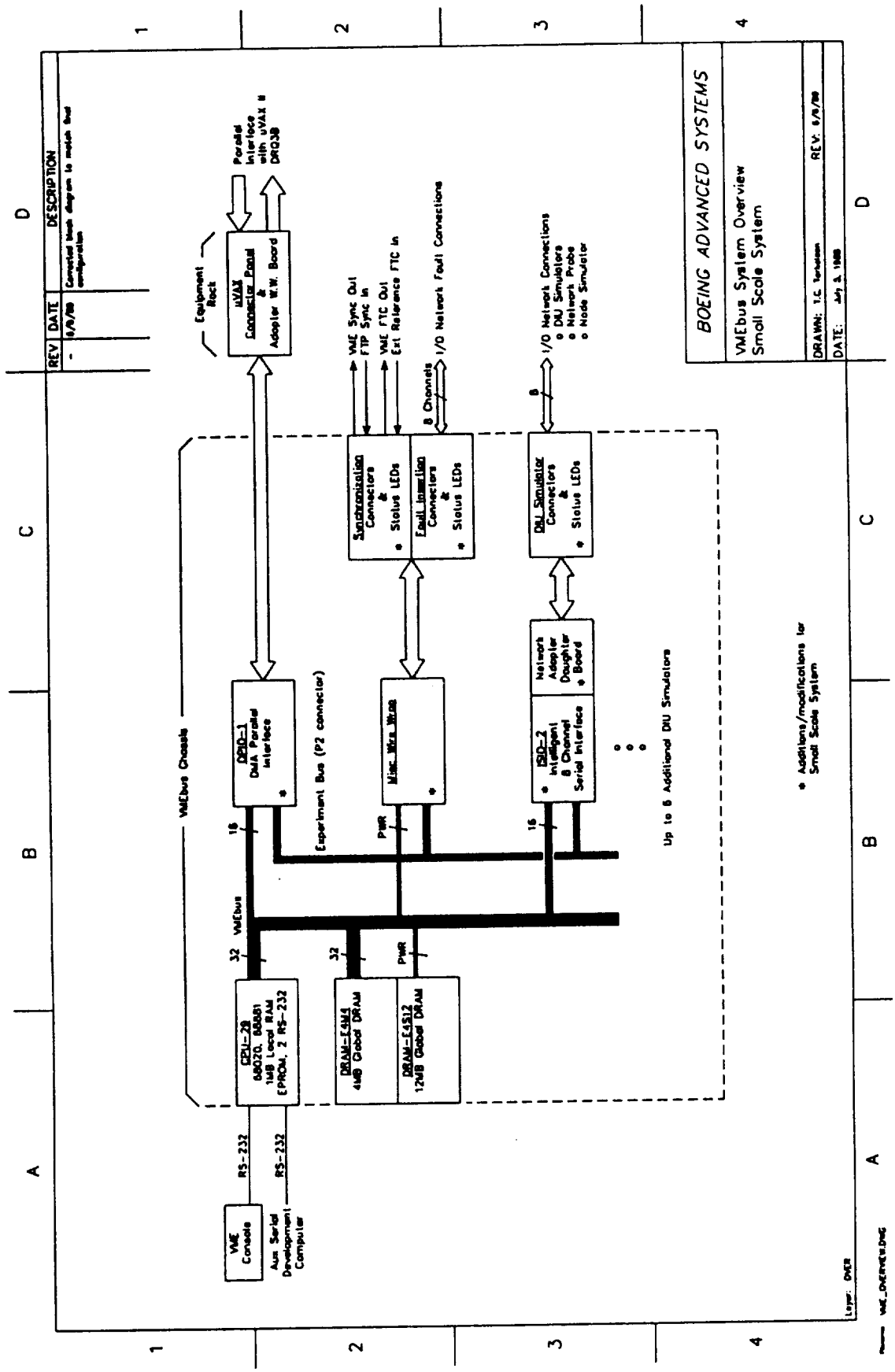


Figure 3.5-1. VMEbus System Overview, Small-Scale System

ORIGINAL PAGE IS
OF POOR QUALITY

3.5.2 Experiment Host - Simulation Host Interface

Figure 3.5-2 is a detailed block diagram of the interface between the VMEbus simulation host OPIO-1 card and the MicroVAX II experiment host DRQ3B card. The MicroVAX connector panel shown serves four functions: to cross-connect the VMEbus and MicroVAX II data lines for correct transfer of ASCII data; to provide miscellaneous function lines for cross-system signaling; to buffer board I/O signals; and to condition the handshake lines of the two interface cards to ensure correct data transfers.

See documentation package B for OPIO-1 layout and schematics.

DRQ3B inputs and outputs are 74S series TTL terminated with 330/220 pull up/down resistors. Inputs must be driven from devices with 22 mA pull-down capabilities. (See ref. 2 for additional DRQ3B information.)

The OPIO-1 card uses high-speed opto-isolators on all input and output lines. The output of the opto-isolators is insufficient to drive the DRQ3B, and there is no real need for output isolation. The output opto-isolators were removed and jumpers inserted to directly connect the output of the opto-isolator driver chips to the DRQ3B inputs.

It was desirable to leave the inputs to the OPIO-1 opto-isolated to minimize the possibility of damaging the inputs to the MC68230 parallel interface timer (PI/T) chips. The output from the DRQ3B does not pull up high enough at logic 1 out to guarantee that the opto-isolators will shut off. To solve this problem, the input LEDs of the opto-isolators were run from a supply one diode drop below 5V. This was adequate to guarantee that the isolators will be shut off at DRQ3B logic 1 out.

Documentation package C contains the drawings used to produce the wire wrap interface and connector panel for the VME-MicroVAX interface.

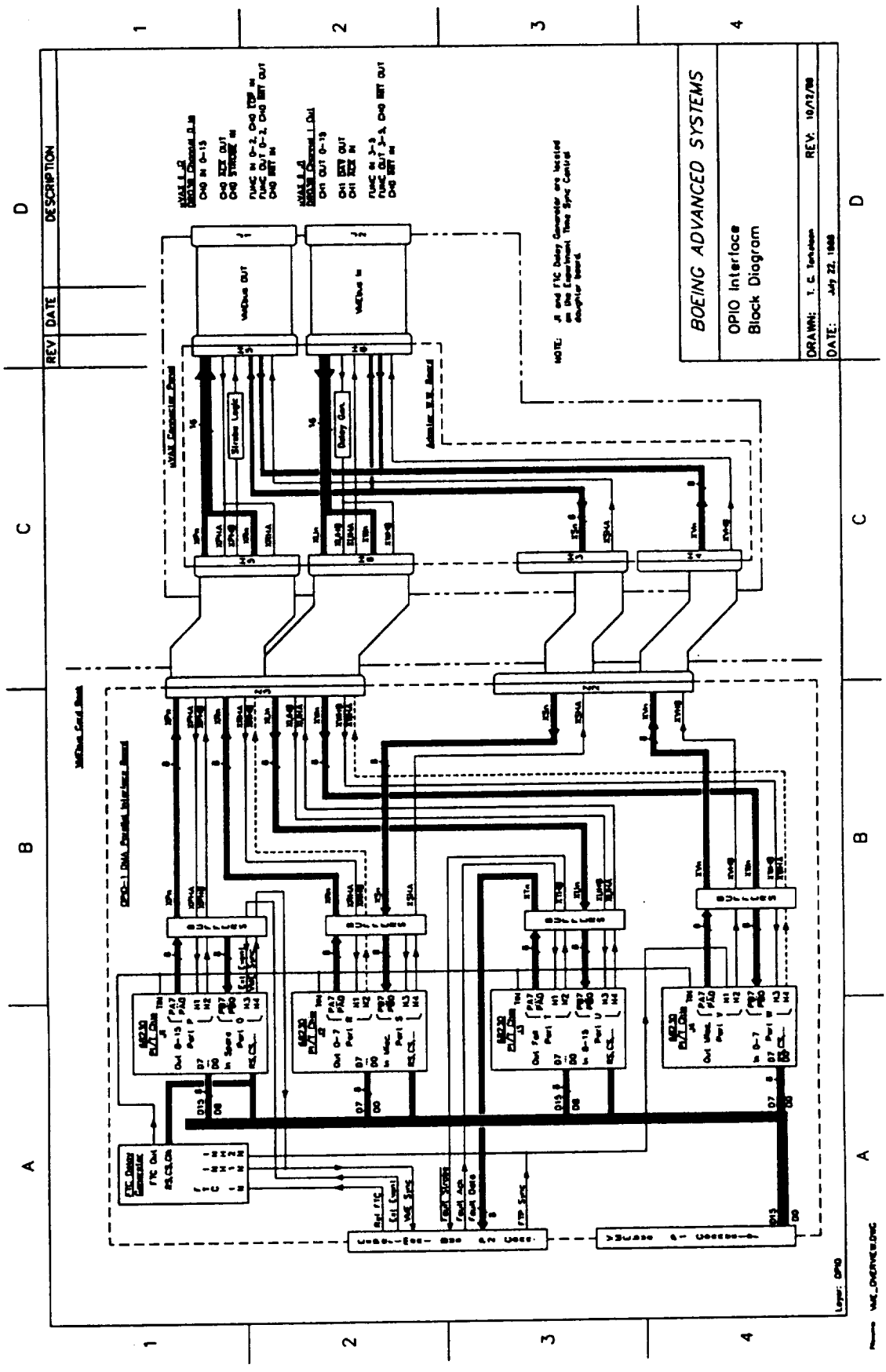


Figure 3.5-2. Experiment Host—Simulation Host Interface Detailed Block Diagram

ORIGINAL PAGE IS
OF POOR QUALITY

Documentation package C illustrates the interface panel design used for the SSS. The only deviation from this design was that a standard rack width U chassis with side panels was used for shielding the interface board instead of the protective cover shown.

The protective cover shown is preferable because it is permanently affixed to the connector panel. The U chassis is secured with the same screws that hold the connector panel to the equipment rack, raising the possibility of damage to the interface wire wrap board or ribbon cables during installation.

The ribbon cables that connect the wire wrap board to the VMEbus OPIO-1 card are held in place with the edge of the side panels attached to the U chassis. The exposed edges of the side panels are covered with alligator grommet to prevent insulation damage to the ribbon cables.

Documentation package C shows the placement of components on the wire wrap interface board and illustrates the routing of cables from the OPIO-1 to the interface board for an installation in which the interface connector panel is mounted in the VMEbus chassis. No spare slots were available in the small-scale system VMEbus chassis for a connector panel, so the interface board was mounted at the rear of the equipment rack, with the ribbon cables routed up through the top of the card cage to the adapter in the back of the equipment rack.

Documentation package C is a schematic for the wire wrap adapter. The left side of the schematic shows the connections and function names for the VMEbus OPIO-1 board; the connections and function names on the right side of the schematic are for the Micro VAX II DRQ3B interface.

Data Representation and Effect on Data Transfer Interface. Close inspection of the interface wire wrap interconnection schematic reveals that the upper and lower data bytes of the interface lines between the MicroVAX and the VME system are reversed. An explanation of this apparent inconsistency is necessary both to understand the design of the

hardware interface and to appreciate the subtleties encountered when attempting to transfer data between a Motorola 68020-based VMEbus system and a DEC VAX system.

The underlying reason for byte swapping the interconnections was a difference in the byte stacking order in the 68020 VMEbus and MicroVAX systems. The VMEbus system is based on a Motorola 68020 microprocessor. The 68020 byte stacking order is reversed from that of the MicroVAX system except for byte data. (The MicroVAX byte stacking order is the same as that used by Intel systems.)

Figure 3.5-3 shows the byte stacking orders of three types of arrays as stored in the two systems. The array data types are long (32 bit), short (16 bit), and char (8 bit). The examples are shown as hexadecimal bytes arranged from lowest address to highest address. The partial C source code to generate these representations is also shown.

For data to retain the same value in both systems, the byte stacking order must be translated when transferring data. To translate the byte stacking order it is necessary to know the type of data being transferred. For systems that use data structures composed of mixed types it is impossible to perform byte stacking order translation without having access to the definition of every specific data structure being transferred.

The solution chosen for this problem was to use C library functions to convert all binary data to ASCII before transfer between the two systems. This has the disadvantage that up to twice as much data storage may be required, and additional time is required to convert binary data to ASCII. Note that data may still be collected in any desired format during real-time operations as long as it is converted to ASCII before transfer to the MicroVAX.

Partial C code used to generate arrays.

```
/* array allocation */
    unsigned char array_1[8];
    unsigned short array_2[4];
    unsigned long array_4[2];

/* assign values to unsigned character array */
    array_1[0] = 0x11;
    array_1[1] = 0x22;
    array_1[2] = 0x33;
    array_1[3] = 0x44;
    array_1[4] = 0xAA;
    array_1[5] = 0xBB;
    array_1[6] = 0xCC;
    array_1[7] = 0xDD;

/* assign values to unsigned short array */
    array_2[0] = 0x1122;
    array_2[1] = 0x3344;
    array_2[2] = 0xAABB;
    array_2[3] = 0xCCDD;

/* assign values to unsigned long array */
    array_4[0] = 0x11223344;
    array_4[1] = 0xAABBCCDD;
```

The above code generates the following data arrays in the VMEbus and uVAX systems.

```
    ------- VMEbus --------      --------- uVAX --------
     0  1  2  3  4  5  6  7       0  1  2  3  4  5  6  7
-- char array
    11 22 33 44 AA BB CC DD     11 22 33 44 AA BB CC DD
-- word array
    22 11 44 33 BB AA DD CC     11 22 33 44 AA BB CC DD
-- long array
    44 33 22 11 DD CC BB AA     11 22 33 44 AA BB CC DD
```

Figure 3.5-3. VMEbus and MicroVAX Byte Stacking Order

Figure 3.5-3 shows that ASCII data is stacked in the same order in both the VMEbus and MicroVAX systems, implying that there is no reason to swap the upper and lower bytes in hardware. The physical interfaces, however, treat all transferred data as word (16 bit) data. When the VMEbus system transfers word data, the most significant byte located at address 0 is sent to bits 8 through 15 of the interface; the least significant byte from address 1 is sent to bits 0 thru 7. Because the data being sent are actually ASCII, the bytes are swapped by the VMEbus system. Swapping the high and low byte lines corrects the problem.

Figures 3.5-4 and 3.5-5 illustrate the transfer of different types of data using both unswapped and swapped lines.

The limitations of this solution have not affected VMEbus system performance adversely enough to require an alternative solution.

Experiment Host to Simulation Host Transfer Protocol. See figure 3.5-6 for typical interface handshake waveforms. Port B of OPIO-1 PI/T devices J3 and J4 serve as the 16-bit VMEbus input port. They are both configured to operate as double-buffered input devices with interlocked input handshakes. (PI/T Port B is set up to operate mode 0, submode 00, double-buffered input, interlocked input handshake protocol.) The input !STROBE is received by both PI/Ts on their H3 pins. The PI/T !ACK output originates on the H4 pin PI/T J3. (A PI/T !ACK output is also available from pin H4 on PI/T J4, but is not used.)

The two handshake protocols have no logical conflicts; however, handshake timing must be modified because of VMEbus PI/T data setup timing requirements. DRQ3B !DAV Out is set low a minimum of 65 ns after data are stable on the out nn lines. At least 100 ns of data setup must be allowed before an input !STROBE is applied to the PI/T chip. To meet the setup time requirements, a delay of approximately 100 ns is placed between the DRQ3B !DAV output and the PI/T input !STROBE, leaving at least a 65 ns margin. The delay is implemented using an RC delay and 74LS14 Schmitt input inverters. (See U1 and associated components in the schematic in documentation package C.)

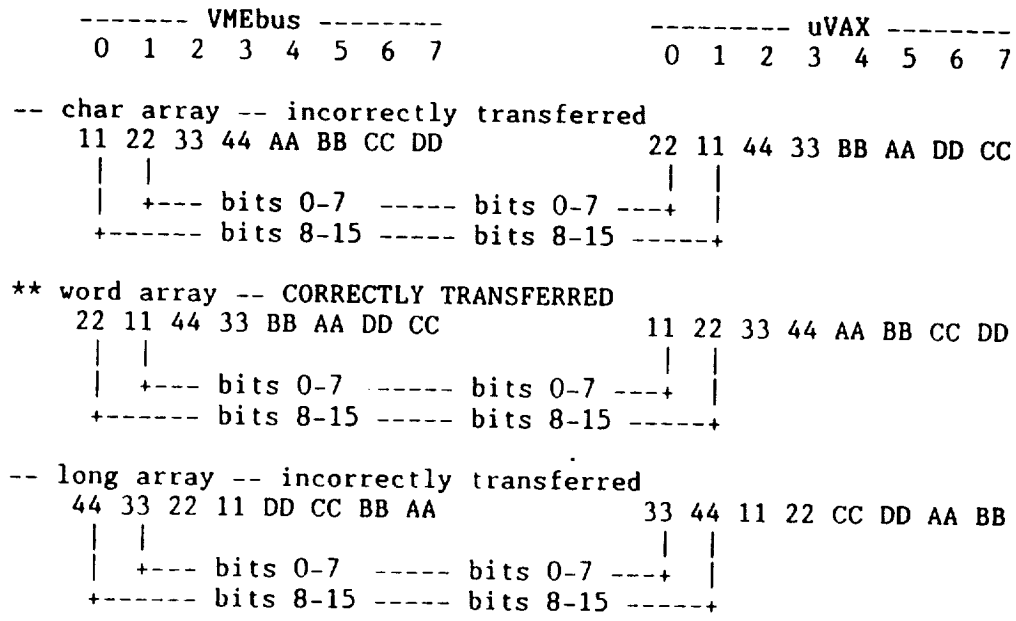


Figure 3.5-4. Data Transfer Without Hardware Byte Swapping

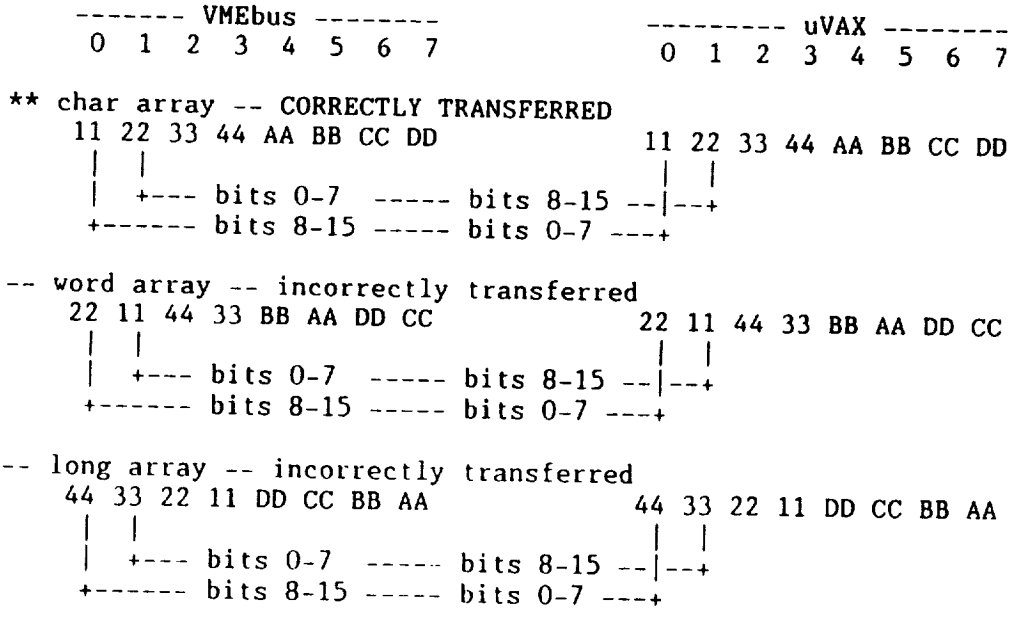


Figure 3.5-5. Data Transfer With Hardware Byte Swapping

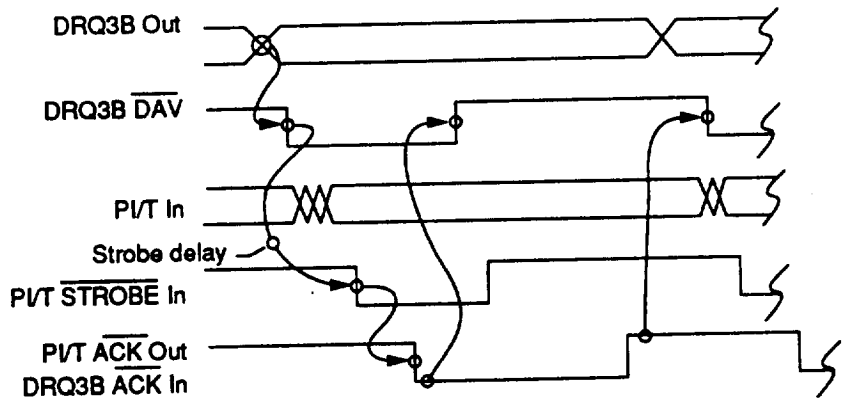


Figure 3.5-6. Experiment Host to Simulation Host Transfer

Note: symbols such as !ACK signify inverted polarity logic.

7The modified MicroVAX to VME transfer operates as follows (see fig. 3.5-6).

a. Data are placed on the DRQ3B Out lines and the DRQ3B !DAV Out is pulled low.

b. After a 100-ns delay, the DRQ3B !DAV Out appears at PI/T !STROBE in, latching data in the PI/T input buffer.

c. After data are latched by the PI/T, PI/T !ACK Out is pulled low.

d. On receiving DRQ3B !ACK In low, DRQ3B !DAV Out returns high.

e. If more space is available in PI/T input buffers, PI/T !ACK Out returns high and the next data transfer cycle begins. If the PI/T input buffers are full, PI/T !ACK remains low until the VMEbus computer removes data from the PI/T input buffer, at which time PI/T !ACK returns high and another data transfer cycle begins.

Notes:

a. PI/T !ACK Out timing is referenced to the falling edge of PI/T !STROBE In, not the rising edge.

b. PI/T !ACK out will return high regardless of the state of PI/T !STROBE In.

Simulation Host to Experiment Host Transfer Protocol. Without modification, the PI/T to DRQ3B transfer handshake protocol does not work. A logical conflict exists, caused by different interpretations of the meaning of the !ACK signal. The DRQ3B uses the !ACK line to first signify receipt of data at the falling edge and then to signify ready for next

transfer on the rising edge. The PI/T, however, interprets the falling edge of the !ACK line to mean data accepted and ready for next transfer.

Without modification, the PI/T begins its next data transfer before the DRQ3B !ACK Out line has returned high. Because the DRQ3B !ACK Out line is still low, the DRQ3B ignores the !STROBE signal and does not produce an !ACK Out handshake for the PI/T. This both causes the data being transferred to be lost and the interface to hang up while the PI/T waits forever for the DRQ3B !ACK Out handshake.

A NAND R-S latch formed from gates in U2 resolves the conflict. The output of the latch is buffered by U3 to provide adequate drive capability for the DRQ3B input.

The operation of the modified interface is as follows (see fig. 3.5-7).

Note: Symbols such as !ACK signify inverted polarity logic.

a. Data are placed on PI/T Out lines and PI/T !DAV Out goes low.

b. On DRQ3B !STROBE In low, data on DRQ3B In lines are read and DRQ3B !ACK Out goes low, keeping !STROBE In low in the U2 latch.

c. The rising edge of PI/T !ACK In causes PI/T !DAV Out to return high and the PI/T starts another output transfer cycle.

d. Data are placed on PI/T Out lines and PI/T !DAV Out goes low; however, because DRQ3B !ACK Out is still low, DRQ3B !STROBE In remains high.

e. When DRQ3B !ACK Out finally returns high, DRQ3B !STROBE immediately goes low and a new data transfer cycle begins.

3.5.3 VMEbus Backplane Modifications

Jumpers were installed between P1a-21 and P1a-22 to maintain the continuity of the I!ACKIN* I!ACKOUT* daisy chain for unused VMEbus connector positions at slots 5, 6, 9, 11, 13, 15, 17, and 19.

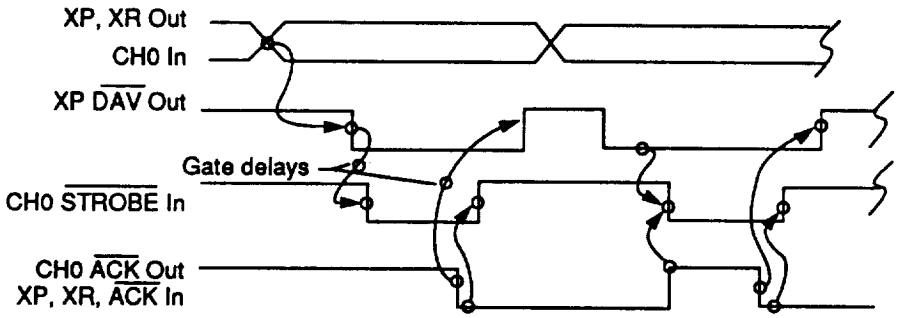


Figure 3.5-7. Simulation Host to Experiment Host Transfer

A 64-conductor ribbon cable with special connectors was fabricated to connect the experiment bus described in appendix G from VMEbus chassis slot 4 P2 connector to P2 connectors in slots 7, 8, 10, 12, 14, 16, 18, and 20. The uncommitted pins of the P2 connectors are used. (See documentation package A.)

3.5.4 DIU Simulator

Figure 3.5-8 is a block diagram showing additions made to Force ISIO-2 VMEbus cards to adapt them to the interface requirements of the AIPS I/O network as described in appendix B. The additions reside on a double sided daughter board that plugs into five IC sockets on the ISIO-2 card. All power and signal connections between the daughter board and the ISIO-2 board are made via these five IC sockets.

The major additions and modifications required to adapt the ISIO-2 card to DIU simulator service were:

a. Operation of DUSCC chips with external 2 MHz receive and transmit clocks and isolation of DUSCC I/O lines from the ISIO-2 board.

b. Addition of AIPS I/O network compatible differential line drivers, receivers, and termination resistors.

c. Addition of external receive clock synchronization circuitry.

d. Addition of transmitter output flag shutdown circuitry.

e. Addition of a 2-MHz transmitter clock generator.

f. Addition of a second 68230 PI/T chip and address decoder for timekeeping, daughter board control, and vectored interrupt selection.

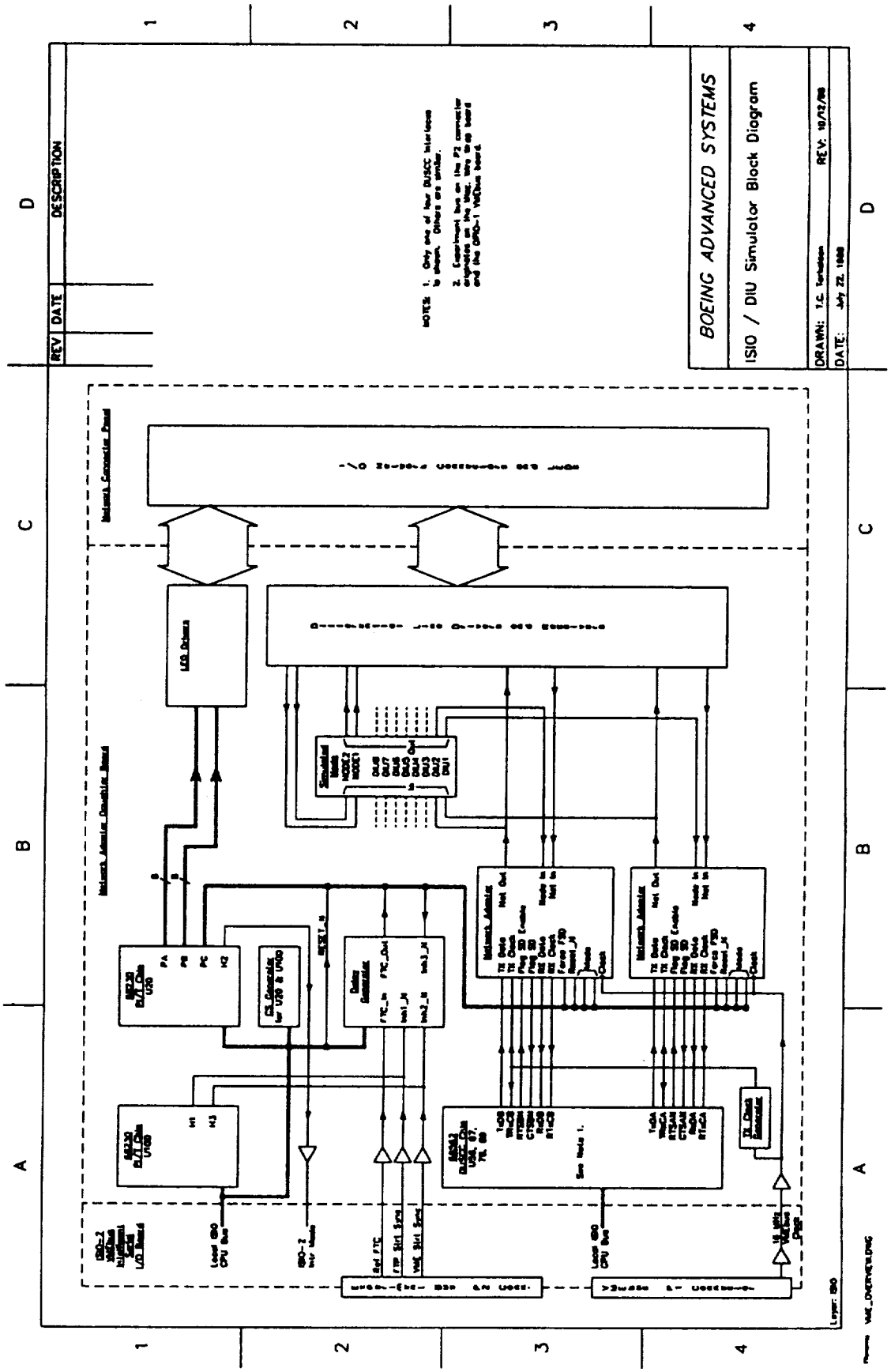


Figure 3.5-8. DIU Simulator Detailed Block Diagram

ORIGINAL PAGE IS
OF POOR QUALITY

The daughter board and its components are described in detail in this section.

Network Connector Panel. The DIU interface to the AIPS I/O network is via DIN audio connectors mounted on the I/O network connector panel. Fabrication details for the sheet metal and silkscreen of the DIU simulator front panel are in documentation package D.

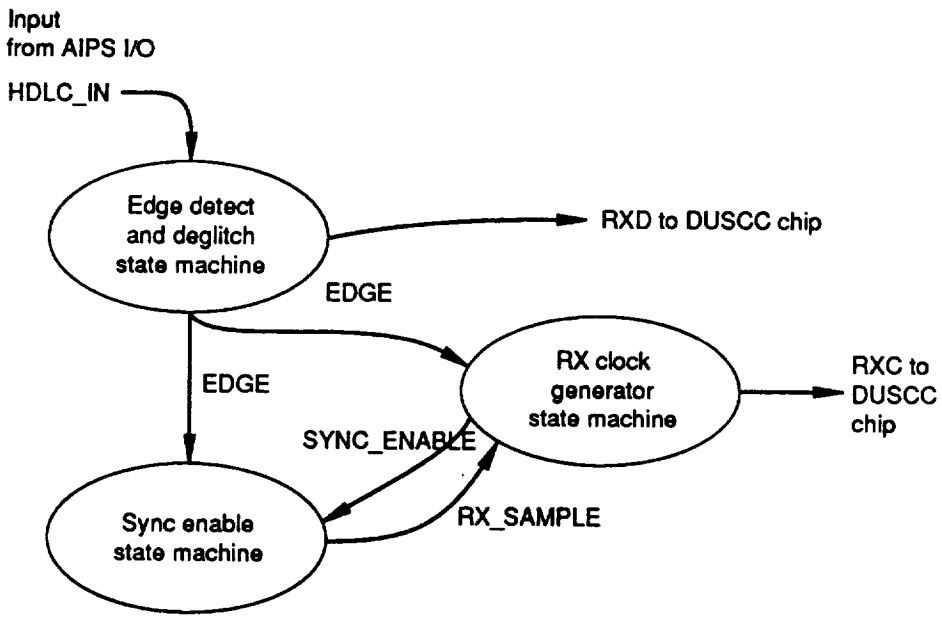
DIU I/O connectors and status LEDs are soldered to a small PC board located behind the panel. Two ribbon cables connect this PC board to the ISIO daughter board: a 50-conductor ribbon cable is used exclusively for I/O connections, and a 20-conductor ribbon cable is used for status LEDs.

Differential Drivers and Receivers. Differential drivers, receivers, isolation resistors, and termination resistors are located on the daughter board. Components used are in accordance with appendix B to ensure DIU I/O interface compatability with AIPS I/O network requirements.

RX Clock Generator EPLD. Data sent over the AIPS I/O network are transmitted at 2 Mb per second. To receive these data each DIU simulator channel must independently regenerate a clock synchronized to its incoming data. Clock stability and synchronization requirements are specified in appendix B.

An Altera EP600 erasable programmable logic device (EPLD) was designed to use three interdependent state machines to synchronize the DIU clock to incoming RX data, meet the clock timing requirements of the DUSCC chip, and to provide deglitching of the received data. All three state machines are driven by the same 16-MHz system clock. Figure 3.5-9 shows their interdependence.

Each RX clock EPLD also provides three outputs capable of driving the mode LEDs on the network interconnection panel. See documentation package for listings of EP600_RX_CLOCK files.



Note: All state machines are clocked at 16 MHz

Figure 3.5-9. Block Diagram RX Clock EPLD

The HDLC_IN signal from the I/O network is conditioned by the edge detect and deglitch state machine. The state transition diagram in figure 3.5-10 shows that only input logic levels that remain for longer than one clock cycle will be passed to the RXD output. Single clock cycle duration inputs that place the state machine in either state 010 or 101 are ignored. Edge detection occurs whenever the RXD output changes from 0 to 1 or 1 to 0. Both the RXD and EDGE outputs are delayed two to three clock cycles from HDLC_IN. The rising edge of the RXC signal sent to the DUSCC chip is synchronized with this delay, ensuring correct operation as shown in the timing diagram in figure 3.5-11.

According to AIPS I/O requirements in appendix B, the RX clock signal must not be synchronized until edges have been absent from the I/O network for at least eight RXC periods. The sync enable state machine shown in figure 3.5-12 is used to prevent erroneous synchronization. The state of EDGE is tested at the falling edge of each RXC (RX_SAMPLE). If no edge is present, the state machine advances to the next state until no edges have been detected for eight RX samples. The state machine stays in state 1000 until an edge is detected. Any edge occurring before reaching state 1000, regardless of RX_SAMPLE, resets the sync enable state machine to state 0000.

The RX clock generator state machine design guarantees that the RXD signal from the first state machine is sampled halfway between transitions. DUSCC chip input timing restrictions require that the minimum pulse width of the RXC output be at least 100 ns. The RXD output to the DUSCC chip receive input is sampled on the rising edge RXC. The state transition diagram of figure 3.5-13 and the timing diagram in figure 3.5-11 illustrate its operation. State 11 ensures that the clock generator meets DUSCC chip timing specifications.

The clock generator will not synchronize to an HDLC_IN edge unless the SYNC_ENABLE signal is present. Because incoming data transitions are not initially synchronized to the state machine internal RXC output, the RX

Input: HDLC_IN
Outputs: EDGE = H0 H1 $\overline{RXD}$ + $\overline{H0}$ $\overline{H1}$ RXD
RXD

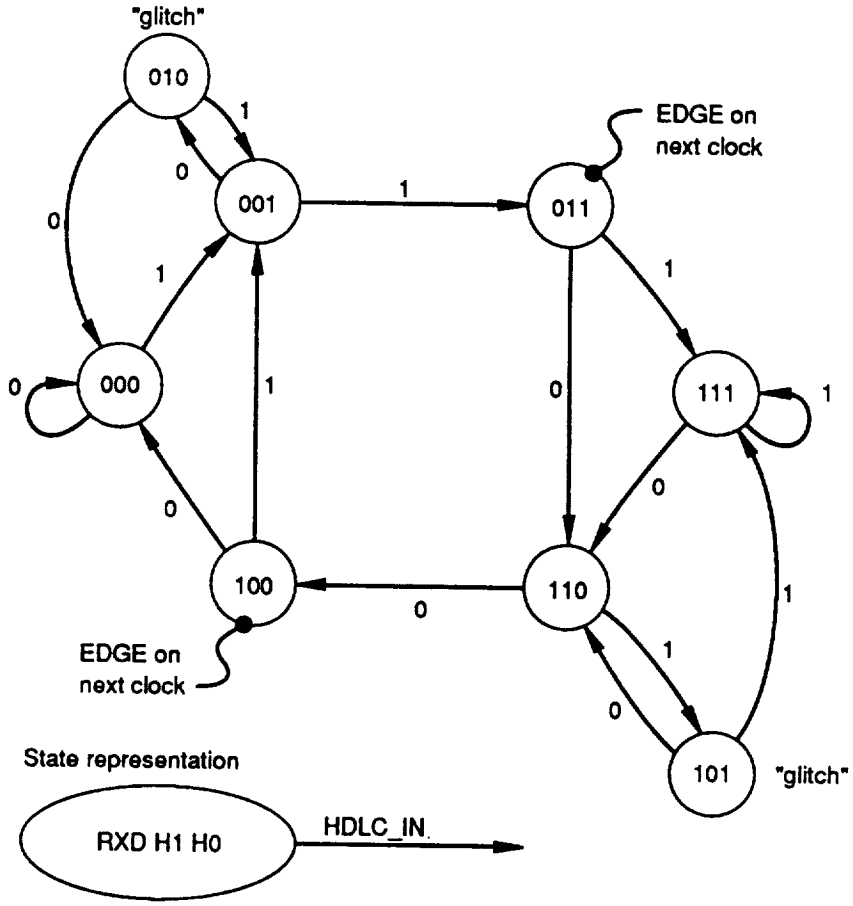


Figure 3.5-10. HDLC Input Edge Detect and Deglitch State Machine

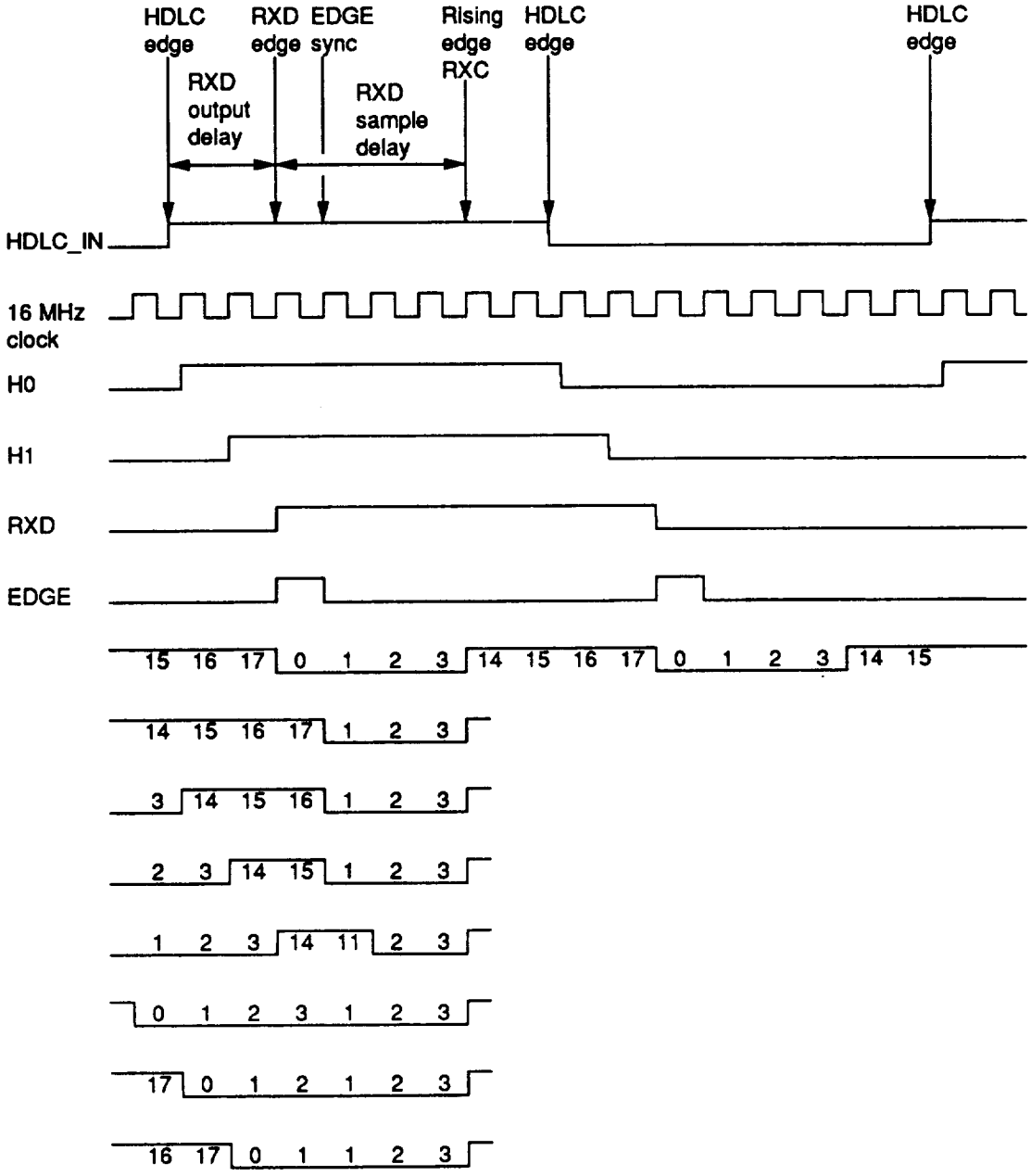
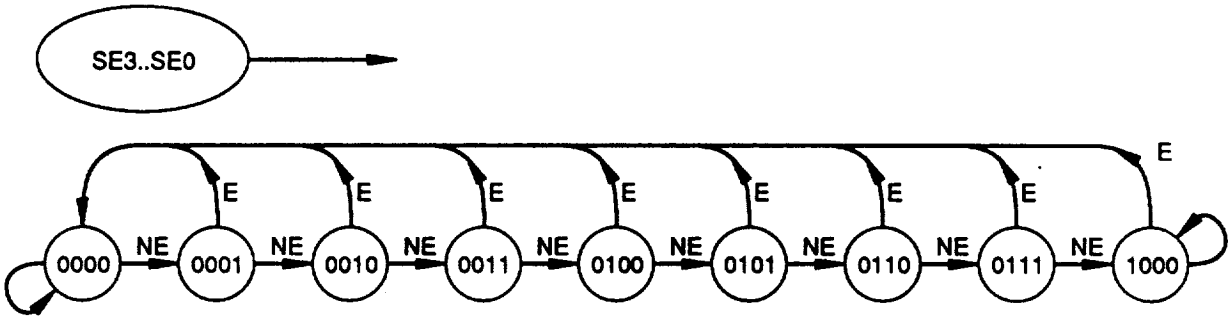


Figure 3.5-11. RX Clock EPLD Timing Diagram

Inputs: EDGE, RX_SAMPLE
Output: SYNC_ENABLE = SE3

State representation



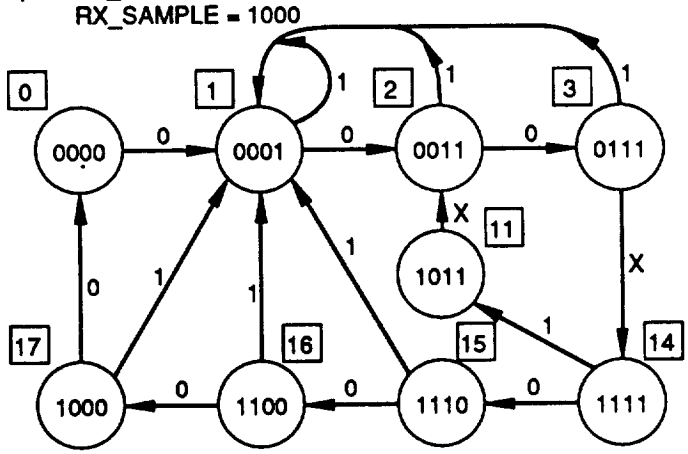
Notes:

1. SYNC_ENABLE is true at state 8 only.
2. NE = no edge was present at RX_SAMPLE of HDLC clock.
3. E = edge detected.
4. EDGE comes from edge detect and deglitch state machine.
5. RX sample comes from RX clock generator state machine.

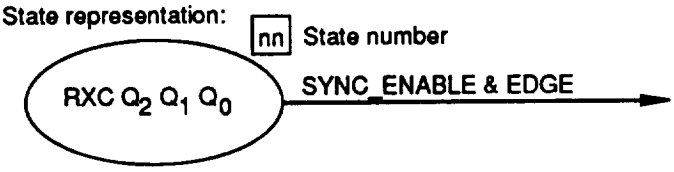
Figure 3.5-12. Sync Enable State Machine

Inputs: EDGE, SYNC_ENABLE

Output: RX_CLOCK
RX_SAMPLE = 1000



State representation:



Notes:

1. SYNC_ENABLE comes from the sync enable state machine.
2. EDGE comes from the HDLC input edge detect and deglitch state machine.
3. X means don't care.

Figure 3.5-13. RX Clock Generator State Machine

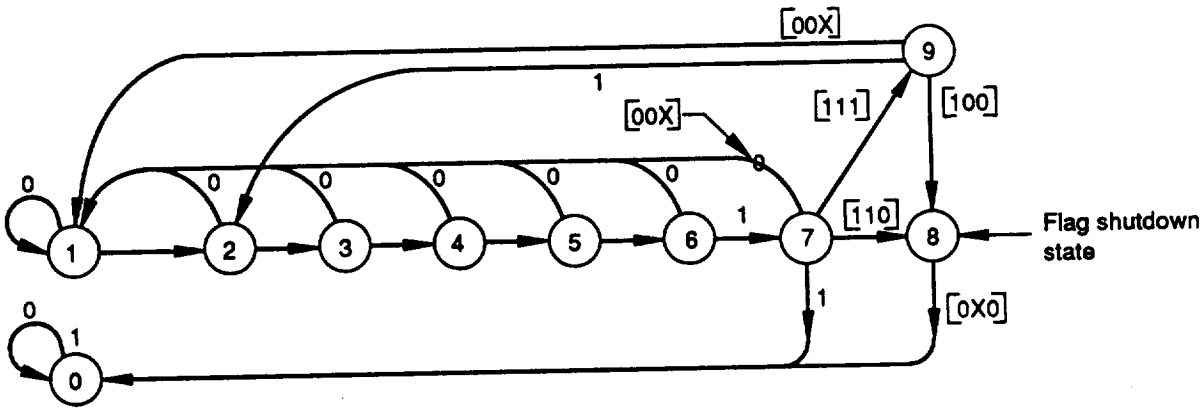
clock state at the input synchronizing edge cannot be predicted. The timing diagram in figure 3.5-11 shows the action of the state machine for all possible HDLC_IN edge/RX clock state conditions.

TX Clock EPLD. An Altera EP320 EPLD implements a state machine used to generate a common transmit clock for all the DUSCC chips and flag shutdown EPLDs on the ISIO daughter board. The DUSCC chip outputs TX data on the falling edge of the external transmit clock. The output delay from the falling edge of the clock is too long to guarantee adequate data setup time for the flag shutdown EPLD, which samples the TX data output on the rising edge of the TX clock. To allow adequate data setup time for the flag shutdown EPLD and still meet DUSCC chip external clock specification, the transmit clock is high for 125 ns and low for 375 ns. This allows maximum setup time in the flag shutdown EPLD while still meeting DUSCC chip requirements.

This EPLD also generates chip select signals for both the existing ISIO-2 68230 PI/T chip (U100) and the new PI/T chip (U20) added to the daughter board.

Flag Shutdown EPLD. One of the requirements of appendix B is that I/O network lines be left in a logic low state, called flag shutdown. Flag shutdown must be synchronized with data sent from the ISIO DUSCC chips to prevent spurious data from appearing on the I/O network. Because of the high output data rate, it is not possible to control the DUSCC chip accurately enough to guarantee these requirements without additional hardware.

The flag shutdown EPLD uses an Altera EP320. The state machine in the EPLD detects HDLC flags in the transmitted output data stream. When the flag shutdown enable (FSE) input is high, the state machine searches for an output flag that meets shutdown requirements. When the FSE input is low, the state machine searches for the conditions necessary to reconnect the DUSCC chip output to the I/O network. Figure 3.5-14 shows a state transition diagram for this state machine.



Abbreviations:

FSE_L	latched flag shutdown enable
L	input logic level
TXD	HDLC transmit data

Figure 3.5-14. Flag Shutdown State Machine

The flag shutdown EPLD also ensures that its associated DUSCC chip transmitter output is not connected to the I/O network at power up. Two inputs ensure that this does not happen: !SYS_RESET is brought low whenever the VMEbus RESET signal is active; FORCE_FSD is connected to an output of the additional PI/T, which always is pulled high at system reset. The local ISIO-2 68010 CPU must program the FORCE_FSD line low before the flag shutdown enable input can be recognized by the EPLD.

Before FORCE_FSD is pulled low, the TX_CLOCK input must be present to the flag shutdown EPLD to ensure that the EPLD remains in the flag shutdown state. Failure to follow this sequence will cause the DUSCC transmitter output to be connected to the I/O network on initial power-up, potentially corrupting the entire I/O network.

LED Driver EPLD. Several bi-color (red/green) LEDs are provided on the network interconnection panel to provide status information concerning operation of the DIU simulator. Two of these EPLDs are provided to control the eight-channel status LEDs. See LED_DRIVER EPLD listings for details.

Simulated Node EPLD. A Cypress Semiconductor 22V10 EPLD is used to condition DUSCC chip inputs and outputs to support the required operating modes of the DIU simulator.

Normal mode directly connects the inputs and outputs of each DIU simulator channel to its appropriate differential driver/receiver and I/O connector.

The partially simulated network of the small-scale system required a full complement of DIU simulators, but with reduced interconnection capabilities. To meet this requirement, the node mode of the 22V10 mixes data in a way similar to an AIPS node; each DUSCC input channel receives the output of all other DUSCC channels on the board plus input signals from the active I/O connectors.

In probe mode, the 22V10 disables transmitter output to the two active I/O connectors. It also combines input and output data that appear on the active I/O connector input and output pins, enabling one DUSCC chip to monitor all data on the I/O network. (This mode of operation is used only for troubleshooting I/O network problems and is not used during experiments.)

Delay Generator. Experiment timekeeping in the DIU simulators is performed by the 24-bit timer in the added PI/T. The input timebase is the reference fault-tolerant clock (FTC) signal on the P2 connector experiment bus.

When a PI/T timer is read by the local CPU, the three bytes of the 24-bit timer must be read individually. There is no way to snapshot the value in all three bytes of the timer with a single operation. This can lead to rollover errors caused by reading the timer while its value is changing. Four consecutive bytes are allocated to the timer in the PI/T, allowing use of the 68010 MOVEP.L instruction to obtain all timer bytes with a single, noninterruptible instruction, the rollover problem is still present, however. The first byte read by the MOVEP.L instruction is a dummy byte, which always returns a value of 0.

A delay generator EPLD using an Altera EP600 was designed that monitors the dummy read address of the timer. When a dummy byte read access is detected, the input FTC is disabled for the four read cycles of the MOVEP.L instruction necessary to obtain the dummy byte and three timer bytes. Any FTC pulse that occurs during the disable period is made up with an extra clock pulse at the end of the last read cycle. This guarantees that the timer bytes will never be read while they are changing and that no input FTC pulses will ever be missed.

External timer input restrictions for the PI/T limit its clock period to eight times the PI/T clock period. The ISIO-2 board PI/T chips use a 7.3728 MHz clock, limiting the external timer input to 1.09 µs. Because the delay generator chip may insert an extra clock pulse after a timer

read cycle, the input period must be greater than the duration of the four read cycles generated by the MOVEP.L instruction + 1.09 μs. For the ISIO-2 board, this restricts the external timer input period to 3.26 μs or longer. The AIPS FTC period is between 4 and 4.25 μs, which meets these restrictions.

The delay generator EPLD also synchronizes the start of timekeeping in the DIU simulator. Inputs are provided that inhibit the FTC until the local inhibit signal is removed and both the VME sync and FTP sync signals from the experiment bus are at the RUN state.

3.5.5 I/O Network Fault Inserter

Additional details of fault inserter construction are located in the documentation for the wire wrap board in documentation package E.

The eight network fault insertion channels are located on the VMEbus wire wrap board and are controlled by signals from the experiment bus described in appendix G. Figure 3.5-15 shows an overall block diagram of the wire wrap board. Figure 3.5-16 shows the design of a network fault insertion channel in greater detail.

Two Altera EP320 EPLDs are used for each fault insertion channel. The NET_FAIL_SELECT EPLD contains a 3-bit logical address register used to map the physical channel address to a logical address. The NET_FAIL_MODE EPLD contains a 4-bit fault register that controls the in and out faults. Channel physical addresses are hard-wired to the SA0-SA2 inputs of the NET_FAIL_SELECT EPLD.

To initialize a physical fault insertion channel, its logical address register and fault register must be programmed. This can only occur while the V-SYNC line is low (simulation host VME sync line in STOP state).

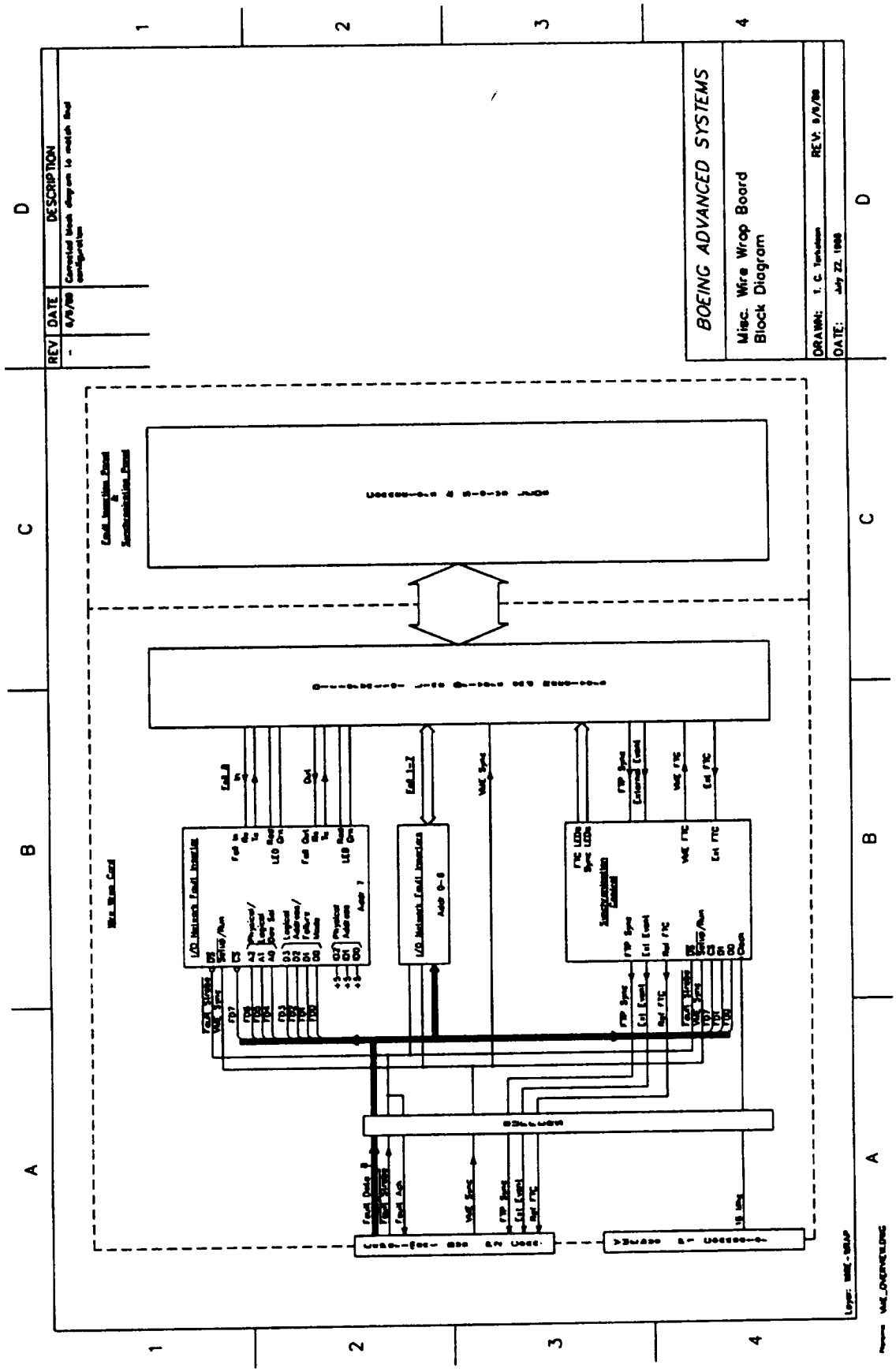


Figure 3.5-15. Fault Insertion, Control Interface Detailed Block Diagram

ORIGINAL PAGE IS
OF POOR QUALITY

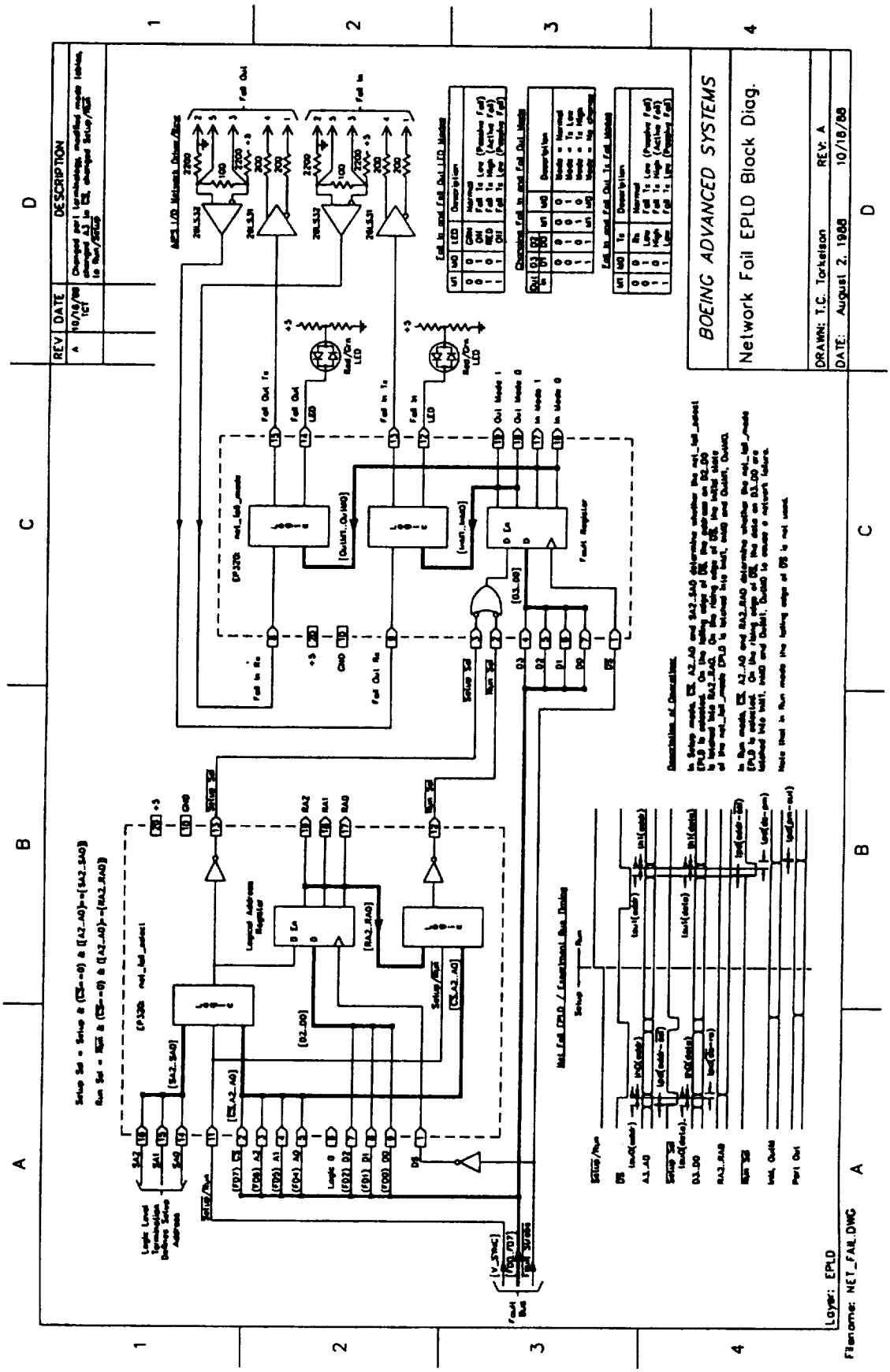


Figure 3.5-16. Network Fault EPLD Block Diagram

ORIGINAL PAGE IS
OF POOR QUALITY

Initialization proceeds as follows:

a. With Fault_Strobe high, the fault bus is set to select the desired physical to logical address mapping: FD7 low, FD4-6 specifies the physical channel address, and FD0-2 specifies the logical address.

b. The Fault_Strobe is brought low to store the logical address in the logical address register of the NET_FAIL_SELECT EPLD.

c. The initial condition for the NET_FAIL_MODE EPLD is placed on FD0-3 while the Fault_Strobe is still low. The state of FD4-7 remains unchanged.

d. The Fault_Strobe is brought high, storing the initial fault condition in the fault register of the NET_FAIL_MODE EPLD. Note that the initial fault condition is written to a physical address, not a logical address.

