AT& T 3B2 COMPUTER HARDWARE

INTERFACE CARDS FOR OEM DESIGNERS

Course UC2302 - Student Guide

AT&T TECHNOLOGIES

JUNE, 1985

xxx-xxx, Issue 1

REFERENCES

1. P. M. Walsh, "3B 2 Circuit Description: Common I/O Hardware Issue 1 (LDI LA - LD)", December 28, 1983.

2. A. D. Niemi, "3B2 Feature Requirements: Common I/O Hardware 2 (CIO2)", March 13, 1984.

3. G. E. Laggis, et al, "3B2 Requirements - I/O Board Hardware and Firmware Architecture and Design," November 4,1983.

4. D. C. Bina, et al, "3B2 System Architecture and Requirements", 49343-0011, February 10, 1983.

5. P. M. Walsh, "3B 2 Component Requirements and Specifications: I/O Bus", March 7, 1983.

6. K. A. McWethy, J. M. Sullivan and L. E. Wallis, "3B2 Feature Requirements - Core System Hardware," March 7, 1983.

7. Intel Corporation, "iAPX 16 High Integration 16-bit Microprocessor", Data sheet, May 1982.

8. A. D. Niemi, "AT&T 3B2 Common I/O 2 Custom Polycell Customer Input Package", July 20, 1984.

9. P. M. Walsh, "3B 2 Component Requirements and Specifications: I/O Bus Re-release", December 30, 1983.

10. E. L. Hepler, D. M. Olien and P. M. Walsh, "3B2 Component Design: Review Guidelines", July 5, 1983.

11. M. Brown, M. Gasper, K. Kolwicz, "CCA Design Guide For System Designers", Preliminary - September 8, 1983.

12. R. R. Hylka and A. D. Niemi, "3B*2 Common I/O Hardware Timing Results", October 11, 1983.

ITEM

LDI ISSUE ____ SHEET 1/10

CODE OR DOCUMENT IDENTIFICATION

C102 - HW

16 MHz XTAL
LC06
MP1

C1 20pf
C2 20pF
C1TN1
CR2IN1

IC01 80186 SOCKET 268-5400-00-1102

IC05 LS08

IC03 7407

PFAIL0 OUT

RD 62 CRDO
WR 63 CWRO
ALE 61 CALE1

CLKOUT

AT&T BELL LABORATORIES
PROPRIETARY (RESTRICTED)
Solely for authorized persons having need to know pursuant to G

A. D. NIEMI 2/21/84
SIGNED DATE

WITNESSED AND UNDER
BELL TELEPH ATORI DATE DRATED

Ordering Information

CIO2 Chip

Contact your AT&T Technologies Sales Account Representative

Commercial code: 327 CA

Internal ordering: part code 327CA rev2

Price ~$30 for one
$20 for qty 100 - 499

Feature Card PC Board

Augat Systems Inc.
P. O. Box 474
Dayton, N. J. 08810-0474
(201) 329-3605
Attn. Sam Mark

Part# X8136 - LG1287 -2T

Price (not loaded) ~$30
(loaded) ~$250

Extender Board

Marlon Z. Kasprzyk
AT&T
Hickory Ridge Training Ctr.
1195 Summerhill Drive
Lisle, Ill. 60532
(312) 971-5195
Cornet 323-5195

AT& T 3B2 COMPUTER HARDWARE

INTERFACE CARDS FOR OEM DESIGNERS

Course UC2302 - Student Guide

AT&T TECHNOLOGIES

JUNE, 1985

xxx-xxx, Issue 1

Produced by

Corporate Education and Training

AT& T Technologies, Inc.

PROFILE: *UNIX* SYSTEM
AT& T 3B2 Computer Hardware Interface Cards for OEM Designers
COURSE UC2302— STUDENT GUIDE

COURSE HISTORY

None

COURSE DESCRIPTION

This course provides a description of the hardware/ software interface requirements for the AT&T 3B2 Computer including I/O, interrupt priorities, and direct memory access.

PREREQUISITES

AT&T 3B2 Computer Hardware Overview course (UC2301). Knowledge of "C" language and some assembler. Knowledge of hex and binary encoding and microcomputer hardware operations or a course on the WE32000 architecture.

TARGET POPULATION

Persons who require hardware specifications to design and (or) maintain system interface cards.

COURSE LENGTH

five days

PROFILE: *UNIX* SYSTEM
AT& T 3B2 Computer Hardware
Interface Cards for OEM Designers
COURSE UC2302— STUDENT GUIDE

COURSE OBJECTIVES

Upon completion of this course the student will be able to:

1. Design programmed and intelligent feature cards for the AT&T 3B2 Computer.
2. Pinpoint potential design problems before designing the board.
3. Use the I/O Bus incorporating the proper handshaking protocol.

CONTENTS: *UNIX* SYSTEM
AT& T 3B2 Computer Hardware
Interface Cards for OEM Designers
COURSE UC2302— STUDENT GUIDE

UNIT 1. CPU - Input/ Output Interface

UNIT 2. Programmable (dumb) Feature Cards

UNIT 3. Design Tools

CONTENTS: *UNIX* SYSTEM
AT& T 3B2 Computer Hardware
Interface Cards for OEM Designers
COURSE UC2302— STUDENT GUIDE

UNIT 4. Interrupts

UNIT 5. Intelligent (smart) Feature Cards

UNIT 6. Interrupts - Intelligent Card

CONTENTS: *UNIX* SYSTEM
AT& T 3B2 Computer Hardware
Interface Cards for OEM Designers
COURSE UC2302— STUDENT GUIDE

UNIT 7. Feature Card Construction

UNIT 8. Appendix

UNIT 1

CPU - INPUT/OUTPUT INTERFACE

CPU - INPUT/OUTPUT INTERFACE

Upon completion of this unit the student will be able to:

1. Describe major components and features of the AT&T 3B2 computer.
2. Identify and differentiate between programmed and intelligent feature cards for the 3B2 computer.
3. Describe the 3B2 system board, dual port dynamic random access memory, and addressing protocols.
4. Interpret and implement pre-design considerations before building a new board.
5. Define I/O Bus signals and examine their application using a dual trace oscilloscope.
6. Manipulate 3B2 feature card timing signals.

UNIT 1

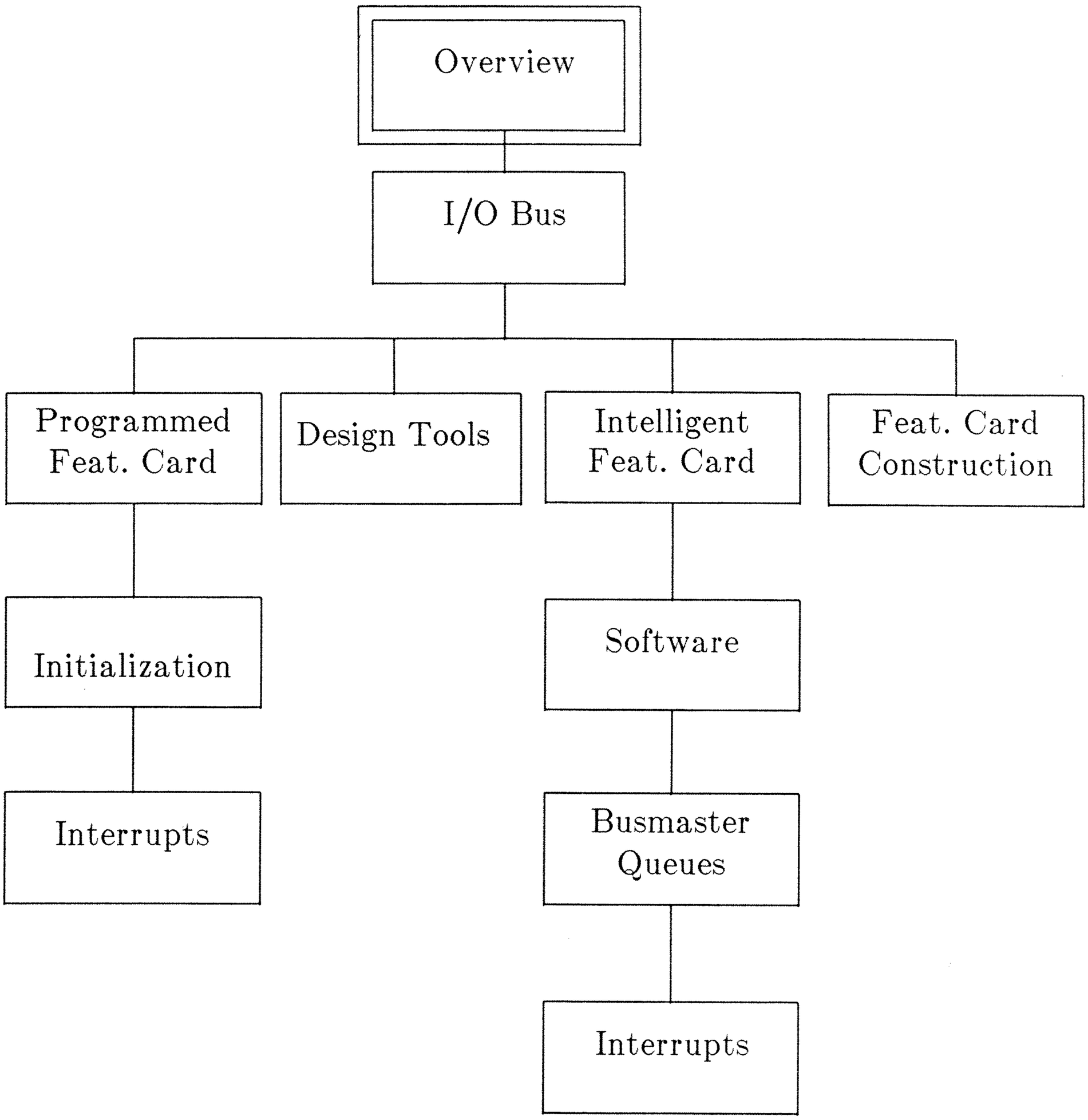
CPU - INPUT/OUTPUT INTERFACE

Lesson 1

Course Overview

This diagram will be used to show the relationships between the various parts of the course and which topics will be covered next. The first two lessons will be a general overview of the AT&T 3B2 Computer.

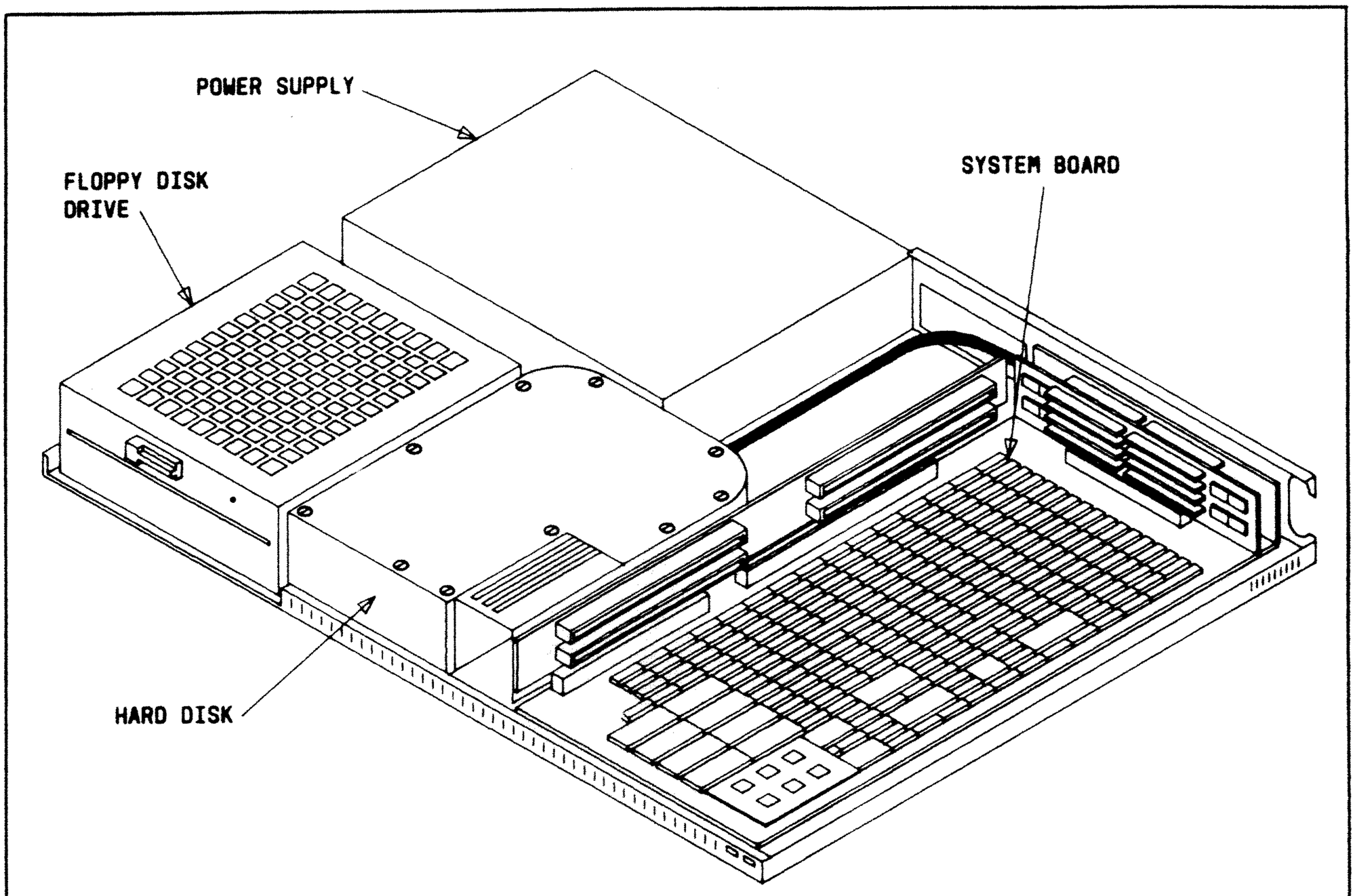
2302 Outline



MAJOR COMPONENTS - 3B2/300/310 COMPUTERS

- System Board
- Hard Disk Drive
- Floppy Disk Drive
- Power supply

MAJOR COMPONENTS
3B2/300/310 COMPUTERS



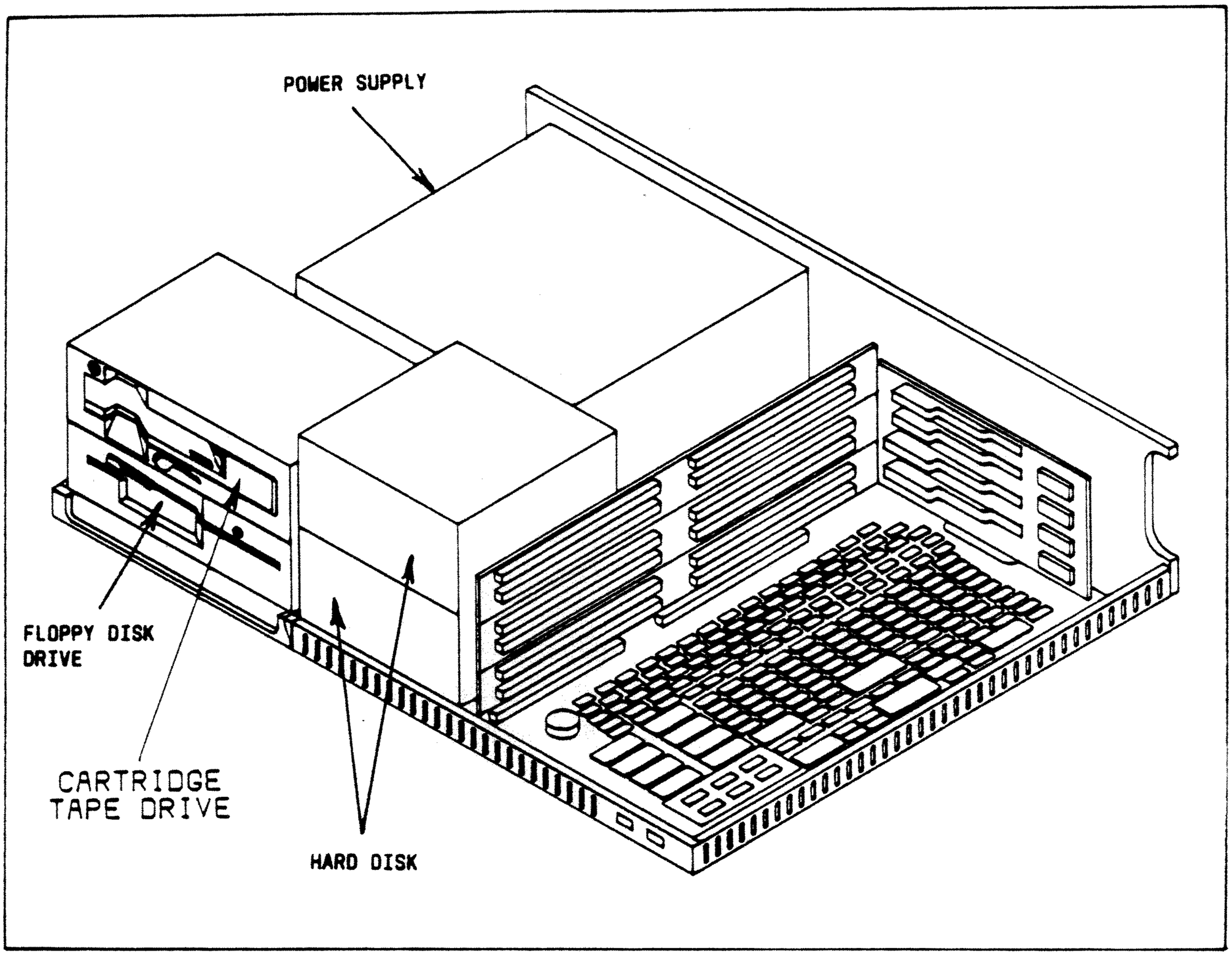
MAJOR COMPONENTS - 3B2/400 COMPUTER

- System Board
- Hard Disk Drive(s)
- Floppy Disk Drive
- Cartridge Tape Drive
- Power supply

64 * 32 bit cache

8 byte inst. Queue

MAJOR COMPONENTS
3B2/400 COMPUTER



FEATURES of the 3B2 COMPUTER

The AT&T 3B2 Computer is capable of supporting a wide range of configurations and usage options and is compatible with an office environment. Up to four optional feature cards can be installed in the 3B2/300/310 to enhance its operation, and if four "Ports" cards are installed, up to 18 users can be connected. The 3B2/400 Computer has room for 10 additional feature cards, and if they were all "Ports" cards, 46 users could be connected.

The unbundling of the UNIX software through the use of UNIX System Utility Packages allows the used to tailor the system to his own needs. This gives more flexibility and control over costs and hard disk usage.

The 3B2 Computer comes equipped with a number of diagnostic programs. Some of these are automatically run when power is turned on to insure that all of the hardware is working correctly before you start to operate the computer. If the hardware is "sane" the automatic configuration system then checks to determine what peripheral equipment is installed. Only those hardware drivers that are needed for the equipped devices are then added to the UNIX kernel. When this automatic self configuration is finished, the UNIX system is ready to run.

Passwords, file access permissions, and data encryption are all available to insure the security of the data and software stored in the computer. The user can control which of these security features are used to safeguard the information.

The Simplified System Administration feature is particularly helpful to first-time computer users. It allows inexperienced users to handle administrative functions that have traditionally been performed only by "super-users". One of the Simple Administration procedures walks the user step-by-step through the initial setup of the computer. Other procedures lead the user through such tasks as adding new users to the computer, formatting floppy disks, making backup copies of data, and installing software packages.

FEATURES of the 3B2 COMPUTER

- Designed for Multi-Tasking
- Uses UNIX Operating System
- Optional UNIX System Utility Packages
- Automatic Diagnostics on Power-Up
- Automatic Execution of Operating System on Power-Up
- Security System Provided
- Simplified System Administration
- Battery ~~Operated~~ Time-Of-Day Clock

Backup

HARD DISK DRIVE

The 3B2 computer can be ordered with either a 10, 30, or 72 megabyte disk drive. It is used to store the operating system, diagnostic programs, etc. When it is being shipped, the read-write heads may be locked. Be sure to unlock the heads before applying power to the computer.

Specifications:

Speed	3600 RPM
Average Latency	8.33 milliseconds
Average Access Time	
(10 Megabyte Disk)	90 Milliseconds
(30 Megabyte Disk)	45 milliseconds
(72 Megabyte Disk)	35 milliseconds
Data Transfer Rate	5,000,000 bits per second

HARD DISK DRIVE

- 5 1/4 inch Winchester
- Either 10, 30, or 72 megabytes
- Non-removable medium

FLOPPY DISK DRIVE

The floppy disks that are used must be double-density, double-sided, soft sectored, 96 tracks per inch, and free of defects. They should not be inserted or removed when the light on the front of the disk drive is on.

In order to store information on the disk the write-enable notch on the left edge of the disk must be **uncovered**. When this notch is **covered** the disk is said to be "write protected."

Specifications:

Tracks per Drive	160
Speed	300 RPM
Average Rotational Latency	100 milliseconds
Average Track Access Time	90 milliseconds
Data Transfer Rate	250,000 bits per seconds

FLOPPY DISK DRIVE

- 5 1/4 inch
- 720 kilobytes
- Removable medium

CARTRIDGE TAPE DRIVE

One of the uses of the Cartridge Tape Drive is for backing up the files on the hard disk. Data is recorded on the tape in six streams or tracks, each holding about four megabytes. The tape moves at 78 inches per second while streaming, and to fill one cartridge tape with data from the hard disk takes about 17 minutes.

A Cartridge Tape Drive can be used in a similar manner to a floppy disk for holding a mounted file system. This would be useful for applications such as data base log files or temporary work files.

The Cartridge Tape Drive can also be used to improve the access efficiency of the hard disk. This is accomplished by first copying the file system from the hard disk to the tape. The hard disk is then restructured and the file system is copied from the tape back onto the hard disk.

The Cartridge Tape Drive interfaces with the 3B2 via a Cartridge Tape Controller (CTC) card that must be installed in one of the I/O slots in the 3B2.

CARTRIDGE TAPE DRIVE

- Used to Backup Hard Disk
- 23 Megabyte Capacity
- 1 MB/min Backup Speed
- Removable 1/4 inch Tape Cartridge
- 6 tracks - Streaming Mode

TYPES OF FEATURE CARDS

Feature cards can communicate with the system board or DPDRAM, but not with another feature card.

Option straps are minimized through the use of unique feature card select signals that the system board generates.

Feature cards must be installed in the I/O expansion board connectors without any intervening empty connectors. This will allow the daisy-chained bus acknowledge and interrupt acknowledge signals to reach all cards.

The single slot feature cards are 6.45 by 7.40 inches, and the double slot cards are 13.40 by 7.40 inches.

When the system board is I/O Bus master it can address either main memory or a feature card. If a feature card is addressed, mutually exclusive select lines are used to identify the card that is to be the slave.

In order to become bus master, a feature card must send a request to the system board. When the system board acknowledges the request the feature card will become bus master. While it is bus master, a feature card can read or write the main memory (DPDRAM), but it can not access the CPU or another feature card.

A feature card slot can also be used to hold a circuit board that is not electrically connected to the I/O Bus. This card could draw its power from the I/O connector, but the input and output signals would come via cables from other feature cards or from external devices.

TYPES OF FEATURE CARDS

- 8 bit or 16 bit data words
- One or two slots wide
- Programmed or Intelligent
- Can be bus master or slave
- Can use up to 10 watts per slot

PROGRAMMED (dumb) FEATURE CARDS

A programmed feature card is one that does **not** use request and completion queues to communicate with the system board. Typically it will not contain a microprocessor.

A deadlock occurs when the system board and a feature card both attempt to gain control of the I/O bus. The designer of the feature card must include circuits to prevent this from happening.

An example of a programmed feature card would be a serial asynchronous communication feature card consisting of eight dual UARTs each having 16 byte-wide internal registers. This card would have 128 registers addressable from the system board in addition to the four standard registers.

Since programmed feature cards just use reads and writes to communicate with the system board, no special firmware protocol is needed for them.

PROGRAMMED (dumb) FEATURE CARDS

- Does **not** use queues
- Usually slave devices, but may be bus master after being programmed
- If it is a bus master, the feature card must prevent any I/O bus deadlocks
- Can read or write DPDRAM
- Contains up to 4 standard registers that can be read by the system board (I.D., Control, Status, and Interrupt)
- May use a DMA controller as an aid in transferring data to or from DPDRAM

INTELLIGENT (smart) FEATURE CARDS

Two examples of intelligent feature card are the Expansion Ports board (CM 195B) and the Network Interface board (CM 195A). Both of these cards use the Common I/O circuit (CIO) to interface with the system board. This circuit contains a 16-bit 80186 CPU, EPROM, RAM, and a logic sequencer (sometimes called a state machine), and occupies approximately 2/3 of the card. The remaining space is used for the application circuit (in these two examples the Ports or Network Interface circuits).

The queues are blocks of main memory (DPDRAM) that can be written by the system board and read by the feature card or vice versa. The system board fills the request queue with all of the job information that is needed by the feature card and then sends a single interrupt to the feature card. The feature card fills the completion queue while doing the job and then it sends an interrupt to the system board to indicate that the completion queue has been filled. This use of queues minimizes the traffic on the I/O Bus, and the time that would be lost if the feature cards communicated directly with the system board.

INTELLIGENT (smart) FEATURE CARDS

- Contain a CPU and firmware
- Restricted interface with the system board
  - -One 8/16 bit register that can be read by the system board
  - -Can read and write DPDRAM via programmed access or DMA
  - -Communicate with the system board by way of two one-way queues in DPDRAM (Request queue and Completion queue)
  - -Interrupts are sent from one CPU to the other CPU to signal a non-empty queue

UNIT 1

CPU - INPUT/OUTPUT INTERFACE

Lesson 2

3B2 Architecture

SYSTEM BOARD

Four sockets are provided for Read-Only Memory (ROM or EPROM) chips. They can hold either 32K bytes or 64K bytes, depending on whether the DEMON Debugging Monitor firmware is included. This memory is also used for the self-test sanity checks, auto configuration, and system initialization.

The nonvolatile memory is used to store system configuration parameters such as terminal baud rates and the firmware password. It also stores error information, in the event of a system failure, for later analysis. This CMOS static RAM holds 1024 4-bit nibbles.

Three timers are on the board. The time-of-day (TOD) timer maintains a time-of-day clock which provides the time in tenths of seconds, and is accurate to within 1.3 minutes per month. The periodic (or interval) timer is used to periodically interrupt the CPU. The bus timer is used to produce an interrupt if some nonresponding address is accessed by either the system board or an I/O Bus device.

The 3B2/300 operates with a clock speed of 7.2 MHz, while the model 310 and 400 have a 10 MHz clock.

The disk controllers provide data and access control for the disk drives. Data transfers are accomplished under DMA control, while controller setup occurs under programmed control.

A dual UART chip is used to provide the two RS-232-C ports for the console and contty connectors on the back of the 3B2.

The main memory is dual-ported; that is it can be accessed from two different directions. The CPU uses one port, while the other port is used by both the I/O Bus devices and the system board resident Direct Memory Access Controller (DMAC).

The 3B2/310 and /400 have room to add the WE32106 Math Acceleration Unit (MAU) co-processor that greatly improves floating point performance.

The system board (CM 190A, 190 B, or 190 C) dissipates 26 watts.

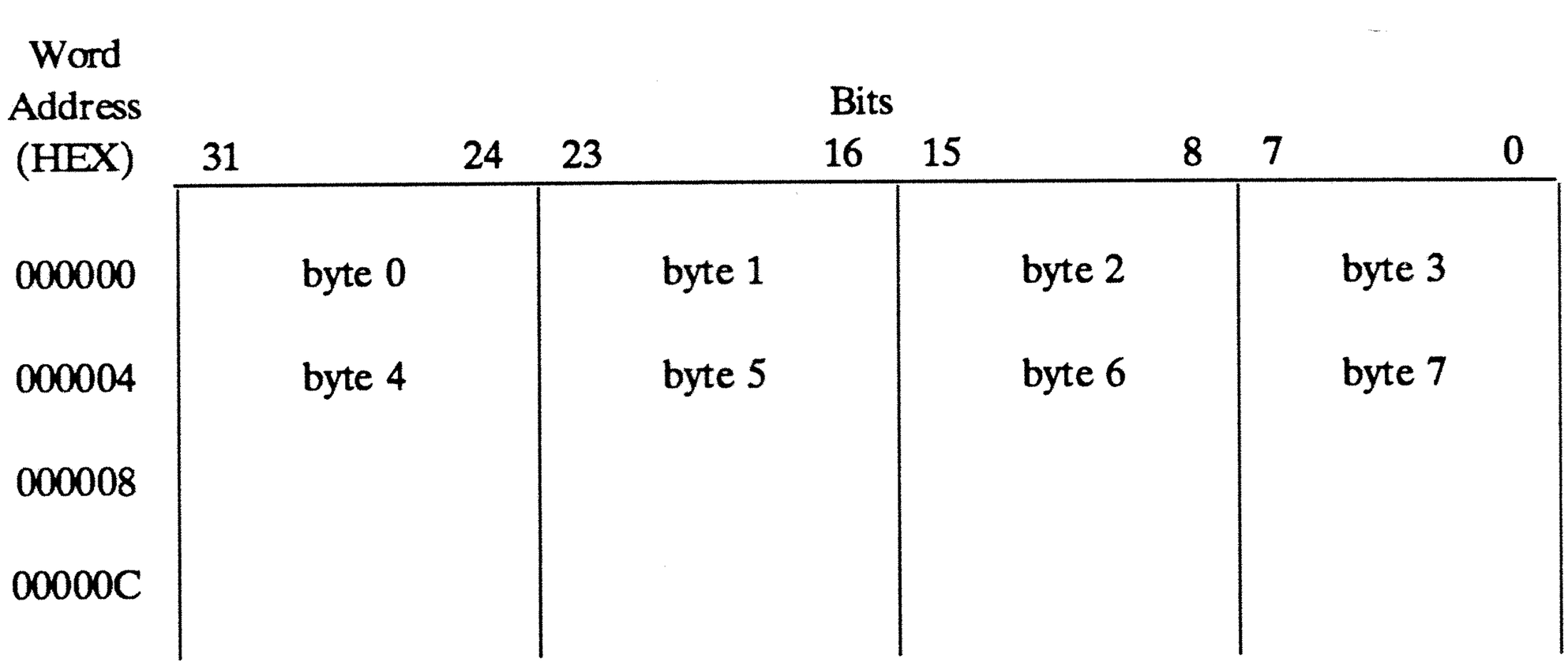
SYSTEM BOARD

- WE32100 or WE32002 Processor Module with a Memory Management Unit (MMU)
- Programmable Read-Only Memory
- Nonvolatile memory
- Timers TOD, interval, bus abort
- Interrupt structure
- 7.2 or 10 MHz Clock
- Controllers for the Floppy Disk and Hard Disk
- Two integral RS-232-C ports
- One or two vertically mounted memory boards Dual-Ported Dynamic RAM (DPDRAM)
- A vertically mounted I/O Expansion Features Board which supports up to four horizontally mounted Feature Cards. (12 Feature Cards for the 3B2/400)
- Room for a MAU (310/400)

MEMORY ADDRESSES

Memory words contain 32 bits plus 4 parity bits. (One for each byte.) The parity bits are calculated for each memory write operation, and automatically checked on each memory read. Even parity is used. The bits in each word are numbered from right to left. (LSB = 0, MSB = 31) An IBM-like byte ordering called high-byte-first is used. When the CPU sends an address to memory, an entire 32 bit word is read and placed on the data bus.

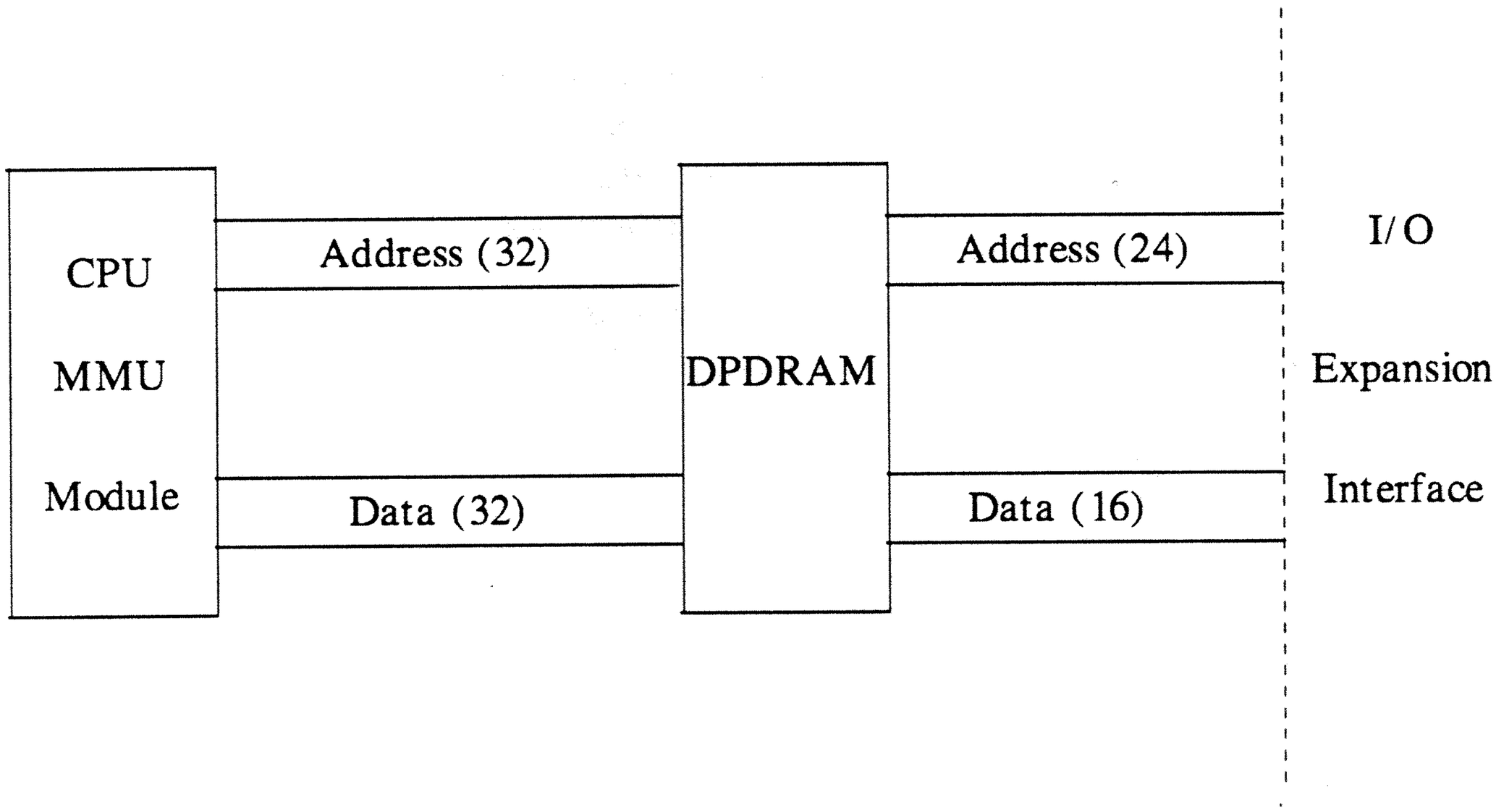
MEMORY ADDRESSES



DUAL PORT DYNAMIC RANDOM ACCESS MEMORY

The bus between the CPU and the DPDRAM is called the Microbus, while the bus to the feature cards is called the I/O bus.

DUAL PORT DYNAMIC RANDOM ACCESS MEMORY (DPDRAM)



SYSTEM BOARD BLOCK DIAGRAM

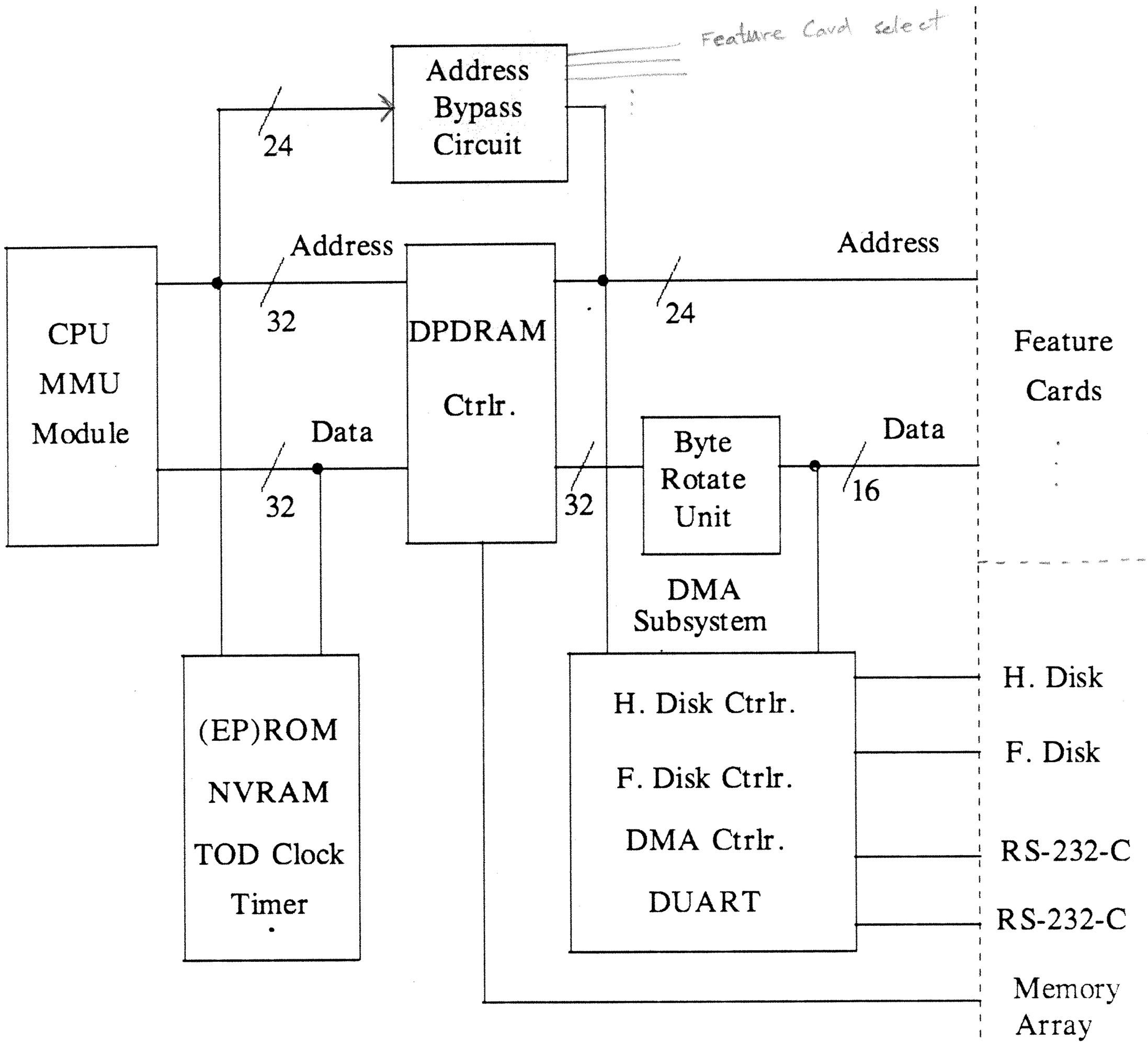
The CPU used is the WE32002 or WE32100 microprocessor module which, like larger machines, uses a full 32 bits for its registers and buses. The Memory Management Unit accepts virtual addresses from the CPU and translates them into physical addresses. This allows each process to have a full four gigabyte virtual address space.

The Address Bypass Circuit is used to allow the CPU to reach across the DPDRAM controller to communicate directly with the feature cards. The Byte Rotate Unit provides data alignment and packing for interfacing 8 and 16 bit feature cards to the CPU's 32 bit data bus.

The Address Bypass Circuit and Byte Rotate Unit also allow the CPU to access a group of circuits on the system board that are known as the DMA Sub-system. The chip selects for these circuits are generated by the same address decoder that produces the feature card select signals.

A unique feature card select signal is generated by the system board for each I/O slot, thus the feature cards do not have to decode the address bus information.

SYSTEM BOARD BLOCK DIAGRAM



CPU PHYSICAL ADDRESS SPECTRUM

The addresses from 200000 to 1ffffff (hex) are used by the system board to access the feature cards. (In the 3B2/300 Computer only the addresses from 200000 to 9fffff are needed for the 4 feature cards.) When the CPU asserts one of these addresses, one of the fifteen peripheral chip selects PCS[15-01]0 becomes active, identifying the card that is selected. The lower 24 bits of the address are sent to the feature cards via the I/O bus.