When the simulation computer enters the run state, the V_SYNC line in the experiment bus goes high. This changes the fault insertion system from physical to logical address control.

To cause a fault to a logical address the following actions occur:

a. The fault bus is set to select the desired logical address and fault condition: FD7 low, FD4-6 specifies the logical address, and FD0-3 specifies the desired fault condition.

b. The rising edge of the Fault_Strobe stores the fault condition in the fault registers of all channels programmed to the selected logical address, causing the fault.

Note that the fault bus can be set up with Fault_Strobe at either a high or low level and that only the rising edge of the Fault_Strobe has any

effect. In practice the Fault_Strobe is always left high when the fault bus is not in use.

3.5.6 Miscellaneous Wire Wrap Board Functions

Detailed documentation of the wire wrap board is found in documentation package E.

The miscellaneous wire wrap board functions are implemented with two Altera EP320 EPLDs (FTC_GEN and FTC_CONTROL) and AIPS I/O compatible differential line drivers, receivers, and terminators.

VME FTC Generation. The FTC_GEN EPLD is primarily used to generate a 4.125 µsec period FTC that can be used for the timebase of the simulation host. This clock source is available on the front panel of the wire wrap board. A bi-level LED indicates amber when the clock is operating correctly. The output of the VME FTC is high for 2.0 µs and low for 2.125 µs to match the normal operation of the AIPS FTP FTC.

VME Sync. VME sync is generated by PI/T J1 H4 output on the OPIO-1 board and routed to the wire wrap board in the experiment bus. It is converted to a differential output signal that meets the AIPS I/O network requirements of appendix B and routed to an AIPS I/O connector the front panel of the wire wrap board. The FTC_CONTROL EPLD drives a bi-color LED on the front panel that indicates the state of the VME sync signal: red for V_STOP, green for V_RUN.

FTP Sync. The FTP Sync signal originates on an AIPS I/O connector on the front panel of the wire wrap board. AIPS I/O network-compatible differential driver and termination resistors convert it to an experiment bus level signal. The FTC-CONTROL EPLD drives a bi-color LED on the front panel that indicates the state of the FTP Sync signal: red for F_STOP, green for F_RUN.

Reference FTC Select. The FTC_CONTROL EPLD is used to select the source of the reference FTC used for the experiment timebase in the simulation host. Either the VME-generated FTC or an external FTC can be selected. As with fault insertion setup, the reference FTC source can only be selected when the V_SYNC line is low.

To select the FTC source and external event polarity in the FTC_CONTROL EPLD:

a. With FSTB_N high, set FD7 high and set FD0 low to select the internal FTC or FD0 high to select the external reference. Set FD1 to the desired external event polarity (see below).

b. Cycle FSTB_N low, then high, while keeping the FD0 and FD7 values stable.

Two LEDs on the wire wrap front panel indicate the clock reference selection. The VME FTC bi-color LED, driven from the FTC_CONTROL EPLD is illuminated when the VME FTC is selected. The EXT FTC bi-color LED, driven from the FTC_GEN EPLD, is illuminated when the external FTC is selected. The appropriate LED is amber when the selected FTC is functioning correctly and red or green when the selected FTC is stuck at logic high or logic low.

External Event Detection. AIPS I/O external event input is provided on the wire wrap board front panel. The active polarity of the event is programmable as described for FTC source selection, above. The logic level of the external event input on the experiment bus is inverted if FD1 is low at programming and non-inverted if FD1 is high. The external event bi-color LED reflects the logic level of the external event signal on the experiment bus. The LED is green for logic low and red for logic high.

The external event signal on the experiment bus is routed to PI/T J1 H3 on the OPIO-1 board.

The external event input was not used in the small-scale system.

3.5.7 VMEbus Computer Timekeeping

The master experiment clock used by the VMEbus system is the timer in PI/T J1 on the OPIO-1 board. A small daughter board adds a delay generator EPLD to control the action of the external timer input. Its output is bused to all other PI/T chip external timer inputs on the OPIO-1 board. The OPIO delay generator operates the same as the DIU simulator delay generator.

The timer in PI/T J3 is used for the fault injection delay timer.

3.6 SOFTWARE DETAILS

3.6.1 AIPS FTP System Services

AIPS system services were modified by CSDL to support the special requirements of small-scale system testing discussed in reference 1. Modifications included adding the means to:

a. Control the time phasing of FDIR execution in both the CP and IOP with respect to experiment start reference time.

b. Disable background self-test routines.

c. Select the amount of RAM included in the exhaustive RAM background self-test.

d. Disable I/O network spare link testing.

e. Selectively control the start of the CRT display tasks.

3.6.2 AIPS FTP Pseudoapplications

Requirements for pseudoapplication software is discussed in reference 1. Listings of pseudoapplications are included on the software tape (see appendix I).

3.6.3 VME System Kernel and Utilities

The VMEbus CPU-29 uses silicon software components from Ready Systems. The Ready Systems concept is that additional functions in EPROM can be added to a core operating system kernel (VRTX32) as required with very little need for custom configuration. The components can be verified to operate correctly independent of the hardware platform on which they will ultimately reside.

The Ready Systems silicon software components installed in EPROM include VRTX32, IFX, and RTscope. A board support package (BSP) from Ready Systems was modified to support the specific requirements of the general purpose test system. The Ready Systems Real Time C (RTC) library, although not a software component, was also placed in EPROM. The RTC library is a sharable library; its inclusion in EPROM allows smaller load modules to be developed for the CPU-29.

The IFX extension to VRTX32 provides MS-DOS compatible RAM disks and allows multiple user tasks to share physical devices such as the simulation host console. Each task can send messages to the console to log its progress.

Further information on the Ready Systems products used can be found in references 3 through 13.

To interface the VMEbus simulation system with the MicroVAX experiment host computer system, the VULTURE program was written. Portions of the VULTURE program reside in both the VMEbus system and the MicroVAX system. Appendix H presents a detailed discussion of this program.

Noncopyrighted portions of the VME system kernel and utilities are included in the software tape (see appendix I).

3.6.4 DIU Kernel

The ISIO-2 board from Force Computers comes with a small operating system kernel that was not adequate for the operation of the DIU simulators for the small-scale system. The firmware operating system was only used to load the DIU kernel, at which point the onboard ISIO-2 firmware was disabled and DIU simulator software was loaded. All kernel functions are implemented as TRAP instructions. User access to kernel functions is via macros that complete the setup for the traps. COMMAND_KERNEL.INC on the software tape discusses the operation of the specific macros.

All DIU kernel operation is polled; no interrupts are used. This allows minimum latency for user programs that use interrupts and kernel functions.

The kernel operates in supervisor mode; user programs may operate in either supervisor or user mode.

3.6.5 DIU Simulator

The core source files for creating DIU simulator software are DIU_INIT, DIU_START, and DIU_SVC. These files are linked with NxDIUy files that define unique DIU configurations. Four unique DIU configurations were created for the small-scale system and reside in absolute files N1DIU1.ABS, N1DIU2.ABS, N2DIU1.ABS, and N2DIU2.ABS.

The NxDIUy.SRC files are created by a DEC C program called MAKE_FRAME_FILES that takes frame definitions from FRAME_DATA.C, DIU.H, and unique DIU definition NxDIUy.DEF files to create the source files. The source files are assembled using the Microtec Assembler on the development host to produce linkable DIU configuration files for inclusion with the core DIU files.

If modification of transaction definitions or timeouts is required, the FRAME_DATA.C file is modified, recompiled, and MAKE_FRAME_FILES is re-linked.

To modify the allocation of DIU addresses to simulator board absolute files the NxDIUy.DEF files are modified, and the DCL command file MAKE_TABLES.COM is used to create the source and object files. Command file LINK_DIU.COM creates the new DIU absolute files.

The new absolute files are copied to the experiment host computer for loading in the simulation host and DIU simulators.

The new set of DIU absolute format files must be converted to executable form in the simulation host and saved on the experiment host hard disk as follows:

a. DOWN_LOAD NxDIUy.ABS NxDIUy.ABS downloads the absolute file to the DRAM: disk in the simulation host.

b. VCONVERT NxDIUy.ABS NxDIUy.EXE converts the Motorola S record format absolute file to an executable image file. Note that the VCONVERT command automatically makes the executable image file a contiguous file.

c. UP_LOAD NxDIUy.EXE NxDIUy.EXE saves the executable image in the experiment host.

DIU simulator software uses a double buffering scheme in the ISIO-2 dual port RAM to collect data during experiment runs. Each buffer is capable of holding 46 kB of data. A handshake scheme with the VMEbus computer was established to allow control of the buffers and to notify the VMEbus computer when a buffer was full. Software locks prevent data corruption in either buffer while allowing real-time buffer flushing by the VMEbus CPU and buffer filling and swapping by the ISIO-2 CPU.

Small scale system tests were short enough in duration that the DIU simulator buffers did not overflow during normal operation and were unloaded at the completion of each experiment run.

The DIU_START module controls the sequencing of operations of the DIU simulator during an experiment. It is responsible for calling subroutines in the DIU_INIT file which set up peripheral chips, initialize interrupts, disable interrupts, etc. The DIU_START module also signals the VMEbus CPU to perform a final buffer flush at the end of an experiment run before returning DIU simulator operation to the DIU kernel. The DIU simulator software is interrupt driven during experiment operation. Interrupt service routines in the DIU_SVC module are used to remove data from the DUSCC chip FIFOs when a frame passes address screening. The DIU_SVC module also validates a received frame, prepares an appropriate response, logs data and errors, and controls the eight channel LEDs on the front panel.

3.6.6 I/O Network Probe

I/O network probe software used all the core DIU simulator routines, replacing DIU_SVC with FAST_PROBE_SVC. No address screening is used in the probe; all data received is logged in the double buffer scheme. Because of the amount of data collected by the probe, the VMEbus CPU is required to flush buffers in real time.

The turn around time of some of the network transactions is so fast that the probe software may not have time to recover and may erroneously record bad data. The validity of probe data can be determined by comparing DIU simulator and probe data and by the selected probe location in the network.

Operation of the probe during small scale system integration showed that the buffers could hold a maximum of 30 seconds of data before overflow occurred. The VMEbus CPU polled each probe every 500 msec to ensure that its buffers were promptly flushed.

3.6.7 DIU Data Formatting

Data from DIU simulators was recorded in the ISIO-2 dual port RAM in binary form to maximize storage capacity. When data are removed from the dual port RAM, the VMEbus CPU program first removes the binary format data to a temporary binary file in DRAM: disk. At the completion of an experiment, the temporary file is then converted to an ASCII data file in the DRAM: disk. Only the DIU simulator data fields required by the data analysis program are saved.

The UNLOAD program operates slightly differently on probe data. It is used in real time during an experiment to remove data from the ISIO-2 probe buffers. Post experiment formatting saves all data for later use.

3.6.8 Fault Insertion Control

The fault insertion control program, FAULT, is an optional program for use only during experiments requiring I/O network fault insertion. It runs autonomously after it is loaded and started on the VMEbus CPU. It requires that a FAULT.DAT file be present on the DRAM: disk which defines the data required to initialize each fault channel, perform physical to logical mapping, set up a time delay to the fault, and define the fault condition.

3.6.9 VME Experiment Control

A master simulation host experiment control program (CONTROL) was used to sequence the simulation host through experiment synchronization handshakes discussed in appendix A. The CONTROL program uses a CONTROL.DAT file in the simulation host DRAM: disk to determine which DIU simulators are active. This file must agree with the actual use of DIU simulators controlled by the experiment control command files.

The CONTROL program controls the VME Sync output to the FTP. It also

monitors the FTP Sync input. The program can be aborted at any time it is active by manually cycling the VME Sync output using VGO and VNO commands.

3.6.10 MicroVAX Interface Software

The MicroVAX interface to the VMEbus simulation computer is controlled by VULTURE software which resides in both the VMEbus computer and on disk in the micro vax. Appendix H discusses this software in detail.

The DRQ3B interface to the VMEbus system is controlled by a DEC supplied driver. DEC C programs were written to interface VULTURE protocol to the DRQ3B driver.

Details on the operation of the VRIP interface which is used to control the FTP are available from CSDL.

3.6.11 Experiment Control Command Files

All experiment operation was controlled from the experiment host. When the experimenter logged in, the account which was active set up several VRIP related aliases. The first operation was to initialize the VRIP interface, providing access to the FTP in the system under test.

Following successful initialization of the VRIP system, the experimenter then set up the environment for data collection, executable image loading, and simulation computer control.

Several classes of DEC DCL command files were used:

a. VRIP initialization control files were supplied by CSDL and are used to set up the interface and screen for FTP control.

b. Definition command files created symbols which accessed command files. VULTURE.COM and SYMBOLS.COM set up the experiment environment.

c. Simulation computer loading was controlled by VME_LOAD_EXE.COM.

d. FTP computer loading was controlled by LD_xx.COM files in experiment directories.

e. FTP program patch files were used to correct problems in FTP IOP programs caused by the VRTX Ada compiler.

f. FTP experiment setup command files defined unique experiment parameters.

g. Program execution was controlled by the RUN_EXP command file. It accessed other command files which loaded DIUs and extracted data from both the simulation computer and the FTP.

h. FTP data collection was controlled by the GET_FTP command file. The command file used the known configuration of FTP memory to extract experiment data without additional operator input.

j. Simulation computer data collection was controlled by the UNL_DIU command file. It controlled the post experiment operation of the UNLOAD programs.

It was not possible to totally automate the control of small scale system testing because of the need to manually record the FTP logs.

REFERENCES

1. G. C. Cohen, et al., **Design of an Integrated Airframe/ Propulsion Control System Architecture,** NASA CR-182004, March 1990.

2. DRQ3B Parallel DMA I/O Module User's Guide, DEC order number EK-47AA-UG-001.

3. VRTX32/68020 Versatile Real-Time Executive for the MC68020 Microprocessor User's Guide, Software Release 1, Ready Systems document number 541331001, April 1987.

4. VRTX32 C Versatile Real-Time Executive User's Guide, software release 1, Ready Systems document number 542101001, April 1987.

5. RTscope 68000 Real-Time Debugger and VRTX32 System Monitor for Motorola 68000 Family User's Guide, Ready Systems document number 531311001, November 1987.

6. IFX I/O and File Executive for Real-Time Systems External Specification, Ready Systems preliminary document number 521311X07, March 1988.

7. Getting Started With Silicon Software Components, Ready Systems document number 590023004, July 1987.

8. How To Write a Board Support Package for VRTX, software release 3, Ready Systems document number 5900430003, November 1986.

9. VRTX Technical Tips, Ready Systems document number MC071000, September 1986.

10. VRTX and Custom Queues, Application Note, Ready Systems document number 40001, November 1983.

11. VRTX32/680x0 Timing Reference, software release 1, Ready Systems document number 540011001, May 1987.

12. Portable C RTL/68000 Installation Guide, Ready Systems document number 61A203002, June 1988.

13. RTC Run Time Library User's Guide, Ready Systems document number 615003004, June 1988.

14. Force Computers CPU-29 User's Manual.

15. Force Computers DRAM-E4 User's Manual.

16. Force Computers ISIO-2 User's Manual.

17. Force Computers MOTH User's Manual.

18. Force Computers OPIO-1 User's Manual.

19. Force Computers PWR-20 User's Manual.

APPENDIX A: SMALL-SCALE SYSTEM EXPERIMENT SYNCHRONIZATION

Introduction

The AIPS FTP and the VMEbus simulation computer require a means of signaling each other of their status for coordination of experiments. The start synchronization interface between them implements this requirement.

Start Synchronization Physical Interface

Interconnections

VMEbus to FTP: [V_SYNC] signals VMEbus simulation computer status
FTP to VMEbus: [F_SYNC] signals FTP status

Signal levels correspond to AIPS I/O network signal levels

V_RUN is logic high	F_RUN is logic high
V_STOP is logic low	F_STOP is logic low

Connectors, differential drivers, receivers, and terminators conform to requirements of reference 1.

VMEbus Simulation Computer States

Initialize [V_STOP]

VMEbus simulation computer hardware and software are initialized. Experiment clocks are initialized to 0.

Ready [V_RUN]

The VMEbus simulation computer signals that it is ready for simulation by changing [V_SYNC] from [V_STOP] to [V_RUN].

In this state DIU simulator initialization is complete. Data logging has not started and the experiment clock is not yet running.

Run [V_RUN]

Transition to the Run state is signaled by the FTP changing [F_SYNC] from [F_STOP] to [F_RUN].

The simulation computer and DIU simulator experiment clocks are started. Data logging starts here.

Abort [V_STOP]

[V_SYNC is manually changed from [V_RUN] to [V_STOP] to abort all VMEbus actions before the normal end of an experiment run. Data collection is terminated, experiment clocks are stopped, and all logs are scrubbed. No data are saved in the VMEbus system.

Halt_ACK [V_STOP]

In response to an FTP request to end the current experiment, [V_SYNC] is changed from [V_RUN] to [V_STOP], data collection is terminated, data logs are flushed, the ending time of the experiment is recorded, and experiment clocks are stopped.

Idle [V_STOP]

Experiment is complete. The VMEbus system waits for further user commands with experiment clocks stopped.

Error [V_STOP]

Reached because of a sequencing error during small-scale system initialization. The FTP must be manually halted by experiment operator.

AIPS FTP Computer States

Initialize [F_STOP]

AIPS FTP hardware and Ada software are initialized, up to the point of determining the absolute time for t0.

Wait [F_STOP]

The FTP waits in this state until [V_SYNC] is at [V_RUN]. If [V_SYNC] is already at [V_RUN], the FTP passes through this state to Ready without error.

Ready [F_STOP]

The absolute time for t0 is determined for scheduling FDIR and application tasks. All tasks are scheduled. A task that will place [F_RUN] line at [F_RUN] is created and scheduled to run 1 sec before t0.

Run [F_RUN]

The FTP signals the start of an experiment by changing [F_SYNC] from [F_STOP] to [F_RUN].

Halt [F_STOP]

Experiment has run for the specified duration. [F_SYNC] is changed from [F_RUN] to [F_STOP] to request the VMEbus simulation computer to end the experiment.

The FTP de-schedules user application tasks and performs any cleanup operations required.

Idle [F_STOP]

Experiment is complete. The FTP waits for further user commands.

Start Synchronization Protocol

Figures A-1 and A-2 illustrate the handshaking between the FTP and the VME system during experiment operation.

Normal operation

Both the VMEbus simulation computer and the AIPS FTP begin in their Initialize states with sync lines at [V_STOP] and [F_STOP] respectively.

The AIPS FTP proceeds to the Wait state when its initialization is complete regardless of the status of [V_SYNC]. If [V_SYNC] is at [V_RUN], it proceeds directly to the Ready state.

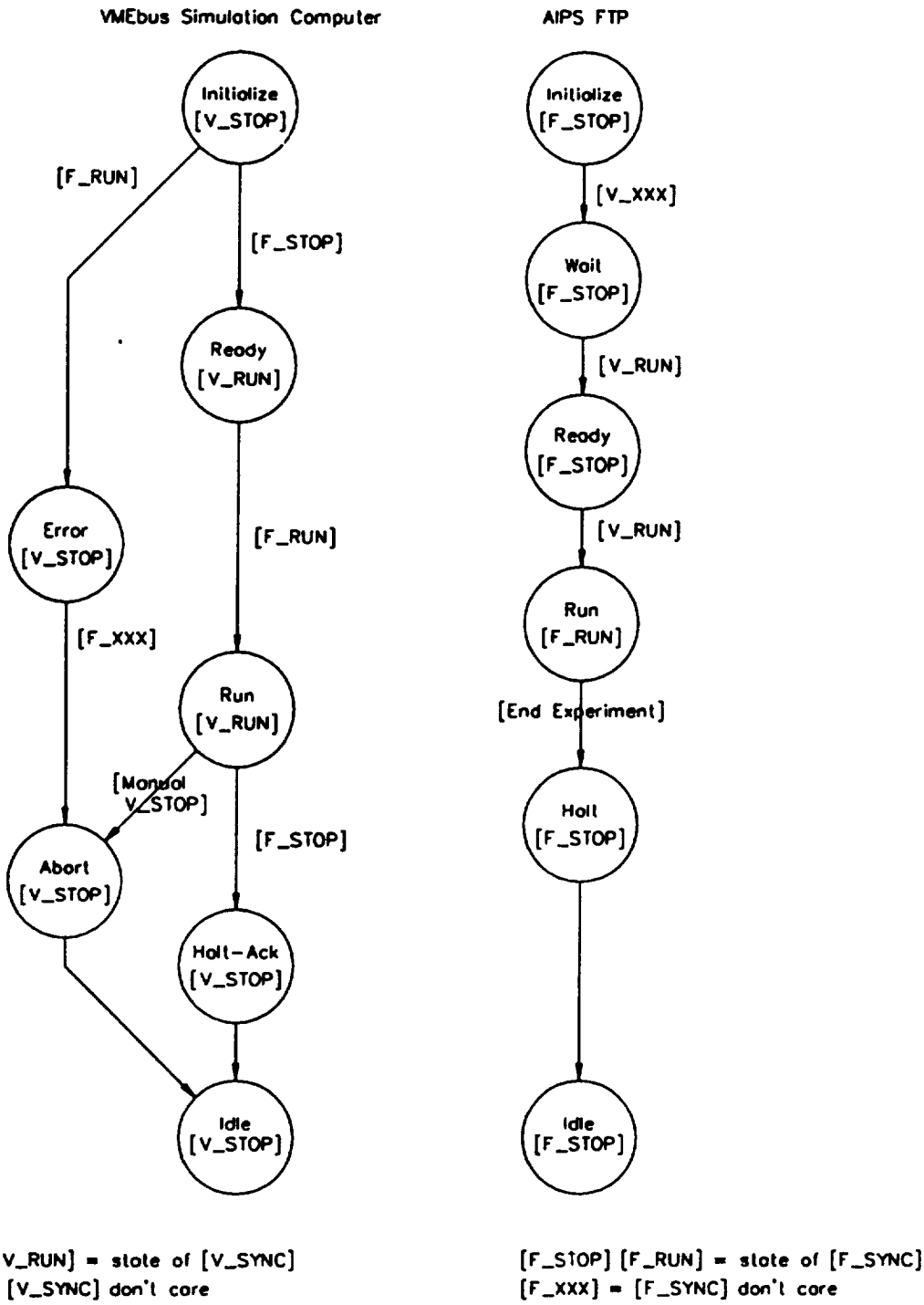
When the VMEbus simulation computer is ready it changes from the Initialize to the Ready state and signals the FTP by changing [V_SYNC] from [V_STOP] to [V_RUN].

In the Ready state the FTP schedules FDIR, application tasks, etc. The Run state may be entered immediately or after a delay to allow the system to settle into normal operation. The FTP signals that it has arrived at the Run state by changing [F_SYNC] from [F_STOP] to [F_RUN]. Transition to [F_RUN] in the small scale system occurs 1 sec before the FTP begins normal operation.

The AIPS FTP can request termination of an experiment by changing its F_SYNC line from F_RUN to F_STOP.

VMEbus-requested termination

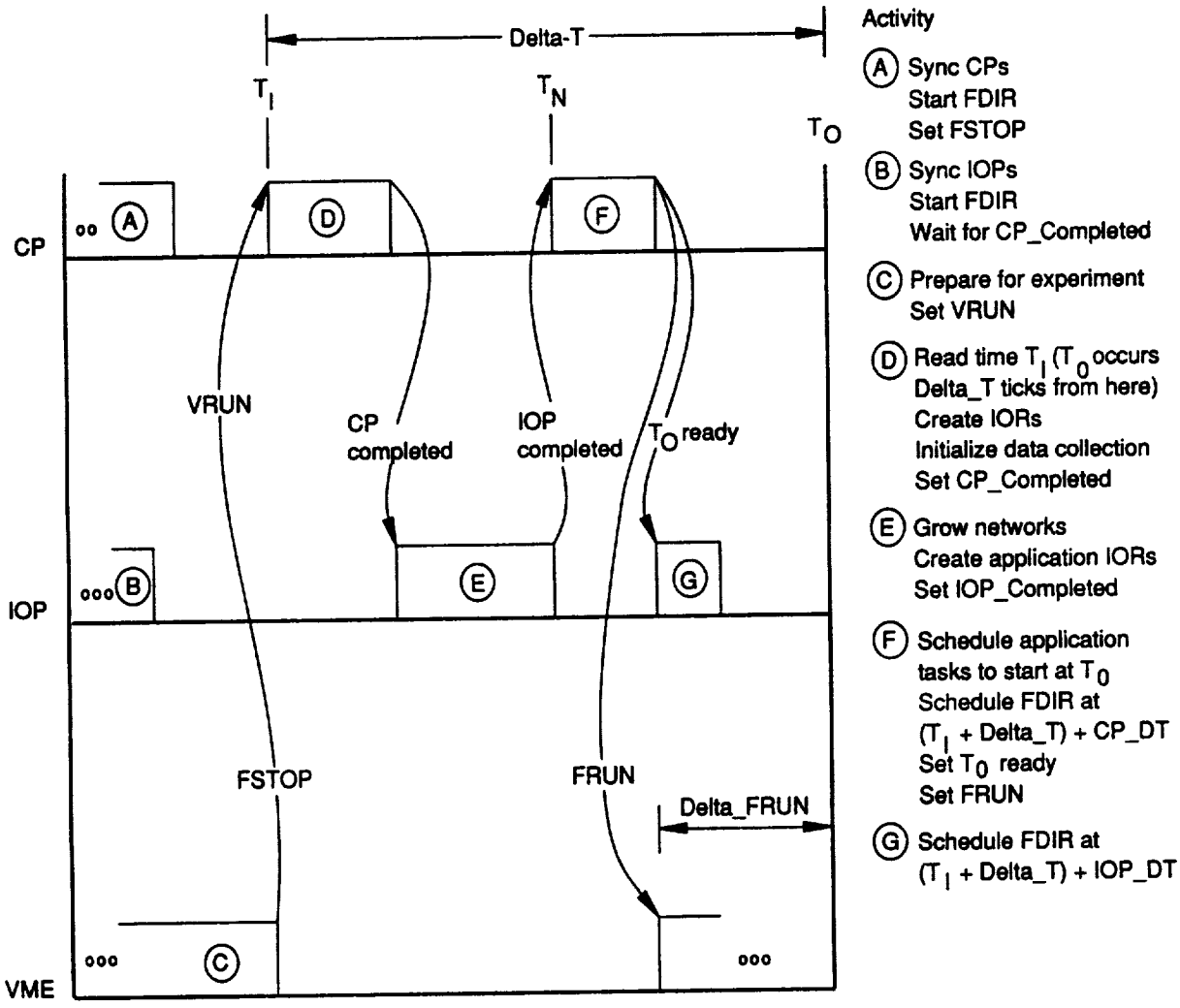
The VMEbus simulation computer is manually forced to abort its action and enters the Abort state, changing [V_SYNC] from [V_RUN] to [V_STOP], terminating data collection, scrubbing its logs, and stopping the experiment clocks. VMEbus computer enters the Idle state.



[V_STOP] [V_RUN] = state of [V_SYNC]
[V_XXX] = [V_SYNC] don't care

[F_STOP] [F_RUN] = state of [F_SYNC]
[F_XXX] = [F_SYNC] don't care

Figure A-1. VMEbus and FTP Experiment Handshaking and Synchronization



Notes:

- FRUN and FSTOP are states of the FTP sync line
- VRUN is a state of the VME sync line

Figure A-2. Small-Scale System Application Initialization

AIPS FTP requested termination

The AIPS FTP enters the Halt state, changes [F_SYNC] from [F_RUN] to [F_STOP], terminates data collection, flushes its data logs, and records the ending time of the experiment. It remains in the Halt state until the VMbus simulation computer signals that it is in the Halt_ACK state by changing [V_SYNC] from [V_RUN] to [V_STOP].

The experiment clocks are stopped by the change of the [F_SYNC] line, and the VMEbus computer and the AIPS FTP enter the Idle

Error operation

The only error considered in the accompanying state diagram is the error that occurs when the AIPS FTP is in the Run state before the VMEbus simulation computer enters the Ready state. When this occurs, the VMEbus simulation computer passes through the Error state, ensuring that [V_SYNC] is [V_STOP].

REFERENCES

1. AIPS I/O Network Interface Requirements.

APPENDIX B: AIPS I/O NETWORK INTERFACE REQUIREMENTS

1.0 AIPS I/O NETWORK

The AIPS I/O network is a dynamically reconfigurable communications network using modified HDLC synchronous serial protocol for data communications between FTPs, nodes, and DIUs. This document describes the physical interface and software protocol necessary to use the network.

2.0 PHYSICAL SPECIFICATIONS

2.1 CABLING

The FTPs, nodes, and DIUs that make up the AIPS I/O network are interconnected by two pair, AWG 24 twisted foil shielded pair cable (Belden Datalene 9729). Shields for each pair are isolated from each other. The nominal cable impedance is 100Ω, velocity of propagation 78%, 12.5 pF per foot between conductors and 22 pf per foot to shield. See figure B-1.

2.2 CONNECTORS

Patch cords and panels for the AIPS I/O network as implemented at CSDL use Switchcraft 5 contact DIN audio connectors. Cable connectors with socket contacts are Switchcraft P/N 06AL5F; panel connectors with pin contacts are P/N 57KD5M. See figures B-1 and B-2.

2.3 DIFFERENTIAL LINE DRIVERS

The AIPS I/O network uses RS-422-compatible differential line drivers (26LS31 or equivalent). The drivers are always connected to the

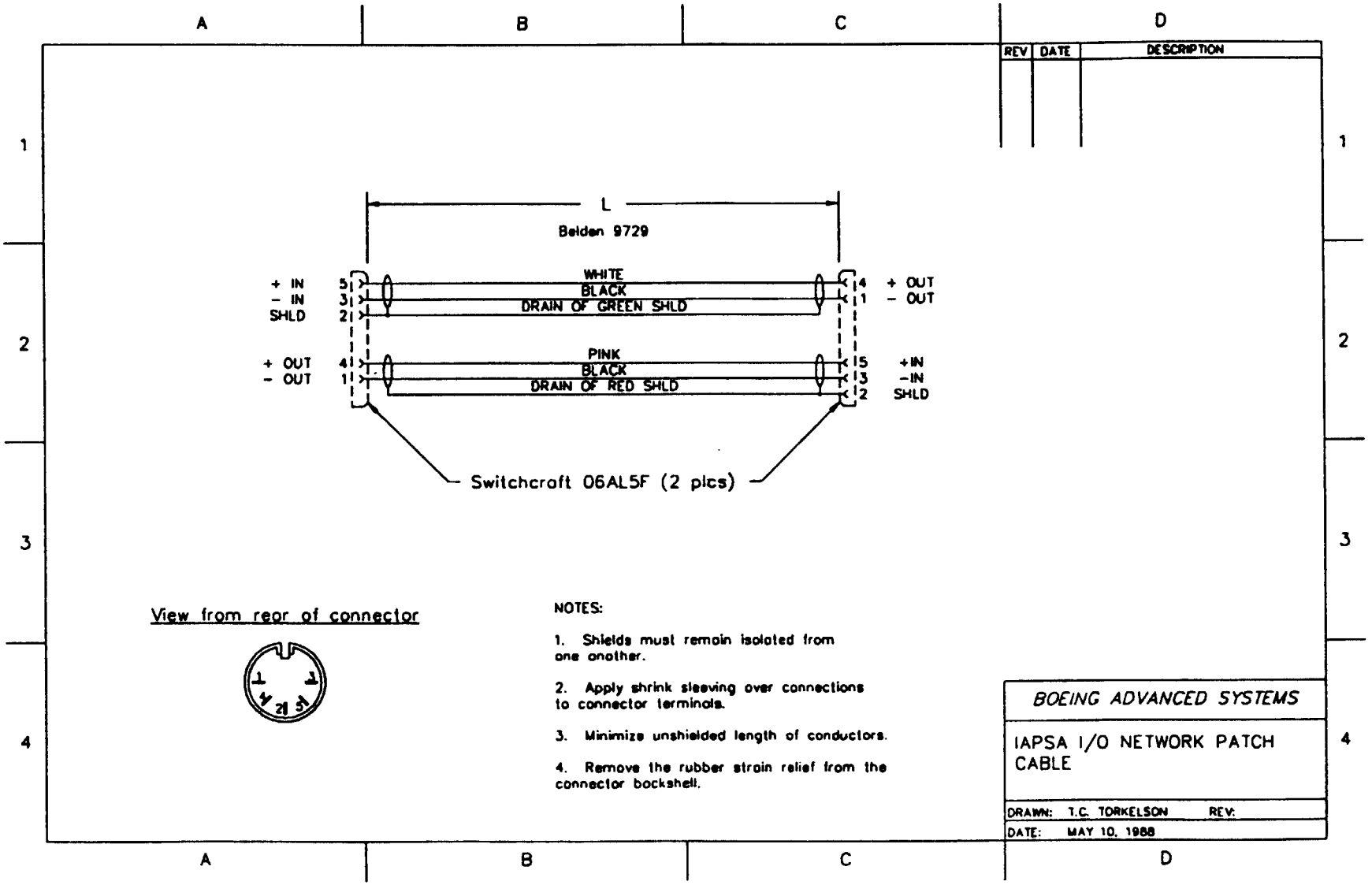


Figure B-1. IAPSA I/O Network Patch Cable

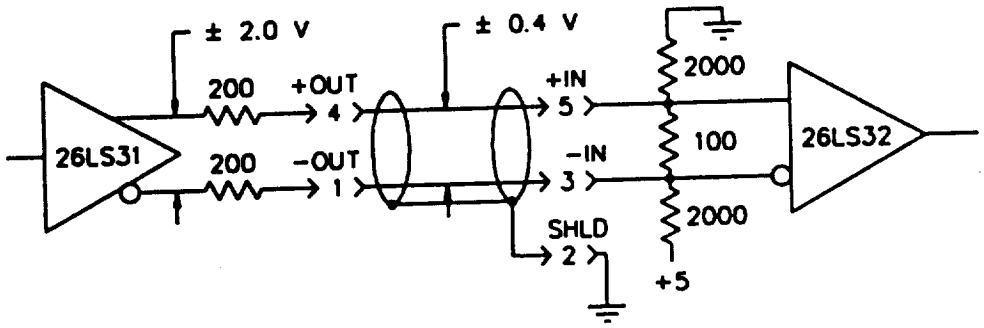


Figure B-2. AIPS I/O Network Interface

network and enabled; no tri-state operation is allowed. Minimum VOH for a driver output is 2.5V (3.2V typical); maximum VOL is 0.5V (0.32V typical).

Differential driver outputs are connected to the AIPS I/O network through 200Ω resistors that are the boundary of the AIPS IOS - AIPS I/O network fault containment region (see fig. B-2.)

2.4 DIFFERENTIAL LINE RECEIVERS

The AIPS I/O network uses RS-422-compatible differential line receivers (26lS32). The receiver differential input voltage sensitivity is +/- 0.2V minimum (+/- 0.06V typical) over a common mode voltage range of +/- 7V. Input hysteresis is typically 0.03V.

Interconnecting cables are terminated at the differential receiver with a 100Ω resistor connected between the noninverting and inverting inputs.

The noninverting input of each differential receiver is pulled to ground by a 2,000Ω resistor, and the inverting input is pulled up to +5V dc by a 2,000Ω resistor (see fig. B-2.)

2.5 I/O SIGNAL LEVELS

Signal levels on the AIPS I/O network with respect to line driver local common will be VOL = 1.30V, VOH = 1.70V worst case (VOL = 1.47V, VOH = 2.05V typical). The differential voltage between the signal lines in the interconnecting cables will be between +/- 0.40V worst case (+/- 0.5 V typical) assuming negligible cable resistance.

See figure B-2 for a typical differential driver/receiver configuration.

2.6 I/O LOGIC LEVELS AND NOISE MARGIN

Logic levels in this specification refer to the differential voltage levels between signal conductors in I/O network cables. Low is synonymous with logic 0; high is synonymous with logic 1.

A device driving I/O network signal lines must ensure that the noninverted signal line is at least 0.40V more positive than the inverted signal line for logic 1 and 0.40V more negative for logic 0.

A receiving device connected to the I/O network must detect a logic 1 when the noninverted signal line is at least 0.20V more positive than the inverted signal and detect a logic 0 when 0.20V more negative. The receiver should incorporate input hysteresis to minimize noise effects at switching thresholds.

These specifications ensure a differential noise margin of +/- 0.20V minimum.

2.7 COMMON MODE VOLTAGE RANGE

The voltage difference between the commons of interconnected elements of the AIPS I/O network shall be less than +/-7V.

3.0 NETWORK POLLING AND NETWORK TRAFFIC

3.1 NETWORK POLLING

Logic levels are used by the AIPS IOS to poll for unsolicited inputs.

Note: The small-scale system does not poll for unsolicited inputs.

3.2 NETWORK TRAFFIC

Network traffic is represented in NRZI format where a 0 is signified by a transition and a 1 is signified by the lack of a transition. NRZI data in the AIPS system is sent at at 2 megabits per second (Mbps). Network traffic is organized into HDLC frames, which are described below.

4.0 AIPS HDLC PROTOCOL

The AIPS HDLC protocol modifies standard HDLC protocol by adding "Flag Shutdown" and by deleting the "Abort" and "Idle" HDLC commands. Physical conditions that represent "Abort" and "Idle" are found on the AIPS I/O network; however, they do represent these commands.

4.1 HDLC DATA

An HDLC data stream is distinguished from an HDLC command by the number of consecutive NRZI 1s allowed. No more than five consecutive NRZI 1s are allowed for a valid HDLC data stream; after five consecutive NRZI 1s, a NRZI 0 is inserted (zero insertion). Inserted NRZI 0s are automatically removed by the receiving HDLC interface chip (zero deletion). Zero insertion and deletion guarantees that HDLC commands can always be distinguished from HDLC data and that edges are always available within a data stream for the synchronization of data clocks.

4.2 HDLC COMMANDS

HDLC commands contain at least six consecutive NRZI 1s. Only the HDLC Flag is defined in the AIPS HDLC protocol. HDLC Abort and HDLC Idle are not recognized by the AIPS system because of a conflict with the Laning Poll protocol. (See ref. 4, appendices A and C.)

a. HDLC Flag is composed of one NRZI 0, six NRZI 1s, and one NRZI 0. An HDLC Flag opens and closes all HDLC data frames.

b. HDLC Abort contains seven or more NRZI 1s. (Not used by AIPS.)

c. HDLC Idle contains 15 or more NRZI 1s. (Not used by AIPS.)

4.3 AIPS HDLC FRAMES

Two types of frames are used in the AIPS system: command frames and response frames. Command frames originate in the AIPS IOS, and response frames originate from either AIPS nodes or DIUs. Response frames occur only at the request of a command frame. (See transactions below.)

AIPS HDLC FRAMES. (See fig. B-3 and appendices A and C of ref. 4.)

Note: Appendices A and C of reference 4 do not agree with respect to maximum data field size for AIPS HDLC frames. It appears that the maximum data field length is between 117 and 122 bytes. small-scale system command and response frames are short enough that the maximum data field length is not approached, and the discrepancy will not be problem.

a. Flag (F) - opening HDLC flag (1 byte minimum; more than one flag may be sent).

b. Address (A) - command frames: identifies a destination AIPS node or DIU to which a frame is addressed. Hardware address screening is used by the physical interface devices in the nodes and DIUs to filter out frames addressed to other devices (1 byte).

Response Frames: identifies the responding device. No address screening is used in the IOS for response frames (1 byte).

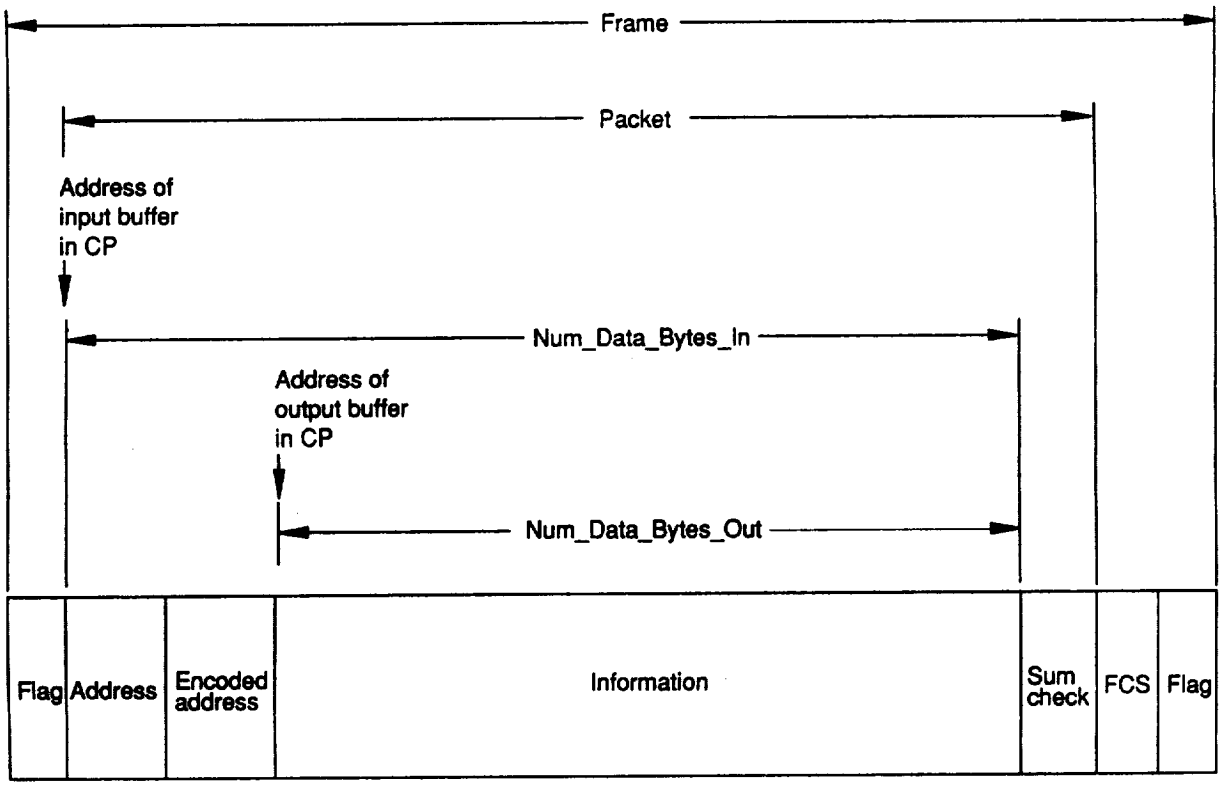


Figure B-3. HDLC Definitions

a. Control (C) - command frames: the 1S complement of the Address field (1 byte).

Response Frames: not defined.

a. Data (D) - One or more bytes of data, the last byte of which is a sum check that is defined to be the 2S complement of the modulo 256 sum of the A, C (if present), and preceding D field bytes. Note that the sum of the A field through the sum check byte is zero.

Command Frame: See note above regarding field length.

Response Frame: See note above regarding field length.

a. Residual bits (RB) - IOS/node communications: 3 residual bits.

IOS/DIU Communications. 5 Residual bits.

a. Frame check sequence (FCS) - 16-bit CRC-CCIT error checking field, using G(X)=X^16+X^12+X^5+1, dividend preset to 1, sent inverted by TX. A received CRC of F0B8 hex indicates valid data. All data between the opening and closing flags are used to compute the FCS (2 bytes).

b. Flag (F) - closing flag (1 or more bytes). (See Flag Shutdown, below).

4.4 DATA CLOCK SPECIFICATIONS

The transmit and receive data clock specifications are adequate to ensure sampling of incoming data at the nominal location, 50% between expected transitions, within +/- 12.5% of a bit time, for a frame length of at least 128 bytes.

4.4.1 Transmit Data Clock

NRZI data must be transmitted at 2 Mbps +/- 0.01%.

4.4.2 Receive Data Clock

To decode data from the I/O network, signal lines must be sampled by a clock synchronized to the incoming NRZI data stream.

After 4 µs of transition-free operation on the I/O network, synchronization of the receive clock is enabled. The first transition on the network after the 4 µs quiet time defines the middle of the sampling clock period; 0.25 µs (50% bit time) after the transition, the logic level on the network is sampled. Sampling continues every 0.5 µs (bit time) thereafter.

The local receive clock must be stable enough to maintain sampling 0.25 us after a transition with a maximum clock skew of +/- 0.125 µs. After synchronization, the receive clock must be stable to +/- 0.01%.

5.0 I/O NETWORK STATES

5.1 IDLE

For the small-scale system, the I/O network is idle if it has been low for at least 4 µs.

5.2 POLL

Not used in small-scale system.

5.3 BUSY

The I/O Network is busy if HDLC data or flags are being sent (a transition has been detected within the last 4 µs).

5.4 STUCK

The network is stuck if it has been high for 4 μs or more.

6.0 NORMAL SMALL-SCALE SYSTEM NETWORK STATE TRANSITIONS

6.1 IDLE TO BUSY

Transition from the Idle state to the Busy state occurs when the network goes from logic 0 to logic 1 on the rising edge of a Flag byte. At least one complete Flag byte must appear on the network before the address field is sent. (See fig. B-4.)

6.2 BUSY TO IDLE (FLAG SHUTDOWN)

Transition from the Busy state to the Idle state is called flag shutdown.

The network is set to logic 0 on the falling edge of a Flag byte. At least one complete Flag byte must be sent at the end of an HDLC frame before the network is placed in Idle. (See fig. B-4.)

7.0 I/O NETWORK TRANSACTIONS

7.1 OUTPUT TRANSACTION

An output transaction is a single HDLC frame transmitted by an FTP to a specific AIPS node or DIU for which a response frame is not required. A typical output transaction sends a command to a DIU.

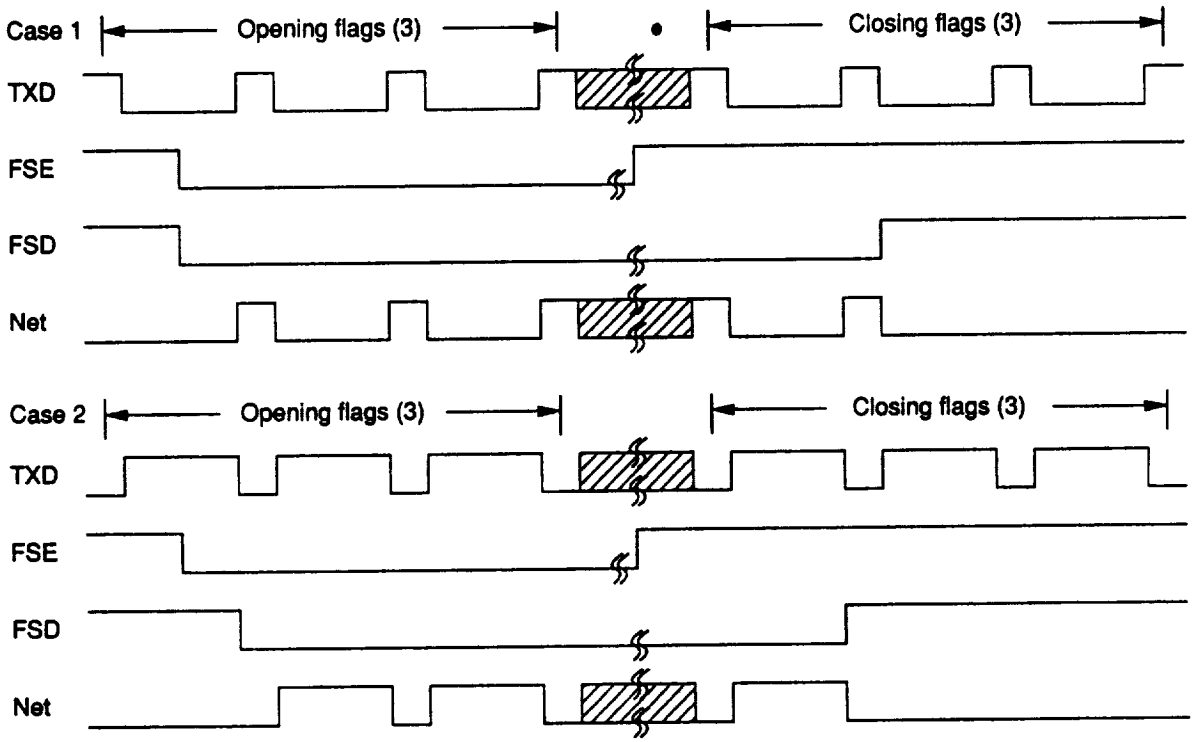


Figure B-4. Flag Shutdown Waveforms

7.2 INPUT TRANSACTION

An input transaction involves two HDLC frames. The first frame is a command frame transmitted by the FTP to a specific node or DIU. It may contain commands and/or request information from the addressed device. The second frame of the transaction is a solicited response frame from the addressed device that is sent to the FTP. The FTP receives all response frames regardless of the address field of the frame.

7.3 CHAINS

A chain is an ordered group of transactions sent by an FTP over the I/O network. Chains are typically associated with application processes that run at different application frame rates. (Not to be confused with HDLC frames.) Chains may contain output, input, or a mix of transaction types.

Chains either run to completion or are terminated by the sending FTP when faulty responses are received.

8.0 DATA POLARITY

AIPS I/O network data passed between the FTP and the IOS are inverted with respect to the definition of NRZI 1 and NRZI 0, above. This requires inversion of both transmitted and received data by devices connected to the AIPS I/O network that use HDLC interface chips with noninverted data buses.

The HDLC interface chip used by AIPS has an inverted data bus. This is transparent to AIPS elements because they all use the same interface chip; however, other designers using newer HDLC interface chips with a noninverted data bus will be required to invert data in software before it can be used.

The inversion of FTP data on the I/O network is not specifically stated in any known CSDL AIPS documentation. It can be inferred from a statement in paragraph 5.1.2 of appendix C of reference 4: "Definitions of bit polarity and sense have been modified to reflect what is seen by the AIPS system."

The effect of this bus inversion is as follows for an HDLC interface chip with a noninverted data bus:

a. A field - the device address must be inverted for a match. For example, if a device is to have address 16#80#, the address must be programmed as 16#7F#.

b. C field - for a command frame the encoded address must be inverted. (This results in the control field being the device address.)

c. D field - all data must be inverted.

d. SC field - sum check must be computed for noninverted A, C, and D field data, then inverted.

e. RB field - no action required as value does not matter.

f. FCS field - no action required; this value is computed within the HDLC chip and is correctly sent and received by either inverted or non-inverted bus HDLC interface chips.

REFERENCES

1. IAPSA II Small-Scale System Description.

2. IAPSA II DIU Simulator Specifications - VMEbus Implementation.

3. VME Simulation Computer Experiment Bus Specifications.

4. NASA Contractor Report: Advanced Information Processing System: Input/Output System Services, The Charles Stark Draper Laboratory, Inc., Cambridge, MA, contract NAS1-18565, March 1989.

APPENDIX C: SMALL SCALE SYSTEM DIU SIMULATOR

1.0 INTRODUCTION

Implementation of the small-scale system of reference 1 requires simulation of DIUs. DIUs interface sensors and actuators to the AIPS I/O network and AIPS FTP.

The DIU simulator must be compatible with the hardware and software protocols of the AIPS I/O network as described in references 2 and 5. Their operation must also obey the experiment configuration and control requirements of references 3 and 6.

2.0 I/O NETWORK INTERFACE REQUIREMENTS

See reference 2.

3.0 DIU SIMULATOR OPERATIONAL REQUIREMENTS

3.1 COMMAND/RESPONSE PROTOCOL

Each DIU in the small-scale system must have a unique HDLC address.

All transactions in the small-scale system are input transactions that consist of both a command frame from the FTP and a response frame from the DIU simulator. Reference 5 specifies the command and response frame formats used in the small-scale system.

A DIU simulator must screen each frame to determine whether it must respond. A DIU simulator may only respond to command frames that include its unique address. The HDLC address and encoded address fields should be screened using hardware address screening capabilities typically found in HDLC interface chips.

After a command frame passes address screening, the following conditions must be met before a response frame is generated:

a. The sum check of all data received must be correct. (See ref. 2.)

b. No hardware detected errors may be present. (See below.)

c. The correct number of residual bits must have been received.

d. The frame ID portion of transaction must be defined for this DIU address.

e. The correct number of bytes must have been received for the frame ID.

When a command frame meets validation criteria, the response frame defined in reference 5 must be sent.

3.2 DIU RESPONSE TIME

The time required for an addressed DIU to validate a command frame and begin transmission of a response frame to the FTP is called DIU response time. An FTP that has requested a response frame will time out the transaction if an addressed DIU does not reply within a user specified time limit.

A DIU must respond no sooner than 4 µs and no later than 512 µs after the closing flag of a command frame. Individual DIUs in a system may have different response times.

The DIU default response time should be minimized. DIU response times must be configurable to be greater than the default.

3.3 DIU SIMULATOR RECOVERY TIME

The time required by an addressed DIU simulator CPU to complete action on a transaction before it becomes ready to receive another frame is called the DIU recovery time. During recovery time, the DIU simulator CPU is unable to receive another frame.

When more than one DIU simulator shares the same CPU, the interaction of the DIU response time with the AIPS IOS turnaround time must be considered when designing network chains and allocating DIU simulator addresses to specific CPUs.

3.4 DIU ERROR HANDLING

The following error handling descriptions assume that a transaction that contains an error has passed the HDLC interface chip's hardware address screening.

3.4.1 Software-Detected Errors

The following errors must be detected by DIU simulator software:

a. Sum check (SC): A sum check error is logged by the DIU and no response frame is sent.

b. Frame ID: A frame ID error is logged by the DIU and no response frame is sent.

c. Data count: A data count error is logged by the DIU if the incorrect number of bytes is received and no response frame is sent.

d. Residual bits: A residual bit error is logged by the DIU and no response frame is sent. (Node frames have 3 residual bits; DIU frames have 5).

e. Long frame: If the received frame contains more than a specified maximum number of bytes, a long frame error is logged by the DIU and no response frame is sent.

3.4.2 Hardware Detected Errors

HDLC interface chips possess error detection logic. The following errors must be detected by reading the appropriate register of the HDLC interface chip:

a. Zero Residual Bits. All transactions in the small-scale system use non-zero residual bits. A transaction with 0 residual bits is in error. The occurrence of zero residual bits is logged by the DIU and no response frame is sent.

b. Data Overrun. Occurs when data is not removed from a DIU interface chip receive buffer fast enough to prevent overwriting data already in the buffer. Data overrun is logged by the DIU and no response frame is sent.

c. Frame check sequence (FCS) - used to detect transmission errors. It is computed by the HDLC interface device in the DIU as a frame is received. FCS error is logged by the DIU and no response frame is sent.

d. Short frame - occurs when a closing flag is detected before all expected HDLC fields have been received. Short frame is logged by the DIU and no action is taken on FTP commands and requests.

Abort detect and Idle detect are not used in the AIPS HDLC protocol. Both of these "errors" occur during normal network operation. They must not be logged by the DIU and must not affect simulator operation.

3.5 MODES OF OPERATION

3.5.1 One-Port Single DIU Simulator

This mode simulates the operation of a typical DIU. Hardware address screening should block frames not addressed to the DIU. Multiple transactions in different chains for which different DIU responses are required must be identified by a frame ID that occupies the Control field of the FTP output frame. (See ref. 5.)

A separate physical interface to the I/O network must be provided for each simulated DIU.

3.5.2 One-Port Multiple DIU Simulator

The operation of this mode must be identical to previous mode with the exception that multiple DIUs must be accessible via a single physical interface to the I/O network.

3.5.3 Network Probe

As a network probe, the DIU simulator should monitor and record all network traffic occurring on both transmit and receive signal lines. Limited error checking should be performed on data, and address screening should not be used. No responses frames may be generated. Both FTP and node transactions must be recorded.

3.6 EXPERIMENT SYNCHRONIZATION

Intercomputer experiment synchronization is provided by the DIU simulator sync and AIPS FTP sync lines. The status of these lines is present on lines of the experiment bus within the simulator (see ref. 4).

Reference 3 describes the operation of the sync lines and the state transitions for both the DIU simulator and the AIPS FTP.

An FTP sync line is provided to synchronize the DIU simulator to the start of an experiment in the FTP. When the FTP sync line is in the STOP state, no experiment is in progress. When an experiment begins, the FTP sync line is placed in the RUN state.

The DIU simulator sync line must be used as a handshake and qualifier for use with the FTP. During initialization, the DIU simulator sync line must be left in the STOP state. When the DIU simulator has completed initialization, the line must be changed to the RUN state.

3.7 LOCAL EXPERIMENT TIME

Experiment time in the DIU simulator must begin with the transition of the FTP sync line from STOP to RUN. Each tick is equivalent to one fault tolerant clock (FTC) period of 4.125 μs. Experiment time must be maintained as a 24-bit value representing the number of FTC ticks since the start of an experiment.

3.8 FAILURE SIMULATION

No DIU failures will be used in the small-scale system testing.

3.9 DATA COLLECTION

Each DIU simulator must record internally generated timing data and information received from the I/O network for later analysis. The minimum data recorded shall include:

a. Number of bytes received, not counting flags or FCS bytes (1 byte).

b. Time frame received relative to experiment start synchronization (3 bytes).

c. DIU address - the value of the address in the HDLC address field (1 byte).

d. Frame ID - the value of the frame ID field as defined in reference 5 (1 byte).

e. Sequential frame count - the value received in the SFC fields of reference 5 (2 bytes).

f. Error status of both hardware and software (2 bytes).

In lieu of the DIU address, frame ID, and sequential frame count, all data received by the DIU simulator may be recorded.

REFERENCES

1. NASA Contractor Report, Design of an Integrated Airframe/ Propulsion Control System Architecture, NASA contract NAS1-18099, May 1989.

2. AIPS I/O Network Interface Requirements.

3. Small Scale System Experiment Start Synchronization.

4. VMEbus Experiment Bus Definition.

5. Small-Scale System I/O Network/DIU Configuration.

APPENDIX D: SMALL-SCALE SYSTEM I/O NETWORK FAULT INSERTION REQUIREMENTS

1.0 INTRODUCTION

The I/O network fault inserter is used to cause failures in the small-scale system I/O network for the purpose of studying the fault recovery behavior of the AIPS system. This specification defines the requirements that govern the design of the fault inserter.

2.0 I/O NETWORK INTERFACE

The I/O network fault inserter must be compatible with the AIPS I/O network as described in reference 2. Two connectors must be provided to allow failing I/O network links and nodes: an in channel connector that is electrically closest to the FTP and an out channel connector that is farthest from the FTP.

3.0 I/O NETWORK CHANNEL FAILURE MODES

Each fault insertion channel must support the following faults on each connector.

3.1 NORMAL

Signals on the in and out connectors are passed through the fault inserter with no modification other than a signal delay caused by the fault insertion circuitry. This delay must be no greater than 100 ns.

3.2 PASSIVE FAILURES

When a passive failure is inserted, the output lines of the failed connector must be set to logic 0. This failure will not actively

propagate to other devices connected to the AIPS I/O network and will only be detected by the lack of a response from a device that uses the failed link.

3.3 ACTIVE FAILURES

When an active failure is inserted, the output lines of the failed connector must be set to logic 1. The failure may be immediately detected by the FTP if it is inbound; if it is outbound it may not be detected until an addressed device fails to respond. An active failure can block traffic to devices that do not use the failed link to connect into the network.

4.0 FAILURE MODE CONTROL

All failure channels must be controllable by the simulation host VMEbus computer. Scheduling of failures and their location must be easily configurable by the experimenter.

4.1 FAULT INSERTION CHANNELS

Each physical I/O network fault insertion channel must be assigned a unique physical address. Enough channels must be available to fail from one to five links simultaneously.

Each physical fault channel must be capable of being assigned to a logical channel that will be the actual channel failed. Multiple physical channels may be assigned to a single logical channel.

4.2 MAPPING PHYSICAL TO LOGICAL CHANNELS

Physical channels may be mapped to logical channels in any manner desired as long as configuration of the fault inserter is easily accomplished by the experimenter.

4.3 INDIVIDUAL CONNECTOR FAULT CONTROL

Fault channel in and out connector modes must be individually programmable. It must be possible to operate the two connectors in similar and/or dissimilar modes.

Small-scale system testing requirements do not require this capability at this time.

4.4 FAULT SCHEDULING

Fault occurrence must be specified in microseconds after the occurrence of FRUN (see ref. 3). The fault delay timer must operate from the common simulation computer FTC reference timebase. User input in microseconds must be converted to the required number of FTC ticks.

Accuracy of fault insertion timing shall be +/- 100 μs minimum with respect to the transition to FRUN.

REFERENCES

1. NASA Contractor Report, Design of an Integrated Airframe/Propulsion Control System Architecture, NASA contract NAS1-18099, May 1989.

2. AIPS I/O Network Interface Requirements.

3. Small Scale System Experiment Synchronization.

APPENDIX E: SMALL-SCALE SYSTEM NETWORK/DIU CONFIGURATION

The configuration of the small-scale system I/O network is based on the flight control computer reference configuration with the exception that only two network interfaces are used.

Note: All addresses and IDs are in hexadecimal.

1.0 HDLC ADDRESS AND FRAME ID ASSIGNMENTS

HDLC address allocation:

Root nodes:	0 - F
Network 1:	odd
Network 2:	even
DIU nodes:	10 - 7F
Network 1:	10 - 1F
Network 2:	20 - 2F
DIU:	80 - FE
Network 1:	80 - 8F
Network 2:	90 - 9F

Command frame IDs:

100 Hz command:	0 - 1F
Network 1:	00 - 0F
Network 2:	10 - 1F
50 Hz command:	40 - 5F
Network 1:	40 - 4F
Network 2:	50 - 5F
25 Hz command:	80 - 9F
Network 1:	80 - 8F
Network 2:	90 - 9F

Response frame IDs:

100 Hz response:	20 - 3F
Network 1:	20 - 2F
Network 2:	30 - 3F
50 Hz response:	60 - 7F
Network 1:	60 - 6F
Network 2:	70 - 7F
25 Hz response:	A0 - AF
Network 1:	A0 - AF
Network 2:	B0 - BF

Network 1: fully simulated

Node name	Node address
FC1	1
FC3	3
S1	10
S2	11
CP1	12
CP2	13
CDL	14
CDR	15
N	16
LER	17
OFL	18
OFR	19
IFL	1A
IFR	1B
TEL	1C
TER	1D
RL	1E
RR	1F

DIU name	DIU address	Command HDLC frame ID			Response HDLC frame ID		
		100 Hz	50 Hz	25 Hz	100 Hz	50 Hz	25 Hz
S1	80	0	40	80	20	60	A0
S2	81	1	41		21	61	
CP1	82		42	82		62	A2
CP2	83		43			63	
CDL	84		44			64	
CDR	85		45			65	
N	86		46			66	
LER	87		47			67	
OFL	88	8			28		
OFR	89	9			29		
IFL	8A	A			2A		
IFR	8B	B			2B		
TEL	8C	C			2C		
TER	8D	D			2D		
RL	8E		4E			6E	
RR	8F		4F			6F	

Network 2: partially simulated

Node name	Node address
FC2	2
FC4	4

DIU name	DIU address	Command HDLC frame ID			Response HDLC frame ID		
		100 Hz	50 Hz	25 Hz	100 Hz	50 Hz	25 Hz
S3	90	10	50	90	30	70	B0
S4	91	11	51		31	71	
CP3	92		52	92		72	B2
CP4	93		53			73	
CDL	94		54			74	
CDR	95		55			75	
N	96		56			76	
LEL	97		57			77	
OFL	98	18			38		
OFR	99	19			39		
IFL	9A	1A			3A		
IFR	9B	1B			3B		
TEL	9C	1C			3C		
TER	9D	1D			3D		
RL	9E		5E			7E	
RR	9F		5F			7F	

2.0 NETWORK TOPOLOGY

The network connections specified are derived from the flight control computer reference configuration with the exception that only two root nodes are used along with two FTP network interfaces.

Node ports are designated: node-address - node-port-number.

Ports are generally assigned as follows for a node:

-0 For root node, this is the inboard port that connects to the network interface in the FTP.

For DIU nodes, this is the inboard port that results from default network growth.

-1 Ports for network interconnectivity

-2

-3

-4 Normal connection port for DIU

Network 1 connections:

Root node	Port	Node name	Port	Cable
FC1	1-0	GPC A NI	IOS-1	------
	1-1	S2	11-0	------
	1-2	FC3	3-1	------
	1-3	S1	10-0	------
	1-4			
FC3	3-0	GPC B NI	IOS-1	------
	3-1	FC1	1-2	------
	3-2	RR	1F-2	------
	3-3	RL	1E-0	------
	3-4			

DIU node	Port	Node name	Port	Cable
S1	10-0	FC1	1-3	------
	10-1	OFL	18-2	------
	10-2	CP1	12-2	------
	10-3			
	10-4	DIU-S1		------
S2	11-0	FC1	1-1	------
	11-1	CP2	13-0	------
	11-2	OFR	19-0	------
	11-3			
	11-4	DIU-S2		------

DIU node	Port	Node name	Port	Cable
CP1	12-0	CDL	14-2	------
	12-1	CP2	13-2	------
	12-2	S1	10-2	------
	12-3			
	12-4	DIU-CP1		------
CP2	13-0	S2	11-1	------
	13-1	CDR	15-1	------
	13-2	CP1	12-1	------
	13-3			
	13-4	DIU-CP2		------
CDL	14-0	IFR	1B-1	------
	14-1	N	16-2	------
	14-2	CP1	12-0	------
	14-3			
	14-4	DIU-CDL		------
CDR	15-0			
	15-1	CP2	13-1	------
	15-2	LER	17-0	------
	15-3	IFL	1A-1	------
	15-4	DIU-CDR		------
N	16-0	RL	1E-2	------
	16-1	OFL	18-1	------
	16-2	CDL	14-1	------
	16-3			
	16-4	DIU-N		------
LER	17-0	CDR	15-2	------
	17-1	OFR	19-1	------
	17-2	RR	1F-0	------
	17-3			
	17-4	DIU-LER		------

DIU node	Port	Node name	Port	Cable
OFL	18-0	IFL	1A-0	------
	18-1	N	16-1	------
	18-2	S1	10-1	------
	18-3			
	18-4	DIU-OFL		------
OFR	19-0	S2	11-2	------
	19-1	LER	17-1	------
	19-2	IFR	1B-0	------
	19-3			
	19-4	DIU-OFR		------
IFL	1A-0	OFL	18-0	------
	1A-1	CDR	15-3	------
	1A-2	TEL	1C-2	------
	1A-3			
	1A-4	DIU-IFL		------
IFR	1B-0	OFR	19-2	------
	1B-1	CDL	14-0	------
	1B-2	TER	1D-0	------
	1B-3			
	1B-4	DIU-IFR		------
TEL	1C-0	TER	1D-2	------
	1C-1	RL	1E-1	------
	1C-2	IFL	1A-2	------
	1C-3			
	1C-4	DIU-TEL		------

DIU node	Port	Node name	Port	Cable
TER	1D-0	IFR	1B-2	------
	1D-1	RR	1F-1	------
	1D-2	TEL	1C-0	------
	1D-3			
	1D-4	DIU-TER		------
RL	1E-0	FC3	3-3	------
	1E-1	TEL	1C-1	------
	1E-2	N	16-0	------
	1E-3			
	1E-4	DIU-RL		------
RR	1F-0	LER	17-2	------
	1F-1	TER	1D-1	------
	1F-2	FC3	3-2	------
	1F-3			
	1F-4	DIU-RR		------

Network 2 connections:

Root node	Port	Node name	Port	Cable
FC2	2-0	GPC B	IOS-2	------
	2-1			
	2-2			
	2-3	FC4	4-3	------
	2-4	DIU-FC2		------
FC4	4-0	GPC C	IOS-2	------
	4-1			
	4-2			
	4-3	FC2	2-3	------
	4-4	DIU-FC4		------

3.0 DIU SIMULATOR BOARD ASSIGNMENTS

Network 1:

DIU board/-port	-1	-2	-3	-4	-5	-6	-7	-8
N1DIU1	S1	OFL	IFL	TEL	CP1	CDL	N	RL
N1DIU2	S2	OFR	IFR	TER	CP2	CDR	LER	RR

Network 2:

DIU board/-port	-1	-2	-3	-4	-5	-6	-7	-8
N2DIU1	S3	OFL	IFL	TEL	CP3	CDL	N	RL
N2DIU2	S4	OFR	IFR	TER	CP4	CDR	LEL	RR

4.0 I/O CHAIN DEFINITIONS

The I/O chains for the small-scale system are described below for the test configuration using all input transactions. Each frame named below corresponds to the previously defined HDLC frames.

Network 1 chains:

100 Hz:

S1, S2, OFL, OFR, IFL, IFR, TEL, TER

50 Hz:

S1, S2, CP1, CP2, CDL, CDR, N, LER, RL, RR

25 Hz:

S1, CP1

Network 2 chains:

100 Hz:

S3, S4, OFL, OFR, IFL, IFR, TEL, TER

50 Hz:

S3, S4, CP3, CP4, CDL, CDR, N, LEL, RL, RR

25 Hz:

S3, CP3

APPENDIX F: SMALL-SCALE SYSTEM I/O NETWORK TRANSACTIONS

The small-scale system is composed of 18 nodes with 16 DIU simulators on network 1 and 2 nodes and 16 DIU simulators on network 2.

The command format to a DIU from the FTP is:

DIU address,
Encoded DIU address (complemented DIU address),
HDLC frame ID,
Sequential frame count (high byte),
Sequential frame count (low byte),
Padding characters (as required),
Sum check

DIU addresses are unique in networks 1 and 2.

HDLC frame IDs are unique in networks 1 and 2.

Padding characters may be changed at a later date.

The response format from a DIU to the FTP is:

DIU address,
Sum check,
HDLC frame ID,
Sequential frame count (high byte),
Sequential frame count (low byte),
Padding characters (as required),
Special pad character

Notes:

a. The DIU address in the response frame is the address of the DIU generating the response frame.

b. The special pad character (spec-pad) is required by a problem with the design of the SCN68562 HDLC interface chip. The placement of the sum check immediately following the DIU address in the response frame does not violate AIPS protocol because no encoded address is used for response frames and the definition for sum check only requires the sum of all bytes sent to be zero.

Abbreviations used in the message definitions are:

sfc-lo = sequential frame count low byte
sfc-hi = sequential frame count high byte
sum-chk = sum check
spec-pad = special pad character

Note: All numbers in HDLC frame formats are in hexadecimal.

Network 1 DIU command HDLC frame format:

100 Hz

S1	80, 7F, 00, sfc-hi, sfc-lo, sum-chk
S2	81, 7E, 01, sfc-hi, sfc-lo, sum-chk
OFL	88, 77, 08, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
OFR	89, 76, 09, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
IFL	8A, 75, 0A, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
IFR	8B, 74, 0B, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
TEL	8C, 73, 0C, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
TER	8D, 72, 0D, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk

50 Hz

S1	80, 7F, 40, sfc-hi, sfc-lo, sum-chk
S2	81, 7E, 41, sfc-hi, sfc-lo, sum-chk
CP1	82, 7D, 42, sfc-hi, sfc-lo, sum-chk
CP2	83, 7C, 43, sfc-hi, sfc-lo, sum-chk
CDL	84, 7B, 44, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
CDR	85, 7A, 45, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
N	86, 79, 46, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
LER	87, 78, 47, sfc-hi, sfc-lo, 10, 20, 30, 40, 50, 60, 70, 80, 90, A0, B0, C0, sum-chk
RL	8E, 71, 4E, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
RR	8F, 70, 4F, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk

25 Hz

S1	80, 7F, 80, sfc-hi, sfc-lo, sum-chk
CP1	82, 7D, 82, sfc-hi, sfc-lo, sum-chk

Network 1 DIU response HDLC frame format:

100 Hz

S1	80, sum-chk, 20, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8 9, A, B, C, spec-pad
S2	81, sum-chk, 21, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, spec-pad
OFL	88, sum-chk, 28, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
OFR	89, sum-chk, 29, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
IFL	8A, sum-chk, 2A, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, spec-pad
IFR	8B, sum-chk, 2B, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, spec-pad
TEL	8C, sum-chk, 2C, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
TER	8D, sum-chk, 2D, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, spec-pad

50 Hz

S1	80, sum-chk, 60, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
S2	81, sum-chk, 61, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
CP1	82, sum-chk, 62, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
CP2	83, sum-chk, 63, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
CDL	84, sum-chk, 64, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
CDR	85, sum-chk, 65, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
N	86, sum-chk, 66, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
LER	87, sum-chk, 67, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, spec-pad
RL	8E, sum-chk, 6E, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
RR	8F, sum-chk, 6F, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad

25 Hz

S1	80, sum-chk, A0, sfc-hi, sfc-lo, 1, 2, spec-pad
CP1	82, sum-chk, A2, sfc-hi, sfc-lo, 1, 2, spec-pad

Network 2 DIU command HDLC frame format:

100 Hz

S3	90, 6F, 10, sfc-hi, sfc-lo, sum-chk
S4	91, 6E, 11, sfc-hi, sfc-lo, sum-chk
OFL	98, 67, 18, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
OFR	99, 66, 19, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
IFL	9A, 65, 1A, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
IFR	9B, 64, 1B, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
TEL	9C, 63, 1C, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
TER	9D, 62, 1D, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk

50 Hz

S3	90, 6F, 50, sfc-hi, sfc-lo, sum-chk
S4	91, 6E, 51, sfc-hi, sfc-lo, sum-chk
CP3	92, 6D, 52, sfc-hi, sfc-lo, sum-chk
CP4	93, 6C, 53, sfc-hi, sfc-lo, sum-chk
CDL	94, 6B, 54, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
CDR	95, 6A, 55, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
N	96, 69, 56, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
LEL	97, 68, 57, sfc-hi, sfc-lo, 10, 20, 30, 40, 50, 60, 70, 80, 90, A0, B0, C0, sum-chk
RL	9E, 61, 5E, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk
RR	9F, 60, 5F, sfc-hi, sfc-lo, 10, 20, 30, 40, sum-chk

25 Hz

S3	90, 6F, 90, sfc-hi, sfc-lo, sum-chk
CP3	92, 6D, 92, sfc-hi, sfc-lo, sum-chk

Network 2 DIU response HDLC frame format:

100 Hz

S1	90, sum-chk, 30, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, spec-pad
S2	91, sum-chk, 31, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, spec-pad
OFL	98, sum-chk, 38, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
OFR	99, sum-chk, 39, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
IFL	9A, sum-chk, 3A, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, spec-pad
IFR	9B, sum-chk, 3B, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, spec-pad
TEL	9C, sum-chk, 3C, trans-id, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
TER	9D, sum-chk, 3D, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, spec-pad

50 Hz

S1	90, sum-chk, 70, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
S2	91, sum-chk, 71, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
CP1	92, sum-chk, 72, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
CP2	93, sum-chk, 73, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, spec-pad
CDL	94, sum-chk, 74, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
CDR	95, sum-chk, 75, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
N	96, sum-chk, 76, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
LEL	97, sum-chk, 77, sfc-hi, sfc-lo, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, spec-pad
RL	9E, sum-chk, 7E, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad
RR	9F, sum-chk, 7F, sfc-hi, sfc-lo, 1, 2, 3, 4, spec-pad

25 Hz

S1	90, sum-chk, B0, sfc-hi, sfc-lo, 1, 2, spec-pad
CP1	92, sum-chk, B2, sfc-hi, sfc-lo, 1, 2, spec-pad

APPENDIX G: EXPERIMENT BUS DESCRIPTION

1.0 BACKGROUND

The simulation computer requires various handshaking lines, clock signals, and so forth. These signals must be routed to the ISIO-2 DIU simulator cards and the wire wrap card.

The OPIO-1 interface card does not use all of the VMEbus P2 connector pins. The remaining pins are used to implement the experiment bus.

2.0 EXPERIMENT BUS SIGNALS

FTP Start Sync [F_SYNC]
TTL level signal: low = FTP stop high = FTP run

VME Start Sync [V_SYNC]
TTL level signal: low = VMEbus stop high = VMEbus run

External Event [X_EVENT]
TTL level signal: level is selectable under software control

Reference Fault Tolerant Clock [R_FTC]
TTL level signal: waveform, 4.125 µs period

Fault Strobe [FSTB]
TTL level signal: high = FDn data invalid low = FDn data valid

Fault Ack [FACK]
TTL level signal: high = waiting for data low = data accepted

Fault Data [FD0..FD7]
TTL level signals: high = logic 1 low = logic 0

3.0 EXPERIMENT BUS PIN ASSIGNMENT

P2-	Name	Description	Logic	Source
1c	[F_SYNC]	FTP Start Sync	Run / !Stop	WW
2c	[V_SYNC]	VME Start Sync	Run / !Stop	OPIO-1
3c	[X_EVENT]	External Event	soft. select	OPIO-1
17c	[R_FTC]	Reference FTC	n/a	WW
18c	[FACK]	Fault Ack	active low	OPIO-1
19c	[FSTB]	Fault Strobe	active low	OPIO-1
20c	[FD0]	Fault data bit 0	non-inverted	OPIO-1
20a	[FD1]	Fault data bit 1	non-inverted	OPIO-1
21c	[FD2]	Fault data bit 2	non-inverted	OPIO-1
21a	[FD3]	Fault data bit 3	non-inverted	OPIO-1
22c	[FD4]	Fault data bit 4	non-inverted	OPIO-1
22a	[FD5]	Fault data bit 5	non-inverted	OPIO-1
23c	[FD6]	Fault data bit 6	non-inverted	OPIO-1
23a	[FD7]	Fault data bit 7	non-inverted	OPIO-1

APPENDIX H: VULTURE PROGRAM DETAILS

1.0 SUMMARY

The VME Ultimate User Environment (VULTURE) is a user-friendly set of programs that provide the user with the tools required to quickly and effectively communicate with a VMEbus system. The purpose of VULTURE is to remove users from the unfriendly confines of the VMEbus test system and place them in an environment with which they feel comfortable.

Three basic capabilities are provided with the VULTURE system. The first feature allows the user to establish or reestablish a communication link between the MicroVAX and the VMEbus system. The second capability allows transfer of byte-oriented data between the two systems. The third feature provides the user with a set of commands to manage and manipulate files on the VMEbus system.

2.0 THIRD PARTY SOFTWARE

VULTURE uses three software silicon components from Ready Systems. The Versatile Real-Time Executive (VRTX) provides a real-time, multitasking operating system for embedded microprocessor applications. The I/O and File Executive (IFX) provides input/output and file management facilities. The C Runtime Library (RTL) provides a C programming language interface to VRTX and IFX. The VULTURE software relies upon these Ready Systems components to assist in managing the VMEbus system and executing user commands.

3.0 FUNCTIONAL DESCRIPTION

Embedded systems often have a very unfriendly user environment. VULTURE's function is to provide a well-known and robust environment for the user. By using digital command language (DCL), VULTURE is able to provide the user with the desired result.

Conceptually, VULTURE provides a subset of the VAX/VMS operating system on a VMEbus platform.

4.0 COMMUNICATIONS PROTOCOL

This section gives an overview of the three basic protocols used by VULTURE. All communications between the MicroVAX and VMEbus system originate in the MicroVAX. No unsolicited commands are recognized from the VMEbus system.

All communications between the MicroVAX and the VMEbus system use VULTURE command and response protocol via two high-speed 16-bit parallel interfaces, one for input and one for output. Each command or response is sent as a 64-byte header record shown in figure H-1. Figure H-2 a list of each VULTURE command and the header fields they uses. Each header record is followed by a long word sum check that is used to ensure error free communications.

The three classes of protocols for communications between the MicroVAX and VMEbus system are shown in figure H-3. Class 1 is used to initialize or reinitialize the interface between the two systems. Class 2 is used for most system management commands. Class 3 is used when data transfer between systems is required.

4.1 CLASS 1 PROTOCOL

Class 1 protocol operates as follows:

a. The MicroVAX sets the interface function lines to a known state, requesting a function line response from the VMEbus system.

b. If a function line echo has been received within a timeout period, the MicroVAX attempts to send a VRESET header to the VMEbus system, followed by a long word sum check.

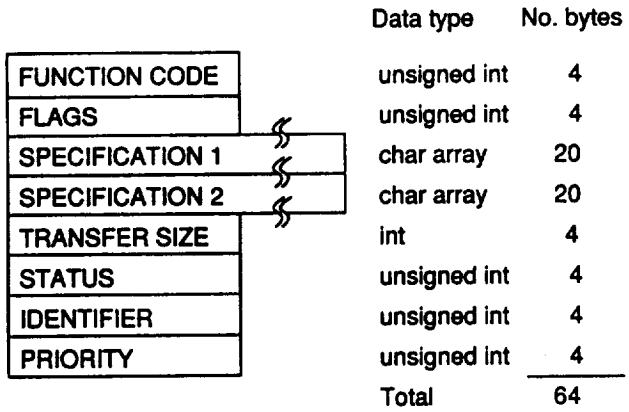
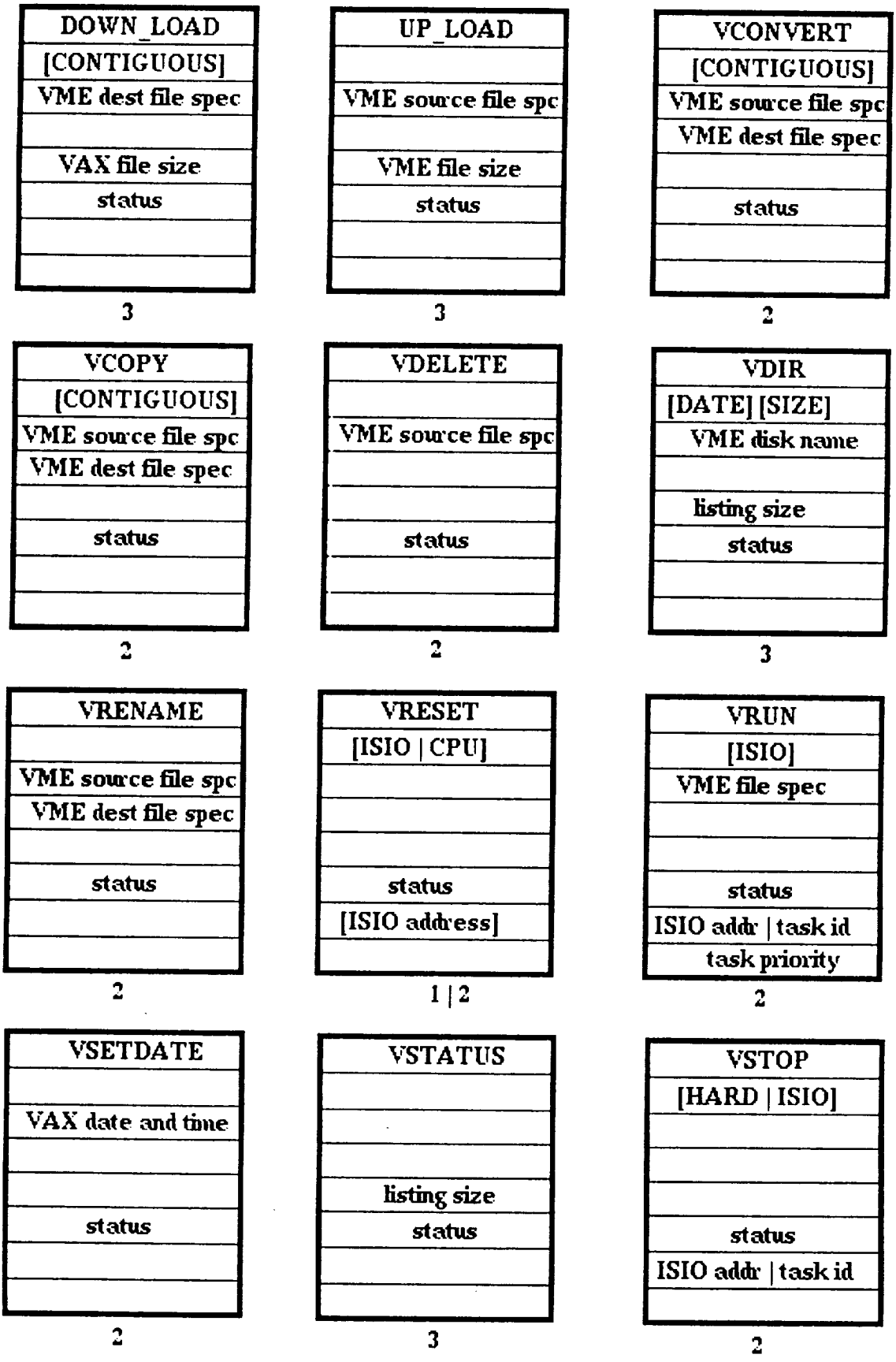


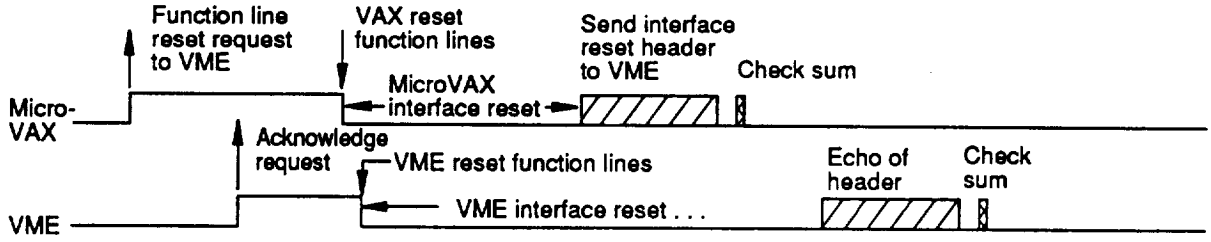
Figure H-1. VULTURE Command/Response Header Format



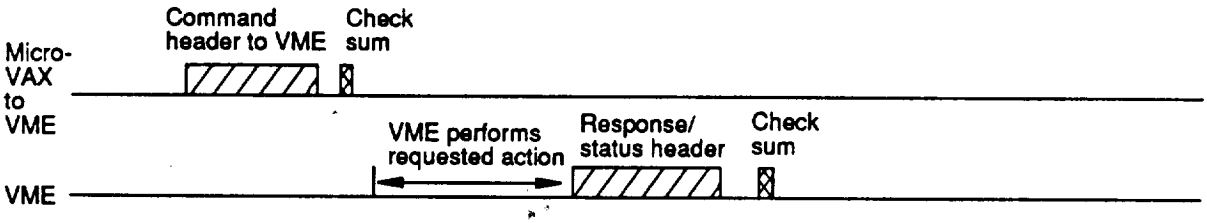
Note: Protocol class is listed below each header; header format is shown in figure H-1

Figure H-2. VULTURE Command Header Format

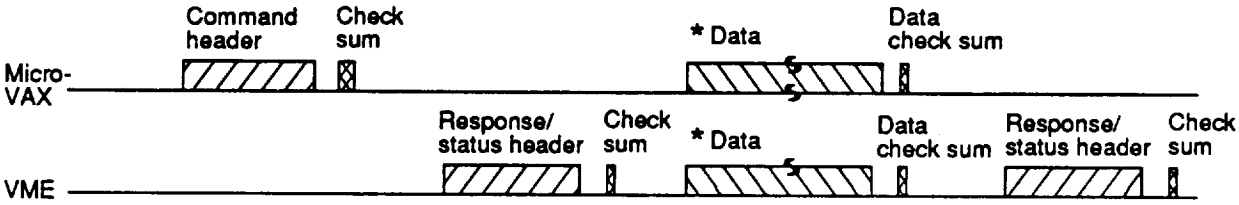
Class 1: Interface Reset



Class 2: Simple Action Request



Class 3: Complex Action Request/Data Transfer



* Data are only sent in one direction at a time. The originator will depend on the command.

Legend:

Header

Check sum

Data

Function line

Figure H-3. VULTURE Communication Protocols

c. The VMEbus system sends a response header followed by a check sum. If the header and sum check are correct, the interface has been successfully reset.

Any timeouts or protocol errors result in an error message appearing on the MicroVAX user's console.

4.2 CLASS 2 PROTOCOL

Class 2 protocol operates as follows:

a. The MicroVAX sends a header that identifies the function to be executed followed by a sum check.

b. The VMEbus system executes the requested function and sends a response header followed by a sum check. The status field of the header is used to indicate errors which occurred during function execution.

4.3 CLASS 3 PROTOCOL

Class 3 protocol operates as follows:

a. The MicroVAX sends a header that identifies the function to be executed followed by a sum check.

b. The VMEbus system performs preliminary actions necessary to execute the requested function and sends a response header followed by a sum check. The status field of the header is used to indicate errors that occurred during function execution.

c. The data are transferred. The number of bytes sent must match the value in the appropriate command or response transfer count field. The data transfer is followed by a check sum.

d. The VMEbus system sends a response header followed by a check sum. The status field of the header is used to indicate errors that occurred during function execution.

5.0 ERROR HANDLING

Any errors that might occur during the execution of a command are displayed on the user's MicroVAX terminal. The error messages are generated by the VMS operating system based on the error code placed in the STATUS field of the header returned by the VMEbus system. If an unexpected error occurs on the VMEbus system, the VMEbus error code (generated by VRTX or IFX) is returned to the MicroVAX to be displayed along with a message telling the user that an unanticipated error has occurred. The user can then reference the Ready Systems user's manuals to determine the cause of the error.

6.0 BYTE SWAPPING

Because of the differences in how the MicroVAX and VMEbus systems internally represent data, a limitation must be imposed on what kind of data can be transferred. Figure H-4 shows how the VAX and Motorola chips represent bytes, words, and longwords internally. As can be seen from this figure, only byte-oriented data are stored in the same format. Figure H-5 shows what happens to data after they have been sent from the MicroVAX to the VMEbus system via the 16-bit parallel interface. Byte data are again the only type of data that are consistent across the two architectures.

Because most executable files can be created in an ASCII S record format, and data files are usually in ASCII format, it was decided that all data transfers would be byte oriented. If users want to use a different data size, they should write a VMEbus-based program that unscrambles data based on the specifications in figure H-5.

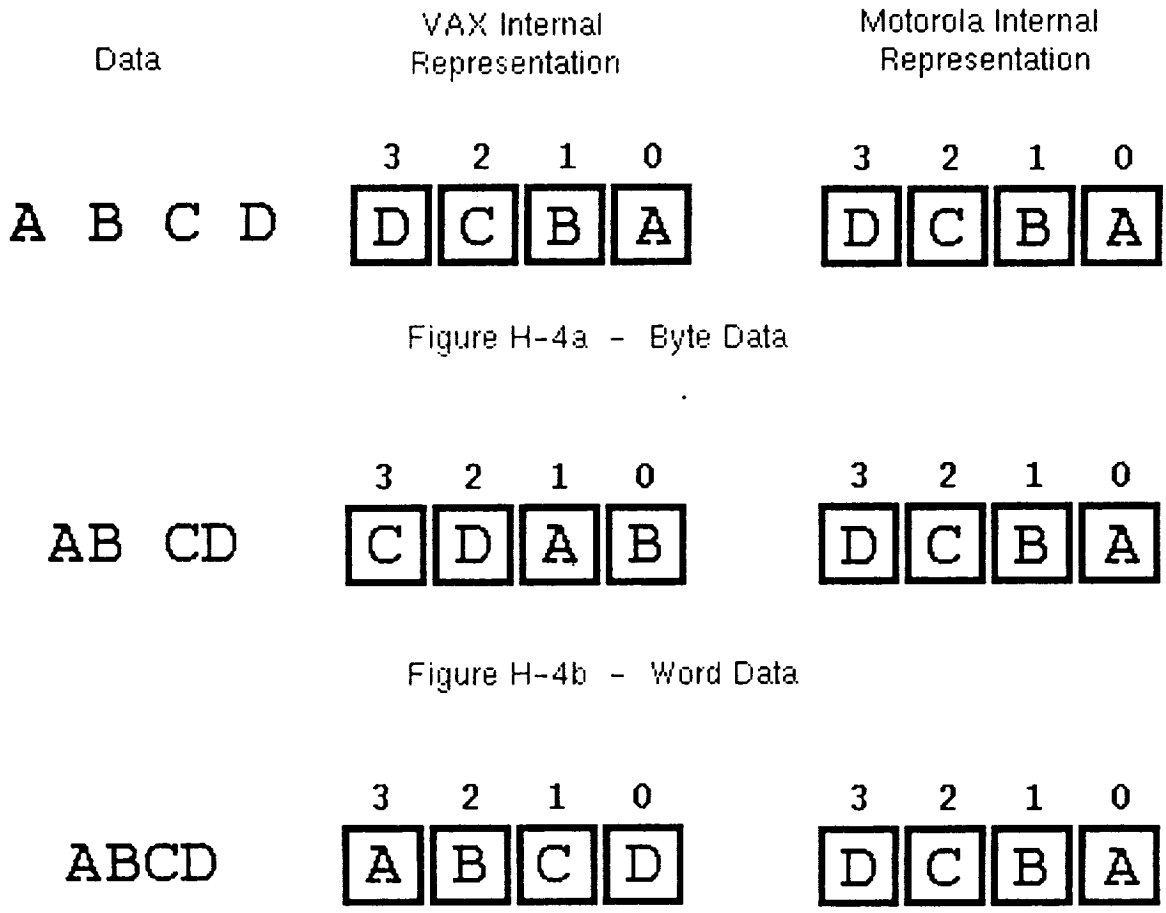


Figure H-4. Internal Data Representation

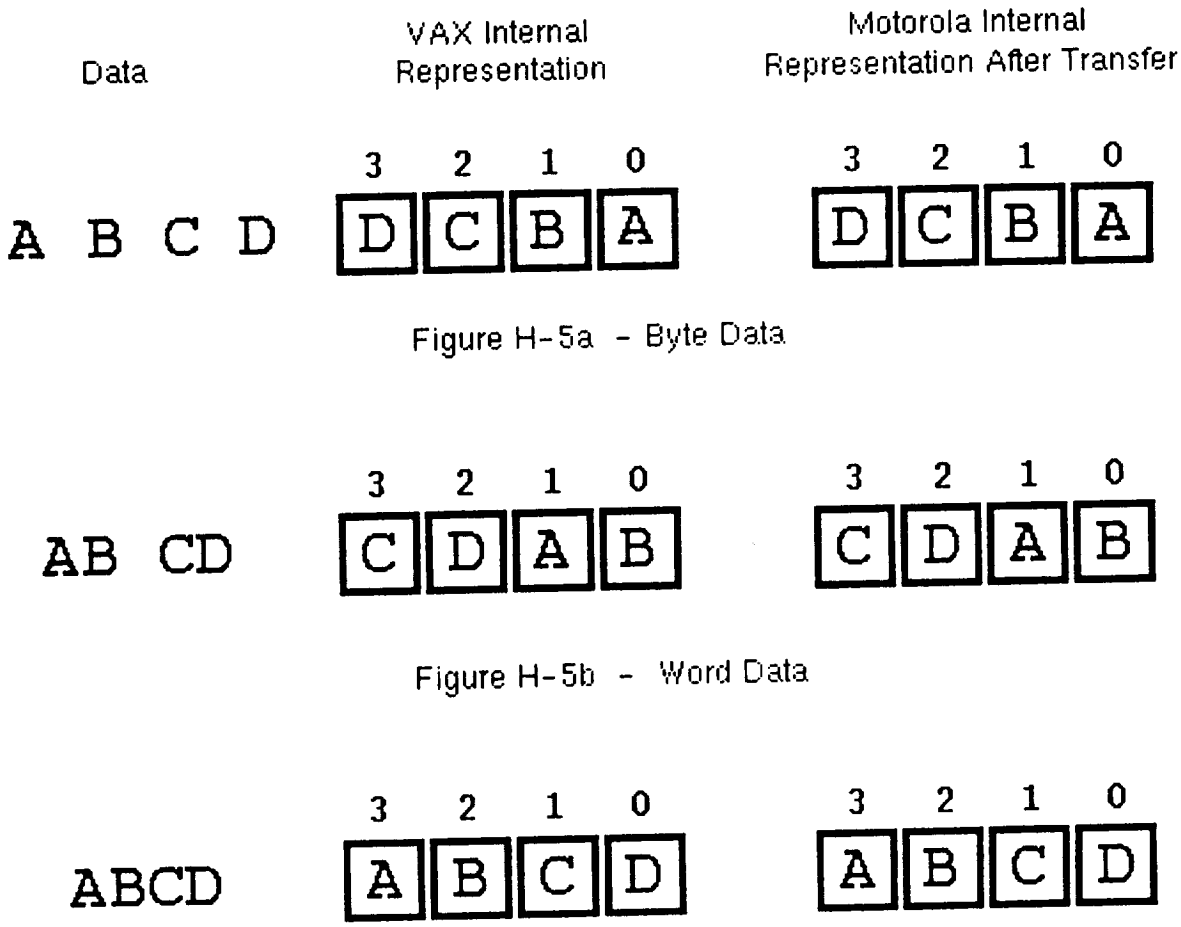


Figure H-5. Internal Data Representation After Transfer

7.0 MEMORY LAYOUT

Figure H-6 shows how the RAM on the VMEbus is arranged. This memory map is reconfigurable by the user. The board support software routines define how RAM is to be configured. See section 11 for more information on modifying the board support package.

8.0 EXAMPLE USER SESSION

Before VULTURE can be used, the VMEbus system must be running. VULTURE becomes active as soon as the power to the Force CPU-29 card is supplied. The user can tell that the communications software is active when the VULTURE logo appears on the operator console connected to the CPU-29 card.

The date and time of the VMEbus system are set by issuing the VSETDATE command. This is the usually the first command issued because it sets the VMEbus system time to be the same as the MicroVAX time.

Next, all code for tasks is downloaded in S-record format using the DOWN-LOAD command. The files are then translated into executable files using the VCONVERT command. If any data files are needed at this point, they too are downloaded to the VMEbus system.

The tasks can now be started by issuing the VRUN command. Once running, the tasks can be monitored using the VSTATUS command. Once the tasks are completed, the UP-LOAD command is used to retrieve any files that might have been created during the execution of the tasks.

9.0 ROTARY DIAL SETTINGS

The VEMbus system has three modes of operation. The mode of operation is specified by setting the rotary dials on the front of the CPU-29 card.

0000 0000	Exception Vector Table
0000 0400	Malloc Table
0000 04F8	xtaskws (longword used by C RTL)
0000 04FC	xncpown (longword used by C RTL)
0000 0500	Work Space
0001 A000	Dynamic Memory
0003 A000	SRAM Disk (1792 512 byte sectors)
0010 0000	DRAM Disk (29952 512 byte sectors)
00FC 0000	Maximum Local RAM Address
FC00 0000	DIU Simulator #1 Controller and RAM
FC02 0000	DIU Simulator #2 Controller and RAM
FC04 0000	DIU Simulator #3 Controller and RAM
FC06 0000	DIU Simulator #4 Controller and RAM
FC08 0000	DIU Simulator #5 Controller and RAM
FC0A 0000	DIU Simulator #6 Controller and RAM
FC0C 0000	DIU Simulator #7 Controller and RAM
FCFF 0000	DMA Controller
FF00 0000	VMEPROM
FF02 0000	RTSCOPE
FF03 0000	IFX
FF04 0000	VRTX
FF04 2000	C Runtime Library
FF04 6000	Board Support Package Without RTSCOPE
FF04 A000	Board Support Package With RTSCOPE
FF80 0000	Local I/O Devices

Figure H-6. VULTURE VMEbus Memory Use

To have VULTURE run as a standalone system, rotary dial 1 should be pointing to 3. If a debugging environment is needed on the VMEbus system, rotary dial 1 should be set to B. In a debugging situation, VULTURE runs under RTSCOPE, the Ready Systems real-time debugger. The third possible mode of operation is VMEPROM, the native operating system on the Force CPU-29 card. To bring up VMEPROM, rotary dial 1 should be pointing to F when the reset switch is toggled on the front of the CPU-29 card.

10.0 WRITING AND COMPILING PROGRAMS

All code that is intended to be run on the CPU-29 card, whether written in C or assembly language, must be written as relocatable code with a base address of zero. All files that are destined to be downloaded to and run on the VMEbus system should use the following qualifiers when compiling:

```
c68k
        /noinit
        /norom
        cpu=68881q
        /long
        /code=pc
        /data=pc
        /opt=all
        /define=IFX
        /stringsintext

a68k
        /nolist
        /flags="case,brl,pcr,e,p=68020"
```

The interface libraries for IFX, VRTX, and the C runtime library must be loaded along with the object code for the task. In addition, the public symbol, C-RTL-BASE, must be specified as the base address of the C runtime library in RAM. All code should be generated with a base address of zero. The following is an example of a linker/loader file:

```
CHIP    68020
LIST    C,0,T,X
ORDER   0,9,14,15
PUBLIC  C-RTL-BASE=$FF042000
BASE    0
**** Put your own routines between here ****
LOAD    test
**************** and here ******************
LOAD    FC-LAB:[VME.LIBRARY.C-RTL]C-RTL-IL
LOAD    FC-LAB:[VME.LIBRARY.ASM-SRC]IFXIL.LIB
LOAD    FC-LAB:[VME.LIBRARY.ASM-SRC]VRTXIL.LIB
FORMAT  S
END
```

If the code is intended to be run on the ISIO cards, no I/O or dynamic memory allocation can be used and the C interface library should not be linked in. Instead, the actual C library should be linked in. The following is an example of a linker/loader file for files that will be run on the ISIO cards:

```
CHIP    68020
LIST    C,0,T,X
ORDER   0,9,14,15
BASE    0
**** Put your own routines between here ****
LOAD
test
**************** and here ******************
LOAD    FC-LAB:[VME.LIBRARY.C-RTL]C-RTL.LIB
```

```
LOAD    FC-LAB:[VME.LIBRARY.ASM-SRC]IFXIL.LIB
LOAD    FC-LAB:[VME.LIBRARY.ASM-SRC]VRTXIL.LIB
FORMAT  S
END
```

11.0 HOW TO BUILD VULTURE

Depending on user requirements, it may become necessary to reconfigure VULTURE. The actual VULTURE code should not be changed; instead, the board support package (BSP) should be modified. Once modified, the BSP can be recompiled and loaded with little effort. The challenge comes when trying to put the BSP in PROM.

To make VULTURE and the BSP programmable, the following qualifiers were used during compilation:

```
c68k
        /noinit
        /norom
        /cpu=68881q
        /long
        /code=pc
        /data=abs
        /opt=all
        /define=IFX
        /stringsintext

a68k
        /nolist
```

Command files assist in the building of all software that goes into PROM. All of these command files can be found in the software tape.

The first command file is BUILD-C-RTL.COM. This command file creates both the C runtime interface library and a programmable version of the C runtime library. After the command file completes, two files should be used. The first file, C-RTL-IL.OBJ, should be included in all C code that is designed to run under VULTURE. The second file, PROM-RTL.ABS, is the relocatable C runtime library, ready to be placed in PROM.

The second command file is BUILD-BSP.COM. This command file creates the basic board support software. Before executing this command file, the user should make sure that the old BSP.LIB has been deleted. In addition, at the time this document was prepared, the MRI C compiler generated bogus code when compiling IFX-SETUP.C. If an error occurs assembling the file the C compiler generates, simply change the BRA.S on the flagged line to a BRA in the IFX-SETUP.SRC file. If the BSP.LIB file still exists when the new library file is attempted, an error will occur and the new library will not be created. Two important files are created from this command file. The first file, BSP.OBJ, must be the first file included in the option file for the communications software discussed later. The second file, BSP.LIB, is a library file that should also be included in the options file for the communications software. The file, CPU29.INC, determines whether RTSCOPE will be linked with the BSP or not. To configure RTSCOPE, uncomment/comment out the corresponding lines in CPU29.INC, which can be found approximately 50 lines down from the top of the file.

The third command file is BUILD-COMM.COM. This file creates the relocatable code for the VULTURE communications software. This file compiles all the software and then links it together with the board support package. The file BSP.ABS contains the relocatable communications software once the command file has finished executing. The user should create two versions of BSP.ABS, one that has RTSCOPE linked in and one that does not.

Once a relocatable version of VULTURE exists (BSP.ABS) it can be burned into PROM. The following instructions show how to burn VULTURE into PROM. All VMEPROM commands are prefixed with a question mark (?), and all VMS commands are prefixed with a dollar sign ($).

1. Clear the memory on the VMEbus system.

```
? BF 6000 FBFFFC 0 L
```

2. Move the current relocatable copies of VMEPROM, RTSCOPE, IFX and VRTX from PROM to RAM.

```
? BM FF000000 FF041FFF A00000
```

3. Using the serial port LTA4:, download the relocatable C runtime library. The code will "land" at 10000.

```
? LO <2
$ COPY PROM-RTL.ABS LTA4:
```

4. Place the C runtime library right after the silicon software components that were downloaded from PROM in step 2.

```
? BM 10000 13FFF A42000
```

5. Using the serial port LTA4:, download the board support package that does not use RTSCOPE. The code will "land" at 80000.

```
? LO <2
$ COPY BSP-NO-RTSCOPE.ABS LTA4:
```

6. Place this version of the board support package after the C runtime library.

```
? BM 80000 83FFF A46000
```

7. Using the serial port LTA4:, download the board support package that does use RTSCOPE. The code will "land" at 80000.

```
? LO <2
$ COPY BSP-RTSCOPE.ABS LTA4:
```

8. Place this version of the board support package after the other version of the board support package.

```
? BM 80000 83FFF A4A000
```

9. Modify the board support package located at 80000 so that it does not zero DRAM.

```
? DI 80000
80000   MOVE.L #$88,A0
80006   LEA.L ($80010,PC),A1
8000A   NOP
8000C   MOVE.L A1,(A0)
8000E   TRAP #2
80010   MOVE.W #$3700,SR
80014   MOVE.L #$A30,A7
8001A   JSR ($8083C,PC)
8001E   JSR ($80224,PC)
80022   CLR.L ($191C4).L
80028   MOVEQ.L #0,D1
8002A   MOVEQ.L #$1,D2
 .
 .
 .

More (cr) ?              <--- enter a period, .

? AS 8001A
```

```
                                                  user input \
8001A                    : JSR ($8083C,PC)                 \
                         : NOP                <----------|
8001C                    : BTST.B #$BA,-(A0)               |
                         : NOP                <----------|
8001E                    :JSR ($80224,PC)                 /
                         : .                  <----------

? DI 80000

80000   MOVE.L #$88,A0
80006   LEA.L ($80010,PC),A1
8000A   NOP
8000C   MOVE.L A1,(A0)
8000E   TRAP #2
80010   MOVE.W #$3700,SR
80014   MOVE.L #$A30,A7
8001A   NOP
8001C   NOP
8001E   JSR ($80224,PC)
80022   CLR.L ($191C4).L
80028   MOVEQ.L #0,D1
8002A   MOVEQ.L #$1,D2
.
.
.

More (cr) ?              <--- enter a period, .
```

10. From a terminal server, connect to VME-AUX.

```
LOCAL> C VME-AUX
```

11. Run the modified board support package.

```
? GO 80000
```

12. Enter a GO at the RTSCOPE prompt on VME-AUX. This will cause the VULTURE software to begin execution.

```
RC> GO
```

13. Load the executable version of the S record split program to the SRAM disk of the VMEbus system. The file it resides in is called SPLIT.EXE.

```
$ DOWN-LOAD/CONTIGUOUS SPLIT.EXE SRAM:SPLIT.EXE
```

14. Execute the split program.

```
$ VRUN/TASK-ID=11 SRAM:SPLIT.EXE
```

15. On the VULTURE op console, enter the addresses from which the split of memory should occur. The addresses should correspond to the location on RAM where the programmable software resides.

```
Starting address: A00000    <------- user input
Ending address: A4DFFF    <----/
```

16. Wait for the split task to complete. The VSTATUS command can be used to monitor its activity.

17. Upload the files from the DRAM disk that contain the split of memory.

```
$ UP-LOAD SPLIT0.S SPLIT0.S
$ UP-LOAD SPLIT1.S SPLIT1.S
$ UP-LOAD SPLIT2.S SPLIT2.S
$ UP-LOAD SPLIT3.S SPLIT3.S
```

18. Initialize four INTEL 27010 PROMs by placing them under the ultraviolet light for 20 min.

19. Hook the DATA I/O prom burner up to the TXA7: port on the MicroVAX.

20. Allocate and set host to TXA7: on the MicroVAX.

```
$ ALLOCATE TXA7:
$ SET TERM/HOST TXA7:
$ SET HOST/DTE TXA7:
```

21. Change the NULL COUNT on the DATA I/O prom burner.

```
press SELECT
press D9
press START
press 01
press START
```

22. Put the prom burner in remote terminal mode. This should cause a menu to be displayed on the MicroVAX.

```
press SELECT
press E1
press START
```

23. Cancel the terminal mode on the prom burner.

```
press SELECT
```

24. Return the MicroVAX to the DCL level.

```
press CTRL\
```

25. Choose Motorola S record format for the prom burner.

    press SELECT
    press 87
    press START

26. Clear the prom burner's RAM.

    press SELECT
    press A4
    press START

27. Prepare the prom burner to receive the first split file. You must know how big the memory that was split is. This can be calculated by subtracting the starting address used in step 15 it from the ending address, then adding 1 to it. Divide this number by four. This is the size of each split file.

    press INPUT
    enter 0 <-- port input address
    press START
    enter 13800 <-- the size of each split file
    press START
    press 0 <-- input RAM address
    press START
    press START <-- this causes the prom burner to wait for data

28. Copy the split file from the MicroVAX to the prom burner.

    $ COPY SPLIT0.S TXA7:

29. Take note of the check sum displayed at the completion of the transfer.

30. Place an INTEL 27010 chip into the prom holder on the prom burner.

31. Burn the PROM.

```
press PROG
enter 0          <-- starting address
press START
enter 13800      <-- size of code in prom burner's RAM
press START
enter 0          <-- program RAM address

if prom burner displays "P6 FAM xx PIN xx" then

press SCROLL
press START
press SCROLL until "INTEL" appears
press START
press SCROLL until "27010" appears
press START
end if
press START
```

32. Make note of the check sum and be sure the last four digits match the last four digits of the check sum given in step 29. If they don't, the INTEL 27010 chip must be reinitialized as described in step 18, and steps 26 through 31 must be repeated.

33. Repeat steps 26 through 32 for the remaining split files (i.e., SPLIT1.S, SPLIT2.S, and SPLIT3.S).

34. Carefully replace the prom chips on the CPU-29 card, remembering that the SPLIT0, SPLIT1, SPLIT2 and SPLIT3 files represent the UPPER UPPER, UPPER MIDDLE, LOWER MIDDLE, and LOWER LOWER bits, respectively.

DOWN-LOAD [/[NO]CONTIGUOUS] microvax-filename vme-filename

Transfers a file from the MicroVAX to the VMEbus.

UP-LOAD vme-filename microvax-filename

Transfers a file from the VMEbus to the MicroVAX.

VCONVERT [/[NO]CONTIGUOUS] source-filename dest-filename

Translates an S record file on the VMEbus into an executable image.

VCOPY [/[NO]CONTIGUOUS] source-filename dest-filename

Duplicates a file on the VMEbus.

VDELETE vme-filename

Removes a file on the VMEbus.

VDIR [/DATE] [/SIZE] [/OUTPUT=filename] disk-name

Transfers a directory of the VMEbus to the MicroVAX.

VRENAME old-filename new-filename

Changes a filename on the VMEbus.

VRESET [/CPU] [/ISIO=address]

Reestablishes the communication link between the MicroVAX and VMEbus or resets the CPU-29 or ISIO cards.

VRUN /TASK-ID=number [/PRIORITY=number] vme-filename
VRUN /ISIO=address vme-filename

Begins execution of a file on the VMEbus or ISIO card.

VSETDATE

Assigns the date on the VMEbus to be equal to the MicroVAX date.

VSTATUS

Displays the task ID, priority, and status of all the tasks residing on the VMEbus.

VSTOP [/HARD] [/ISIO=address] [task-id]

Terminates execution of a task created with the VRUN command.

13.0 USER'S MANUAL

13.1 DOWN_LOAD

This command transfers a file from the MicroVAX to the VMEbus system.

Format

DOWNLOAD microvax_filename vme_filename

Parameters

microvax_filename

Specifies the name of the microvax input file to be downloaded. Wildcards cannot be used in the file specification.

vme_filename

Specifies the name of the VMEbus system output file into which the input file will be copied. Wildcards cannot be used in the file specification.

Description

The DOWNLOAD command copies a file residing on the MicroVAX to a VMEbus system file. If a file exists on the VMEbus system with the same file name as the user-specified output file, the existing file is replaced by the downloaded file. The creation date for the new file is set to the current time and date of the VMEbus system.

Qualifier

/CONTIGUOUS
/NOCONTIGUOUS (default)

Indicates whether the output file is to be contiguous, that is, whether the file must occupy consecutive physical disk blocks. This qualifier is only applied to the output file.

This qualifier should be used if the output file will be executed using the VRUN command after it has been downloaded.

13.2 UP_LOAD

This command transfers a file from the VMEbus system to the MicroVAX.

Format

UPLOAD vme_filename microvax_filename

Parameters

vme_filename

Specifies the name of the VMEbus system input file to be uploaded. Wildcards cannot be used in the file specification.

microvax_filename

Specifies the name of the MicroVAX output file into which the input file will be copied. Wildcards cannot be used in the file specification.

Description

The UPLOAD command copies a file residing on the the VMEbus system to a MicroVAX file. The creation date for the new file is set to the current time and date of the MicroVAX system.

13.3 VCONVERT

This command translates a VMEbus S record file into an executable file.

Format

VCONVERT source_filename dest_filename

Parameters

msource_filename

Specifies the name of the input file to be converted. Wildcards cannot be used in the file specification.

dest_filename

Specifies the name of the output file into which the input file will be converted. Wildcards cannot be used in the file specification.

Inscription

The VCONVERT command translates an S record file on the VMEbus system into an executable file. The new executable file also resides on the VMEbus system.

Specifying the same filename for both the source file and the destination file is not recommended. The probability of corrupting the data in the file is quite high.

The creation date for the new file is set to the current time and date of the VMEbus system.

Qualifier

/CONTIGUOUS (default)
/NOCONTIGUOUS

Indicates whether the output file is to be contiguous, that is, whether the file must occupy consecutive physical disk blocks. This qualifier is only applied to the output file.

This qualifier should be used if the output file will be executed using the VRUN command.

13.4 VCOPY

This command duplicates a file on the VMEbus.

Format

VCOPY source_filename dest_filename

Parameters

source_filename

Specifies the name of the input file to be copied. Wildcards cannot be used in the file specification.

dest_filename

Specifies the name of the output file into which the input file will be copied. Wildcards cannot be used in the file specification.

Description

The VCOPY command creates a new file, which is identical in contents and attributes to the input file.

Specifying the same filename for both the source file and the destination file is not recommended. The probability of corrupting the data in the file is quite high.

The creation date for the new file is set to the current time and date of the VMEbus system.

Qualifier

/CONTIGUOUS
/NOCONTIGUOUS (default)

Indicates whether the output file is to be contiguous, that is, whether the file must occupy consecutive physical disk blocks. This qualifier is only applied to the output file.

This qualifier should be used if the input file is in executable format and the new output file will be executed using the VRUN command.

13.5 VDELETE

This command removes a file on the VMEbus.

Format

VDELETE vme_filename

Parameter

vme_filename

Specifies the name of the file to be deleted. Wildcards cannot be used in the file specification.

Description

The VDELETE command removes a file residing on the VMEbus system.

It is important to remember that an executable file should never be deleted until it is certain that there are no tasks executing out of that file. Deletion of a file that a task is executing out of could potentially have disastrous effects on the entire VMEbus system.

13.6 VDIR

This command provides a directory listing of a VMEbus disk.

Format

VDIR disk_name

Parameter

diskname

Specifies the disk on the VMEbus system from which the directory information is to be obtained.

Description

The VDIR command obtains directory information of a given VMEbus disk. This information can then be displayed on the screen or routed to a file.

Qualifiers

/DATE

Indicates that the date and time of last modification should be displayed along with the file name. The date and time of last modification can be either the creation date of the file or the last date the file was changed. The date and time are displayed in MM-DD-YY HH:MM:SS format.

/SIZE

Specifies that the size of the file should be displayed in addition to the file name. The size of the file is specified in bytes.

/OUTPUT=filename

Indicates that the listing of the directory should be sent to a file instead of to the screen. If this qualifier is left off, the output of the directory defaults to the screen.

13.7 VRENAME

This command changes the name of a file on the VMEbus.

Format

VRENAME old_filename new_filename

Parameters

old_filename

Specifies the name of the input file to be renamed. Wildcards cannot be used in the file specification.

new_filename

Specifies the name to which the input file should be changed.

Description

The VRENAME command changes the file name of a VMEbus system. No other attributes of the file are affected by the file name change.

13.8 VRESET

This command reestablishes the communication link between the MicroVAX and VMEbus.

Format

VRESET

Description

The VRESET command resets different aspects of the VMEbus system.

If no qualifiers are specified, this command reopens the communication link between the MicroVAX and VMEbus after the link has hung. If the CPU qualifier is specified, the CPU-29 card is reset. If the ISIO qualifier is specified, the ISIO card identified by the inputed address is reset.

To reestablish the communication link, the VMEbus communication task is deleted and any pending I/O requests are cancelled. The VMEbus communication task is then recreated, thus reopening the communication link between the two systems.

If the system is still hung after a VRESET command is issued, the system must be reset via a hard reset. A hard reset causes the system to return to its initial state. This will cause all files to be lost and all task to be deleted.

The VRESET command will not affect the execution of any user created tasks or the state of any files used by these tasks.

Care should be taken when using the VMEbus system so that the VRESET command does not need to be issued. It is possible that very large segments of dynamic memory could be lost if a VRESET is done because a file transfer was abnormally exited (e.g., entering CTRL-C during a DOWN-LOAD or UP-LOAD). This could cause the transfer rates to deteriate dramatically, and if done too frequently may cause the VMEbus system to hang, necessitating a hard reset.

Qualifiers

/CPU

Indicates that the CPU-29 card should be reset without resetting any peripheral cards. This is done by transferring control to the board level support code. This qualifier cannot be used in conjunction with the /ISIO qualifier.

/ISIO=address

Indicates that an ISIO card should be reset. The entered address specifies which card will be reset.

This qualifier cannot be used in conjunction with the /CPU qualifier.

13.9 VRUN

This command begins execution of a file on the VMEbus or ISIO card.

Format

VRUN vme_filename

Parameter

vme_filename

Specifies the name of a VMEbus file that should be executed. This file must be in executable format. To change an S record file into an executable file, use the VCONVERT file.

If this file is to be executed on the VMEbus it must be a contiguous file. Contiguous files can be generated using VCONVERT, VCOPY, or DOWN_LOAD.

Description

The VRUN command begins execution of a file.

If the ISIO qualifier is specified, the VMEbus file is copied into the given ISIO card address. The copied file is then executed by the 68010 processor on the ISIO card.

If the ISIO qualifier is absent, a task is created to begin executing out of the VMEbus file. This means that the file will be executed on the CPU-29 card in a multitasking environment.

Qualifiers

/TASK-ID=number

Indicates the integer value that will be associated with the task that is created to execute the file.

This qualifier cannot be used in conjunction with the /ISIO qualifier. If the /ISIO qualifier is not specified, then the /TASK_ID qualifier must be used.

Valid task identifiers are integer values between 0 and 255. All task identifiers must be unique, except for the special identifier of 0. Many tasks can be created with an identifier of 0, but tasks with this identifier are special tasks. They cannot be deleted via the VSTOP command. In addition, they will not be displayed when the VSTATUS command is issued. For further information on tasks with identifiers of 0 see the VRTX32/68020 User's Guide.

All tasks are created in user mode with interrupts enabled (interrupt level 0).

The task identifiers of 2 and 3 are reserved for use by the communication tasks.

/PRIORITY=number (default=64)

Specifies the priority at which the created task should run.

This qualifier cannot be used in conjunction with the /ISIO qualifier.

Valid priorities are integer values between 0 and 255, with 0 being the highest priority.

/ISIO=address

Indicates the address of an ISIO card to which the VMEbus file should be copied to and run.

This qualifier cannot be used in conjunction with either the /TASK-ID or the /PRIORITY qualifier.

The address should be specified in hexadecimal format, and should lie in the CPU-29 address space.

The contents of the VMEbus file are copied to the given address. The address is then masked to determine the starting address in ISIO address space. The starting address is then written to the appropriate ISIO card, and the 68010 processor on the ISIO begins execution of the code.

Files executed on an ISIO card will not appear when the VSTATUS command is issued.

13.10 VSETDATE

This command assigns the date and time on the VMEbus to correspond to the MicroVAX.

Format

VSETDATE

Description

The current date and time of the MicroVAX are read and sent to the VMEbus system. The VMEbus upon receipt of the information sets its internal clock to the date and time specified.

By default, the VMEbus system initializes to 01-01-70 00:00:00 for a date and time. If the VSETDATE command is not issued, all files will be time stamped relative to the default date and time.

13.11 VSTATUS

This command displays the task ID, priority, and status of all the tasks executing on the VMEbus.

Format

VSTATUS

Description

The VSTATUS command shows the state of the multitasking environment on the VMEbus system. All tasks, except those with identifiers of zero, are displayed by showing their task identifier, priority, and current status. Please refer to the VRUN command for more information of the identifier

and priority of a task. Please see the VRTX32/68020 User's Guide for more information on how to interpret a task's current status.

Any files executing under the control of processors other than the 68020 on the CPU-29 card are not displayed. This means that files executing on an ISIO card would not be displayed using the VSTATUS command.

13.12 VSTOP

This command terminates execution of a task created with the VRUN command.

Format

VSTOP task_id

Parameter

task_id

Specifies the identifier of the task to be deleted. The task identifier is the same as the one specified in VRUN.

Description

If no qualifiers are specified the VSTOP command issues an IFX_ TDELETE command to delete the specified task. The task will not be deleted until it completes its current I/O call or until it makes its next I/O call.

If the HARD qualifier is specified, the VSTOP command issues a SC-TDELETE command to delete the specified task. The task will be deleted without regard to its current state or activity.

If the ISIO qualifier is specified, the halt program opcode will be written to the ISIO card identified by the address entered.

The VSTOP command should not be used indiscriminately. It may cause files to be left open and/or dynamic memory to be lost (the VMEbus system does not do garbage collection). When these adverse affects accumulate, the VMEbus system may hang, necessitating a hard reset.

Qualifiers

/HARD

Requests that the task be deleted without waiting for the next I/O operation to begin or complete.

This qualifier cannot be used in conjunction with the /ISIO qualifier.

/ISIO=address

Indicates that the HALTPROG opcode should be written to an ISIO card. The entered address specifies which card will be stopped. It is up to the operating system on the ISIO card to stop the currently executing program.

This qualifier cannot be used in conjunction with the /HARD qualifier.

APPENDIX I: Software Tape Listings

1.0 INTRODUCTION

This software tape contains copies of source files used to create the small-scale system. No object, absolute, or listing files are included.

The source listings on this tape represent the software delivered with the small-scale system for experimentation at NASA Langley in May, 1989.

2.0 TAPE FORMAT

The tape is 1600 bpi DEC VMS version 5.1 BACKUP format.