CPU PHYSICAL ADDRESS SPECTRUM

Address	Description	Width	Size
0	(EP)ROM	32	32/64KB
40000	MMU	-	-
41000	TOD Clock	8	16B
42000	PIT (8253)	8	4B
42010	CSR-6 Clear	1	1B
43000	NVRAM	4	1KN
44000	CSR	16	2B
45000	DMA Page Reg 1 (Hard Disk)	8	1B
46000	DMA Page Reg 2 (UART A)	8	1B
47000	DMA Page Reg 3 (UART B)	8	1B
48000	Int. DMA Control (9517)	8	16B
49000	UARTs (2681)	8	16B
49010	Clr UDMA Int.	8	1B
4A000	Int. Disk (7261)	8	2B
4B000	(Reserved)		
4C000	DPDRAM Size Reg.	8	1B
4D000	Floppy Disk (2797)	8	4B
4E000	DMA Page Reg 4 (Floppy)	8	1B
4F000	(Reserved)		
200000	I/O Board 1	8/16	2MB max
400000	I/O Board 2	8/16	2MB max
600000	I/O Board 3	8/16	2MB max
800000	I/O Board 4	8/16	2MB max
A00000	I/O Board 5	8/16	2MB max
C00000	I/O Board 6	8/16	2MB max
E00000	I/O Board 7	8/16	2MB max
1000000	I/O Board 8	8/16	2MB max
1200000	I/O Board 9	8/16	2MB max
1400000	I/O Board 10	8/16	2MB max
1600000	I/O Board 11	8/16	2MB max
1800000	I/O Board 12	8/16	2MB max
1A00000	I/O Select 13	8/16	2MB max
1C00000	I/O Select 14	8/16	2MB max
1E00000	I/O Select 15	8/16	2MB max
2000000	Main Memory (DPDRAM)	32	4MB max

FEATURE CARD OFFSET ADDRESSES

PROGRAMMED FEATURE CARD

The four registers shown are the standard ones that are used by programmed feature cards. These cards can also use other registers to communicate with the CPU.

The offset address is added to the base address for the selected feature card slot (see page 1.2.10) to get the actual address that is sent out on the I/O Bus.

eg: To access the status register of the card in slot 3 the address used would be 600005. (A base address of 600000 for slot 3, and an offset of 5 for the status register.)

INTELLIGENT FEATURE CARD

INT0 performs two functions. On power-up or after a reset of the feature card it is used to read the feature card identification code. After that its function is to signal the feature card that there is an express job request entry in the request queue in the DPDRAM. INT0 is therefore known as an 'express' interrupt.

INT1 is an 'attention' interrupt that is used by the system board to indicate to the feature card that there is a new entry in the job request queue. INT1 is known as a 'normal' interrupt.

By addressing the Status register of an intelligent feature card, that one card can be reset without having any effect on the other feature cards.

FEATURE CARD OFFSET ADDRESSES

PROGRAMMED FEATURE CARD

OFFSET ADDRESS	REGISTER NAME	SIZE
1	Identification	8/16 bits
3	Control	8/16 bits
5	Status	8/16 bits
7	Interrupt Vector	8 bits

INTELLIGENT FEATURE CARD

OFFSET ADDRESS	REGISTER NAME	ACTION TAKEN
1*	ID/Vector	INT0
3	Control	INT1
5	Status	Reset

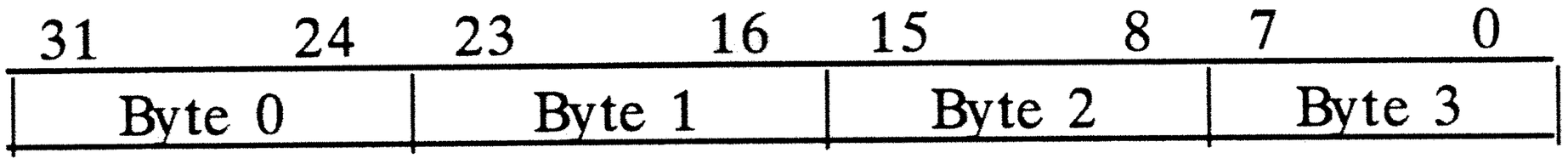
* For a 16-bit Feature Card the offset address is zero for the ID register.

BYTE FORMATS (DATA BUS)

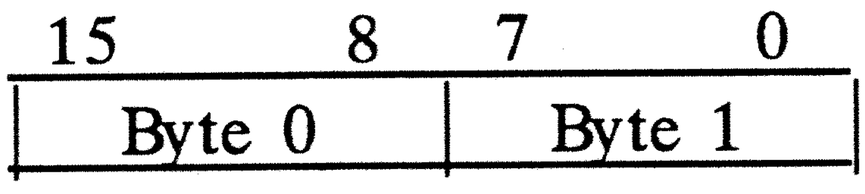
The Byte Rotate Unit (BRU) on the system board provides the data alignment and packing for interfacing 8 and 16-bit feature cards with the 32-bit main memory. Regardless of which device initiates the data transfer, it is always the feature card that indicates to the BRU whether it is capable of transferring 8 or 16 bits at a time.

BYTE FORMATS (DATA BUS)

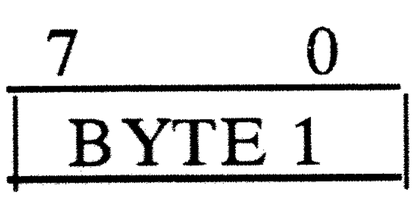
DATA BUS to MEMORY



DATA BUS to 16-BIT FEATURE CARD



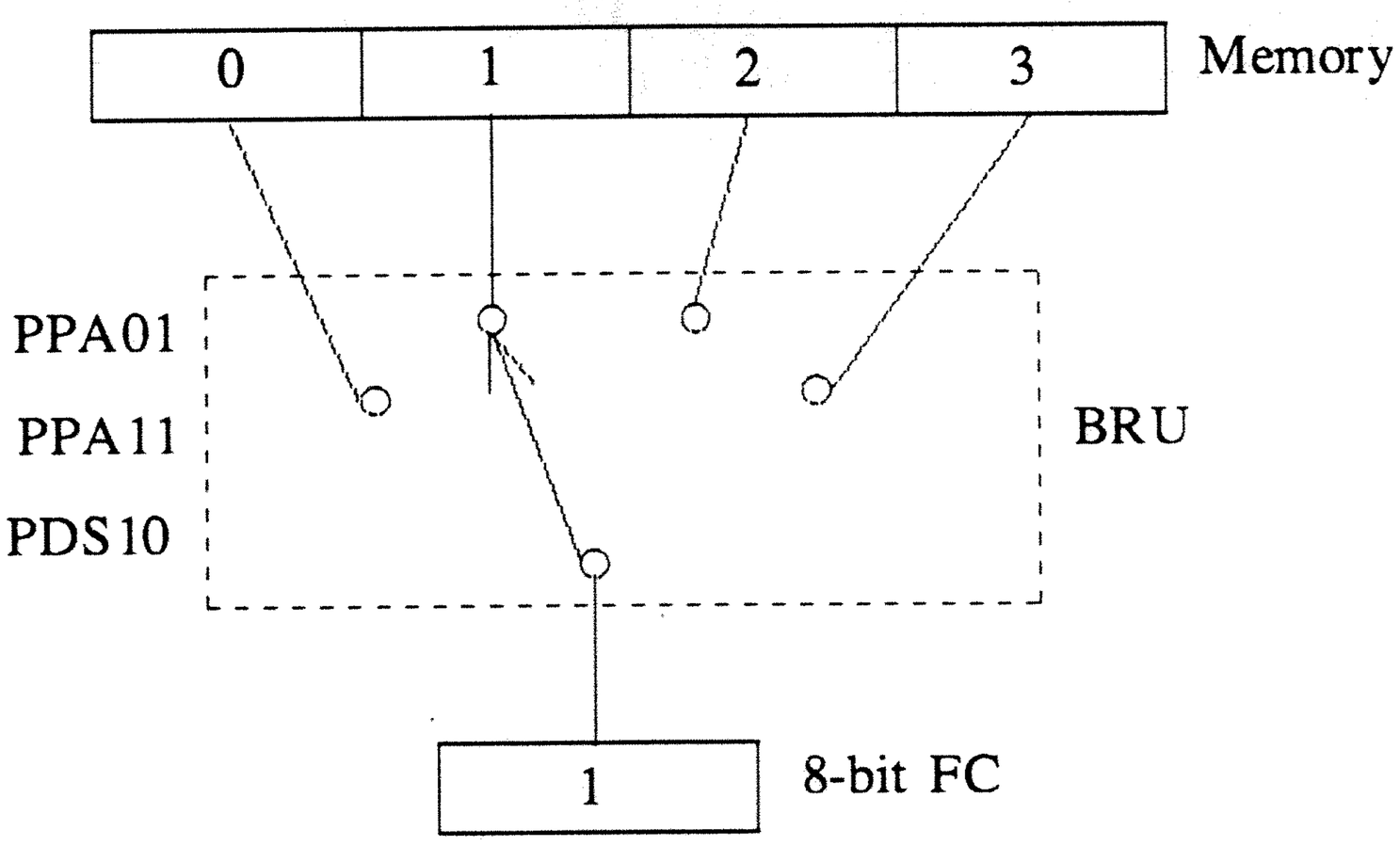
DATA BUS to 8-BIT FEATURE CARD

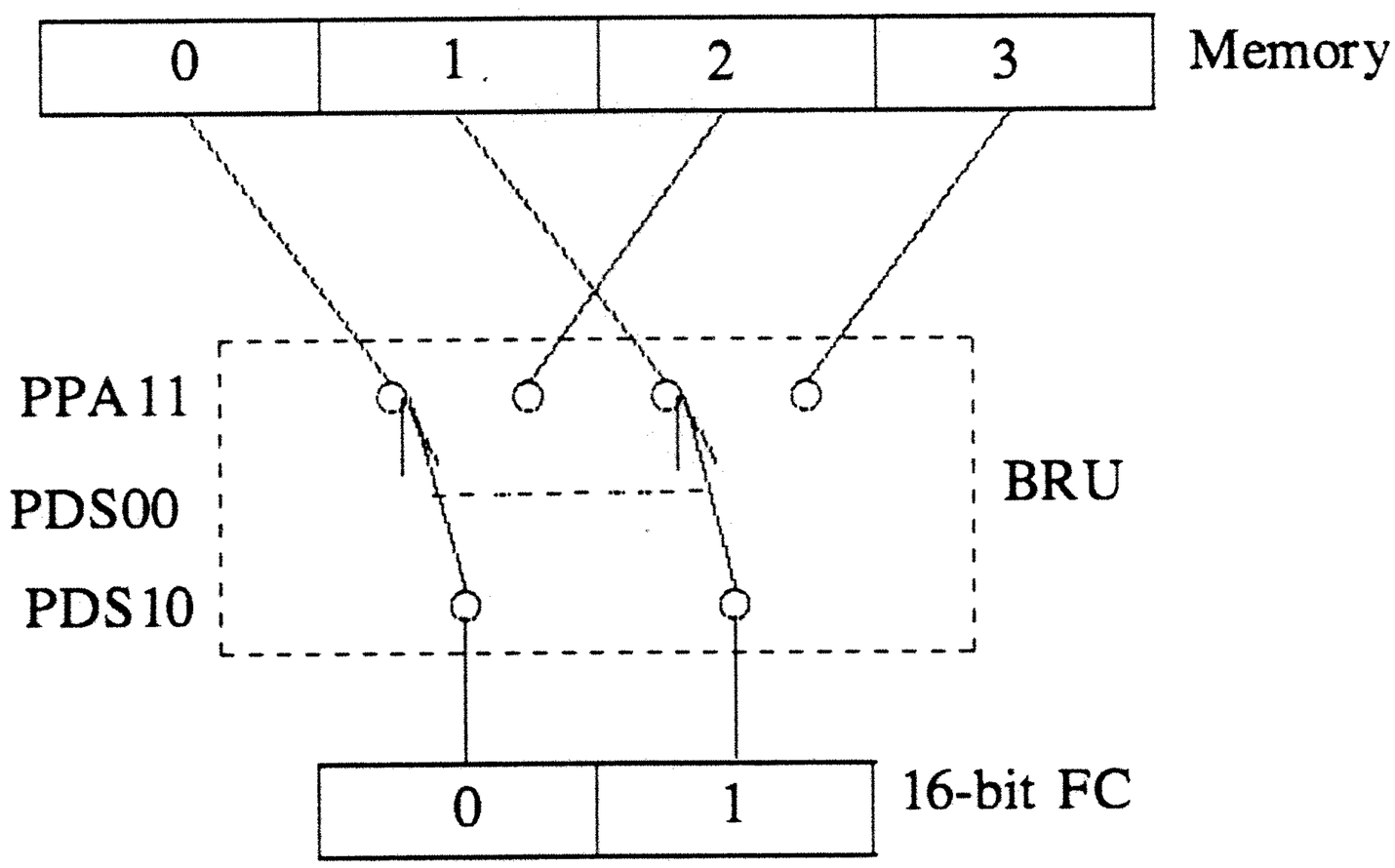


BRU OPERATION

The operation of the BRU (Byte Rotate Unit) in the system board is somewhat like that of a rotary switch or a single-pole double-throw (SPDT) switch. It connects the 8 or 16-bit data bus from the feature card to the appropriate byte or pair of bytes in the DPDRAM (Main Memory). The I/O Bus lines that control the position of the "switch" are shown in the two drawings.

BRU OPERATION





I/O BUS BYTE SELECTION

Five I/O bus signals, PSIZE160, PPA[1]1, PPA[0]1, PDS[0]0, and PDS[1]0 are used to select which bytes of the 32-bit main memory word are to be accessed.

I/O BUS BYTE SELECTION

	Address bits		Data Strobes		Valid Bytes	
PSIZE16	PPA[1]1	PPA[0]1	PDS[0]0	PDS[1]0	Memory	I/O
16-bit						
0	0	x	0	0	0,1	0,1
0	0	x	0	1	0	0
0	0	x	1	0	1	1
0	0	x	1	1	illegal	-
0	1	x	0	0	2,3	0,1
0	1	x	0	1	2	0
0	1	x	1	0	3	1
0	1	x	1	1	illegal	-
8-bit						
1	0	0	x	0	0	1
1	0	1	x	0	1	1
1	1	0	x	0	2	1
1	1	1	x	0	3	1
1	x	x	x	1	illegal	-

x = don't care

PRE-DESIGN CONSIDERATIONS

Before starting the design of a new feature card there are several options that should be considered. The choices that are made here will have a major effect on the design project and the effectiveness of the product produced.

PRE-DESIGN CONSIDERATIONS

1. Hardware Design Options

- Programmable or Intelligent Feature Card
- Type of DMA (Direct transfer vs Fetch and Deposit)
- Multiple Access Use of I/O Bus
- Bus Abort Feature
- Which of the Three Interrupt Signals to use
- Use of the CIO Circuit
- Number of Devices and Subdevices

2. Firmware Design Options

- Partitioning of Feature Card Memory Between ROM-Resident and Downloaded Ram-Resident code
- Polled or Interrupt-Driven Service Routines

UNIT 1

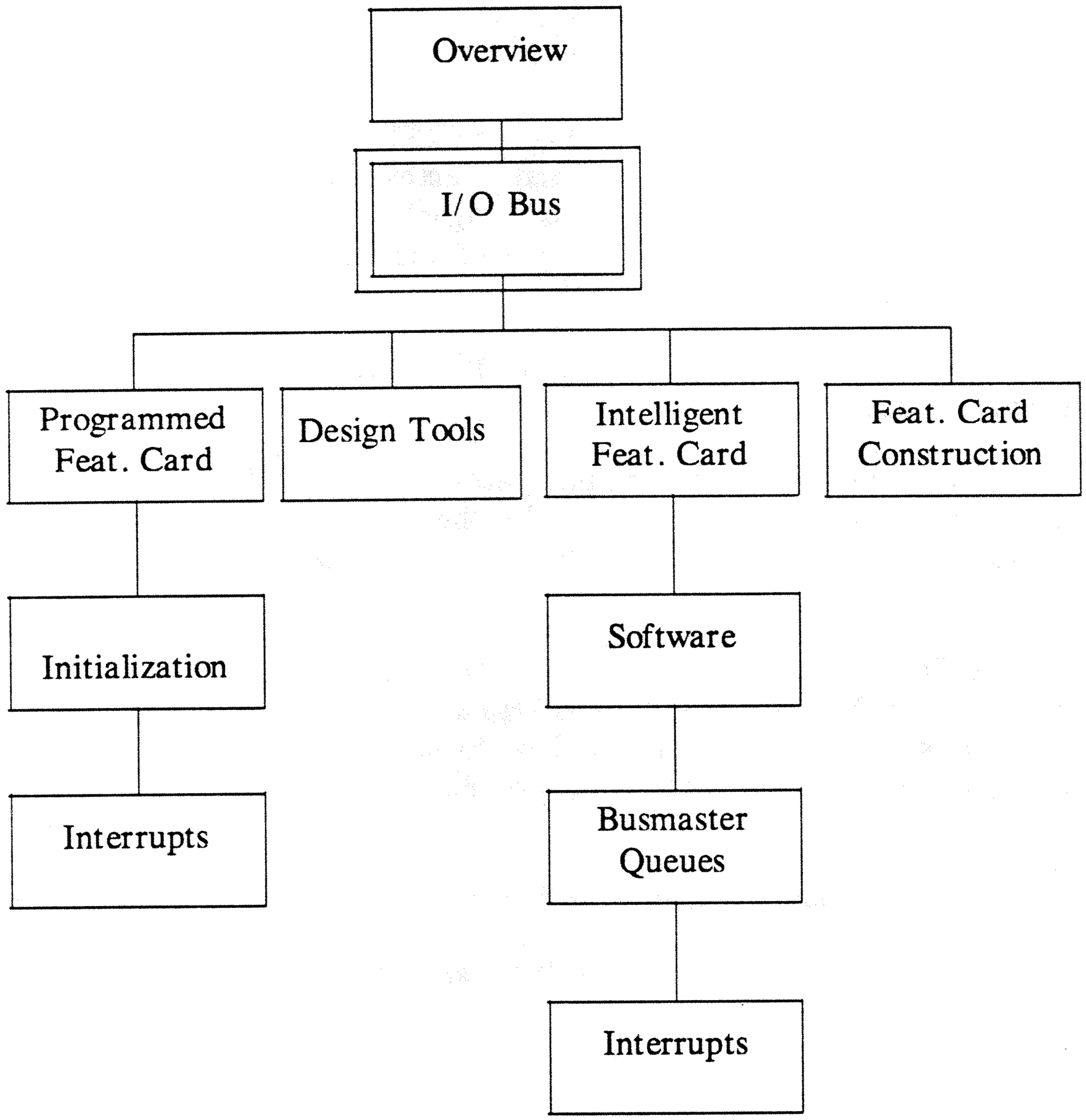
CPU - INPUT/OUTPUT INTERFACE

Lesson 3

Input/ Output Bus Signals

The next two lessons and the lab exercise that follows will introduce you to the Input/Output Bus signals and timing.

2302 Outline



I/O BUS SIGNALS

The I/O Bus Connector pin numbers for all of these lines are listed in Appendix A at the end of this book.

All I/O bus signals except for RQRST0 & SYSRST0 are prefixed with a 'P'. A suffix of 0 (1) indicates that the signal is active LOW (HIGH). If the first and last characters of an I/O bus signal name are removed, the remaining characters describe the function of the signal. eg: PBRQ0 = BRQ the abbreviation for Bus Request.

Totem-pole drivers are low impedance IC circuits that actively pull the output voltage high or low.

Tri-state drivers have three output states; high (logic 1), low (logic 0), and high impedance (floating). In the high impedance state the output can be pulled high or low by other drivers connected to the same line.

Open-collector drivers require an external pull-up load resistor connected to the line. If several drivers are connected to the same line, any one of them can pull the line low (logic 0). The output impedance of these circuits is low in the logic 0 condition.

NOTE:

The PLOCK0 line is reserved for future use. At the present time, the feature cards must drive this line high with the same timing specifications as PPA[23-00]1, PR1W0, and PSIZE160.

I/O BUS SIGNALS - (86 Lines)

- Control - (11)

  PBUSY0 - Feature card busy - ts
  from feature card to system board & all other feature cards

  PCS[15-01]0 - (12) Feature card select - tp
  one to each feature card connector from system board

  PDS[1-0]0 - (2) - Feature card data strobes - ts
  from master to slave - bi-dir
  select valid bytes on data bus

  PDTACK0 - Feature card data transfer acknowledge - oc
  from slave to master - bi-dir

  PLOCK0 - Feature card memory lock - ts
  from feature card to system board

tp = totem-pole
ts = tri-state
oc = open-collector

I/O BUS SIGNALS (CONT.)

The RQRST0 signal is not being used by any of the feature card at this time.

I/O BUS SIGNALS (CONT.)

- Control - (cont.)

  PPAS0 - Feature card physical address
  strobe - ts
  from master - bi-dir
  a valid address is on the bus

  PR1W0 - Feature card read-write - ts
  from master - bi-dir
  indicates direction of data transfer

  PSIZE160 - Feature card size - oc
  from feature card to system board
  indicates 8 or 16 bit card

  RQRST0 - Request system reset - oc
  from feature card to CPU

  SYSRST0 - System reset - tp
  from system board to all feature cards

tp = totem-pole
ts = tri-state
oc = open-collector

I/O BUS SIGNALS (CONT.)

Currently, PPA[23-22]1 are not used by the feature cards when accessing DPDRAM but they must be driven low when the other address lines are being asserted. Using the lower 22 address lines, the feature cards can address 4 megabytes of main memory.

I/O BUS SIGNALS (CONT.)

- Address bus - (24)
  PPA[23-00]1 - Physical address - ts
  from master to slave - bi-dir

- Data bus - (16)
  PD[15-00]1 - 8 or 16 bit data words - ts
  bi-dir

- Bus exchange - (3)
  PBRQ0 - Feature card bus request - oc
  one line from all feature cards to system board

  PBACKI0 - PBACKO0 - Feature card bus
  acknowledge - tp
  from system board to all feature cards
  daisy-chained through all of the feature cards
  PBACKI0 & PBACKO0 are the signals into
  and out of the card

tp = totem-pole
ts = tri-state
oc = open-collector

I/O BUS SIGNALS (CONT.)

The interrupt signal PINT[2]0 has the highest priority and PINT[0] the lowest.

NOTE: PINT[2]0 is reserved for future system use, and should not be used by any feature cards.

I/O BUS SIGNALS (CONT.)

- Interrupt - (9)

  PINT[2-0]0 -(3) - Feature card interrupt - oc
  from feature card to system board
  each feature card can only use one of the three lines

  PIAKI[2-0]0 & PIAKO[2-0]0 - (6) - Feature card interrupt acknowledge - tp
  from system board to feature cards
  daisy-chained through all of the feature cards
  PIAKI[2]0 & PIAKO[2]0 are the signals into and out of the card

tp = totem-pole
oc = open-collector

I/O BUS SIGNALS (CONT.)

The backup battery can supply up to 10 microamps per feature card.

I/O BUS SIGNALS (CONT.)

- Error reporting - (2)
  - PFAIL0 - Feature card board fail - oc
    from feature card to system board
    indicates a sanity failure
  - PFLT0 - Feature card bus fault - oc
    bi-dir - from slave to master
    indicates a parity error or bus timeout

- Power - (21)
  - VCC - +5 volts - (3)
  - GND - Ground - (15)
  - V12P - +12 volts - (1)
  - V12N - -12 volts - (1)
  - VBKUP - +3 volts backup - (1)

oc = open-collector

UNIT 1

CPU - INPUT/OUTPUT INTERFACE

Lesson 4

Input/ Output Bus Timing Requirements

TIMING

Because of the time that it takes for signals to travel the length of the I/O Bus, it is very important to understand where the signals are being measured when talking about the bus timing. Most of the diagrams in this book will discuss the times as seen at the feature card connector end of the bus.

- Time of flight - For a fully loaded bus, a maximum time of flight of 40 nanoseconds is possible. This number directly affects the overall timing of a bus cycle.
- Setup times - 20 nanoseconds - The time of flight must be added to this. For example, to ensure 20 nanoseconds setup time at the slave, the bus master must guarantee a stable address for 60 nanoseconds before activating the strobe (PPAS0).
- Hold times - 10 nanoseconds - To ensure this 10 nanoseconds at the slave, the master must wait at least 50 nanoseconds after removing the address strobe before it removes the address.
- Daisy-chain delay - 9 nanoseconds - PBACK0 is daisy-chained through all of the feature cards. For each card the delay from PBACKI0 to PBACKO0 should be less than 9 nanoseconds. A circuit made up of one Schottky gate delay plus the time of flight through the board wiring and the two card connectors can meet this requirement.
- Bus cycle termination times - As one bus cycle ends, another may soon be starting; so care must be taken to prevent overlap between the current and previous bus occupants. The bus master always dictates when the cycle can be ended, but it can only do so after the data has been received. The action that ends the cycle is the removal of the address and data strobes (and PBUSY0 if a feature card is master). After deactivating the strobes, the master must wait at least 40 nanoseconds before deactivating its other signals.
- Bus cycle length - timeouts - 5 microseconds - This is measured from the time PBACK0 is asserted until the address strobe PPAS0 is deactivated. A bus timeout or data parity error will produce a PFLT0 signal that sets a system board Control and Status Register (CSR) bit and then interrupts the system board CPU.

TIMING

- Time of flight 40 nS
- Setup times 20 + 40 nS
- Hold times 10 + 40 nS
- Daisy-chain delay 9 nS
- Bus cycle termination times
- Bus cycle length - timeouts 5 nS

12 in/nS velocity of electricity.

RQRST0 TIMING

The Request System Reset (RQRST0) signal is sent from a feature card to the system board CPU to cause a system reset. Although the use of this capability will probably be limited, a typical application would be the resetting of a remote node on a computer network.

NOTE: All signals discussed in this section are as seen at the feature card connector.

The following nomenclatures will apply on **all timing diagrams** in this book.

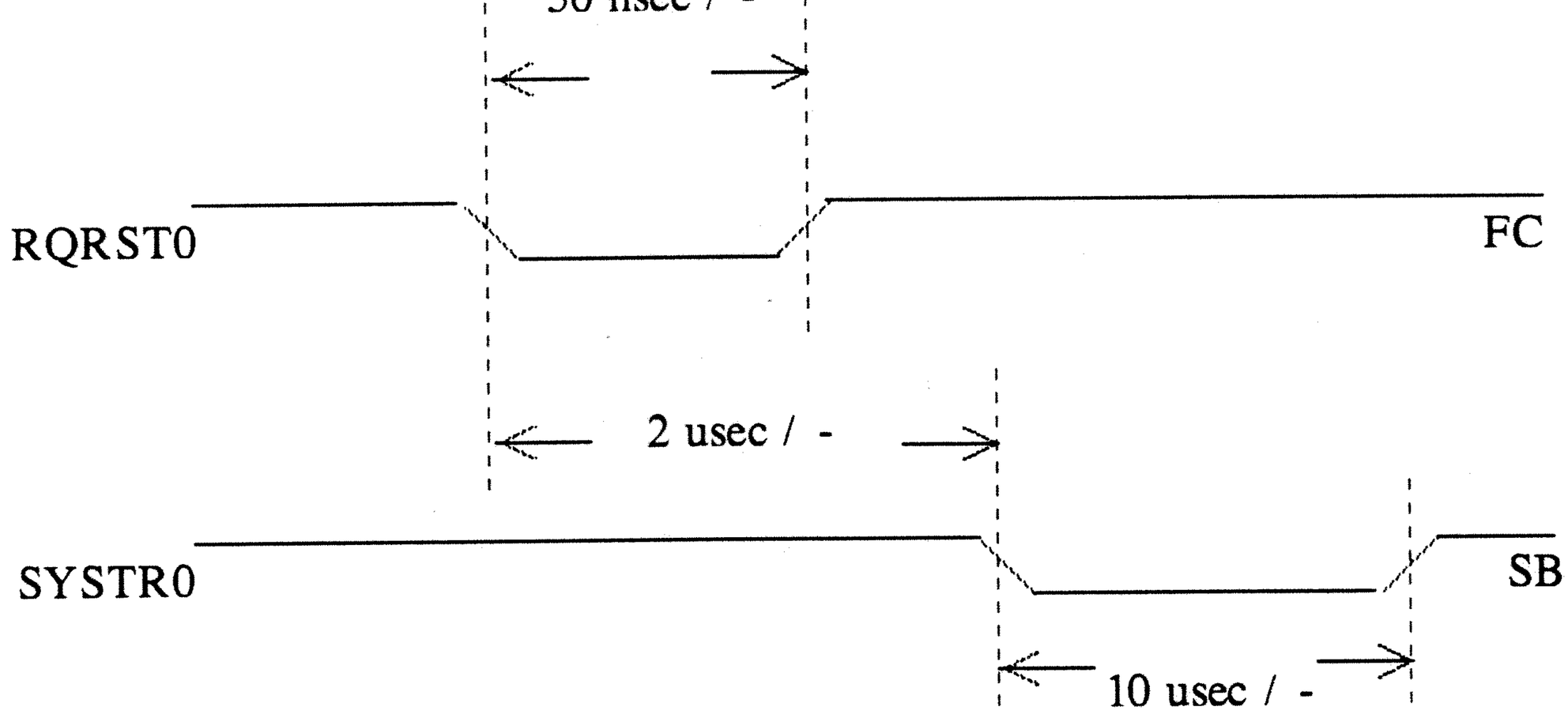
All times shown are in nanoseconds, and are specified in a minimum/maximum format.

(- means don't care)

FC indicates a signal that originates in the Feature Card.

SB indicates a signal that originates in the System Board.

RQRST0 TIMING



EIGHT I/O BUS OPERATIONS

The eight different types of I/O Bus operations are shown here. A feature cards can not access another feature card or any circuits on the system board except for the main memory and the interrupt circuit.

EIGHT I/O BUS OPERATIONS

1. SB Read of FC
2. SB Write to FC
3. FC Read of Main Memory
4. FC Write to Main Memory
5. FC Interrupt of SB CPU
6. FC Asserts PFAIL0
7. SB Reset of all FC
8. FC Reset of System

UNIT 1

CPU - INPUT/OUTPUT INTERFACE

Lesson 5

Lab Exercise #1

LAB PROCEDURES

For the lab exercises in this course you will be using a 3B2 Computer with two terminals. One terminal is attached to the console port, and the other one (labeled user) is attached to the HR1 feature card. Many of the procedures that you will be doing will require that you be logged in as super user or root.

When using the root password, PLEASE use extreme care, as you will have the power to corrupt and destroy the system. It is very important that these machine be available for all of your exercises without loss of time. To fully restore a corrupted system will take about one and a half hours.

Many of the lab exercises will require you to use a scope probe to take measurements on the feature card. Several precautions should be taken so as not to damage the feature cards. Be careful not to short any of the conductors on the cards to another conductor or to ground. Always touch some grounded metal before handling a card to minimize the danger from electrostatic discharge (ESD). Handle a feature card only by the faceplate or the edges. Never touch the components or connector pins.

Please use extreme caution while doing the lab exercises. If in doubt, please consult the instructor. Do not corrupt the system.

The operation of the HR1 feature card is described in Appendix G.

LAB PROCEDURES

- Terminals

- Logins

  Root Password

  iodrv

- Card Handling

  Shorts

ESD

LAB EXERCISE #1

Equipment required:

3B2 Computer

HR1 Custom Feature Card (installed in 3B2 Computer)

HR1 Schematic Diagram

Console Terminal

User Terminal

High quality, 50 MHZ bandwidth, Dual Trace Oscilloscope.

LAB EXERCISE #1

This lab session will enable the student to become familiar with what activity is on the I/O Bus during the normal multiuser/multitasking operation mode. The I/O Bus signals were described in Unit 1, Lesson 3.

1. Turn on the terminals and notice that they are labeled. One is attached to the console port and the other is attached to the HR1 card serial port. The console terminal will serve as the console for the system and will receive messages sent to the console by our driver. When you login as root, be careful as to where you go and what you touch. You are super user and a corrupted system will require considerable time to restore. After you are logged on, be sure to look at your .profile and be aware of how we have set your logins. Login on the console terminal as instructed.

2. Turn on the oscilloscope and set it as follows:

   Vertical Amplifier (single trace) 0.1 Volts/Div

   Horizontal Amplifier 1 us/Div

3. Make sure that the 3B2 Computer is in the "Quiescent Stage". The Quiescent Stage is being defined here as a state when no I/O is being performed (i.e. No typing at the terminal's keyboard, no displying on the terminal's CRT, etc.) The console terminal displays the prompt and is waiting for a command.

LAB EXERCISE #1 Answer Page.

	Step 4	Step 5	Step 6
Q1	________	________	________
Q2	________	________	________
Q3	Active Lines	Active Lines	Active Lines
	________	________	________
	________	________	________
	________	________	________
	________	________	________
	________	________	________
	________	________	________
	________	________	________
	________	________	________
	________	________	________
	________	________	________
Q4	________		

LAB EXERCISE #1

4. Examine the HR1 schematic Drawing. Notice that it is composed of several sections marked FILE f1, FILE f10, FILE g1, etc. Sections f6 and f4 show address and data connections. Probe (by touching the scope probe to the IC pins) the Data Lines PD001-PD151 and the Address Lines PPA001-PPA151. Notice the activity (if any) of these lines and voltage levels (TTL level Low, High, or Tri-state).

   Q1 What activity is there on Address Bus when the 3B2 Computer is in the "Quiescent State"?

   Q2 What activity is there on Data Bus when the 3B2 Computer is in the "Quiescent State"?

   Q3 Is there any activity on any of the other Bus Lines in the "Quiescent State"? List the Bus Lines that show any activity.

   Q4 What is the logic level of Address and Data Lines if no activity is detected?

5. Repeat step 4, but this time while the 3B2 Computer is performing some I/O operation. A simple I/O operation would be something such as typing on the computer terminal's keyboard or displying information at the terminals CRT. For example, typing the "ls -l" command will invoke an I/O operation listing the contents of the current directory on the console terminal.

6. Repeat step 4 while the 3B2 Computer is communicating with a Feature Card. This can be done by executing the program **lab1.**

UNIT 2

PROGRAMMED (dumb) FEATURE CARDS

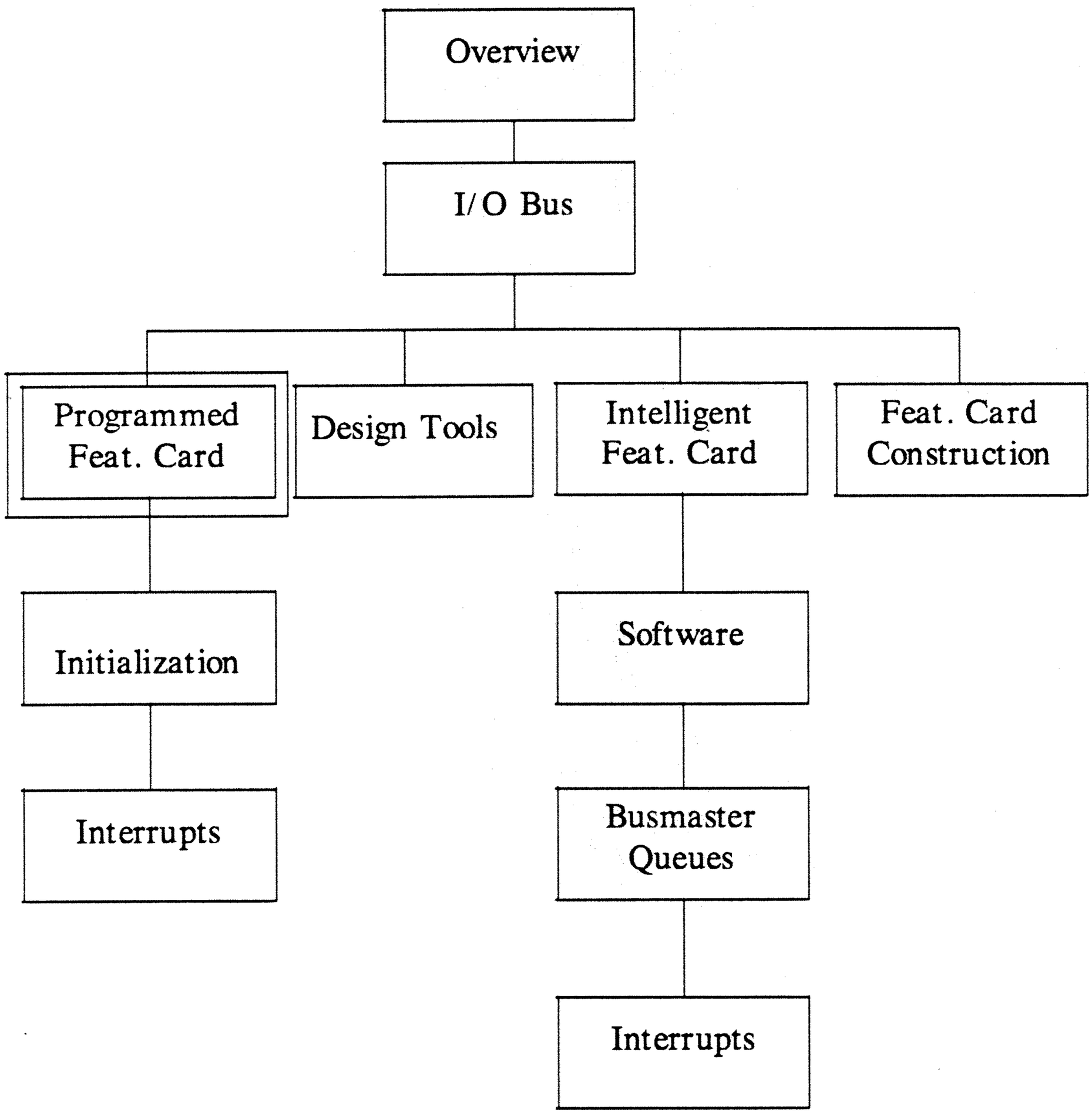
PROGRAMMED (dumb) FEATURE CARDS

Upon completion of this unit and given a 3B2, two terminals, HR1 Custom feature card, HR1 schematic, dual trace oscilloscope and student guide, the student will be able to:

1. Describe system board read/writes to feature card, power up routine, file system, self configuration, EDT table, and initialization processes.
2. Determine the ID number of the HR1 feaature card using appropriate references.
3. Discover the ID Numbers and feature cards supported by the laboratory 3B2 computer.
4. Display and interpret current system configuration information using the "prtconf" command.
5. Change the current ID number of the HR1 feature card by changing the SW1 switch setting on the dip switch and interpret the results after login.
6. Examine the "edt_data" file and compare this with the output from the "prtconf" command.

The first lesson in this unit will discuss the procedures used when the system board either reads or writes to a programmed (dumb) feature card.

2302 Outline



UNIT 2

PROGRAMMED (dumb) FEATURE CARDS

Lesson 1

System Board Read/ Write of Feature Card

SYSTEM BOARD READ OF FEATURE CARD

A System Board Read of a Feature Card begins when the system board issues PR1W0 and address leads PPA[23-00]1 simultaneously with a feature card select PCS0. When the select signal is received, the feature card asserts PSIZE160 defining its data width (8 bits = logic 1, 16 bits = logic 0).