3.0 TAPE DIRECTORY STRUCTURE

The tape directory structure from the root SCRATCH directory is as follows:

AIPS	FTP Ada software from CSDL templates
SSS_SW	Ada software common to lower directories
FTPOTP_A	Experiment 10
FTPOTP_B	Experiment 11, tests 1 and 2
FTPOTP_C	Experiment 11, test 3
FTPOTP_D	Experiment 12 and 13
FTPOTP_E	Experiment 14
FTPOTP_F	Experiment 15
DIU	DIU related utilites and source code
C	DEC VAX C for DIU addr screen and frame def
FRAMES	DIU addr assignment and frame def files
COMMON	source files defining common memory params

INCLUDE	include files for common symbols
KERNEL	DIU Kernel assembly 68010 assy language
NORMAL	DIU assembly 68010 assy language
PROBE	Probe assembly 68010 assy language

EXPERIMENT	Experiment control *.COM files
COM	common to all experiments
E11	unique to Experiment 11
T1	unique to test 1
T2	unique to test 2
T3	unique to test 3
E12	unique to Experiment 12
E13	unique to Experiment 13
E14	unique to Experiment 14
E15	unique to Experiment 15
OTP	
A	
B	
C	
D	FTPOTP_D patches
E	FTPOTP_E patches
F	FTPOTP_F patches

VME	VMEbus CPU experiment control and utilities
CLOCK	FTC source selection
CONTROL	Experiment control
FAULT	Fault insertion control
OPIO	OPIO interface test program
SYNC	Manual synchronization routines
UNLOAD	DIU data unload and formatting

VULTURE	VULTURE source for simulation computer control
BSP	Board support initializiation routines
COMM	VMEbus VULTURE source

C_RTL	Code to make C library sharable
INCLUDE	Common include files
SPLIT	Split code for EPROM generation
VAX	VAX VULTURE source and .CLD files

3.0 TAPE DIRECTORY LISTING

```
Directory FC_LAB:[SCRATCH]

AIPS.DIR;1                              1  22-JUN-1989 16:12:50.57
DIU.DIR;1                               1  22-JUN-1989 17:29:34.28
EXPERIMENT.DIR;1                        1  22-JUN-1989 17:49:04.95
VME.DIR;1                               1  22-JUN-1989 16:12:47.04
VULTURE.DIR;1                           1  22-JUN-1989 16:12:42.83

Total of 5 files, 5 blocks.

Directory FC_LAB:[SCRATCH.AIPS]

SSS_SW.DIR;1                            2  22-JUN-1989 16:13:39.84

Total of 1 file, 2 blocks.

Directory FC_LAB:[SCRATCH.AIPS.SSS_SW]

EXPERIMENT_CONTROL.A;47                11   7-APR-1989 16:45:51.24
EXPERIMENT_CONTROL_B.A;10              10   5-MAY-1989 18:31:17.38
FTPOTP_A.DIR;1                          1  22-JUN-1989 16:13:52.63
FTPOTP_B.DIR;1                          1  22-JUN-1989 16:13:53.83
FTPOTP_C.DIR;1                          1  22-JUN-1989 16:13:55.52
FTPOTP_D.DIR;1                          1  22-JUN-1989 16:13:57.23
FTPOTP_E.DIR;1                          1  22-JUN-1989 16:13:59.18
FTPOTP_F.DIR;1                          1  22-JUN-1989 16:14:00.76
```

```
   IOSS_COMM_SPEC_TASK_B.A;12          13   1-MAY-1989 14:36:47.22
   IOSS_DPM_MAP_B.A;9                  55  31-MAR-1989 14:06:34.32
   IOSS_IOR_SPEC_B.A;55                73   3-MAY-1989 11:01:28.87
   IOSS_NET_MGR_CONFIG_B.A;2           99  28-MAR-1989 13:04:44.83
   IOSS_POWERUP_IOP_B.A;5              12   1-MAY-1989 14:18:44.71
   SSS_CENTRAL_DATABASE_B.A;2          79  27-MAR-1989 11:41:22.97
   SSS_CONSTANTS.A;10                  11  29-APR-1989 11:08:48.37
   SSS_GLOBAL_MEM_UTIL_B.A;4           14  12-APR-1989 11:22:35.22
   SSS_IO_REQUESTS.A;16                36  27-APR-1989 17:57:23.03
   SSS_IO_REQUESTS_B.A;43             103   4-MAY-1989 15:00:08.67
   SSS_NET_TYPES_CONST.A;4             29   8-MAR-1989 15:46:21.66
   TEST_SHARED_MEMORY_DEFS.A;6          5   1-MAY-1989 20:33:12.80

Total of 20 files, 556 blocks.

Directory FC_LAB:[SCRATCH.AIPS.SSS_SW.FTPOTP_A]

   SSS_MAIN_INIT_CP.A;6                 6   8-FEB-1989 17:12:35.84
   SSS_MAIN_PROG_CP.A;5                17  28-MAR-1989 18:23:27.94
   SSS_MAIN_PROG_IOP.A;7                8  28-MAR-1989 18:21:55.94

Total of 3 files, 31 blocks.

Directory FC_LAB:[SCRATCH.AIPS.SSS_SW.FTPOTP_B]

   SSS_MAIN_INIT_CP.A;1                 3  26-APR-1989 18:15:14.93
   SSS_MAIN_INIT_CP_B.A                10  29-APR-1989 16:58:06.01
   SSS_MAIN_PROG_CP.A;7                 8  29-APR-1989 14:30:33.95
   SSS_MAIN_PROG_IOP.A;32              13  29-APR-1989 15:51:13.87
   SSS_OD_APPLICATION_TASKS.A;1         3  23-DEC-1988 10:27:18.25
   SSS_OD_APPLICATION_TASKS_B.A;63     79  29-APR-1989 17:03:44.68

Total of 6 files, 116 blocks.
```

```
Directory FC_LAB:[SCRATCH.AIPS.SSS_SW.FTPOTP_C]

SSS_MAIN_INIT_CP.A;19                11  29-APR-1989 17:22:46.47
SSS_MAIN_PROG_CP.A;4                  6  29-MAR-1989 11:38:07.50
SSS_MAIN_PROG_IOP.A;48               15  29-APR-1989 17:01:37.84
SSS_OD_APPLICATION_TASKS.A;1          3  23-DEC-1988 10:27:18.25
SSS_OD_APPLICATION_TASKS_B.A;73      69   5-MAY-1989 18:40:33.29

Total of 5 files, 104 blocks.

Directory FC_LAB:[SCRATCH.AIPS.SSS_SW.FTPOTP_D]

IDLE_TIMER.A;10                       5   4-MAY-1989 04:03:12.24
SSS_MAIN_PROG_IOP.A;14               15   1-MAY-1989 22:51:52.35
SSS_OD_APPLICATION_TASKS.A;2          4  29-APR-1989 19:40:21.09
SSS_OD_APPLICATION_TASKS_B.A;65      86  22-JUN-1989 16:46:07.68
SSS_OD_MAIN_INIT_CP.A;12             13   1-MAY-1989 23:24:25.85
SSS_OD_MAIN_PROG_CP.A;9               7  29-APR-1989 20:07:35.40
SSS_PER_APPLICATION_TASKS.A;4         4  29-APR-1989 20:09:15.05
SSS_PER_APPLICATION_TASKS_B.A;53     78  22-JUN-1989 16:52:02.66
SSS_PER_MAIN_INIT_CP.A;15            13   1-MAY-1989 23:23:49.18
SSS_PER_MAIN_PROG_CP.A;8              6  29-APR-1989 20:06:23.52

Total of 10 files, 231 blocks.

Directory FC_LAB:[SCRATCH.AIPS.SSS_SW.FTPOTP_E]

FAULT_SHARED_MEMORY_DEFS.A;4          3   4-MAY-1989 19:27:31.59
IOP.A;1                              18   3-MAY-1989 22:15:44.48
SSS_MAIN_PROG_IOP.A;17               20   4-MAY-1989 19:37:54.63
SSS_OD_MAIN_INIT_CP.A;15             21   4-MAY-1989 19:51:19.27
SSS_OD_MAIN_PROG_CP.A;9               7  29-APR-1989 20:07:35.40
SSS_PER_MAIN_INIT_CP.A;19            21   4-MAY-1989 20:00:25.85
SSS_PER_MAIN_PROG_CP.A;8              6  29-APR-1989 20:06:23.52
```

```
Total of 7 files, 96 blocks.

Directory FC_LAB:[SCRATCH.AIPS.SSS_SW.FTPOTP_F]

   SSS_IO_REQUESTS.A;4                37  29-APR-1989 21:03:47.63
   SSS_IO_REQUESTS_B.A;17            105   4-MAY-1989 21:07:17.34
   SSS_OD_APPLICATION_TASKS_B.A;72    89  22-JUN-1989 16:52:07.20
   SSS_PER_APPLICATION_TASKS_B.A;59   81  22-JUN-1989 16:55:48.46

Total of 4 files, 312 blocks.

Directory FC_LAB:[SCRATCH.DIU]

   C.DIR;1                            1  22-JUN-1989 17:30:57.31
   COMMON.DIR;1                       1  22-JUN-1989 17:30:58.87
   INCLUDE.DIR;1                      1  22-JUN-1989 17:31:03.45
   KERNEL.DIR;1                       1  22-JUN-1989 17:31:04.39
   NORMAL.DIR;1                       1  22-JUN-1989 17:31:06.02
   PROBE.DIR;1                        1  22-JUN-1989 17:31:10.19

Total of 6 files, 6 blocks.

Directory FC_LAB:[SCRATCH.DIU.C]

   DIU_ADDR.H;1                       3   7-MAR-1989 17:15:10.89
   DIU_CONSTANTS.H;1                  3  16-NOV-1988 15:12:44.28
   FRAMES.DIR;1                       1  22-JUN-1989 17:31:15.39
   FRAME_DATA.C;1                    18  13-APR-1989 11:25:22.04
   FRAME_ID.H;1                       6   1-MAR-1989 11:24:27.75
   FRAME_TYPES.H;1                    2  15-NOV-1988 16:38:59.71
   MAKE_FRAME_FILE.C;1               21   7-APR-1989 11:01:06.70
   NODE_ADDR.H;1                      2   7-MAR-1989 17:14:24.96
```

```
Total of 8 files, 56 blocks.

Directory FC_LAB:[SCRATCH.DIU.C.FRAMES]

   ALL_DATA.DEF;1                3   7-APR-1989 10:58:59.39
   ALL_DATA.SRC;1               56   2-MAY-1989 11:13:27.17
   MAKE_TABLES.COM;1             2   2-MAY-1989 11:08:51.24
   N1DIU1.DEF;1                  3   2-MAY-1989 11:04:42.33
   N1DIU1.SRC;1                 23   2-MAY-1989 11:13:10.38
   N1DIU2.DEF;1                  3   2-MAY-1989 11:05:03.40
   N1DIU2.SRC;1                 20   2-MAY-1989 11:13:14.51
   N2DIU1.DEF;1                  3   2-MAY-1989 11:05:25.40
   N2DIU1.SRC;1                 22   2-MAY-1989 11:13:18.68
   N2DIU2.DEF;1                  3   2-MAY-1989 11:05:13.31
   N2DIU2.SRC;1                 20   2-MAY-1989 11:13:23.67

Total of 11 files, 158 blocks.

Directory FC_LAB:[SCRATCH.DIU.COMMON]

   BUFFER_ALLOCATION.SRC;1       4   8-MAR-1989 16:19:54.91
   BUFFER_CONTROL.SRC;1          9   9-APR-1989 22:11:20.81
   INTERFACE_RAM.SRC;1           5  11-APR-1989 10:34:07.02

Total of 3 files, 18 blocks.

Directory FC_LAB:[SCRATCH.DIU.INCLUDE]

   COMMAND_KERNEL.INC;1         31   9-APR-1989 18:03:51.64
   DEFS_KERNEL.INC;1             5  22-FEB-1989 12:17:15.16
   DIU_ADDR.INC;1                2  28-OCT-1988 15:16:34.21
   DIU_ERROR.INC;1               2   1-MAR-1989 15:40:55.81
   DIU_SCREEN.INC;1              4   3-MAY-1989 23:49:21.72
   FRAME_ID.INC;1                5  28-OCT-1988 15:16:32.58
```

```
   FRAME_TYPES.INC;1                    3  22-MAR-1989 13:03:46.05
   INCLUDE_FILES.SRC;1                  3   4-MAY-1989 00:07:23.59
   INTERFACE_KERNEL.INC;1               9  11-APR-1989 10:26:15.96
   ISIO.INC;1                          15  25-FEB-1989 17:13:09.10
   MC68230.INC;1                        9  24-FEB-1989 11:20:43.06
   SCN68562.INC;1                      12  30-MAR-1989 11:04:52.14

Total of 12 files, 100 blocks.

Directory FC_LAB:[SCRATCH.DIU.KERNEL]

   KERNEL.OPT;1                         2   1-MAR-1989 11:19:37.31
   KERNEL.SRC;1                        78   8-MAR-1989 16:30:30.55

Total of 2 files, 80 blocks.

Directory FC_LAB:[SCRATCH.DIU.NORMAL]

   ASM_DIU.COM;1                        1  31-MAR-1989 10:17:24.37
   DIU_INIT.SRC;1                      42  10-APR-1989 20:47:44.44
   DIU_START.SRC;1                     28   3-MAY-1989 14:03:03.41
   DIU_SVC.SRC;1                       95   3-MAY-1989 14:59:39.53
   LINK_DIU.COM;1                       1  30-MAR-1989 18:04:04.83
   N1DIU1.OPT;1                         3   2-MAY-1989 11:10:02.43
   N1DIU2.OPT;1                         3   2-MAY-1989 11:10:12.28
   N2DIU1.OPT;1                         3   2-MAY-1989 11:10:36.37
   N2DIU2.OPT;1                         3   2-MAY-1989 11:10:27.15

Total of 9 files, 179 blocks.

Directory FC_LAB:[SCRATCH.DIU.PROBE]

   ADDR_SCREEN.SRC;1                   23   3-MAY-1989 14:06:45.35
   FAST_PROBE_SVC.SRC;1                61  27-APR-1989 10:22:23.02
```

```
   FPROBE.OPT;1                          3  27-APR-1989 10:44:49.38

Total of 3 files, 87 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT]

   COM.DIR;1                             1  22-JUN-1989 17:51:36.20
   E11.DIR;1                             1  22-JUN-1989 17:52:05.91
   E12.DIR;1                             1  22-JUN-1989 17:52:25.63
   E13.DIR;1                             1  22-JUN-1989 17:52:28.38
   E14.DIR;1                             1  22-JUN-1989 17:52:30.55
   E15.DIR;1                             1  22-JUN-1989 17:52:35.83
   OTP.DIR;1                             1  22-JUN-1989 17:52:50.98

Total of 7 files, 7 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.COM]

   FAIL_HI.COM;1                         1   6-MAY-1989 01:37:19.21
   FAIL_LO.COM;1                         1   6-MAY-1989 01:36:40.31
   FAIL_NORM.COM;1                       1   6-MAY-1989 01:58:38.29
   GET_DIU.COM;3                         3  17-MAY-1989 16:52:38.91
   GET_FTP.COM;2                        17  19-MAY-1989 11:46:23.55
   RUN_DIU.COM;5                         3  17-MAY-1989 16:34:47.92
   RUN_EXP.COM;12                        6  19-MAY-1989 12:33:06.56
   SET_FTP.COM;1                         7  19-MAY-1989 11:44:02.20
   SYMBOLS.COM;11                        2  19-MAY-1989 12:26:14.44
   UNL_DIU.COM;3                         2  17-MAY-1989 16:47:35.52
   VME_LOAD_EXE.COM;1                    3   5-MAY-1989 22:48:32.21
   VULTURE.COM;2                         2   5-MAY-1989 22:55:13.21

Total of 12 files, 48 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E11]
```

```
   T1.DIR;1                                1  22-JUN-1989 17:52:11.31
   T2.DIR;1                                1  22-JUN-1989 17:52:13.36
   T3.DIR;1                                1  22-JUN-1989 17:52:15.10

Total of 3 files, 3 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E11.T1]

   LD_B.COM;1                              4   5-MAY-1989 23:21:35.56

Total of 1 file, 4 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E11.T2]

    LD_B.COM;2                             4  22-JUN-1989 17:58:10.96
    LD_B.COM;1                             4   5-MAY-1989 23:21:35.56

Total of 2 files, 8 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E11.T3]

    LD_C.COM;2                             4   6-MAY-1989 00:35:41.62

Total of 1 file, 4 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E12]

    LD_OD.COM;1                            1  18-MAY-1989 17:58:35.70
    LD_PER.COM;1                           1  18-MAY-1989 17:58:19.52

Total of 2 files, 2 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E13]
```

```
    LD_OD.COM;3                              1  18-MAY-1989 17:58:35.70
    LD_PER.COM;2                             1  18-MAY-1989 17:58:19.52

Total of 2 files, 2 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E14]

    LD_OD.COM;4                              1  18-MAY-1989 18:07:25.88
    LD_PER.COM;3                             1  18-MAY-1989 18:07:57.11

Total of 2 files, 2 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.E15]

   LD_OD.COM;5                               1  18-MAY-1989 18:10:19.37
   LD_PER.COM;4                              1  18-MAY-1989 18:10:40.34

Total of 2 files, 2 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.OTP]

   A.DIR;1                                   1  22-JUN-1989 17:52:58.55
   B.DIR;1                                   1  22-JUN-1989 17:52:59.87
   C.DIR;1                                   1  22-JUN-1989 17:53:01.53
   D.DIR;1                                   1  22-JUN-1989 17:53:02.76
   E.DIR;1                                   1  22-JUN-1989 17:53:04.82
   F.DIR;1                                   1  22-JUN-1989 17:53:06.92

Total of 6 files, 6 blocks.

Directory FC_LAB:[SCRATCH.EXPERIMENT.OTP.D]

    PATCH_IOP.COM;1                          1  18-MAY-1989 17:47:01.10
```

```
Total of 1 file, 1 block.

Directory FC_LAB:[SCRATCH.EXPERIMENT.OTP.E]

   PATCH_IOP.COM;2                          1  18-MAY-1989 18:05:36.48

Total of 1 file, 1 block.

Directory FC_LAB:[SCRATCH.EXPERIMENT.OTP.F]

    PATCH_IOP.COM;1                         1  18-MAY-1989 18:09:22.36

Total of 1 file, 1 block.

Directory FC_LAB:[SCRATCH.VME]

   CLOCK.DIR;1                              1  22-JUN-1989 17:39:04.81
   CONTROL.DIR;1                            1  22-JUN-1989 17:39:06.53
   C_SYMBOLS.COM;2                          1  22-JUN-1989 17:48:32.12
   FAST_OPIO.OPT;2                          1  16-JUN-1988 16:31:07.40
   FAST_OPIO.SRC;2                          7   7-JUN-1988 15:19:00.30
   FAULT.DIR;1                              1  22-JUN-1989 17:39:08.78
   INIT_UTIL.SRC;5                          7  16-JUN-1988 17:29:01.46
   OPIO.DIR;1                               1  22-JUN-1989 17:36:40.73
   OPIO_INIT.SRC;18                         8  26-OCT-1988 16:57:51.98
   SYNC.DIR;1                               1  22-JUN-1989 17:39:10.70
   UNLOAD.DIR;1                             1  22-JUN-1989 17:39:12.52

Total of 11 files, 30 blocks.

Directory FC_LAB:[SCRATCH.VME.CLOCK]

   VFTC.C;1                                 4   6-APR-1989 11:14:51.73
```

```
   VFTC.OPT;1                              2   6-APR-1989 11:16:51.57

Total of 2 files, 6 blocks.

Directory FC_LAB:[SCRATCH.VME.CONTROL]

   CONTROL.COM;1                           1   6-APR-1989 11:06:03.03
   CONTROL.DAT;4                           2   6-APR-1989 14:16:52.89
   CONTROL.EDIT;1                          2   5-APR-1989 18:01:32.42
   CONTROL.OPT;2                           2   5-APR-1989 19:28:25.12
   CONTROL_TASK.C;41                      28  11-APR-1989 10:41:33.70

Total of 5 files, 35 blocks.

Directory FC_LAB:[SCRATCH.VME.FAULT]

   FAULT.DAT;1                             2   5-APR-1989 17:54:35.34
   FAULT.EDIT;1                            3   4-APR-1989 19:29:31.57
   FAULT.FDL;2                             1   5-APR-1989 09:11:03.00
   FAULT_BUS_INIT.C;4                      4  10-APR-1989 12:00:46.33
   FAULT_ISR.SRC;17                        6   5-APR-1989 13:28:44.58
   FAULT_TASK.C;77                        12  10-APR-1989 11:36:09.34
   FBUSINIT.OPT;1                          2  10-APR-1989 11:43:30.42
   FINSERT.OPT;3                           2   4-APR-1989 19:16:06.22
   LOAD_TIMER.SRC;6                        3   5-APR-1989 10:35:10.75
   PRINT_FAULT.C;11                        3   9-APR-1989 20:08:48.33

Total of 10 files, 38 blocks.

Directory FC_LAB:[SCRATCH.VME.SYNC]

   ASSERT.C;2                              3  10-APR-1989 00:39:22.05
   ASSERT.OPT;2                            2  10-APR-1989 00:44:32.15
   FBUS.C;2                                3  10-APR-1989 00:59:07.61
```

```
   FBUS.OPT;1                              2  10-APR-1989 00:44:44.77
   FSYNC.C;14                              4  10-APR-1989 01:07:49.60
   FSYNC.OPT;1                             2   9-APR-1989 20:30:26.84
   GSYNC.C;1                               4  13-APR-1989 14:19:49.86
   GSYNC.OPT;1                             2  13-APR-1989 14:20:27.58
   VSYNC.C;11                              3   9-APR-1989 20:44:34.13
   VSYNC.OPT;3                             2   3-APR-1989 16:43:40.04

Total of 10 files, 27 blocks.

Directory FC_LAB:[SCRATCH.VME.UNLOAD]

   CONVERT.C;37                            8   4-APR-1989 15:17:09.13
   GET_EXPERIMENT_TIME.C;5                 3  28-MAR-1989 08:45:05.06
   PRINT_UTILITIES.C;10                    5   9-APR-1989 23:18:17.88
   READ_TIMER.SRC;7                        6  28-MAR-1989 13:50:20.85
   UNLOAD.C;84                            32  11-APR-1989 12:28:43.61
   UNLOAD.COM;9                            1  27-MAR-1989 15:32:16.01
   UNLOAD.OPT;6                            2  27-MAR-1989 15:30:50.18
   UNLOAD_BUFFER.C;9                       3   3-APR-1989 16:55:15.87

Total of 8 files, 60 blocks.

Directory FC_LAB:[SCRATCH.VULTURE]

   BSP.DIR;1                               1  22-JUN-1989 16:13:24.17
   COMM.DIR;1                              2  22-JUN-1989 16:13:26.13
   C_RTL.DIR;1                             1  22-JUN-1989 16:13:28.50
   INCLUDE.DIR;1                           1  22-JUN-1989 16:13:29.89
   SPLIT.DIR;1                             1  22-JUN-1989 16:13:31.99
   VAX.DIR;1                               2  22-JUN-1989 16:13:34.46

Total of 6 files, 8 blocks.
```

```
Directory FC_LAB:[SCRATCH.VULTURE.BSP]

BUILD_BSP.COM;1                1  24-JAN-1989 08:36:14.43
IFX_SETUP.C;1                  4  12-JAN-1989 15:04:37.31
INIT_C_RTL.C;1                 7  25-JAN-1989 07:19:36.28
INIT_DRAM.SRC;1                3  18-JAN-1989 10:30:33.58
INIT_UTIL.SRC;1                7   6-DEC-1988 13:44:31.86
OPIO_INIT.SRC;1               10  16-JAN-1989 10:38:43.75
RECEIVE_INIT.SRC;1             2   5-DEC-1988 11:10:11.42
VAX_ISR.SRC;1                  4  24-JAN-1989 08:38:42.58

Total of 8 files, 38 blocks.

Directory FC_LAB:[SCRATCH.VULTURE.COMM]

BSP.OPT;1                      3  25-JAN-1989 07:23:02.30
BUILD_COMM.COM;1               2  24-JAN-1989 08:36:50.11
CHECK_ADDRESS.SRC;1            5  24-JAN-1989 14:56:41.04
COMM.C;1                       7  16-JAN-1989 07:45:05.67
DOWN_LOAD.C;1                  6  21-NOV-1988 16:39:35.46
ESTABLISH_LINK.C;1             5  14-NOV-1988 13:38:12.77
MAIN.C;2                       3   2-FEB-1989 13:38:01.32
RESET_CPU.SRC;1                1  23-JAN-1989 11:18:16.39
RESET_LINK.C;1                 1  22-SEP-1988 11:04:26.00
STAT_TO_VAX.C;1                6  20-JAN-1989 12:54:03.12
TOOL.C;1                      21  16-JAN-1989 07:40:26.68
UP_LOAD.C;1                    5  14-NOV-1988 13:38:48.35
VCONVERT.C;1                  16  18-NOV-1988 16:07:21.65
VCOPY.C;1                      4  17-NOV-1988 10:13:21.13
VDELETE.C;1                    2  29-NOV-1988 10:17:53.03
VDIR.C;1                      18  29-NOV-1988 10:38:48.28
VRENAME.C;1                    2  14-NOV-1988 13:40:45.52
VRESET.C;1                     6  23-JAN-1989 11:26:31.25
VRUN.C;1                      11  23-JAN-1989 09:26:34.07
```

```
   VSETDATE.C;1                          3  10-NOV-1988 08:42:07.86
   VSTATUS.C;1                           5  28-NOV-1988 15:00:19.99
   VSTOP.C;1                             4  23-JAN-1989 09:19:51.98

Total of 22 files, 136 blocks.

Directory FC_LAB:[SCRATCH.VULTURE.C_RTL]

   BUILD_C_RTL.COM;1                     1  23-JAN-1989 14:17:02.07
   C_RTL_IL.INC;1                        6  22-NOV-1988 10:21:45.27
   C_RTL_IL.SRC;1                       15  18-NOV-1988 10:47:40.48
   PROM_RTL.SRC;1                       11  22-NOV-1988 10:20:56.32

Total of 4 files, 33 blocks.

Directory FC_LAB:[SCRATCH.VULTURE.INCLUDE]

   DRQ3B.H;20                            6  23-JAN-1989 09:41:18.22
   HX$DEF.H;1                            9   2-MAY-1988 09:30:45.82
   OPIO.H;9                             16  27-OCT-1988 09:42:39.75
   OPIO.INC;17                          14  18-OCT-1988 14:47:37.19
   OPIO_DEF.H;45                         7  23-JAN-1989 09:41:42.37

Total of 5 files, 52 blocks.

Directory FC_LAB:[SCRATCH.VULTURE.SPLIT]

   SPLIT.C;2                            11   2-FEB-1989 13:36:07.62
   SPLIT.OPT;1                           1  23-JAN-1989 08:35:31.95

Total of 2 files, 12 blocks.

Directory FC_LAB:[SCRATCH.VULTURE.VAX]
```

```
CLEAR.C;1                          2  16-DEC-1988 15:24:24.40
DOWN_LOAD.C;2                     16  16-JAN-1989 10:49:41.09
DOWN_LOAD.CLD;2                    1  11-JAN-1989 07:53:41.00
DRQ3B_LIB.C;2                     19  16-JAN-1989 12:30:01.88
ESTABLISH_LINK.C;1                 8   2-NOV-1988 14:08:50.28
ESTABLISH_LINK.CLD;1               1  11-AUG-1988 14:00:39.63
UP_LOAD.C;2                       15  16-JAN-1989 10:51:39.95
UP_LOAD.CLD;2                      1  11-JAN-1989 07:53:54.00
VCONVERT.C;1                       8   2-DEC-1988 15:38:05.20
VCONVERT.CLD;2                     1  11-JAN-1989 07:54:09.00
VCOPY.C;1                          8   2-DEC-1988 15:38:19.64
VCOPY.CLD;2                        1  11-JAN-1989 07:54:33.00
VDELETE.C;1                        7   2-DEC-1988 15:38:32.23
VDELETE.CLD;2                      1  11-JAN-1989 07:54:46.00
VDIR.C;2                          14  16-JAN-1989 11:01:55.94
VDIR.CLD;2                         1  11-JAN-1989 07:54:55.00
VRENAME.C;1                        7   2-DEC-1988 15:39:06.10
VRENAME.CLD;2                      1  11-JAN-1989 07:55:12.00
VRESET.C;5                        10  23-JAN-1989 09:47:36.38
VRESET.CLD;4                       1  23-JAN-1989 09:43:18.68
VRUN.C;4                          11  19-JAN-1989 08:56:17.24
VRUN.CLD;2                         1  11-JAN-1989 07:55:38.00
VSETDATE.C;1                       7   2-DEC-1988 15:39:44.84
VSETDATE.CLD;2                     1  11-JAN-1989 07:55:52.00
VSTATUS.C;1                        9   2-DEC-1988 15:40:06.64
VSTATUS.CLD;2                      1  11-JAN-1989 07:56:11.00
VSTOP.C;4                          8  17-JAN-1989 07:33:03.45
VSTOP.CLD;7                        1  17-JAN-1989 07:23:49.24

Total of 28 files, 162 blocks.

Grand total of 44 directories, 279 files, 2865 blocks.
```

APPENDIX J: DOCUMENTATION PACKAGES

DOCUMENTATION PACKAGE A: VMEBUS SIMULATION COMPUTER CONFIGURATION

Subject: VMEbus Simulation Computer Addressing
Date: June 16, 1988
Rev: August 4, 1988

INTRODUCTION:

This document describes the basics of VMEbus memory addressing and the use of each memory area by the CPU-29 and DMA VMEbus masters in the VMEbus simulation computer.

The example used for discussion of memory addressing herein is the implementation of DIU simulators for the IAPSA Small Scale System using the VMEbus simulation computer as a base.

There are three types of memory present on the VMEbus in this example: local memory, such as CPU-29 local RAM, accessable only to devices on a particular board; global memory, such as the DRAM-E4-xxx dynamic RAM, accessable to all VMEbus masters; and dual port memory, such as that on the ISIO-2, accessable to both local devices and VMEbus masters.

DISCUSSION:

VMEbus Data Size and Addressing:

The VMEbus specification supports three types of board addressing: extended 32 bit, standard 24 bit and short 16 bit addressing. A board may recognize or ignore the 8 or 16 upper address bits depending on its design. Board data may be 8, 16, 24, or 32 bit.

The VMEbus specification includes 6 lines in addition to the address and control lines which are used to qualify board selection. These are called the address modifier (AM) lines. An address modifier code is output by a bus master any time the VMEbus is accessed. The codes used in the VMEbus simulation computer specify the addressing mode: extended (A32), standard (A24), or short (A16); the privilege: supervisory (S) or non-privileged (N); and the type of access: program (PA) or data (DA). (For example, A32:NPA represents extended addressing non-privileged program access.)

CPU-29 VMEbus Interface:

The CPU-29 uses a Motorola 68020 to implement a full 32 bit VMEbus. Local memory and I/O devices occupy a portion of the 32 bit (4 GB) 68020 address range. Local memory and I/O devices are not accessible from the VMEbus. Access to these devices by the local 68020 causes no activity on the VMEbus.

To accomodate the access requirements of different boards residing on the VMEbus, the CPU-29 maps its address space for different types of accesses. The bus sizing for several of the address ranges, below, is software programmable to be either D16 or D32. The DIU simulation computer is programmed to use the D32 bus size for these ranges. Only boards which have a 32 bit data path can be located in a D32 address range. D16 boards which are accessed as D32 boards will only drive the D0 thru D15 data lines and because of the VMEbus termination, data appearing on D16 thru D31 will at logic 1.

Note that bus sizing does not affect the instructions which can be used in the 68020. The bus sizing only affects the way in which the data are obtained from the boards. A D32 board may be placed in D16 address space, but at the price of additional access cycles for long word access.

The Table 1, below, describes the mapping of the CPU-29 68020 address space to the VMEbus address space, including addressing mode and data bus size:

TABLE 1: CPU-29 to VMEbus Mapping

CPU-29	VMEbus	addr:data_size
0000 0000 - 000F FFFF	(no access: CPU-29 local RAM)	
0010 0000 - FAFF FFFF	0100 0000 - FAFF FFFF	A32:PROGRAMMABLE
FB00 0000 - FBFE FFFF	FB00 0000 - FBFE FFFF	A24:PROGRAMMABLE
FBFF 0000 - FBFF FFFF	FBFF 0000 - FBFF FFFF	A16:PROGRAMMABLE
FC00 0000 - FCFE FFFF	FC00 0000 - FCFE FFFF	A24:D16
FCFF 0000 - FCFF FFFF	FCFF 0000 - FCFF FFFF	A16:D16
FD00 0000 - FEFF FFFF	FD00 0000 - FEFF FFFF	A24:PROGRAMMABLE
FF00 0000 - FF7F FFFF	(no access: CPU-29 local EPROM)	
FF80 0000 - FFFF FFFF	(no access: CPU-29 local I/O)	

NOTES:
1. A24 devices on the VMEbus ignore the upper two address bytes. Addresses xx00 0000 thru xxFE FFFF are decoded as 00 0000 thru FE FFFF.
2. A16 devices on the VMEbus ignore the upper four address bytes. Addressess xxxx 0000 thru xxxx FFFF are decoded as 0000 thru FFFF.
3. PROGRAMMABLE access areas are defined as D32.

Global RAM (DRAM-E4xxx) VMEbus Interface:

The 14.75 MB VMEbus RAM is a D32 two board set configured to respond to A32:NPA, NDA, SPA, SDA and A24:NPA, SPA access. The VMEbus address range of the memory is XX10 0000 to XXFB FFFF, repeating every 16 MB.

DIU Simulator (ISIO-2) VMEbus Interface:

The DIU simulators (ISIO-2 boards) are D16 boards, with 128 KB of RAM, 120KB of which is dual port RAM. They are configured to respond only to A24:NDA and A24:SDA access with base addresses in the range of XX00 0000 to XX0C 0000.

DMA Controller (OPIO-1) VMEbus Interface:

As a slave, the DMA controller (OPIO-1 board) is configured to respond only to A16:NDA and A16:SDA accesses. As a bus master (when performing DMA operations) it only supports A24:D16 transfers using one of NDA, NPA, SDA, or SPA software programmable address modifiers.

Fault Insertion Controller (OPIO-1) VMEbus Interface:

The fault insertion controller is implemented using devices located on the OPIO-1 board. It is accessed as a VMEbus slave device.

Description of Board Addressing Scheme:

The board addressing scheme described above, allows access to the VMEbus RAM and the DIU simulator dual port RAM by both the CPU-29 and the DMA controller. It also provides a degree of protection to inadvertently accessing data in the DIU simulator dual port RAM when using the DMA controller to transfer data between the VMEbus and the uVAX II.

The CPU-29 accesses local and VMEbus RAM as contiguous A32:D32 RAM from 0 thru 00FB FFFF, giving it access to a total of 15.75 MB of contiguous RAM. (VMEbus RAM is accessed by the CPU-29 as either A32:D32:NDA, NPA, SDA, or SPA.) The DIU simulator dual port RAM is accessed as A24:D16:NDA or SDA over the address range FC00 0000 thru FC0D FFFF. The DMA controller is accessed as A16:D16:NDA or SDA with a base address of FCFF 0000.

The DMA controller has access to the 14.75 MB VMEbus RAM using A24:NPA,SPA over the address range of 10 0000 thru FB FFFF. The DIU simulator dual port RAM is accessed using A24:NDA or SDA over the address range of 0 thru 0D FFFF. By using the NPA or SPA AM codes for DMA access to VMEbus RAM and NDA or SDA AM codes for DMA access to the DIU simulator dual port RAM, inadvertent access to either area is prevented.

The VMEbus specification states that A32 boards MUST monitor all 32 address lines; A24 devices MUST monitor the lower 24 address lines and MAY monitor the upper 8 address lines. The DRAM board responds to both A32 and A24 addressing while ignoring the upper 8 address lines. This is not strictly in accordance with the VMEbus specification for an A32 device.

CPU-29 VMEbus Address Definitions:

Table 2 shows the CPU-29 addressing assignments used for the boards in the VMEbus simulation computer for the Small Scale System:

Table 2: CPU-29 Addressing

Address range	VMEbus Axx:Dxx:AM	Description
0000 0000 - 000F FFFF	n/a	Local CPU-29 SRAM
XX10 0000 - XXFB FFFF	A32:D32:NDA,NPA,SDA,SPA	VMEbus 14.75 MB DRAM
FC00 0000 - FC01 FFFF	A24:D16:NDA,SDA	DIUSIM1 Cntrl and RAM
FC02 0000 - FC03 FFFF	A24:D16:NDA,SDA	DIUSIM2 Cntrl and RAM
FC04 0000 - FC05 FFFF	A24:D16:NDA,SDA	DIUSIM3 Cntrl and RAM
FC06 0000 - FC07 FFFF	A24:D16:NDA,SDA	DIUSIM4 Cntrl and RAM
FC08 0000 - FC09 FFFF	A24:D16:NDA,SDA	DIUSIM5 Cntrl and RAM
FC0A 0000 - FC0B FFFF	A24:D16:NDA,SDA	DIUSIM6 Cntrl and RAM
FC0C 0000 - FC0D FFFF	A24:D16:NDA,SDA	DIUSIM7 Cntrl and RAM
FCFF 0000 - FCFF 01FF	A16:D16:NDA,SDA	DMA Controller
FF00 0000 - FF3F FFFF	n/a	Local EPROM
FF80 0000 - FFFF FFFF	n/a	Local I/O Devices

NOTE: Some of the boards are also accessable at other addresses. Only the addressing defined above will be used by the CPU-29.

DMA Controller Address Definitions:

Table 3 shows the DMA controller addressing assignment for VMEbus boards in the Small Scale System:

Table 3: DMA Controller Addressing

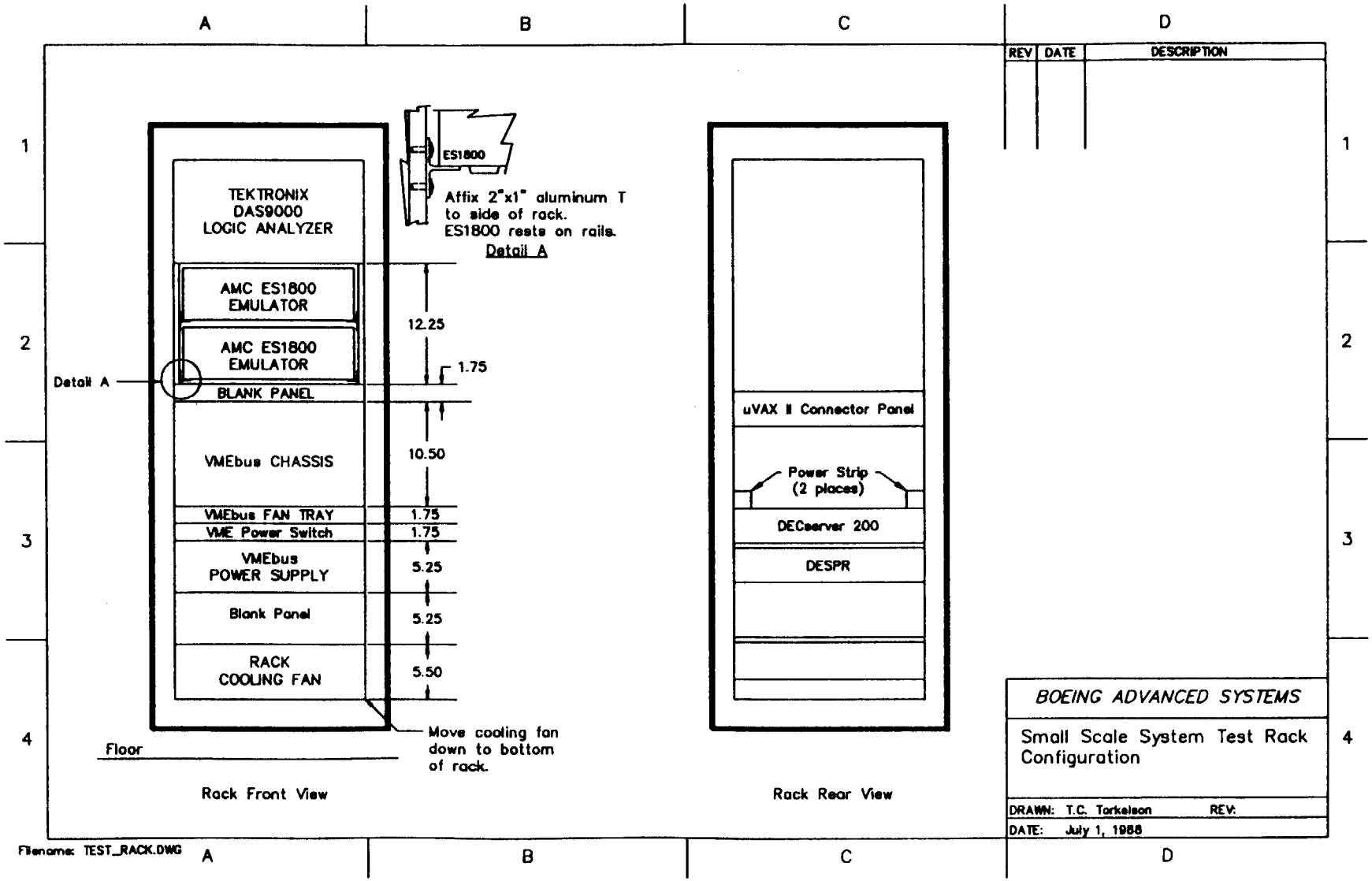
Address range	VMEbus Axx:Dxx:AM	Description
00 2000 - 01 FFFF	A24:D16:NDA,SDA	DIUSIM0 Dual Port RAM
02 2000 - 03 FFFF	A24:D16:NDA,SDA	DIUSIM1 Dual Port RAM
04 2000 - 05 FFFF	A24:D16:NDA,SDA	DIUSIM2 Dual Port RAM
06 2000 - 07 FFFF	A24:D16:NDA,SDA	DIUSIM3 Dual Port RAM
08 2000 - 09 FFFF	A24:D16:NDA,SDA	DIUSIM4 Dual Port RAM
0A 2000 - 0B FFFF	A24:D16:NDA,SDA	DIUSIM5 Dual Port RAM
0C 2000 - 0D FFFF	A24:D16:NDA,SDA	DIUSIM6 Dual Port RAM
10 0000 - FB FFFF	A24:D16:NPA,SPA	VMEbus 14.75 MB DRAM

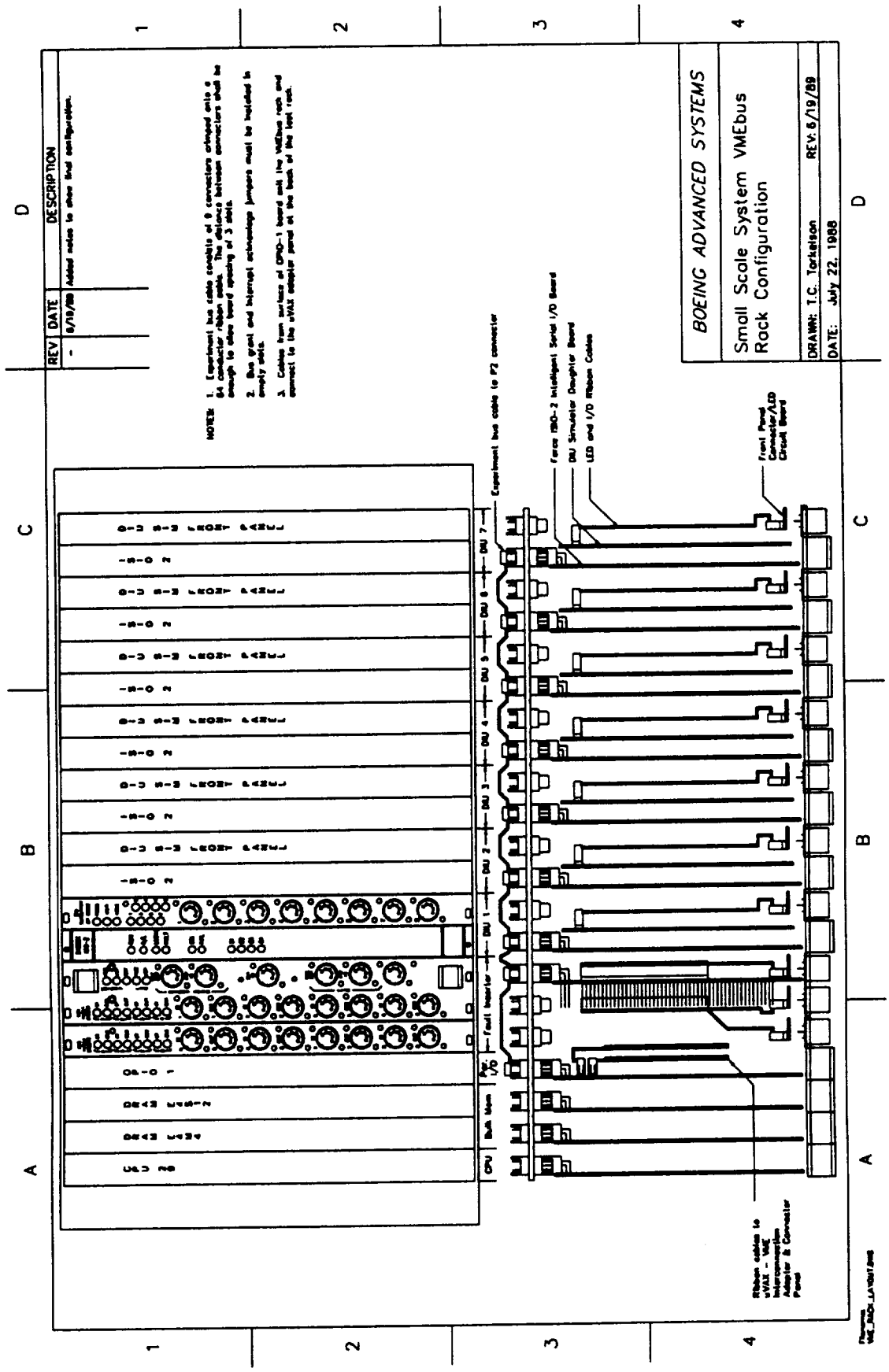
Physical Board Setups:

DRAM-E4M4 / DRAM-E4S12 Setup:

Address start	= $XX10 0000
First not on board address	= $XXFC 0000
AM	= A32:NDA,SDA,NPA,SPA, A24:NPA,SPA

ISIO-2:	DIUSIM0	board base = $00 0000	AM = A24:NDA,SDA
ISIO-2:	DIUSIM1	board base = $02 0000	AM = A24:NDA,SDA
ISIO-2:	DIUSIM2	board base = $04 0000	AM = A24:NDA,SDA
ISIO-2:	DIUSIM3	board base = $06 0000	AM = A24:NDA,SDA
ISIO-2:	DIUSIM4	board base = $08 0000	AM = A24:NDA,SDA
ISIO-2:	DIUSIM5	board base = $0A 0000	AM = A24:NDA,SDA
ISIO-2:	DIUSIM6	board base = $0C 0000	AM = A24:NDA,SDA
OPIO-1:		board base = $FF 0000	AM = A16:NDA,SDA


A
B
C
D
1
2
3
4
REV
DATE
DESCRIPTION
TEKTRONIX
DAS9000
LOGIC ANALYZER
AMC ES1800
EMULATOR
AMC ES1800
EMULATOR
Detail A
BLANK PANEL
VMEbus CHASSIS
VMEbus FAN TRAY
VME Power Switch
VMEbus
POWER SUPPLY
Blank Panel
RACK
COOLING FAN
12.25
1.75
10.50
1.75
1.75
5.25
5.25
5.50
ES1800
Affix 2"x1" aluminum T
to side of rack.
ES1800 rests on rails.
Detail A
Move cooling fan
down to bottom
of rack.
Floor
Rack Front View
uVAX II Connector Panel
Power Strip
(2 places)
DECserver 200
DESPR
Rack Rear View
BOEING ADVANCED SYSTEMS
Small Scale System Test Rack
Configuration
DRAWN: T.C. Torkelson
REV:
DATE: July 1, 1988
Filename: TEST_RACK.DWG



ORIGINAL PAGE IS
OF POOR QUALITY

DOCUMENTATION PACKAGE B: OPIO-1 PARALLEL INTERFACE MODIFICATIONS

Subject: OPIO-1 Modifications for Simulation Computer
By: T.C. Torklson
Date: May 27, 1988
Rev: September 29, 1988

Introduction:

To use the Force OPIO-1 as an interface to the DEC uVAX II DRQ3B interface will require an adapter board and some modifications to the OPIO-1 itself because of DRQ3B interface logic levels.

Modifications:

1. Remove HP optoisolators from J13-J18, J25-J30, J37-J42, and J49-J54.

2. Install .3" shorting plugs between pins 2/7 and 3/6 of J13-J18, J25-J30, J37-J42, and J49-J54. (48 locations)

3. Install .3" shorting plugs between JP4-1/JP14-1, JP4-2/JP14-2, JP5-1/JP15-1, JP6-1/JP16-1, and JP7-1/JP17-1

4. Install a diode capable of sustaining 100 mA between JP5-2/JP15-2, JP6-2/JP16-2, and JP7-2/JP17-2. A device in a small signal diode package is preferred, as it will plug directly into the sockets at the JP locations. Cathode of diodes are connected to JP5-2, JP6-2 and JP7-2.

5. Install the OPIO_DELAY_GENERATOR daughter board.

   a. Remove the chip at J1.

   b. Connect wire wrap wire jumpers between J4-32 / J3-32 / J2-32.

   c. Install daughter board in place in J1.

   d. Connect wire wrap jumper from J2-32 to pin 1 of daughter board.

   e. Connect wire wrap jumper from J4-13 to pin 2 of daughter board.

   f. Connect wire wrap jumper from daughter board pin 3 to P2-17c.

   g. Install 68230 and OPIO_DELAY_GENERATOR EPLD in the sockets provided on the daughter board

PRECEDING PAGE BLANK NOT FILMED

PAGE J-8 INTENTIONALLY BLANK

Subject: Allocation of OPIO-1 I/O Connections
By: T.C. Torkelson

Date: May 12, 1988

Reference: OPIO-1 Modifications for Simulation Computer

NOTES: 1. All signals appearing on Z2 also appear on the VMEbus P2 connector, labelled Z1 in the OPIO-1 manual.

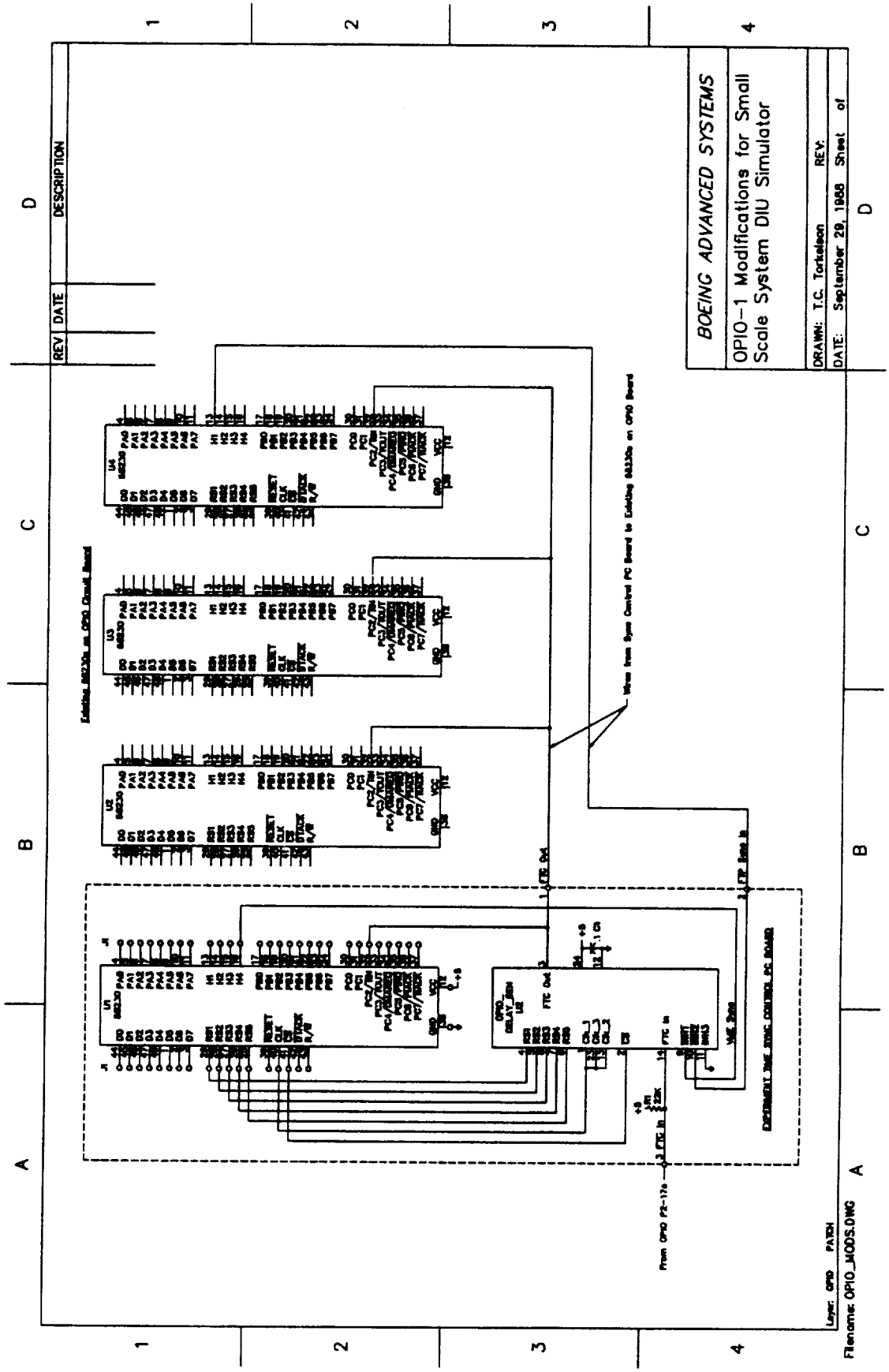
2. Blank entries are unused or spare.