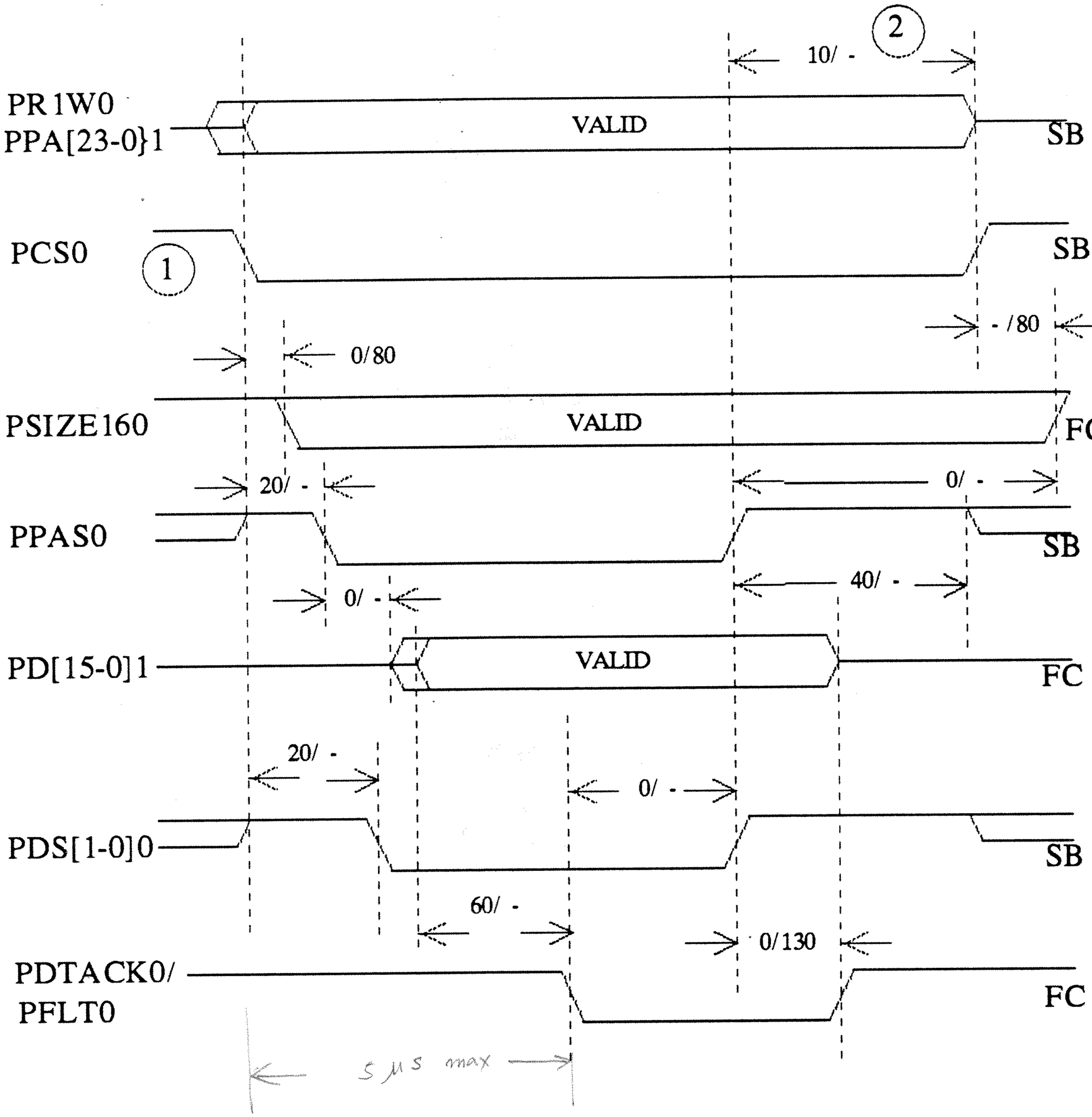
Once the address lines are stable the Physical Address Strobe signal PPAS0 is delivered to the feature card. Feature Card Data Strobes PDS[1-0]0 are then generated by the system board selecting the byte or bytes to be returned during the data bus transaction. The feature card then places its data on the I/O bus PD[15-00]1 and asserts PDTACK0 after set-up time. Following this, the system board relinquishes the bus by driving all lines inactive or tri-state. The feature card gets off of the bus when it sees the inactive strobe signals.

All signals discussed in this section are as seen at the feature card connector.

TIMING DIAGRAM NOTES:

1. PCS0 has the same timing specifications as PPA[23-00]1.
2. There is a minimum of 10 nanoseconds hold time on the address after PPAS0 goes high.

SYSTEM BOARD READ OF FEATURE CARD



SYSTEM BOARD WRITE TO FEATURE CARD

This operation also begins when the system board issues PR1W0 and address leads PPA[23-00]1 simultaneously with a feature card select PCS0. When the select signal is received, the feature card asserts PSIZE160 defining its data width (8 or 16 bits).

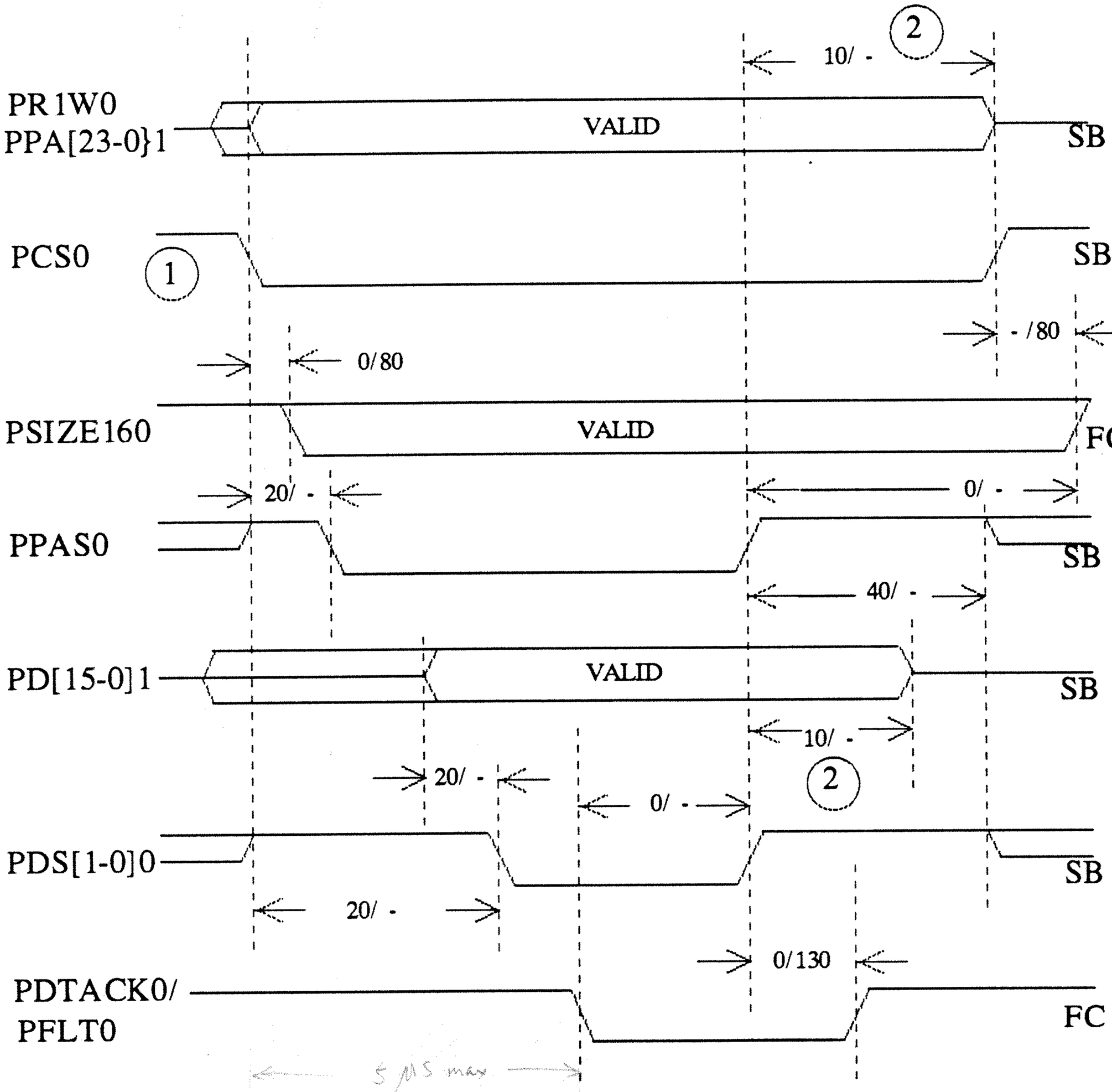
Once the address lines are stable the Physical Address Strobe signal PPAS0 is delivered to the feature card. The system board places the data on the I/O bus PD[15-00]1, and then generates data strobes PDS[1-0]0 indicating which byte(s) were placed on the 16 bit I/O bus.

Once the data has been taken by the feature card, PDTACK0 is generated to inform the system board. Should a problem take place in the data transaction, PFLT0 is sent instead of PDTACK0. As with reads, the system board relinquishes the bus first, and then the feature card gets off the bus when it sees the inactive strobes.

TIMING DIAGRAM NOTES:

1. PCS0 has the same timing specifications as PPA[23-00]1.
2. There is a minimum of 10 nanoseconds hold time on the address and data after the removal of PPAS0 and PDS[1-0]0 respectively.

SYSTEM BOARD WRITE TO FEATURE CARD



UNIT 2

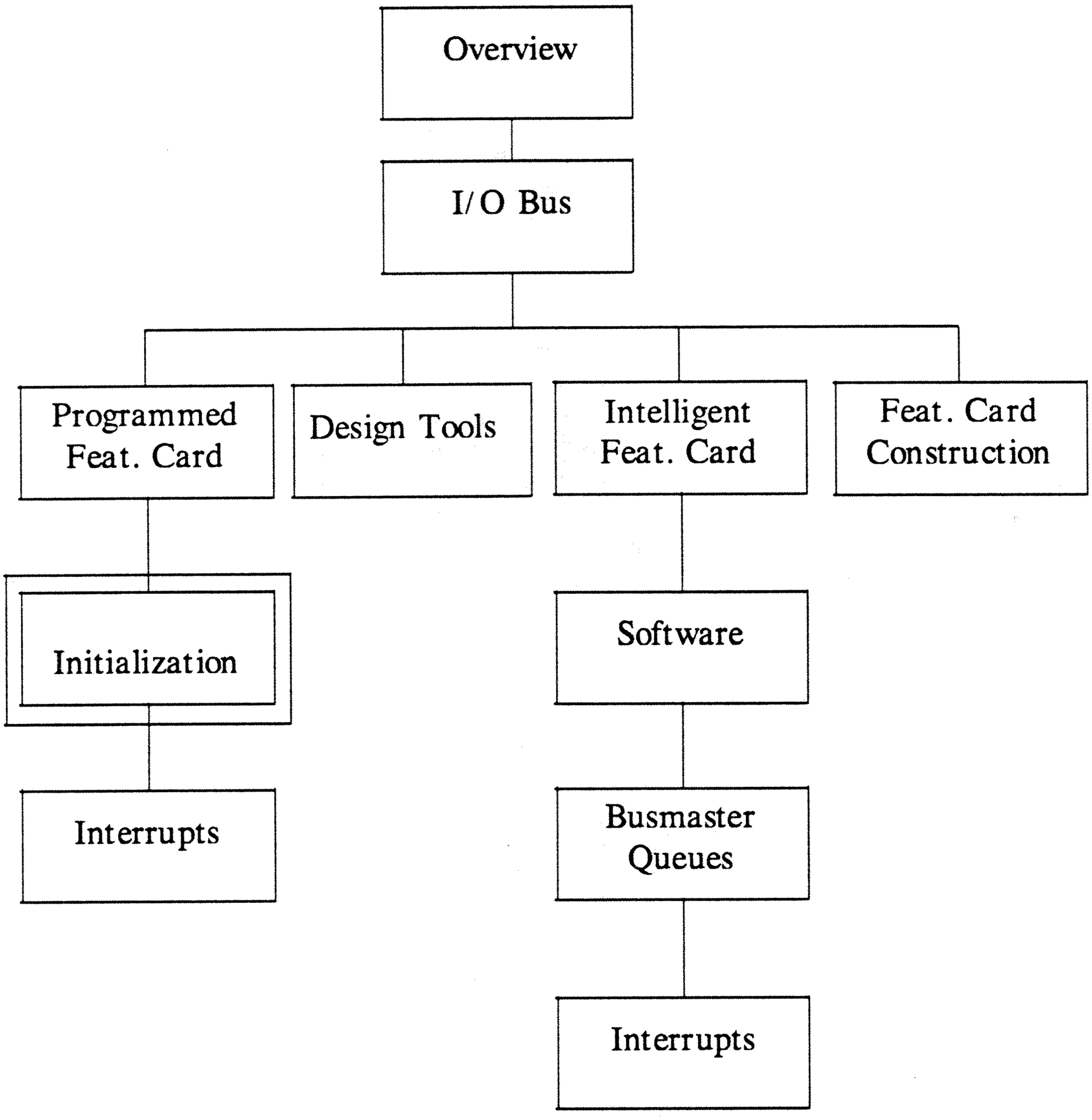
PROGRAMMED (dumb) FEATURE CARDS

Lesson 2

Initialization

The next two lessons and the lab exercise that follows will discuss the initialization procedures for the programmed feature cards.

2302 Outline



POWER UP ROUTINE

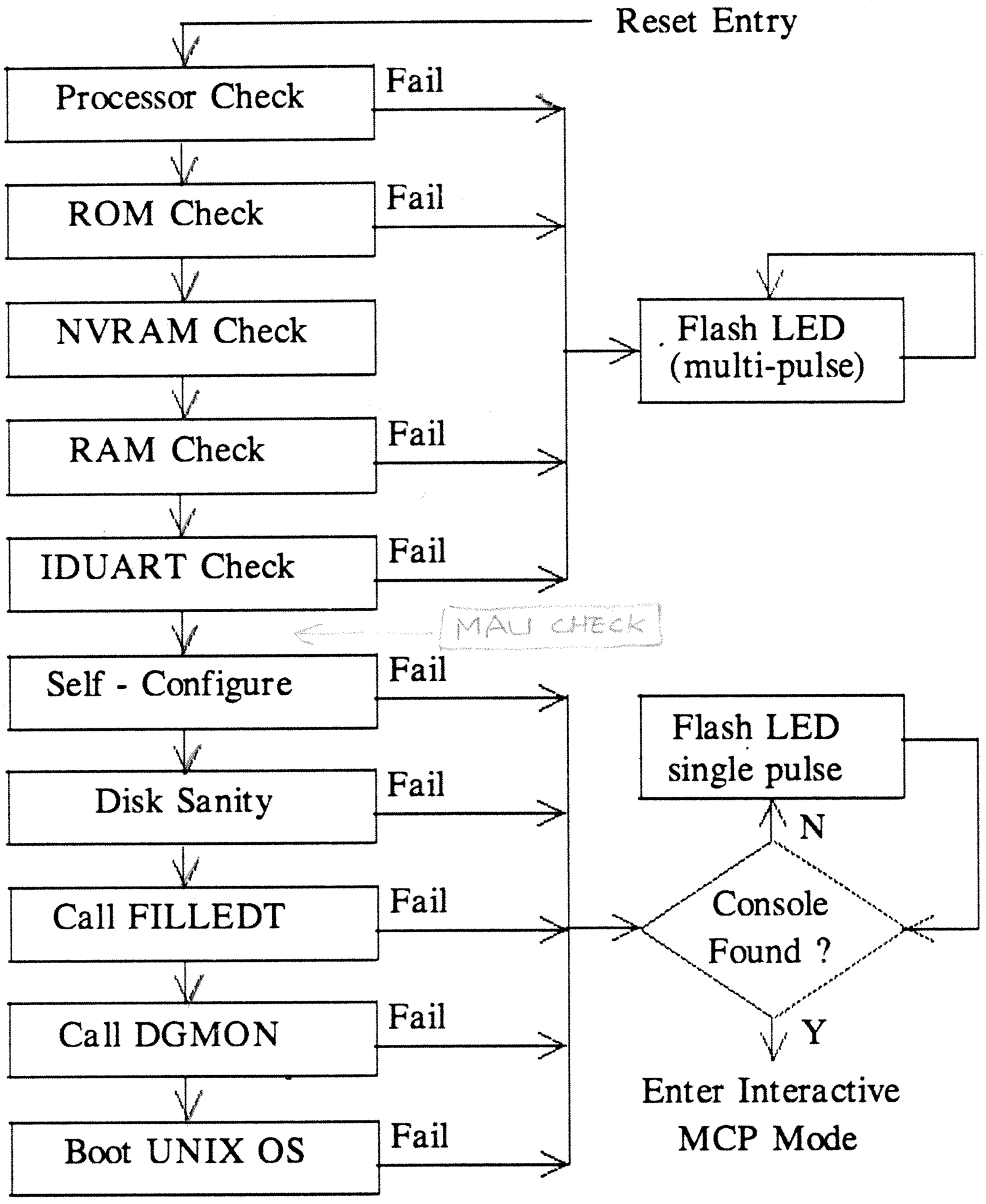
The system contains several diagnostic programs called 'Diagnostic Phases" that make up the Maintenance and Control Program (MCP). Each phase consists of a series of tests used to troubleshoot system failures or to locate intermittent problems. The Maintenance and Control Program operates in one of two modes: NONINTERACTIVE or INTERACTIVE.

The noninteractive mode is entered on power-up, where a total system reset occurs and basic sanity checks are made on the computer hardware. If a problem occurs during sanity checks, the front panel Diagnostic LED pulses on and off in a defined pattern which identifies the type of sanity failure (see the table below). Upon completion of the sanity checks a self-configuration process takes place where the computer identifies each feature card and takes note of their type and position on the bus.

After the self-configuration terminates, more extensive diagnostics are run (on the floppy and hard-disk drives) causing the computer to enter either the interactive mode of the MCP or exit altogether by booting the UNIX Operating System. The UNIX Operating System is only entered if no failures occurred during sanity checks, self-configuration, or diagnostics. If a problem occurred after sanity tests, and a console was identified during the self-configuration process, the interactive mode of the MCP is entered which will prompt for a password. If no console is found during the self-configuration process, the front panel Diagnostic LED will blink off and on until a terminal is connected to the default console port.

Pulse count	Failure Type
1	System is at firmware null state with no console terminal.
2	Processor sanity test failed.
3	ROM sanity test failed.
4	RAM (first 16K) sanity test failed.
5	IDUART sanity test failed.

POWER UP ROUTINE

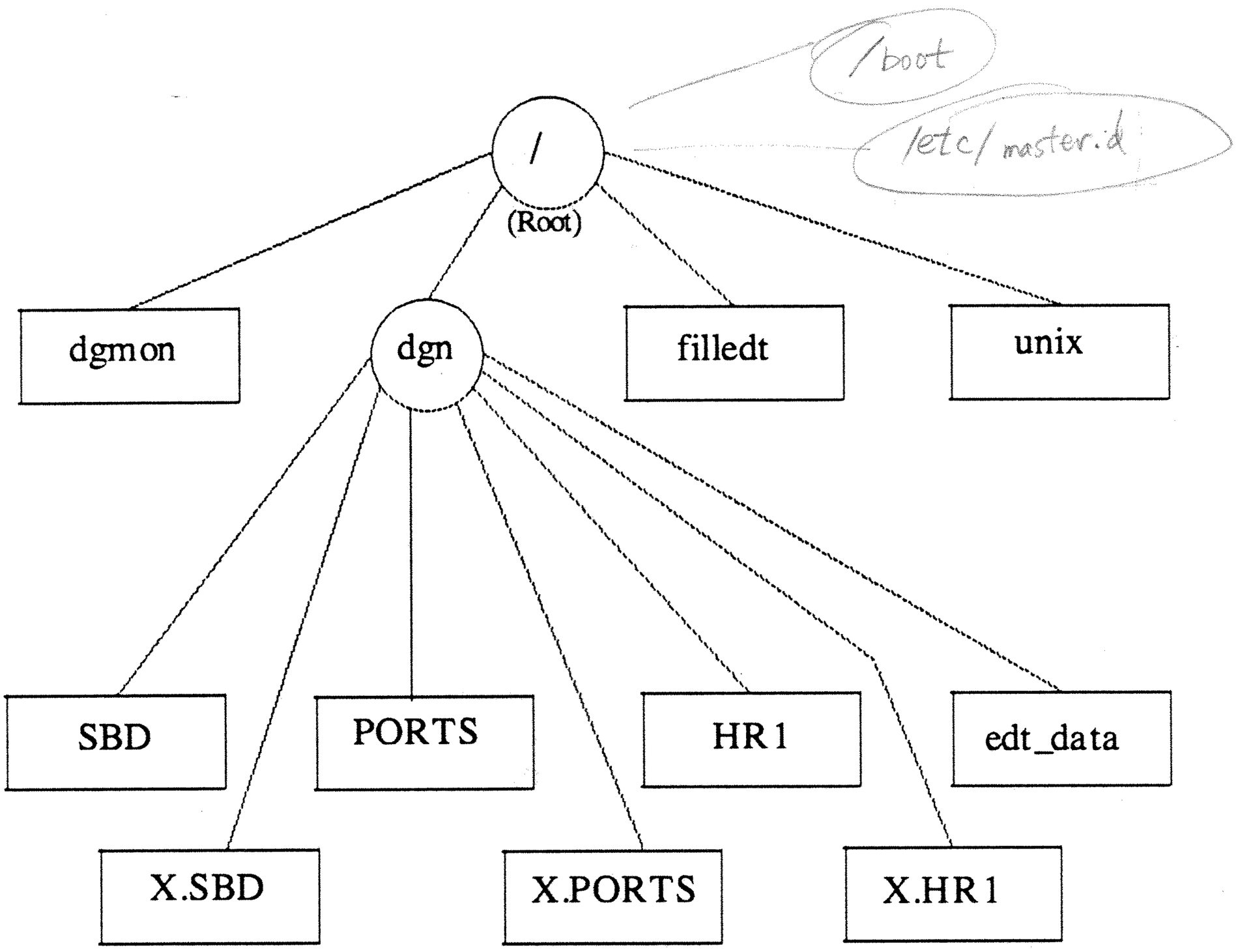


FIRMWARE UTILITY FILE SYSTEM

The Maintenance and Control Program (MCP) is contained in firmware located in the computer's EPROM along with boot firmware for the diagnostics and utilities. The diagnostics and utilities reside on disk in the ROOT file system.

The INIT file system that was formerly used has been merged into / (root). The /dgn directory contains two diagnostic files per device type. For the ports board these files would be PORTS and X.PORTS. The PORTS file contains the phase table for the ports card diagnostics, and the X.PORTS file has the actual diagnostics that are downloaded to the card. As new feature cards are added, their diagnostics and phase tables must be added to the /dgn directory. This means that new feature cards may be added without changing the EPROM contents. All that is required is that the file containing the correct diagnostic phases is added to the disk under the root file system. The **edittbl** command lets users change the file "/dgn/edt_data" that **filledt** reads during self-configuration to set the device and subdevice lookup tables.

FIRMWARE UTILITY FILE SYSTEM



DYNAMIC SELF CONFIGURATION - BOOT TIME

The boot process links together a kernel and a set of drivers to form a complete system. Independently compiled driver objects must be preprocessed by the **mkboot** command and stored in the /boot directory. Two types of specification files direct the self configuring process. The first type is a set of files contained in the /etc/master.d directory. Each file contains configuration information for a particular hardware or software driver. This includes the driver's entry points, interrupt vectors, and memory requirements over and above the data required by the driver object. Interrupt vectors and memory are usually a function of the number of devices equipped : for example, separate interrupt vectors and status data structures must be provided for each controller. The **mkboot** command extracts this information and adds it to the driver objects stored in the /boot directory. The collection of files in /etc/master.d defines the set of drivers supported by the system. Note that this set of files has the same function as the single /etc/master file on older systems and it is often convenient to refer to these files collectively as the **master** file.

The /etc/system file specifies the set of drivers that are to be included in the system. This may possibly supplement the information obtained from polling self-identifying hardware to determine the actual configuration. During boot, the needed drivers are merged with the kernel on a section at a time basis. That is, the text sections are merged, then the data sections (i.e., initialized data), and finally the bss sections (i.e., uninitialized data). The text and data sections are loaded with any initialized data structure being set up as required and space is allocated for the combined bss. Any unresolved external references detected in the kernel and the drivers are resolved at this time using symbol table information obtained during loading the various object modules and memory allocation. At the end of this process, if there are any remaining unresolved references, a warning message is printed and the boot process continues. A combined symbol table is also constructed and left in memory for later use.

The results of a self configuring boot, referred to as an "absolute" kernel is saved by the **mkunix** command. The /unix file produced in this way is an image of the complete system as booted, including symbol table. It can be used subsequently to boot the system directly, saving the time needed for the dynamic linking operation in a self configuring boot.

DYNAMIC SELF CONFIGURATION BOOT TIME

links kernel and drivers

mkboot

/etc/master.d directory

/etc/system

mkunix

/unix

EQUIPPED DEVICE TABLE

The Equipped Device Table (EDT) is a data structure built and maintained by the resident firmware. The EDT contains information for each peripheral installed on the system. In order to perform the hardware self configuration, the boot program accesses the EDT to get the identity and hardware address of each peripheral. All software drivers for hardware devices that do not appear in the EDT must be explicitly included in the /etc/ system file.

See Appendix E for information about the **edittbl** command that is used to build the /dgn/edit_data file.

EQUIPPED DEVICE TABLE (in DPDRAM)

Data structure built and maintained by the resident firmware

Identity and hardware address

/etc/system

MASTER FILE

The master file, or actually the collection of files in the /etc/master.d directory, is a **database** of the device hardware characteristics required by the UNIX operating system. Additional entries are included for non-device-related configurable modules or software drivers. (The kernel also has a master file entry which specifies the values of tunable system parameters.)

The master file is used by the **mkboot** program to obtain device information when generating the device driver files and by the **sysdef** program to obtain the names of supported devices. It also contains specifications for the generation and optional initialization of all memory resident data structures required by a module.

The Master File is explained in more detail in Appendix C.

MASTER FILE

Collection of files in / etc/ master.d

Database of the device hardware characteristics required by the UNIX Operating System

Used by the **mkboot** for device information

Initialization for all memory resident data structures required

SYSTEM FILE

The /etc/system file contains configuration information that cannot be obtained automatically at system boot time. This file generally contains a list of software drivers to include in the load as well as instructions for manually overriding the drivers selected by the boot program, as part of its interrogation of the EDT. It also includes the assignment of the system devices **dumpdev, rootdev, pipedev,** and **swapdev**.

The System File is explained in Appendix D.

SYSTEM FILE

/etc/system

- Contains information that is not obtained automatically at system boot time
- List of software drivers to include
- List of software drivers to exclude
- Instructions for manually overriding the drivers

UNIT 2

PROGRAMMED (dumb) FEATURE CARDS

Lesson 3

Feature Card Initialization

SELF-CONFIGURATION

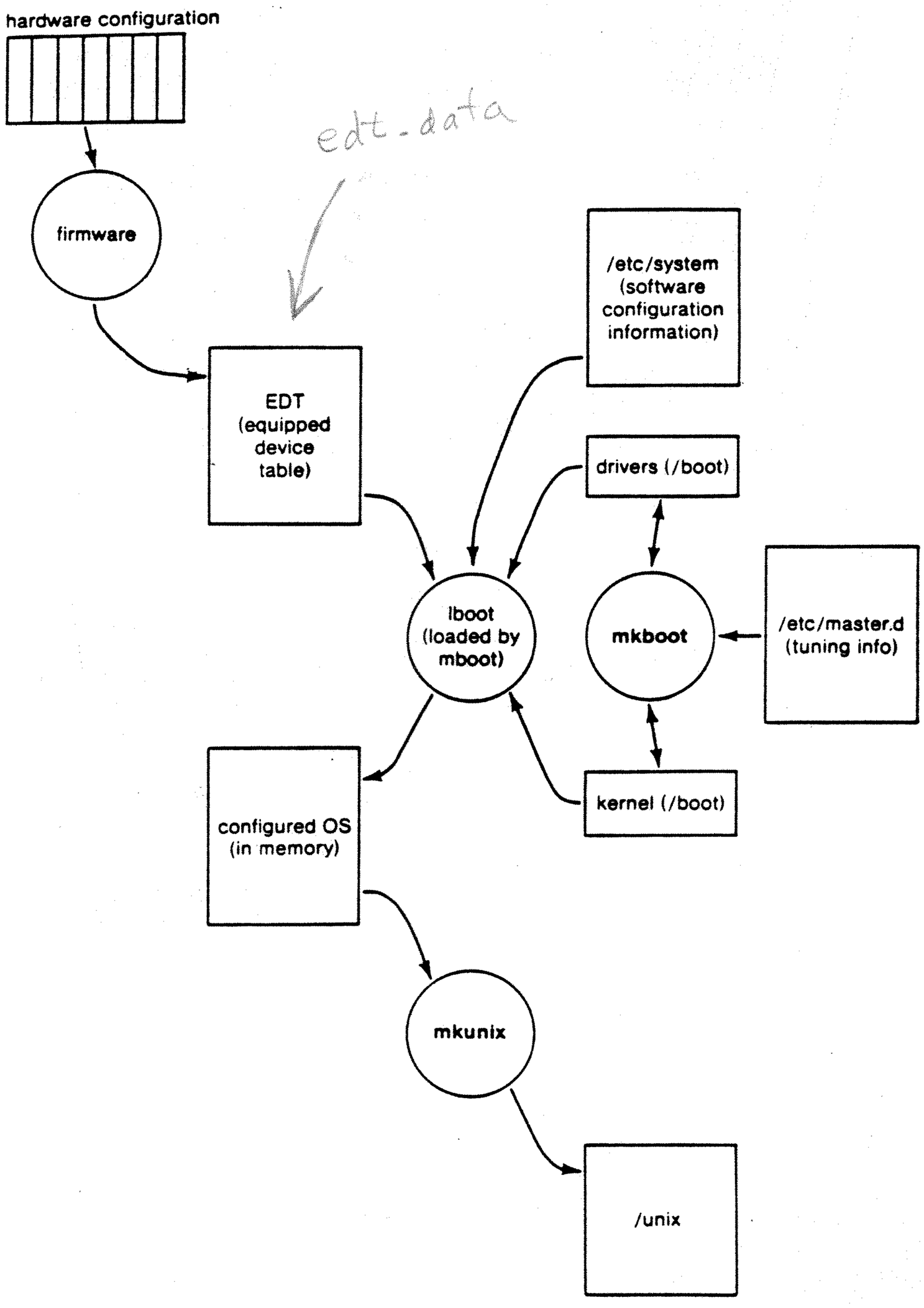
AT&T 3B2 and 3B5 Computers use a process known as self-configuration to manage hardware and software configuration. This procedure is different from the procedure used by other machines running UNIX System V, such as VAX and AT&T 3B20 Computers.

Principal features of self-configuration include:

1. Automatic loading of drivers necessary to support the system's hardware configuration.
2. Selection of software drivers for loading at boot time
3. Development and loading of local drivers without kernel source.
4. Inclusion of only required drivers in the kernel.
5. Generation of pre-configured kernel (/unix).
6. Less impact on the operating system for features that require new drivers. New hardware and its corresponding drivers can be released without releasing a completely new operating system.

Every feature card installed in the 3B2 Computer must have software associated with it which will tell the operating system (UNIX) how to communicate with the feature card. This software is called a HARDWARE DRIVER. In the Unix Operating System hardware drivers must be include as a part of the kernel. A hardware driver must be present in a special file at the time the feature card is installed. The match between feature card and its associated hardware driver is performed by the system board based on feature card's unique IDENTIFICATION CODE.

SELF-CONFIGURATION



INITIALIZATION PROCESS

The following discussion pertains to both programmed and intelligent feature cards.

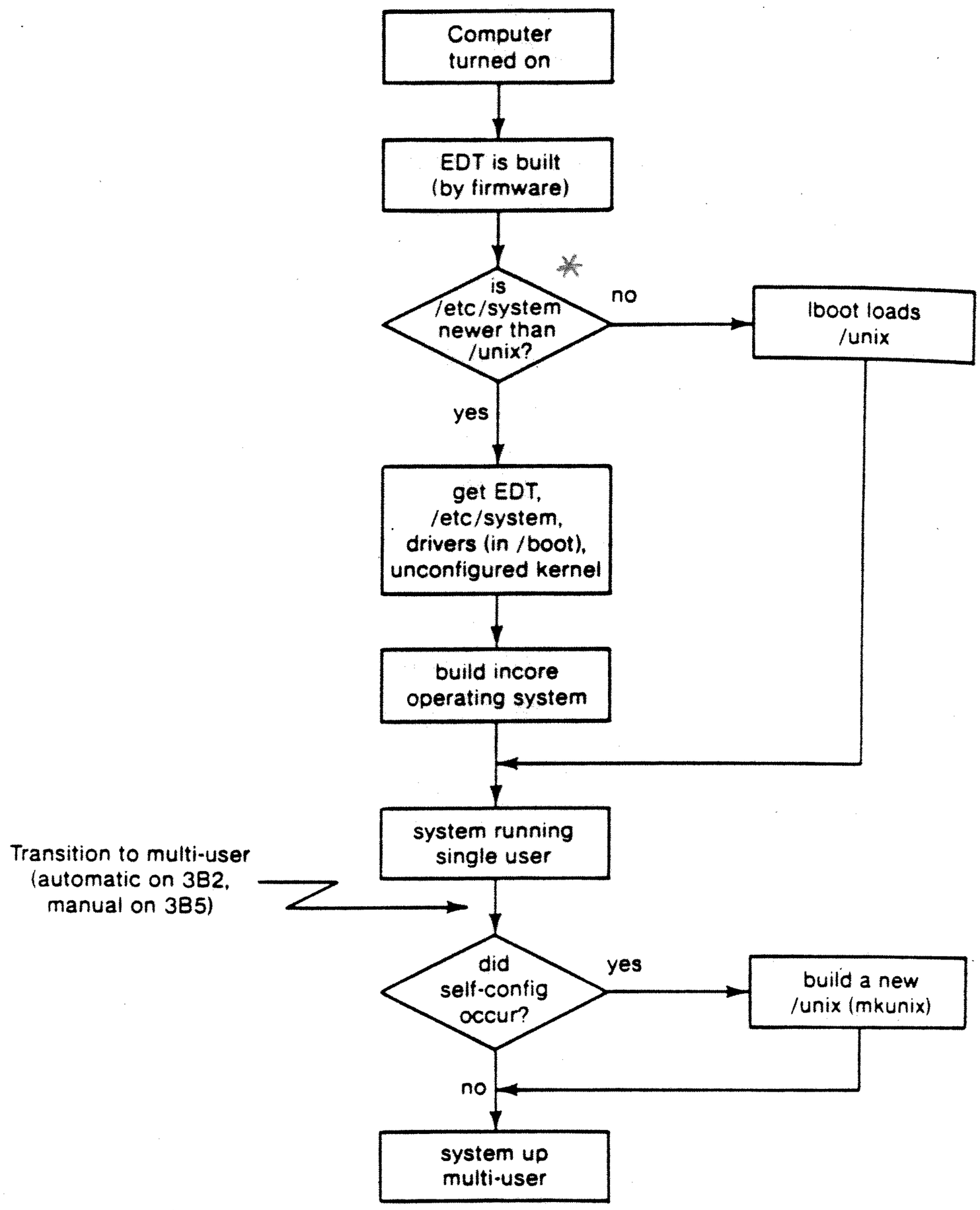
During "power up," the system board signals all feature cards that the initialization process is taking place by asserting SYSRST0 signal. A feature card should detect this condition and act upon it by writing its IDENTIFICATION CODE into the Identification/Vector (I/V) register. A feature card has 100 milliseconds (max.) to write this code into I/V register. The I/V register resides at address 000001 Hex of the feature cards address spectrum and it is a physical register rather then a memory location of the onboard microprocessor. The Identification Code (assigned by AT&T) is usually resident in the feature card's PROM or set by DIP switches.

[HR1 feature card, used in the class exercises, has its ID code set by DIP switches SW1. Settings of the SW1 DIP switch are latched into octal latch IC 9. Upon detection of SYSRST0 the HR1 card transfers the latched data from IC 9 octal latch into the I/V register (IC 6). Once the data is latched into the I/V register, the feature card ID code is ready to be read by the system board.]

Next, the feature card is waiting to be identified. This happens when the system board performs SYSTEM BOARD READ OF FEATURE CARD. The feature card should adhere to the SYSTEM BOARD READ OF FEATURE CARD signals and timing protocol during this time.

The system board reads address 000001 of the feature card and the feature card responds with its Identification Code. Feature cards have an address range of 1FFFFF Hex (2,097,151 Decimal). Next the system will examine file "/dgn/etd_data" to see if the Hardware Driver asked for by the feature card exists. (File "/dgn/edt_data" defines the set of drivers supported by the system. However, if the hardware driver has just been added to "/dgn/edt_data" but the corresponding feature card does not exist on the I/O Bus - no new unix will be generated). Entrys in "/dgn/edt_tbl" that correspond to a given feature card, point to a software handler assigned to that card in the "/boot" directory.

INITIALIZATION PROCESS



GENERATION of a NEW UNIX

A new unix will be generated during the "Power On Sequence" if any one of the following is true:

1. A feature card is found on the I/O Bus who's ID number correspond to the ID number found in file "/dgn/edt_data" but its software handler is not part of the existing unix (card has just been plugged in).

2. A feature card's ID is defined in "/dgn/edt_data" and its corresponding software handler is included in the existing unix but the feature card is NOT found on I/O Bus (card just has been removed).

3. The file "/etc/system" is newer than the existing "unix". File "etc/system" is used to specify software drivers to be included in the generation of a new unix, and to exclude hardware drivers although they are specified in "/dgn/edt_data" file and the corresponding feature cards reside on the I/O Bus.

In summary - The system board correlates the location of the feature card on the I/O Bus by address and Identification Code read; therefore, the system board knows the type of the feature card and where on the bus it is located. Furthermore, the system will know what software to use to control the feature card by associating the feature card's ID number with appropriate software handler.

The above discussion pertains to both intelligent and programmed feature cards and represents the ABSOLUTE MINIMUM requirements for a feature card to be functional on the 3B2 I/O Bus. If the feature card does not use interrupts or the pump code feature, the initialization of the feature card will normally end here.

Cases when a feature card uses interrupts or pump code will be discussed in later lessons.

GENERATION of a NEW UNIX

A new UNIX is generated if:

1. A new feature card is plugged in
2. A feature card has been removed
3. **/etc/system** is newer than **/unix**

4 1. & 2. (different cards interchanged)

5 root directory corrupted.

6 etc. ...

UNIT 2

PROGRAMMED (dumb) FEATURE CARDS

Lesson 4

Lab Exercise #2

LAB EXERCISE #2

This page is for notes.

LAB EXERCISE #2

Equipment required:

3B2 Computer

HR1 custom feature card (installed in 3B2 Computer)

HR1 schematic diagram

Console terminal

User terminal

This lab session will enable the student to gain an understanding of the initialization process for feature cards installed in the 3B2 Computer INPUT/OUTPUT bus.

1. Information gained in this step might be useful in debugging a newly constructed feature card . It will help you to determine if the feature card is able to be "tolerated" by the 3B2 Computer up to the point of identification.

   Turn off the 3B2 Computer, and then turn it back on. Make note of the messages appearing on the console terminal, user terminal and the Diagnostic Light action (the red LED on the front of the computer). Try to "catch" the timing sequence of the messages appearing on both terminals and the diagnostic light action. It may be necessary to repeat this process to "catch" the timing of the events. Turn on the 3B2 Computer and login as "root".

2. Refer to the class material and Appendix E on the **edittbl** command. (Appendix A,B,C,D might also be helpful.) Based on the above material determine the ID number of the HR1 feature card.

LAB EXERCISE #2 Answer Page.

1. Sequence of events

1. light
2. id - - -
3. self-check

2. HR1 ID = 72

3. Other Devices SBD PORTS

ID Numbers 1 5

4. Configuration

Difference

Why

5. Console

User Terminal

New UNIX?

6. Difference no

7. Console

User Terminal

New UNIX?

LAB EXERCISE #2

3. Using the same reference material as in step 2 above, determine what "other" devices or feature cards are supported by the computer. What are the ID numbers of the "other" devices.

4. The command "**prtconf**" displays information taken directly from the current UNIX about the current system's configuration. Note the configuration. Is the configuration information different from the information obtained in step 3 above? If different why? What does that mean?

5. Shutdown the 3B2 Computer. Change the ID code of the HR1 feature card by changing the setting of SW1 DIP switch to 70. Turn on the 3B2 Computer and observe both the console and user terminal. Take note of the messages. Was a new UNIX generated?

6. Following step 5 above, use the "**edittbl**" command to examine "edt_data". What is the difference (if any) between the configuration obtained in step 4 above and the present configuration?

7. Shutdown the 3B2 Computer. Set the ID code of the HR1 feature card back to the original ID number. Turn on the 3B2 Computer and observe the messages on the console and user terminal. Was a new UNIX generated this time?