Connnector	OPIO Name	Signal name	Chip		
Z2-1c		F_SYNC	J4-13	(H1)	in
Z2-1a					
Z2-2c	XQHA	V_SYNC	J1-16	(H4)	out
Z2-2a	XPQGND	gnd			
Z2-3c	XQHB	EXT_EVENT	J1-15	(H3)	in
Z2-3a	XQDD				
Z2-4c	XQD0		J1-17	(PB0)	in
Z2-4a	XQD1		J1-18	(PB1)	in
Z2-5c	XQD2		J1-19	(PB2)	in
Z2-5a	XQD3		J1-20	(PB3)	in
Z2-6c	XQD4		J1-21	(PB4)	in
Z2-6a	XQD5		J1-22	(PB5)	in
Z2-7c	XQD6		J1-23	(PB6)	in
Z2-7a	XQD7		J1-24	(PB7)	in
Z2-8c	XPQVCC	+5 vdc			
Z2-8a	XPQGND	gnd			
Z2-9c					
Z2-9a					
Z2-10c	XSHA	CH1 !CLR IN	J2-16	(H4)	out
Z2-10a	XRSGND	gnd			
Z2-11c	XSHB		J2-15	(H3)	in
Z2-11a	XSDD				
Z2-12c	XSD0	FUNCT OUT 0	J2-17	(PB0)	in
Z2-12a	XSD1	FUNCT OUT 1	J2-18	(PB1)	in
Z2-13c	XSD2	FUNCT OUT 2	J2-19	(PB2)	in
Z2-13a	XSD3	FUNCT OUT 3	J2-20	(PB3)	in
Z2-14c	XSD4	FUNCT OUT 4	J2-21	(PB4)	in
Z2-14a	XSD5	FUNCT OUT 5	J2-22	(PB5)	in
Z2-15c	XSD6	CH0 !INIT OUT	J2-23	(PB6)	in
Z2-15a	XSD7	CH1 !INIT OUT	J2-24	(PB7)	in
Z2-16c	XRSVCC	+4.4 vdc			
Z2-16a	XRSGND	gnd			

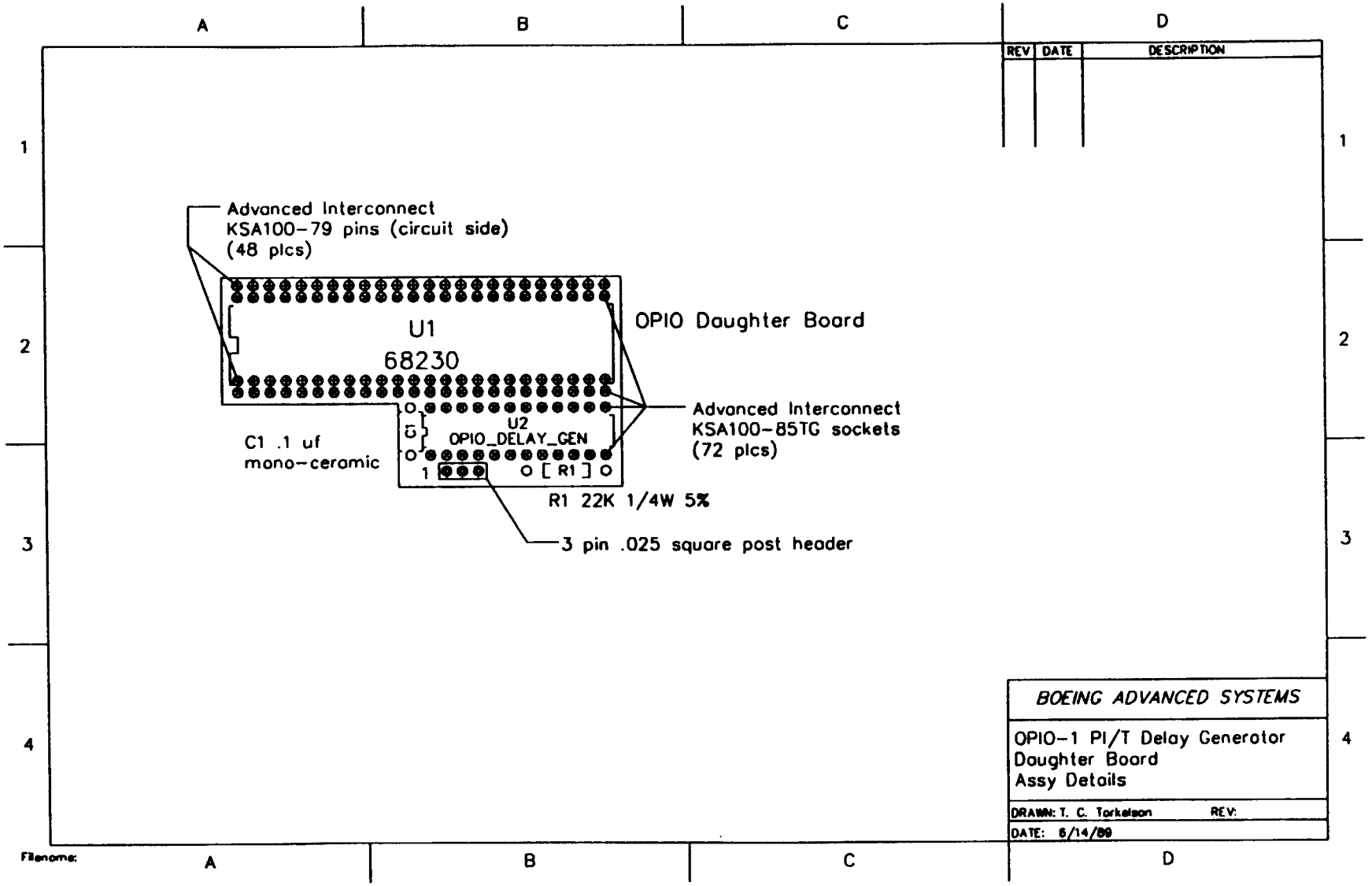
Connnector	OPIO Name	Signal name	Chip		
Z2-17c		R_FTC			
Z2-17a					
Z2-18c	XTHA	!FACK	J3-13	(H1)	in
Z2-18a	XTUGND	gnd			
Z2-19c	XTHB	!FTSB	J3-14	(H2)	out
Z2-19a	XTDD		J3-30	(PC0)	out
Z2-20c	XTD0	FB0	J3-4	(PA0)	out
Z2-20a	XTD1	FB1	J3-5	(PA1)	out
Z2-21c	XTD2	FB2	J3-6	(PA2)	out
Z2-21a	XTD3	FB3	J3-7	(PA3)	out
Z2-22c	XTD4	FB4	J3-8	(PA4)	out
Z2-22a	XTD5	FB5	J3-9	(PA5)	out
Z2-23c	XTD6	FB6	J3-10	(PA6)	out
Z2-23a	XTD7	FB7	J3-11	(PA7)	out
Z2-24c	XTUVCC	+4.4 vdc			
Z2-24a	XTUGND	gnd			
Z2-25c					
Z2-25a					
Z2-26c	XVHA	F_SYNC	J4-13	(H1)	in
Z2-26a	XVWGND	gnd			
Z2-27c	XVHB		J4-14	(H2)	out
Z2-27a	XVDD	CH0 !CLR IN	J4-30	(PC0)	out
Z2-28c	XVD0	FUNCT IN 0	J4-4	(PA0)	out
Z2-28a	XVD1	FUNCT IN 1	J4-5	(PA1)	out
Z2-29c	XVD2	FUNCT IN 2	J4-6	(PA2)	out
Z2-29a	XVD3	FUNCT IN 3	J4-7	(PA3)	out
Z2-30c	XVD4	FUNCT IN 4	J4-8	(PA4)	out
Z2-30a	XVD5	FUNCT IN 5	J4-9	(PA5)	out
Z2-31c	XVD6	STROBE !CLR	J4-10	(PA6)	out
Z2-31a	XVD7	CH0 !EOP IN	J4-11	(PA7)	out
Z2-32c	XVWVCC	+4.4 vdc			
Z2-32a	XVWGND	gnd			

Connnector	OPIO Name	Signal name	Chip		
Z3-1c					
Z3-1a					
Z3-2c	XPHA	CH0 !ACK OUT	J1-13	(H1)	in
Z3-2a	XPQGND	gnd			
Z3-3c	XPHB	CH0 !STROBE IN	J1-14	(H2)	out
Z3-3a	XPDD		J1-30	(PC0)	out
Z3-4c	XPD0	CH0 IN 0	J1-4	(PA0)	out
Z3-4a	XPD1	CH0 IN 1	J1-5	(PA1)	out
Z3-5c	XPD2	CH0 IN 2	J1-6	(PA2)	out
Z3-5a	XPD3	CH0 IN 3	J1-7	(PA3)	out
Z3-6c	XPD4	CH0 IN 4	J1-8	(PA4)	out
Z3-6a	XPD5	CH0 IN 5	J1-9	(PA5)	out
Z3-7c	XPD6	CH0 IN 6	J1-10	(PA6)	out
Z3-7a	XPD7	CH0 IN 7	J1-11	(PA7)	out
Z3-8c	XPQVCC	+5 vdc			
Z3-8a	XPQGND	gnd			
Z3-9c					
Z3-9a					
			J4-13	(H1)	in
Z3-10c	XRHA	CH0 !ACK OUT			
Z3-10a	XRSGND		J2-14	(H2)	out
Z3-11c	XRHB		J2-30	(PC0)	out
Z3-11a	XRDD		J2-4	(PA0)	out
Z3-12c	XRD0	CH0 IN 8	J2-5	(PA1)	out
Z3-12a	XRD1	CH0 IN 9	J2-6	(PA2)	out
Z3-13c	XRD2	CH0 IN 10	J2-7	(PA3)	out
Z3-13a	XRD3	CH0 IN 11	J2-8	(PA4)	out
Z3-14c	XRD4	CH0 IN 12	J2-9	(PA5)	out
Z3-14a	XRD5	CH0 IN 13	J2-10	(PA6)	out
Z3-15c	XRD6	CH0 IN 14	J2-11	(PA7)	out
Z3-15a	XRD7	CH0 IN 15			
Z3-16c	XRSVCC	+4.4 vdc			
Z3-16a	XRSGND	gnd			

Connnector	OPIO Name	Signal name	Chip		
Z3-17c					
Z3-17a					
Z3-18c	XUHA	CH1 !ACK IN	J3-16	(H4)	out
Z3-18a	XTUGND	gnd			
Z3-19c	XUHB	CH1 !DAV OUT	J3-15	(H3)	in
Z3-19a	XUDD				
Z3-20c	XUD0	CH1 OUT 0	J3-17	(PB0)	in
Z3-20a	XUD1	CH1 OUT 1	J3-18	(PB1)	in
Z3-21c	XUD2	CH1 OUT 2	J3-19	(PB2)	in
Z3-21a	XUD3	CH1 OUT 3	J3-20	(PB3)	in
Z3-22c	XUD4	CH1 OUT 4	J3-21	(PB4)	in
Z3-22a	XUD5	CH1 OUT 5	J3-22	(PB5)	in
Z3-23c	XUD6	CH1 OUT 6	J3-23	(PB6)	in
Z3-23a	XUD7	CH1 OUT 7	J3-24	(PB7)	in
Z3-24c	XTUVCC	+4.4 vdc			
Z3-24a	XTUGND	gnd			
Z3-25c					
Z3-25a					
Z3-26c	XWHA		J4-16	(H4)	out
Z3-26a	XVWGND	gnd			
Z3-27c	XWHB	CH1 !DAV OUT	J4-15	(H3)	in
Z3-27a	XWDD				
Z3-28c	XWD0	CH1 OUT 8	J4-17	(PB0)	in
Z3-28a	XWD1	CH1 OUT 9	J4-18	(PB1)	in
Z3-29c	XWD2	CH1 OUT 10	J4-19	(PB2)	in
Z3-29a	XWD3	CH1 OUT 11	J4-20	(PB3)	in
Z3-30c	XWD4	CH1 OUT 12	J4-21	(PB4)	in
Z3-30a	XWD5	CH1 OUT 13	J4-22	(PB5)	in
Z3-31c	XWD6	CH1 OUT 14	J4-23	(PB6)	in
Z3-31a	XWD7	CH1 OUT 15	J4-24	(PB7)	in
Z3-32c	XVWVCC	+4.4 vdc			
Z3-32a	XVWGND	gnd			



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

```
"*****************************************************************************
"  FILENAME:    OPIO_DELAY_GEN.ABL      Declarations unique to OPIO delay
"       DATE:   January 31, 1989
"         BY:   Tom Torkelson
"*****************************************************************************

module opio_delay_gen

flag '-r3','-t0'

title 'OPIO FTC Delay Generator EPLD for MC68230

BOEING ADVANCED SYSTEMS
Designed by:  T.C. Torkelson            Latest Revision:  31 JAN 89'

" This module is used with the MC68230 PIT to prevent the timer_register from
" changing when the timer_register is being read.  This module was designed
" with the consideration that the MOVEP instruction must be used to access
" the timer_register on the MC68230.
"
" The first byte is read by the MOVEP instruction is actually a dummy byte
" which is read as zero.  The CS for the dummy byte causes the EPLD to skip
" the next rising edge of the FTC, whether it occurs during the read of the
" timer or not.  The next rising and falling edges each generate a pulse to
" the 68230, making up for the swallowed rising edge.
"
" A limitation on the 68230 is that clock pulses must not be spaced closer
" than the input clock frequency of the chip / 8.  The OPIO 68230 is clocked
" at 8 Mhz, thus the minimum spacing between pulses is 1 usec. This
" works with the 4.125 usec FTC clock.

" declarations

        OPIO_DELAY_GEN device 'E0600';  "uses the Altera EP600 chip

" inputs unique to OPIO_DELAY_GEN
        !INH_1          pin 9;          " TICK inhibit, active low
        !INH_2          pin 10;         " TICK inhibit, active low
        INH_3           pin 11;         " TICK inhibit, active high

" get common code for delay generator

        @INCLUDE 'DELAY_GEN.INC'

end opio_delay_gen
```

```
"*****************************************************************************
"  FILENAME:    DELAY_GEN.INC   FTC pulse delay generator common logic
"       DATE:   January 31, 1989
"         BY:   Art Pannek
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"=============================================================================
"   A   10/31/88        TCT     Placed test vectors separate .TST file
"                               Changed pin allocation for pc board
"                               Changed INHB_A & _B pol to active low
"                               Added INHB_C, active high
"                               Changed pin numbers of inhibits for ISIO
"
"   B    1/30/89        TCT     Changed design of EPLD to always swallow
"                               one rising edge, then make it up with a
"                               falling edge later
"
"   C    1/31/89        TCT     Changed state progression, separated out .abl
"                               code common to ISIO_DELAY_GEN & OPIO_DELAY_GEN
"
"   D    2/ 2/89        TCT     Changed FTC latch to FTC D flop clocked
"                               async by falling edge of CLK3
"*****************************************************************************


    " define ABEL .. commands
        C, K, P, X = .C., .K., .P., .X.;
        H, L = 1, 0;

    " inputs

        CLK1            pin  1;         " MC68230 clock
        CLK2            pin 13;         " MC68230 clock
        CLK3            pin 23;         " MC68230 clock

        !CS             pin  2;         " CS active low to select MC68230

        RS1             pin  4;         " MC68230 register select bits
        RS2             pin  5;
        RS3             pin  6;
        RS4             pin  7;
        RS5             pin  8;

        FTC             pin 14;         "Fault Tolerant Clock; 8 MHz / 33

    " outputs

        FTC_TICK        pin  3;         " Tick output to 68230

        CNTRX_SELECT    pin 22;         " CS of cntrx register detected
            CNTRX_SELECT        istype  'pos, reg_D, feed_reg';
            CNTRX_SELECT.C      istype  'eqn';  " async clock
            CNTRX_SELECT.AR     istype  'eqn';  " async reset

        CNTRX_SELECT_LATCH      pin 21;
            CNTRX_SELECT_LATCH  istype  'pos, com, feed_pin';

        FTC_LATCH       pin 20;
" -- Rev D  TCT 2/2/89
            FTC_LATCH           istype  'pos, reg_D, feed_reg';
            FTC_LATCH.C         istype  'eqn';
```

```
"           FTC_LATCH           istype  'pos, com, feed_pin';

        FTC_LATCH_DELAY pin 19;
            FTC_LATCH_DELAY     istype  'pos, reg_D, feed_reg';

        SKIP0, SKIP1    pin 17, 18;
            SKIP0, SKIP1        istype  'pos, reg_D, feed_reg';

        INH_LATCH       pin  16;
            INH_LATCH           istype  'pos, com, feed_pin';

        TICK            pin  15;        " outputs pulses w/o regard to INH
            TICK                istype  'pos, reg_D, feed_reg';

    " define states

        rs = [RS5..RS1];        " input register select
            RS_CNTRX = ^b10110;         " select dummy
            RS_CNTRH = ^b10111;         " select high byte
            RS_CNTRM = ^b11000;         " select middle byte
            RS_CNTRL = ^b11001;         " select low byte

        inh = [INH_1,INH_2,INH_3];      " inhibit
            TICK_ENABLE = ^b000;

        clk = [CLK1,CLK2,CLK3];         "Clock the same inputs
            CLK_C = [C,C,C];              "Clk_Group is Clocked
            CLK_H = [H,H,H];              "Clk_Group is High
            CLK_L = [L,L,L];              "Clk_Group is Low

        ftc = [FTC_LATCH, FTC_LATCH_DELAY];
            FTC_RISE_EDGE = ^b10;          " rising edge of FTC
            FTC_FALL_EDGE = ^b01;          " falling edge of FTC

        skip = [SKIP1,SKIP0];           " FTC edge skip states
            SKIP_RESET    = ^b00;           " pass + edges
            SKIP_INHIBIT = ^b01;            " inhibit all
            SKIP_PASS_HI = ^b11;            " pass + edge
            SKIP_PASS_LO = ^b10;            " pass - edge

    " macros

        " latch on gate level, pass thru on !gate level

        LATCH macro (out, in, gate)
            {?out = ?out & ?gate # ?in & !?gate;}

equations

    CNTRX_SELECT    := (rs == RS_CNTRX);
    CNTRX_SELECT.C  = CS;               " clock on leading edge of CS

    " The following is really not required unless only one CS is received.
    CNTRX_SELECT.AR = (skip == SKIP_PASS_LO) & !FTC_LATCH & FTC_LATCH_DELAY;

    " synchronize with input clock, hold when clock low, pass clock high

    LATCH (CNTRX_SELECT_LATCH, CNTRX_SELECT, !CLK3)

" -- Rev D  TCT  2/2/89
"    LATCH (FTC_LATCH, FTC, !CLK3)
```

```
    LATCH (INH_LATCH, (INH_1 # INH_2 # INH_3), !CLK3)

" -- Rev D TCT  2/2/89
    FTC_LATCH  := FTC;
    FTC_LATCH.C = !CLK3;        " clock on falling edge of CLK3

    " FTC delayed one input clock pulse
    FTC_LATCH_DELAY := FTC_LATCH;

    " FTC tick conditioned by skip states and inhibits
    FTC_TICK := !INH_LATCH &
                (   (skip == SKIP_RESET)   & (ftc == FTC_RISE_EDGE)
                  # (skip == SKIP_PASS_HI) & (ftc == FTC_RISE_EDGE)
                  # (skip == SKIP_PASS_LO) & (ftc == FTC_FALL_EDGE)
                );

    " TICK output for test purposes, not affected by INH

    TICK :=       (skip == SKIP_RESET)   & (ftc == FTC_RISE_EDGE)
                # (skip == SKIP_PASS_HI) & (ftc == FTC_RISE_EDGE)
                # (skip == SKIP_PASS_LO) & (ftc == FTC_FALL_EDGE)
                ;

state_diagram   skip

    " This state machine is clocked by the system clock.  After the
    " initial state change, state changes only occur on edges of FTC.
    "
    " The state machine inhibits an output pulse on the first rising
    " edge following the selection of CNTRX.  The second rising edge
    " and the following falling edge both generate output pulses.

    state SKIP_RESET:   if CNTRX_SELECT_LATCH then SKIP_INHIBIT
                        else SKIP_RESET;

    state SKIP_INHIBIT: if (ftc == FTC_RISE_EDGE) then SKIP_PASS_HI
                        else SKIP_INHIBIT;

    state SKIP_PASS_HI: if (ftc == FTC_RISE_EDGE) then SKIP_PASS_LO
                        else SKIP_PASS_HI;

    state SKIP_PASS_LO: if (ftc == FTC_FALL_EDGE) then SKIP_RESET
                        else SKIP_PASS_LO;
```

ABEL(tm) 3.00b - Document Generator 02-Feb-89 05:42 PM
OPIO FTC Delay Generator EPLD for MC68230

BOEING ADVANCED SYSTEMS
Designed by: T.C. Torkelson Latest Revision: 31 JAN 89
Equations for Module opio_delay_gen

Device OPIO_DELAY_GEN

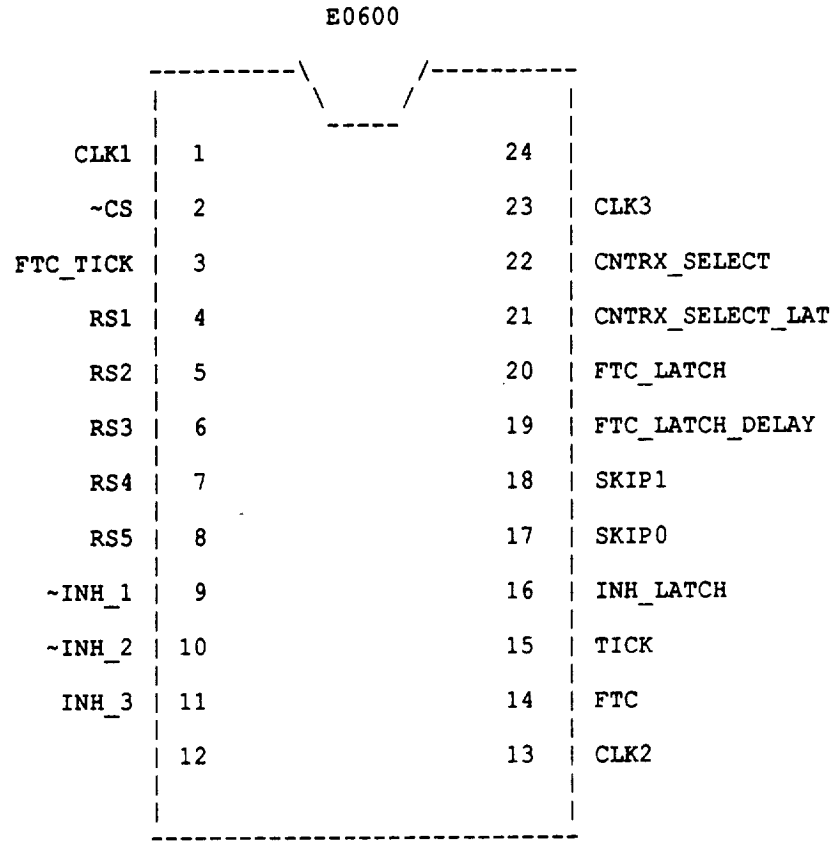
- Reduced Equations:

```
    CNTRX_SELECT := (!RS1 & RS2 & RS3 & !RS4 & RS5);

    _CNTRX_SELECT_C = (!~CS);

    _CNTRX_SELECT_RE = (!FTC_LATCH & FTC_LATCH_DELAY & !SKIP0 & SKIP1);

    CNTRX_SELECT_LATCH = (CLK3 & CNTRX_SELECT # !CLK3 & CNTRX_SELECT_LATCH);

    INH_LATCH = (CLK3 & INH_3
                 # CLK3 & !~INH_2
                 # CLK3 & !~INH_1
                 # !CLK3 & INH_LATCH);


    FTC_LATCH := (FTC);

    _FTC_LATCH_C = (!CLK3);

    FTC_LATCH_DELAY := (FTC_LATCH);

    FTC_TICK := (!FTC_LATCH & FTC_LATCH_DELAY & !INH_LATCH & !SKIP0 & SKIP1

                 # FTC_LATCH & !FTC_LATCH_DELAY & !INH_LATCH & SKIP0 & SKIP1

                 # FTC_LATCH & !FTC_LATCH_DELAY & !INH_LATCH & !SKIP0 &
                !SKIP1);

    TICK := (!FTC_LATCH & FTC_LATCH_DELAY & !SKIP0 & SKIP1
             # FTC_LATCH & !FTC_LATCH_DELAY & SKIP0 & SKIP1
             # FTC_LATCH & !FTC_LATCH_DELAY & !SKIP0 & !SKIP1);


    SKIP1 := (!FTC_LATCH_DELAY & SKIP1
              # FTC_LATCH & SKIP1
              # SKIP0 & SKIP1
              # FTC_LATCH & !FTC_LATCH_DELAY & SKIP0);


    SKIP0 := (FTC_LATCH_DELAY & SKIP0
              # !FTC_LATCH & SKIP0
              # SKIP0 & !SKIP1
              # CNTRX_SELECT_LATCH & !SKIP1);
```

ABEL(tm) 3.00b - Document Generator 02-Feb-89 05:42 PM
OPIO FTC Delay Generator EPLD for MC68230

BOEING ADVANCED SYSTEMS
Designed by: T.C. Torkelson Latest Revision: 31 JAN 89
Chip diagram for Module opio_delay_gen

Device OPIO_DELAY_GEN



end of module opio_delay_gen

DOCUMENTATION PACKAGE C: VMEBUS-MICROVAX PARALLEL INTERFACE ADAPTER

Subject: Fabrication of VMEbus/uVAX Interface
By: T.C. Torkelson

Date: June 14, 1989

Introduction:

The interconnection of the DEC uVAX II DRQ3B and the Force VMEbus system OPIO-1 requires a few mechanical and electrical adaptations: a connector panel must be produced which adapt the DEC interconnect cables to the Force OPIO-1 board; modifications are also required to the Force OPIO-1 card; an additional protocol conversion board must be produced.

Mechanical:

Instead of the VME chassis mounted connector panel, a rear mounted rack panel is used. This requires routing longer 64 conductor ribbon cables from the OPIO-1 Z2 and Z3 connectors to the back of the equipment rack.

These cables should be as short as possible to reach the adapter panel without undue strain. They must be long enough to allow connection to the OPI0-1 card while it is out of the VME chassis.

The cables are routed up through the top of the VME chassis, between two card guides, then into the adapter chassis.

Adapter Board:

The adapter board is a small vector board adapter. Its primary function is the correct interconnection of the OPIO-1 and DRQ3B. It also corrects some of the protocol problems discussed elsewhere.

The board is mounted behind the DEC compatible I/O connectors on standoffs. No unusual precautions other than standard shop practices need to be observed.

PRECEDING PAGE BLANK NOT FILMED

PAGE J-22 INTENTIONALLY BLANK

Subject: Test program to verify OPIO-1 operation
Date: January 5, 1989
By: T.C. Torkelson

To test the OPIO-1 card, the FAST_OPIO.ABS program must first be downloaded to the CPU-29 card using VMEPROM and VAX VMS DCL commands.

Commands on the VAX side are preceded by "$". The commands on the VMEPROM side are preceded by "?".

Initial setup:

```
$ ALLOCATE LTA4:
$ SET TERM/HOSTSYNC LTA4:

? BP D02 1 1
```

Actual program transfer:

```
? LO <2

$ COPY FAST_OPIO.ABS LTA4:
```

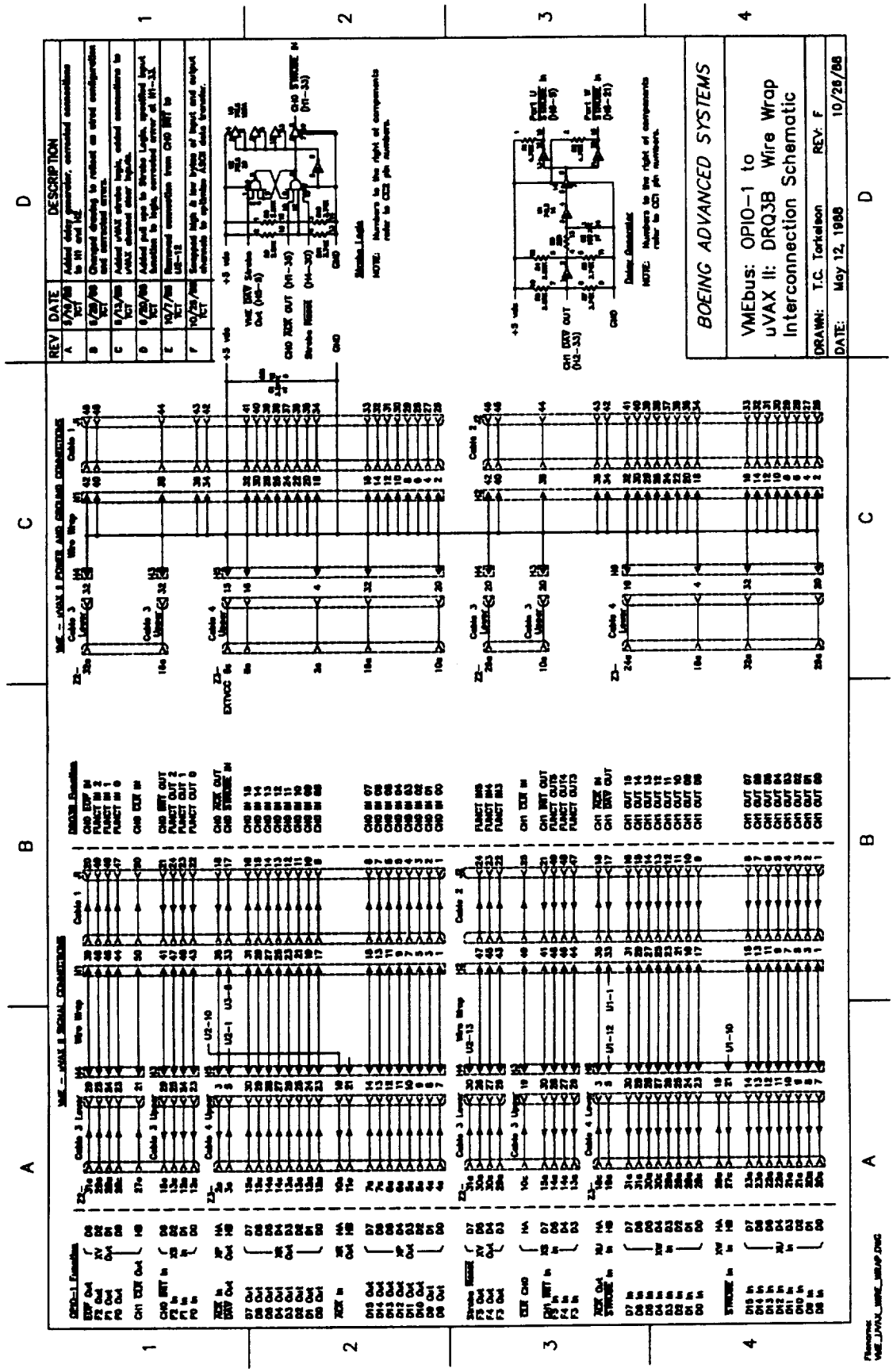
To run the test program, the loop back cable must be installed to jumper the two ports on the VME to uVAX interface box to each other.

The VMEPROM command to run the program:

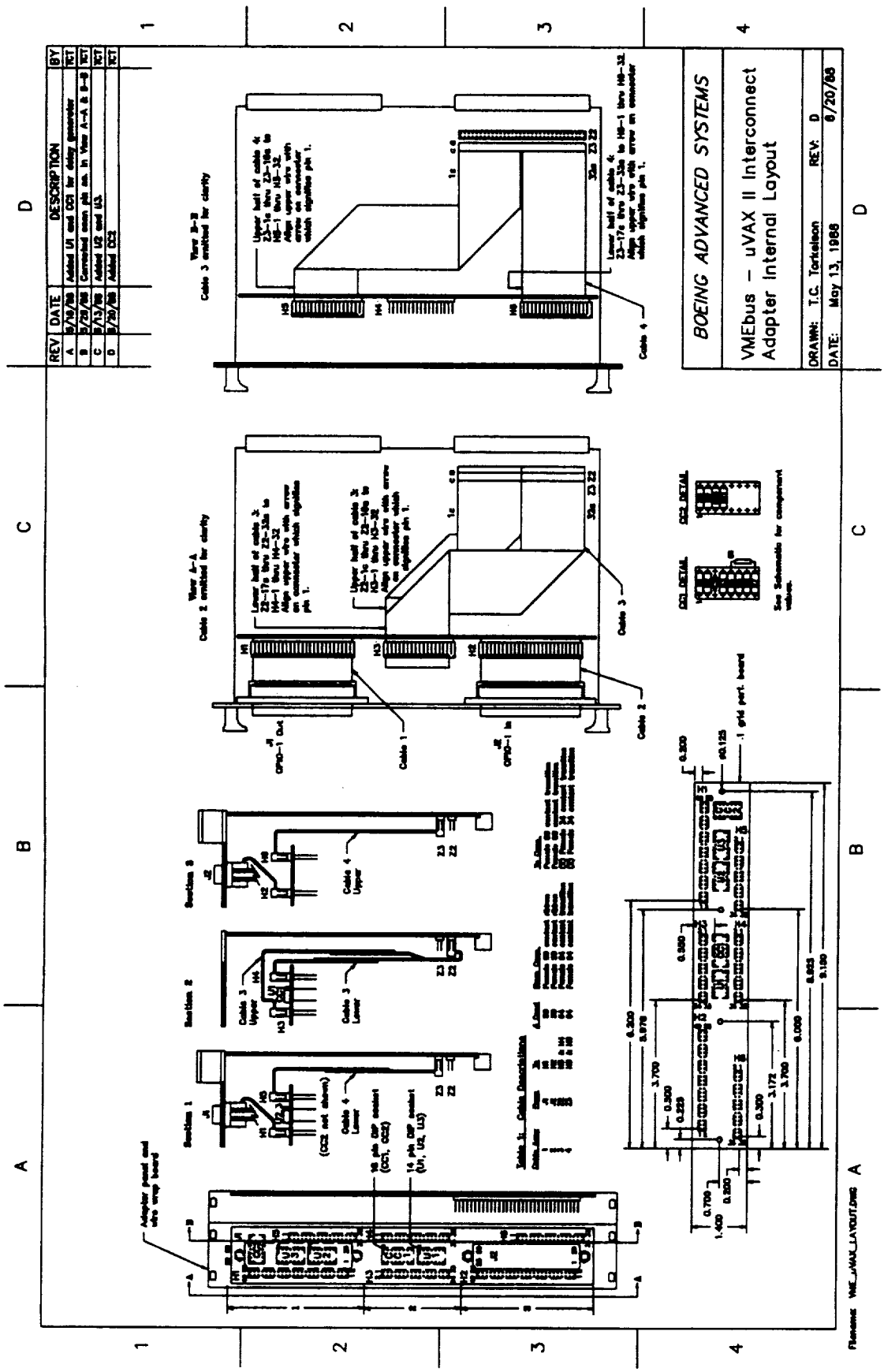
```
? GO 8000
?
```

At the return prompt, receive buffer in memory at 20000 through 2FFFF should be byte swapped from transmit buffer in memory at 10000 through 1FFFF.

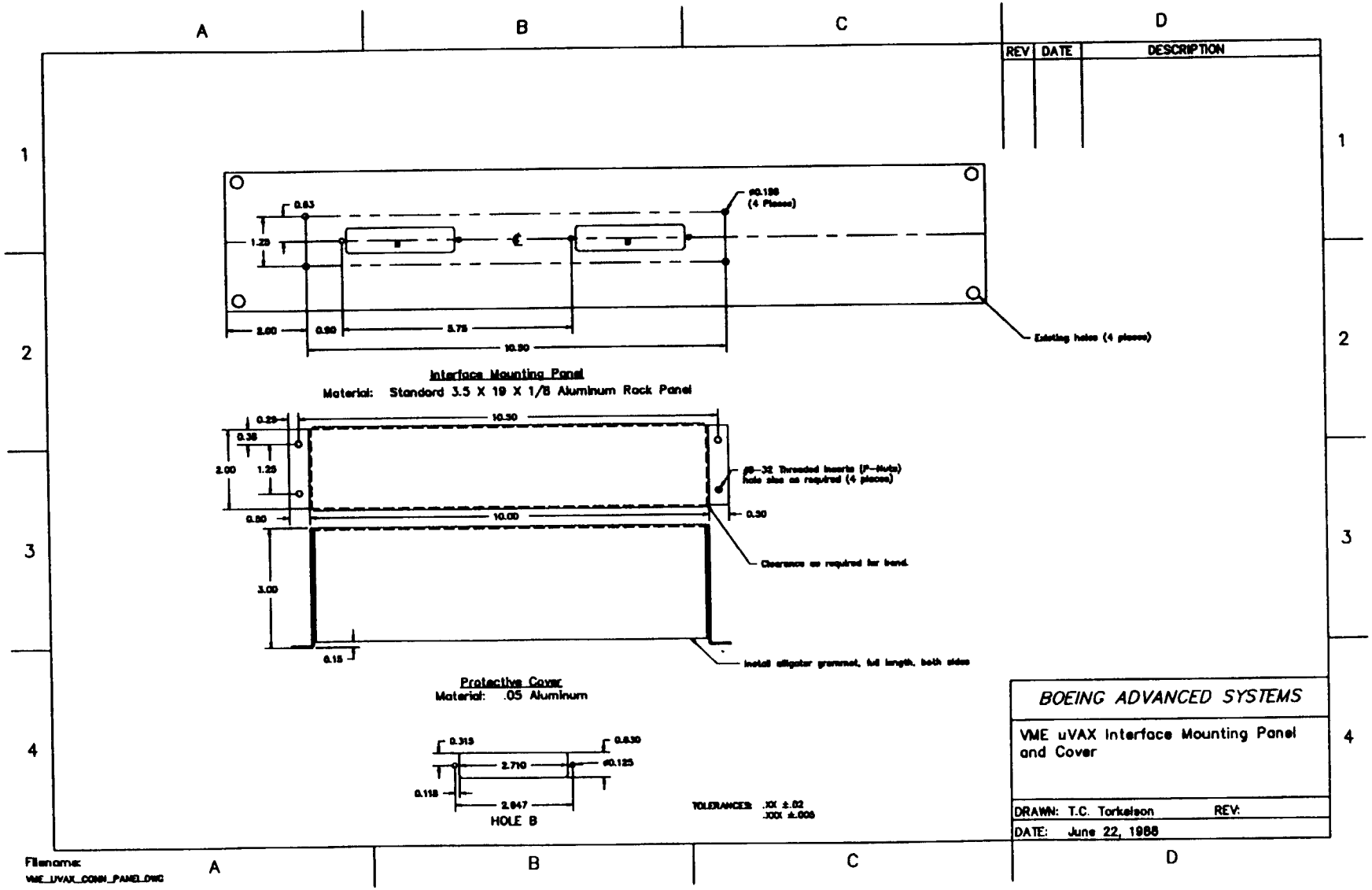
After setting



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

DOCUMENTATION PACKAGE D: ISIO-2/DIU SIMULATOR DAUGHTER BOARD

Subject: ISIO-2 Modifications for IAPSA DIU Simulation
Date: August 23, 1988
Revised: June 14, 1989

By: T.C. Torkelson

References:
1. IAPSA II DIU Simulator Specifications - VMEbus Implementation
2. AIPS I/O-network Interface Requirements
3. Small Scale System Experiment Start Synchronization
4. Experiment Bus Descripion
5. VMEbus Simulation Computer Addressing

INTRODUCTION

As presented in Reference 2, the VMEbus simulation computer and the FTP must be able to signal each other of their status. The DIU simulator must also be controlled for proper experiment synchronization.

REQUIREMENTS

The DIU simulator as implemented on the ISIO cards does not have access to the VMEbus. Communications with VMEbus masters is through a message exchange protocol using the ISIO dual port RAM.

The presence of a message for the VMEbus master is signalled by a VMEbus interrupt caused by the ISIO. Similarly, the local ISIO CPU can be signalled of the presence of a message from a VMEbus master by a write to a special ISIO address which causes an ISIO local interrupt.

The local CPU must maintain 24 bit experiment time. Each tick represents one FTC tick. The start of experiment time is controlled by the FTP sync line going from !Stop to Run.

The network adapter daughter board is used to interface ISIO hardware to [V_SYNC], [F_SYNC], and [R_FTC] signals which are present on the experiment bus

IMPLEMENTATION

The daughter board for the ISIO-2 attaches to the elevated IC sockets for the 68562 DUSCC chips and the 68230 PIT. The daughter board allows the disconnection of ISIO-2 board signals from chip pins, freeing the chips for special use on the daughter board. Some of the disconnected chip pins from the ISIO_2 are used to connect signals from the experiment bus to the daughter board.

Experiment time is maintained using an added (U20) (the 68230 PIT). The reference FTC on the experiment bus is conditioned by an EP600 EPLD which prevents the 68230 PIT counter from being incremented when the PIT timer is being read or when the FTP sync is at STOP.

The ISIO-2 board uses the TIN pin of the J100 PIT as PC2 for controlling the sysfail function of the ISIO-2. This must be considered in the software for controlling the DIU simulator.

PRECEDING PAGE BLANK NOT FILMED

PAGE J-28 INTENTIONALLY BLANK

ISIO-2 MODS

1. Remove unused ICs and shorting jumpers
   B23-1 thru B38-4
   J57-J66
   J68-J77
   J79-J88
   J90-J99

2. Remove ICS to be moved to daughter board
   J56, J67, J78, J89, J100

3. Set addressing for as required. (See ISIO manual and reference 5.)

4. Add Jumpers

   a. Miscellaneous

From	To	Name
P1-10a	J51-11	16 Mhz VME sys clock

   b. Connections to Daughter Board

From	To	Via	Name
P2-1c	B23-2	J56-34	[F_SYNC]
P2-2c	B23-1	J56-33	[V_SYNC]
P2-17c	B23-3	J56-39	[R_FTC]
J51-9	B23-4	J56-40	16MHz
B41-1	B25-1	J56-16	Intr. vector mode

Daughter Board Mods (See sh 2 of board loading diagram)

Component side of board:

1. Cut the trace to U100-13.

2. Cut the trace to U100-15.

Circuit side of board:

1. Cut trace from U19012 to the feed thru near U17-1 at both ends. connect a 5.5" wire wrap jumper from U19-12 to U7-1.

2. Cut trace from U19-14 at U19-14. Connect a 226 ohm resistor from U19-14 to the trace. Connect a 332 ohm resistor from U19-14 to U19-20.

3. Cut trace from U20-40 to U21-1. Bridge the cut with a 44.2 ohm resistor.

4. Cut traces from U17-4 and U17-5 to ground bus. Connect U17-4 to the feedthru of the trace to U8-2. Connect U17-5 to the feedthru of the trace to U8-23.

5. Install 24 pin screw machine SIP strip sockets and daughter board connection pins at locations U56, U67, U78, U89, and U100.

SIP sockets must be used to allow access for soldering the pins.

To assemble, install the SIP socket for pin 1-24, then install the pins associated with the socket. Next install the socket for 25-48 and associated pins. Installation must be in this order or it will not be possible to solder all the components in the restricted space available.

The use of resistance soldering for pin installation is strongly recommended.

Advanced Interconnections KSA100-79G pins are installed at the following locations:

U56: 1-4, 6-7, 16, 18-24, 25-31, 33-34, 42-43, 45-48

U67: 1-4, 6-7, 18-24, 25-31, 42-43, 45-48
U78: 1-4, 6-7, 18-24, 25-31, 42-43, 45-48
U89: 1-4, 6-7, 18-24, 25-31, 42-43, 45-48

U100: All locations except 14 and 16.

To protect the pins, install two 24 pin screw machine DIP sockets on the bottom of each of the completed pin installations. (These sockets are also be used at final assembly.)

6. Connect daughter board pin 13 to U100-13 and daughter board pin 15 to U100-15.

7. Connect a jumper from the feedthru opposite U100-13 to U100-14.

8. Connect a jumper from the feedthru opposite U100-15 to U100-16.

CAUTION: It is very important that proper static sensitive device handling precautions are observed during all the following operations!

It is especially important that all tools used are grounded. If a screwdriver with an insulated handle is used during installation or removal of the daughter board, it MUST be held in such a way that its metal parts are at the same potential as the person performing the operation.

Daughter Board Installation

1. Install all ICs an resistor networks in the daughter board.

2. Make certain that all the protective 24 pin dip sockets are installed on the daughter board pins.

3. Carefully position the daughter board over the elevated ISIO-2 sockets to which it will mate.

4. Press down uniformly to firmly seat all the contacts in the elevated sockets.

5. Install spacers and #4-40 hardware in the two mounting holes near the P1 and P2 connectors.

6. Plug the ribbon cables from the DIU front panel into J1 and J2 on the daughter board. Check that no parts on the front panel board will either short out or mechanically interfere with the operation of the ISIO-2 board.

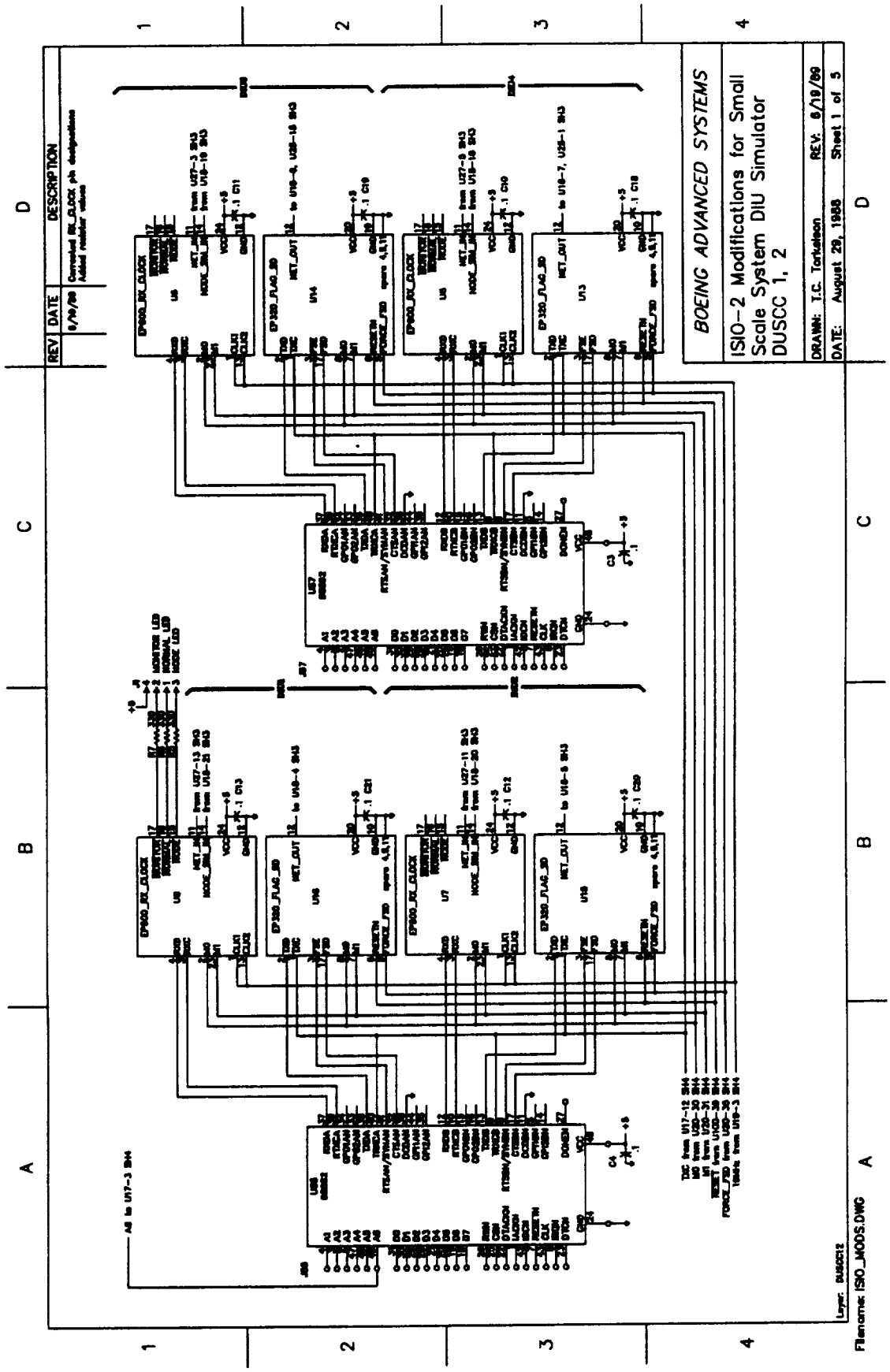
Daughter Board Removal

1. Unplug the DIU front panel board from J1 and J2.

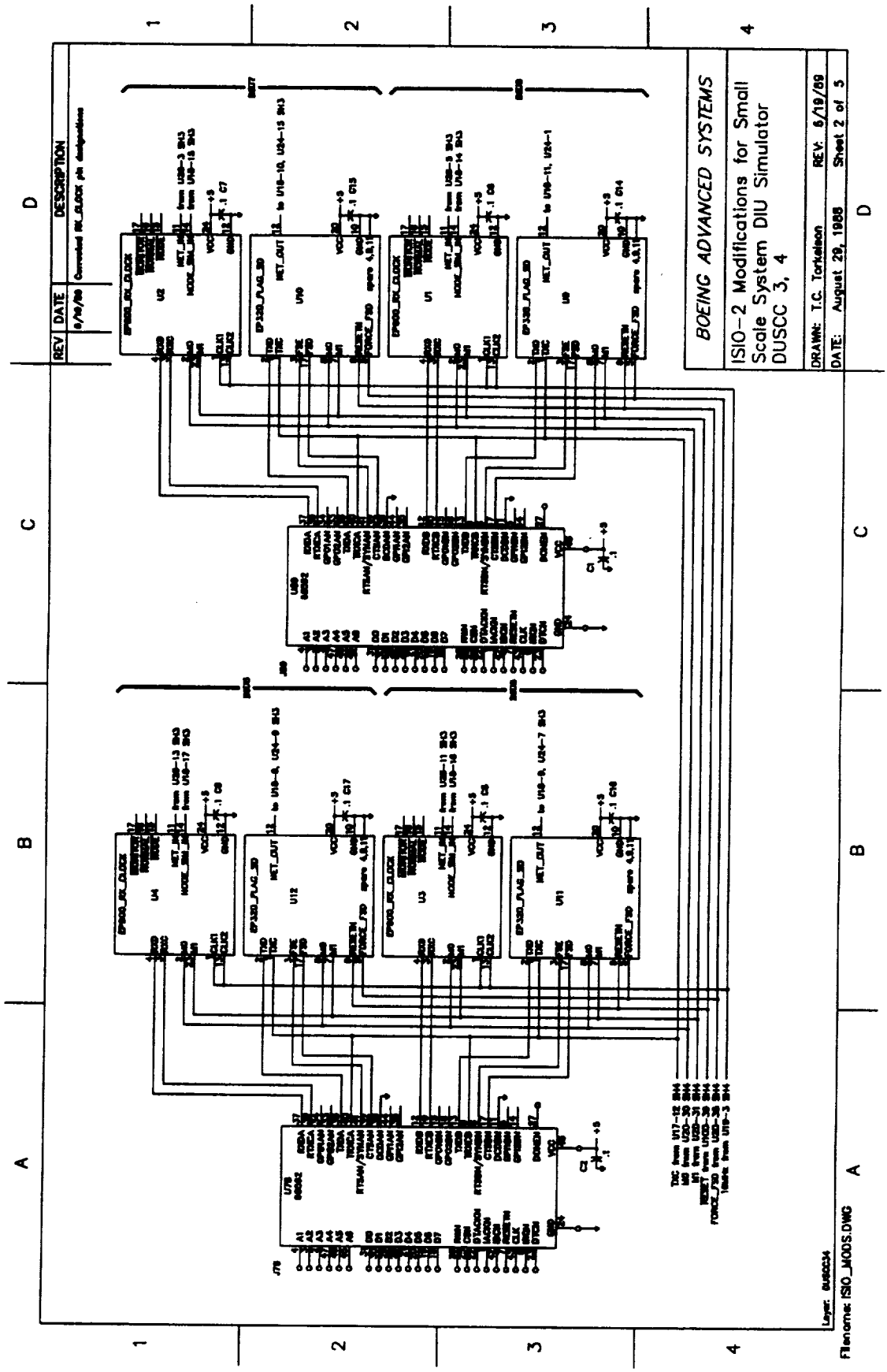
2. Remove the #4-40 hardware holding the two boards together.

3. Carefully pry the daughter board off the elevated ISIO-2 sockets. A large screwdriver can be used for this purpose. Make certain that no components on the ISIO-2 board are in danger of being mechanically damaged and observe the CAUTION, above.

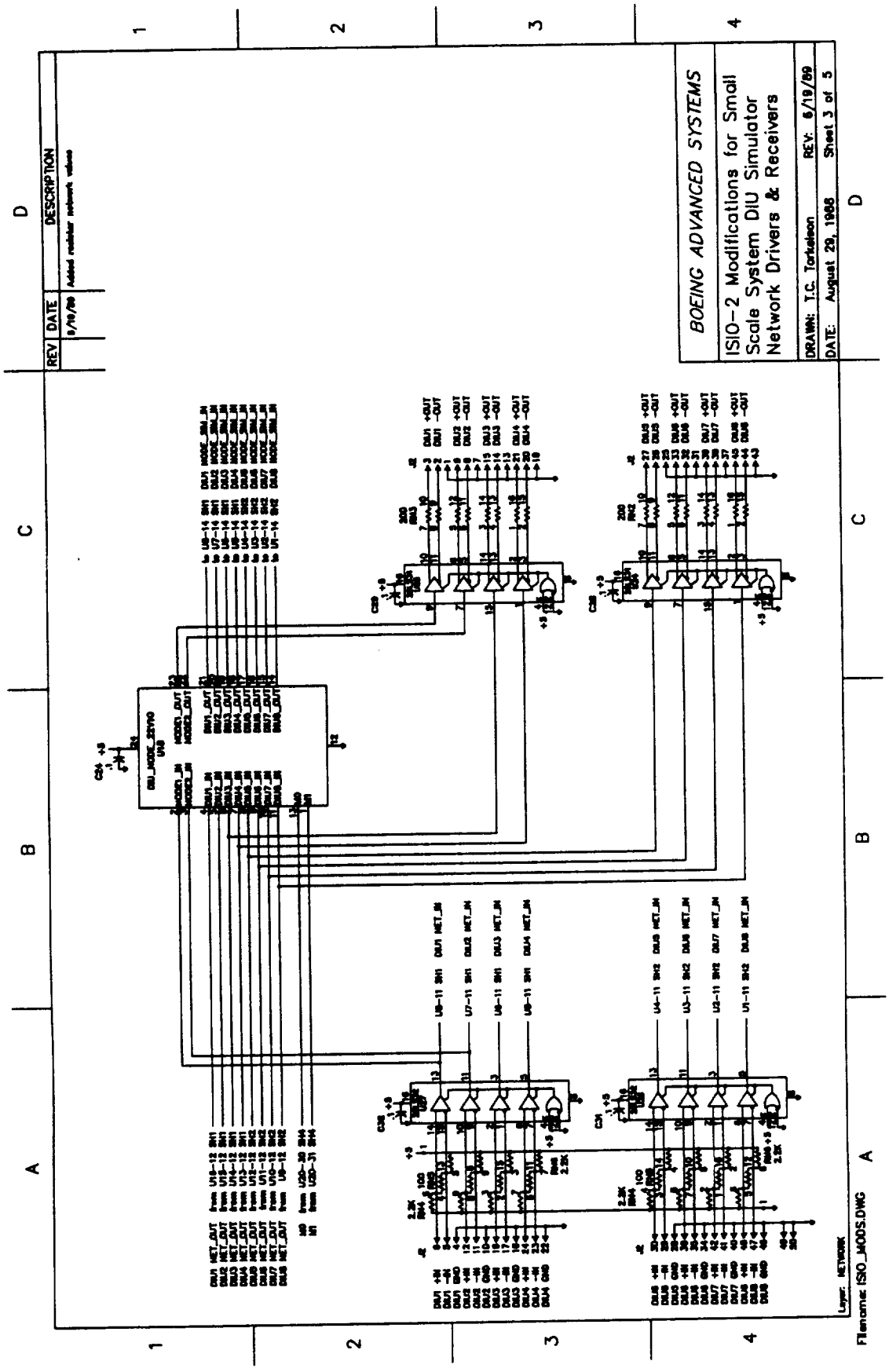
4. Remove any 24 pin DIP sockets which remained stuck in the ISIO-2 elevated sockets and IMMEDIATELY re-install them on the daughter board pins.



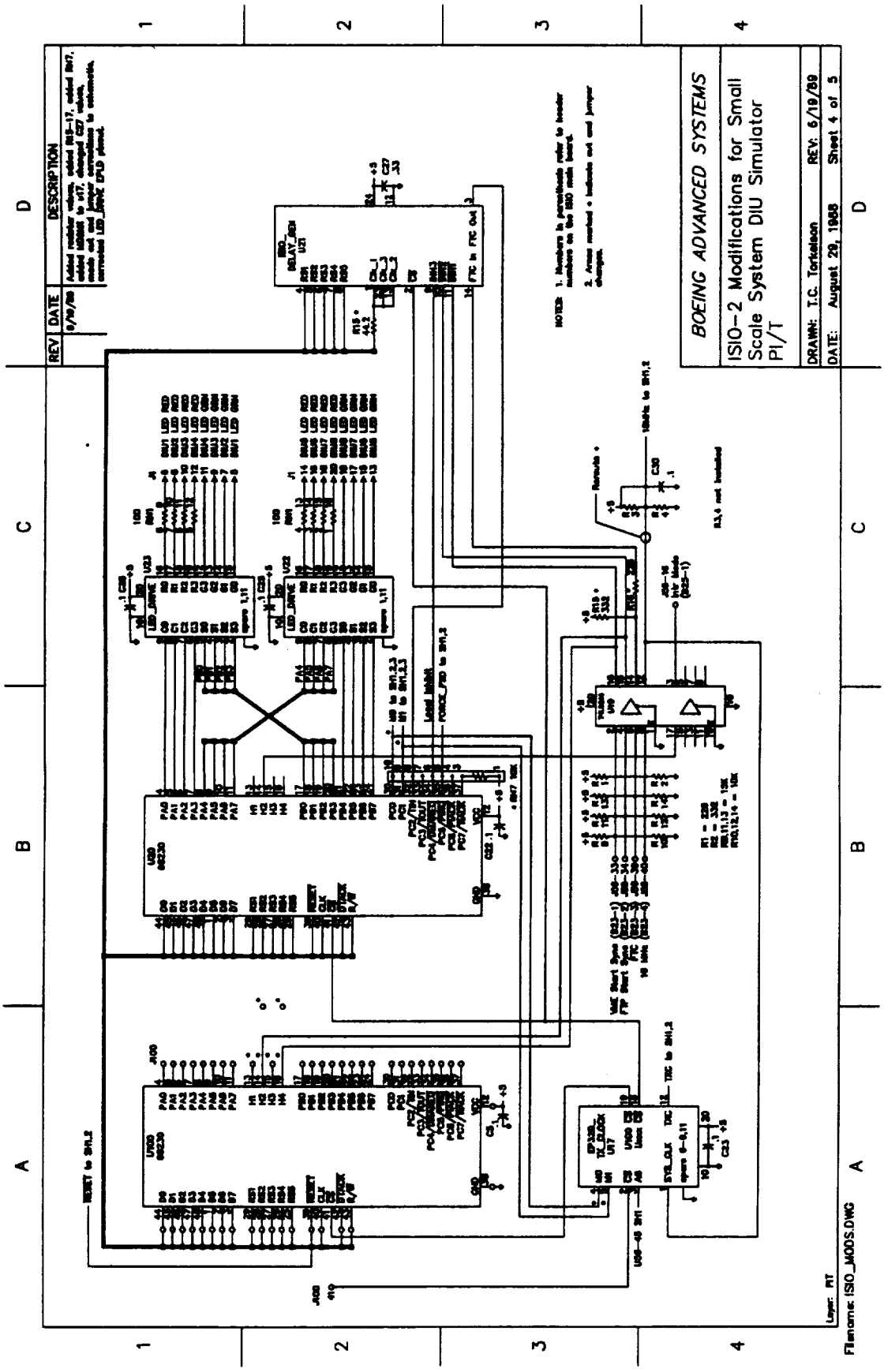
ORIGINAL PAGE IS
OF POOR QUALITY



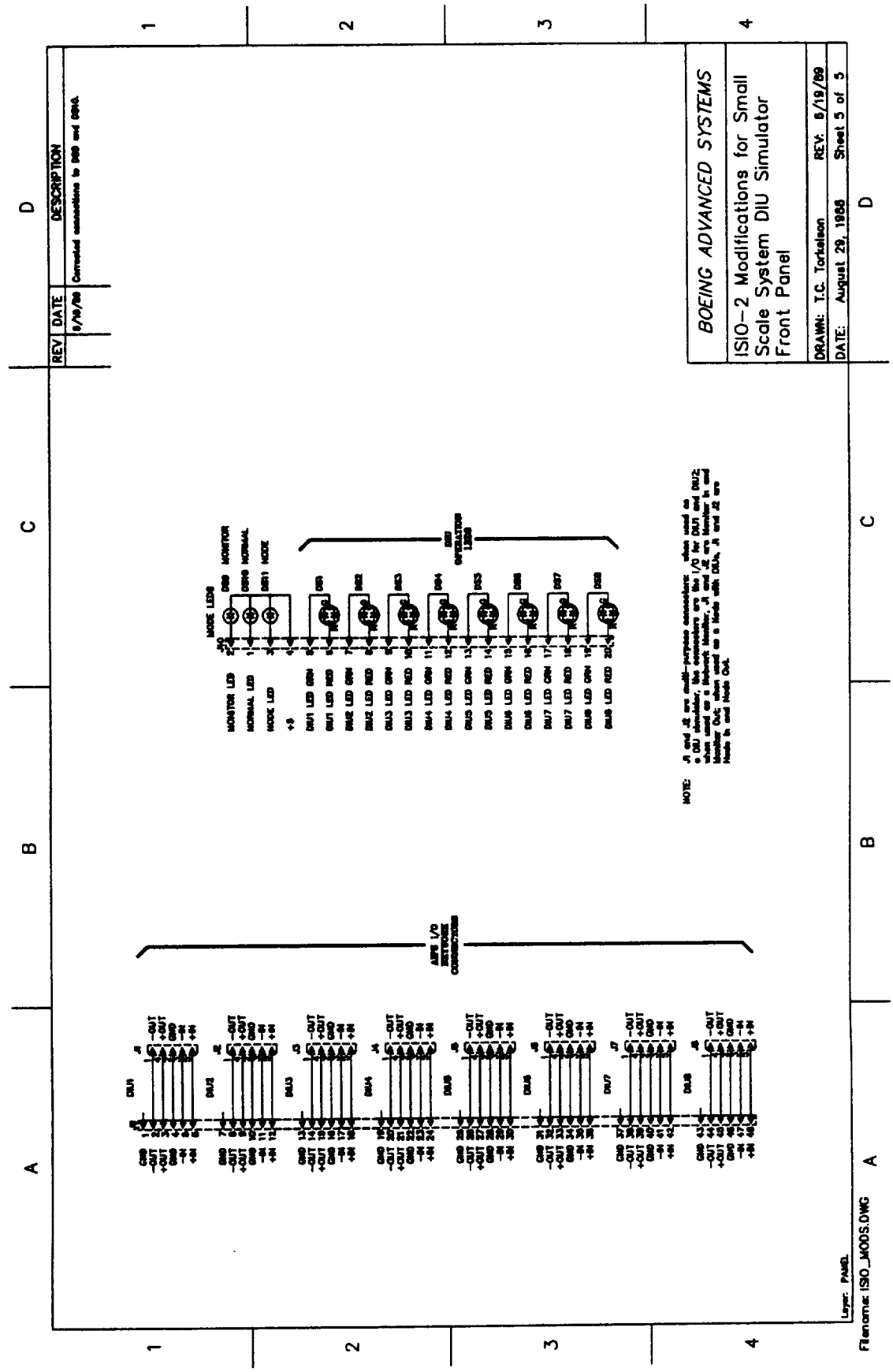
ORIGINAL PAGE IS
OF POOR QUALITY



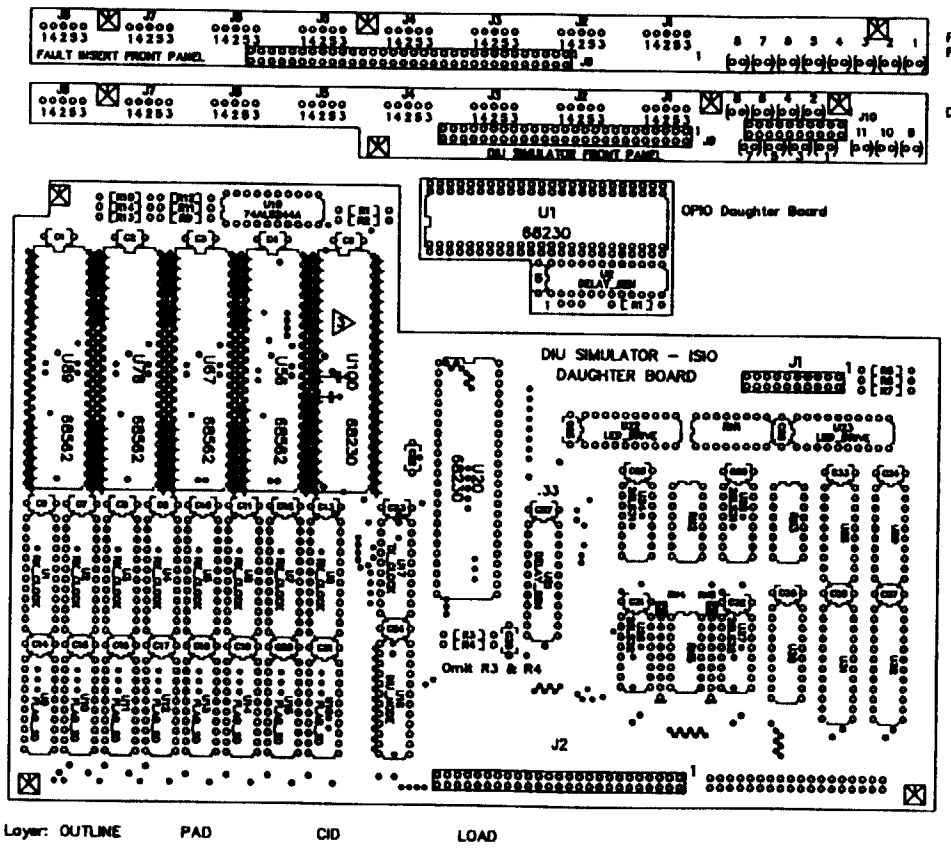
ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



REV	DATE	DESCRIPTION

NOTES:
1. All capacitors are .1 uf monolithic ceramic unless otherwise noted.
2. Make cut and jumper changes on reverse side before installing pins. (See sh 2/2)
3. Cut traces to U100–13 and U100–15 on top of board.
4. Install pins at locations marked + for U56, U67, U78, U89, and U100 for connection to ISIO–2 sockets.
5. Install screw machine sockets for all ICs and SIP resistor networks.
6. Ribbon cable headers are open frame headers. (Headers J1 and J2.)