UNIT 3

DESIGN TOOLS

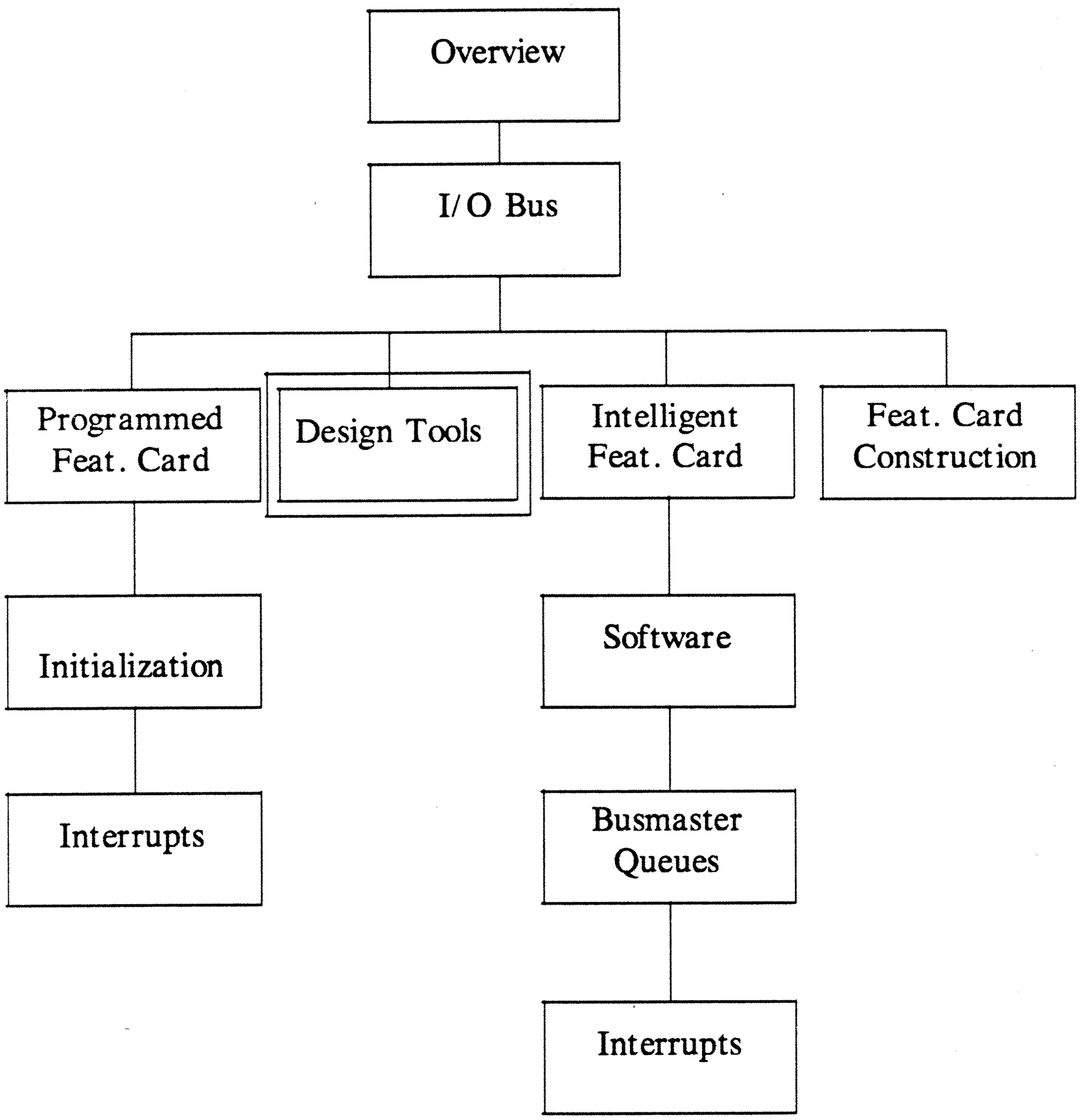
DESIGN TOOLS

Upon completion of this unit and given a 3B2, two terminals, HR1 Custom feature card, HR1 schematic, dual trace oscilloscope and student guide, the student will be able to:

1. Describe, analyze and apply the concepts of MCP, diagnostic monitor, dgn, demom debug monitor, and emulator.
2. Read and write Hex code to the installed HR1 feature card.
3. Use Demon Debug Monitor to debug feature card hardware and firmware.
4. Write a character string to the HR1 user's terminal.
5. Input a character string on the HR1 user terminal keyboard and read it on the 3B2 console.
6. Write and store hexcode in the HR1 RAM starting at a given address and read it back.
7. Manipulate the HR1 feature card's ICs and interpret the results.

This unit will discuss the use of the Diagnostic Monitor, and various DEMON Debug Monitor commands as tools to aid in designing and debugging new feature cards. It will also cover the use of the Intel Personal Development System as another tool to aid in the design process.

2302 Outline



UNIT 3

DESIGN TOOLS

Lesson 1

Diagnostic Monitor

MAINTENANCE and CONTROL PROGRAM (MCP)

Maintenance is an important part of any digital computer system. Routines should be run upon powerup to insure the integrity of the hardware. If for any reason there are problems in the system, the console operator should know before confusion set in because of a faulty system. In the 3B2 Computer the Maintenance and Control Program (MCP) is used to control this diagnostic testing.

There are two modes in which the MCP operates: noninteractive and interactive. The noninteractive mode is entered on powerup where total system reset occurs and basic sanity checks are made on the computer hardware. The sanity checks include diagnosing of its own processor module, EPROM, NVRAM, IDUART and dual port dynamic RAM (DPDRAM). If a problem occurs during sanity checks the console receives a message about the sanity failure.

Upon successful completion of its sanity checks a self-configuration process takes place where the computer identifies each feature card and takes note of their type and position on the bus.

After self configuration, more extensive diagnostics are run on the floppy and the hard-disk drives causing the computer to enter either the interactive mode of the MCP or exit altogether by booting the UNIX Operating System. The UNIX Operating System is entered if no failures occurred during the sanity checks, self configuration, or diagnostics. If a problem occurred after sanity tests the interactive mode of the MCP is entered which will prompt for a password. The password is entered, and diagnostics and utilities can now be run to identify the problem. If the diagnostics fail then the system may lock-up and the console displays "diagnostics failed" and will not ask for login.

The MCP is contained in firmware located in the computer's EPROM along with boot firmware for the diagnostics and utilities. The diagnostics and utilities reside on disk in the ROOT file system.

The interactive entry to the MCP may be forced by resetting the system during a diagnostic sequence, thus simulating a failure.

MAINTENANCE and CONTROL PROGRAM (MCP)

Insure hardware integrity

Noninteractive mode

Sanity checks

Interactive mode

DIAGNOSTIC MONITOR

The Diagnostic Monitor is a program located in the root file system (/**dgmon**) that is used to run various diagnostic phases. The phases are located in the /**dgn** directory, which contains two diagnostic files per device type. There are three types of diagnostic phases present on the hard disk: NORMAL, DEMAND, and INTERACTIVE. Each phase consists of a series of tests used to determine sanity, troubleshoot system failures, or locate intermittent problems.

DIAGNOSTIC PHASES

NORMAL - The Normal phases are automatically run each time the system is powered up as part of the system sanity checks. They can also be run on demand using the Diagnostic Monitor Utility Program.

DEMAND - The demand phases are diagnostics that run only on a manual request basis via the Diagnostic Monitor Utility Program. These phases DO NOT run automatically as part of the power-up sequence.

INTERACTIVE - Interactive diagnostics are diagnostics that are manually run via the Diagnostic Monitor Utility Program and require operator intervention. The operator intervention usually consists of inserting a floppy disk into the floppy disk drive and/ or entering data via the keyboard.

WARNING

Some of the interactive diagnostic phases can cause file damage or other unrecoverable system damage such as destruction of the NVRAM contents.

DIAGNOSTIC MONITOR

PHASES

- NORMAL
- DEMAND
- INTERACTIVE

ENTERING the DIAGNOSTIC MONITOR

The console operator may call in diagnostics or special utilities from the disk via the boot command to help repair problems or perform maintenance. The utilities are a series of disk based programs designed to provide a user friendly interface for performing system maintenance. Their inclusion on disk allows the firmware to have access to functions which are usually too large to be included in the firmware subsystem EPROM, but which are attractive to the user. Thus, extended diagnostics, system exercisers, etc., may be added at user discretion by purchasing an additional floppy disk.

The computer must be in the i5 run level to use these diagnostic programs. When it is in the i5 run level "Firmware Mode" will be displayed on the console terminal. It is then waiting for the firmware password, and nothing can be done until you enter the password and a <cr>. (Unless it has been changed, the password will be "mcp".) You should then type in "boot <cr>" and it will respond by asking the name of the program that you wish to run. To enter the diagnostic monitor type in "/dgmon <cr>". It will prompt you for a "load device option number", and you should select the hard disk. Once entered the diagnostic prompt "DGMON >" will appear. If the program does not exist on the entered device, then a directory of available bootable modules will be given and the prompt sequence restarted. In the event of boot failure, the boot firmware will return control to the interactive portion of the MCP.

ENTERING the DIAGNOSTIC MONITOR

- Firmware Mode
  1. At Console
  2. As Root
  3. No Other Users
  4. **shutdown -y -i5**
  5. Firmware Password
- Execute the **dgmon** Program

boot < cr>
Enter name of program to execute:
/dgmon
Enter load device option number:
< cr> (HD is the default)

DIAGNOSTIC MONITOR COMMANDS

When the command to boot the DGMON is given, DGMON begins execution by completing the EDT, similar to the autoboot mode. After the system configuration has been completed, the DGMON command prompt is printed at the console. The DGMON functions may then be invoked and they are:

- Help - This command will provide a list of the DGMON commands and their arguments at the console. No key word is required.
- Show - This command will provide a list of devices that are included in the Equipped Device Table (EDT).
- List - This command will print a real time list of the phase table for a specific device. The list shall be segmented in groups of 15 phases until 18 or fewer phases remain to be displayed. The display shall include the phase number, phase type and its descriptive character string from the phase table.
- Quit - The quit command shall cause a return to the MCP via a system reset. No keyword will be required.
- Dgn - Run diagnostic phases.

dgn is used to run diagnostic phases. All demand diagnostics have the general form listed below, where spaces, commas, tabs, or dashes are used as delimiters. This is described further on the next page.

dgn [device [=device#]] [ph=a[-b]] [rep=n] [ucl] [soak]

DIAGNOSTIC MONITOR COMMANDS

- Help - list of the DGMON commands and their arguments

- Show - list of devices from the EDT

- List - print a list of the phase table for a specific device

- Quit - return to the MCP via a system reset

- Dgn - Run diagnostic phases.

dgn [device [=device#]] [ph=a[-b]] [rep=n] [ucl] [soak]

DGN COMMAND

The **dgn** command is used to run the diagnostic phases. All demand diagnostics have the general form listed below, where spaces, commas, tabs, or dashes are used as delimiters.

dgn [device [=device#]] [ph=a[-b]] [rep=n] [ucl] [soak]

The monitor will then download files, two per feature card in the EDT table from the ROOT file systems directory **dgn**. These files contain the diagnostic phases with accompanying phase tables. Once execution has taken place, the prompt "DGMON >" will return. Typing "q" will cause the interactive mode of MCP to be reentered.

If keywords are not specified, the computer executes all powerup diagnostic routines. If keywords are to be specified, the feature card must be included; **device#** is only used in the event that multiple feature cards of the same type exist. The **device#** given has origin zero and refers to an ascending address order of the cards. Feature cards in slots three and four would be referred to as feature card numbers zero and one respectively. If no **device#** is specified, the default is zero.

The **rep=n** specifies a decimal number of times to repeat the diagnostic. **ph=a[-b]** specifies phase range to be run (a through b). Phases are subsets of diagnostics which provide diagnostic with a specialized test. In the **ucl** unconditional mode testing will not stop when a phase fails, and the results of each phase are displayed as it is completed. A **soak** option allows the user to run extensive diagnostics on the system, or any subunit of the system, silently and repetitively.

With the exception of **soak**, the diagnostic monitor continues to download and execute phases until its parsed input conditions have been satisfied or a failure occurs. Under either of these conditions, the loop exited. If the MCP is interactive and a failure occurred, a message is returned to the user terminal and the MCP is reentered via a return. If diagnostics fail and the MCP is noninteractive, the MCP interactive mode is entered.

DGN COMMAND

OPTIONS	NORMAL	DEMAND	INTER-ACTIVE	OUTPUT ON
dgn *	x			no
dgn device	x			no
dgn device ucl	x			yes
dgn device rep=n	x			no
dgn device ph=a-b	x	x	x	yes
dgn device ph=a-b rep=d ucl	x	x	x	no
dgn soak	x	x		no
dgn soak device	x	x		no

* These are run on power-up.

UNIT 3

DESIGN TOOLS

Lesson 2

DEMON Debug Monitor

DEMON - DEBUG MONITOR

Demon is designed to aid in verifying and using hardware, and debugging software or firmware. Additional information for the use of DEMON will be found in the DEMON Manual located near the lab stations.

DEMON does all the functions associated with a monitor program. It allows you to look at all the CPU registers, insert breakpoints, and perform virtual to physical address translations. It performs all the functions normally associated with a monitor.

To use DEMON effectively requires a disassembled listing of your program.

A self-configuration feature is provided on the 3B2 Computer which has the capability called *magic mode*. This allows you to set breakpoints and examine memory before executing a new kernel.

DEMON requires UNIX System V Release 2.0 and the C Programming Utilities. DEMON exists on four EPROMs that replace the standard EPROMs in the 3B2 Computer.

The way to identify that DEMON is on your machine is by noting the initial response to firmware mode. Shutdown to the firmware mode is **-i5**. If the DEMON firmware is loaded the response is :

```
3B2 Computer Monitor/Control Program -
erase "H', kill '@'
Physical Mode
>
```

DEMON is different from the standard software utilities in that removing it does not free up space on the hard disk. Therefore, there is no reason to remove the DEMON EPROMs except to put them into another 3B2 Computer.

DEMON - DEBUG MONITOR

With DEMON you can:

Examine CPU registers

Set breakpoints

Perform address translations

To identify a DEMON equipped machine:

shutdown -i5

3B2 Computer Monitor/ Control Program -
erase '^H', kill '@'
Physical Mode
>

ENTERING DEMON

If the DEMON EPROMS have been installed in your 3B2 Computer, the DEMON commands will be available any time you enter the firmware mode. Another way to enter DEMON is through the *magic mode*. *Magic mode* allows you to gain control after the boot process but before the transfer is made to the UNIX System kernel.

To enter DEMON using the *magic mode*:

After the initial firmware response you would type:

boot <CR>

You continue to hit <CR> until the system prompt asks for "path name". You then type:

magic mode <CR>

The system will respond with "POOF". Moments later it will again ask for "path name" and you should type:

/etc/system <CR>

The system will then respond with a configuration summary and a load map. When the load map is finished you will again be prompted for the firmware password.

After entering the password you will be in the DEMON debug monitor and can set break points. Use **go** when you want the program to execute.

ENTERING DEMON

Firmware Mode

mcp < CR>

3B2 Computer Monitor/ Control Program -
erase '^H', kill '@'
Physical Mode
>

magic mode

boot < CR>

Enter name of program to execute

< CR>

Enter load device option number

< CR>

Path name:

magic mode < CR>

POOF!
Path name:

/ etc/ system < CR>

(Load map)
Firmware Mode

mcp < CR>

DEMON CONVENTIONS

All commands and data for the Debug Monitor Utilities Package are on four EPROM chips.

DEMON commands may contain any number of spaces or tabs as separators from the arguments. Multiple commands may be entered on the same line by using a semicolon (;) to separate them. Commands lines should not exceed 80 characters.

Repeat the previous command by typing a carriage return alone on a line.

Commands and hexadecimal numbers may be entered in either upper- or lowercase.

BACKSPACE	Also shown as ^H. This deletes one character each time it is depressed.
@	Deletes the entire line just typed. It returns the cursor and provides a line feed.
Control s	Temporarily stops the output. Allows you to read the output before it scrolls off the screen.
Control q	Restarts the output after a control s.
BREAK	This kills the current command. It is useful for stopping the output on commands that continue for long periods of time. Kill the command and stop the output after you have seen what you want.

DEMON CONVENTIONS

Spaces or tabs separate arguments

Semicolon (;) separates commands

< CR> - Repeat previous command

Upper- and lowercase are the same

BACKSPACE

@

Control s

Control q

BREAK

DEMON CONVENTIONS - ADDRESSES

Physical Mode

Addresses can be specified using physical memory. The first physical address that can be safely addressed is 2005000 hex. Memory below this point is used for DEMON and UNIX System boot.

Virtual Mode

Virtual address space is defined as follows :

ADDRESS	USE
0-3FFFFFFF	kernel gate tables, interrupt vectors and physical hardware vectors
40000000-7FFFFFFF	kernel text, data, & bss
80000000-BFFFFFFF	user text, data, & bss
C0000000-FFFFFFFF	user ublock, & stack

Modifiers

A modifier is a single letter which precedes the address entered. p -- Physical mode and v -- Virtual mode.

Qualifiers

Qualifiers make addressing of memory easier. This mechanism is a base and displacement scheme. A qualified number consists of qualifier and increment parts. The qualifier specifies the hex base for the reference while the increment provides an offset from that base. A qualified number is written as **qualifier.offset** where **qualifier** is the number of the qualifier (0 to f hex) and **offset** is any positive hex value. Only one qualifier per reference is allowed. A qualifier must be initialized before it can be referenced.

DEMON CONVENTIONS (ADDRESSES)

Physical Mode Modifier

p

Virtual Mode Modifier

v

Qualifiers

qualifier.offset

DEMON COMMANDS - MEMORY ACCESS

cp, copy	Copy a block of memory.
db	Display one byte of physical memory.
dd	Display two words (8 bytes) of physical memory.
dh	Display one half-word (2 bytes) of physical memory.
dw	Display one word (4 bytes) of physical memory.
dm	Display memory. This command displays a minimum of 16 bytes (4 words).
fm, fill	Fill memory with a particular data pattern.
sh	Set memory location to the specified data in half-word (2 bytes) increments.
sm	Set memory location to the specified data.
sw	Set memory location to the specified data a word at a time.

DEMON COMMANDS (MEMORY ACCESS)

cp, copy

db

dd

dh

dw

dm

fm, fill

sh

sm

sw

DEMON COMMANDS - REGISTER

old	Restore user registers.
q, qual	Display or set a qualifier addressing aid.
r	Display all user registers.
rn, sp, ap, fp, isp, pc, pcbp, psw	Display or set the specified register. **rn** is **r**[0-8].
slb, srama, sramb, sub	Display or set the specified pseudo-register.

DEMON COMMANDS (REGISTER)

old

q, qual
r

rn, sp,
ap, fp,
isp, pc,
pcbp,
psw

slb,
srama,
sramb,
sub

DEMON COMMANDS - CONSOLE and LINK

bd, baud Change the baud rate of the debugging console or print the baud rates of the debugging console.

bdh, baudh Change the baud rate of the data link or print the baud rates of the debugging console.

cisc Console is console.

lisc Link is console.

DEMON COMMANDS (CONSOLE and LINK)

bd, baud

bdh, baudh

cisc

lisc

DEMON COMMANDS - SYSTEM MODE

boot	
bugout	Exit DEMON.
flush	Flush cache.
ident	Identify or print a process id. The maximum number of processes that can be identified is 20.
info	Print addressing mode and possibly breakpoint number.
map	Translate the specified address to its physical equivalent.
p, phys	Set the addressing mode to physical.
reset	Invoke the system reset sequence.
switch	Process switch.
unident	Remove a process id from the ident table.
v, virt	Set the addressing mode to virtual.
?	Print the DEMON command list.

DEMON COMMANDS
(SYSTEM MODE)

boot

bugout

flush

ident

info

map

p, phys

reset

switch

unident

v, virt

?

DEMON COMMANDS - PROGRAM CONTROL

br	Display or set the software breakpoints. The maximum number of breakpoints that can be set is 20.
c, call	Call a procedure.
f, sf	Provide a nonreal time trace of the program execution skipping function calls.
g, go	Initiate program execution.
n, next	Set or display the next address range.
rm	Remove breakpoints.
st, strace	Provide a nonreal time silent trace of program execution.
t, trace	Provide a nonreal time trace of program execution.
td	Display or set up the trace buffer.

DEMON COMMANDS (PROGRAM CONTROL)

br

c, call

f, sf

g, go

n, next

rm

st, strace

t, trace

td

UNIT 3

DESIGN TOOLS

Lesson 3

Emulator

EMULATION

The DEMON Debug Monitor in the 3B2 Computer is very useful when debugging the driver software for a new feature card, but it does not give you much access to the software that is in the feature card itself. For this we need a development system that will allow the emulation of the microcontroller or microcomputer that is in the feature card that is under development.

Emulation is the controlled execution of the user's software in a hardware environment that duplicates the microcontroller of the user's prototype system. Emulation allows the interactive debugging of the user's programs. With it the user can externally control program execution while operating in the prototype system. Some of the things that the emulators will permit you to do are the reading and writing of registers and system memory, and program disassembly.

The Feature card that we are using in the lab exercises is based on the Intel 8751 microcontroller chip. To study and debug these cards we will be using the Intel Personal Development System (iPDS) equipped with the EMV-51A emulation vehicle. The software for the development system is supplied on one floppy disk, and the software for the EMV-51A is on another disk. For the lab exercises the two disks have been concatenated to form a single disk.

EMULATION

- Microcontroller
- Development System
- Emulator Vehicle
- Emulator Software

INSTALLATION PROCEDURES

When changing any of the connections between the 3B2 Computer and the development system the power should be turned off in both devices. Insert the EMV-51A controller box into the right side of the iPDS system, making sure that it seats firmly. Remove the microcontroller chip from the 3B2 feature card and insert the EMV-51A emulator into this socket. If necessary, 40-pin sockets can be placed between the emulator and the target system to clear other components on the card.

The EMV-51A emulator board contains jumpers that select the supply voltage source, the clock source, the reset source, and internal or external program memory. For our feature card the power for the chip is supplied by the development system, so that the 3B2 Computer - feature card power-on sequence can be traced. The reset and clock signals come from the feature card.

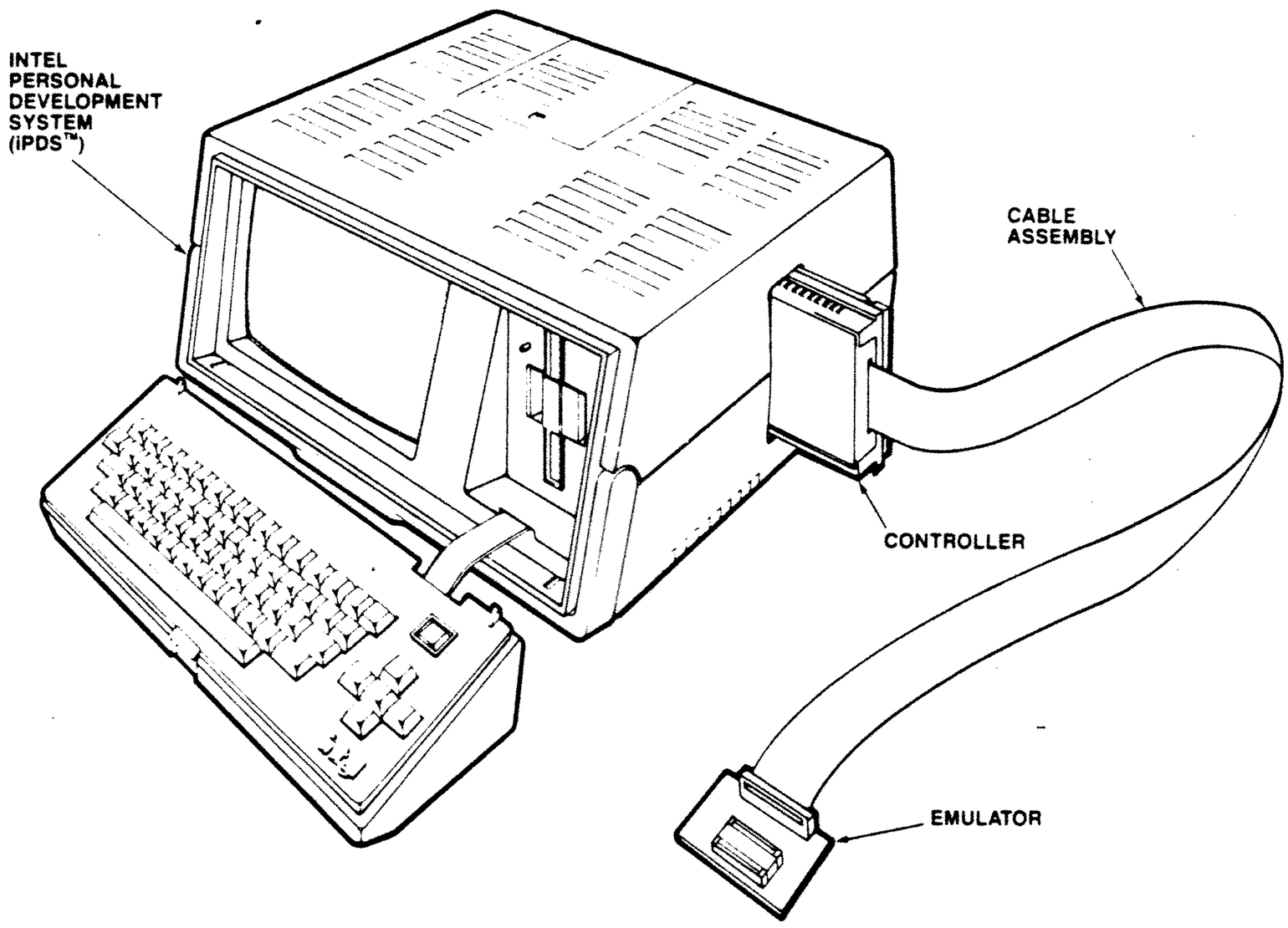
The 8751 microcontroller chip that is used on the feature card has 4K of on-board EPROM. When it is removed from its socket to connect the emulator, the program must be supplied by the development system. To allow this, the emulator is jumpered for what is called stand-alone operation.

Six terminal posts are located on the side of the EMV-51A controller. They give you access to the following test signals for use with an oscilloscope or a logic analyzer: instruction fetch clock, address latch enable, program store enable, emulation, external break input, and signal ground.

CAUTION

The emulator uses MOS devices which can be easily damaged by static electricity, even when the system is powered down. Follow all of the usual ESD precautions including the use of a grounding strap when handling any of the equipment.

INSTALLATION PROCEDURES



EMULATOR STARTUP

To start the emulation you should:

1. Apply power to the iPDS unit.
2. Insert the disk in drive 0.
3. Press RESET to initialize the iPDS system.
4. Key-in "EMV51A<cr>".
   An asterisk (*) prompt will indicate that the EMV-51A software is ready for a command line.
5. Key-in "LOAD HR1.OBJ<cr>".
   Any of the emulator commands can now be entered from the keyboard.

EMULATOR STARTUP

1. Apply Power
2. Insert Disk
3. Press RESET
4. "EMV51A<cr>"
5. "LOAD HR1.OBJ<cr>"

HELP COMMAND

The HELP command displays information about the emulator commands and keywords. It operates in two modes; **general help mode** where a list of available commands and keywords is provided and **specific mode** where format and requirements about specific commands and keywords are displayed.

To get the list of commands and keywords with the general help mode, just type in the word "HELP". For the specific mode, type in the word help followed by the command name or keyword. For example, if you enter "HELP LOAD" you will see the information about the load command.

HELP COMMAND

- General Mode
  eg: "HELP< cr> "

- Specific Mode
  eg: "HELP LOAD< cr> "

OTHER EMULATOR COMMANDS

NUMBERS:

Binary	1011Y
Octal	765Q
Decimal	1985T
Hex	F9H (default)

BASE - Display the current number base for outputs.

BAS = T - Set the output number base to decimal.

SUFFIX - Display the current number base for inputs.

SUF = Q - Set the input number base to octal.

R2 - Display the contents of register 2.

R5 = F3H - Change the contents of register 5 to F3H.

BR0 - Display the current value of breakpoint 0.

BR1 = 21H - Set breakpoint 1 at address 21H.

DBY 11H - Display the number stored in address 11H of RAM.

GO - Start executing the program at the address that is currently in the PC (program counter) register.

GO FROM 103H - Start executing the program at address 103H.

EXIT - Returns control to iPDS.

OTHER EMULATOR COMMANDS

BASE

BAS = T

SUFFIX

SUF = Q

R2

R5 = F3H

BR0

BR1 = 21H

DBY 11H

GO

GO FROM 103H

EXIT

UNIT 3

DESIGN TOOLS

Lesson 4

Lab Exercise #3

LAB EXERCISE #3

Equipment required:

3B2 Computer

"Ports" card installed in 3B2 Computer in slot #1.

HR1 custom feature card installed in 3B2 Computer in slot #2.

Console terminal

User terminal

LAB EXERCISE #3

This lab session will enable the student to exercise a feature card using the Diagnostic Monitor "Dgmon".

1. Get to firmware mode from multi-user mode by being logged on as "root" in the "root" directory and typing at the console:

   shutdown -i5 -y -g0

2. When in the firmware mode, invoke "DEMON" at the ">" prompt by typing "boot" and then "dgmon".

3. Run all of the "Normal" and "Demand" phases on the system board, Ports board and the HR1 board.

4. Run all of the "Interactive" phases on the above boards.

LAB EXERCISE #3 Answer Page.

5.

DEVICE	COMMAND	COMMAND	COMMAND
SBD			
PHASES	NORMAL	DEMAND	INTERACTIVE
TOTAL # PHASES	12	4	4
# PHASES PASSED			
# PHASES FAILED			
DEVICE	COMMAND	COMMAND	COMMAND
PORTS			
PHASES	NORMAL	DEMAND	INTERACTIVE
TOTAL # PHASES	3	15	4
# PHASES PASSED			
# PHASES FAILED			
DEVICE	COMMAND	COMMAND	COMMAND
HR1			
PHASES	NORMAL	DEMAND	INTERACTIVE
TOTAL # PHASES			
# PHASES PASSED			
# PHASES FAILED			

LAB EXERCISE #3

5. Fill in the table at the left.

UNIT 4

INTERRUPTS

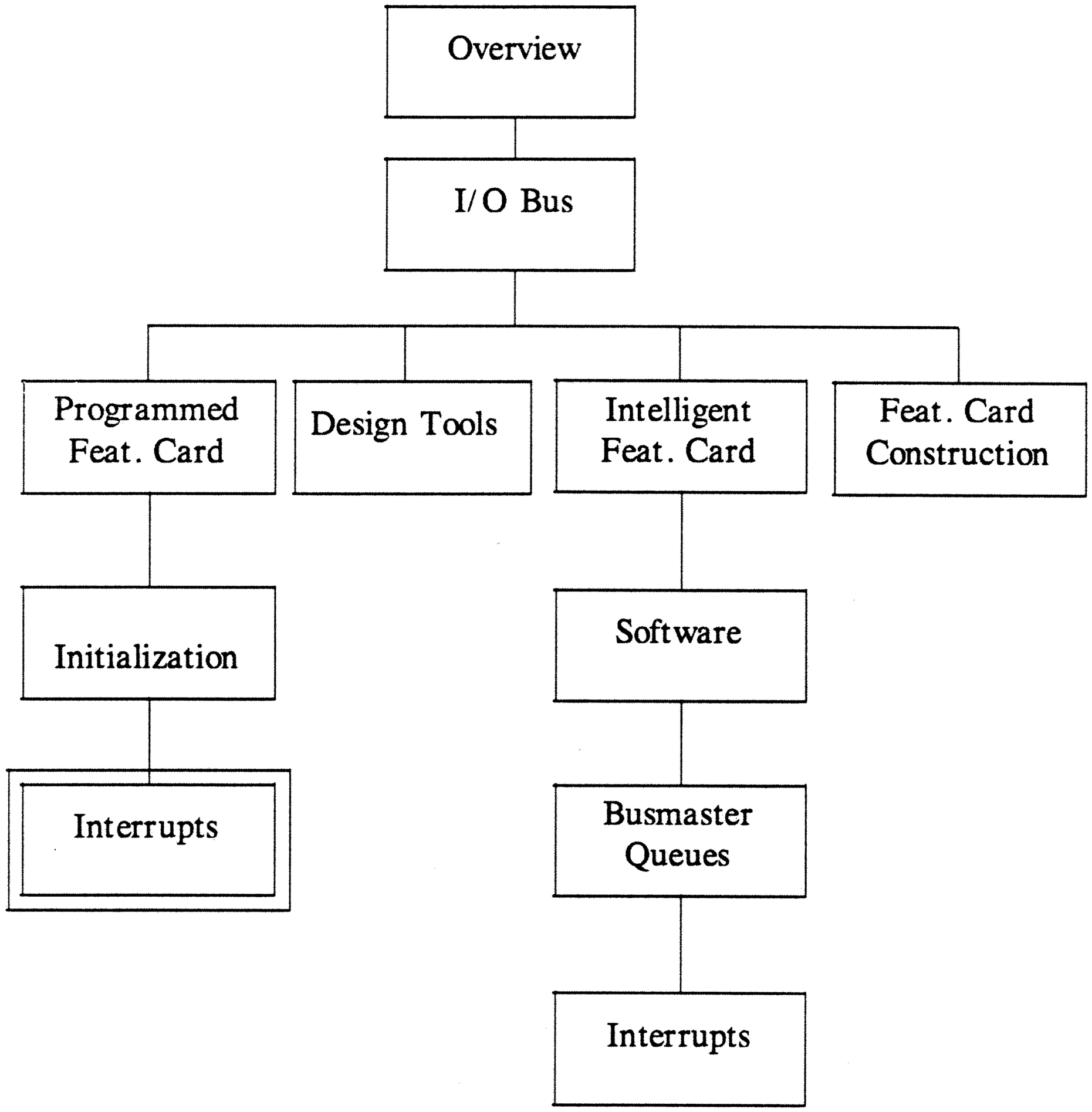
INTERRUPTS

Upon completion of this unit and given a 3B2, two terminals, HR1 Custom feature card, HR1 schematic, dual trace oscilloscope and student guide, the student will be able to:

1. Explain and apply system board interrupt priorities, sequence, timing, and daisy chained circuits.
2. Exercise a feature card using Intel iPDS Development System with the MCS51 Emulation Vehicle installed in the HR1 Processor Socket.
3. Use EMV51A commands to examine and interpret HR1 feature card ICs and operations.
4. Set EMV51A break points to "catch" a 3B2 byte write to the HR1 card offset, to display the content of the RAM location, to "catch" a byte written to the HR1 application port output and serial port.

The two lessons in this unit will cover the operation of the circuits that allow the programmed feature cards to interrupt the 3B2 Computer.

2302 Outline



UNIT 4

INTERRUPTS

Lesson 1

Interrupt Operation

SYSTEM BOARD INTERRUPT PRIORITIES

The system board supports 8 levels of interrupts, numbered 15 through 8, with level 15 having the highest priority. Five of these levels are used by the system board circuits, and the other three are used by the feature cards. They are called PINT[2]0, PINT[1]0, and PINT[0]0, with PINT[2]0 having the highest priority. A feature card can use one (and only one) of these lines to initiate an interrupt of the system board.

The priority of each I/O signal is further determined by the physical position of the feature card on the I/O expansion board. Since the interrupt acknowledge signals are daisy-chained through all of the feature card sockets, the card in the lowest socket has the highest priority.

NOTE: PINT[2]0 is reserved for future system use, and should **not** be used by any feature cards.

System Board Interrupt Levels

Level	Source
15	Bus Timeout, Parity Error, I/O Board Failure, Periodic Timer
14	PINT20 from I/O Bus
13	UARTs & DMA Complete
12	PINT10 from I/O Bus
11	Integral Disks
10	PINT00 from I/O Bus
09	PIR-9 (from CSR)
08	PIR-8 (from CSR)

SYSTEM BOARD

INTERRUPT PRIORITIES

FEATURE CARD SIGNAL	SYSTEM BOARD INTERRUPT LEVEL
PINT[2]0	14
PINT[1]0	12
PINT[0]0	10

INTERRUPT SEQUENCE

PINT[i]0 is the interrupt request lead that is used by the feature card to signal the system board.

Upon receipt of the interrupt signal, if there are no higher priority interrupt requests pending, the signals PIAK[i]0 and PR1W0 are asserted when the CPU reaches an instruction boundary.

PIAK[i]0 is the interrupt acknowledge lead that is used by the system board when it responds to the interrupt request.

PR1W0 is asserted by the system board (logic 1) to indicate a read operation. An unspecified address is placed on the Address Bus, but the feature card does not look at the address on an interrupt operation. The vector is then read from the ID/Vector register of the feature card. This vector is always an 8 bit value on lines 7 to 0 of the data bus. The address of the new Process Control Block Pointer (PCBP) is (Vector * 4) + 140. The PCBP is then used in the initiating of a process switch to the interrupt handling routine.

INTERRUPT SEQUENCE

1. PINT[i]0 Asserted by Feature Card

2. PIAK[i]0 Asserted by System Board

3. PR1W0 Asserted by System Board

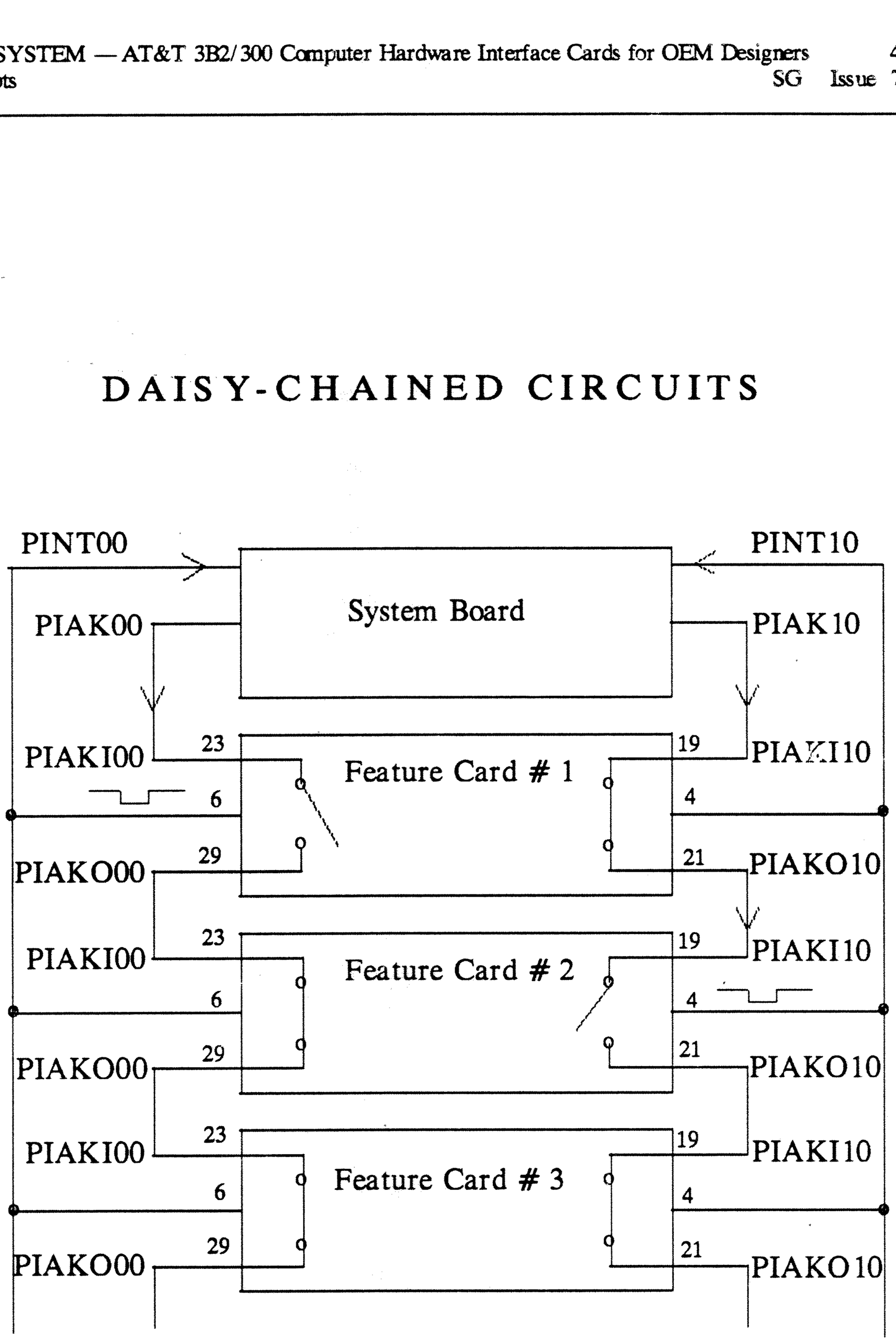
4. Vector from ID/Vector Register is Placed on the I/O Bus

5. Vector Used by System Board to Calculate the PCBP for the Interrupt Process

DAISY-CHAINED CIRCUITS

The 3B2 Computer has three interrupt request leads, PINT[2]0, PINT[1]0, and PINT[0]0, and three interrupt acknowledge leads, PIAK[2]0, PIAK[1]0, and PIAK[0]0. The interrupt acknowledge leads are daisy-chained through all of the feature cards. When interrupting the system board, a feature card will break this chain, thus giving the cards that are electrically closer to the system board a higher priority than the ones further away.

DAISY-CHAINED CIRCUITS



I/O BUS INTERRUPT TIMING

The protocol for the I/O bus requires the interrupt acknowledge signal PIAK[i]0 be contained within PINT[i]0. This allows a feature card to use PINT[i]0 to block daisy-chained PIAKO[i]0 from propagating to the lower priority feature cards. When the self-configuration is done after power-up, a feature card interrupt vector is passed to the feature card from the operating system. The vector is always a single byte of data on I/O bus byte 1 (bits 0 through 7).

A feature card begins an interrupt operation by asserting its particular interrupt signal PINT[i]0. The system board CPU will recognize the interrupt when the current program execution level and all pending interrupt requests are lower than the requesting level, and when execution reaches an instruction boundary. The system board will then request control of the I/O bus. (This request is transparent to the feature cards.)

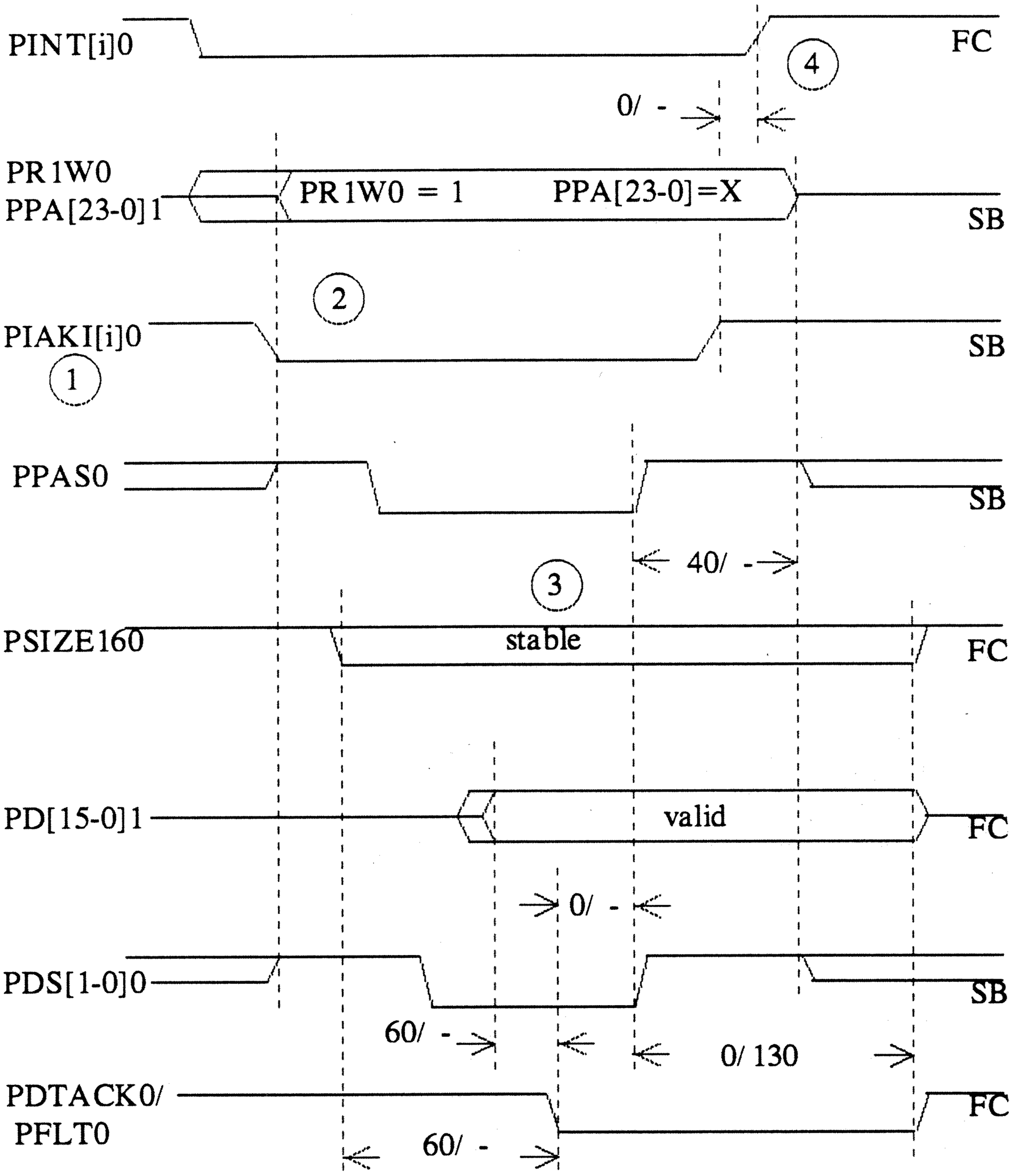
When granted the bus, it will generate the appropriate interrupt acknowledge signal PIAK[i]0. The system board will indicate a read operation (PR1W0 = 1) to read the vector from the feature card ID/Vector register. The presence of PIAKI[i]0 differentiates an interrupt acknowledge operation from a normal read of a feature card. Since the vector is always 8 bits wide, the feature card need not drive PSIZE160 during an interrupt acknowledge cycle, however, it must remain stable during the cycle. The system board's CPU uses the vector to finally identify which feature card requested the interrupt, and initiates the interrupt process.

When the feature card receives the interrupt acknowledge signal, it will place the contents of the 8 bit vector register on the data bus PD[7-0]1. Once the data is stable, PDTACK0 is sent to the system board. On receipt of PDTACK0 the system board will remove PIAK[i]0 allowing the feature card to remove PINT[i]0.

TIMING DIAGRAM NOTES:

1. PIAKO[i]0 of the acknowledged feature card remains blocked (high) for the duration of the cycle.
2. Due to the daisy-chain delay, PIAK[i]0 may go low much later in the cycle than is shown.
3. PSIZE160 may be 0 or 1 during the PIAK cycle, but it must be stable during the period indicated.
4. PINT[i]0 must be removed before the completion of the current interrupt service routine.

I/O BUS INTERRUPT TIMING



UNIT 4

INTERRUPTS

Lesson 2

Lab Exercise #4

LAB EXERCISE #4

Equipment required:

3B2 Computer

HR1 Custom Feature Card installed in 3B2 Computer

"Ports" Card installed in 3B2 Computer

HR1 Schematic Diagram

Console Terminal

User Terminal

Static Control Wrist Strap

Demon Manual

LAB EXERCISE #4

This lab session will enable the students to exercise a feature card using the DEMON - DEBUG MONITOR. Although DEMON is designed to aid in debugging the 3B2 Computer's (and its feature cards) hardware, software and firmware, in this course we will concentrate mainly on debugging feature card's hardware and firmware.

1. Review the HR1 feature card specifications in Appendix G and the DEMON commands covered in class. CAUTION - When using the "sm" command invoke it as "sm -n" (no verification after modify). Using "sm" by itself will give a false reading.

2. Turn on the 3B2 Computer with the HR1 feature card installed and login as "root".

3. With no other users on the machine run shutdown.

   shutdown -y -i5 -g0<CR>

   This **-i5** option will bring you to firmware mode.

4. Type the firmware password. Unless it is changed it will be **mcp**. Since these lab machines are equipped with DEMON EPROMs, the prompt that returns will confirm this and should be:

   3B2 Computer Monitor/ Control Program -
   erase '^H', kill '@'
   Physical Mode
   >

   Any other prompt to firmware password indicates that the machine is not DEMON equipped.

LAB EXERCISE #4 Answer Page.

5.

Message _____

6.

Message _____

7.

Message _____

8.

DEMON Commands Used _____

_____

9.

DEMON Commands Used _____

_____

10.

DEMON Commands Used _____

_____

11.

DEMON Commands Used _____

_____

LAB EXERCISE #4

5. Type the command "reset" and note the messages on the user terminal.

6. Read the contents of the register at offset address 01 (ID register) and note the user terminal message.

7. Write 20Hex to offset address 07 (vector register) and again note the message on the user terminal.

8. Write 11Hex to the HR1 application output port (offset address ff). Repeat this step, writing 22Hex, 44Hex, 88Hex, ffHex and finally 00Hex to the port.

9. Read Hex coded bytes from the application input port switches (address ff) that will "spell out" your name observing the following rules:

   A. Reading the application input port requires a "double read".

   B. The last command is repeated by DEMON firmware when "return" only is pressed.

   C. Use qualifiers whenever possible.

10. Write your name to the HR1 users terminal (address fe).

11. Type your name on the HR1 user terminal keyboard (address 05) and read it on the 3B2 console.

LAB EXERCISE #4 Answer Page.

12.

DEMON Commands Used _____

_____

14.

Message _____

Why? _____

16.

PORTS ID _____

PORTS ID _____

PORTS ID _____

18.

Messages _____

_____

_____

_____

_____

19.

PDTACK0 = _____

LAB EXERCISE #4

12. "Hex code" your last name and store it in the HR1 RAM starting at address BASE + 11Hex and read it back.

13. Carefully remove IC 17 on the HR1 feature card. (Wear the static control wrist strap!)

14. Repeat step 8 above and notice the console message if any. Why?

15. Install IC 17 back into HR1 and repeat steps 5,6,7,&8 above. Make sure that everything is operational.

16. Type the command "reset all". Read the ID code of the Ports feature card from offset 01. Read the ID code a second time. Read offset 03 and then try to read the ID code a third time. (The initialization sequence for this smart card will be explained on page 5.2.3.)

17. Type the command "?" and note the list of DEMON commands.

18. Boot the system from DEMON using "/etc/system". Observe the console messages. Explain the messages.

19. Write the Boolean expression for PDTACK0.

UNIT 5

INTELLIGENT (smart) FEATURE CARDS

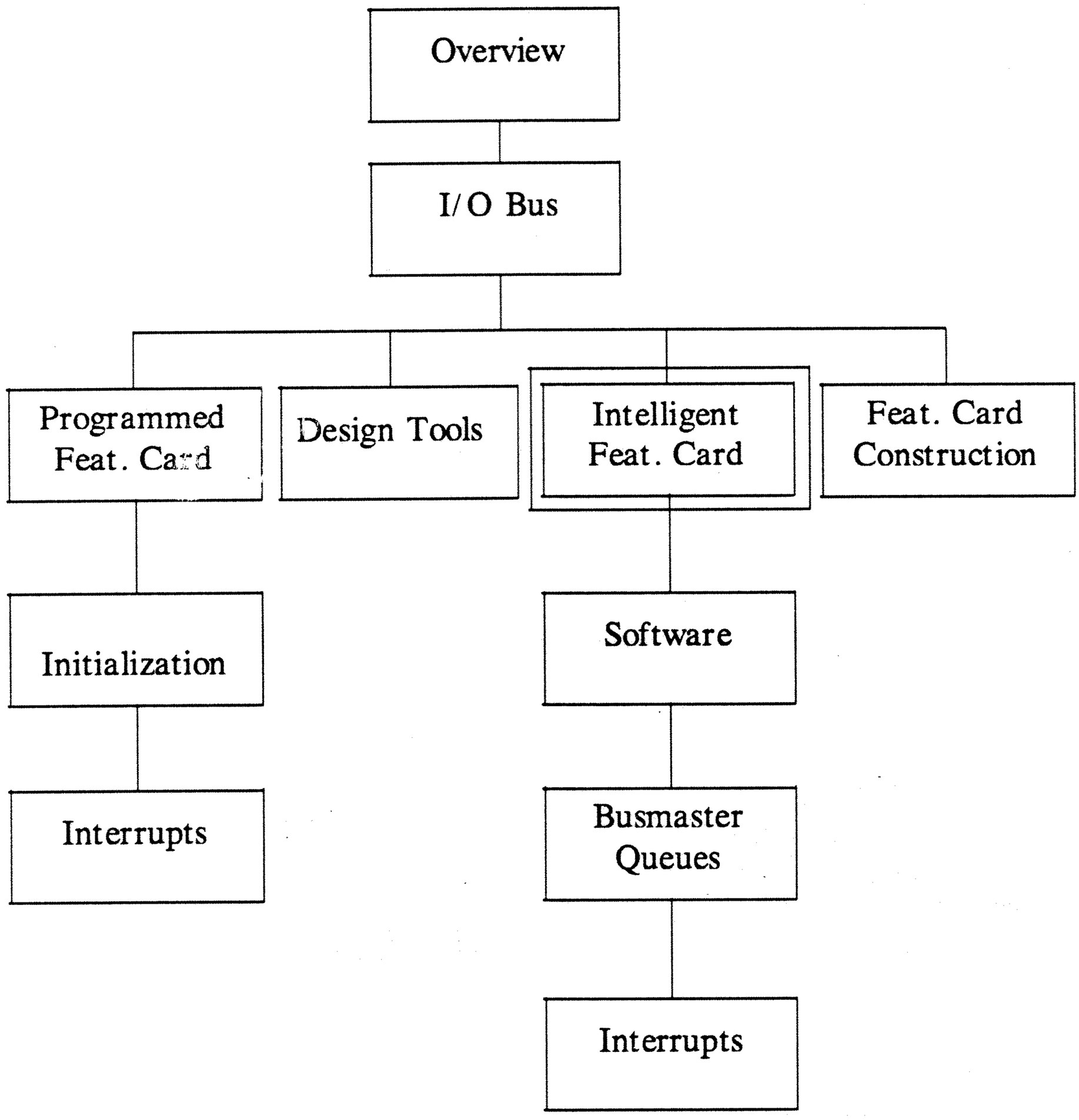
INTELLIGENT (smart) FEATURE CARDS

Upon completion of this unit and given a 3B2, two terminals, HR1 Custom feature card, HR1 schematic, dual trace oscilloscope and student guide, the student will be able to:

1. Explain and apply EDT table, and intelligent feature card initialization and sequence.
2. Describe feature card read and write to main memory, bus protocol, exchange sequence, logic, timing and queue structures.

The next lesson will describe some of the software files that are closely connected to the design and operation of the intelligent feature cards.

2302 Outline



UNIT 5

INTELLIGENT (smart) FEATURE CARDS

Lesson 1

Software Considerations

EQUIPPED DEVICE TABLE - (EDT)

The EDT is built in two stages. The first stage is completed in the self configuration. Here a identification code is read from the Identification/Vector register on each feature card.

To complete the EDT, FILLEDT downloads tables of devices and subdevices in /**dgn/edt_data** from the boot device's root file system. Each device code entry in the EDT is then matched to a code in the look-up table and corresponding look-up table data are copied to the EDT.

Any intelligent device may have between 0 and 15 subdevices associated with it. Each device marked as intelligent in the EDT is then SYSGENed and queried for its subdevice codes. The subdevice codes are then matched to entries in a subdevice look-up table; the subdevice names are copied to the EDT, completing the self configuration process.

If the **cons_found** flag is not set, FILLEDT will search for a terminal. This search begins with the system board and continues through the slots in increasing order until a console terminal is found or all equipped slots have been tested. Console INPUT/OUTPUT is directed to the default port if no console device responds. The FILLEDT routine completes the EDT.

EQUIPPED DEVICE TABLE (EDT)

Built in two stages

1. Self configuration

   Read ID code from the feature card

2. FILLEDT

   Downloads tables of devices and subdevices

EQUIPPED DEVICE TABLE - (EDT)

```
/* Copyright 1984 AT&T */

/* Structure for the Equipped Device Table */

1.    #define EDTSBD 0
2.    #define E_SINGLE 0
3.    #define E_DOUBLE 1
4.    #define E_8BIT 0
5.    #define E_16BIT 1
6.    #define E_DUMB 0
7.    #define E_SMART 1
8.    #define E_NAMLEN 10
9.    #define MAX_IO 12
10.   struct subdevice {
11.   unsigned short opt_code;
      /* sixteen bit option code for */
      /* subdevice stored as a short */
12.   unsigned char name[E_NAMLEN];
      /* ASCII name of subdevice */
13.   };
14.   struct edt {
15.   unsigned opt_code:16;
      /* sixteen bit option code */
16.   unsigned opt_slot:4;
      /* slot in which this board is ! */
17.   unsigned opt_num:4;
      /* option type for this board */
18.   unsigned rq_size:8;
      /* request queue entry size */
19.   unsigned cq_size:8;
      /* completion queue entry size */
20.   unsigned resrvd:14;
      /* reserved for future use */
21.   unsigned cons_cap:1;
      /* one means it can support console */
      /* zero means it cannot support console */
22.   unsigned cons_file:1;
      /* zero = device has no pump file */
      /* for floating console */
      /* one = device has a floating */
      /* console pump file */
23.   unsigned boot_dev:1;
      /* one for possible boot device */
24.   unsigned word_size:1;
      /* zero = 8 bit; one = 16 bit */
25.   unsigned brd_size:1;
      /* zero = single width board */
      /* one = double width board */
26.   unsigned smrt_brd:1;
      /* zero = dumb board */
      /* one = intelligent board */
27.   unsigned n_subdev:4;
      /* subdevice count */
28.   struct subdevice *subdev;
      /* pointer to array of n_subdev */
      /* subdevice structures */
29.   char dev_name[E_NAMLEN];
      /* ASCII name of device */
30.   char diag_file[E_NAMLEN];
      /* name of UNIX resident phase */
      /* containing diagnostic phases */
31.   }
```

EQUIPPED DEVICE TABLE (EDT)

```
/* Copyright 1984 AT& T */

/* Structure for the Equipped Device Table */

1.  #define EDTSBD 0
2.  #define E_SINGLE 0
3.  #define E_DOUBLE 1
4.  #define E_8BIT 0
5.  #define E_16BIT 1
6.  #define E_DUMB 0
7.  #define E_SMART 1
8.  #define E_NAMLEN 10
9.  #define MAX_IO 12
10. struct subdevice {
11. unsigned short opt_code;
12. unsigned char name[E_NAMLEN];
13. };
```

EQUIPPED DEVICE TABLE - (EDT)

```
/* Copyright 1984 AT&T */

/* Structure for the Equipped Device Table */

1.      #define EDTSBD 0
2.      #define E_SINGLE 0
3.      #define E_DOUBLE 1
4.      #define E_8BIT 0
5.      #define E_16BIT 1
6.      #define E_DUMB 0
7.      #define E_SMART 1
8.      #define E_NAMLEN 10
9.      #define MAX_IO 12
10.     struct subdevice {
11.     unsigned short opt_code;
        /* sixteen bit option code for */
        /* subdevice stored as a short */
12.     unsigned char name[E_NAMLEN];
        /* ASCII name of subdevice */
13.     };
14.     struct edt {
15.     unsigned opt_code:16;
        /* sixteen bit option code */
16.     unsigned opt_slot:4;
        /* slot in which this board is ! */
17.     unsigned opt_num:4;
        /* option type for this board */
18.     unsigned rq_size:8;
        /* request queue entry size */
19.     unsigned cq_size:8;
        /* completion queue entry size */
20.     unsigned resrvd:14;
        /* reserved for future use */
21.     unsigned cons_cap:1;
        /* one means it can support console */
        /* zero means it cannot support console */
22.     unsigned cons_file:1;
        /* zero = device has no pump file */
        /* for floating console */
        /* one = device has a floating */
        /* console pump file */
23.     unsigned boot_dev:1;
        /* one for possible boot device */
24.     unsigned word_size:1;
        /* zero = 8 bit; one = 16 bit */
25.     unsigned brd_size:1;
        /* zero = single width board */
        /* one = double width board */
26.     unsigned smrt_brd:1;
        /* zero = dumb board */
        /* one = intelligent board */
27.     unsigned n_subdev:4;
        /* subdevice count */
28.     struct subdevice *subdev;
        /* pointer to array of n_subdev */
        /* subdevice structures */
29.     char dev_name[E_NAMLEN];
        /* ASCII name of device */
30.     char diag_file[E_NAMLEN];
        /* name of UNIX resident phase */
        /* containing diagnostic phases */
31.     }
```

EQUIPPED DEVICE TABLE (EDT)

```
14. struct edt {
15. unsigned opt_code:16;
16. unsigned opt_slot:4;
17. unsigned opt_num:4;
18. unsigned rq_size:8;
19. unsigned cq_size:8;
20. unsigned resrvd:14;
21. unsigned cons_cap:1;
22. unsigned cons_file:1;
23. unsigned boot_dev:1;
24. unsigned word_size:1;
25. unsigned brd_size:1;
26. unsigned smrt_brd:1;
27. unsigned n_subdev:4;
28. struct subdevice *subdev;
29. char dev_name[E_NAMLEN];
30. char diag_file[E_NAMLEN];
31. }
```

/etc/master.d/ports

This is the contents of the /etc/master.d/ports file.

/etc/master.d/ports

```
*       <@(#)ports 12>
*
* PORTS
*

*FLAG    #VEC    PREFIX    SOFT    #DEV    IPL    DEPENDENCIES/VARIABLES
  act      1        pp       -       5      10
                                                  pp_tty[#C*#D] (%0x58)
                                                  ppcbid[#C*#D] (%s)
                                                  ppcpid[#C*#D] (%s)
                                                  pp_cnt(%i) ={#C*#D}
                                                  pp_board[#C] (%0x6fc)
                                                  savee[#C*SAVEXP] (%0xc)
                                                  maxsavexp(%i)={#C*SAVEXP}
                                                  csbit[#C] (%i)

$$$

SAVEXP =      5
```

/etc/master.d/hr1

This is the contents of the /etc/master.d/hr1 file.

/etc/master.d/hr1

```
* MEM
*
*FLAG   #VEC   PREFIX   SOFT   #DEV   IPL   DEPENDENCIES/VARIABLES
ac      1      hr1      -      1      10
```

UNIT 5

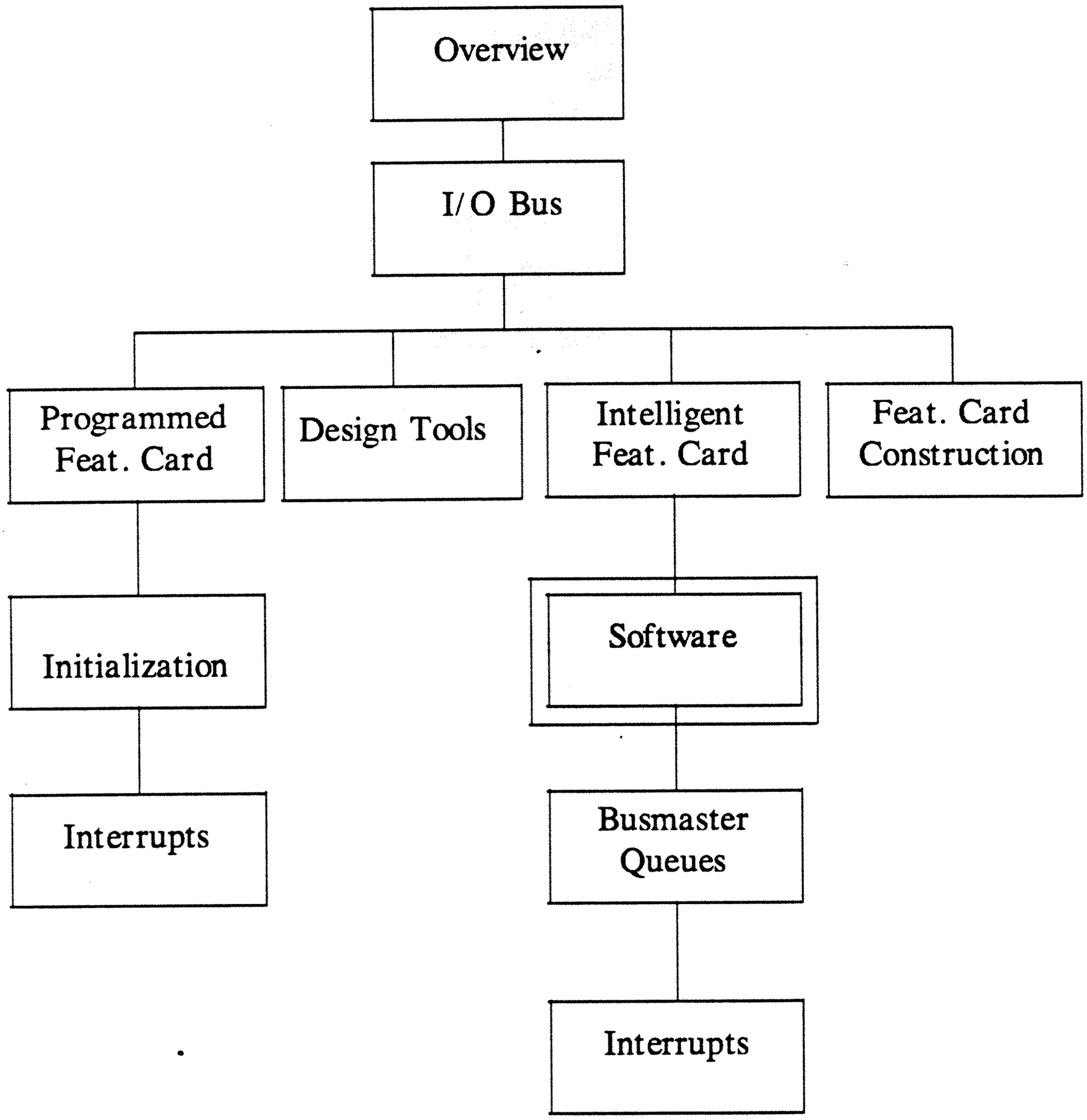
INTELLIGENT (smart) FEATURE CARDS

Lesson 2

Software Initialization and Pump

This lesson and the lab exercise that follows will deal with how the software initializes the intelligent feature cards.

2302 Outline

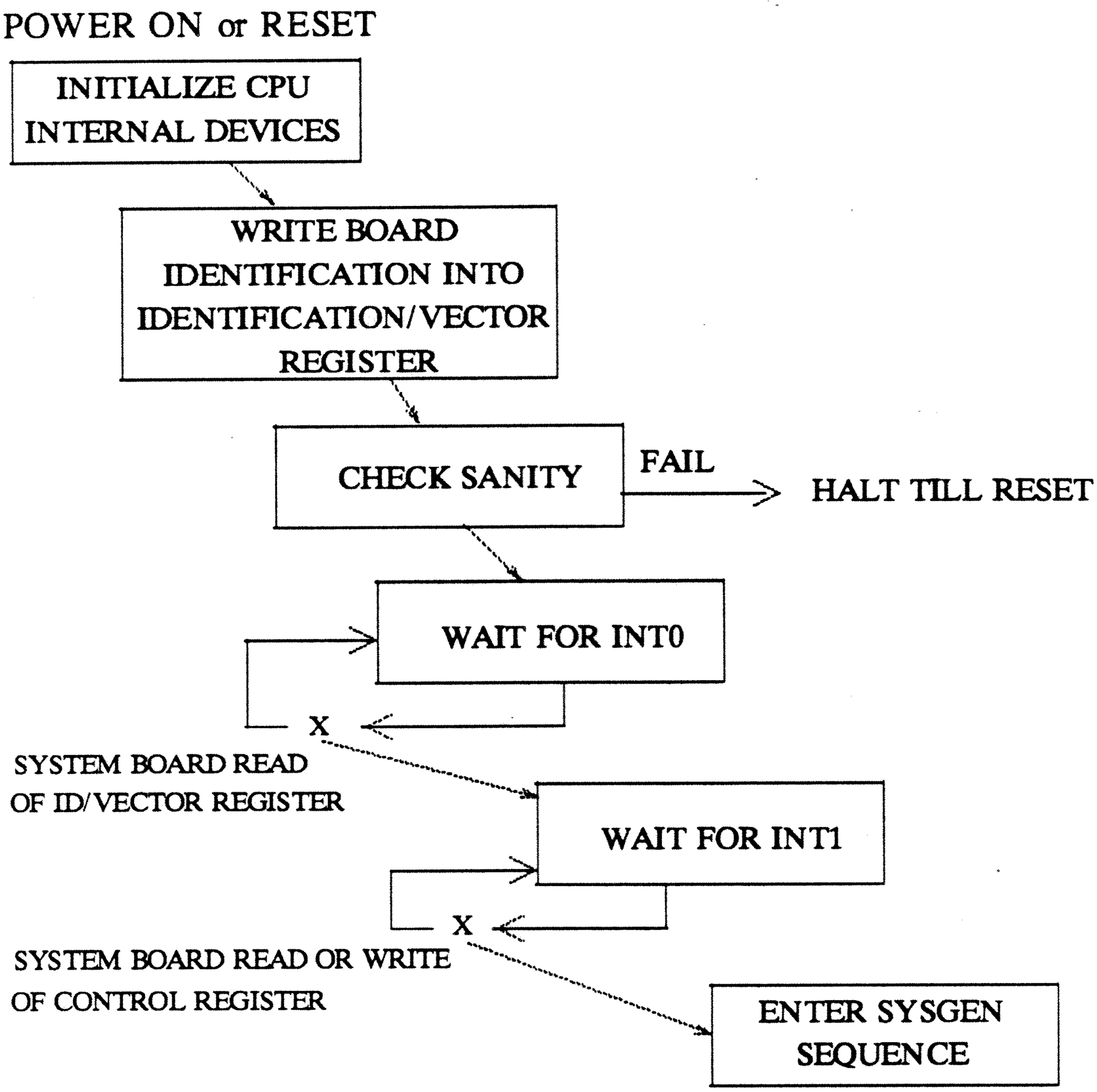


INTELLIGENT FEATURE CARD INITIALIZATION SEQUENCE

On power-up the I/O bus signal SYSRST0 resets all of the feature cards. Each intelligent feature card then starts its initialization routine, and must load the identification code into its ID/Vector register within 100 milliseconds.

The system board will then read this code and store it in the Equipment Device Table in DPDRAM. The system board then knows what type of card this is and where it is located on the I/O bus. The reading of this code produces an INT0 interrupt in the feature card indicating that the feature card should now be waiting for the next signal, an attention interrupt INT1. INT1 signals the feature card to start its SysGen sequence.

INTELLIGENT FEATURE CARD INITIALIZATION SEQUENCE



PUMP COMMAND

The pump command is located in **/etc/pump**. The full description of the command can be found in the "System Administration Utilities Guide" (select code 305-422).

PUMP COMMAND

NAME

pump - Download B16 or X86 a.out file to a Feature Card

SYNOPSIS

pump / dev/ devname file

DESCRIPTION

The *pump* command will read a B16 or X86 **a.out** file's sections into a buffer according to the physical address of the section. Pump expects a section in the **a.out** file called ".start". Once it has found this section. pump will inform the peripheral to start executing at the address that it found in ".start" after it has downloaded the **a.out** file.

UNIT 5

INTELLIGENT (smart) FEATURE CARDS

Lesson 3

Lab Exercise #5

LAB EXERCISE #5

Equipment required:

3B2 Computer

iPDS-100 Intel Development System.

HR1 Custom Feature Card (installed in 3B2 Computer) with EMV51 Emulation Vehicle Installed.

HR1 Schematic Diagram

Console Terminal

User Terminal

LAB EXERCISE #5

This lab session will enable the students to exercise a feature card using the Intel iPDS Development System with the MCS51 Emulation Vehicle installed in the HR1 processor socket. When you are in the EMV51A command mode (at the "*" prompt), you will use the following commands:

A. Command "BR[numb]=[Hex address]" will set break points at the at specified addresses. For example, "BR0=21" will set break point number 0 at the address 21Hex. Caution - ALL THE BREAK POINTS MUST BE SET AT THE BEGINNING OF THE INSTRUCTION.

B. Typing "R" in the emulator command mode will display content all the registers.

C. Command "DBY [Hex Address]" will display content of the onboard RAM. For example, "DBY 11" will display the contents of address 11Hex in RAM.

D. Command "DBY 11 TO 17" will display the contents of memory starting at address 11 and finishing at 17.

1. Examine HR1 schematic and verify that the following is true:

   A. When the feature card is selected, the offset address is latched by octal latch IC31.

   B. If the offset address (from 3B2 via internal address bus LB2A001-LB2A071) is > 7 (/A0-3 logic Hi) the data from 3B2 (via internal data bus 3BPD001-3BPD071) is latched in TEST READ IC5 or TEST WRITE IC3, depending on status of PR1W0. (Data will be read from the feature card by the system board or data will be written by the system board to the feature card.)

LAB EXERCISE #5

This page is for notes.

LAB EXERCISE #5

C. The 8751 microprocessor can retrieve the address from the IC31 octal latch via port P0 after enabling output of the octal latch (IC31 pin 1 /OE). The output of the this octal latch (as well as the others) is enabled indirectly by the 8751 microprocessor - port P3 - local bus LB1A01-LB1A31.

2. Examine the HR1 firmware listing and verify that the following is true:

   A. Upon the feature card select signal PCS0 true (low), 8751 is being interrupted (/INT0). The interrupt 0 service routine - E0_INT - starts at line 59 of the HR1 firmware listing.

   B. One of the first tasks performed during the interrupt is address load subroutine - line 68 - CALL ADLD. This subroutine will return the address asserted by 3B2 system board in register R0.

   C. The next task performed by the above routine is a decision if read or write occurred, line 69.

   D. Memory write begins at line 85. At this time the A register contains the data and register R0 contains the address.

3. Shutdown the 3B2 Computer with the HR1 feature card installed in it. Remove the 8751 microprocessor and install the EMV51 Emulator header in place of it.

4. Install the floppy diskette marked HR1 Firm in the IPDS floppy drive. Turn on the IPDS (power switch in the back). IPDS will run the diagnostics and will then display a prompt. After the prompt is displayed type "EMV51A". This will load the MCS51 emulation vehicle software. Wait for the EMV51A prompt "*". At this point type "LOAD HR1.OBJ" and then type "GO".

LAB EXERCISE #5

This page is for notes.

LAB EXERCISE #5

5. Turn on the 3B2 Computer with the HR1 feature card installed and with the EMV51A Emulation Vehicle installed in it in turn. Notice that the user terminal plugged into the HR1 feature card serial port displays the messages: "ID CODE WRITTEN TO ID REGISTER" and "CARD HAS BEEN IDENTIFIED!" as the system progresses through initialization.

6. Login as root and execute "shutdown -i5 -g0 -y". When in DEMON, set the appropriate qualifiers to access the HR1 feature card.

7. "Hex code" your name and write it to memory beginning at location 11Hex. Remember to use "the odd boundary rule" and use the "sm -n [addr] [Hex byte]" format.

8. Hit "ESC" key on IPDS keyboard. This breaks the emulation. Set break points in appropriate places to do the following.

   A. Catch the address and data when 3B2 writes byte to feature card offset 11Hex. Display the appropriate registers.

   B. A data byte has just been written to the RAM location. Display the content of the RAM location to verify that has indeed happen.

   C. Repeat the same procedures from steps A and B above to trace the writing of the system board to the feature card the words "CUSTOM 3B2" (start at address 11Hex in RAM).

   D. "Catch" a byte being written to the APPLICATION PORT OUTPUT.

   E. "Catch" a byte being written to the HR1 SERIAL PORT.

[illegible]

[illegible]
OBJECT MODULE PLACED IN :F0:HR.OBJ
[illegible]

LOC OBJ LINE SOURCE

```
                    1       ;
                    2       ;*              COPYRIGHT 1985 AT&T
                    3       ;*
                    4       ;*      AUTHOR:
                    5       ;*      DATE: 6/07/85
                    6       ;*      LOCATION: AT&T HICKORY RIDGE TRAINING CENTER
                    7       ;*                AT&T TECHNOLOGIES
                    8       ;*                ROOM K101
                    9       ;*                1195 SUMMERHILL DRIVE
                   10       ;*                LISLE, IL. 60532
                   11       ;*
                   12       ;
                   13       ;
0020                14       LET_INDEX       EQU     32      ;LETTER INDEX
                   15       ;
0090               16       BUS     EQU     090H    ;BUS RESIDES ON PORT P1
00B2               17       PCS0    EQU     0B2H    ;PERIPHERIAL CARD SELECT - SENSE P3.3
00FF               18       APPA    EQU     0FFH    ;APPLICATION PORT ADDRESS
00FE               19       APSER   EQU     0FEH    ;WRITE SERIAL ADDRESS
                   20       ;
00B4               21       STAT1   EQU     0B4H    ;STAT DIGIT 1 CONTROLLED BY P3.4
00B5               22       STAT2   EQU     0B5H    ; "     "     2       "         P3.5
00B6               23       STAT4   EQU     0B6H    ; "     "     4       "         P3.6
00B7               24       STAT8   EQU     0B7H    ; "     "     8       "         P3.7
                   25       ;
                   26       ;
                   27       ;
                   28       ;
                   29       ;
                   30       ;
                   31       ;FOR READABLE REGISTER TEST SET THE PPAS0,PDS10,PCS0.
                   32       ;PRIW0 SWITCH UP.
                   33       ;THE READABLE REGISTERS ARE:
                   34       ;REGISTER NAME          ADDRESS
                   35       ;ID/VECT                01
                   36       ;STATUS                 05
                   37       ;VECTOR                 07
                   38       ;TEST READ              09      ;ANY DATA IN ADD.  RANGE 08H - 0FFH.
                   39       ;
                   40       ;THE WRITABLE REGISTERS ARE:
                   41       ;REGISTER NAME          ADDRESS
                   42       ;CONTROL                03
                   43       ;VECTOR                 07
                   44       ;TEST WRITE             09      ;ANY DATA IN ADDRESS RANGE 08H - 0FFH
                   45       ;
                   46       ;THE WRITABLE REGISTERS CAN BE TESTED BY SETTING PPAS0, PDS10,PCS0 DOWN.
                   47       ;PRIW0 SWITCH DOWN.
                   48       ;SET DATA TO BE WRITTEN TO THE WRITABLE REGISTERS ON DATA SWITCHES.
                   49       ;
                   50       ;
```

```
LOC  OBJ                  LINE     SOURCE

                            51     ;       ORG     0000H   ;JUMP AROUND  INTERRUPT
0000 020100                 52             JMP     INIT
                            53     ;
0003                        54             ORG     0003H           ;EXTRNAL INT0 PROCESS ROUTINE
0003 C2B4                   55             CLR     0B4H            ;CLEAR INTERRUPT FLIP FLOP
0005 D2B4                   56             SETB    0B4H            ;TO PREVENT ANOTHER INTERRUPT
0007 02000A                 57             JMP     E0_INT          ;PROCESS ROUTINE
                            58
                            59     ;THIS INTERRUPT PROCESS ROUTINE WILL TAKE PLACE EVERY TIME
                            60     ;/PCS0 * /PDS10 * PDS00 = PDACK0
                            61     ;
                            62     E0_INT:
000A C0F0                   63             PUSH    B
000C C0D0                   64             PUSH    PSW
000E C0E0                   65             PUSH    ACC
                            66     ;
                            67     ;
0010 120192                 68             CALL    ADLD            ;CALL ADDR. LOAD ROUT. - ADDR. IN REG. R0, BANK
0013 20B51D                 69             JB      P3.5,MEMRD      ;CHECK READ/WRITE FLIP FLOP
0016 120182                 70             CALL    TSRWL           ;DATA WAS CAPTURED BY -TEST REGISTER- INTO -A-
                            71     ;
                            72     ;TEST IF THE ADDRESS IS -APPLICATION SPECIAL-
                            73     ;
0019 FC                     74             MOV     R4,A            ;SAVE ACQUIRED DATA
001A E8                     75             MOV     A,R0            ;ADDRESS NOW IN -A-
001B B4FF04                 76             CJNE    A,#APPA,WRSER   ;CHECK IF -WRITE APPL. SPECIAL-
001E EC                     77             MOV     A,R4            ;MOVE ADDRESS DATA BACK TO -A-
001F 12019A                 78             CALL    AOL             ;LOAD APPICATION PORT
                            79     ;
0022 B4FE09                 80     WRSER:  CJNE    A,#APSER,NSPWR  ;CHECK IF -WRITE SERIAL SPECIAL-
0025 120182                 81             CALL    TSRWL           ;GET THE CHAR
0028 120338                 82             CALL    C_OUT           ;DISPLY IT
002B 020081                 83             JMP     E0_EXT          ;------> EXIT INTERRUPT ROUTINE
                            84     ;
                            85     ;---------NORMAL MEMORY WRITE - NO SPECIALS ----------
002E EC                     86     NSPWR:  MOV     A,R4            ;MOVE DATA BACK TO ACC.
002F F6                     87             MOV     @R0,A           ;MOVE DATA TO ADDR. POINTED BY R0
0030 020045                 88             JMP     CONT1           ;MEM WRITE COMPLETED
                            89     ;
                            90     ;
                            91     MEMRD:
0033 FC                     92             MOV     R4,A            ;SAVE ACQUIRED DATA
0034 E8                     93             MOV     A,R0            ;ADDRESS NOW IN -A-
0035 B4FF09                 94             CJNE    A,#APPA,NSRD    ;CHECK IF -READ APPL. SPECIAL-
0038 1201A2                 95             CALL    AIL             ;CALL APPLICATION REGISTER INPUT ROUT.
003B 12017A                 96             CALL    TSRRL           ;XFER THE ACQ. DATA TO TEST REGISTER TO BE READ
003E 020045                 97             JMP     CONT1           ;BY 3B2.
                            98     ;
                            99     ;--------- NORMAL MEMORY READ - NO SPECIALS ----------
0041 E6                    100     NSRD:   MOV     A,@R0           ;MOVE DATA FROM ADDR. POINTED BY R0 TO ACC
0042 12017A                101             CALL    TSRRL           ;PLACE THE DATA IN -TEST REGISTER-
                           102     ;
                           103     ;
0045 020081                104     CONT1:  JMP     E0_EXT          ;(======== REMOVE THIS STATMENT FOR TRACE
                           105     ;
```

```
LOC  OBJ             LINE     SOURCE

                      106
0048 12035B           107              CALL    PMES_2  ;REGISTERS CONTAIN THE FOLLOWING DATA
004B 120361           108              CALL    PMES_3  ;MES. ADD. REG  DATA
                      109      ;
004E 7433             110              MOV     A,#33H  ;ADD. FOR REG.
0050 120338           111              CALL    C_OUT   ;DISPLAY IT
0053 120367           112              CALL    PMES_4  ;TAB OVER
0056 120172           113              CALL    CNRWL   ;EXECUTE CONT REG. WRITE LOAD
0059 1201AE           114              CALL    DEC_UN  ;DISPLAY THE ACQUIRED CHARACTER ON TERM
005C 12036D           115              CALL    PMES_5  ;CR, LF
                      116      ;
005F 7437             117              MOV     A,#37H  ;ADD. FOR REG.
0061 120338           118              CALL    C_OUT   ;DISPLAY IT
0064 120367           119              CALL    PMES_4  ;TAB OVER
0067 12016A           120              CALL    VRWL    ;EXECUTE VECTOR REG.  WRITE LOAD
006A 1201AE           121              CALL    DEC_UN  ;DISPLAY THE ACQUIRED CHARACTER ON TERM
006D 12036D           122              CALL    PMES_5  ;CR, LF
                      123      ;
0070 7454             124              MOV     A,#54H  ;ADD. FOR REG.
0072 120338           125              CALL    C_OUT   ;DISPLAY IT
0075 120367           126              CALL    PMES_4  ;TAB OVER
0078 120182           127              CALL    TSRWL   ;EXECUTE TEST REG. WRITE LOAD
007B 1201AE           128              CALL    DEC_UN  ;DISPLAY THE ACQUIRED CHARACTER ON TERM
007E 12036D           129              CALL    PMES_5  ;CR, LF
                      130      ;
                      131      ;
                      132      E0_EXT:
0081 D0E0             133              POP     ACC
0083 D0D0             134              POP     PSW
0085 D0F0             135              POP     B
0087 32               136              RETI
                      137      ;
                      138      ;
0100                  139              ORG     100H
                      140      INIT:
0100 758170           141              MOV     SP,#70H         ;STACK POINTER STARTS AT LOC 100
                      142      ;
0103 C28E             143              CLR     TR1
0105 758DEE           144              MOV     TH1,#0EEH       ;8MHZ 1200B
0108 758920           145              MOV     TMOD,#20H
010B 759850           146              MOV     SCON,#50H       ;8BIT UART,RECEIVE ENABLE
010E D28E             147              SETB    TR1
0110 D299             148              SETB    TI
                      149      ;
0112 D2B4             150              SETB    0B4H            ;MAKE /INT0 HIGH
0114 C2B4             151              CLR     0B4H            ;-------|_______|-------
0116 D2B4             152              SETB    0B4H
                      153      ;
0118 12018A           154              CALL    IDSRL           ;LOAD ID SWITCH SETTING INTO ITS REGISTER
011B 120152           155              CALL    IDVRL           ;BUS REGISTER NOW IN ID /VECTOR REG.
011E 12034F           156              CALL    PMES_0
                      157      ;
                      158      ;
0121 20B2FD           159      PCSWT:  JB      PCS0,PCSWT      ;WAIT UNTIL CARD /CARD SELECT TRUE
                      160      ;
```