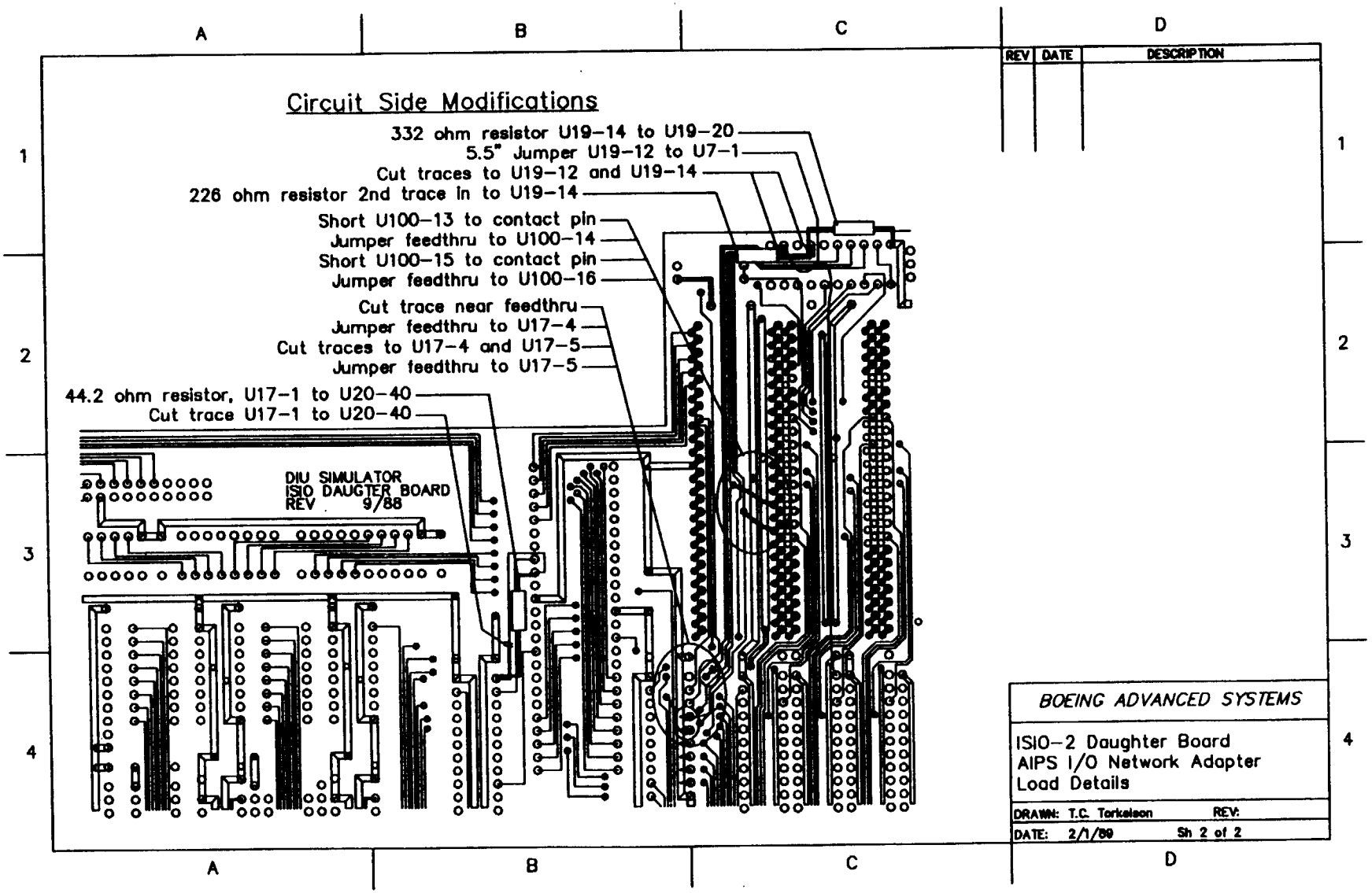
BOEING ADVANCED SYSTEMS

ISIO–2 Daughter Board
AIPS I/O Network Adapter
Load Details

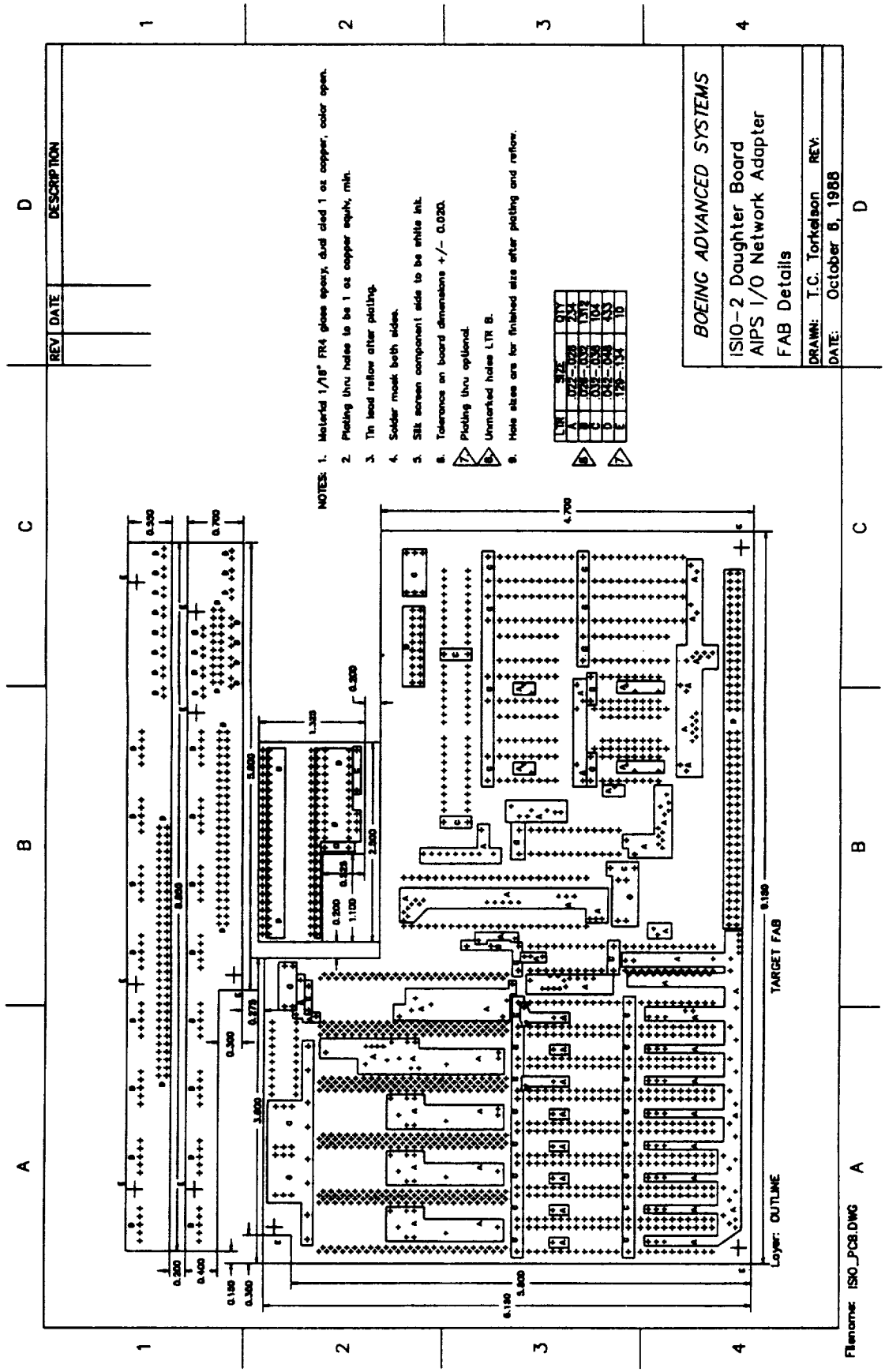
DRAWN: T.C. Torkelson REV:
DATE: October 6, 1988 Sh 1 of 2

Filename: ISIO_PCB.DWG

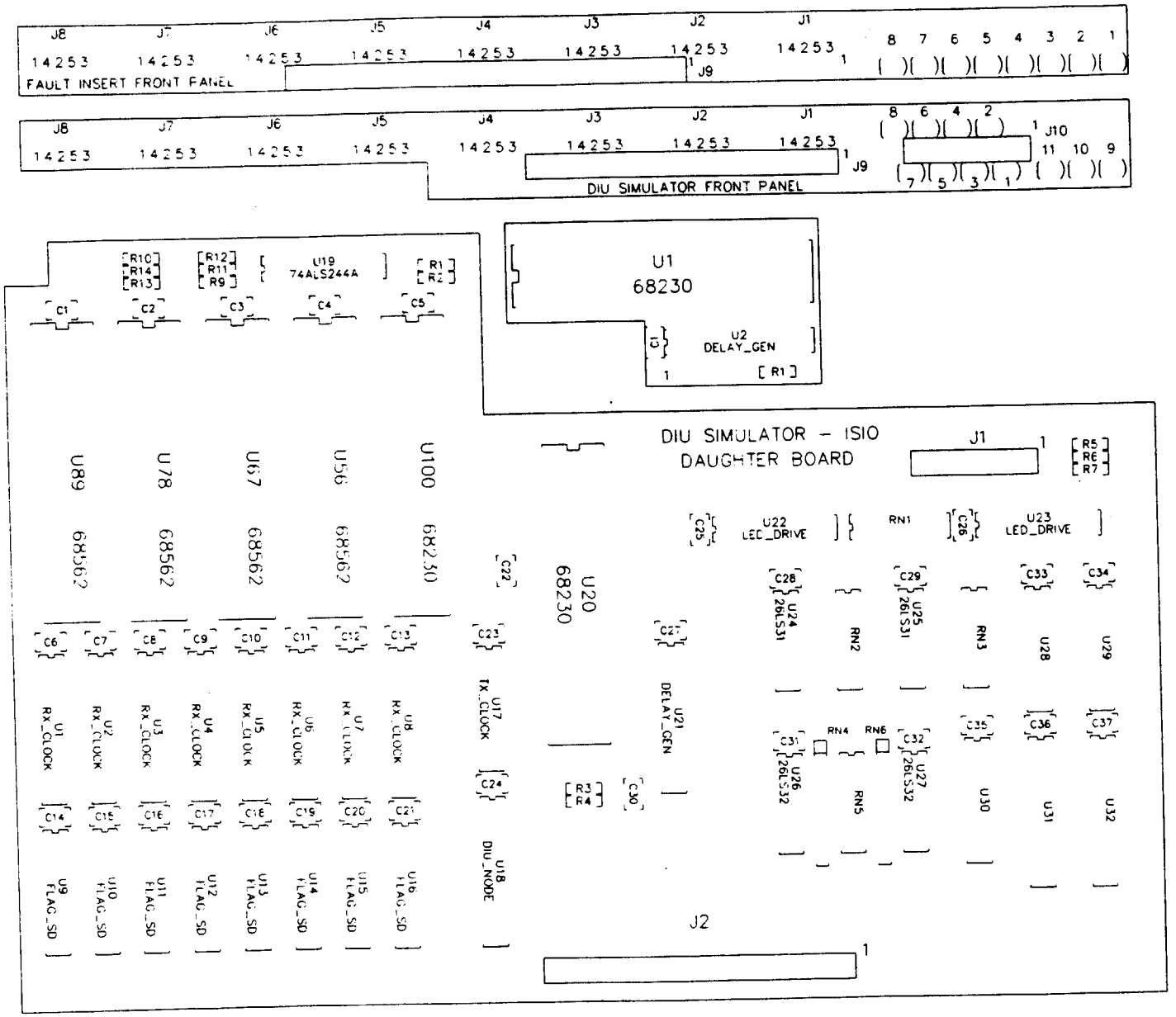
ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

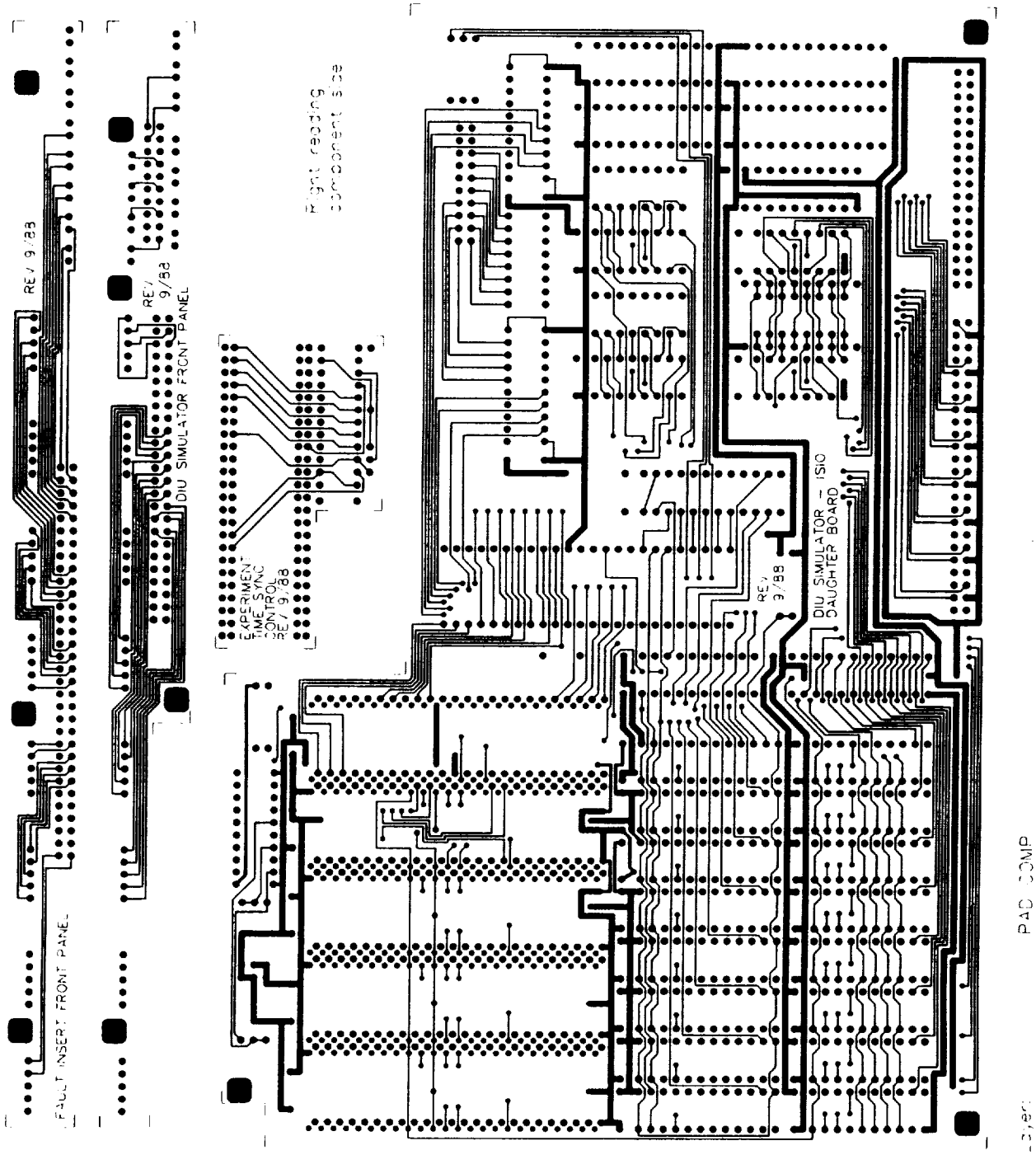


Layer: OUTLINE

CID

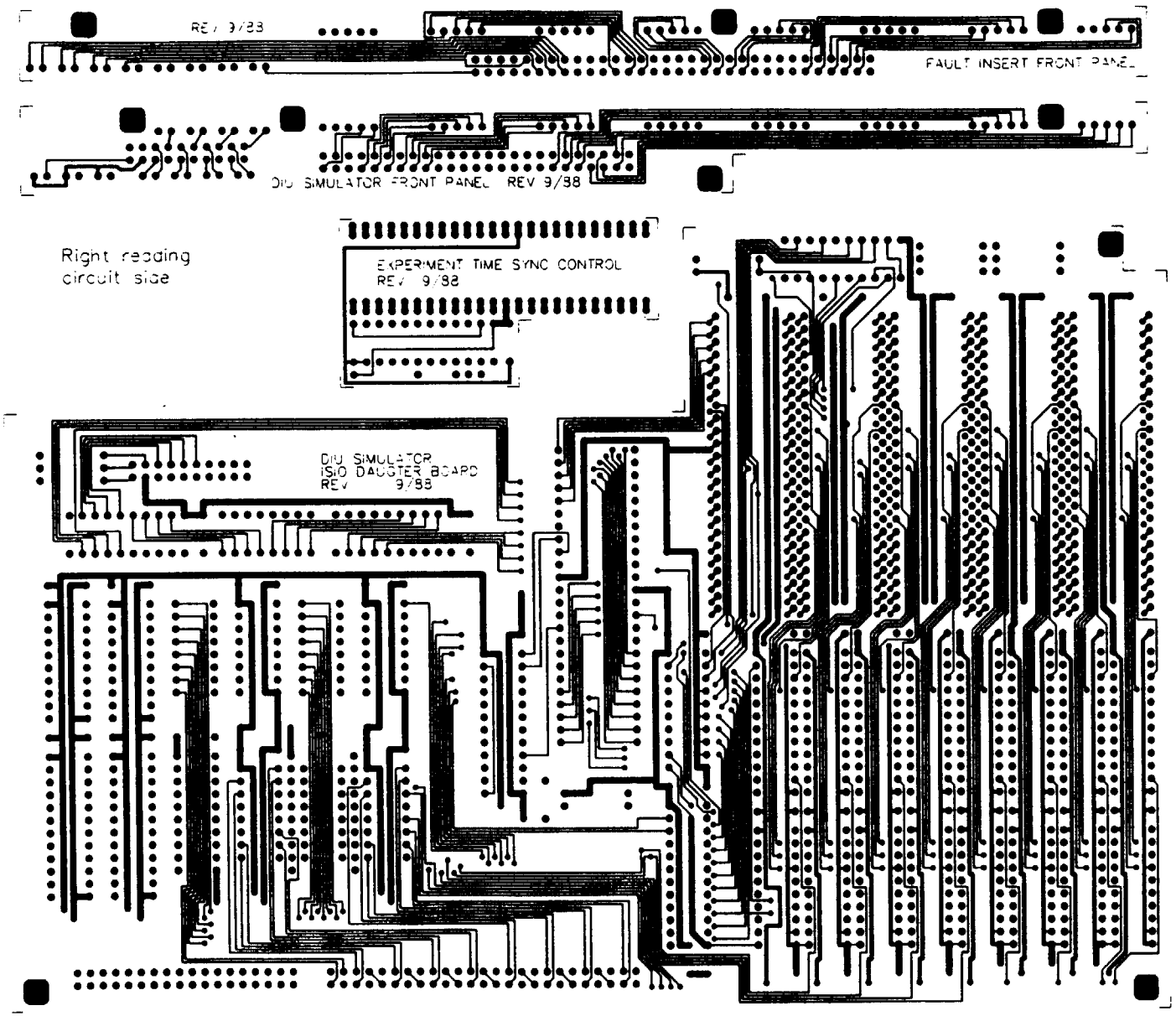
DIU Daughter Board Component ID Film

ORIGINAL PAGE IS
OF POOR QUALITY



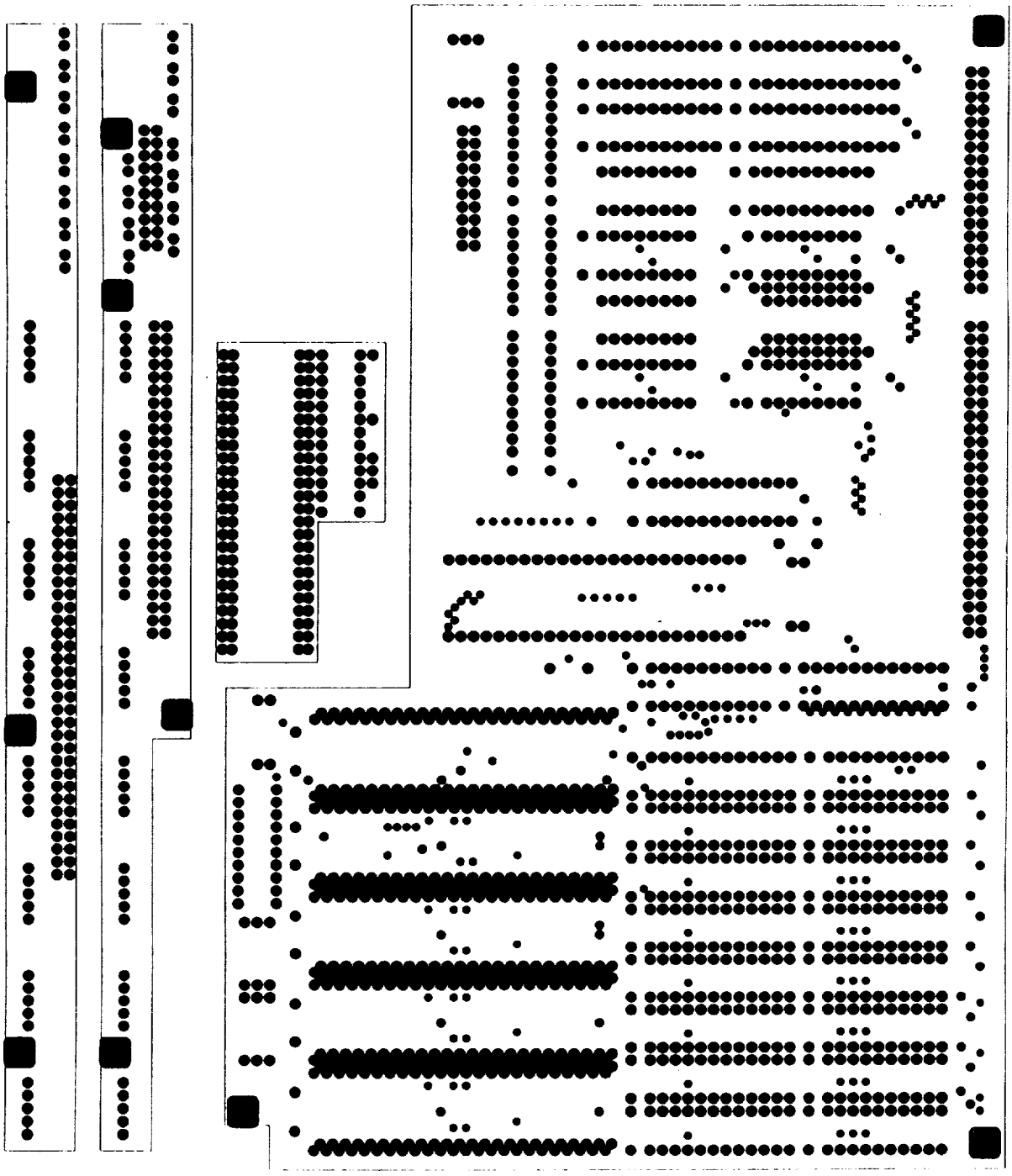
DIU Daughter Board Component Side Film

ORIGINAL PAGE IS
OF POOR QUALITY



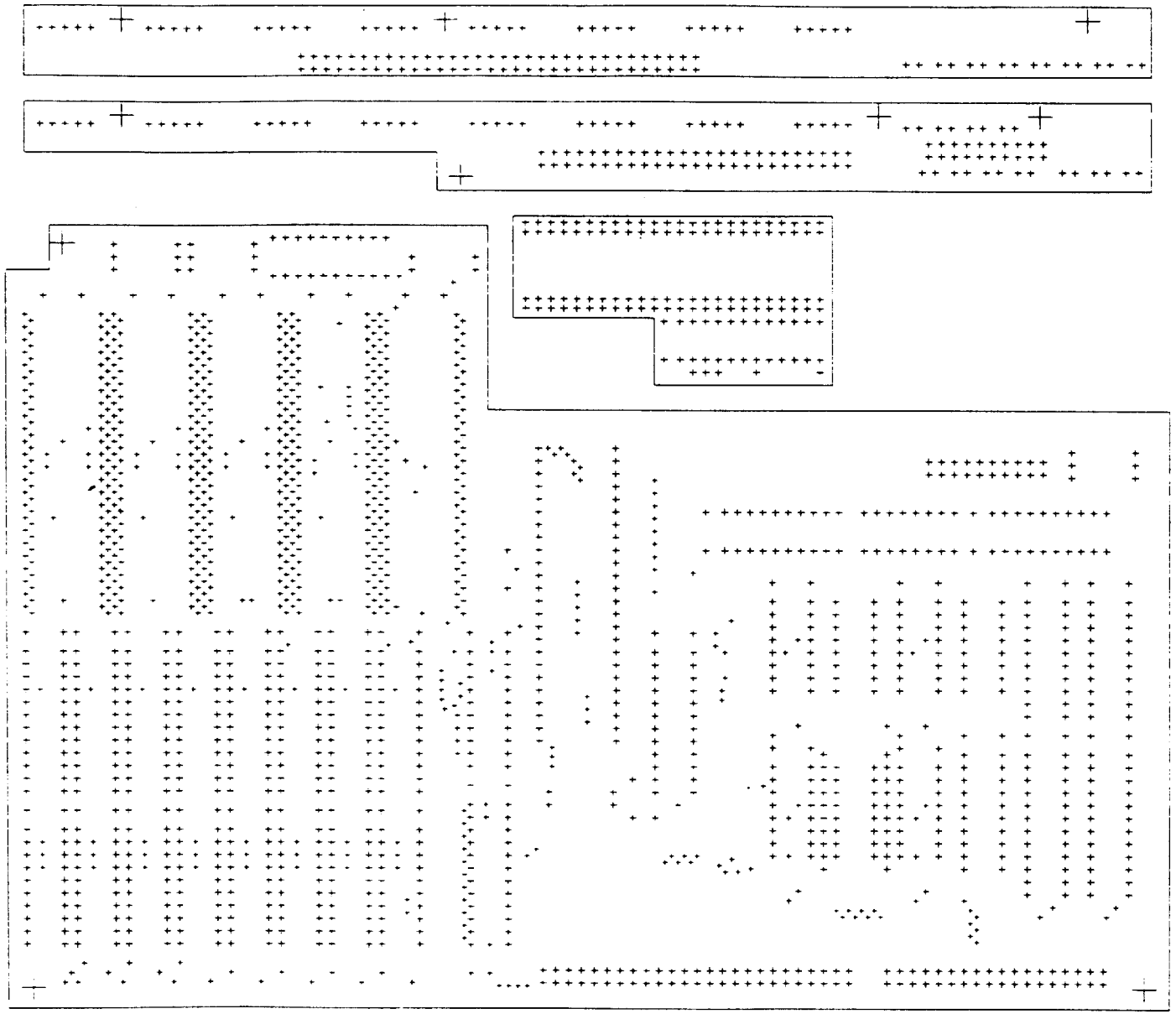
DIU Daughter Board Circuit Side Film

ORIGINAL PAGE IS
OF POOR QUALITY



DIU Daughter Board Solder Mask Film

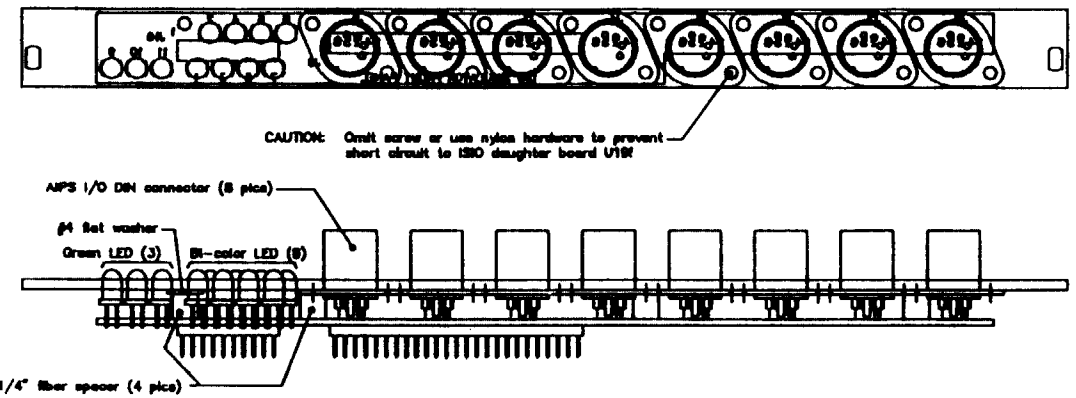
ORIGINAL PAGE IS
OF POOR QUALITY



DIU Daughter Board Drill Target Film

ORIGINAL PAGE IS
OF POOR QUALITY

REV	DATE	DESCRIPTION



NOTES:
1. Solder short lengths of bus wire to DIN connectors before assembly.
2. Form wires into holes in PC board then temporarily attach board to front panel sheet metal to properly space the connectors. When connector spacing is correctly set, solder connectors to PC board.
3. LED spacing should allow roughly 1/16" projection in front of front panel.
4. Note flat of LED for installation.
5. Use standard #4-40 hardware to affix DIN connectors and PC board to front panel. Locking hardware should be used.

Layer: 0 HOLE ASSY

Filename: DIU_FRONT_PANEL.DWG

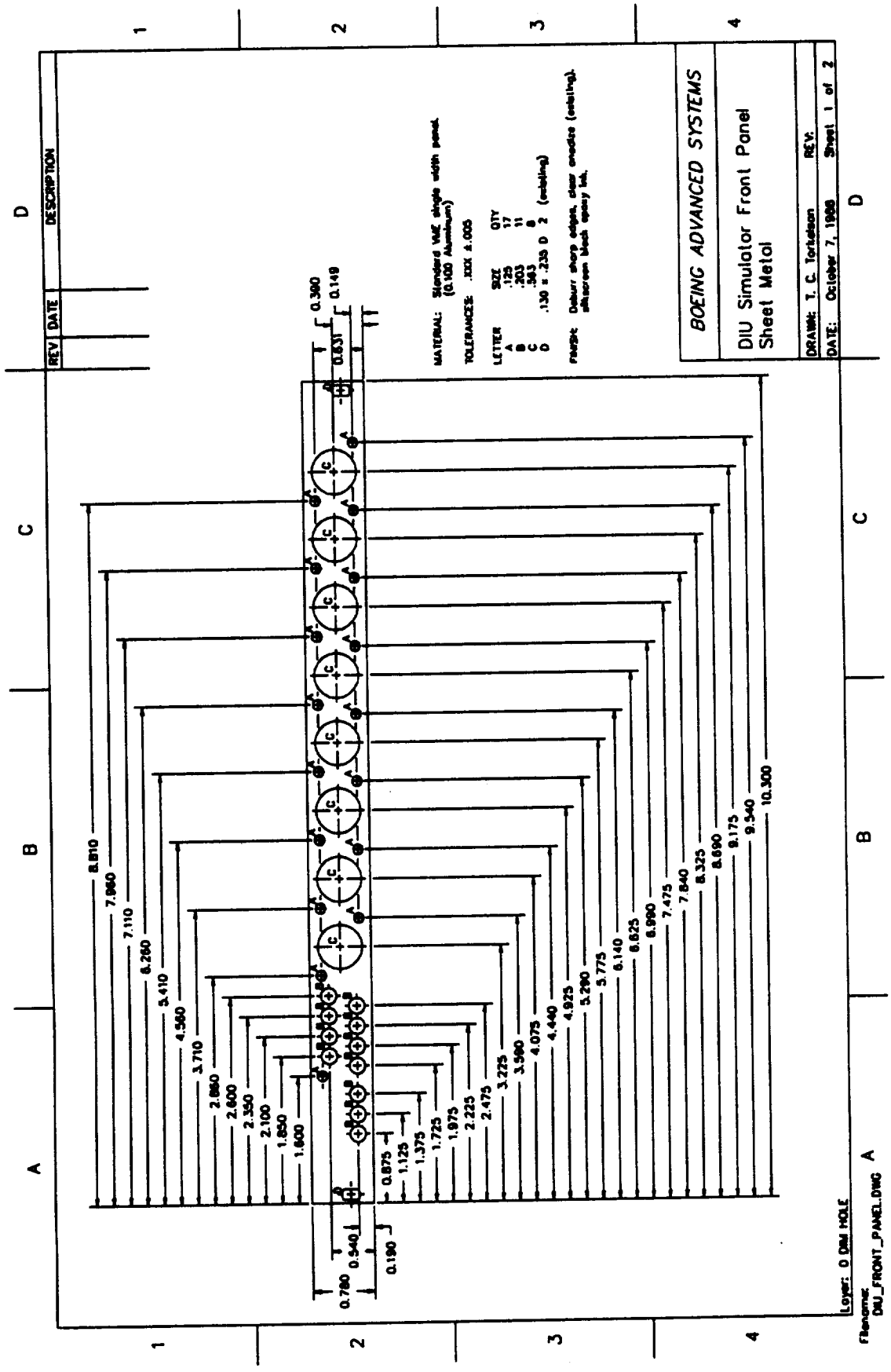
BOEING ADVANCED SYSTEMS

DIU Simulator Front Panel Assembley details.

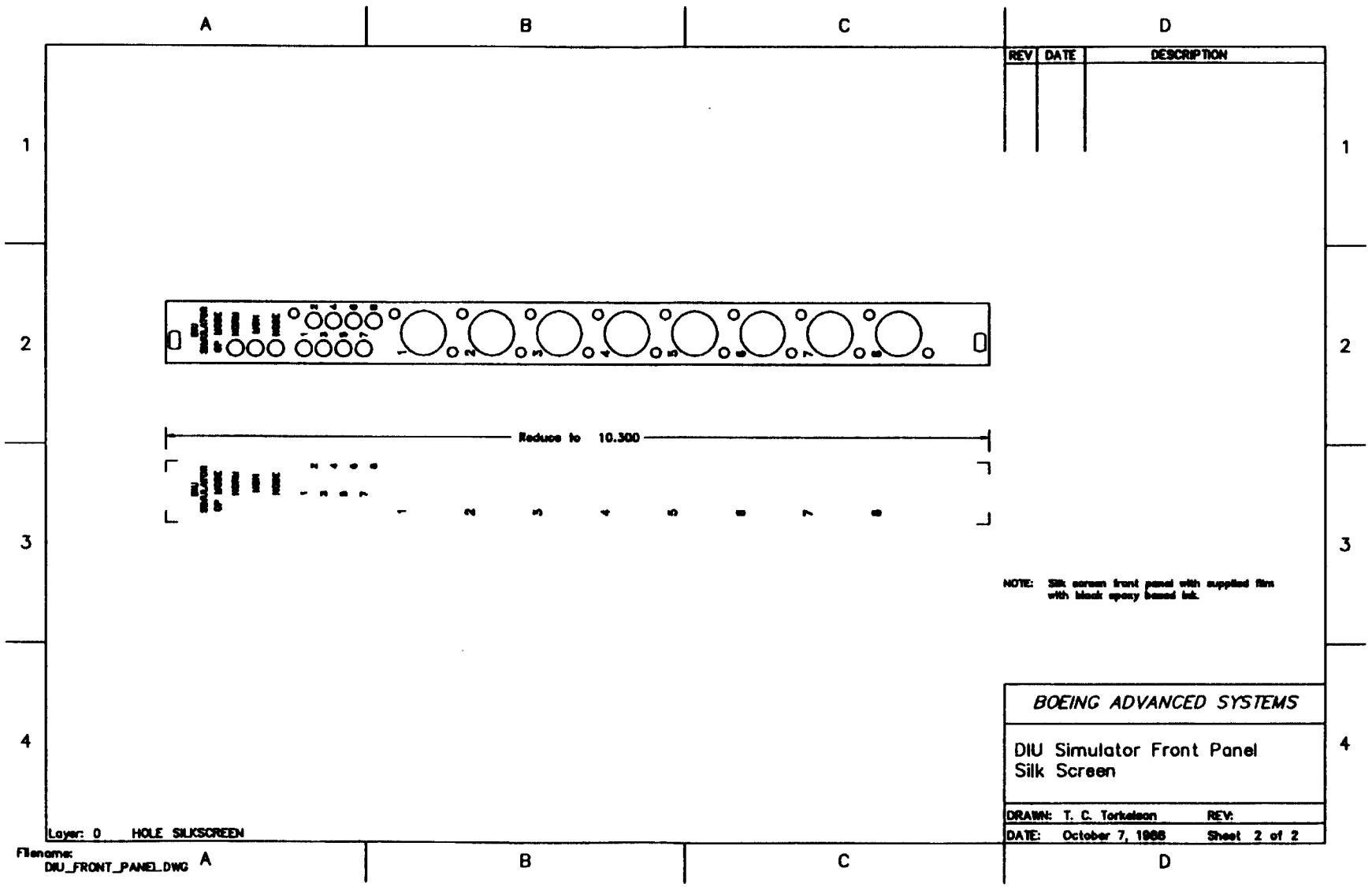
DRAWN: T. C. Torkelson REV:

DATE: October 7, 1988 Sheet of

ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

```
"*****************************************************************************
"  FILENAME:     INPUT_MODE.STATE        mode line states
"       DATE:    October 31, 1988
"         BY:    T.C. Torkelson
"*****************************************************************************
"
" This file defines input modes on lines M1, M0 of the ISIO daughter board.
" All devices which have M1 and M0 inputs should include this file for
" consistent definitions.
"

    input_mode = [M1, M0];

        NORMAL_INPUT = 0;        " input on NET_IN
        MONITOR_INPUT = 1;       " input on NET_IN and MONITOR_IN
        NODE_INPUT = 2;          " input on NODE_SIM_IN
        RESET_INPUT = 3;         " no inputs
```

```
"*****************************************************************************
"  FILENAME:    EP600_RX_CLOCK.ABL              AIPS RX Clock EPLD
"       DATE:   October 31, 1988
"         BY:   T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     separated test vectors from .abl
"                                revised pinout to match circuit board
"   B   6/13/89          TCT     changed glitch eqn to state machine
"                                fixed error in clock sync state machine
"                                revised bit assignments for HCSxx to fit EPLD
"*****************************************************************************

module ep600_rx_clock

flag    '-r3','-t1'

title  'AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev:  10/31/88
 '

"  REVISED      BY      DESCRIPTION
"  2/18/88      TCT     added glitch filter to prevent erroneous edge detect
"  5/18/88      TCT     revised for ABEL 3.0
"  8/29/88      TCT     added inputs for monitor, node simulator, mode select
"  8/31/88      TCT     changed SES to D FF, used state diagram instead of eqn
"  9/7/88       TCT     changed pinout, changed MONITOR_IN to macro,
"                       provided MONITOR_IN1 and MONITOR_IN2
"  10/31/88     TCT     separated out test vectors, matched pinout to circuit brd
"                       added LED driver outputs
"  11/20/88     TCT     revised LED driver output pin assignment
"                       made RXD high for input_mode == RESET_INPUT
"                       disabled RXC output until input_mode != RESET_INPUT
"                       changed Reset_LED output to OUTPUT_ENABLE
"Declarations:

        EP600_RX_CLOCK  device          'E0600';

    " define ABEL .. commands
        C,K,P,X,Z = .C.,.K.,.P.,.X.,.Z.;

    " inputs:

        SYS_CLOCK1      pin     1;      " 16 Mhz system clock from VME bus
        SYS_CLOCK2      pin     13;     " 16 Mhz system clock from VME bus

        NET_IN          pin     11;     " HDLC data from I/O network
        NODE_SIM_IN     pin     14;     " input from node simulator

        M1,M0           pin     23,2;   " mode select inputs

    " outputs:

        RXC,Q2,Q1,Q0    pin     3,5,6,7;        " RX clock registers
        EDGE,RXD,H1,H0  pin     8,4,9,10;       " Edge detect registers

        RXC,Q2,Q1,Q0    istype  'pos, reg_D, feed_reg';
        EDGE,RXD,H1,H0  istype  'pos, reg_D, feed_reg';
        RXC.OE          istype  'eqn';          " control for RXC clock output
```

```
    SE3,SE2,SE1,SE0 pin     19,20,21,22;     " Edge detect enable registers
    SE3,SE2,SE1,SE0 istype  'pos, reg_D, feed_reg';

    OUTPUT_ENABLE   pin     18;              " high to enable RXC output
    OUTPUT_ENABLE   istype  'com, feed_pin';" not enough terms for RXC out enable

    !Normal_LED     pin     17;              " low for Normal operation
    !Normal_LED     istype  'com';
    !Monitor_LED    pin     16;              " low for Monitor operation
    !Monitor_LED    istype  'com';
    !Node_LED       pin     15;              " low for Node operation
    !Node_LED       istype  'com';


" states:
    sys_clock = [SYS_CLOCK2,SYS_CLOCK1];

    edge_state = [RXD, H1, H0];

    HS0 = ^b000;
    HS1 = ^b001;
    HS2 = ^b010;
    HS3 = ^b011;     " positive edge
    HS4 = ^b100;     " negative edge
    HS5 = ^b101;
    HS6 = ^b110;
    HS7 = ^b111;

    sync_state = [SE3, SE2, SE1, SE0];

    SES0 = 0;
    SES1 = 1;
    SES2 = 2;
    SES3 = 3;
    SES4 = 4;
    SES5 = 5;
    SES6 = 6;
    SES7 = 7;
    SES8 = ^o10;     " edge sync enabled

    hdlc_clock = [RXC, Q2, Q1, Q0];

    HCS0 = 0;
    HCS1 = 1;
    HCS2 = 3;
    HCS3 = 7;
    HCS11 = ^o13;
    HCS14 = ^o17;
    HCS15 = ^o16;
    HCS16 = ^o14;
    HCS17 = ^o10;

  @INCLUDE    '[-]INPUT_MODE.STATE'

" macros:

    SYNC_ENABLE macro {SE3};
    SYNC_EDGE macro {(SYNC_ENABLE & EDGE)};

  " The following macro selects NET_IN for normal operation,
```

```
    " NODE_SIM_IN for either monitor or node operation, and causes
    " HDLC_IN to be high for reset operation

        HDLC_IN macro ((
             (input_mode == NORMAL_INPUT)   & NET_IN #
             (input_mode == MONITOR_INPUT) & NODE_SIM_IN #
             (input_mode == NODE_INPUT)     & NODE_SIM_IN #
             (input_mode == RESET_INPUT)
                       ));

        RX_SAMPLE macro ((hdlc_clock == HCS17));

equations

        OUTPUT_ENABLE = (input_mode != RESET_INPUT);
        RXC.OE = OUTPUT_ENABLE;

        EDGE := (edge_state == HS3) # (edge_state == HS4);

        Normal_LED  = (input_mode == NORMAL_INPUT);
        Node_LED    = (input_mode == NODE_INPUT);
        Monitor_LED = (input_mode == MONITOR_INPUT);

state_diagram hdlc_clock

        state HCS0:     goto HCS1;

        state HCS1:     if SYNC_EDGE then HCS1
                        else HCS2;

        state HCS2:     if SYNC_EDGE then HCS1
                        else HCS3;

        state HCS3:     if SYNC_EDGE then HCS1
                        else HCS14;

        state HCS14:    if SYNC_EDGE then HCS11
                        else HCS15;

        state HCS11:    goto HCS2;

        state HCS15:    if SYNC_EDGE then HCS1
                        else HCS16;

        state HCS16:    if SYNC_EDGE then HCS1
                        else HCS17;

        state HCS17:    if SYNC_EDGE then HCS1
                        else HCS0;

state_diagram sync_state

        state SES0:     if RX_SAMPLE & !EDGE then SES1
                        else SES0;

        state SES1:     if RX_SAMPLE & !EDGE then SES2
                        else if EDGE then SES0
                        else SES1;

        state SES2:     if RX_SAMPLE & !EDGE then SES3
                        else if EDGE then SES0
```

```
                        else SES2;

        state SES3:     if RX_SAMPLE & !EDGE then SES4
                        else if EDGE then SES0
                        else SES3;

        state SES4:     if RX_SAMPLE & !EDGE then SES5
                        else if EDGE then SES0
                        else SES4;

        state SES5:     if RX_SAMPLE & !EDGE then SES6
                        else if EDGE then SES0
                        else SES5;

        state SES6:     if RX_SAMPLE & !EDGE then SES7
                        else if EDGE then SES0
                        else SES6;

        state SES7:     if RX_SAMPLE & !EDGE then SES8
                        else if EDGE then SES0
                        else SES7;

        state SES8:     if EDGE then SES0
                        else SES8;


state_diagram edge_state

        state HS0:      if HDLC_IN then HS1
                        else HS0;

        state HS1:      if HDLC_IN then HS3
                        else HS2;               " glitch

        state HS3:      if HDLC_IN then HS7
                        else HS6;

        state HS7:      if !HDLC_IN then HS6
                        else HS7;

        state HS6:      if !HDLC_IN then HS4
                        else HS5;               " glitch

        state HS4:      if !HDLC_IN then HS0
                        else HS1;

" glitch states
        state HS2:      if HDLC_IN then HS1
                        else HS0;

        state HS5:      if !HDLC_IN then HS6
                        else HS7;

" Comment out the following line for production parts
"
"    @INCLUDE   'EP600_RX_CLOCK.TST'

end ep600_rx_clock
```

```
"*****************************************************************************
"  FILENAME:    EP600_RX_CLOCK.TST      RX clock test vectors
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV          DATE       BY          DESCRIPTION
"   A         10/31/88     TCT         separated test vectors from .abl
"   B         11/21/88     TCT         added vectors to check RESET_INPUT
"*****************************************************************************
"
" NOTE:  A complete test of the state machines is made in TEST_RX_CLOCK
"       files which provide special inputs to test the individual state
"       machines.
"
test_vectors    'Test Edge Detector'
                ([sys_clock, NET_IN, input_mode] -> [EDGE, RXD, H1, H0])

    " place in known state
                [C,0,NORMAL_INPUT] -> [X,X,X,X];
                [C,0,NORMAL_INPUT] -> [X,X,X,X];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test no edge condition - input low
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test high glitch - one sample
                [C,1,NORMAL_INPUT] -> [0,0,0,1];
                [C,0,NORMAL_INPUT] -> [0,0,1,0];        " this is a + glitch
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test positive edge - two samples
                [C,1,NORMAL_INPUT] -> [0,0,0,1];
                [C,1,NORMAL_INPUT] -> [0,0,1,1];        " next clock detects + edge
                [C,0,NORMAL_INPUT] -> [1,1,1,0];
                [C,0,NORMAL_INPUT] -> [0,1,0,0];        " next clock detects - edge
                [C,0,NORMAL_INPUT] -> [1,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test positive edge - three samples
                [C,1,NORMAL_INPUT] -> [0,0,0,1];
                [C,1,NORMAL_INPUT] -> [0,0,1,1];        " next clock detects + edge
                [C,1,NORMAL_INPUT] -> [1,1,1,1];
                [C,0,NORMAL_INPUT] -> [0,1,1,0];
                [C,1,NORMAL_INPUT] -> [0,1,0,1];        " this is a - glitch
                [C,1,NORMAL_INPUT] -> [0,1,1,1];
                [C,0,NORMAL_INPUT] -> [0,1,1,0];
                [C,0,NORMAL_INPUT] -> [0,1,0,0];        " next clock detects - edge
                [C,0,NORMAL_INPUT] -> [1,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];

" test_vectors  'Test Sync Enable State Machine'
"       ([NET_IN, input_mode, sys_clock, sync_state, hdlc_clock, edge_state]
"         -> [sync_state, hdlc_clock, edge_state])

test_vectors ' Test various input modes'
            ([sys_clock,input_mode,NET_IN,NODE_SIM_IN] ->
             [H0,RXC]);

              [C,NORMAL_INPUT,0,0] -> [0,X];
              [C,NORMAL_INPUT,1,0] -> [1,X];
              [C,NORMAL_INPUT,0,1] -> [0,X];

              [C,MONITOR_INPUT,0,0] -> [0,X];
```

```
                [C,MONITOR_INPUT,1,0] -> [0,X];
                [C,MONITOR_INPUT,0,1] -> [1,X];

                [C,NODE_INPUT,0,0] -> [0,X];
                [C,NODE_INPUT,1,0] -> [0,X];
                [C,NODE_INPUT,0,1] -> [1,X];

                [C,RESET_INPUT,0,0] -> [1,Z];
                [C,RESET_INPUT,1,0] -> [1,Z];
                [C,RESET_INPUT,0,1] -> [1,Z];

test_vectors ' Test LED outputs'
             (input_mode -> [OUTPUT_ENABLE, Normal_LED, Node_LED, Monitor_LED])

             RESET_INPUT          -> [0,0,0,0];
             NORMAL_INPUT         -> [1,1,0,0];
             NODE_INPUT           -> [1,0,1,0];
             MONITOR_INPUT        -> [1,0,0,1];
```

ABEL(tm) 3.00b - Document Generator 21-Jun-89 05:55 PM
AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module ep600_rx_clock

Device EP600_RX_CLOCK

- Reduced Equations:

```
    OUTPUT_ENABLE = (!M0 # !M1);

    _RXC_E = (OUTPUT_ENABLE);

    EDGE := (!H0 & !H1 & RXD # H0 & H1 & !RXD);

    ~Normal_LED = !(!M0 & !M1);

    ~Node_LED = !(!M0 & M1);

    ~Monitor_LED = !(M0 & !M1);

    RXC := (!EDGE & !Q0 & Q2 & RXC
            # !Q0 & Q2 & RXC & !SE3
            # Q0 & Q1 & Q2 & RXC
            # !EDGE & Q0 & Q1 & Q2
            # Q0 & Q1 & Q2 & !SE3);


    Q2 := (!EDGE & Q1 & Q2 & RXC
           # Q1 & Q2 & RXC & !SE3
           # !EDGE & Q0 & Q1 & !RXC
           # Q0 & Q1 & !RXC & !SE3);


    Q1 := (Q0 & Q1 & RXC
           # !EDGE & Q0 & Q1
           # Q0 & Q1 & !SE3
           # !EDGE & Q0 & !Q2 & !RXC
           # Q0 & !Q2 & !RXC & !SE3);


    Q0 := (EDGE & !Q0 & !Q1 & RXC & SE3
           # EDGE & !Q0 & Q2 & RXC & SE3
           # Q0 & Q1 & !Q2
           # Q0 & Q1 & !RXC
           # EDGE & Q0 & Q1 & SE3
           # !Q1 & !Q2 & !RXC);


    SE3 := (!EDGE & !SE0 & !SE1 & !SE2 & SE3
            # !EDGE & !Q0 & !Q1 & !Q2 & RXC & SE0 & SE1 & SE2 & !SE3);
```

ABEL(tm) 3.00b - Document Generator 21-Jun-89 05:55 PM
AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module ep600_rx_clock

Device EP600_RX_CLOCK

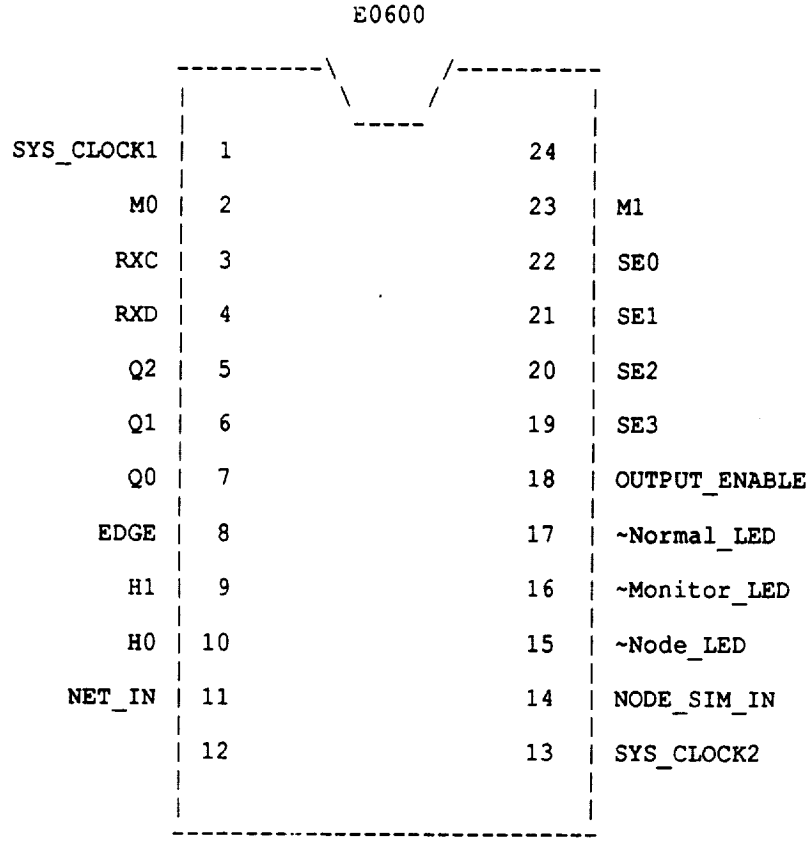
```
    SE2 := (!EDGE & !SE0 & SE2 & !SE3
            # !EDGE & Q0 & SE2 & !SE3
            # !EDGE & Q1 & SE2 & !SE3
            # !EDGE & Q2 & SE2 & !SE3
            # !EDGE & !RXC & SE2 & !SE3
            # !EDGE & !SE1 & SE2 & !SE3
            # !EDGE & !Q0 & !Q1 & !Q2 & RXC & SE0 & SE1 & !SE2 & !SE3);


    SE1 := (!EDGE & Q0 & SE1 & !SE3
            # !EDGE & Q1 & SE1 & !SE3
            # !EDGE & Q2 & SE1 & !SE3
            # !EDGE & !RXC & SE1 & !SE3
            # !EDGE & !SE0 & SE1 & !SE3
            # !EDGE & !Q0 & !Q1 & !Q2 & RXC & SE0 & !SE1 & !SE3);


    SE0 := (!EDGE & Q0 & SE0 & !SE3
            # !EDGE & Q1 & SE0 & !SE3
            # !EDGE & Q2 & SE0 & !SE3
            # !EDGE & !RXC & SE0 & !SE3
            # !EDGE & !Q0 & !Q1 & !Q2 & RXC & !SE0 & !SE3);


    RXD := (H0 & RXD # H1 & RXD # H0 & H1);

    H1 := (H0);

    H0 := (M0 & M1
           # M1 & NODE_SIM_IN
           # M0 & NODE_SIM_IN
           # !M0 & !M1 & NET_IN);
```

ABEL(tm) 3.00b - Document Generator 21-Jun-89 05:55 PM
AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Chip diagram for Module ep600_rx_clock

Device EP600_RX_CLOCK



end of module ep600_rx_clock

```
"*****************************************************************************
"  FILENAME:    TEST_RX_CLOCK.ABL       AIPS RX Clock EPLD
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     separated test vectors from .abl
"                                revised pinout to match circuit board
"   B   6/13/89          TCT     changed glitch eqn to state machine
"                                fixed error in clock sync state machine
"                                revised bit assignments for HCSxx to fit EPLD
"   -   changed title and eliminated LED outputs to allow testing by breaking
"       path from ENABLE and SE3 to other state machines
"*****************************************************************************

module test_rx_clock

flag    '-r3','-t1'

title  'AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev:  10/31/88
'

"  REVISED      BY      DESCRIPTION
"  2/18/88      TCT     added glitch filter to prevent erroneous edge detect
"  5/18/88      TCT     revised for ABEL 3.0
"  8/29/88      TCT     added inputs for monitor, node simulator, mode select
"  8/31/88      TCT     changed SES to D FF, used state diagram instead of eqn
"  9/7/88       TCT     changed pinout, changed MONITOR_IN to macro,
"                       provided MONITOR_IN1 and MONITOR_IN2
"  10/31/88     TCT     separated out test vectors, matched pinout to circuit brd
"                       added LED driver outputs
"  11/20/88     TCT     revised LED driver output pin assignment
"                       made RXD high for input_mode == RESET_INPUT
"                       disabled RXC output until input_mode != RESET_INPUT
"                       changed Reset_LED output to OUTPUT_ENABLE
"Declarations:

        TEST_RX_CLOCK   device          'E0600';

    " define ABEL .. commands
        C,K,P,X,Z = .C.,.K.,.P.,.X.,.Z.;

    " inputs:

        SYS_CLOCK1      pin     1;      " 16 Mhz system clock from VME bus
        SYS_CLOCK2      pin     13;     " 16 Mhz system clock from VME bus

        NET_IN          pin     11;     " HDLC data from I/O network
        NODE_SIM_IN     pin     14;     " input from node simulator

        M1,M0           pin     23,2;   " mode select inputs

    " outputs:

        RXC,Q2,Q1,Q0    pin     3,5,6,7;        " RX clock registers
        EDGE,RXD,H1,H0  pin     8,4,9,10;       " Edge detect registers

        RXC,Q2,Q1,Q0    istype  'pos, reg_D, feed_reg';
```

```
        EDGE,RXD,H1,H0  istype  'pos, reg_D, feed_reg';
        RXC.OE          istype  'eqn';          " control for RXC clock output

        SE3,SE2,SE1,SE0 pin     19,20,21,22;    " Edge detect enable registers
        SE3,SE2,SE1,SE0 istype  'pos, reg_D, feed_reg';

        OUTPUT_ENABLE   pin     18;             " high to enable RXC output
        OUTPUT_ENABLE   istype  'com, feed_pin';" not enough terms for RXC out enable

" these are inputs added for test purposes
        EDGE_IN         pin     17;
        SYNC_ENABLE_IN  pin     16;
        RX_SAMPLE       pin     15;

    " states:
        sys_clock = [SYS_CLOCK2,SYS_CLOCK1];

        edge_state = [RXD, H1, H0];

        HS0 = ^b000;
        HS1 = ^b001;
        HS2 = ^b010;
        HS3 = ^b011;    " positive edge
        HS4 = ^b100;    " negative edge
        HS5 = ^b101;
        HS6 = ^b110;
        HS7 = ^b111;

        sync_state = [SE3, SE2, SE1, SE0];

        SES0 = 0;
        SES1 = 1;
        SES2 = 2;
        SES3 = 3;
        SES4 = 4;
        SES5 = 5;
        SES6 = 6;
        SES7 = 7;
        SES8 = ^o10;    " edge sync enabled

        hdlc_clock = [RXC, Q2, Q1, Q0];

        HCS0 = 0;
        HCS1 = 1;
        HCS2 = 3;
        HCS3 = 7;
        HCS11 = ^o13;
        HCS14 = ^o17;
        HCS15 = ^o16;
        HCS16 = ^o14;
        HCS17 = ^o10;

    @INCLUDE    '[-]INPUT_MODE.STATE'

 " macros:

        SYNC_EDGE macro {(SYNC_ENABLE_IN & EDGE_IN)};

    " The following macro selects NET_IN for normal operation,
    " NODE_SIM_IN for either monitor or node operation, and causes
    " HDLC_IN to be high for reset operation
```

```
        HDLC_IN macro ((
             (input_mode == NORMAL_INPUT)   & NET_IN #
             (input_mode == MONITOR_INPUT) & NODE_SIM_IN #
             (input_mode == NODE_INPUT)     & NODE_SIM_IN #
             (input_mode == RESET_INPUT)
                       )};

equations

        OUTPUT_ENABLE = (input_mode != RESET_INPUT);
        RXC.OE = OUTPUT_ENABLE;

        EDGE := (edge_state == HS3) # (edge_state == HS4);

state_diagram hdlc_clock

        state HCS0:     goto HCS1;

        state HCS1:     if SYNC_EDGE then HCS1
                        else HCS2;

        state HCS2:     if SYNC_EDGE then HCS1
                        else HCS3;

        state HCS3:     if SYNC_EDGE then HCS1
                        else HCS14;

        state HCS14:    if SYNC_EDGE then HCS11
                        else HCS15;

        state HCS11:    goto HCS2;

        state HCS15:    if SYNC_EDGE then HCS1
                        else HCS16;

        state HCS16:    if SYNC_EDGE then HCS1
                        else HCS17;

        state HCS17:    if SYNC_EDGE then HCS1
                        else HCS0;

state_diagram sync_state

        state SES0:     if RX_SAMPLE & !EDGE_IN then SES1
                        else SES0;

        state SES1:     if RX_SAMPLE & !EDGE_IN then SES2
                        else if EDGE_IN then SES0
                        else SES1;

        state SES2:     if RX_SAMPLE & !EDGE_IN then SES3
                        else if EDGE_IN then SES0
                        else SES2;

        state SES3:     if RX_SAMPLE & !EDGE_IN then SES4
                        else if EDGE_IN then SES0
                        else SES3;

        state SES4:     if RX_SAMPLE & !EDGE_IN then SES5
                        else if EDGE_IN then SES0
```

```
                        else SES4;

        state SES5:     if RX_SAMPLE & !EDGE_IN then SES6
                        else if EDGE_IN then SES0
                        else SES5;

        state SES6:     if RX_SAMPLE & !EDGE_IN then SES7
                        else if EDGE_IN then SES0
                        else SES6;

        state SES7:     if RX_SAMPLE & !EDGE_IN then SES8
                        else if EDGE_IN then SES0
                        else SES7;

        state SES8:     if EDGE_IN then SES0
                        else SES8;


state_diagram edge_state

        state HS0:      if HDLC_IN then HS1
                        else HS0;

        state HS1:      if HDLC_IN then HS3
                        else HS2;               " glitch

        state HS3:      if HDLC_IN then HS7
                        else HS6;

        state HS7:      if !HDLC_IN then HS6
                        else HS7;

        state HS6:      if !HDLC_IN then HS4
                        else HS5;               " glitch

        state HS4:      if !HDLC_IN then HS0
                        else HS1;

" glitch states
        state HS2:      if HDLC_IN then HS1
                        else HS0;

        state HS5:      if !HDLC_IN then HS6
                        else HS7;

"*****************************************************************************
"  FILENAME:    TEST_RX_CLOCK.TST       RX clock test vectors
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV          DATE       BY           DESCRIPTION
"   A         10/31/88     TCT          separated test vectors from .abl
"   B         11/21/88     TCT          added vectors to check RESET_INPUT
"*****************************************************************************

test_vectors    'Test Edge Detector'
                ([sys_clock, NET_IN, input_mode] -> [EDGE, RXD, H1, H0])

    " place in known state
                [C,0,NORMAL_INPUT] -> [X,X,X,X];
                [C,0,NORMAL_INPUT] -> [X,X,X,X];
```

```
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test no edge condition - input low
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test high glitch - one sample
                [C,1,NORMAL_INPUT] -> [0,0,0,1];
                [C,0,NORMAL_INPUT] -> [0,0,1,0];        " this is a + glitch
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test positive edge - two samples
                [C,1,NORMAL_INPUT] -> [0,0,0,1];
                [C,1,NORMAL_INPUT] -> [0,0,1,1];        " next clock detects + edge
                [C,0,NORMAL_INPUT] -> [1,1,1,0];
                [C,0,NORMAL_INPUT] -> [0,1,0,0];        " next clock detects - edge
                [C,0,NORMAL_INPUT] -> [1,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
    " test positive edge - three samples
                [C,1,NORMAL_INPUT] -> [0,0,0,1];
                [C,1,NORMAL_INPUT] -> [0,0,1,1];        " next clock detects + edge
                [C,1,NORMAL_INPUT] -> [1,1,1,1];
                [C,0,NORMAL_INPUT] -> [0,1,1,0];
                [C,1,NORMAL_INPUT] -> [0,1,0,1];        " this is a - glitch
                [C,1,NORMAL_INPUT] -> [0,1,1,1];
                [C,0,NORMAL_INPUT] -> [0,1,1,0];
                [C,0,NORMAL_INPUT] -> [0,1,0,0];        " next clock detects - edge
                [C,0,NORMAL_INPUT] -> [1,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];
                [C,0,NORMAL_INPUT] -> [0,0,0,0];

test_vectors    'Test Various Input Modes'
    ([sys_clock,input_mode,NET_IN,NODE_SIM_IN] -> [H0,RXC]);

               [C,NORMAL_INPUT,0,0] -> [0,X];
               [C,NORMAL_INPUT,1,0] -> [1,X];
               [C,NORMAL_INPUT,0,1] -> [0,X];

               [C,MONITOR_INPUT,0,0] -> [0,X];
               [C,MONITOR_INPUT,1,0] -> [0,X];
               [C,MONITOR_INPUT,0,1] -> [1,X];

               [C,NODE_INPUT,0,0] -> [0,X];
               [C,NODE_INPUT,1,0] -> [0,X];
               [C,NODE_INPUT,0,1] -> [1,X];

               [C,RESET_INPUT,0,0] -> [1,Z];
               [C,RESET_INPUT,1,0] -> [1,Z];
               [C,RESET_INPUT,0,1] -> [1,Z];

test_vectors    'Test Sync Enable - Normal operation'
    ([sys_clock, EDGE_IN, RX_SAMPLE, sync_state] -> [sync_state])

              [P,0,0,!SES0] -> [SES0];

              [C,0,0,X] -> [SES0];
              [C,0,1,X] -> [SES1];

              [C,0,0,X] -> [SES1];
              [C,0,1,X] -> [SES2];

              [C,0,0,X] -> [SES2];
              [C,0,1,X] -> [SES3];
```

```
            [C,0,0,X] -> [SES3];
            [C,0,1,X] -> [SES4];

            [C,0,0,X] -> [SES4];
            [C,0,1,X] -> [SES5];

            [C,0,0,X] -> [SES5];
            [C,0,1,X] -> [SES6];

            [C,0,0,X] -> [SES6];
            [C,0,1,X] -> [SES7];

            [C,0,0,X] -> [SES7];
            [C,0,1,X] -> [SES8];

            [C,0,0,X] -> [SES8];
            [C,0,1,X] -> [SES8];

            [C,1,0,X] -> [SES0];

test_vectors    'Test Sync Enable - Normal edge reset'
    ([sys_clock, EDGE_IN, RX_SAMPLE, sync_state] -> [sync_state])

            [P,0,0,!SES0] -> [SES0];
            [C,1,0,SES0] -> [SES0];

            [P,0,0,!SES1] -> [SES1];
            [C,1,0,SES1] -> [SES0];

            [P,0,0,!SES2] -> [SES2];
            [C,1,0,SES2] -> [SES0];

            [P,0,0,!SES3] -> [SES3];
            [C,1,0,SES3] -> [SES0];

            [P,0,0,!SES4] -> [SES4];
            [C,1,0,SES4] -> [SES0];

            [P,0,0,!SES5] -> [SES5];
            [C,1,0,SES5] -> [SES0];

            [P,0,0,!SES6] -> [SES6];
            [C,1,0,SES6] -> [SES0];

            [P,0,0,!SES7] -> [SES7];
            [C,1,0,SES7] -> [SES0];

test_vectors    'Test Sync Enable - Edge / RX_Sample contention'
    ([sys_clock, EDGE_IN, RX_SAMPLE, sync_state] -> [sync_state])

            [P,0,0,!SES0] -> [SES0];
            [C,1,1,SES0] -> [SES0];

            [P,0,0,!SES1] -> [SES1];
            [C,1,1,SES1] -> [SES0];

            [P,0,0,!SES2] -> [SES2];
            [C,1,1,SES2] -> [SES0];

            [P,0,0,!SES3] -> [SES3];
            [C,1,1,SES3] -> [SES0];
```

```
            [P,0,0,!SES4] -> [SES4];
            [C,1,1,SES4] -> [SES0];

            [P,0,0,!SES5] -> [SES5];
            [C,1,1,SES5] -> [SES0];

            [P,0,0,!SES6] -> [SES6];
            [C,1,1,SES6] -> [SES0];

            [P,0,0,!SES7] -> [SES7];
            [C,1,1,SES7] -> [SES0];


test_vectors    'Test Clock Generator State Machine Free Run'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS0] -> [HCS0];

            [C,0,0,HCS0] -> [HCS1];
            [C,0,0,X] -> [HCS2];
            [C,0,0,X] -> [HCS3];
            [C,0,0,X] -> [HCS14];
            [C,0,0,X] -> [HCS15];
            [C,0,0,X] -> [HCS16];
            [C,0,0,X] -> [HCS17];
            [C,0,0,X] -> [HCS0];

test_vectors    'Test Clock Generator Sync Operation at HCS0'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS0] -> [HCS0];

            [C,1,1,HCS0] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS1'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS1] -> [HCS1];

            [C,1,1,HCS1] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS2'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS2] -> [HCS2];

            [C,1,1,HCS2] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS3'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS3] -> [HCS3];

            [C,1,1,HCS3] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS14'
```

```
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS14] -> [HCS14];

            [C,1,1,HCS14] -> [HCS11];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS15'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS15] -> [HCS15];

            [C,1,1,HCS15] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS16'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS16] -> [HCS16];

            [C,1,1,HCS16] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Sync Operation at HCS17'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS17] -> [HCS17];

            [C,1,1,HCS0] -> [HCS1];
            [C,0,0,X] -> [HCS2];

test_vectors    'Test Clock Generator Enabled, No Edge'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS0] -> [HCS0];

            [C,0,1,HCS0] -> [HCS1];
            [C,0,1,X] -> [HCS2];
            [C,0,1,X] -> [HCS3];
            [C,0,1,X] -> [HCS14];
            [C,0,1,X] -> [HCS15];
            [C,0,1,X] -> [HCS16];
            [C,0,1,X] -> [HCS17];
            [C,0,1,X] -> [HCS0];


test_vectors    'Test Clock Generator Disabled, Edge'
    ([sys_clock, EDGE_IN, SYNC_ENABLE_IN, hdlc_clock] -> [hdlc_clock])

            [P,0,0,!HCS0] -> [HCS0];

            [C,1,0,HCS0] -> [HCS1];
            [C,1,0,X] -> [HCS2];
            [C,1,0,X] -> [HCS3];
            [C,1,0,X] -> [HCS14];
            [C,1,0,X] -> [HCS15];
            [C,1,0,X] -> [HCS16];
            [C,1,0,X] -> [HCS17];
            [C,1,0,X] -> [HCS0];
```

```
end test_rx_clock
```

```
ABEL(tm) 3.00b  -  Document Generator              21-Jun-89 03:21 PM
AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev:  10/31/88

Equations for Module test_rx_clock

Device TEST_RX_CLOCK


- Reduced Equations:

    OUTPUT_ENABLE = (!M0 # !M1);

    _RXC_E = (OUTPUT_ENABLE);

    EDGE := (!H0 & !H1 & RXD # H0 & H1 & !RXD);

    RXC := (!EDGE_IN & !Q0 & Q2 & RXC
            # !Q0 & Q2 & RXC & !SYNC_ENABLE_IN
            # Q0 & Q1 & Q2 & RXC
            # !EDGE_IN & Q0 & Q1 & Q2
            # Q0 & !Q1 & Q2 & !SYNC_ENABLE_IN);


    Q2 := (!EDGE_IN & Q1 & Q2 & RXC
           # Q1 & Q2 & RXC & !SYNC_ENABLE_IN
           # !EDGE_IN & Q0 & Q1 & !RXC
           # Q0 & !Q1 & !RXC & !SYNC_ENABLE_IN);


    Q1 := (Q0 & Q1 & RXC
           # !EDGE_IN & Q0 & Q1
           # Q0 & !Q1 & !SYNC_ENABLE_IN
           # !EDGE_IN & Q0 & !Q2 & !RXC
           # Q0 & !Q2 & !RXC & !SYNC_ENABLE_IN);


    Q0 := (EDGE_IN & !Q0 & !Q1 & RXC & SYNC_ENABLE_IN
           # EDGE_IN & !Q0 & Q2 & RXC & SYNC_ENABLE_IN
           # Q0 & Q1 & !Q2
           # Q0 & Q1 & !RXC
           # EDGE_IN & Q0 & Q1 & SYNC_ENABLE_IN
           # !Q1 & !Q2 & !RXC);


    SE3 := (!EDGE_IN & !SE0 & !SE1 & !SE2 & SE3
            # !EDGE_IN & RX_SAMPLE & SE0 & SE1 & SE2 & !SE3);


    SE2 := (!EDGE_IN & !RX_SAMPLE & SE2 & !SE3
            # !EDGE_IN & !SE0 & SE2 & !SE3
            # !EDGE_IN & !SE1 & SE2 & !SE3
            # !EDGE_IN & RX_SAMPLE & SE0 & SE1 & !SE2 & !SE3);
```