UNIT 5

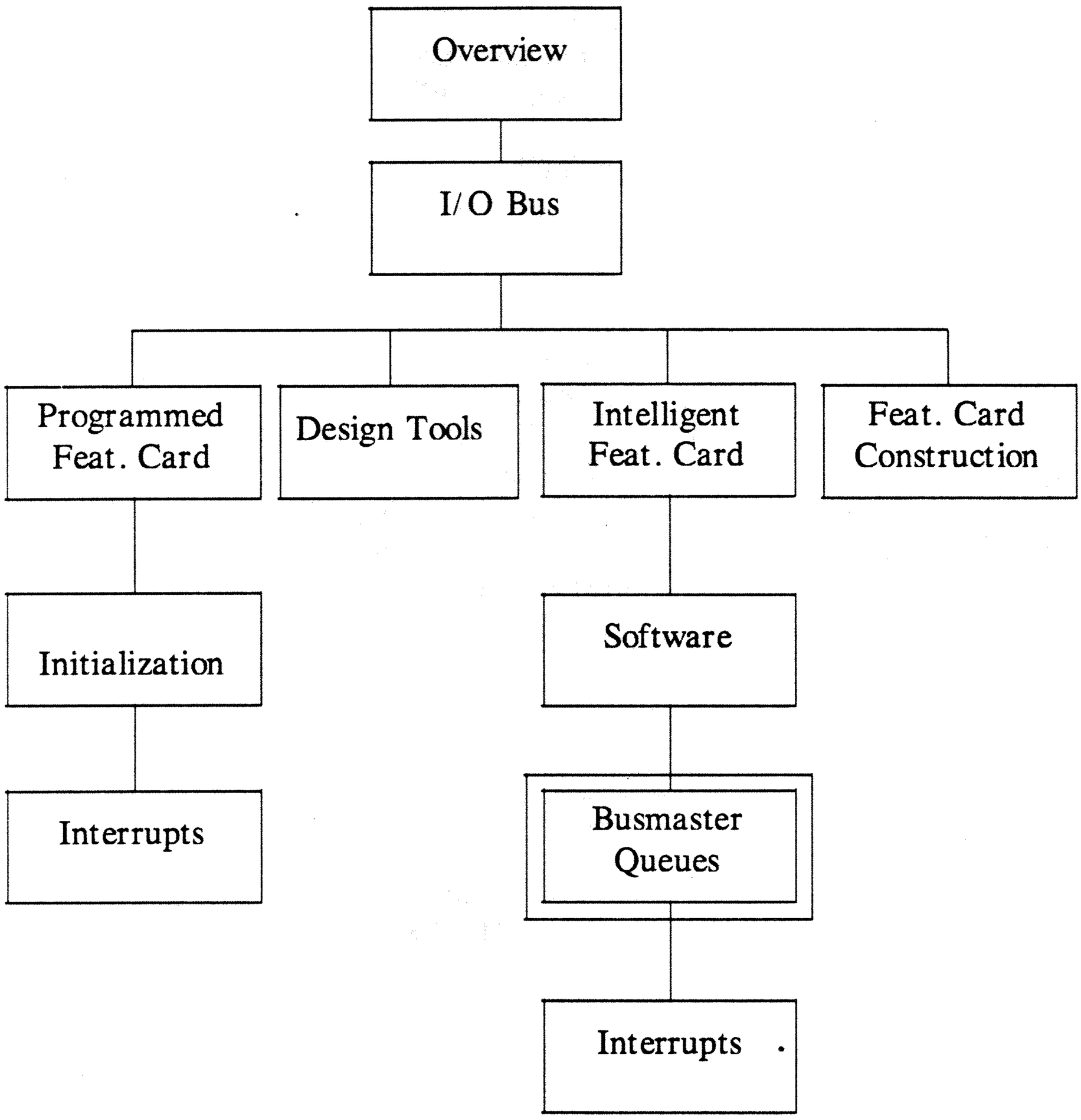
INTELLIGENT (smart) FEATURE CARDS

Lesson 4

Busmaster

The next two lessons will discuss what has to be done for the intelligent feature card to take control of the I/O Bus, and how queues are used to communicate with the system board.

2302 Outline



FEATURE CARD READ OF MAIN MEMORY

A Feature Card Read of Main Memory begins with the feature card requesting control of the I/O bus via PBRQ0. On receipt of bus acknowledge (PBACK0), the feature card (bus master) asserts the bus busy signal (PBUSY0) causing the system board to remove PBACK0.

The feature card starts the data transfer by sending PPA[23-00]1, PR1W0 (HIGH for read), and PSIZE160 to the system board. Once these are stable, PPAS0 and PDS[1-0]0 follow. PBRQ0 must be removed after PPAS0 has been asserted and PBACK0 has been removed.

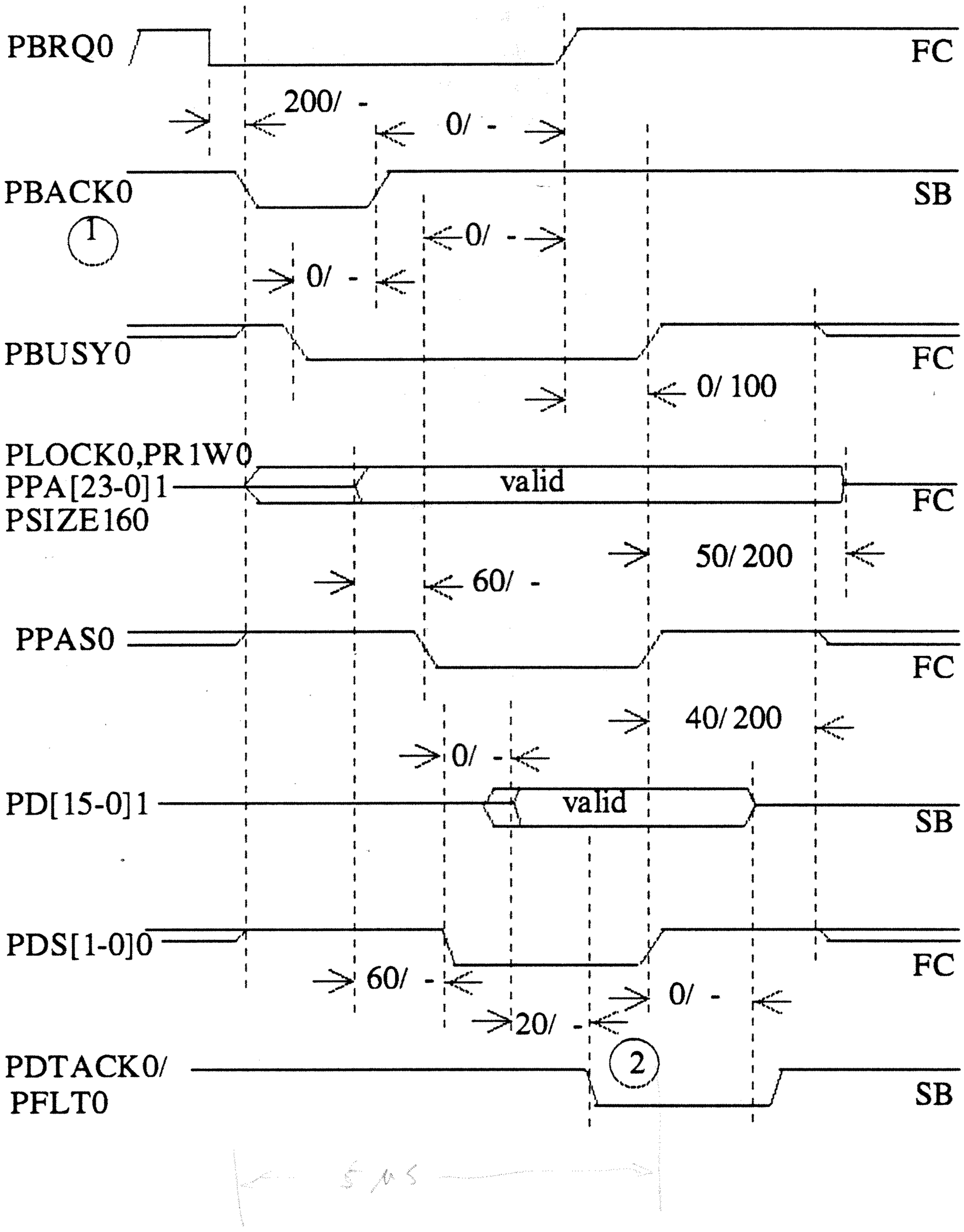
The data strobes cause the main memory controller (slave) to gate the data onto the I/O bus. When the data is stable, the slave asserts data transfer acknowledge PDTACK0 allowing the feature card to latch the data. Following this, the feature card will drive high the address strobe, data strobe, and bus busy signal before tri-stating PLOCK0, PR1W0, and PPA[23-00]1. When the data strobes are removed the main memory controller will remove the data and PDTACK0.

All times shown are in nanoseconds as seen by the feature card.

TIMING DIAGRAM NOTES:

1. PBACKO0 of acknowledged feature card shall remain blocked (high).
2. Data is guaranteed valid for a minimum of 20 nanoseconds at the feature card connector before PDTACK0.

FEATURE CARD READ OF MAIN MEMORY



FEATURE CARD WRITE TO MAIN MEMORY

The bus request/grant procedure for write is identical to that for the preceding read protocol. Following the receipt of PBACK0 the feature card again starts the data transfer by sending PA[23-00]1, PR1W0, and PSIZE160 to the system board. After the required set-up time the feature card will send the address strobe PPAS0. Also, after PBACK0 has been received, the feature card will gate the write data onto the bus.

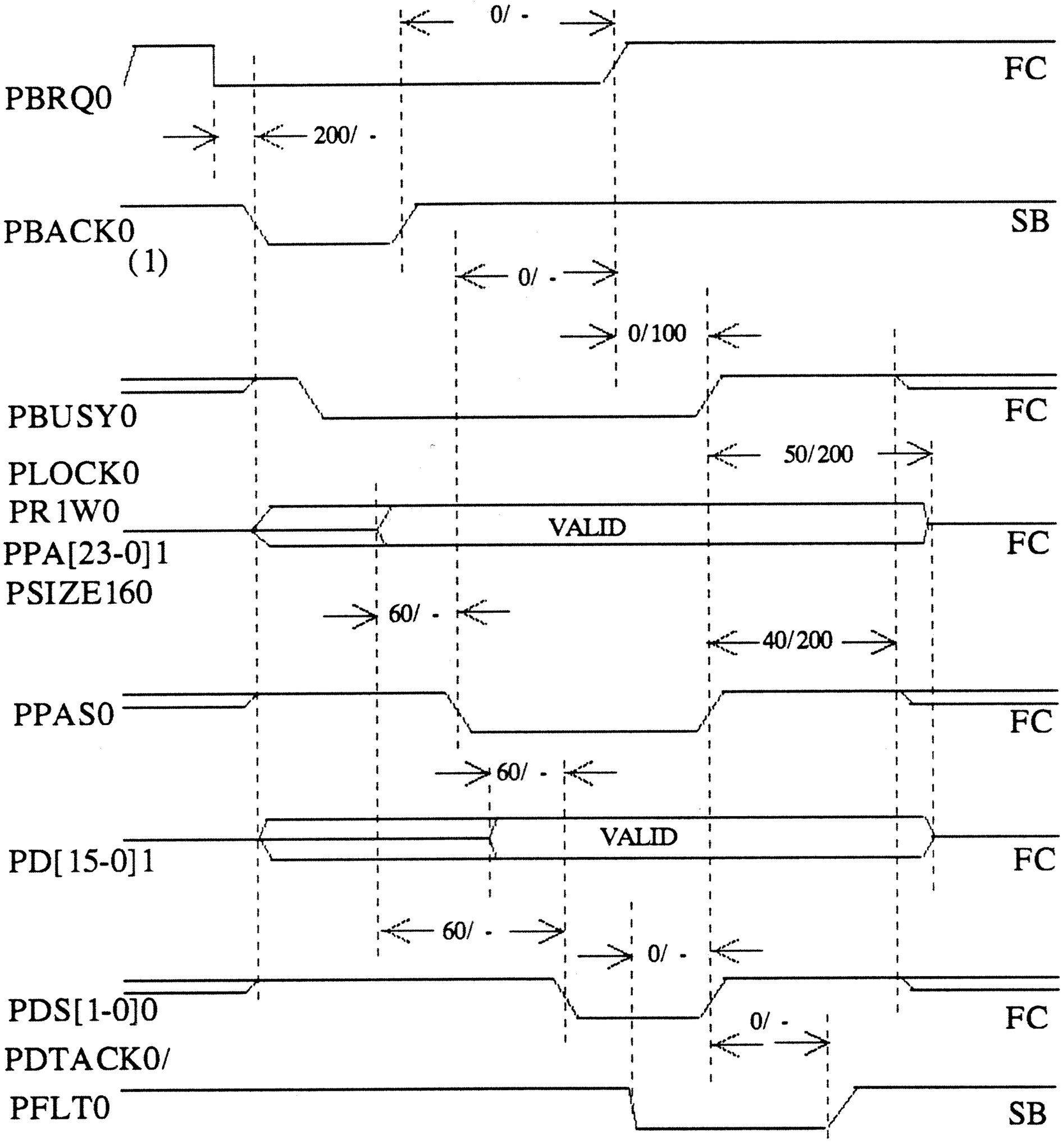
When the data is stable the feature card asserts the appropriate combination of data strobes to indicate the valid byte(s) of data on the I/O bus. The main memory controller accepts the data and activates PDTACK0. PDTACK0 causes the feature card (master) to drive PBUSY0, PPAS0, and PDS[1-0]0 high before tri-stating these along with the address, data, and status signals. PDS[1-0]0 going high causes the main memory controller (slave) to remove PDTACK0.

All times shown are in nanoseconds as seen by the feature card.

TIMING DIAGRAM NOTES:

1. PBACKO0 of the acknowledged feature card shall remain blocked (high) for duration of the cycle.

FEATURE CARD WRITE TO MAIN MEMORY



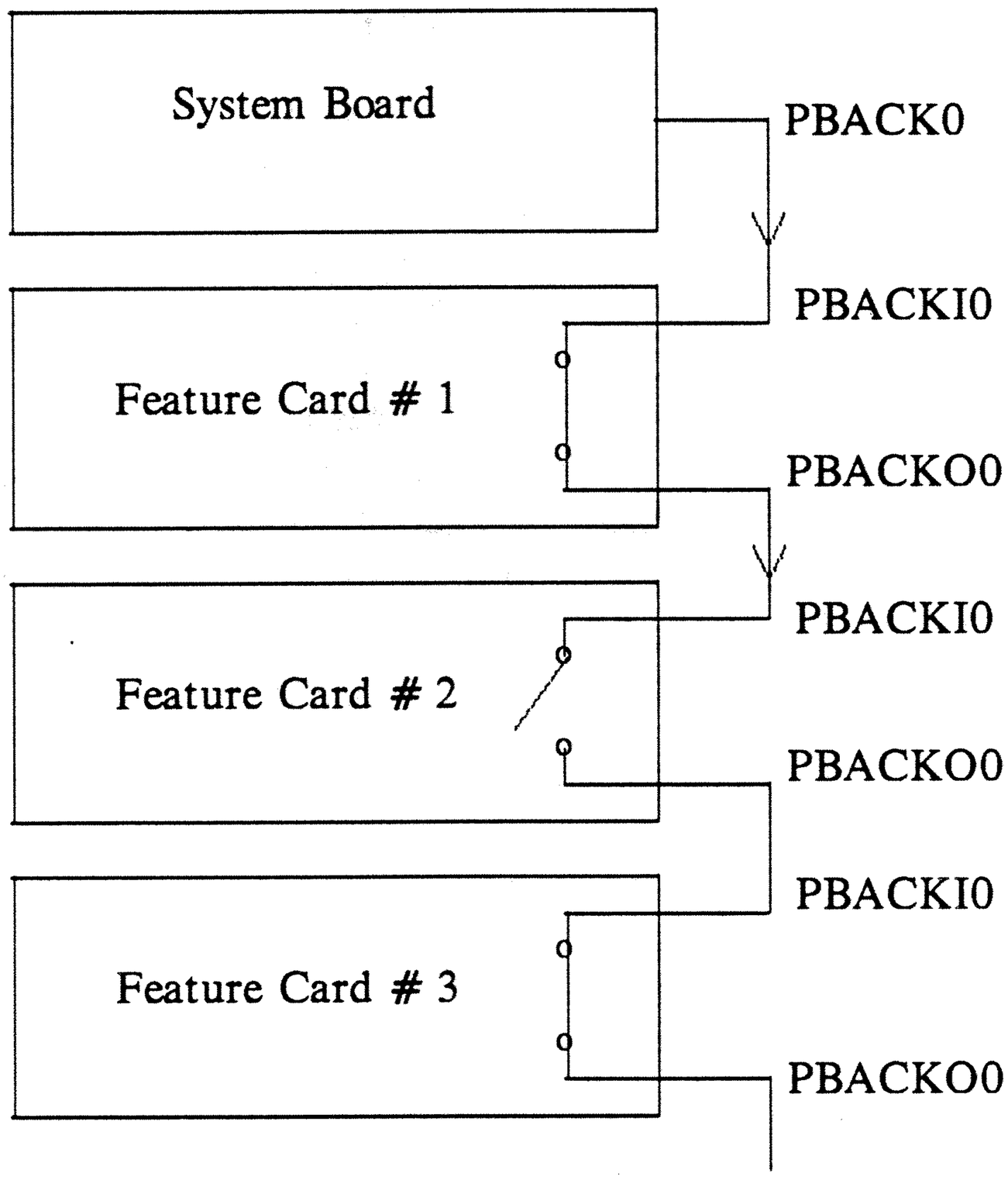
DAISY-CHAINED CIRCUITS

The Bus Acknowledge and Interrupt Acknowledge leads are daisy-chained through all of the feature cards. When requesting control of the bus, or interrupting the system board, each feature card can break the chain. This gives the cards that are electrically closer to the system board a higher priority than the ones further away.

Some of the daisy-chain requirements are:

1. While bus master, a peripheral must not allow PBACKO0 to propagate to the next device in the chain.
2. A peripheral must not drive PBRQ0 if PBRQ0 is currently active. Once PBRQ0 is recognized as inactive, the peripheral must delay before asserting it.
3. A peripheral may not glitch the daisy-chained output signals.
4. A peripheral may not preempt another's bus mastership.
5. The latency from PBACKI0 to PBACKO0 should be kept under 9 nanoseconds.

DAISY-CHAINED CIRCUITS



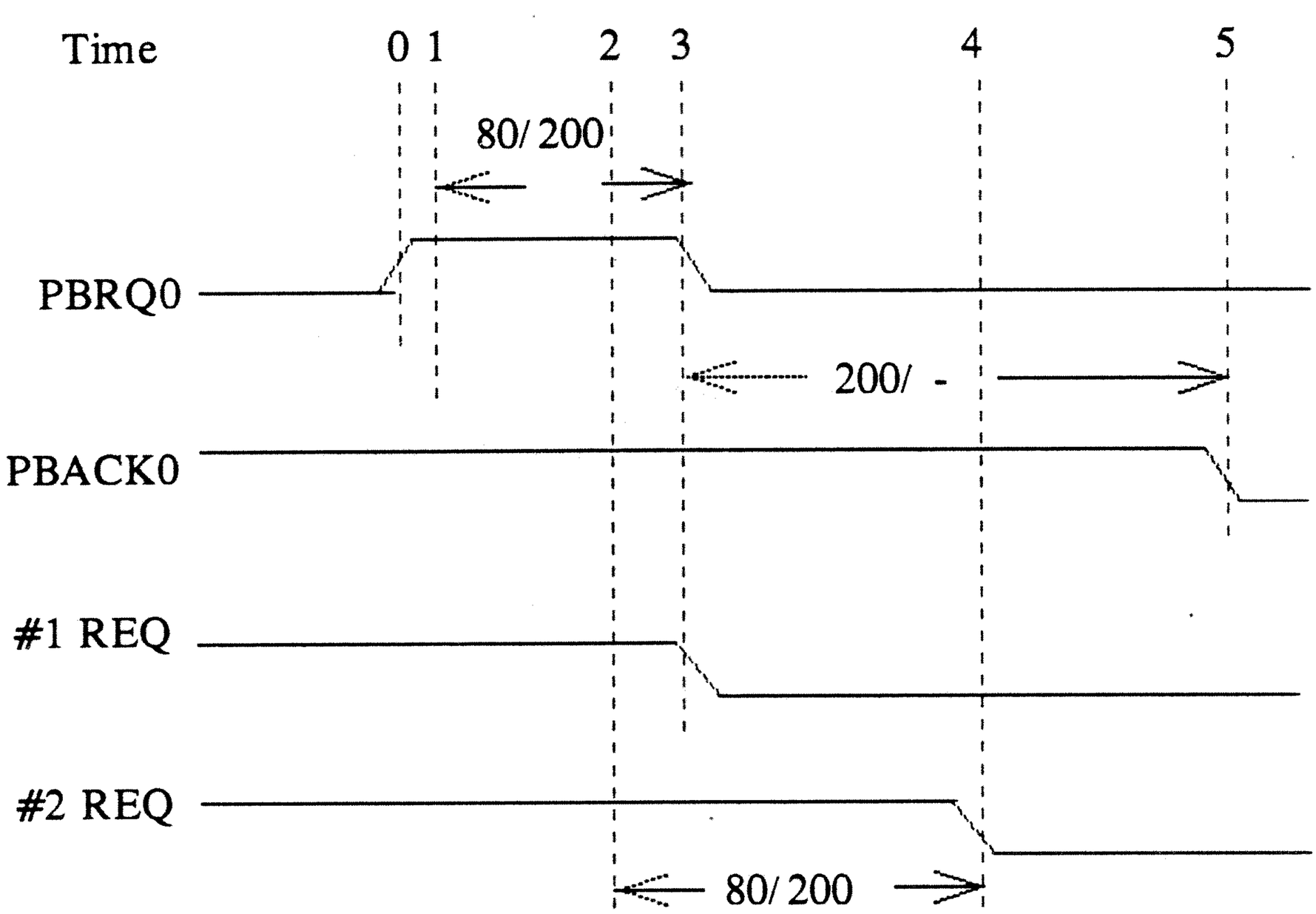
BUS REQUEST - BUS ACKNOWLEDGE PROTOCOL

A feature card may not drive PBRQ0 if PBRQ0 is currently active. Instead, it must hold off until it sees PBRQ0 go inactive, delay a minimum of 80 nanoseconds, and then drive PBRQ0. This delay guarantees a window large enough for all feature cards to see the inactive PBRQ0. The arbiter on the system board must guarantee a delay of at lest 200 nanoseconds from PBRQ0 going active to sending out PBACK0. Each feature card, therefore, has 200 nanoseconds to block PBACKI0 from propagating to PBACKO0, assuming it wishes to use the bus.

The chart illustrates the sequence of events when two feature cards, #1 and #2, both attempt to gain control of the I/O bus at about the same time. At time T0 PBRQ0 goes inactive from some previous use. Card #1 sees that this is inactive at T1, and #2 sees it at T2. At T3, 80 to 200 ns after T1, card #1 asserts PBRQ0 and blocks PBACKO0. At T4, 80 to 200 ns after T2, card #2 asserts PBRQ0 and blocks its PBACKO0. The acknowledge signal PBACK0 is sent out at T5 time, at least 200 ns after T3. The feature card that is nearest the system board will get this signal and will start its bus cycle, while blocking the acknowledge from going to the other cards.

All times shown are in nanoseconds as seen by the feature card.

BUS REQUEST - BUS ACKNOWLEDGE PROTOCOL



TYPICAL BUS EXCHANGE SEQUENCE

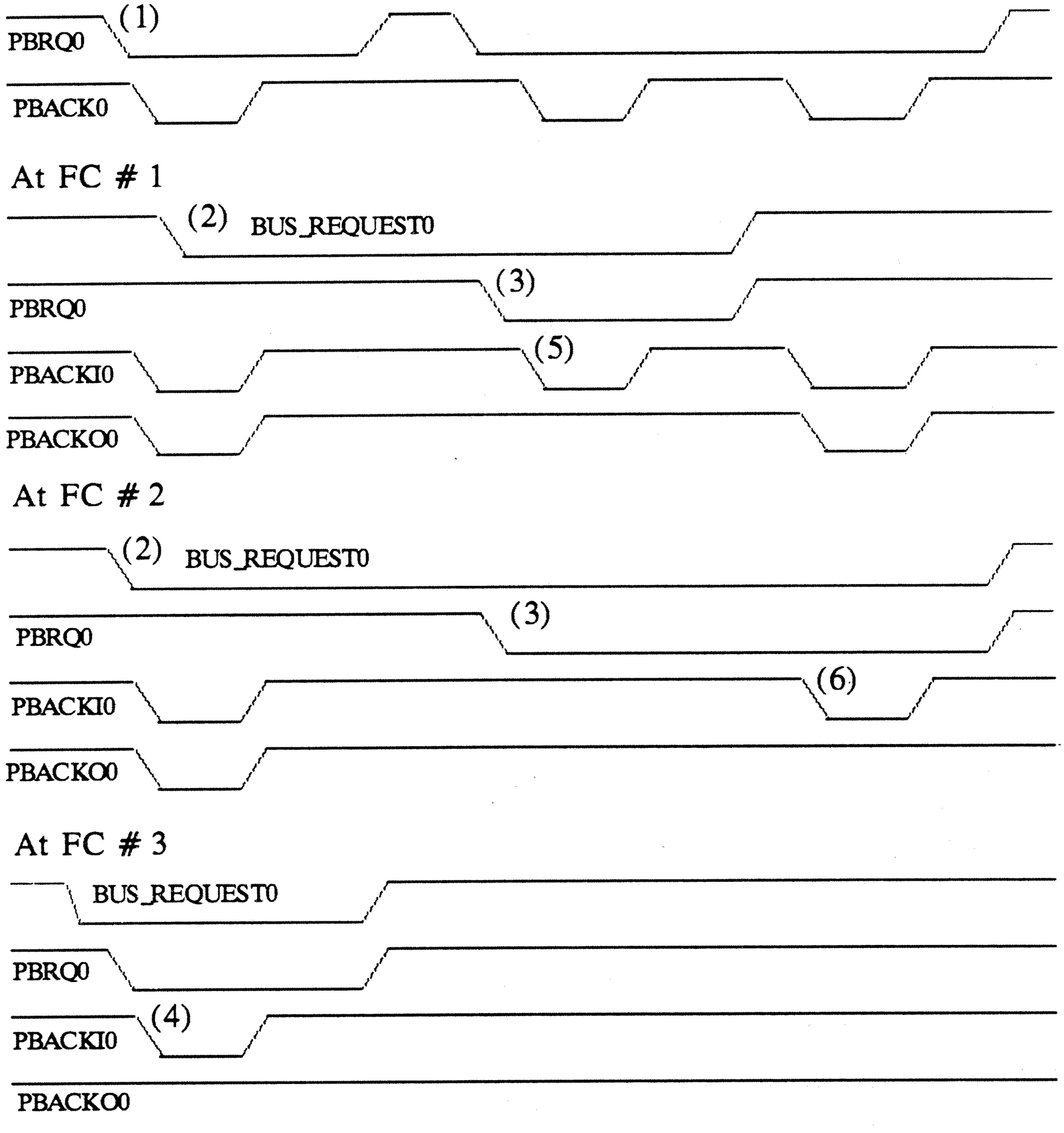
This chart shows the signals that are generated when three feature cards attempt to gain control of the I/O bus. Card #1 is first in the daisy-chain and #3 is last (furthest from the system board). The top part of the chart shows the actual signals at the system board, while the other three parts show what each of the feature cards is doing.

As indicated by the BUS_REQUEST0 lines, peripheral #3 wishes to use the I/O bus first, followed by #2 and then #1. Card #3 is granted use of the bus first. However, #1 gets control next because it is nearer to the system board (higher priority) than card #2.

TIMING DIAGRAM NOTES:

1. PBRQ0 (at the system board) indicates the actual I/O Bus signal PBRQ0 as shared by all peripherals. PBRQ0 (at each peripheral) indicates how the given peripheral is driving the I/O Bus PBRQ0 signal.
2. BUS_REQUEST0 indicates when a given peripheral wishes to use the I/O Bus.
3. PBRQ0 (of peripheral) is not asserted, however, until the shared I/O Bus signal PBRQ0 (at the system board) is detected as inactive.
4. Peripheral #3 gets on the bus. (Peripherals #1 and #2 pass PBRQ0 on through.)
5. Peripheral #1 gets on the bus.
6. Peripheral #2 gets on the bus. (Peripheral #1 passes PBACK0 on through.)

TYPICAL BUS EXCHANGE SEQUENCE



TYPICAL REQUEST/ACKNOWLEDGE LOGIC

This circuit uses an 8-megahertz clock to generate the greater than 80 nanosecond delay before asserting PBRQ0. Flip-flop number one is clocked low when the BUS_REQUEST0 signal is active and PBRQ0 is inactive. The output of the flip-flop is used to block the propagation of the bus acknowledge signal PBACK0. One clock cycle later F/F2 is clocked low generating a PBRQ0 signal and allowing the card to accept the next PBACKI0 signal.

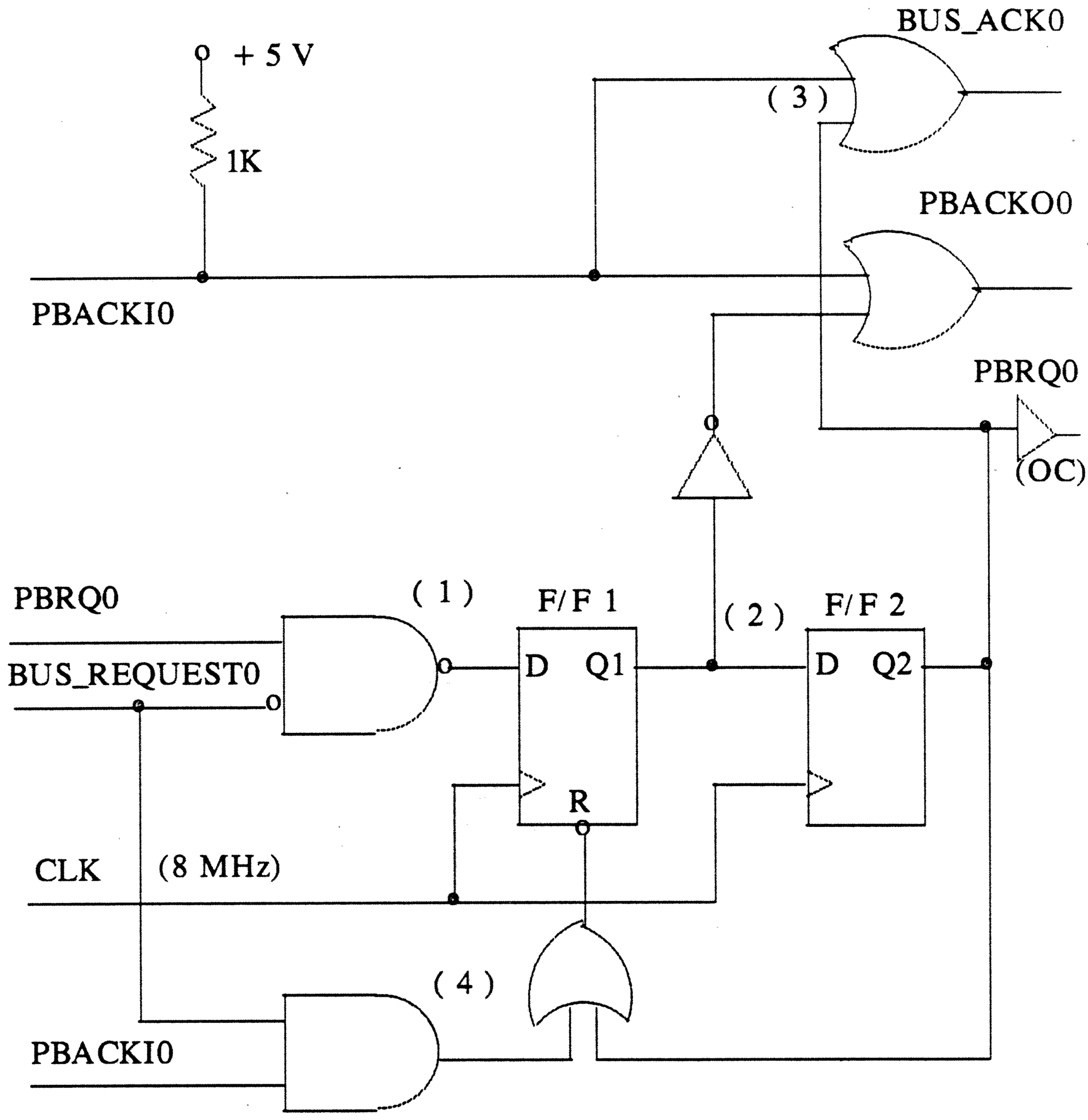
F/F1 is held in the off condition until both PBACKI0 and BUS_REQUEST0 go inactive. On the next clock cycle it is turned on allowing PBACKI0 to be passed on to the other feature cards. One clock cycle later F/F2 is turned on putting PBRQ0 in the inactive state.

The delay from PBACKI0 to PBACKO0 should be less than 9 nanoseconds. This delay is made up of one Schottky gate delay together with the time of flight through the board wiring and two feature card connectors.

LOGIC DRAWING NOTES:

1. F/F1 is clocked low when BUS_REQUEST0 is active & PBRQ0 is inactive. Q1 low forces PBACKO0 inactive. (That is PBACKI0 to PBACKO0 is blocked.)
2. One cycle later, F/F2 is clocked low & PBRQ0 is asserted. Also, F/F1 is held low via reset pin.
3. The next PBACKI0 is interpreted as an acknowledge.
4. When BUS_REQUEST0 & PBACKI0 go away, F/F1 & then PBRQ0 are deactivated.

TYPICAL REQUEST/ACKNOWLEDGE LOGIC



MULTIPLE ACCESS TIMING

A feature card can perform more than one data transfer in a single bus cycle. Such multiple accesses occur without bus arbitration overhead and are thus performance effective. (The system board is not capable of these multiple accesses over the I/O bus.)

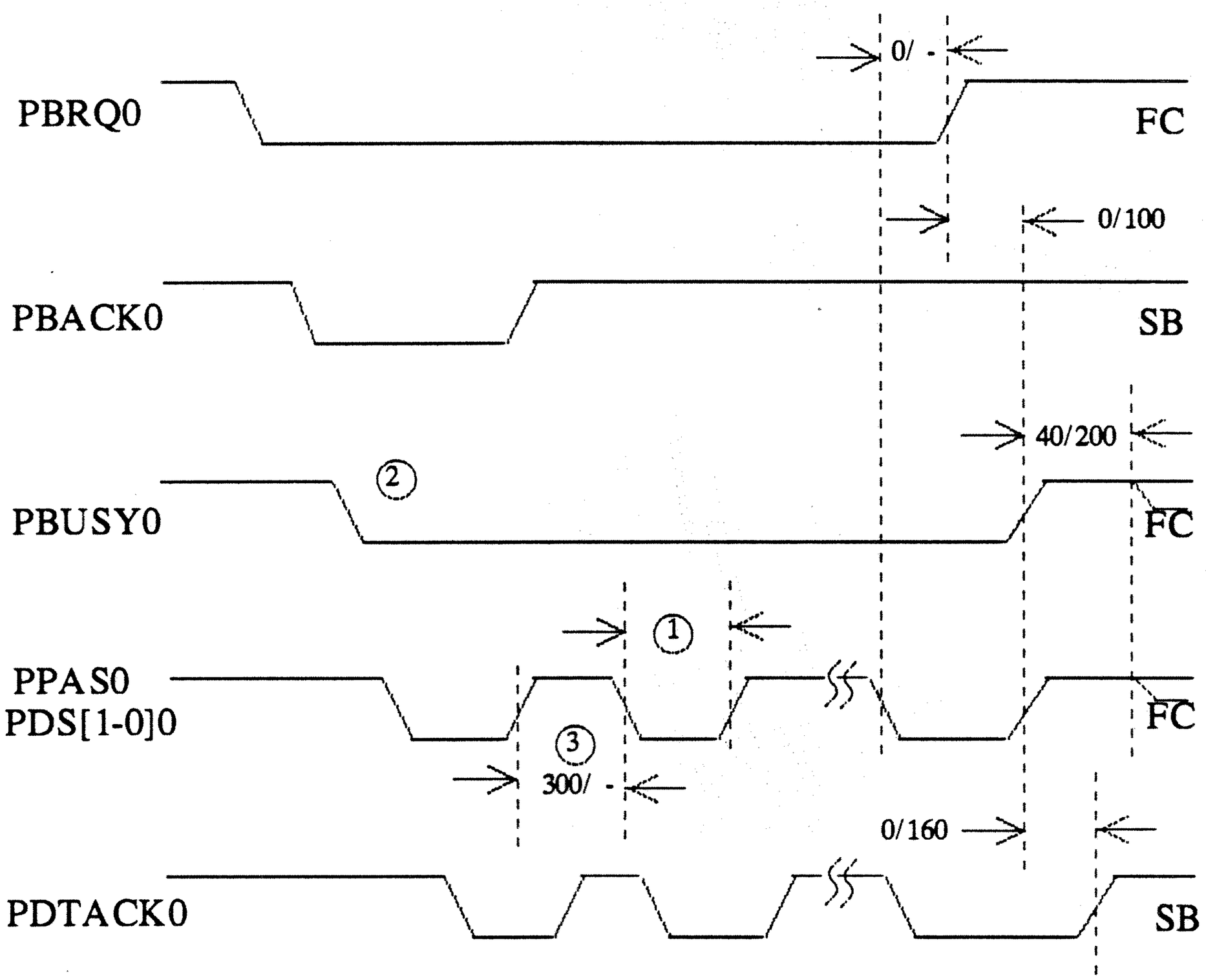
The bus busy signal PBUSY0 implements the multiple access feature. In particular, a cycle starts as a "normal" bus cycle: bus request, wait for acknowledge, and then activate PBUSY0. At this point, the bus master has effectively locked out all devices from using the I/O bus and may therefore perform several uninterrupted data transfers.

The lengths of the entire bus cycle and each separate data transfer must be constrained to prevent timeout failures. Multiple access is ~~limited to a maximum of 4 transfers with the I/O bus cycle~~ limited to 8 microseconds or less. Each data transfer is limited to a maximum of 5 microseconds, and is measured by timing the feature card physical address strobe PPAS0.

TIMING DIAGRAM NOTES:

1. The system board times each transfer.
2. The system board uses PBUSY0 to inhibit further PBACK0's.
3. The system board can temporarily preempt a feature card bus cycle to force a main memory refresh. Such a preemption can only occur on a transaction boundary.