ABEL(tm) 3.00b - Document Generator 21-Jun-89 03:21 PM
AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module test_rx_clock

Device TEST_RX_CLOCK

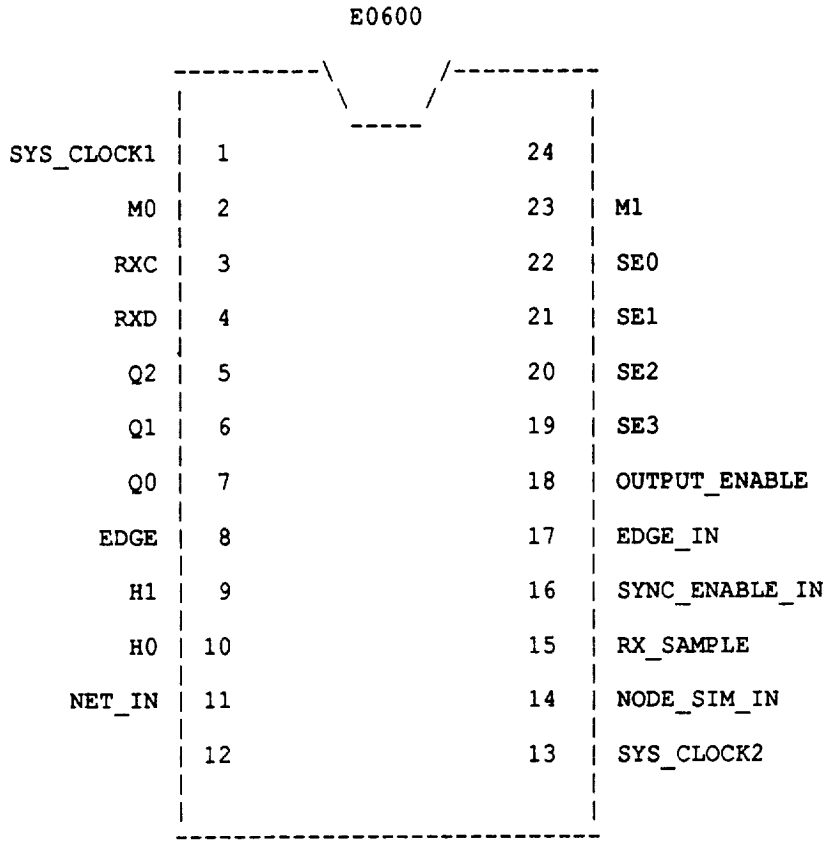
```
SE1 := (!EDGE_IN & !RX_SAMPLE & SE1 & !SE3
      # !EDGE_IN & !SE0 & SE1 & !SE3
      # !EDGE_IN & RX_SAMPLE & SE0 & !SE1 & !SE3);


SE0 := (!EDGE_IN & !RX_SAMPLE & SE0 & !SE3
      # !EDGE_IN & RX_SAMPLE & !SE0 & !SE3);


RXD := (H0 & RXD # H1 & RXD # H0 & H1);

H1 := (H0);

H0 := (M0 & M1
      # M1 & NODE_SIM_IN
      # M0 & NODE_SIM_IN
      # !M0 & !M1 & NET_IN);
```

ABEL(tm) 3.00b - Document Generator 21-Jun-89 03:21 PM
AIPS I/O Network HDLC Received Data Clock Sync PAL

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Chip diagram for Module test_rx_clock

Device TEST_RX_CLOCK



end of module test_rx_clock

```
"*****************************************************************************
"  FILENAME:    DIU_NODE_22V10.ABL      Simulated node for DIUs
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Changed M1, M0 pinout to agree with PC board
"                                Separated out test vectors
"*****************************************************************************

module diu_node_22v10

flag    '-r2'

title   'AIPS Simulated I/O Network Node EPLD

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev: 10/31/88
 '

"  REVISED      BY      DESCRIPTION

"
"

" This device is used on the network adapter to implement non-existent nodes.

"Declarations:

        DIU_NODE_22V10  device          'P22V10';

    " inputs:

        NODE_IN1, NODE_IN2              pin     2,3;

        DIU1_IN, DIU2_IN                pin     4,5;
        DIU3_IN, DIU4_IN                pin     6,7;
        DIU5_IN, DIU6_IN                pin     8,9;
        DIU7_IN, DIU8_IN                pin     10,11;

        M1, M0                          pin     1,13;

    " outputs:

        NODE_OUT1, NODE_OUT2            pin     23,22;

        DIU1_OUT, DIU2_OUT              pin     21,20;
        DIU3_OUT, DIU4_OUT              pin     19,18;
        DIU5_OUT, DIU6_OUT              pin     17,16;
        DIU7_OUT, DIU8_OUT              pin     15,14;

        NODE_OUT1, NODE_OUT2            istype  'neg, com';

        DIU1_OUT, DIU2_OUT              istype  'neg, com';
        DIU3_OUT, DIU4_OUT              istype  'neg, com';
        DIU5_OUT, DIU6_OUT              istype  'neg, com';
        DIU7_OUT, DIU8_OUT              istype  'neg, com';

    " states

        input_mode = [M1, M0];
```

```
        NORMAL_INPUT = 0;
        MONITOR_INPUT = 1;
        NODE_INPUT = 2;
        RESET_INPUT = 3;

equations

        NODE_OUT1 = (input_mode == NODE_INPUT) &
                        (NODE_IN2 #
                         DIU1_IN # DIU2_IN # DIU3_IN # DIU4_IN #
                         DIU5_IN # DIU6_IN # DIU7_IN # DIU8_IN) #
                    (input_mode == MONITOR_INPUT) &
                        (NODE_IN2) #
                    (input_mode == NORMAL_INPUT) &
                        (DIU1_IN) #
                    (input_mode == RESET_INPUT) &
                        (0);

        NODE_OUT2 = (input_mode == NODE_INPUT) &
                        (NODE_IN1 #
                         DIU1_IN # DIU2_IN # DIU3_IN # DIU4_IN #
                         DIU5_IN # DIU6_IN # DIU7_IN # DIU8_IN) #
                    (input_mode == MONITOR_INPUT) &
                        (NODE_IN1) #
                    (input_mode == NORMAL_INPUT) &
                        (DIU2_IN) #
                    (input_mode == RESET_INPUT) &
                        (0);

        DIU1_OUT =  (input_mode == NODE_INPUT) &
                        (NODE_IN1 # NODE_IN2 #
                                    DIU2_IN # DIU3_IN # DIU4_IN #
                         DIU5_IN # DIU6_IN # DIU7_IN # DIU8_IN) #
                    (input_mode == MONITOR_INPUT) &
                        (NODE_IN1 # NODE_IN2) #
                    (input_mode == NORMAL_INPUT) &
                        (0) #
                    (input_mode == RESET_INPUT) &
                        (0);

        DIU2_OUT =  (input_mode == NODE_INPUT) &
                        (NODE_IN1 # NODE_IN2 #
                         DIU1_IN #          DIU3_IN # DIU4_IN #
                         DIU5_IN # DIU6_IN # DIU7_IN # DIU8_IN) #
                    (input_mode == MONITOR_INPUT) &
                        (0) #
                    (input_mode == NORMAL_INPUT) &
                        (0) #
                    (input_mode == RESET_INPUT) &
                        (0);

        DIU3_OUT =  (input_mode == NODE_INPUT) &
                        (NODE_IN1 # NODE_IN2 #
                         DIU1_IN # DIU2_IN #           DIU4_IN #
                         DIU5_IN # DIU6_IN # DIU7_IN # DIU8_IN) #
                    (input_mode == MONITOR_INPUT) &
                        (0) #
                    (input_mode == NORMAL_INPUT) &
                        (0) #
                    (input_mode == RESET_INPUT) &
```

```
                    (0);

DIU4_OUT =  (input_mode == NODE_INPUT) &
                (NODE_IN1 # NODE_IN2 #
                 DIU1_IN # DIU2_IN # DIU3_IN #
                 DIU5_IN # DIU6_IN # DIU7_IN # DIU8_IN) #
            (input_mode == MONITOR_INPUT) &
                (0) #
            (input_mode == NORMAL_INPUT) &
                (0) #
            (input_mode == RESET_INPUT) &
                (0);

DIU5_OUT =  (input_mode == NODE_INPUT) &
                (NODE_IN1 # NODE_IN2 #
                 DIU1_IN # DIU2_IN # DIU3_IN # DIU4_IN #
                           DIU6_IN # DIU7_IN # DIU8_IN) #
            (input_mode == MONITOR_INPUT) &
                (0) #
            (input_mode == NORMAL_INPUT) &
                (0) #
            (input_mode == RESET_INPUT) &
                (0);

DIU6_OUT =  (input_mode == NODE_INPUT) &
                (NODE_IN1 # NODE_IN2 #
                 DIU1_IN # DIU2_IN # DIU3_IN # DIU4_IN #
                 DIU5_IN #           DIU7_IN # DIU8_IN) #
            (input_mode == MONITOR_INPUT) &
                (0) #
            (input_mode == NORMAL_INPUT) &
                (0) #
            (input_mode == RESET_INPUT) &
                (0);

DIU7_OUT =  (input_mode == NODE_INPUT) &
                (NODE_IN1 # NODE_IN2 #
                 DIU1_IN # DIU2_IN # DIU3_IN # DIU4_IN #
                 DIU5_IN # DIU6_IN #           DIU8_IN) #
            (input_mode == MONITOR_INPUT) &
                (0) #
            (input_mode == NORMAL_INPUT) &
                (0) #
            (input_mode == RESET_INPUT) &
                (0);

DIU8_OUT =  (input_mode == NODE_INPUT) &
                (NODE_IN1 # NODE_IN2 #
                 DIU1_IN # DIU2_IN # DIU3_IN # DIU4_IN #
                 DIU5_IN # DIU6_IN # DIU7_IN          ) #
            (input_mode == MONITOR_INPUT) &
                (0) #
            (input_mode == NORMAL_INPUT) &
                (0) #
            (input_mode == RESET_INPUT) &
                (0);

" Comment out the following line to compile a production .JED file
"
"       @INCLUDE        'DIU_NODE_22V10.TST'
```

```
end diu_node_22v10
```

```
"*****************************************************************************
"  FILENAME:    DIU_NODE_22V10.TST      Simulated node for DIUs test vectors
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Separated out test vectors
"*****************************************************************************

test_vectors  ' Test Node Simulator EPLD'
               ([NODE_IN1,NODE_IN2,DIU1_IN,DIU2_IN,DIU3_IN,
               DIU4_IN,DIU5_IN,DIU6_IN,DIU7_IN,DIU8_IN, input_mode] ->
               [NODE_OUT1,NODE_OUT2,DIU1_OUT,DIU2_OUT,DIU3_OUT,
               DIU4_OUT,DIU5_OUT,DIU6_OUT,DIU7_OUT,DIU8_OUT])

    " test NODE_INn
        [0,0,0,0,0,0,0,0,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [1,0,0,0,0,0,0,0,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,1,0,0,0,0,0,0,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    " test DIUn_IN
        [0,0,1,0,0,0,0,0,0,0,NORMAL_INPUT] -> [1,0,0,0,0,0,0,0,0,0];
        [0,0,0,1,0,0,0,0,0,0,NORMAL_INPUT] -> [0,1,0,0,0,0,0,0,0,0];
        [0,0,0,0,1,0,0,0,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,1,0,0,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,1,0,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,0,1,0,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,0,0,1,0,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,0,0,0,1,NORMAL_INPUT] -> [0,0,0,0,0,0,0,0,0,0];

    " test NODE_INn
        [0,0,0,0,0,0,0,0,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [1,0,0,0,0,0,0,0,0,0,MONITOR_INPUT] -> [0,1,1,0,0,0,0,0,0,0];
        [0,1,0,0,0,0,0,0,0,0,MONITOR_INPUT] -> [1,0,1,0,0,0,0,0,0,0];
    " test DIUn_IN
        [0,0,1,0,0,0,0,0,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,1,0,0,0,0,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,1,0,0,0,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,1,0,0,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,1,0,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,0,1,0,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,0,0,1,0,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,0,0,0,0,0,0,0,0,1,MONITOR_INPUT] -> [0,0,0,0,0,0,0,0,0,0];

    " test NODE_INn
        [0,0,0,0,0,0,0,0,0,0,NODE_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [1,0,0,0,0,0,0,0,0,0,NODE_INPUT] -> [0,1,1,1,1,1,1,1,1,1];
        [0,1,0,0,0,0,0,0,0,0,NODE_INPUT] -> [1,0,1,1,1,1,1,1,1,1];
    " test DIUn_IN
        [0,0,1,0,0,0,0,0,0,0,NODE_INPUT] -> [1,1,0,1,1,1,1,1,1,1];
        [0,0,0,1,0,0,0,0,0,0,NODE_INPUT] -> [1,1,1,0,1,1,1,1,1,1];
        [0,0,0,0,1,0,0,0,0,0,NODE_INPUT] -> [1,1,1,1,0,1,1,1,1,1];
        [0,0,0,0,0,1,0,0,0,0,NODE_INPUT] -> [1,1,1,1,1,0,1,1,1,1];
        [0,0,0,0,0,0,1,0,0,0,NODE_INPUT] -> [1,1,1,1,1,1,0,1,1,1];
        [0,0,0,0,0,0,0,1,0,0,NODE_INPUT] -> [1,1,1,1,1,1,1,0,1,1];
        [0,0,0,0,0,0,0,0,1,0,NODE_INPUT] -> [1,1,1,1,1,1,1,1,0,1];
        [0,0,0,0,0,0,0,0,0,1,NODE_INPUT] -> [1,1,1,1,1,1,1,1,1,0];

    " test NODE_INn
        [0,0,0,0,0,0,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [1,0,0,0,0,0,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
        [0,1,0,0,0,0,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
```

```
" test DIUn_IN
    [0,0,1,0,0,0,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,1,0,0,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,0,1,0,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,0,0,1,0,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,0,0,0,1,0,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,0,0,0,0,1,0,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,0,0,0,0,0,1,0,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
    [0,0,0,0,0,0,0,0,0,1,RESET_INPUT] -> [0,0,0,0,0,0,0,0,0,0];
```

ABEL(tm) 3.00b - Document Generator 20-Dec-88 01:04 PM
AIPS Simulated I/O Network Node EPLD

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module diu_node_22v10

Device DIU_NODE_22V10

- Reduced Equations:

```
    NODE_OUT1 = !(!DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN &
                !DIU6_IN & !DIU7_IN & !DIU8_IN & !NODE_IN2
                  # !DIU1_IN & !M0 & !M1
                  # M0 & !NODE_IN2
                  # M0 & M1);


    NODE_OUT2 = !(M0 & M1
                  # !DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN &
                !DIU6_IN & !DIU7_IN & !DIU8_IN & !NODE_IN1
                  # !DIU2_IN & !M0 & !M1
                  # M0 & !NODE_IN1);


    DIU1_OUT = !(M0 & M1
                  # !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN & !DIU6_IN &
               !DIU7_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2
                  # !M1 & !NODE_IN1 & !NODE_IN2
                  # !M0 & !M1);


    DIU2_OUT = !(!DIU1_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN & !DIU6_IN &
               !DIU7_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2
                  # M0
                  # !M1);

    DIU3_OUT = !(M0
                  # !M1
                  # !DIU1_IN & !DIU2_IN & !DIU4_IN & !DIU5_IN & !DIU6_IN &
               !DIU7_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2);

    DIU4_OUT = !(M0
                  # !M1
                  # !DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU5_IN & !DIU6_IN &
               !DIU7_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2);

    DIU5_OUT = !(M0
                  # !M1
                  # !DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU6_IN &
               !DIU7_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2);

    DIU6_OUT = !(M0
                  # !M1
                  # !DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN &
```

ABEL(tm) 3.00b - Document Generator 20-Dec-88 01:04 PM
AIPS Simulated I/O Network Node EPLD

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module diu_node_22v10

Device DIU_NODE_22V10

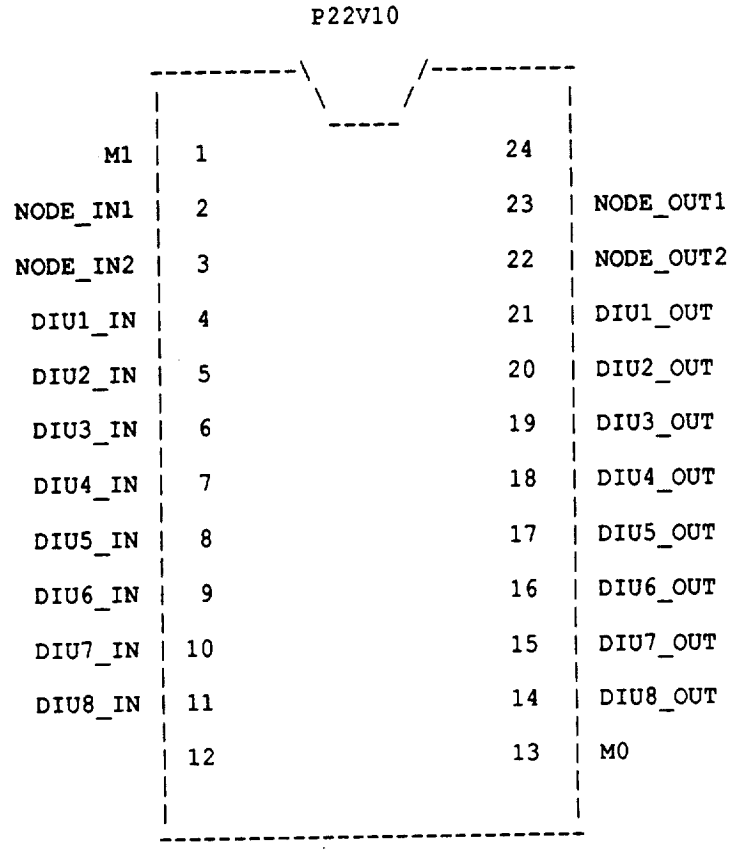
```
              !DIU7_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2);

    DIU7_OUT = !(M0
                 # !M1
                 # !DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN &
              !DIU6_IN & !DIU8_IN & !NODE_IN1 & !NODE_IN2);

    DIU8_OUT = !(M0
                 # !M1
                 # !DIU1_IN & !DIU2_IN & !DIU3_IN & !DIU4_IN & !DIU5_IN &
              !DIU6_IN & !DIU7_IN & !NODE_IN1 & !NODE_IN2);
```

ABEL(tm) 3.00b - Document Generator 20-Dec-88 01:04 PM
AIPS Simulated I/O Network Node EPLD

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Chip diagram for Module diu_node_22v10

Device DIU_NODE_22V10



end of module diu_node_22v10

```
"*****************************************************************************
"  FILENAME:    EP320_TX_CLOCK.ABL      2 MHz AIPS I/O transmit clock
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Modified to match ISIO PC board requirements
"                                Separated out test vectors
"   B   11/17/88         TCT     Changed input pin out to match PC layout
"   C   11/21/88         TCT     Added M0 and M1 inputs
"                                Added OUTPUT_ENABLE
"                                Disable TXC while input_mode == RESET_INPUT
"*****************************************************************************

module ep320_tx_clock

flag    '-r2'

title  '68562 Transmit Clock Generator

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev:  10/31/88
 '

" rev           description

" This EPLD buffers the system clock and provides a transmit clock for the DUSCC
" chip which provides a non-square waveform to meet the falling TXC to TXD outpu
t
" valid delay of 240 ns.  The TXD clock will be low for 6 input clocks (375 ns)
" and high for two (125 ns).

"Declarations

        EP320_TX_CLOCK          device          'E0320';

    " define ABEL .. commands
        C,K,P,X,Z = .C.,.K.,.P.,.X.,.Z.;

    " inputs
        SYS_CLOCK       pin     1;      " 16 Mhz system clock

        !PIT_CS         pin     2;      " chip select from U100
        A6              pin     3;      " addr line 6 from ISIO card

        M0,M1           pin     4,5;    " DIU sim op mode input

    " outputs

        TXC,Q2,Q1,Q0    pin     12,13,14,15;
        TXC,Q2,Q1,Q0    istype  'pos, reg, feed_pin';
        TXC.EN          istype  'eqn';

        !U100_CS        pin     19;
        U100_CS         istype  'neg, com';

        !U20_CS         pin     18;
        U20_CS          istype  'neg, com';

        SPARE1          pin     16;
        SPARE1          istype  'pos, com';
```

```
        OUTPUT_ENABLE   pin     17;
        OUTPUT_ENABLE   istype  'pos, com, feed_pin';

    " states

        txc_ctr = [TXC,Q2..Q0];
"
        S0 = ^b0000;
        S1 = ^b0001;
        S2 = ^b0011;
        S3 = ^b0010;
        S4 = ^b0110;
        S5 = ^b0100;
        S6 = ^b1100;
        S7 = ^b1000;

        @INCLUDE        'INPUT_MODE.STATE'
equations
        SPARE1 = 0;

        OUTPUT_ENABLE = (input_mode != RESET_INPUT);
        TXC.EN = OUTPUT_ENABLE;

        U100_CS = PIT_CS & !A6; " select U100 when A6 = 0
        U20_CS  = PIT_CS &  A6; " select U20 when A6 = 1

state_diagram txc_ctr

        state S0:       goto S1;
        state S1:       goto S2;
        state S2:       goto S3;
        state S3:       goto S4;
        state S4:       goto S5;
        state S5:       goto S6;
        state S6:       goto S7;
        state S7:       goto S0;

" Comment out the following line to compile a production .JED file
"
"    @INCLUDE   'EP320_TX_CLOCK.TST'

end ep320_tx_clock
```

```
"*****************************************************************************
"  FILENAME:    EP320_TX_CLOCK.TST      2 MHz AIPS I/O transmit clock vectors
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Modified to match ISIO PC board requirements
"                                Separated out test vectors
"   B   11/21/88         TCT     Added test vectors for TXC output enable
"*****************************************************************************

test_vectors    ' Test Chip Selects '
        ([PIT_CS, A6] -> [U100_CS, U20_CS])

        [0, 0] -> [0, 0];
        [0, 1] -> [0, 0];
        [1, 0] -> [1, 0];
        [1, 1] -> [0, 1];

test_vectors    ' Set TXC Clock Generator to Known State'
        ([SYS_CLOCK, txc_ctr] -> txc_ctr)

        [P, S0] -> S0;


test_vectors    ' Test TXC Clock Generator'
                ([SYS_CLOCK, txc_ctr] -> txc_ctr)

        [C, S0] -> S1;
        [C, S1] -> S2;
        [C, S2] -> S3;
        [C, S3] -> S4;
        [C, S4] -> S5;
        [C, S5] -> S6;
        [C, S6] -> S7;
        [C, S7] -> S0;

test_vectors    ' Test TXC output enable'
        (input_mode -> [OUTPUT_ENABLE, TXC])

        NORMAL_INPUT  -> [1, X];
        NODE_INPUT    -> [1, X];
        MONITOR_INPUT -> [1, X];
        RESET_INPUT   -> [0, Z];
```

ABEL(tm) 3.00b - Document Generator 21-Nov-88 04:36 PM
68562 Transmit Clock Generator

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module ep320_tx_clock

Device EP320_TX_CLOCK

- Reduced Equations:

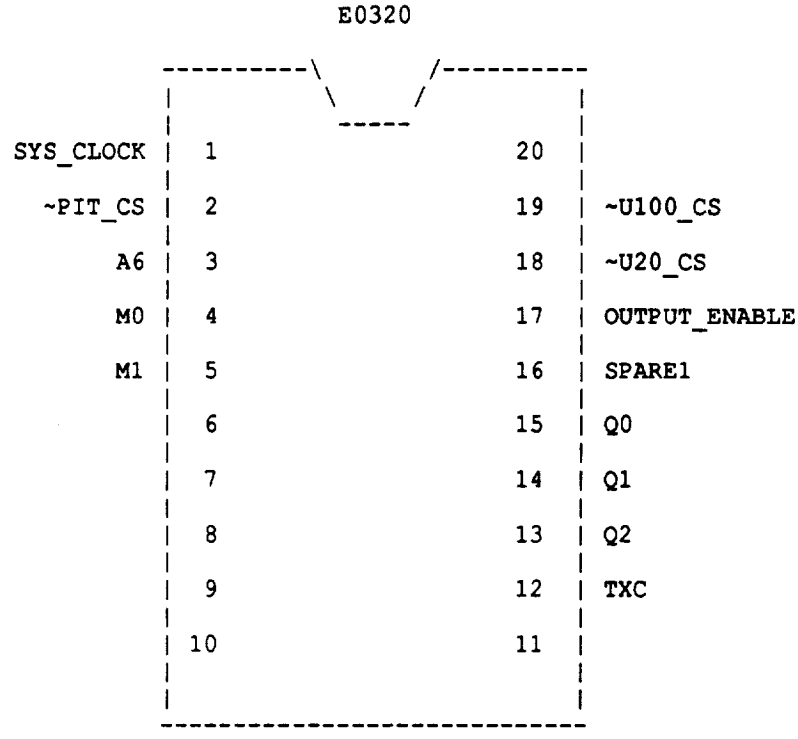
```
SPARE1 = (0);

OUTPUT_ENABLE = (!M0 # !M1);

enable TXC = (OUTPUT_ENABLE);

~U100_CS = !(!A6 & !~PIT_CS);

~U20_CS = !(A6 & !~PIT_CS);

TXC := (!Q0 & !Q1 & Q2);

Q2 := (!Q0 & Q2 & !TXC # !Q0 & Q1 & !TXC);

Q1 := (Q1 & !Q2 & !TXC # Q0 & !Q2 & !TXC);

Q0 := (!Q1 & !Q2 & !TXC);
```

ABEL(tm) 3.00b - Document Generator 21-Nov-88 04:36 PM
68562 Transmit Clock Generator

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Chip diagram for Module ep320_tx_clock

Device EP320_TX_CLOCK



end of module ep320_tx_clock

```
"*****************************************************************************
"  FILENAME:     ISIO_DELAY_GEN.ABL      Declarations unique to ISIO delay
"       DATE:    January 31, 1989
"         BY:    Tom Torkelson
"*****************************************************************************

module isio_delay_gen

flag '-r3','-t0'

title 'ISIO FTC Delay Generator EPLD for MC68230

BOEING ADVANCED SYSTEMS
Designed by:  T.C. Torkelson              Latest Revision:  31 JAN 89'

" This module is used with the MC68230 PIT to prevent the timer_register from
" changing when the timer_register is being read.  This module was designed
" with the consideration that the MOVEP instruction must be used to access
" the timer_register on the MC68230.
"
" The first byte is read by the MOVEP instruction is actually a dummy byte
" which is read as zero.  The CS for the dummy byte causes the EPLD to skip
" the next rising edge of the FTC, whether it occurs during the read of the
" timer or not.  The next rising and falling edges each generate a pulse to
" the 68230, making up for the swallowed rising edge.
"
" A limitation on the 68230 is that clock pulses must not be spaced closer
" than the input clock frequency of the chip / 8.  The ISIO 68230 is clocked
" at 7.38 Mhz, thus the minimum spacing between pulses is 1.08 usec. This
" works with the 4.125 usec FTC clock.

" declarations

        ISIO_DELAY_GEN device 'E0600';  "uses the Altera EP600 chip

" inputs unique to ISIO_DELAY_GEN
        !INH_1          pin 11;         " TICK inhibit, active low
        !INH_2          pin 10;         " TICK inhibit, active low
        INH_3           pin  9;         " TICK inhibit, active high

" get common code for delay generator

        @INCLUDE 'DELAY_GEN.INC'

end isio_delay_gen
```

```
"*****************************************************************************
"  FILENAME:    DELAY_GEN.INC   FTC pulse delay generator common logic
"      DATE:    January 31, 1989
"        BY:    Art Pannek
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"=============================================================================
"   A   10/31/88         TCT     Placed test vectors separate .TST file
"                                Changed pin allocation for pc board
"                                Changed INHB_A & _B pol to active low
"                                Added INHB_C, active high
"                                Changed pin numbers of inhibits for ISIO
"
"   B    1/30/89         TCT     Changed design of EPLD to always swallow
"                                one rising edge, then make it up with a
"                                falling edge later
"
"   C    1/31/89         TCT     Changed state progression, separated out .abl
"                                code common to ISIO_DELAY_GEN & OPIO_DELAY_GEN
"
"   D    2/ 2/89         TCT     Changed FTC latch to FTC D flop clocked
"                                async by falling edge of CLK3
"*****************************************************************************


    " define ABEL .. commands
        C, K, P, X = .C., .K., .P., .X.;
        H, L = 1, 0;

    " inputs

        CLK1            pin  1;         " MC68230 clock
        CLK2            pin 13;         " MC68230 clock
        CLK3            pin 23;         " MC68230 clock

        !CS             pin  2;         " CS active low to select MC68230

        RS1             pin  4;         " MC68230 register select bits
        RS2             pin  5;
        RS3             pin  6;
        RS4             pin  7;
        RS5             pin  8;

        FTC             pin 14;         "Fault Tolerant Clock; 8 MHz / 33

    " outputs

        FTC_TICK        pin  3;         " Tick output to 68230

        CNTRX_SELECT    pin 22;         " CS of cntrx register detected
            CNTRX_SELECT        istype  'pos, reg_D, feed_reg';
            CNTRX_SELECT.C      istype  'eqn';  " async clock
            CNTRX_SELECT.AR     istype  'eqn';  " async reset

        CNTRX_SELECT_LATCH      pin 21;
            CNTRX_SELECT_LATCH  istype  'pos, com, feed_pin';

        FTC_LATCH       pin 20;
" -- Rev D  TCT 2/2/89
            FTC_LATCH           istype  'pos, reg_D, feed_reg';
            FTC_LATCH.C         istype  'eqn';
```

```
"          FTC_LATCH              istype  'pos, com, feed_pin';

        FTC_LATCH_DELAY pin 19;
            FTC_LATCH_DELAY       istype  'pos, reg_D, feed_reg';

        SKIP0, SKIP1    pin 17, 18;
            SKIP0, SKIP1          istype  'pos, reg_D, feed_reg';

        INH_LATCH       pin  16;
            INH_LATCH             istype  'pos, com, feed_pin';

        TICK            pin  15;        " outputs pulses w/o regard to INH
            TICK                  istype  'pos, reg_D, feed_reg';

    " define states

        rs = [RS5..RS1];            " input register select
            RS_CNTRX = ^b10110;           " select dummy
            RS_CNTRH = ^b10111;           " select high byte
            RS_CNTRM = ^b11000;           " select middle byte
            RS_CNTRL = ^b11001;           " select low byte

        inh = [INH_1,INH_2,INH_3];        " inhibit
            TICK_ENABLE = ^b000;

        clk = [CLK1,CLK2,CLK3];           "Clock the same inputs
            CLK_C = [C,C,C];                "Clk_Group is Clocked
            CLK_H = [H,H,H];                "Clk_Group is High
            CLK_L = [L,L,L];                "Clk_Group is Low

        ftc = [FTC_LATCH, FTC_LATCH_DELAY];
            FTC_RISE_EDGE = ^b10;           " rising edge of FTC
            FTC_FALL_EDGE = ^b01;           " falling edge of FTC

        skip = [SKIP1,SKIP0];           " FTC edge skip states
            SKIP_RESET    = ^b00;           " pass + edges
            SKIP_INHIBIT = ^b01;            " inhibit all
            SKIP_PASS_HI = ^b11;            " pass + edge
            SKIP_PASS_LO = ^b10;            " pass - edge

    " macros

        " latch on gate level, pass thru on !gate level

        LATCH macro (out, in, gate)
            {?out = ?out & ?gate # ?in & !?gate;}

equations

    CNTRX_SELECT    := (rs == RS_CNTRX);
    CNTRX_SELECT.C  = CS;                 " clock on leading edge of CS

    " The following is really not required unless only one CS is received.
    CNTRX_SELECT.AR = (skip == SKIP_PASS_LO) & !FTC_LATCH & FTC_LATCH_DELAY;

    " synchronize with input clock, hold when clock low, pass clock high

    LATCH (CNTRX_SELECT_LATCH, CNTRX_SELECT, !CLK3)

" -- Rev D  TCT  2/2/89
"    LATCH (FTC_LATCH, FTC, !CLK3)
```

```
    LATCH (INH_LATCH, (INH_1 # INH_2 # INH_3), !CLK3)

" -- Rev D TCT  2/2/89
    FTC_LATCH  := FTC;
    FTC_LATCH.C = !CLK3;        " clock on falling edge of CLK3

    " FTC delayed one input clock pulse
    FTC_LATCH_DELAY := FTC_LATCH;

    " FTC tick conditioned by skip states and inhibits
    FTC_TICK := !INH_LATCH &
                (   (skip == SKIP_RESET)    & (ftc == FTC_RISE_EDGE)
                  # (skip == SKIP_PASS_HI)  & (ftc == FTC_RISE_EDGE)
                  # (skip == SKIP_PASS_LO)  & (ftc == FTC_FALL_EDGE)
                );

    " TICK output for test purposes, not affected by INH

    TICK :=       (skip == SKIP_RESET)    & (ftc == FTC_RISE_EDGE)
                # (skip == SKIP_PASS_HI)  & (ftc == FTC_RISE_EDGE)
                # (skip == SKIP_PASS_LO)  & (ftc == FTC_FALL_EDGE)
                ;

state_diagram   skip

    " This state machine is clocked by the system clock.  After the
    " initial state change, state changes only occur on edges of FTC.
    "
    " The state machine inhibits an output pulse on the first rising
    " edge following the selection of CNTRX.  The second rising edge
    " and the following falling edge both generate output pulses.

    state SKIP_RESET:   if CNTRX_SELECT_LATCH then SKIP_INHIBIT
                        else SKIP_RESET;

    state SKIP_INHIBIT: if (ftc == FTC_RISE_EDGE) then SKIP_PASS_HI
                        else SKIP_INHIBIT;

    state SKIP_PASS_HI: if (ftc == FTC_RISE_EDGE) then SKIP_PASS_LO
                        else SKIP_PASS_HI;

    state SKIP_PASS_LO: if (ftc == FTC_FALL_EDGE) then SKIP_RESET
                        else SKIP_PASS_LO;
```

ABEL(tm) 3.00b - Document Generator 02-Feb-89 04:59 PM
ISIO FTC Delay Generator EPLD for MC68230

BOEING ADVANCED SYSTEMS
Designed by: T.C. Torkelson Latest Revision: 31 JAN 89
Equations for Module isio_delay_gen

Device ISIO_DELAY_GEN

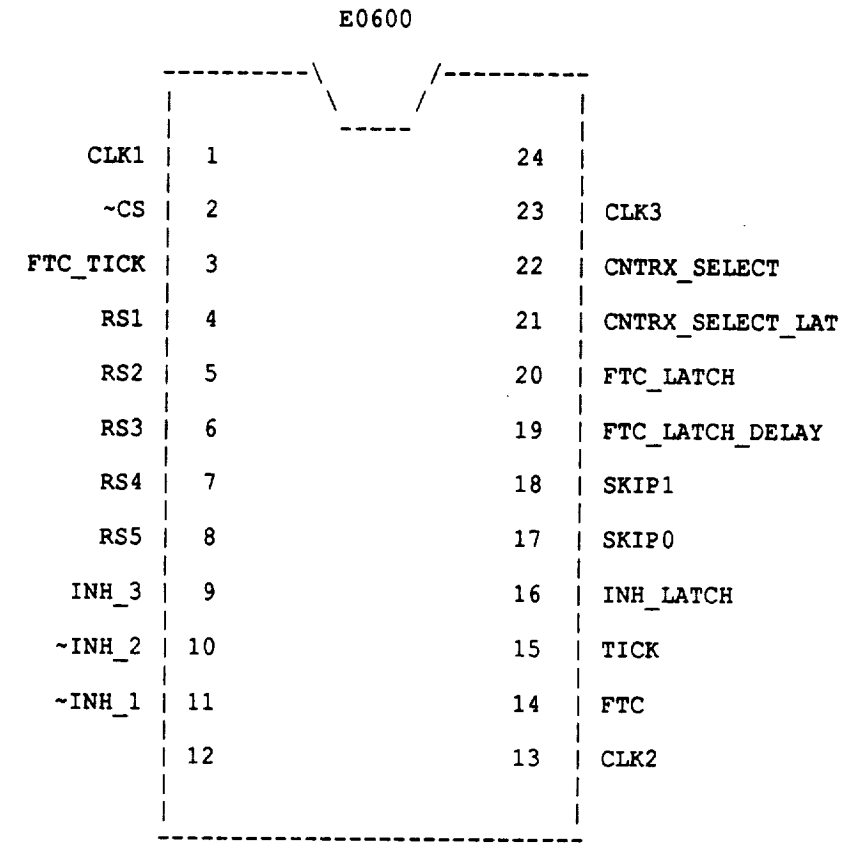
- Reduced Equations:

```
CNTRX_SELECT := (!RS1 & RS2 & RS3 & !RS4 & RS5);

_CNTRX_SELECT_C = (!~CS);

_CNTRX_SELECT_RE = (!FTC_LATCH & FTC_LATCH_DELAY & !SKIP0 & SKIP1);

CNTRX_SELECT_LATCH = (CLK3 & CNTRX_SELECT # !CLK3 & CNTRX_SELECT_LATCH);

INH_LATCH = (CLK3 & INH_3
             # CLK3 & !~INH_2
             # CLK3 & !~INH_1
             # !CLK3 & INH_LATCH);


FTC_LATCH := (FTC);

_FTC_LATCH_C = (!CLK3);

FTC_LATCH_DELAY := (FTC_LATCH);

FTC_TICK := (!FTC_LATCH & FTC_LATCH_DELAY & !INH_LATCH & !SKIP0 & SKIP1

             # FTC_LATCH & !FTC_LATCH_DELAY & !INH_LATCH & SKIP0 & SKIP1

             # FTC_LATCH & !FTC_LATCH_DELAY & !INH_LATCH & !SKIP0 &
            !SKIP1);

TICK := (!FTC_LATCH & FTC_LATCH_DELAY & !SKIP0 & SKIP1
         # FTC_LATCH & !FTC_LATCH_DELAY & SKIP0 & SKIP1
         # FTC_LATCH & !FTC_LATCH_DELAY & !SKIP0 & !SKIP1);


SKIP1 := (!FTC_LATCH_DELAY & SKIP1
          # FTC_LATCH & SKIP1
          # SKIP0 & SKIP1
          # FTC_LATCH & !FTC_LATCH_DELAY & SKIP0);


SKIP0 := (FTC_LATCH_DELAY & SKIP0
          # !FTC_LATCH & SKIP0
          # SKIP0 & !SKIP1
          # CNTRX_SELECT_LATCH & !SKIP1);
```

ABEL(tm) 3.00b - Document Generator 02-Feb-89 04:59 PM
ISIO FTC Delay Generator EPLD for MC68230

BOEING ADVANCED SYSTEMS
Designed by: T.C. Torkelson Latest Revision: 31 JAN 89
Chip diagram for Module isio_delay_gen

Device ISIO_DELAY_GEN



end of module isio_delay_gen

```
"*****************************************************************************
"  FILENAME:    LED_DRIVER.ABL          FTC & misc control EPLD
"       DATE:   November 1, 1988
"         BY:   T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"    A    12/20/88      TCT     changed Gx pinout to match PC board
"*****************************************************************************

module led_driver

flag    '-r3','-t1'

title  'ISIO daughter board LED drivers

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev:  12/20/88
 '

"Declarations:

        LED_DRIVER      device          'E0320';

    " define ABEL .. commands
        C,K,P,X = .C.,.K.,.P.,.X.;

    " inputs:

        C3,C2,C1,C0     pin     6,7,8,9;        " color: 1 = red, 0 = grn
        S3,S2,S1,S0     pin     2,3,4,5;        " select: 1 = on, 0 = off

    " outputs:
        R3,R2,R1,R0             pin     19,18,17,16;    " low for RED
        R3,R2,R1,R0             istype  'neg, com';

        G3,G2,G1,G0             pin     12,13,14,15;    " low for GRN
        G3,G2,G1,G0             istype  'neg, com';

    " states, etc.

        led_red = [R3..R0];
        led_grn = [G3..G0];
        color   = [C3..C0];
        select  = [S3..S0];

        RED_OUT = [0,0,0,0];
        GRN_OUT = [1,1,1,1];
        OFF_OUT = RED_OUT;

        RED_IN = [1,1,1,1];
        GRN_IN = [0,0,0,0];

        LED_ON  = [0,0,0,0];
        LED_OFF = [1,1,1,1];

equations
        led_red = !color & !select;
        led_grn = color & !select;

" Comment out the following command to compile production .JED files
"
```

```
"    @INCLUDE    'LED_DRIVER.TST'
    end led_driver
```

```
"*****************************************************************************
"  FILENAME:     LED_DRIVER.TST          FTC & misc control EPLD
"       DATE:    November 1, 1988
"         BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"
"*****************************************************************************

test_vectors     ' Test LED driver '
        ([color, select] -> [led_red, led_grn])

        [RED_IN, LED_OFF] -> [OFF_OUT,  OFF_OUT];
        [RED_IN,  LED_ON] -> [RED_OUT, !RED_OUT];
        [GRN_IN, LED_OFF] -> [OFF_OUT,  OFF_OUT];
        [GRN_IN,  LED_ON] -> [GRN_OUT, !GRN_OUT];
```

ABEL(tm) 3.00b - Document Generator 03-Jan-89 02:51 PM
ISIO daughter board LED drivers

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 12/20/88

Equations for Module led_driver

Device LED_DRIVER

- Reduced Equations:

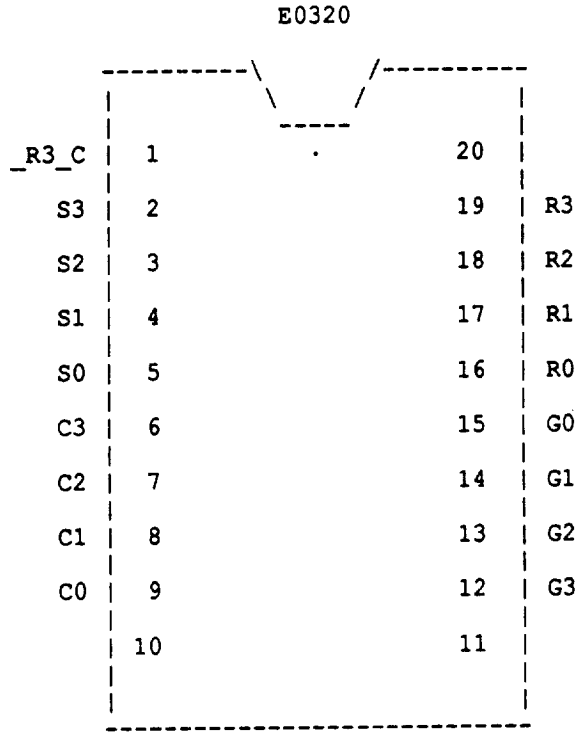
```
R3 = !(S3 # C3);

R2 = !(S2 # C2);

R1 = !(S1 # C1);

R0 = !(S0 # C0);

G3 = !(S3 # !C3);

G2 = !(S2 # !C2);

G1 = !(S1 # !C1);

G0 = !(S0 # !C0);
```

ABEL(tm) 3.00b - Document Generator 03-Jan-89 02:51 PM
ISIO daughter board LED drivers

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 12/20/88

Chip diagram for Module led_driver

Device LED_DRIVER



end of module led_driver

DOCUMENTATION PACKAGE E: FAULT INSERTION AND CONTROL WIRE WRAP BOARD

Subject: IAPSA II Wire Wrap Card Fabrication Notes
By: T.C. Torkelson

Date: June 14, 1989

Introduction:

The wire wrap board which was fabricated for the small scale system portion of the IAPSA II contract is fully documented in schematics and layout drawings. The board was produced from that documentation.

Fabrication Notes:

1. A distinction is been made on the schematics between the symbol for common (a triangle) and GND. Connections to GND are made to the wire_wrap board backplane with solder preforms. Connections to the symbol for common are made with wire wrap connections to dedicated ground pins on the wire wrap board.

2. A distinction has been made on the schematics between +5 and VCC. Connections to VCC are made to the wire_wrap board frontplane with solder preforms. Connections to the symbol for +5 are made with wire wrap connections to dedicated power pins on the wire wrap board.

3. No assembly drawing was produced to show front panel construction. The front panels are assembled in a manner similar to the DIU front panel. Its drawings can be used as a guide for the wire wrap front panel construction.

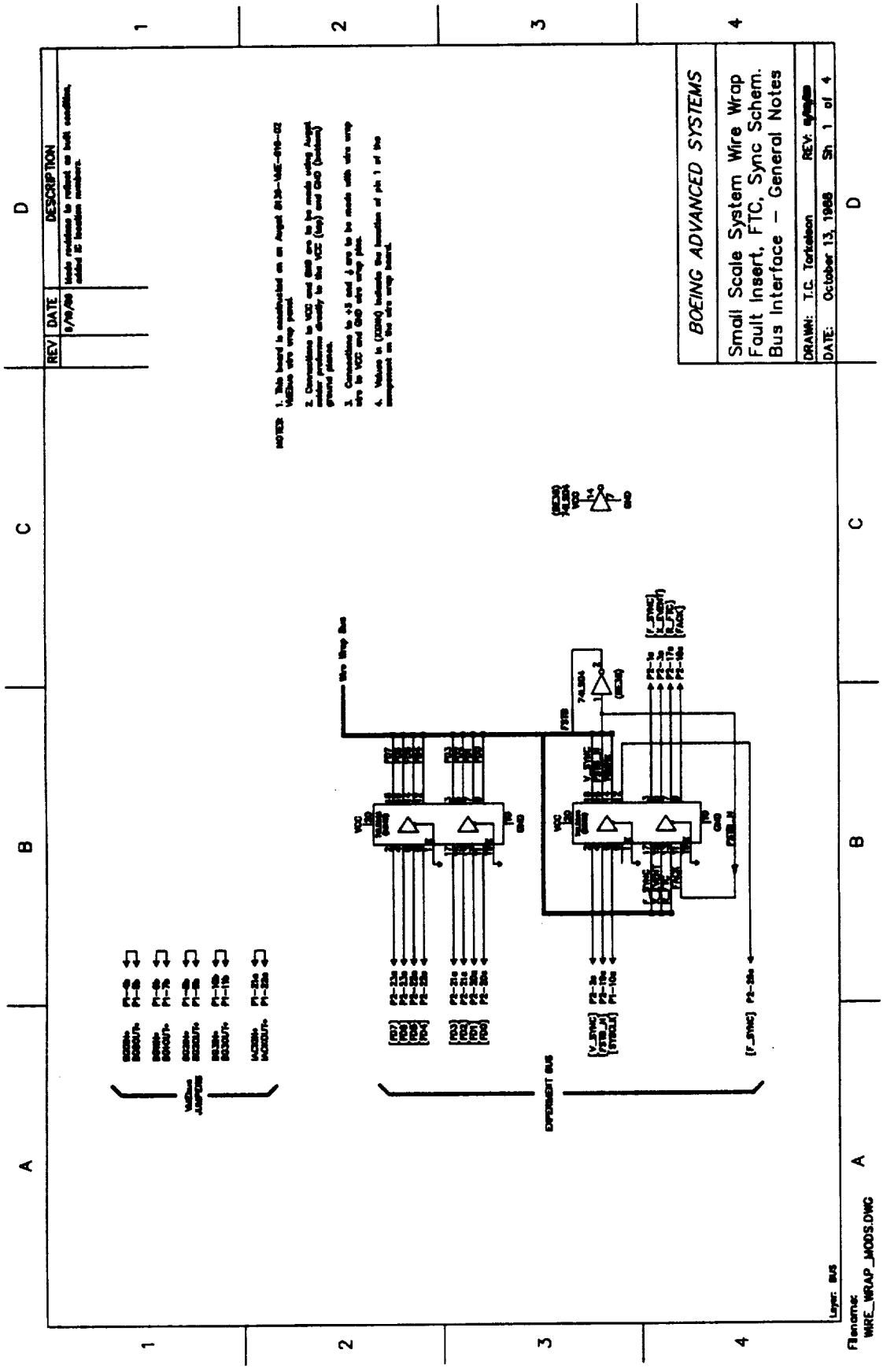
4. The ribbon cables which connect the wire wrap board with the front panel boards must be routed and split to avoid interference with other boards in the VME chassis. Two cables originate on the pin side of the board; one on the front.

The cable on the front of the board is most likely to cause interference. To shield it from other boards, a piece of perforated Vector board was cut which spans the space from the BG45 to the AD45 connectors. This board is placed over the pins for these connectors and held in place with wire wraps on several pins.

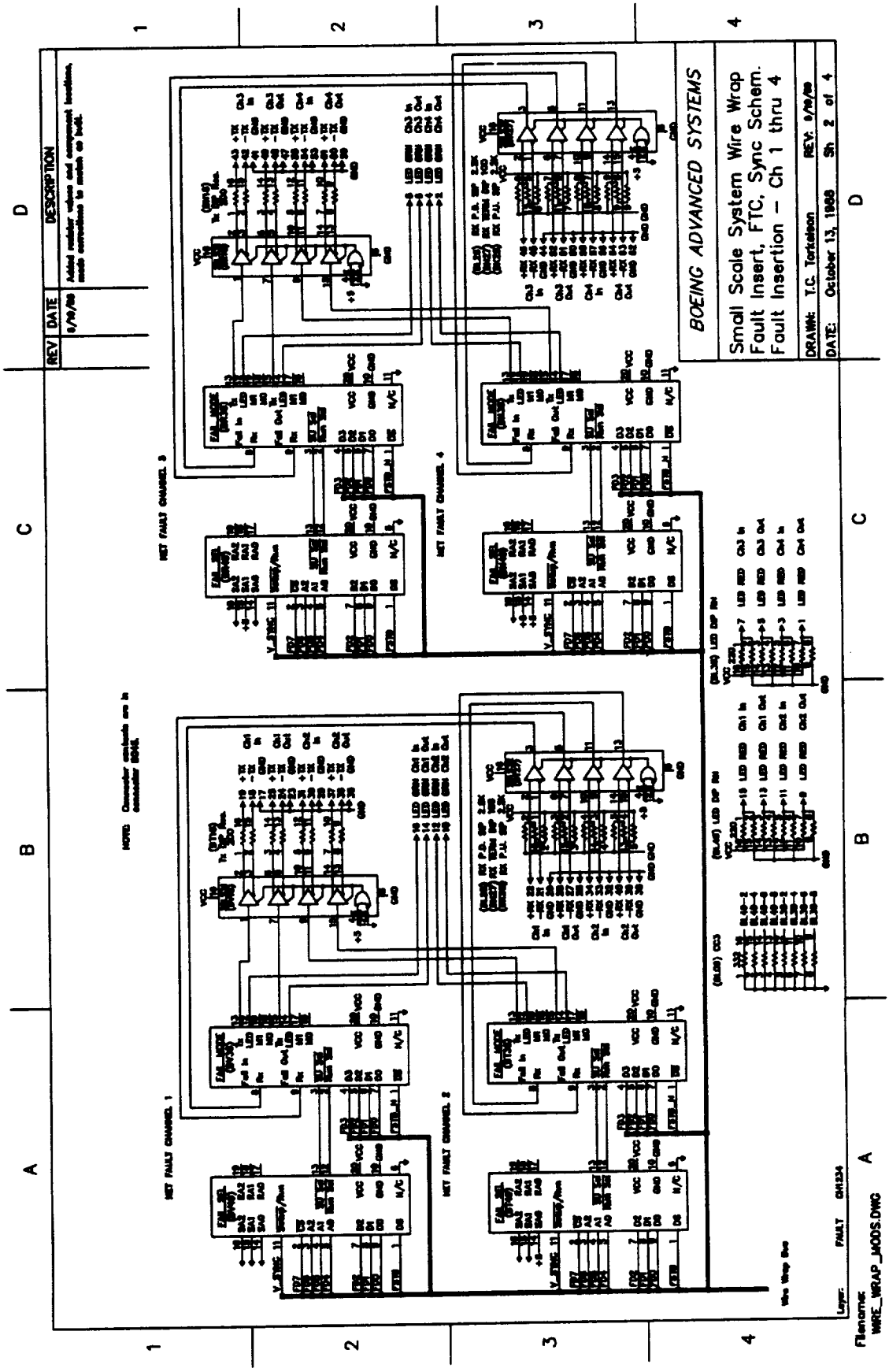
Care must be taken that the wire wraps which hold the shield on do not short out any connector pins.

PRECEDING PAGE BLANK NOT FILMED

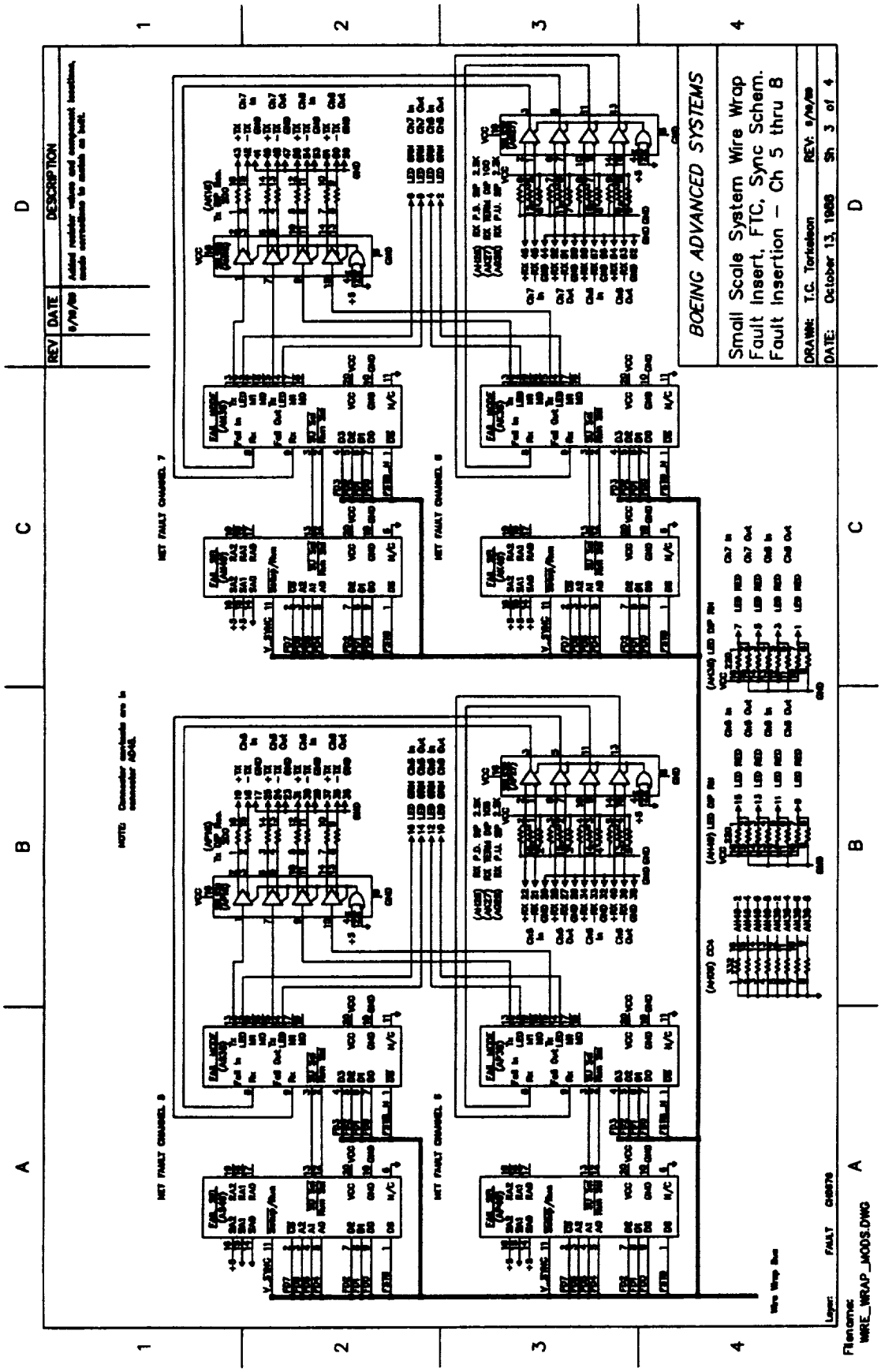
PAGE J-96 INTENTIONALLY BLANK



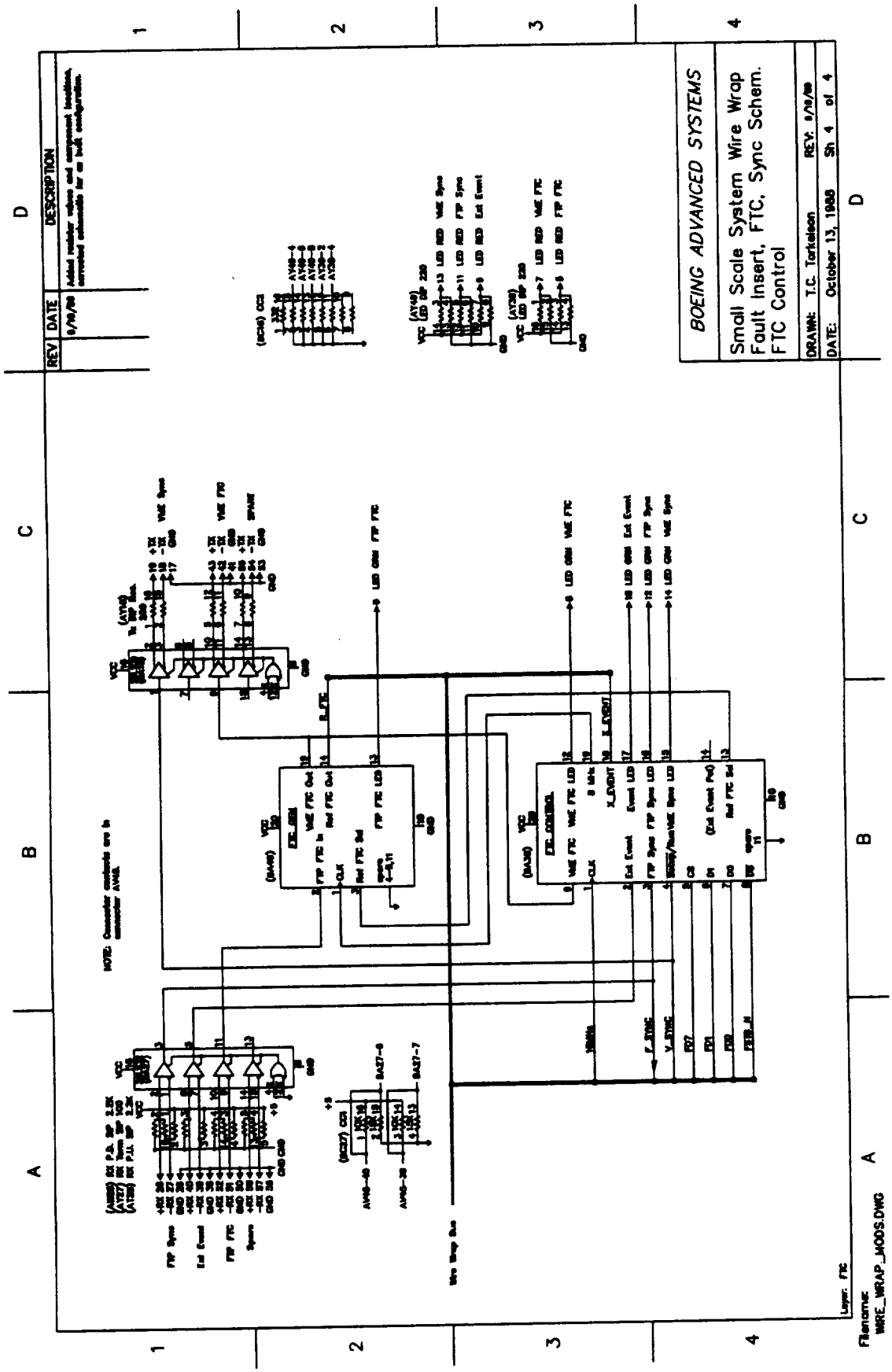
ORIGINAL PAGE IS
OF POOR QUALITY



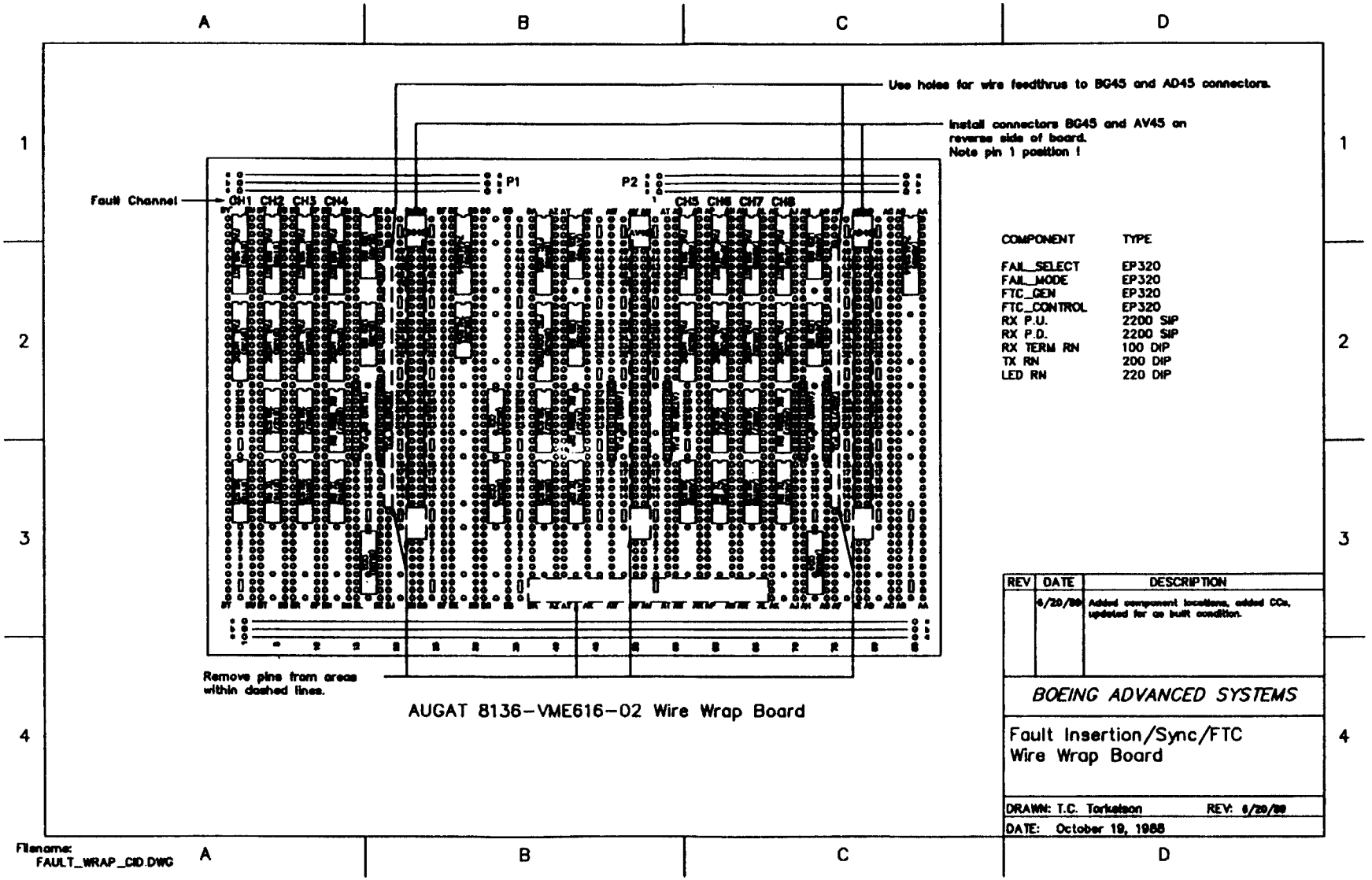
ORIGINAL PAGE IS
OF POOR QUALITY



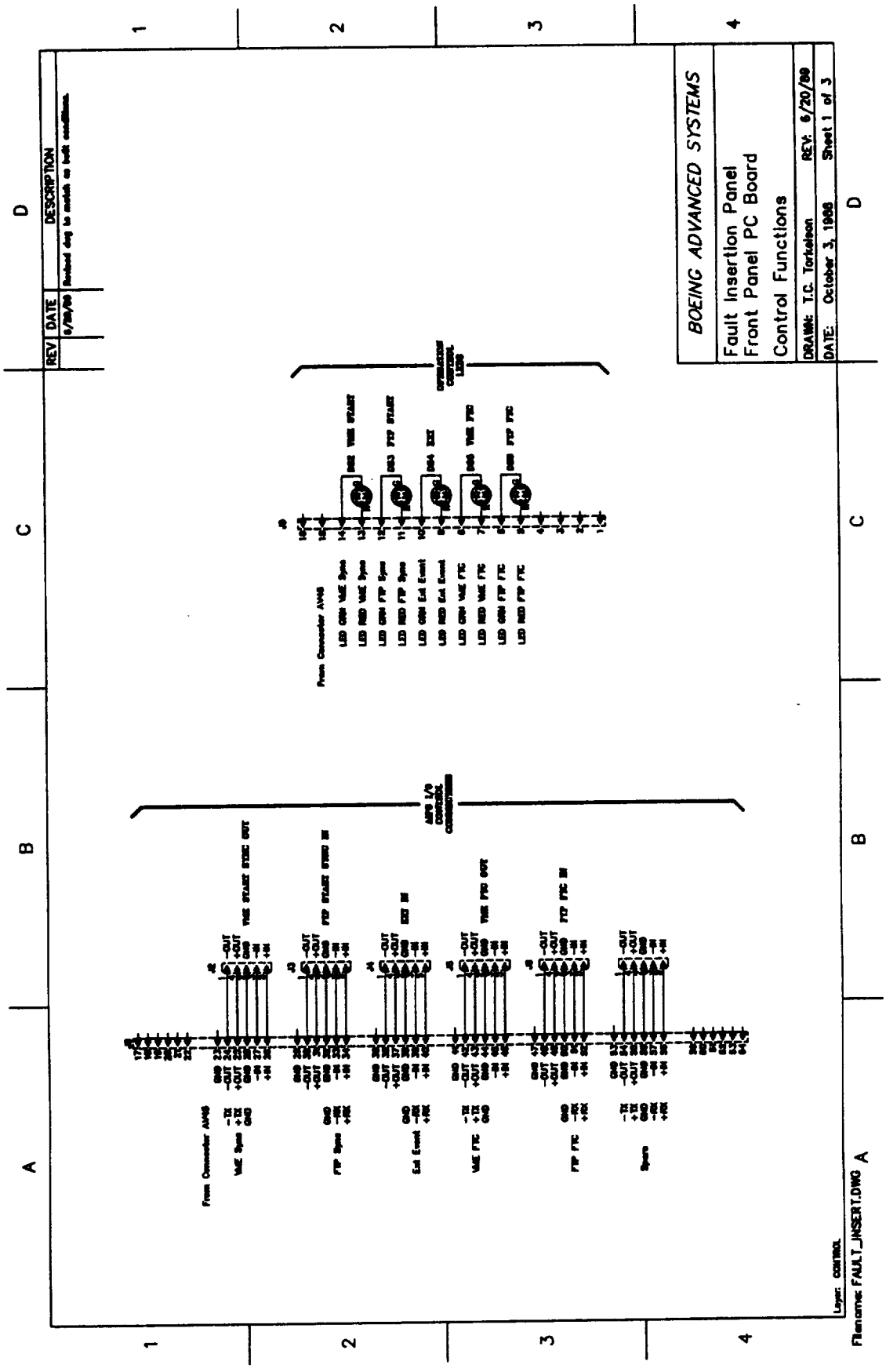
ORIGINAL PAGE IS
OF POOR QUALITY



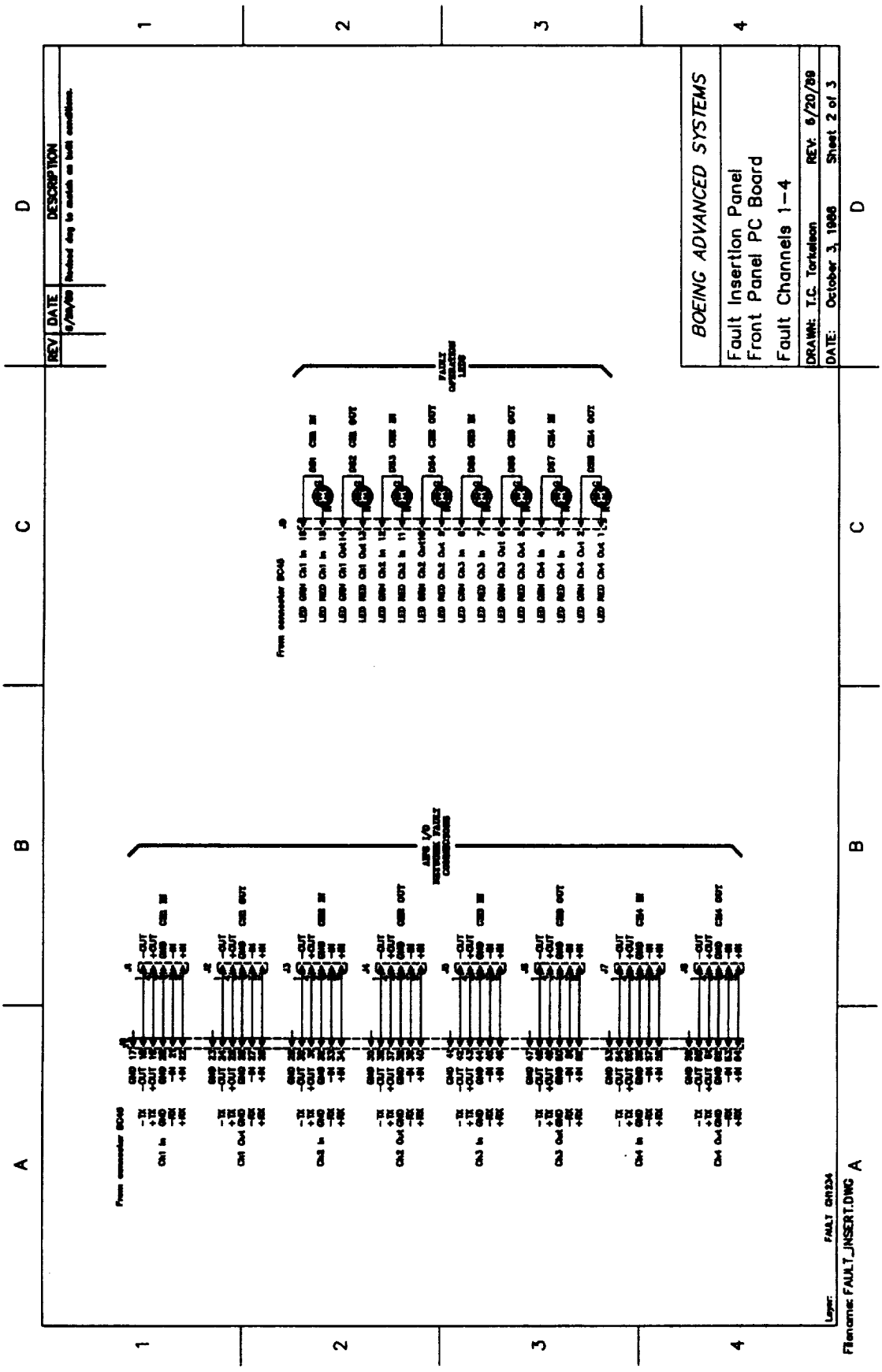
ORIGINAL PAGE IS
OF POOR QUALITY



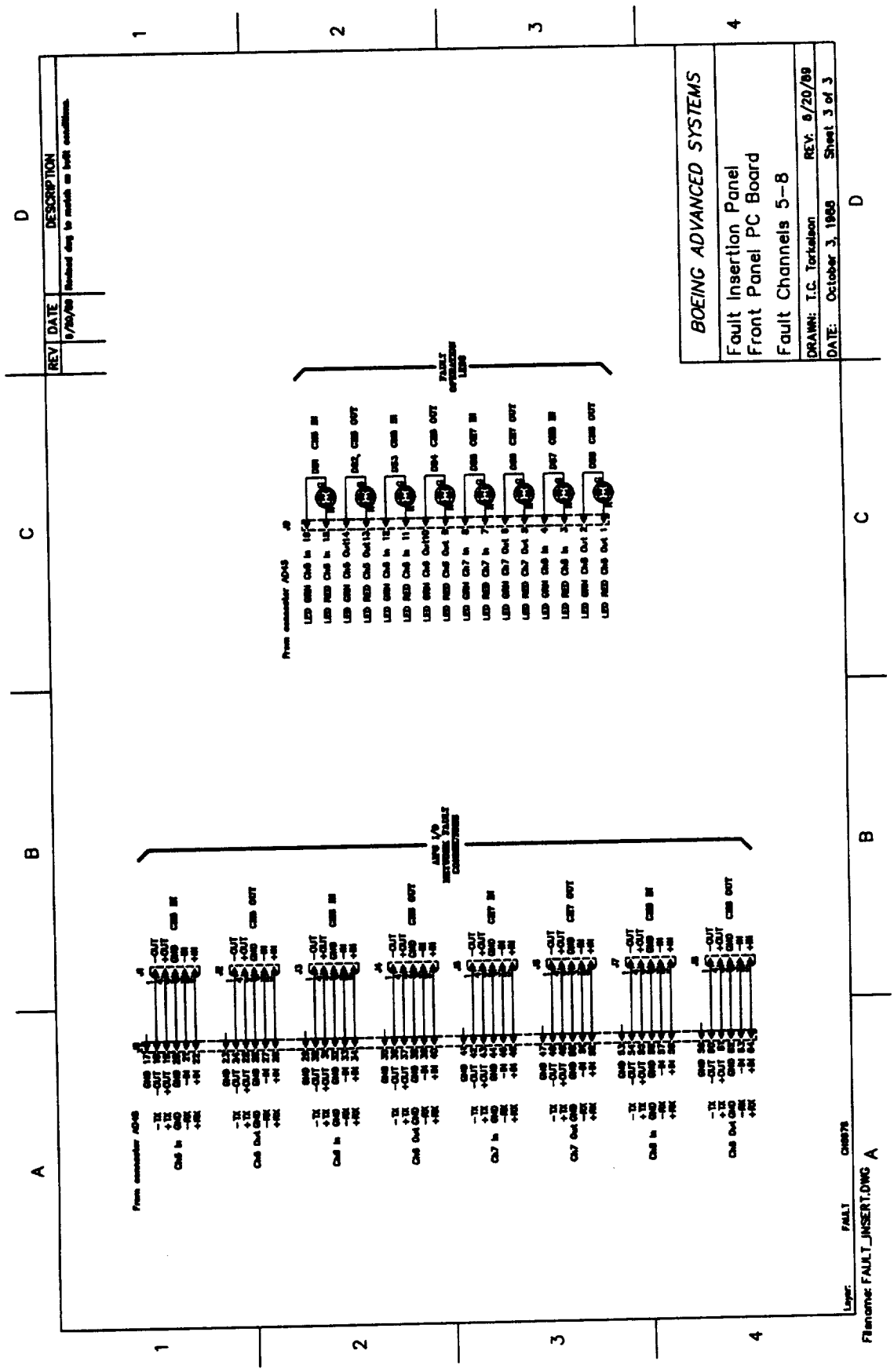
ORIGINAL PAGE IS
OF POOR QUALITY



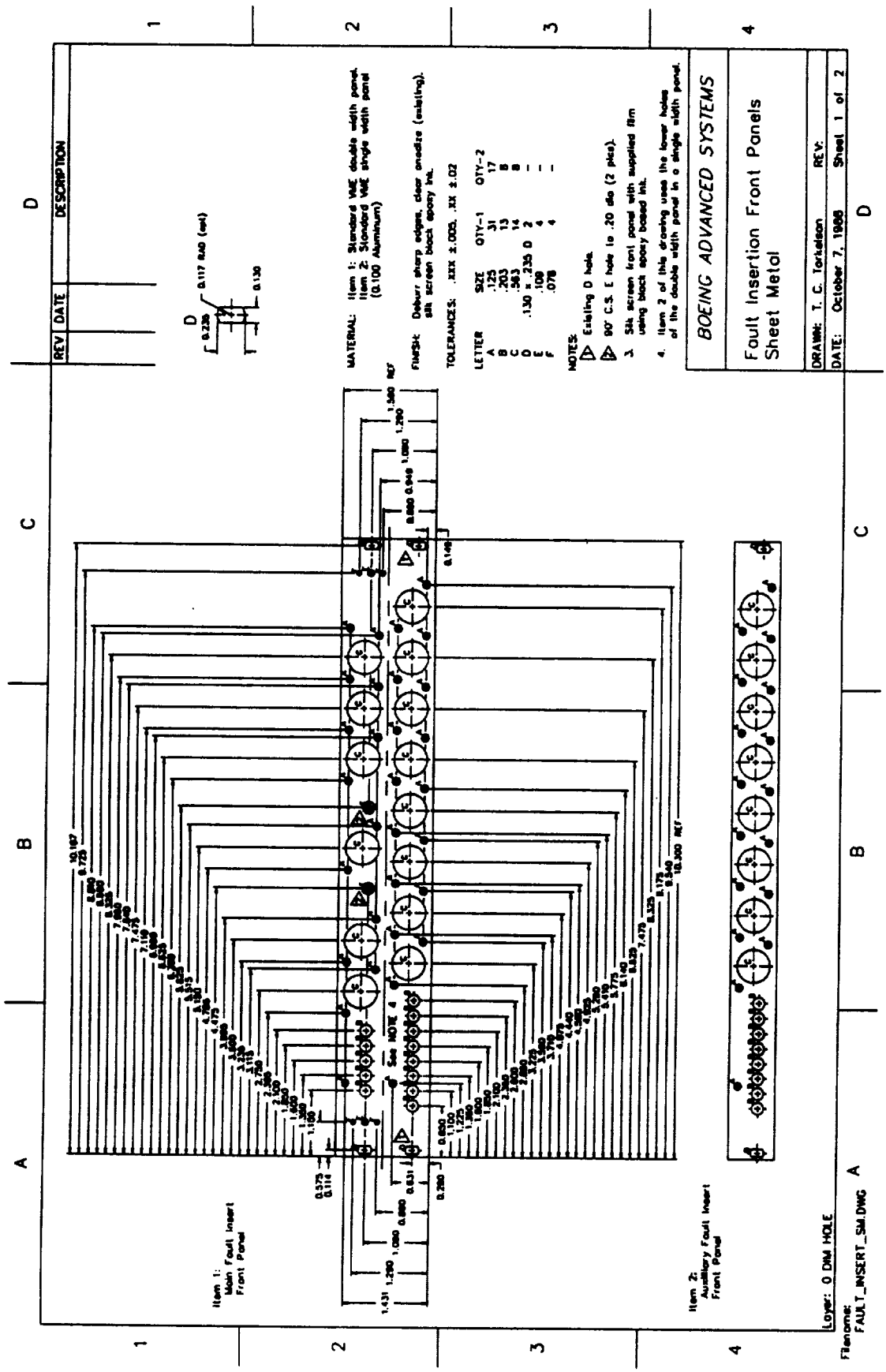
ORIGINAL PAGE IS
OF POOR QUALITY



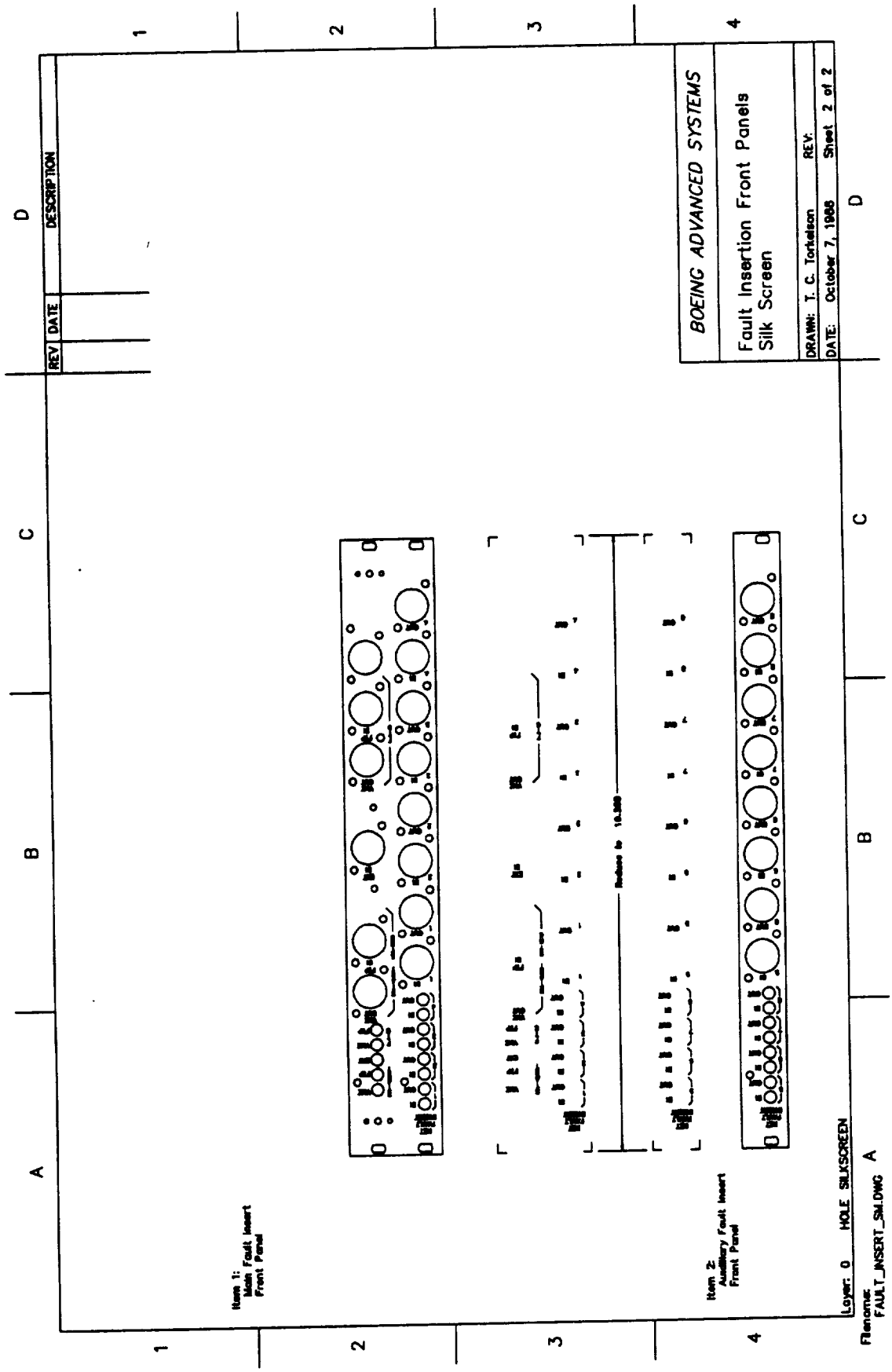
ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

```
"*****************************************************************************
"  FILENAME:    NET_FAIL_SELECT.ABL     Network fault insert select
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Changed pin designations to match schematic
"                                Separated out test vectors
"*****************************************************************************
"
" NOTE:  A3 input is the !CS input of the chip on the schematic.
"
module net_fail_select

flag    '-r3','-t2'

title   'AIPS I/O Network Failure Insertion Select EPLD

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev: 10/31/88
 '
"   REV         BY      DESCRIPTION
" 8/30/88       TCT     changed to match NET_FAIL.DWG
"
"Declarations:

        NET_FAIL_SELECT device  'E0320';

    " define ABEL .. commands
        C, K, P, X, Z = .C., .K., .P., .X., .Z.;


    " inputs

        DS                      pin     1;

        !CS,A2,A1,A0            pin     2,3,4,5;
        D2,D1,D0                pin     7,8,9;

        !Setup                  pin     11;             " !Setup / Run

        SA2,SA1,SA0             pin     16,15,14;

    " outputs

        RA2,RA1,RA0             pin     19,18,17;               " RA3 unused
        RA2,RA1,RA0             istype  'pos, reg, feed_pin';

        !Run_Sel                pin     12;
        !Run_Sel                istype  'neg, com';

        !Setup_Sel              pin     13;
        !Setup_Sel              istype  'neg, com, feed_or';

    " sets

        setup_addr_in   = [SA2..SA0];
        setup_addr      = [0,SA2..SA0];         " !CS must be 0
        run_addr_latch  = [RA2..RA0];
        run_addr        = [0,RA2..RA0];         " !CS must be 0
        addr            = [!CS,A2..A0];
```

```
        data            = [D2..D0];

    " macros:

        SETUP_SELECT macro {(Setup & (addr == setup_addr))};

        RUN_SELECT macro {(!Setup & (addr == run_addr))};

equations

        Run_Sel = RUN_SELECT;
        Setup_Sel = SETUP_SELECT;

        run_addr_latch := data & Setup_Sel # run_addr_latch & !Setup_Sel;

" Comment out the following line to compile production .JED files
"
"    @INCLUDE   'NET_FAIL_SELECT.TST'

end net_fail_select
```

```
"*****************************************************************************
"  FILENAME:    NET_FAIL_SELECT.TST     Network fault insert select vectors
"       DATE:   October 31, 1988
"         BY:   T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Changed pin designations to match schematic
"                                Separated out test vectors
"*****************************************************************************

test_vectors    'Test setup select logic'
        ([Setup, addr, setup_addr_in] -> Setup_Sel)

        " Test select with !CS = 0

            [0, ^o00, 0] -> 0;
            [1, ^o00, 0] -> 1;

            [0, ^o01, 1] -> 0;
            [1, ^o01, 1] -> 1;

            [0, ^o02, 2] -> 0;
            [1, ^o02, 2] -> 1;

            [0, ^o03, 3] -> 0;
            [1, ^o03, 3] -> 1;

            [0, ^o04, 4] -> 0;
            [1, ^o04, 4] -> 1;

            [0, ^o05, 5] -> 0;
            [1, ^o05, 5] -> 1;

            [0, ^o06, 6] -> 0;
            [1, ^o06, 6] -> 1;

            [0, ^o07, 7] -> 0;
            [1, ^o07, 7] -> 1;

       " Test select with !CS = 0

            [0, ^o10, 0] -> 0;
            [1, ^o10, 0] -> 0;

            [0, ^o11, 1] -> 0;
            [1, ^o11, 1] -> 0;

            [0, ^o12, 2] -> 0;
            [1, ^o12, 2] -> 0;

            [0, ^o13, 3] -> 0;
            [1, ^o13, 3] -> 0;

            [0, ^o14, 4] -> 0;
            [1, ^o14, 4] -> 0;

            [0, ^o15, 5] -> 0;
            [1, ^o15, 5] -> 0;

            [0, ^o16, 6] -> 0;
            [1, ^o16, 6] -> 0;
```

```
        [0, ^o17, 7] -> 0;
        [1, ^o17, 7] -> 0;

    " Test a few selects with addr != setup_addr

        [0, ^o00, 1] -> 0;
        [1, ^o00, 1] -> 0;

        [0, ^o01, 2] -> 0;
        [1, ^o01, 2] -> 0;

        [0, ^o02, 3] -> 0;
        [1, ^o02, 3] -> 0;

        [0, ^o03, 4] -> 0;
        [1, ^o03, 4] -> 0;

        [0, ^o04, 5] -> 0;
        [1, ^o04, 5] -> 0;

        [0, ^o05, 6] -> 0;
        [1, ^o05, 6] -> 0;

        [0, ^o06, 7] -> 0;
        [1, ^o06, 7] -> 0;

        [0, ^o07, 0] -> 0;
        [1, ^o07, 0] -> 0;

test_vectors    'Test run select logic'
        ([DS, Setup, addr, setup_addr_in, data] -> [run_addr_latch, Run_Sel])

    " Test operation of run_addr latch and Run_Sel

        [C, 1, 0, 0, 0] -> [0, 0];
        [X, 0, 0, X, X] -> [0, 1];
        [X, 1, 0, X, X] -> [0, 0];

        [C, 0, 0, 0, 1] -> [0, 1];
        [C, 1, 0, 0, 1] -> [1, 0];
        [X, 0, 1, X, X] -> [1, 1];
        [X, 1, 1, X, X] -> [1, 0];

        [C, 0, 0, 0, 2] -> [1, 0];
        [C, 1, 0, 0, 2] -> [2, 0];
        [X, 0, 2, X, X] -> [2, 1];
        [X, 1, 2, X, X] -> [2, 0];

        [C, 0, 0, 0, 3] -> [2, 0];
        [C, 1, 0, 0, 3] -> [3, 0];
        [X, 0, 3, X, X] -> [3, 1];
        [X, 1, 3, X, X] -> [3, 0];

        [C, 0, 0, 0, 4] -> [3, 0];
        [C, 1, 0, 0, 4] -> [4, 0];
        [X, 0, 4, X, X] -> [4, 1];
        [X, 1, 4, X, X] -> [4, 0];

        [C, 0, 0, 0, 5] -> [4, 0];
        [C, 1, 0, 0, 5] -> [5, 0];
```

```
[X, 0, 5, X, X] -> [5, 1];
[X, 1, 5, X, X] -> [5, 0];

[C, 0, 0, 0, 6] -> [5, 0];
[C, 1, 0, 0, 6] -> [6, 0];
[X, 0, 6, X, X] -> [6, 1];
[X, 1, 6, X, X] -> [6, 0];

[C, 0, 0, 0, 7] -> [6, 0];
[C, 1, 0, 0, 7] -> [7, 0];
[X, 0, 7, X, X] -> [7, 1];
[X, 1, 7, X, X] -> [7, 0];

[C, 0, 0, 0, 0] -> [7, 0];
```

ABEL(tm) 3.00b - Document Generator 03-Jan-89 03:38 PM
AIPS I/O Network Failure Insertion Select EPLD

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Equations for Module net_fail_select

Device NET_FAIL_SELECT

- Reduced Equations:

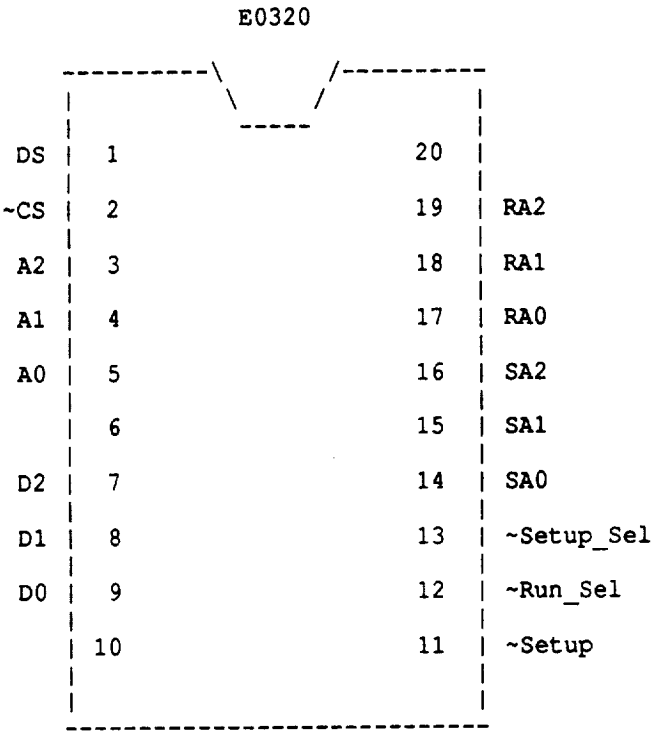
```
~Run_Sel = !(A0 & A1 & A2 & !~CS & RA0 & RA1 & RA2 & ~Setup
             # A0 & A1 & !A2 & !~CS & RA0 & RA1 & !RA2 & ~Setup
             # A0 & !A1 & A2 & !~CS & RA0 & !RA1 & RA2 & ~Setup
             # A0 & !A1 & !A2 & !~CS & RA0 & !RA1 & !RA2 & ~Setup
             # !A0 & A1 & A2 & !~CS & !RA0 & RA1 & RA2 & ~Setup
             # !A0 & A1 & !A2 & !~CS & !RA0 & RA1 & !RA2 & ~Setup
             # !A0 & !A1 & A2 & !~CS & !RA0 & !RA1 & RA2 & ~Setup
             # !A0 & !A1 & !A2 & !~CS & !RA0 & !RA1 & !RA2 & ~Setup);


~Setup_Sel = !(A0 & A1 & A2 & !~CS & SA0 & SA1 & SA2 & !~Setup
               # A0 & A1 & !A2 & !~CS & SA0 & SA1 & !SA2 & !~Setup
               # A0 & !A1 & A2 & !~CS & SA0 & !SA1 & SA2 & !~Setup
               # A0 & !A1 & !A2 & !~CS & SA0 & !SA1 & !SA2 & !~Setup
               # !A0 & A1 & A2 & !~CS & !SA0 & SA1 & SA2 & !~Setup
               # !A0 & A1 & !A2 & !~CS & !SA0 & SA1 & !SA2 & !~Setup
               # !A0 & !A1 & A2 & !~CS & !SA0 & !SA1 & SA2 & !~Setup
               # !A0 & !A1 & !A2 & !~CS & !SA0 & !SA1 & !SA2 & !~Setup);


RA2 := (RA2 & ~Setup_Sel # D2 & !~Setup_Sel);

RA1 := (RA1 & ~Setup_Sel # D1 & !~Setup_Sel);

RA0 := (RA0 & ~Setup_Sel # D0 & !~Setup_Sel);
```

ABEL(tm) 3.00b - Document Generator 03-Jan-89 03:38 PM
AIPS I/O Network Failure Insertion Select EPLD

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Chip diagram for Module net_fail_select

Device NET_FAIL_SELECT



end of module net_fail_select

```
"*****************************************************************************
"  FILENAME:    NET_FAIL_MODE.ABL
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Separated out test vectors
"   B   1/3/89           TCT     Changed led output to match as built
"*****************************************************************************

module net_fail_mode

flag    '-r3','-t2'

title   'AIPS I/O Network Failure Mode EPLD

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev: 10/31/88
'

"Declarations

        NET_FAIL_MODE   device  'E0320';

    " define ABEL .. commands:
        C, K, P, X, Z = .C., .K., .P., .X., .Z.;

    " define logic states
        HI, LO = 1, 0;

    " inputs

         DSN                        pin     1;

        !Run_Sel                    pin     2;
        !Setup_Sel                  pin     3;

        D3,D2,D1,D0                 pin     4,5,6,7;

        Fail_In_RX                  pin     8;
        Fail_Out_RX                 pin     9;

    " outputs

        Fail_Out_TX, Fail_In_TX             pin     15,13;
        Fail_Out_TX, Fail_In_TX             istype  'pos, com';

        Fail_Out_LED, Fail_In_LED           pin     14,12;
        Fail_Out_LED, Fail_In_LED           istype  'pos, com';

        Fail_Out_LED.OE, Fail_In_LED.OE istype  'eqn';

        In_M1, In_M0                pin     19,18;
        In_M1, In_M0                istype  'pos, reg, feed_pin';

        Out_M1, Out_M0              pin     17,16;
        Out_M1, Out_M0              istype  'pos, reg, feed_pin';

     " sets

        fail_out_mode = [Out_M1,Out_M0];
```

```
        fail_out_data = [D3..D2];

        fail_in_mode = [In_M1,In_M0];
        fail_in_data = [D1..D0];

    " states

        " data in to mode out
            NO_CHANGE = ^b11;
            OUT_HIGH  = ^b10;
            OUT_LOW   = ^b01;
            NORMAL    = ^b00;

        " modes
            UNUSED    = ^b11;
"           OUT_HIGH  = ^b10;
"           OUT_LOW   = ^b01;
"           NORMAL    = ^b00;

        " define levels for LED colors
            RED = 0;            " TCT 1/3/89
            GRN = 1;            " TCT 1/3/89
            OFF = .Z.;

    " macros

        SELECT macro {(Run_Sel # Setup_Sel)};

equations

        fail_out_mode := fail_out_data & SELECT & (fail_out_data != NO_CHANGE)
                      # fail_out_mode & (!SELECT # (fail_out_data == NO_CHANGE));

        fail_in_mode := fail_in_data & SELECT & (fail_in_data != NO_CHANGE)
                     # fail_in_mode & (!SELECT # (fail_in_data == NO_CHANGE));

        Fail_In_LED.OE = (fail_in_mode == NORMAL) #  (fail_in_mode == OUT_HIGH);

        Fail_Out_LED.OE = (fail_out_mode == NORMAL)  #  (fail_out_mode == OUT_HIGH);


truth_table ([fail_in_mode, Fail_Out_RX] -> Fail_In_TX)

        [NORMAL,   LO] -> LO;
        [NORMAL,   HI] -> HI;
        [OUT_HIGH,  X] -> HI;
        [OUT_LOW,   X] -> LO;
        [UNUSED,    X] -> LO;   " treat unused as OUT_LOW

truth_table ([fail_out_mode, Fail_In_RX] -> Fail_Out_TX)

        [NORMAL,   LO] -> LO;
        [NORMAL,   HI] -> HI;
        [OUT_HIGH,  X] -> HI;
        [OUT_LOW,   X] -> LO;
        [UNUSED,    X] -> LO;   " treat unused as OUT_LOW

truth_table (fail_out_mode -> Fail_Out_LED)

        NORMAL          -> GRN;
        OUT_HIGH        -> RED;
```

```
        OUT_LOW         -> X;
        UNUSED          -> X;

truth_table (fail_in_mode -> Fail_In_LED)

        NORMAL          -> GRN;
        OUT_HIGH        -> RED;
        OUT_LOW         -> X;
        UNUSED          -> X;

" Comment out the following instruction to compile production .JED files
"
"    @INCLUDE   'NET_FAIL_MODE.TST'

end net_fail_mode
```

```
"*****************************************************************************
"  FILENAME:    NET_FAIL_MODE.TST
"      DATE:    October 31, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   10/31/88         TCT     Separated out test vectors
"*****************************************************************************

test_vectors    'Test affect of Data on Mode and LED, and output'
    ([DSN, Run_Sel, Setup_Sel, fail_in_data, fail_out_data, Fail_Out_RX, Fail_In
_RX] ->
     [fail_in_mode, fail_out_mode, Fail_In_TX, Fail_Out_TX, Fail_In_LED, Fail_Ou
t_LED])

    " Test Channel 1

        [C, 1, 1, NORMAL,     NORMAL, LO, LO] -> [NORMAL,   NORMAL, LO, LO, GRN, GRN];
        [C, 1, 1, NO_CHANGE, NORMAL, LO, LO] -> [NORMAL,   NORMAL, LO, LO, GRN, GRN];
        [X, X, X, X,          X,      HI, LO] -> [NORMAL,   NORMAL, HI, LO, GRN, GRN];
        [X, X, X, X,          X,      LO, HI] -> [NORMAL,   NORMAL, LO, HI, GRN, GRN];
        [X, X, X, X,          X,      HI, HI] -> [NORMAL,   NORMAL, HI, HI, GRN, GRN];

        [C, 0, 0, OUT_HIGH,   NORMAL, LO, LO] -> [NORMAL,   NORMAL, LO, LO, GRN, GRN];
        [C, 0, 1, OUT_HIGH,   NORMAL, LO, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [C, 0, 1, NO_CHANGE, NORMAL, LO, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [X, X, X, X,          X,      HI, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [X, X, X, X,          X,      LO, HI] -> [OUT_HIGH, NORMAL, HI, HI, RED, GRN];
        [X, X, X, X,          X,      HI, HI] -> [OUT_HIGH, NORMAL, HI, HI, RED, GRN];

        [C, 0, 0, OUT_LOW,    NORMAL, LO, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [C, 1, 0, OUT_LOW,    NORMAL, LO, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
        [C, 1, 0, NO_CHANGE, NORMAL, LO, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
        [X, X, X, X,          X,      HI, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
        [X, X, X, X,          X,      LO, HI] -> [OUT_LOW,  NORMAL, LO, HI, OFF, GRN];
        [X, X, X, X,          X,      HI, HI] -> [OUT_LOW,  NORMAL, LO, HI, OFF, GRN];

        [C, 0, 0, NORMAL,     NORMAL, LO, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];


test_vectors    'Test affect of Data on Mode and LED, and output'
    ([DSN, Run_Sel, Setup_Sel, fail_out_data, fail_in_data, Fail_In_RX, Fail_Out
_RX] ->
     [fail_out_mode, fail_in_mode, Fail_Out_TX, Fail_In_TX, Fail_Out_LED, Fail_I
n_LED])

    " Test Channel 2

        [C, 1, 1, NORMAL,     NORMAL, LO, LO] -> [NORMAL,   NORMAL, LO, LO, GRN, GRN];
        [C, 1, 1, NO_CHANGE, NORMAL, LO, LO] -> [NORMAL,   NORMAL, LO, LO, GRN, GRN];
        [X, X, X, X,          X,      HI, LO] -> [NORMAL,   NORMAL, HI, LO, GRN, GRN];
        [X, X, X, X,          X,      LO, HI] -> [NORMAL,   NORMAL, LO, HI, GRN, GRN];
        [X, X, X, X,          X,      HI, HI] -> [NORMAL,   NORMAL, HI, HI, GRN, GRN];

        [C, 0, 0, OUT_HIGH,   NORMAL, LO, LO] -> [NORMAL,   NORMAL, LO, LO, GRN, GRN];
        [C, 1, 1, OUT_HIGH,   NORMAL, LO, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [C, 0, 1, NO_CHANGE, NORMAL, LO, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [X, X, X, X,          X,      HI, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
        [X, X, X, X,          X,      LO, HI] -> [OUT_HIGH, NORMAL, HI, HI, RED, GRN];
        [X, X, X, X,          X,      HI, HI] -> [OUT_HIGH, NORMAL, HI, HI, RED, GRN];
```

```
[C, 0, 0, OUT_LOW,    NORMAL, LO, LO] -> [OUT_HIGH, NORMAL, HI, LO, RED, GRN];
[C, 1, 1, OUT_LOW,    NORMAL, LO, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
[C, 1, 1, NO_CHANGE,  NORMAL, LO, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
[X, X, X, X,          X,      HI, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
[X, X, X, X,          X,      LO, HI] -> [OUT_LOW,  NORMAL, LO, HI, OFF, GRN];
[X, X, X, X,          X,      HI, HI] -> [OUT_LOW,  NORMAL, LO, HI, OFF, GRN];

[C, 0, 0, NORMAL,     NORMAL, LO, LO] -> [OUT_LOW,  NORMAL, LO, LO, OFF, GRN];
```

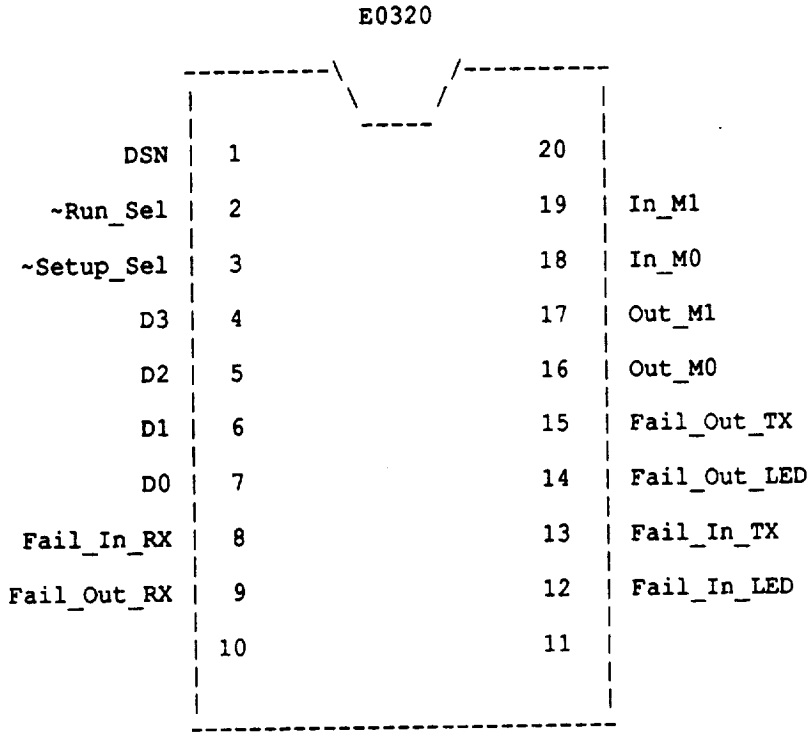
```
ABEL(tm) 3.00b  -  Document Generator              03-Jan-89 03:41 PM
AIPS I/O Network Failure Mode EPLD

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev: 10/31/88

Equations for Module net_fail_mode

Device NET_FAIL_MODE


- Reduced Equations:

    Out_M1 := (D3 & Out_M1
               # Out_M1 & ~Run_Sel & ~Setup_Sel
               # !D2 & D3 & !~Setup_Sel
               # !D2 & D3 & !~Run_Sel);


    Out_M0 := (D2 & Out_M0
               # Out_M0 & ~Run_Sel & ~Setup_Sel
               # D2 & !D3 & !~Setup_Sel
               # D2 & !D3 & !~Run_Sel);


    In_M1 := (D1 & In_M1
              # In_M1 & ~Run_Sel & ~Setup_Sel
              # !D0 & D1 & !~Setup_Sel
              # !D0 & D1 & !~Run_Sel);


    In_M0 := (D0 & In_M0
              # In_M0 & ~Run_Sel & ~Setup_Sel
              # D0 & !D1 & !~Setup_Sel
              # D0 & !D1 & !~Run_Sel);


    enable Fail_In_LED = (!In_M0);

    enable Fail_Out_LED = (!Out_M0);

    Fail_In_TX = (!In_M0 & In_M1 # Fail_Out_RX & !In_M0);

    Fail_Out_TX = (!Out_M0 & Out_M1 # Fail_In_RX & !Out_M0);

    Fail_Out_LED = (Out_M0 # !Out_M1);

    Fail_In_LED = (In_M0 # !In_M1);
```

ABEL(tm) 3.00b - Document Generator 03-Jan-89 03:41 PM
AIPS I/O Network Failure Mode EPLD

BOEING ADVANCED SYSTEMS
Designed by: Tom Torkelson Current rev: 10/31/88

Chip diagram for Module net_fail_mode

Device NET_FAIL_MODE



end of module net_fail_mode

```
"*************************************************************************
"  FILENAME:    FTC_CONTROL.ABL         FTC & misc control EPLD
"      DATE:    November 1, 1988
"        BY:    T.C. Torkelson
"*************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   11/2/88          TCT     Added input for VME FTC, modified VME_FTC_LED
"   B   1/3/89           TCT     Added CS and !SETUP to transparent latches
"*************************************************************************

module ftc_control

flag    '-r3','-t1'

title  'Wire Wrap Board FTC & Misc Control EPLD

BOEING ADVANCED SYSTEMS
Designed by:  Tom Torkelson     Current rev:  11/1/88
 '

"Declarations:

       FTC_CONTROL     device          'E0320';

    " define ABEL .. commands
       C,K,P,X = .C.,.K.,.P.,.X.;

    " inputs:

       CLOCK           pin     1;      " 16 Mhz clock

       EXT_EVENT       pin     2;      " external event input

       FTP_SYNC        pin     3;      " FTC sync input

       !SETUP          pin     4;      " !Setup / Run

       CS              pin     5;      " chip select in, high to select

       D1,D0           pin     6,7;    " data inputs from experiment bus

       !DS             pin     8;      " low passes data to output, high latches

       VME_FTC         pin     9;      " VME FTP FTC input

    " outputs:

       CLOCK_8_MHZ     pin     19;
       CLOCK_8_MHZ     istype  'pos, reg_D, feed_reg';

       X_EVENT         pin     18;             " X_EVENT is pos true
       X_EVENT         istype  'pos, com';

       EXT_EVENT_LED   pin     17;             " GRN no event, RED event
       EXT_EVENT_LED   istype  'neg, com';

       FTP_SYNC_LED    pin     16;             " RED waiting for sync, GRN sync
       FTP_SYNC_LED    istype  'neg, com';

       VME_SYNC_LED    pin     15;             " RED waiting for sync, GRN sync
       VME_SYNC_LED    istype  'neg, com';
```

```
        EXT_EVENT_POL   pin     14;
        EXT_EVENT_POL   istype  'pos, com, feed_pin';

        FTC_SEL         pin     13;
        FTC_SEL         istype  'pos, com, feed_pin';

        VME_FTC_LED     pin     12;
        VME_FTC_LED     istype  'pos, com, feed_pin';
        VME_FTC_LED.OE  istype  'eqn';

"       FTC_SEL output levels
            VME_SEL = 0;
            FTP_SEL = 1;

    " constant declarations for LED outputs
        OFF = .Z.;
        RED = 0;
        GRN = 1;

    " internal equates

        VME_SYNC = !SETUP;

        VRUN = !SETUP;
        VSTOP = SETUP;

    " macros
        " form latch which passes in to out ONLY when in VSTOP, CS, and !DS

        LATCH macro (out, in) {?out = ?out & (!CS # VRUN # !DS) # ?in & CS & SETUP & DS
;}

equations

        CLOCK_8_MHZ := !CLOCK_8_MHZ;

    " Transparent latches:
"       FTC_SEL       = D0 & DS # FTC_SEL & !DS;
"       EXT_EVENT_POL = D1 & DS # EXT_EVENT_POL & !DS;

        LATCH (FTC_SEL, D0)
        LATCH (EXT_EVENT_POL, D1)

        VME_FTC_LED.OE = (FTC_SEL == VME_SEL);  " enable LED on VME_SEL

truth_table     ([FTC_SEL, VME_FTC] -> VME_FTC_LED)
"
" When the external FTC reference is deselected, the LED is off.
"
" When the external FTC reference is selected, the LED is RED for input low,
" GRN for input high, and AMBER for input oscillation.

        [VME_SEL, 0] -> RED;
        [VME_SEL, 1] -> GRN;

        [FTP_SEL, 0] -> X;
        [FTP_SEL, 1] -> X;

truth_table ([EXT_EVENT_POL, EXT_EVENT] -> X_EVENT)
```


C-3

```
        [0, 0] -> 1;
        [0, 1] -> 0;

        [1, 0] -> 0;
        [1, 1] -> 1;

truth_table (X_EVENT -> EXT_EVENT_LED)

        0 -> GRN;
        1 -> RED;

truth_table (FTP_SYNC -> FTP_SYNC_LED)

        0 -> RED;
        1 -> GRN;

truth_table (VME_SYNC -> VME_SYNC_LED)

        0 -> RED;
        1 -> GRN;

" Comment out the following command to compile production .JED files
"
"    @INCLUDE   'FTC_CONTROL.TST'

end ftc_control
```

```
"*****************************************************************************
"  FILENAME:    FTC_CONTROL.TST          FTC & misc control EPLD test vectors
"      DATE:    November 1, 1988
"        BY:    T.C. Torkelson
"*****************************************************************************
"  REV    DATE          BY      DESCRIPTION
"   A   11/2/88         TCT     Added input for VME FTC, modified VME_FTC_LED
"   B   1/3/89          TCT     Added CS and !SETUP to transparent latches
"*****************************************************************************

test_vectors    'Test FTP_SYNC_LED output'
        (FTP_SYNC -> FTP_SYNC_LED)

                0 -> RED;
                1 -> GRN;

test_vectors    'Test VME_SYNC_LED output'
        (VME_SYNC -> VME_SYNC_LED)

                0 -> RED;
                1 -> GRN;

test_vectors    'Test FTC_SEL and VME_FTC_LED output'
        ([!DS, CS, !SETUP, D0, VME_FTC] ->
         [FTC_SEL, VME_FTC_LED])

    " test low inputs
        [1, 1, 0, VME_SEL, 0] -> [X, X];
        [1, 1, 0, VME_SEL, 1] -> [X, X];
        [0, 1, 0, VME_SEL, 1] -> [VME_SEL, GRN];
        [0, 1, 0, VME_SEL, 0] -> [VME_SEL, RED];
        [1, 1, 0, VME_SEL, 0] -> [VME_SEL, RED];
        [1, 1, 0, VME_SEL, 1] -> [VME_SEL, GRN];
    " test high inputs
        [1, 1, 0, FTP_SEL, 1] -> [VME_SEL, GRN];
        [1, 1, 0, FTP_SEL, 0] -> [VME_SEL, RED];
        [0, 1, 0, FTP_SEL, 0] -> [FTP_SEL, OFF];
        [0, 1, 0, FTP_SEL, 1] -> [FTP_SEL, OFF];
        [1, 1, 0, FTP_SEL, 1] -> [FTP_SEL, OFF];
        [1, 1, 0, FTP_SEL, 0] -> [FTP_SEL, OFF];
    " low inputs
        [1, 1, 0, VME_SEL, 0] -> [FTP_SEL, OFF];
        [1, 1, 0, VME_SEL, 1] -> [FTP_SEL, OFF];

test_vectors    'Test EXT_EVENT logic'
        ([!DS, CS, !SETUP, D1, EXT_EVENT] ->
         [EXT_EVENT_POL, X_EVENT, EXT_EVENT_LED])

    " test low inputs
        [1 ,1, 0, 0, 0] -> [X, X, X];
    " strobe in low polarity and test
        [0 ,1, 0, 0, 0] -> [0, 1, RED];
        [1, 1, 0, 0, 0] -> [0, 1, RED];
        [1 ,1, 0, 0, 1] -> [0, 0, GRN];
        [1 ,1, 0, 1, 1] -> [0, 0, GRN];
        [1, 1, 0, 1, 0] -> [0, 1, RED];
    " strobe in high polarity and test
        [0 ,1, 0, 1, 0] -> [1, 0, GRN];
        [1 ,1, 0, 1, 0] -> [1, 0, GRN];
        [1 ,1, 0, 1, 1] -> [1, 1, RED];
        [1, 1, 0, 0, 1] -> [1, 1, RED];
```

```
test_vectors    'Test DS, CS, SETUP logic'
        ([!DS, CS, !SETUP, D1] -> [EXT_EVENT_POL])

    " set to known state - test transparency
        [0, 1, 0, 0] -> 0;
        [0, 1, 0, 1] -> 1;
    " see if DS works as latch
        [1, 1, 0, 1] -> 1;      " latch
        [1, 1, 0, 0] -> 1;      " change input
        [0, 1, 0, 0] -> 0;      " enable
        [1, 1, 0, 0] -> 0;      " latch
        [1, 1, 0, 1] -> 0;      " change input
        [0, 1, 0, 1] -> 1;      " enable

    " see if CS works as latch
        [0, 0, 0, 1] -> 1;      " latch
        [0, 0, 0, 0] -> 1;      " change input
        [0, 1, 0, 0] -> 0;      " enable
        [0, 0, 0, 0] -> 0;      " latch
        [0, 0, 0, 1] -> 0;      " change input
        [0, 1, 0, 1] -> 1;      " enable

    " see if !SETUP works as latch
        [0, 1, 1, 1] -> 1;      " latch
        [0, 1, 1, 0] -> 1;      " change input
        [0, 1, 0, 0] -> 0;      " enable
        [0, 1, 1, 0] -> 0;      " latch
        [0, 1, 1, 1] -> 0;      " change input
        [0, 1, 0, 1] -> 1;      " enable
```