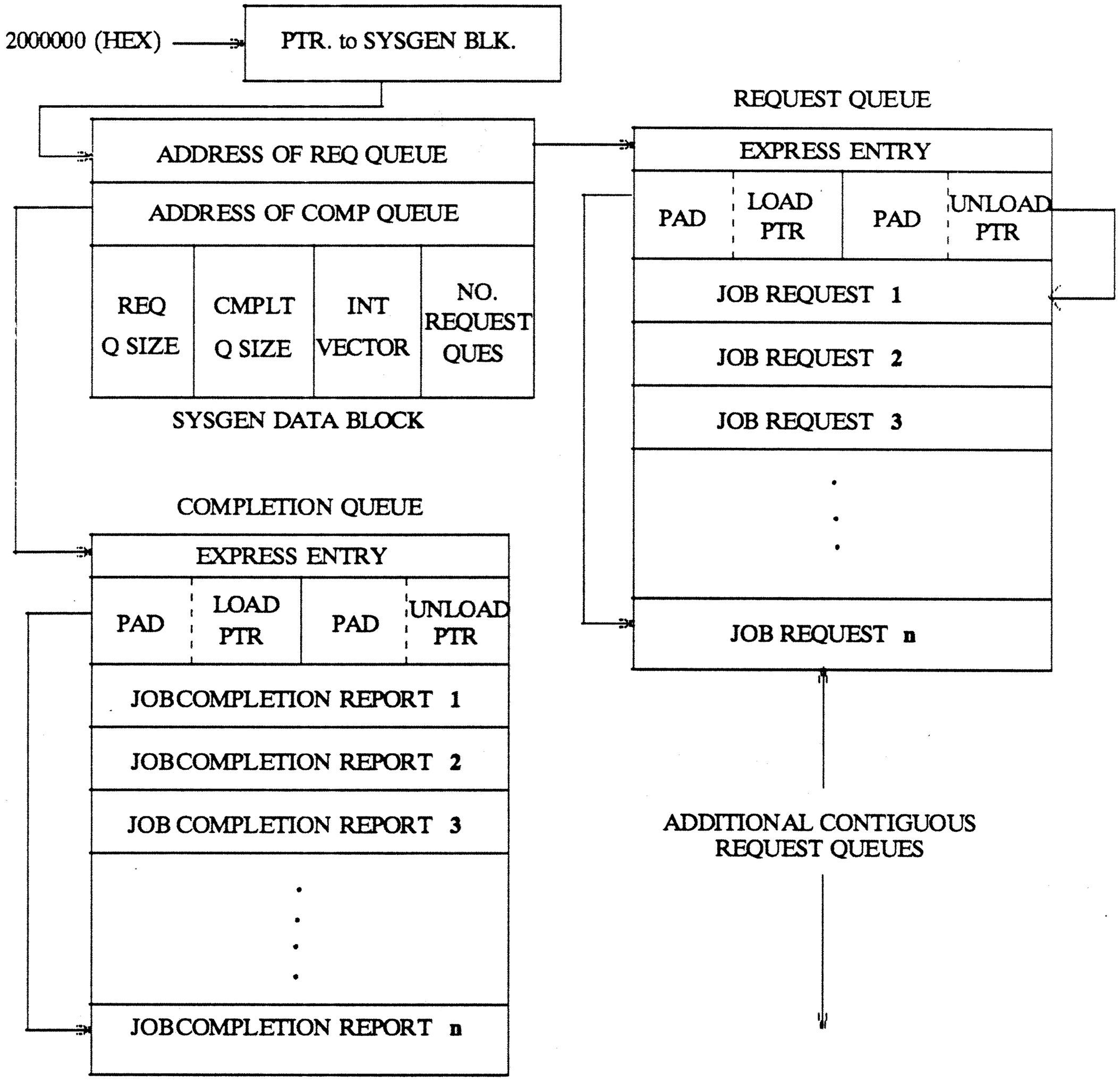
MULTIPLE ACCESS TIMING



QUEUE STRUCTURES

This diagram shows the queue structure that is used by the common I/O based feature cards. Other queue structures could be used for new feature cards.

QUEUE STRUCTURES



UNIT 5

INTELLIGENT (smart) FEATURE CARDS

Lesson 5

Lab Exercise #6

LAB EXERCISE #6 Answer Page.

2. Console Message _____

_____

_____

LAB EXERCISE #6

Equipment required:

3B2 Computer

"Ports" card installed in slot #1.

HR1 Custom Feature Card installed in slot #2.

Console Terminal

User Terminal

Oscilloscope 150 MHZ min.

This lab session will give the student an opportunity to review the AT&T 3B2 Computer feature card interrupts.

1. When analyzing the interrupt circuits refer to the HR1 schematic as well as to the relevant material covered in class. Notice that the interrupt circuitry is drawn in red. The HR1 schematic drawing consists of six sections named FILE f10, rpg1, f4, f1, f9 and f6.
2. To generate an interrupt to the 3B2 from HR1, press "Control g" at the user terminal. This action causes lead INTFFS (FILE f1, MCS51 pin 21) go low. This is the beginning of the interrupt sequence. (Lead INTFFR is activated only at the HR1 initialization time to put flip flop 39 into a known state). Note the message that is displayed on the console.

LAB EXERCISE #6 Answer Page.

3A. How? _____

Time? _____

INTVWR = _____

3B. Time? _____

3C. Purpose? _____

3D. Purpose? _____

3E. Purpose? _____

3F. Purpose? _____

4. Av Time = _____

5. Av Time = _____

Why Different? _____

LAB EXERCISE #6

3. Examine the HR1 circuit drawing and determine the following (using local net names):

   A. On the FILE f6 page how is the signal INTVWR generated? At what time in the interrupt process is this signal "true"? Write a Boolean expression for it.

   B. On the FILE f6 page at what time after FF 39 has been set by INTFFS is FF 39 reset?

   C. On the FILE f6 page what is the purpose of gate 38 (pins 1, 2, &3)?

   D. What is the purpose of the interrupt circuit on the FILE f1 page?

   E. What is the purpose of the interrupt circuit on the FILE f9 page?

   F. What is the purpose of the interrupt circuit on the FILE f4 page?

4. Use a Scope to measure the time duration of PINTO (IC 17,2). (Scope settings - vert. amp. 1V/Div, horz. amp. 2 Microseconds/Div.) Repeat this procedure 5 times and get the average time.

   Caution! Use Grounding Strap.

5. Repeat step 4, but this time generate the interrupts while the command "ls -l /bin" is being performed. Explain differences in PINTO time durations here and in the step 4 above (if any).

UNIT 6

INTERRUPTS - INTELLIGENT CARD

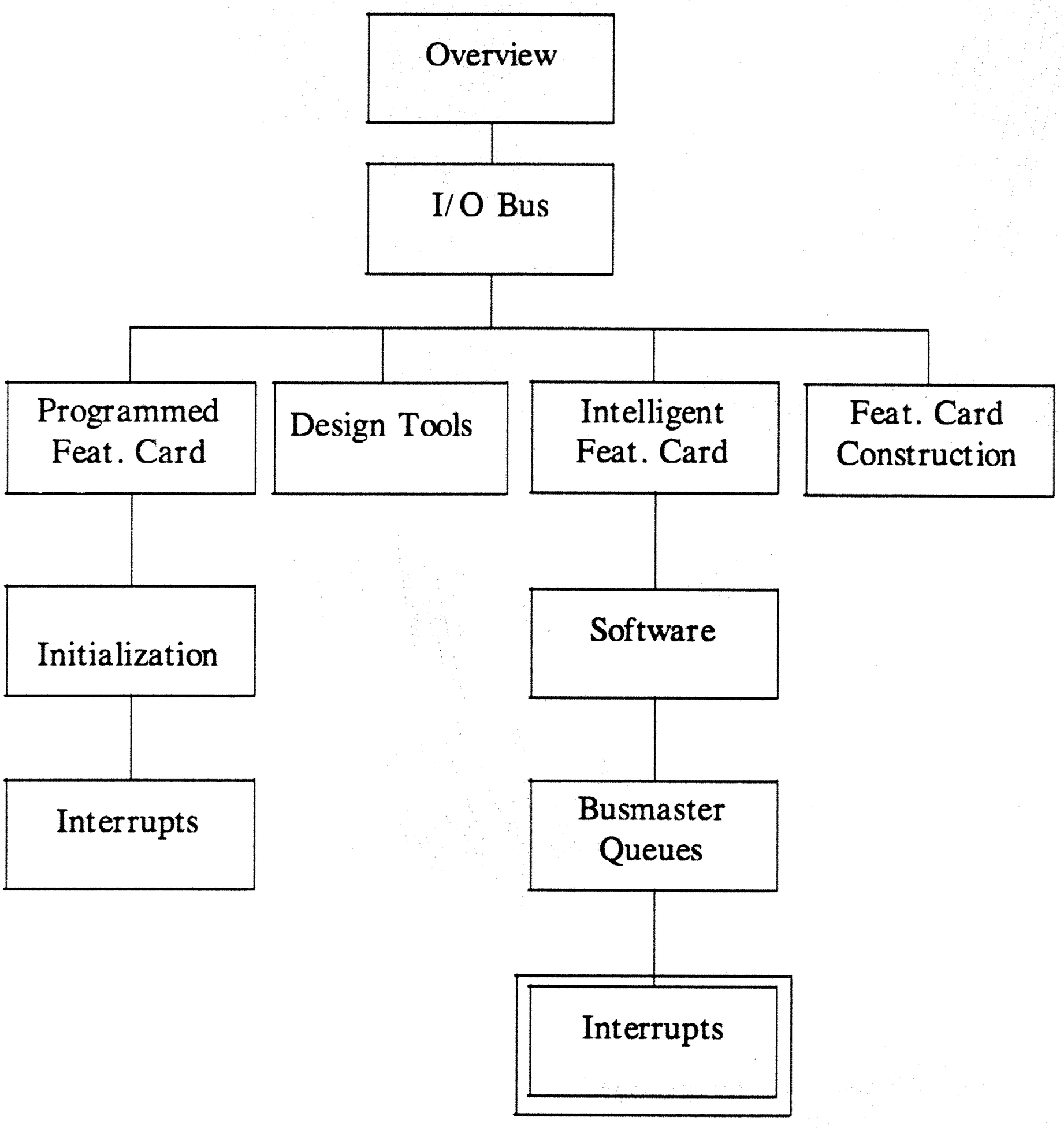
INTERRUPTS - INTELLIGENT CARD

Upon completion of this unit and given a 3B2, two terminals, HR1 Custom feature card, HR1 schematic, dual trace oscilloscope and student guide, the student will be able to:

1. Describe and explain intelligent feature card CPU signaling, and significant I/O Bus leads for Feature Card interrpts of the system board.

The lessons in this unit describe how the interrupts are used by the intelligent feature cards to communicate with the system board.

2302 Outline



UNIT 6

INTERRUPTS - INTELLIGENT CARD

Lesson 1

Intelligent Feature Card Interrupts

INTELLIGENT CARD - CPU SIGNAL

The intelligent cards have a very limited list of things that they can do when interfacing with the system board. Like the programmed cards, they can interrupt the system board to signal that some event has taken place on the feature card. For the CIO-based cards this interrupt is used to signal that the completion queue (in main memory) has been filled by the feature card, and that it is now the time for the system board to take some action with this data. (See Unit 4 Lesson 1 for programmed card interrupt operation.)

For the CIO-based cards the system board has a very limited access to the feature card. Three feature card registers can be accessed by the system board CPU. The first, at offset address 000000 or 000001 hex, can only be read by the system board. At one time it will hold the identification code for the card, and after that it contains the interrupt vector. The other two registers, Control at offset 000003 and Status at 000005, are nothing more than addressable locations, and can not be written into or read by the system board. (See page 1.2.12.)

Other intelligent cards are free to interface with the signals from the system board in any way that they wish, but the timing restraints are quite severe. Each card has a unique 2-megabyte address range as far as the system board is concerned, but the addresses used probably will have to connect directly with feature card hardware devices. The time allowed for the feature card to respond is too short for the CPU on the FC to give a programmed response via some sort of interrupt program, however, a DMA access to the feature card would be possible.

INTELLIGENT CARD - CPU SIGNALING

- Feature Card to System Board

  3 Interrupt Request leads

- System Board to Feature Card

  Access (read or write) of Feature Card registers will cause an interrupt of the FC CPU (CIO-based card)

- Feature Card I/O Bus Master

  Feature Card read of Main Memory

  Feature Card write of Main Memory

I/O BUS LEADS USED WHEN FC INTERRUPTS SB

See page 4.1.7 for a description of how these signals are used.

I/O BUS LEADS USED WHEN FC INTERRUPTS SB

PINT20

PINT10

PINT00

PIAK20

PIAK10

PIAK00

PR1W0

PPAS0

PD[15-0]1

PDS[1-0]0

PDTACK0

UNIT 7

FEATURE CARD CONSTRUCTION

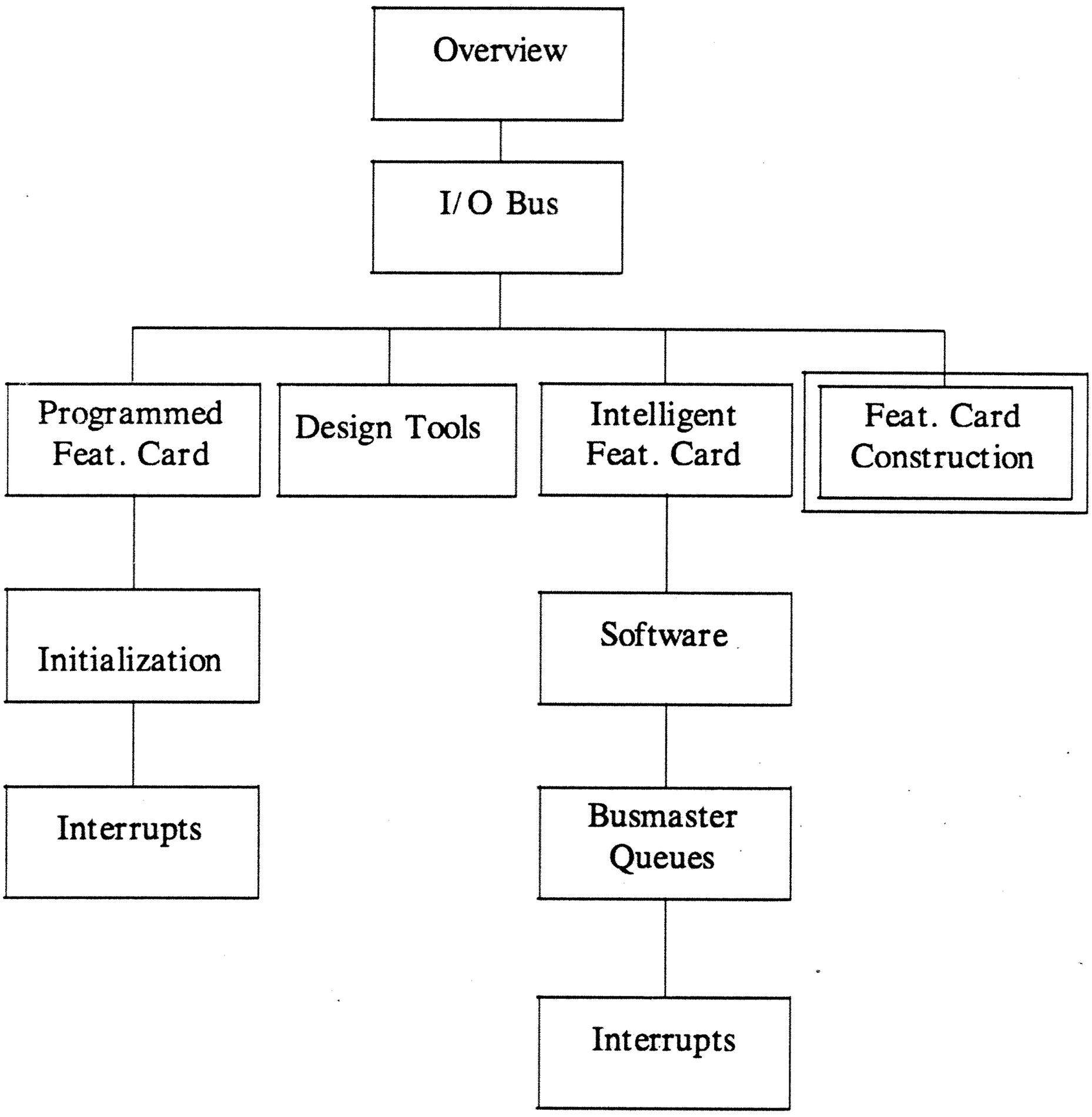
FEATURE CARD CONSTRUCTION

Upon completion of this unit and given a 3B2, two terminals, HR1 Custom feature card, HR1 schematic, dual trace oscilloscope and student guide, the student will be able to:

1. Describe the electrical, physical and testability requirements for any new feature cards for the 3B2 computer
2. List suggested 3B2 feature card components.
3. Explain the common I/O circuit hardware, components, and block diagram.

The four lessons in this final unit will discuss the electrical and physical requirements for any new feature cards for the 3B2 Computer. Some suggestions on the selection of components for new cards will be given, and the common I/O circuit will be described.

2302 Outline



UNIT 7

FEATURE CARD CONSTRUCTION

Lesson 1

Electrical Requirements

I/O BUS TERMINATIONS

Resistor networks are resident on the I/O expansion board. They eliminate signal reflections caused by the bus length being greater than its signals rise time. An unloaded bus has a characteristic impedance of about 59 ohms and a propagation delay of about 2.5 nanoseconds per foot. Loading the bus with taps increases the total bus capacitance, further lowering the bus characteristic impedance. While terminating the bus with its characteristic impedance would eliminate bus reflections, it would be extremely difficult to drive such a bus. Some termination, however, is necessary. Therefore, the address, data and status lines are terminated.

Address, data, and status signals are terminated in a voltage divider network consisting of 390 ohms to VCC and 560 ohms to ground. This brings an undriven line to about 3 volts instead of 5 volts, resulting in less voltage swing, therefore, minimizing cross talk.

All feature card select signals have a 1 kilohm pull-up resistor located on the feature card. The daisy-chained signals have a 1 kilohm pull-up resistor located on the feature card input signal.

Termination of the I/O bus also helps in minimizing radio frequency interference (RFI).

I/O BUS TERMINATIONS
(in Ohms)

Signal	Type	Termination	Location
PBACKI0	tp	1k	p
PBACKO0	tp	-	na
PBRQ0	oc	390	bp
PBUSY0	ts	390	bp
PCS[15-01]0	tp	1k	p
PDS[1-0]0	ts	390	bp
PDTACK0	oc	195	bp
PD[15-00]1	ts	*	bp
PFAIL0	oc	390	bp
PFLT0	oc	195	bp
PIAKI[2-0]0	tp	1k	p
PIAKO[2-0]0	tp	-	na
PINT[2-0]0	oc	390	bp
PLOCK0	ts	*	bp
PPAS0	ts	390	bp
PPA[23-00]1	ts	*	bp
PR1W0	ts	*	bp
PSIZE160	oc	195	bp
RQRST0	oc	390	bp
SYSRST0	tp	390	bp

Notes:
ts = tri-state
oc = open-collector
tp = totem-pole
* = 390 to VCC and 560 to GND
- = none
bp = backplane
p = peripheral
na = not applicable

I/O BUS LOAD CURRENTS

In the chart, current flowing out of a terminal is given as a negative value.

I/O BUS LOAD CURRENTS

Signal	Type	Iol (mA)	Ioh (mA)	Iil (mA)	Iih (uA)
PBACKI0	S	20	-1	-4	100
PBACKO0	S	20	-1	-4	100
PBRQ0	LS	40	oc	-.8	40
PBUSY0	LS	24	-15	-.8	40
PCS[15-01]0	LS	8	-.4	-.8	40
PDS[1-0]0	LS	24	-15	-.8	40
PDTACK0	S	40	oc	-.4	20
PD[15-00]1	LS	24	-15	-.8	40
PFAIL0	LS	40	oc	-.8	40
PFLT0	S	40	oc	-.4	20
PIAKI[2-0]0	LS	8	-.4	-.8	40
PIAKO[2-0]0	LS	8	-.4	-.8	40
PINT[2-0]0	LS	40	oc	-.8	40
PLOCK0	LS	24	-15	-.8	40
PPAS0	LS	24	-15	-.8	40
PPA[23-00]1	LS	24	-15	-.8	40
PR1W0	LS	24	-15	-.8	40
PSIZE160	S	40	oc	-.4	20
RQRST0	LS	40	oc	-.8	40
SYSRST0	LS	24	-1.2	-.8	40

Notes:
S = Schottky driver
LS = Low Power Schottky driver
oc = open collector circuit

I/O EXPANSION CARD CURRENT LIMITS

The values shown are the maximum **total** currents allowed for all of the feature cards that are plugged into the expansion card.

The backup battery is also used to supply power to the time-of-day clock and the NVRAM on the system board. The lifetime of this battery will depend on the temperature, the duty cycle, and the standby current drain. It will supply standby power to the system board for about six years. The table below shows how this lifetime will be derated by I/O Bus current.

Standby Current (microamps)	Lifetime (years)
0	6.0
10	4.0
20	3.0
30	2.4
40	2.0
50	1.5
60	1.1

Low frequency capacitors should be provided on all power supply inputs to the feature cards. Local high frequency capacitors should be distributed on the feature cards to filter transient voltage spikes on the power leads. These capacitors should have low effective series resistance and inductance. Suggested values are:

Low frequency - 100 to 400 microfarads

High frequency - 0.01 to 0.1 microfarads

I/O EXPANSION CARD CURRENT LIMITS

- VCC 7.2 amperes (3B2/300)
  21.6 amperes (3B2/400)
- V12P 75 milliamperes
- N12P 75 milliamperes
- VBKUP 20 microamperes

TESTABILITY

The following testability requirements are suggested for all feature card designs. The objective is to increase access to a circuit in order to provide observability and control of the circuit. While these are not hard and fast requirements, it is suggested that they be observed whenever possible.

TESTABILITY

- Do not exceed 2400 signal nets.
- Use tri-state output devices wherever possible.
- All oscillators must be capable of being disabled.
- Provide initialization of all latches and registers.
- Avoid long counter chains.
- Provide test access between counter stages.
- Provide parallel load capability to counters.
- Avoid the use of common pull up resistors to the CLEAR/PRESET inputs to flip-flops.
- Where internal tri-stateable buses are not to be always active, provide pull up resistors.
- Unused input pins should be connected to ground or Vcc through a resistor (50 ohms minimum).

TESTABILITY (cont)

TESTABILITY (cont)

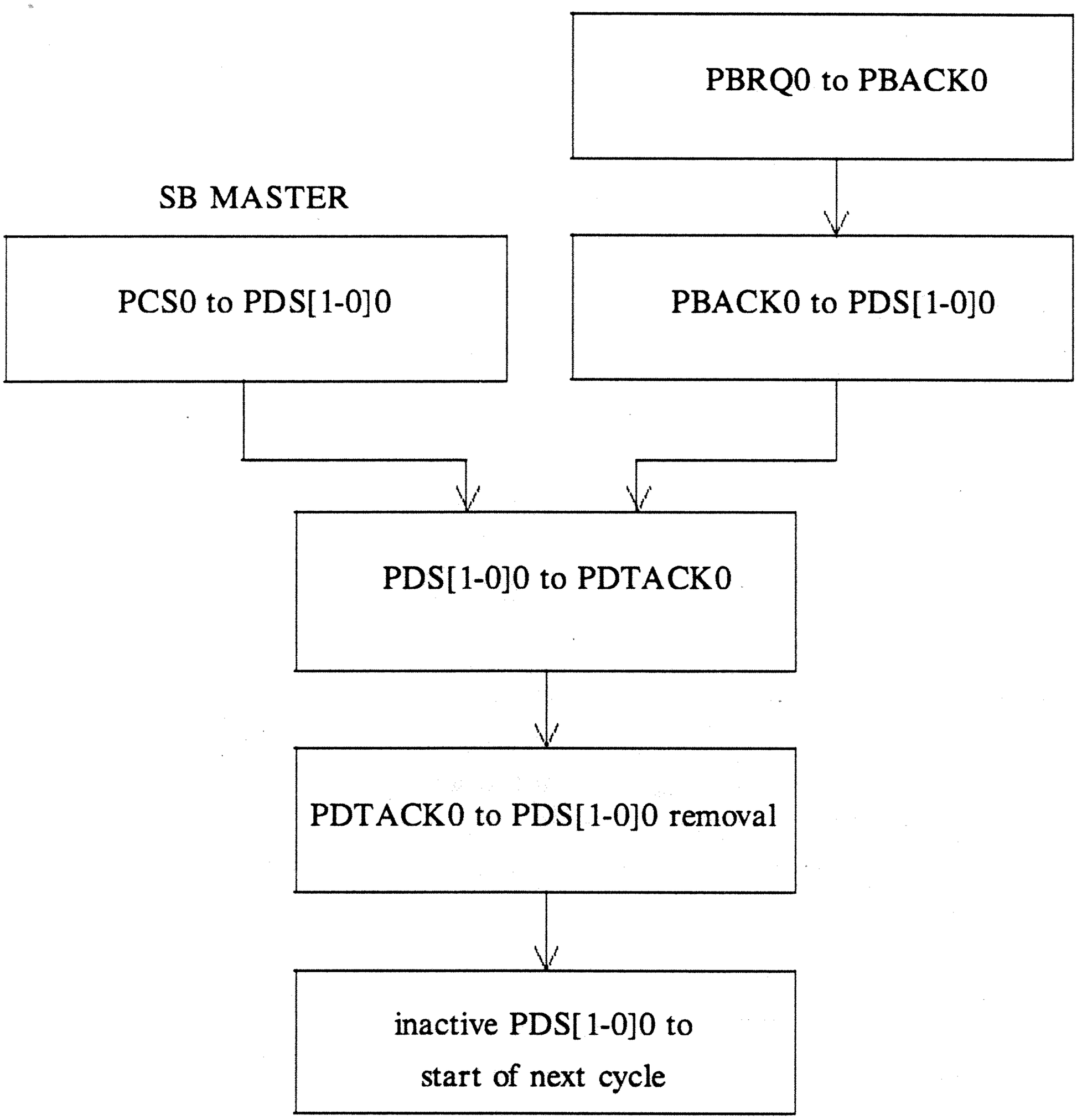
- An output pin may not be tied to an input pin of the same IC.
- Logic drives to LEDs and display circuits should be available to the tester.
- Feedback connections between the device inputs and outputs should be made through a simple blocking function.
- Custom gate arrays must always have an output disable control pin.
- MIP packages must be scratch probe-able.
- Do not use "WIRED AND or WIRED OR" logic functions.
- The dynamic stay-a-live speed must be less than 10 MHz.
- Power-on initialization or faceplate initialization switches must be monitorable.
- Buffer all edge-connector I/O.

I/O BUS BANDWIDTH - (CRITICAL PATH)

The amount of information (data) which can be transferred over the I/O Bus defines bus bandwidth. It is a function of the architecture of the bus as well as the efficiency of the system board arbiter and feature card design. The drawing shows the critical paths when the system board is busmaster and when the feature card is busmaster.

To get a complete definition of bus bandwidth we must take into account the bandwidth available to each feature card. The bus access priority for a feature card will have an effect on the bandwidth for that card. An increase in the number of feature cards will decrease the bandwidth available to each card. Since the system board DMA subsystem also resides on the I/O Bus, an increase in disk activity will cause a decrease in bus availability for the feature cards.

I/O BUS BANDWIDTH - (CRITICAL PATH)



BUS CAPACITANCE

PBACKI0 and PIAKI[i]0 are allowed a maximum of **three** input loads and one output load.

PSIZE160, PDTACK0, and PFLT0 are allowed a maximum of **one** input load and one output load.

The feature cards must be able to drive 700 picofarads per tap.

DISTRIBUTED LOAD CAPACITANCE PER TAP
(picofarads)

Device	Capacitance
Input Buffer	5
Output	14
Via *	1
MLB Routing #	2.4 per inch
Edge Connector	3

* Via = Plated through hole on feature card printed circuit board (maximum 6)

MLB = Multilayer board (maximum routing 6 inches)

BUS CAPACITANCE

- Two 74LS TTL Input Loads
- One 74LS Output Load
- 45 Picofarads Maximum

EXCEPTIONS

PBACKI0 and PIAKI[i]0

PSIZE160, PDTACK0, and PFLT0

UNIT 7

FEATURE CARD CONSTRUCTION

Lesson 2

Physical Requirements

FEATURE CARD DESIGN CONSIDERATIONS

When designing feature cards, certain design parameters should be considered in order to develop peripheral equipment which will work in the same environment as the 3B2/300 Computer.

- Noise: The 3B2/300 Computer has been designed for audible noise limits of 40/42 dbA, measured 1.0 meter from the cabinet.

- Shock: The 3B2/300 Computer will continue to operate under intermittent shocks of up to 3.0 g, not to exceed 10.0 milliseconds, with less than two repetitions per second. Severe shock (25 g for 10 msec) and vibration will not cause permanent mechanical or electrical damage; however, continuous operation through such periods is not guaranteed.

- Temperature-Humidity-Altitude: Worst case requirements are shown below. Specifications given on humidity assume no condensation and a maximum gradient of 10 percent per hour.

  - Temperature: From 5 to 38 degrees Celsius at sea level (operating)
  - Temperature: 34 degrees Celsius at 6000 feet (operating)
  - Temperature: -10 to 50 degrees Celsius (storage)
  - Relative humidity: 20 to 80 percent (operating)
  - Relative humidity: 10 to 95 percent (storage)
  - Altitude for shipping: 983 feet below sea level to 12,000 feet above sea level (ground transportation)
  - Altitude for shipping: To 10,000 feet above sea level or to a barometric equivalent of 688 millibars (air transportation)

  Provisions have been made to protect the 3B2/300 Computer against thermal runaway and excessive output voltage from its 5-volt supply. Thermal shutdown occurs at approximately 80 degrees Celsius plus or minus 5 degrees. Over voltage shutdown occurs at 6.25 volts plus or minus 0.75 volts. The computer can be reset by removing the AC power.

- Fire: Feature card parts and materials should be fire retardant in compliance with Underwriters Laboratories specifications number 478.

FEATURE CARD DESIGN CONSIDERATIONS

- Noise
- Shock
- Temperature
  - Humidity
  - Altitude
- Fire

SAFETY STANDARDS

The standards shown must be adhered to in order to meet the safety requirements of the United States, Canada, and most European countries. By following the IEC specs using the VDE addendum most of the common market countries will be covered in addition to some other countries which do not have specs.

For a complete listing of test specifications see:

R. J. Brandt, "UL, CSA, and Foreign Standards," November 1, 1983.

SAFETY STANDARDS

COUNTRY	STANDARD
United States	UL478 (UL)
Canada	C22.2 No.154 (CSA)
Europe	IEC380 (IEC)
	IEC435
West Germany	VDE0806/8.81 (VDE)

EMC STANDARDS

All I/O feature cards are required to meet these Electromagnetic Compatibility (EMC) standards when installed in the 3B2 Computer. The 3B2 must meet both the UL and VDE standards by a margin of 6 decibels.

For a complete listing of the test conditions see:

J. J. Fischer, "EMC Requirements for AT&T 3B2 I/O Feature Cards," December 4, 1984.

EMC STANDARDS

COUNTRY	STANDARD
United States	FCC Part 15, Subpart J, Class A
West Germany	VDE Specification 0871 Class A, and 0875 Class A

UNIT 7

FEATURE CARD CONSTRUCTION

Lesson 3

Feature Card Component Selection

COMPONENT SELECTION

The objective of the following component selection suggestions is to design a circuit that is easily manufactured and tested. If the suggestions are followed, the product should be less expensive to build and repair.

COMPONENT SELECTION

- All components must be industry standard values.
- All components must be used within the data sheet specifications.
- The use of military components should be avoided.
- No gold leaded devices shall be used.
- Components should be machine insertable.
- Components shall be compatible with both chlorinated solvents and Water Soluble Flux and Total Immersion Cleaning (WSF/TIC).
- Sole source parts should not be specified.
- Oscillators must have a built-in output disable function.
- Consideration must be given to repair procedures if a component must be replaced.

INTEGRATTED CIRCUIT SELECTION

Integrated circuits are the single largest material cost item in most computer products. In order to avoid procurement problems, the following selection rules are recommended.

INTEGRATTED CIRCUIT SELECTION

- No gold leads shall be permitted.
- EPROMs and other socketed ICs shall be specified with tin leads.
- Lead fusing shall not be specified.
- Whenever possible, devices shall use plastic packaging.
- ICs must be specified with + or - 5% Vcc tolerance.
- PROMs shall be avoided where possible.
- All ICs shall be used within the vendor's data sheet specifications.
- No ICs shall be used that require any "special selection" tests.

INTEGRATTED CIRCUIT SELECTION (cont)

INTEGRATTED CIRCUIT SELECTION (cont)

- All ICs shall be specified over the standard 0 to 70 degrees C temperature range.
- Custom gate arrays must have an output disable pin.
- All pins on Multiple In-line Package (MIP) devices must be probe-able from the top of the circuit board.
- Provision for the mixing of PROM, EPROM, RAM, and ROM devices from multiple vendors must be planned for during the initial design phase.
- Sockets should be avoided, but where they are required, tin leaded sockets shall be used.

UNIT 7

FEATURE CARD CONSTRUCTION

Lesson 4

Common I/O Circuit

COMMON I/O BASED FEATURE CARDS

The Common I/O (CIO) circuits have been designed to simplify the job of designing intelligent (smart) feature cards for the 3B2 Computer. CIO contains all of the logic needed to interface with the I/O Bus, and leaves room for the applications circuits on the card. The 80186 CPU is used to both control the I/O Bus interface and to operate the application hardware.

COMMON I/O BASED FEATURE CARDS

- CIO Hardware
  I/O Bus Interface

- Application Hardware

CIO SUBCOMPONENTS

- CPU (80186) - The 80186 CPU plus the 82S105 Field Programmable Logic Sequencer (FPLS) provide the control of the feature card interface with the I/O Bus. Three types of operation are supported.

  1. 80186 read or write of main memory
  2. System board read or write of feature card
  3. Interrupt acknowledge

- ID/Vector Register - An 8/16-bit register which initially contains a given feature card's identification code and later its 8-bit interrupt vector. The register can be written only by the 80186 and read only by the system board.

- Page Register - The Mid-Range Chip Selects of the 80186 are programmed to provide a 128 kilobyte (17-bit) address that is used when accessing the 3B2 Computer's main memory. The 7-bit page register is used to extend the address to 24 bits. The upper two bits (6 and 5) of the register must be zero, effectively yielding 32 pages with 128 kilobytes per page.

- Control and Status Register - PCSR is an 8-bit register that is byte readable by the 80186. Six of the bits are used by the CIO hardware and firmware.

- Local RAM Support - The Lower Memory Chip Select (LCS) of the 80186 is used to access the on-board RAM. The address space is 256 kilobytes, but the CIO does not specify how much or what kind of memory is actually used.

- Local Rom Support - The Upper Memory Chip Select (UCS) of the 80186 is used to access the on-board ROM. CIO provides two 28-pin sockets for EPROM or ROM chips. Using the 27128 chips this will hold up to 32 kilobytes. The total reserved ROM space is 64 kilobytes.

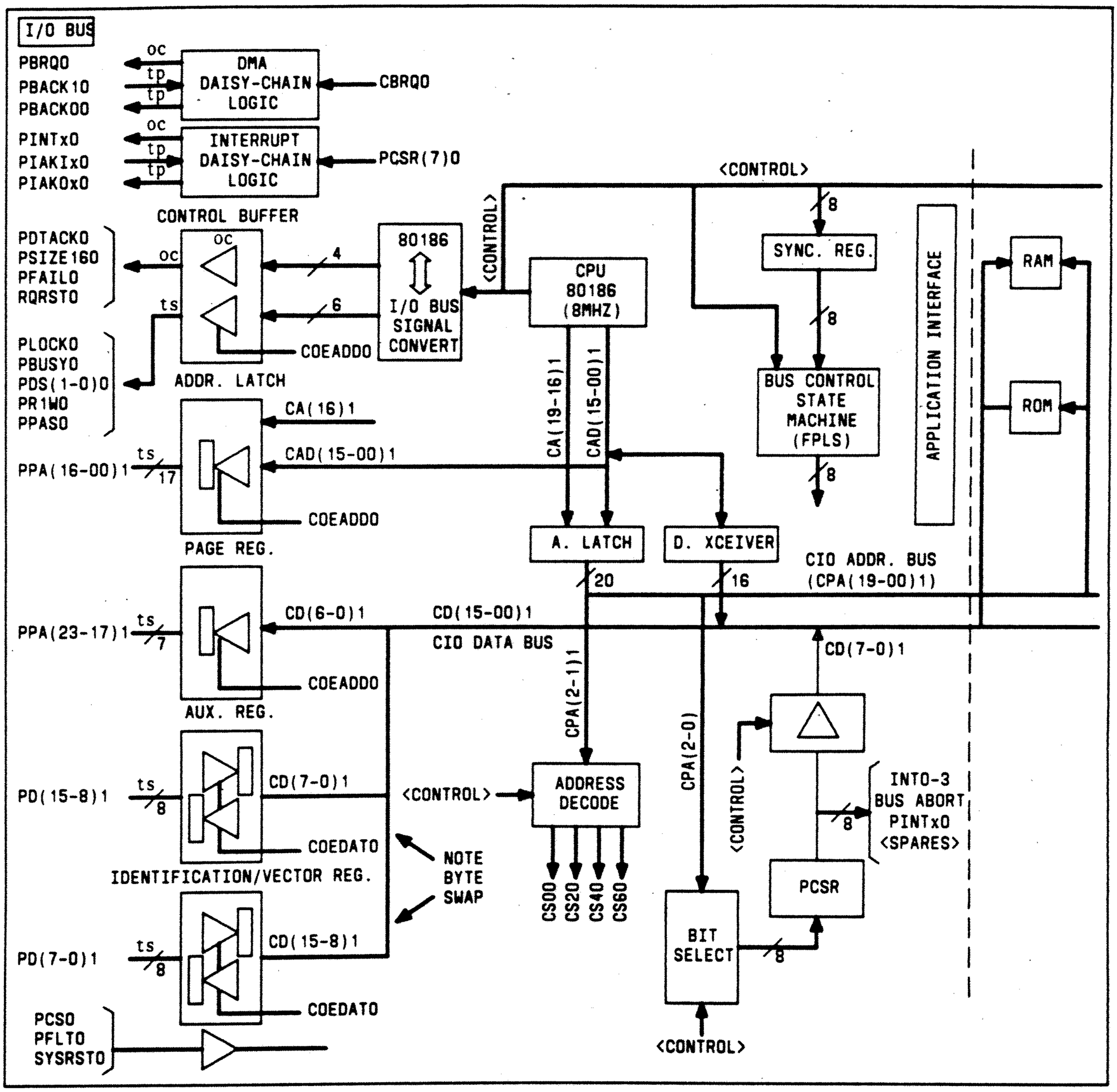
- Misc. Support Logic - This includes the address latches to demultiplex the 80186 address/data lines and the I/O Bus daisy-chain circuits.

CIO SUBCOMPONENTS

- CPU (80186)
- I/O Bus Control Logic
- I/O Bus Buffers/Latches
- ID/Vector Register
- Page Register
- Control and Status Register
- Local RAM Support
- Local Rom Support
- Misc. Support Logic

CIO BLOCK DIAGRAM

CIO BLOCK DIAGRAM



UNIT 8

APPENDIX

To Order by Telephone:

1-800-432-6600 (toll-free within the Continental United States)

1-317-352-8556 (Outside the Continental United States)

. AT&T 3B2/300 Computer Application Design Guide
Order Code: 305-496 Price: $30

This document presents a set of requirements, guidelines and templates for the application developer so that his/her software can be installed and uninstalled using the simple system administration commands.

. AT&T 3B5 Computer Application Design Guide
Order Code: 305-131 Price: $35

. AT&T 3B2/300 Computer CRASH Analysis Guide
Order Code: 305-491 Price: $30

CRASH is an interactive tool for examining the UNIX* system memory images. One can examine either the image of the running system or the image saved after a system PANIC.

. AT&T 3B2 and 3B5 Computer Driver Design Guide
Order Code: 305-495 Price: $35

This guide provides the necessary information to write UNIX System V device drivers for the AT&T 3B2/300 and 3B5 computers. It assumes that the reader has an understanding of UNIX Operating System Internals.

. AT&T 3B2 Computer Error Message Manual
Order Code: 305-493 Price: $30

This manual identifies and explains those messages displayed on a terminal when a problem occurs in the AT&T 3B2/300 computer.

. AT&T 3B2 Computer Feature Card Interface Design Manual
Order Code: 305-492 Price: $50

This document contains design information used for the 3B2 Computer INPUT/OUTPUT bus. This book is written for hardware/software computer circuit designers. It is assumed that the reader has a good understanding of digital circuits, software and microprocessors.

. AT&T 3B2 Computer Off-Line Diagnostics Manual
Order Code: 305-494 Price: $30

This manual documents the diagnostic phase and test descriptions for all diagnostics that run on the 3B2 computer. It is a reference for all development and maintenance personnel who use the 3B2 computer.

3B2 ID CODE and NAME REQUEST

To assure that device ID codes and names are used that uniquely identify 3B2 devices and subdevices, as well as UNIX drivers, the following information must be submitted to the 3B2 ID Code Administrator.

Company Name: _____

Address: _____

_____

Contact: _____

Phone #: _____

AT&T Contact: _____

Phone #: _____

Product Description:

Preferrences:

Equipment Device Table name: _____

Driver Initials: _____

Package Name: _____

Package Abbreviation: _____

The administrator will supply ID codes for hardware devices and check names for uniqueness. Please submit the completed form to:

AT&T Information Systems
4513 Western Ave.
Lisle, Il 60532
Department Chief
Dept. T2CU030250

APPENDIX A

I/O BUS PINOUT

APPENDIX A - I/O BUS PINOUT

I/O Bus Connector (86 pin)				
Pin Number	Signal		Pin Number	Signal
1	V12P		2	PINT[2]0
3	V12N		4	PINT[1]0
5	PBACKI0		6	PINT[0]0
7	PCS0		8	RQRST0
9	GRD		10	SYSRST0
11	VBKUP		12	PFAIL0
13	PIAKO[2]0		14	PFLT0
15	PIAKI[2]0		16	GRD
17	VCC		18	PDTACK0
19	PIAKI[1]0		20	PDS[1]0
21	PIAKO[1]0		22	PD[00]1
23	PIAKI[0]0		24	PD[02]1
25	GRD		26	PD[03]1
27	PBUSY0		28	PD[05]1
29	PIAKO[0]0		30	PD[07]1
31	PSIZE160		32	GRD
33	PDS[0]0		34	PD[08]1
35	PD[01]1		36	PD[10]1
37	GRD		38	PD[12]1
39	PD[04]1		40	VCC
41	GRD		42	PD[13]1
43	PD[06]1		44	PD[15]1
45	PD[09]1		46	PBRQ0
47	PD[11]1		48	GRD
49	GRD		50	PPAS0

APPENDIX A - I/O BUS PINOUT

I/O Bus Connector (86 pin)				
Pin Number	Signal		Pin Number	Signal
51	PD[14]1		52	PLOCK0
53	PBACKO0		54	PPA[00]1
55	PR1W0		56	GRD
57	GRD		58	PPA[02]1
59	PPA[01]1		60	PPA[04]1
61	PPA[03]1		62	PPA[05]1
63	PPA[06]1		64	VCC
65	GRD		66	PPA[07]1
67	PPA[09]1		68	PPA[08]1
69	PPA[10]1		70	PPA[11]1
71	PPA[12]1		72	GRD
73	GRD		74	PPA[13]1
75	PPA[15]1		76	PPA[14]1
77	PPA[17]1		78	PPA[16]1
79	PPA[20]1		80	GRD
81	GRD		82	PPA[18]1
83	PPA[23]1		84	PPA[19]1
85	PPA[21]1		86	PPA[22]1

APPENDIX A - I/O BUS PINOUT

Feature Card Via Hole numbering
(component side)

PPA221	86	□	□	85	PPA211
PPA231	83	□	□	84	PPA191
PPA181	82	□	□	81	GRD
PPA201	79	□	□	80	GRD
PPA161	78	□	□	77	PPA171
PPA151	75	□	□	76	PPA141
PPA131	74	□	□	73	GRD
PPA121	71	□	□	72	GRD
PPA111	70	□	□	69	PPA101
PPA091	67	□	□	68	PPA081
PPA071	66	□	□	65	GRD
PPA061	63	□	□	64	VCC
PPA051	62	□	□	61	PPA031
PPA011	59	□	□	60	PPA041
PPA021	58	□	□	57	GRD
PR1W0	55	□	□	56	GRD
PPA001	54	□	□	53	PBACKO0
PD141	51	□	□	52	PLOCK0
PPAS0	50	□	□	49	GRD
PD111	47	□	□	48	GRD
PBRQ0	46	□	□	45	PD091
PD061	43	□	□	44	PD151
PD131	42	□	□	41	GRD
PD041	39	□	□	40	VCC
PD121	38	□	□	37	GRD
PD011	35	□	□	36	PD101
PD081	34	□	□	33	PDS00
PSIZE160	31	□	□	32	GRD
PD071	30	□	□	29	PIAKO00
PBUSY0	27	□	□	28	PD051
PD031	26	□	□	25	GRD
PIAKI00	23	□	□	24	PD021
PD001	22	□	□	21	PIAKO10
PIAKI10	19	□	□	20	PDS10
PDTACK0	18	□	□	17	VCC
PIAKI20	15	□	□	16	GRD
PFLT0	14	□	□	13	PIAKO20
VBKUP	11	□	□	12	PFAIL0
SYSRST0	10	□	□	09	GRD
PCS0	07	□	□	08	RQRST0
PINT00	06	□	□	05	PBACKI0
V12N	03	□	□	04	PINT10
PINT20	02	□	□	01	V12P

APPENDIX B

SELF-CONFIG COMMANDS

APPENDIX B - SELF-CONFIG COMMANDS

This appendix contains additional information on some of the self configuration administrative commands useful during driver development. These commands should not be called directly as a part of a driver installation procedure. The driver developer should understand their functions.

mkboot

The **mkboot** utility prepares an object file for use by the boot program. The object file is either a configurable module or an unresolved UNIX kernel. Each module object file named must correspond to an entry in the master file. Correspondence is established by matching the object file name stripped of any optional path prefix or ".o" suffix. The alphabetic case of the resulting name is immaterial. A UNIX kernel object file is identified with a command line option, and the master file entry is always **kernel**.

The master file is read and the configuration information associated with each object file is extracted. Tunable parameters may be assigned values either from the command line, or from additional parameter files identified with a command line option. For each object file, a new file is created containing this configuration information. The new kernel object file is written to the current directory, and is named **KERNEL**. New object files for modules are written to the /boot directory and are given the name (in capital letters) of the corresponding master file entry.

mkunix

The **mkunix** utility will create a bootable kernel namelist file (also termed the **absolute** boot file) from the current contents of memory; this file will be named **a.out** and will be written to the current directory by default. This file contains the UNIX kernel object file, and all drivers and modules which were loaded. Typically, **mkunix** would be run following an auto-configuring boot with a new system configuration.

APPENDIX B - SELF-CONFIG COMMANDS

The resulting absolute boot file must be used as the namelist file for **ps, crash,** etc. In addition, this file may be booted directly, bypassing the self configuration feature process.

The unresolved kernel object file used in the boot operation must be available at the time **mkunix** is run. This is the path name specified as the **BOOT** program in the /etc/system file. This file is read to obtain the section names and the symbol table for the basic kernel.

APPENDIX C

MASTER FILE

APPENDIX C - MASTER FILE

The master configuration database is a collection of files. Each file contains configuration information for a device or module that may be included in the system. A file is named with the module name to which it applies. This collection of files is maintained in a directory called /etc/master.d. Each individual file has an identical format. For convenience, this collection of files will be referred to as the master file, as though it were a single file. This will allow a reference to the **master** file to be understood to mean the **individual file in the master.d directory that corresponds to the name of a device or module**. This file is used by the **mkboot** program to obtain device information to generate the device driver and configurable module files. It is also used by the **sysdev** program to obtain the names of supported devices. **Master** consists of two parts; they are separated by a line with a dollar sign($) in column 1. Part 1 contains device information for both hardware and software devices, and loadable modules. Part 2 contains parameter declarations used in part 1. Any line with an asterisk (*) in column 1 is treated as a comment.

Hardware devices, software drivers and loadable modules are defined with a line containing the following information. Field 1 must begin in the left most position on the line. Fields are separated by white space(tab or blank).

FLAG:	Element Characteristics:
o	Specify only once
r	Required device
b	Block device
c	Character device
a	Generate segment descriptor array
t	Initialize cdevsw[].d_ttys
s	Software driver
x	Not a driver; a loadable module
number	The first interrupt vector for an integral device

APPENDIX C - MASTER FILE

#VEC:	Number of interrupt vectors required by a hardware device; "-" if none.
PREFIX:	Handler prefix (4 chars. maximum).
SOFT:	Software driver external major number; "-" if not a software driver.
#DEV:	Number of sub-devices per device; "-" if none.
IPL:	Interrupt priority level of the device; "-" if none.
DEPENDENCIES/ VARIABLES	Dependency list ; this is a comma separated list of other drivers or modules that must be present in the configuration if this module is to be included.

For each module, two classes of information are required by **mkboot**: external routine references and variable definitions. Routine and variable definition lines begin with white space and immediately follow the initial module specification line. These lines are free form thus they may be continued arbitrarily between non-blank tokens as long as first character of a line is white space.

If UNIX system kernel or other dependent module contains external references to a module, but the module is not configured, then these external references would be undefined. Therefore, the **routine reference** lines are used to provide the information necessary to generate appropriate dummy functions at boot time when the driver is not loaded.

Routine references are defined as:

Field 1:	Routine_name()
Field 2:	The routine type; one of
{}	Routine_name(){}
{nosys}	Routine_name(){return nosys();}
{nodev}	Routine_name(){return nodev();}
{false}	Routine_name(){return 0;}
{true}	Routine_name(){return 1;}

APPENDIX C - MASTER FILE

Variable definition lines are used to generate all variables required by the module. The variable generated may be of arbitrary size, be initialized or not, or be arrays containing an arbitrary number of elements.

Variables references are defined as follows: **Routine references** are defined as:

Field 1:	Variable name
Field 2:	[expr] - optional field used to indicate array size
Field 3:	(length) - required field indicating the size of the variable
Field 4:	={ expr, ... } - optional field used to initialize individual elements of a variable

The **length** field is mandatory. It is an arbitrary sequence of length specifiers, each of which may be one of the following:

%i	An integer
%l	A long integer
%s	A short integer
%c	A single character
%number	A field which is **number** bytes long
%number c	A character string which is **number** bytes long

For example, the length field

(%8c %l %0x58 %l %c %c)

could be used to identify a variable consisting of a character string 8 bytes long, a long integer, a 0x58 byte structure of any type, another long integer, and two characters. Appropriate alignment and the variable length is rounded up to the next word boundary during processing.

APPENDIX C - MASTER FILE

The expression for the optional array size and initialization are infix expressions consisting of the usual operators for addition, subtraction, multiplication and division: +, -, *, and /. Multiplication and division have the higher precedence, but parentheses may be used to override the default order. The builtin functions **min** and **max** accept a pair of expressions, and return the mixture of the following:

&name	Address of name where **name** is any symbol defined by the kernel, any module loaded or any variable definition line of any module loaded.
#name	Size of name where **name** is any variable defined by a variable definition for any module loaded; the size is that of the individual variable -- not the size of an entire array.
#C	Number of controllers present; this number is determined by the EDT for hardware devices, or by the number provided in the **system** file for non-hardware drivers or modules.
#C(name)	Number of controllers present for the module **name**; this number is determined by the EDT for hardware devices, or by the number provided in the **system** file for non- hardware drivers or modules.
#D	Number of devices per controller taken directly from the current **master** file entry.
#D(name)	Number of devices per controller taken directly from the **master** file entry for the module **name**.
#M	The internal major number assigned to the current module if it is a device driver; zero if this module is not a device driver.
#M(name)	The internal major number assigned to the module **name** if it is a device driver; zero if that module is not a device driver.

APPENDIX C - MASTER FILE

name	Value of a parameter as defined in the second part of **master**.
number	Arbitrary number (octal, decimal, or hex allowed).
string	A character string enclosed within double quote (all of the character string conventions supported by the C language are allowed); this operand has a value which is the address-of a character array containing the specified string.

When initializing a variable, one initialization expression should be provided for each %i, %l, %s, or %c of the length of the field. The only initializers allowed for a "%number c" are either a character string (the string may not be longer than **number**), or an explicit zero. Initialization expressions must be separated by commas. and variable initialization will proceed element by element. Note that %number specifications cannot be initialized, they are set to zero. Only the first element of an array can be initialized the other elements are set to zero. If there are more initializers than size specifications, it is an error and execution of the **mkboot** program will be aborted. If there are fewer initializations than size specifications, zeros will be used to pad the variable.

```
={ "V2.11",#c*#d, max(10,#D,) #C(OTHER), #M(OTHER) }
```

would be a possible initialization of the variable whose length field was given in the preceding example.

APPENDIX C - MASTER FILE

Parameters declarations may be used to refer to a value symbolically. Values can be associated with identifiers and these identifiers may be used in the **variable definition** lines.

Parameters are defined as follows:

Field 1:	Identifier (8 characters maximum)
Field 2:	=
Field 3:	Value the value may be a number (decimal, octal, or hex allowed), or a string.

APPENDIX D

SYSTEM FILE

APPENDIX D - SYSTEM FILE

Detailed descriptions of the entries that are contained in /etc/system file.

Lines may appear in any order. Comment lines must begin with an asterisk. Blank lines or comment lines may be inserted at any point. Entries for EXCLUDE and INCLUDE are cumulative. For all other entries, the last line to appear in the file is used -- any earlier entries are ignored. Since the parser is case sensitive, all earlier upper case strings must be entered exactly as shown.

BOOT	Pathname -- The pathname specifies the object file to be booted; if the object file is fully resolved (such as that produced by the **mkunix** program) then no other lines in the **system** file have any effect.
EXCLUDE:	Name ... -- This identifies names in the EDT that are to be ignored -- this may either be because no driver exists for the specific EDT entry, or because the driver is not to be loaded for some other reason.
INCLUDE	Name [(number)] ... -- This line is necessary to identify software drivers or loadable modules from the /boot directory which are to be included in the load. It has no effect for hardware drivers. The optional "(number)" specifies the number (default of 1) of "devices" to be controlled by the driver. This number corresponds to the built-in variable **#C** which may be referred to by expressions in part one of the **master** file.

APPENDIX D - SYSTEM FILE

```
DUMPDEV:    {spec-dev-path name | DEV(major, minor)}
ROOTDEV:    {spec-dev-path name | DEV(major, minor)}
PIPEDEV:    {spec-dev-path name | DEV(major, minor)}
SWAPDEV:    {spec-dev-path name | DEV(major, minor)}
            swplo  nswap
```

These lines identify the system device to be used for writing a crash dump, the device containing the root file system, the device to be used for pipe space, and the device to be used for swap space (with the beginning block number for swap space **swplo** and the number of swap blocks available **nswap**). These are normally used only in the 3B5 computer; on the 3B2 computer, this information is derived from the disk's volume table of contents (VTOC). The device may be specified in either of two ways. A pathname of a special device file may be provided -- the major and minor numbers are obtained from the **inode**. An alternative form is allowed in which the major and minor numbers are specified explicitly.

NOTE : On the 3B2 Computer these items are taken care of by VTOC unless the system is booted manually and you are using system entries. However, these can be specified and put into the system file at any time.

APPENDIX E

EDITTBL COMMAND

APPENDIX E - EDITTBL COMMAND

edittbl -- to edit /dgn/edt_data file

edittbl [-d] [-s] [-g] [-i] [-l] [-r] [-t] [file]

Edittbl is a user level command that permits changes to **edt_data**, the file in the root file system that the diagnostic monitor, DGMON, reads during self configuration to get the device and subdevice look-up tables. This utility permits independent selection of device or subdevice tables, generation of either base table, new entry installation for either table, entry removal for the device table and entry listings for either or both tables.

Edittbl prints the option list if the command has no arguments. The arguments are:

-d This option selects the device look-up table for the utility's operation(s).

-s This option selects the subdevice look-up table for the utility's operation(s).

-g This option will generate the base look-up table entries for the selected look-up table(s). For the device table, these base entries are NULL, SBD, NI, and PORTS. For the subdevice table, they are NULL, FD5, HD10, and HD30.

-i This option specifies that new entries are added to the selected table. The ID codes for table entries and the input are compared; only new codes are installed. Formats for entries are described below. An EOF or "." is the end of data input.

APPENDIX E - EDITTBL COMMAND

-l This option specifies that the selected table(s) are listed.

-r This option specifies that entries are to be removed from the device look-up table. When removing subdevice look-up table entries from **/dgn/edt_data** conjunction with removing a device entry, this command will check the Equipped Device Table (EDT) to verify that no subdevices specified for removal are present. The ID codes of the table are compared to the input and entries are removed for matches. The format is identical to that for the **-i** option and is listed above. An EOF or "." is the end the data input.

-t This option suppresses the program headings and user prompts; warnings and errors are not affected. this option is primarily useful in installation and removal scripts.

file The user may specify a target path name for the utilities. If none is specified, **edt_data** is the default.

INPUT FORMAT

Data for installation/removal are entered as hex format numbers or characters strings, one line for each table entry. The data fields must be supplied in the sequence described.

APPENDIX E - EDITTBL COMMAND

Devices

ID_code This field is a number between 0x0 and 0xffff that a device uses to identify itself. ID codes are administered by AT&T Technologies.

dev_name This field is a character string (maximum of 9 characters) that holds the user-recognizable name for a device. Device names are administered by AT&T Technologies. This string is also the file name that DGMON loads to diagnose a device.

rq_size This is a number between 0x0 and 0xff for the count of entries in a device's job request queue.

cq_size This is a number between 0x0 and 0xff for the count of entries in a device's job completion queue.

boot_dev This field determines whether a device may be used to boot programs. A "1" means that it is bootable, a "0" that it is not.

word_size This field shows the word size of a device's I/O bus. A "1" is used for devices with 16 bit bus word; a "0" is used for devices with an 8 bit bus word.

brd_size This field specifies the I/O connector slots that a device requires. A "1" indicates that two slots are needed, while a "0"means that one is required.

APPENDIX E - EDITTBL COMMAND

smrt_brd This field determines whether a device is intelligent, i.e., requires downloaded code for normal operation or supports subdevices. A "1" indicates an intelligent device, while a "0" specifies a "dumb"device.

cons_cap This field shows whether a device can support the system console terminal. A "1" is used for devices that can, a "0" for those that cannot.

cons_file This field shows whether a device requires pump code to provide a system console interface. A "1" in this field means that the board cannot support the console interface without extra code. This field may have the "1" value only when the **cons_cap** field does also. A "0" in this field means that the device can support s system console terminal with PROM-based code when **cons_cap** has the value "1". This field must have a "0" value when **cons_cap** is "0".

Subdevices

ID_code This field is a number between 0x0 and 0xffff for the code that identifies a subdevice. Subdevice ID codes are administered by AT&T Technologies.

APPENDIX E - EDITTBL COMMAND

subdev_name This field is a character string (maximum of 9 characters) for a subdevice's name. Subdevice names are upper case and are administered by AT&T Technologies.

dev_name This field is a string (maximum of 9 characters) for the device name to which a subdevice is associated. If a device table entry is to be removed, associated subdevice table entries may also be removed in a separate program call. The device name is necessary for a Equipped Device Table (EDT) check that will verify that a subdevice table entry is needed only for a device entry that is to be removed.

EXAMPLES

Generate and list the base entries for both the devices and subdevices tables, saving the results in **edt_data**. You must be superuser to execute this command.

```
edittbl -g -l -s -d
```

Install subdevice entries with new ID codes from the file **subdev.in** into the existing file **edt_data**.

```
edittbl -i -s < subdev.in
```

List the device table entries found in an existing copy of the file that DGMON loads, the ROOT file system's **edt_data** file.

```
edittbl -l -d /dgn/edt_data
```

APPENDIX F

ASCII CODE

APPENDIX F - ASCII CODE

(Octal)

000 nul	001 soh	002 stx	003 etx	004 eot	005 enq	006 ack	007 bel
010 bs	011 ht	012 nl	013 vt	014 np	015 cr	016 so	017 si
020 dle	021 dc1	022 dc2	023 dc3	024 dc4	025 nak	026 syn	027 etb
030 can	031 em	032 sub	033 esc	034 fs	035 gs	036 rs	037 us
040 sp	041 !	042 "	043 #	044 $	045 %	046 &	047 '
050 (	051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [	134 \	135]	136 ^	137 _
140 ‘	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174 |	175 }	176 ~	177 del

APPENDIX F - ASCII CODE

(Hex)

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0a	nl	0b	vt	0c	np	0d	cr	0e	so	0f	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	can	19	em	1a	sub	1b	esc	1c	fs	1d	gs	1e	rs	1f	us
20	sp	21	!	22	"	23	#	24	$	25	%	26	&	27	'
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	‘	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c	|	7d	}	7e	~	7f	del

APPENDIX G

HR1 FEATURE CARD

APPENDIX G - HR1 FEATURE CARD

HR1 CARD HARDWARE DESCRIPTION

The following is a brief description of hardware capabilities of the HR1 feature card.

Serial Port (Address 254) 0xFE

This 1200 baud serial link is connected to a terminal (other than the 3B2 console terminal) to be used as the board's diagnostic/user terminal.

Serial Port Read 0x05

Pressing any key on the user terminal causes that character to be echoed on the terminal. The character can be read by reading address offset 05 (base address for the HR1 slot plus the offset of 05). No dummy read is required for this operation.

Serial Port Write

A write to address offset fe will write the characters to serial port and display it on the user terminal.

Application Port (Address 255) 0x FF

A write to address offset ff will output the byte to the Application Output Port that is presently equipped with 8 LED's (socket 35).

A read (with dummy read preceding it) of address offset ff will read a byte from the Application Input Port that is presently equipped with switches (socket 36).

Vector Register 0x07

At address 07 there is a register that is used to hold the interrupt vector for the HR1 card.

APPENDIX G - HR1 FEATURE CARD

HR1 CARD HARDWARE DESCRIPTION (cont.)

Writing to the general purpose memory

The address space from offset 09 to 6f can be used as a general purpose RAM. Two limitation apply:

1. Only odd boundary bytes can be accessed.
2. Every read has to be preceded with a "dummy read" (every address read twice).

APPENDIX G - HR1 FEATURE CARD

lab1 COMMAND

(/usr/bin/lab1)

Read	r	address [length]<cr>
Read Continuous	rc	address [length]<cr>
Write	w	address [length]<cr> data
Write Continuous	wc	address [length]<cr> data
Single Read	S	
Double Read	s	(default)
Increment by Single Address	D	
Increment by Double Address	d	(default)
Quit	q	

With "continuous" the operation will repeat over and over until address 05 on the HR1 card contains a <cr> character. (Press the return key on the user terminal, followed by some other key to remove the return code from address 05.)

When control g is pressed on the user keyboard an interrupt signal will be sent to the 3B2.

NOTE: The address and length are input as **decimal** values.

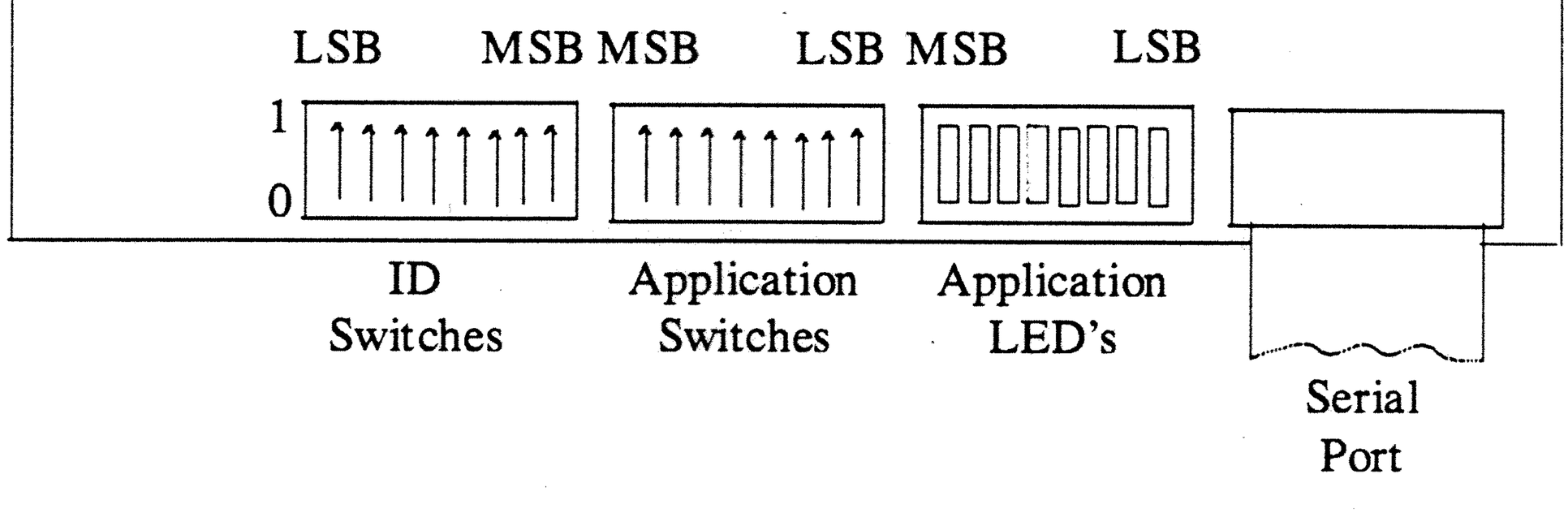
Length is optional. If it is not given, the length will be one.

The lab1 prompt character is: >

APPENDIX G - HR1 FEATURE CARD

HR1 FEATURE CARD

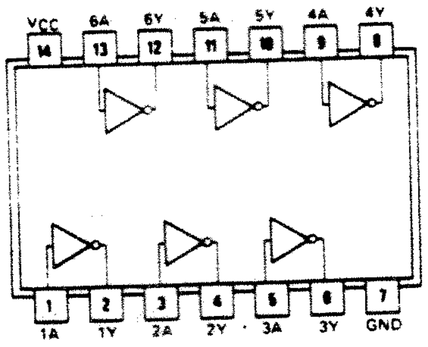
Switches and LED's



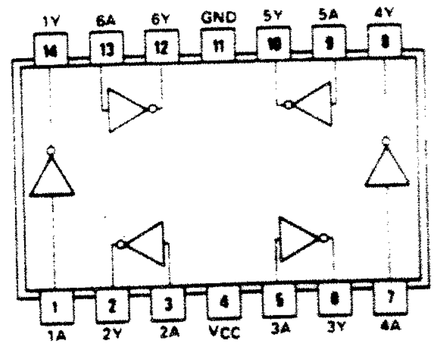
IC Chips Used in the HR1 Board

04
HEX INVERTERS

positive logic:
$Y = \overline{A}$



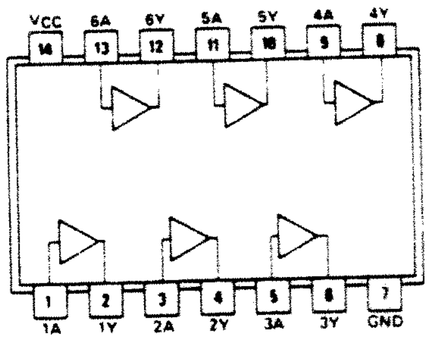
SN5404/SN7404(J, N)
SN54H04/SN74H04(J, N)
SN54L04/SN74L04(J, N)
SN54LS04/SN74LS04(J, N, W)
SN54S04/SN74S04(J, N, W)



SN5404/SN7404(W)
SN54H04/SN74H04(W)
SN54L04/SN74L04(T)

07
HEX BUFFERS/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS

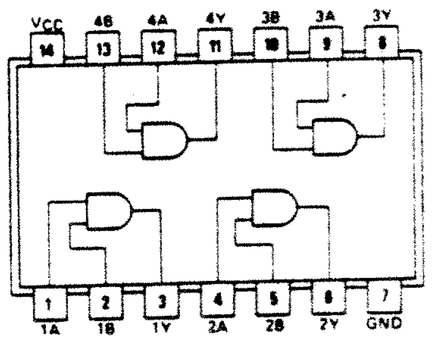
positive logic:
$Y = A$



SN5407/SN7407(J, N, W)

08
QUADRUPLE 2-INPUT POSITIVE-AND GATES

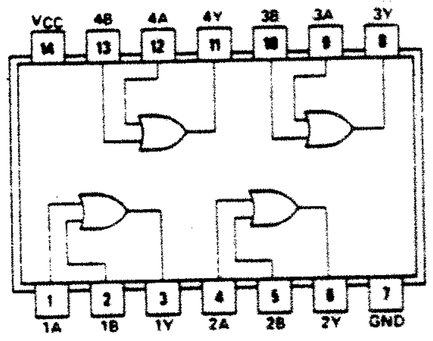
positive logic:
$Y = AB$



SN5408/SN7408(J, N, W)
SN54LS08/SN74LS08(J, N, W)

32
QUADRUPLE 2-INPUT POSITIVE-OR GATES

positive logic:
$Y = A+B$

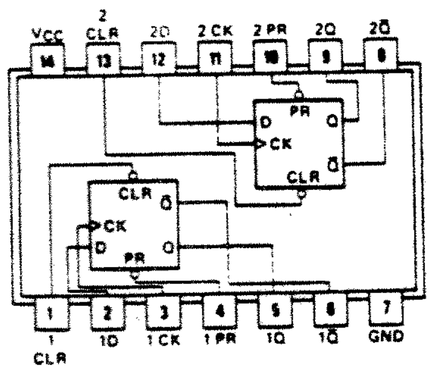


SN5432/SN7432(J, N, W)
SN54LS32/SN74LS32(J, N, W)

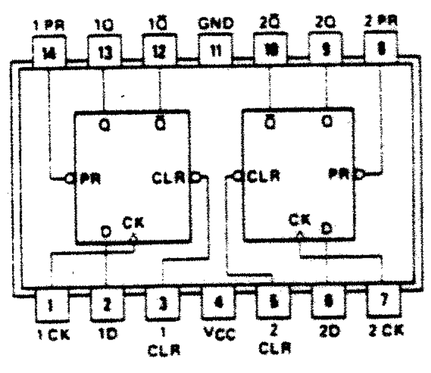
74
DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$



SN5474/SN7474(J, N)
SN54H74/SN74H74(J, N)
SN54L74/SN74L74(J, N)
SN54LS74/SN74LS74(J, N, W)
SN54S74/SN74S74(J, N, W)



SN5474/SN7474(W)
SN54H74/SN74H74(W)
SN54L74/SN74L74(T)

H = high level (steady state), L = low level (steady state), X = irrelevant
⎍ = high-level pulse; data inputs should be held constant while clock is high; data is transferred to output on the falling edge of the pulse.
↑ = transition from low to high level, ↓ = transition from high to low level
Q_0 = the level of Q before the indicated input conditions were established.
TOGGLE: Each output changes to the complement of its previous level on each active transition (pulse) of the clock.
*This configuration is nonstable; that is, it will not persist when preset and clear inputs return to their inactive (high) level.

DECODER/DEMULTIPLEXERS — 54/74154, LS154

1-of-16 Decoder/Demultiplexer

- **16-line demultiplexing capability**
- **Mutually exclusive outputs**
- **2-input enable gate for strobing or expansion**

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74154	21ns	34mA
74LS154	15ns	9mA

DESCRIPTION

The '154 decoder accepts four active HIGH binary address inputs and provides 16 mutually exclusive active LOW outputs. The 2-input enable gate can be used to strobe the decoder to eliminate the normal decoding "glitches" on the outputs, or it can be used for expansion of the decoder. The enable gate has two AND'ed inputs which must be LOW to enable the outputs.

The '154 can be used as a 1-of-16 demultiplexer by using one of the enable inputs as the multiplexed data input. When the other enable is LOW, the addressed output will follow the state of the applied data.

ORDERING CODE

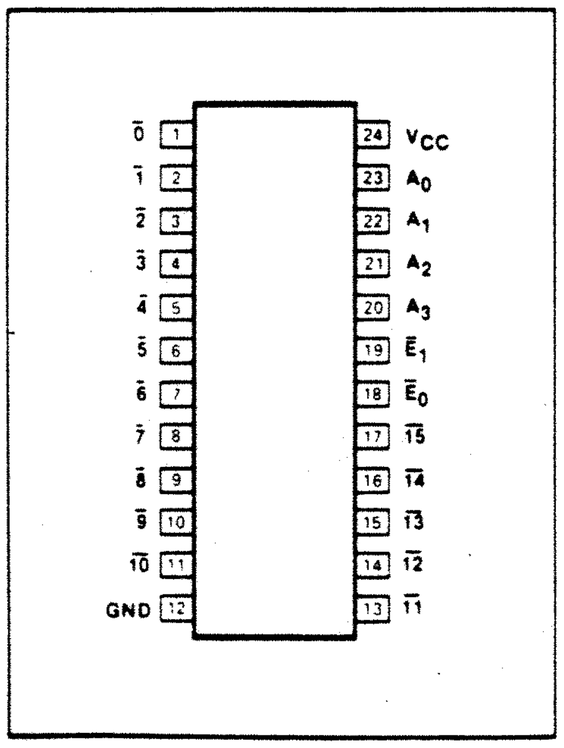
PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ± 5%; T_A = 0°C to +70°C	MILITARY RANGES V_{CC} = 5V ± 10%; T_A = −55°C to +125°C
Plastic DIP	N74154N • N74LS154N	
Ceramic DIP		S54154F • S54LS154F
Flatpack		S54154W • S54LS154W

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

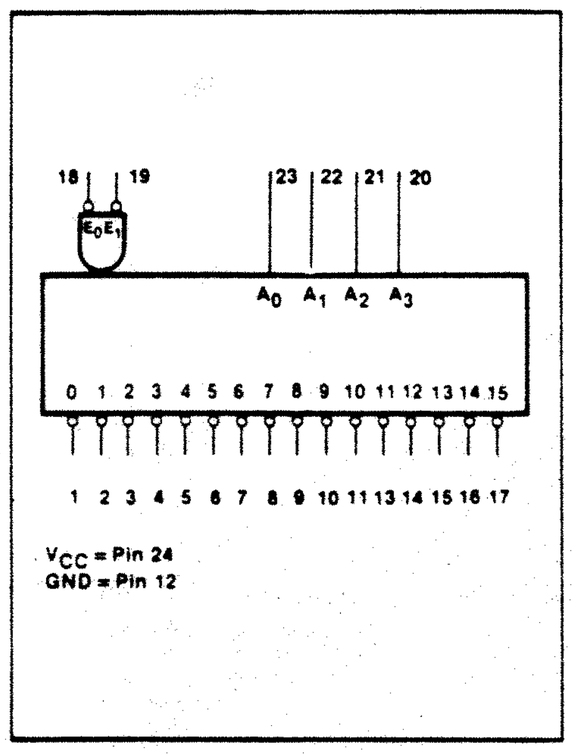
PINS	DESCRIPTION	54/74	54/74LS
All	Inputs	1ul	1LSul
All	Ouputs	10ul	10LSul

NOTE
Where a 54/74 unit load (ul) is understood to be 40µA I_{IH} and − 1.6mA I_{IL}, and a 54/74LS unit load (LSul) is 20µA I_{IH} and − 0.4mA I_{IL}.

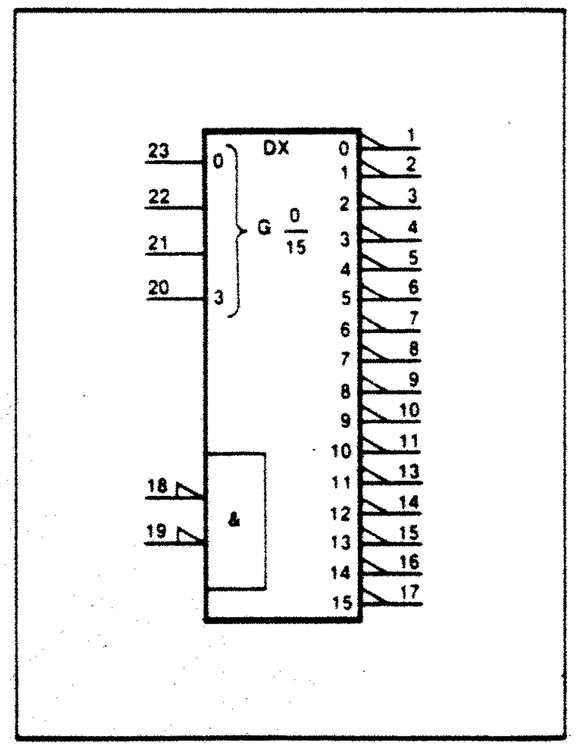
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



BUFFERS

54/74LS244, S244

Octal Buffers (3-State)

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74LS244	12ns	25mA
74S244	6ns	112mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ± 5%; T_A = 0°C to + 70°C	MILITARY RANGES V_{CC} = 5V ± 10%; T_A = -55°C to + 125°C
Plastic DIP	N74LS244N • N74S244N	
Plastic SO	74LS244D	
Ceramic DIP		S54LS244F • S54S244F
LLCC		S54LS244G

FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{OE}_a$	I_a	$\overline{OE}_b$	I_b	Y_a	Y_b
L	L	L	L	L	L
L	H	L	H	H	H
H	X	H	X	(Z)	(Z)

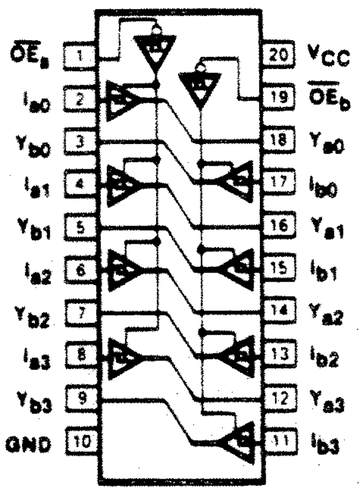
H = HIGH voltage level
L = LOW voltage level
X = Don't care
(Z) = HIGH impedance (off) state

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

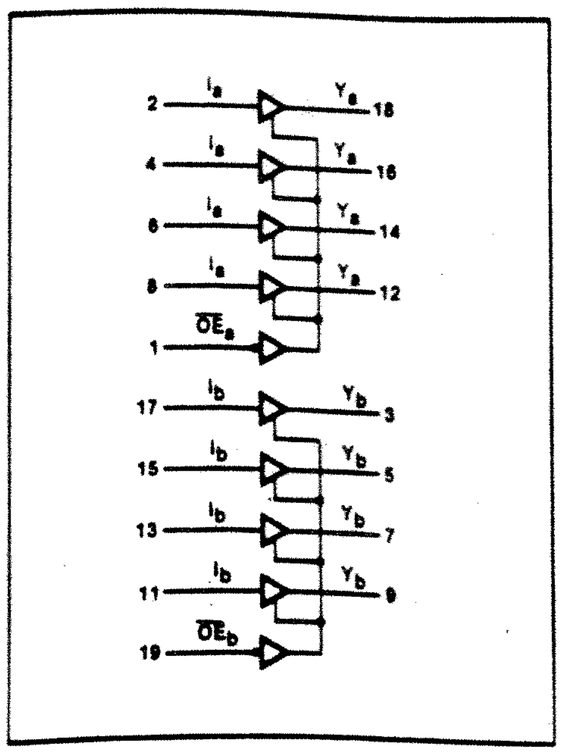
PINS	DESCRIPTION	54/74S	54/74LS
All	Inputs	1Sul	1LSul
All	Outputs	24Sul	30LSul

NOTE
A 54/74S unit load (Sul) is 50μA I_{IH} and − 2.0mA I_{IL}, and a 54/74LS unit load (LSul) is 20μA I_{IH} and − 0.4mA I_{IL}.

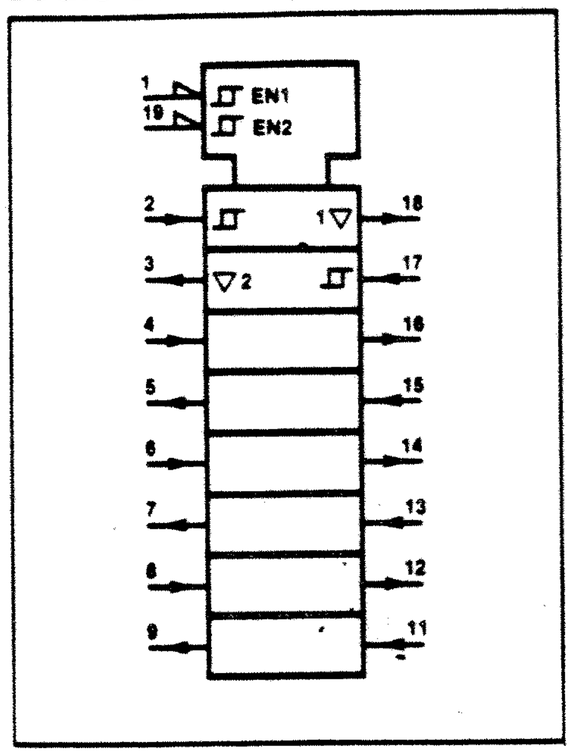
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



LATCHES/FLIP-FLOPS

54/74LS373, 54/74LS374, S373, S374

'373 Octal Transparent Latch With 3-State Outputs
'374 Octal D Flip-Flop With 3-State Outputs

- 8-bit transparent latch — '373
- 8-bit positive, edge-triggered register — '374
- 3-State output buffers
- Common 3-State Output Enable
- Independent register and 3-State buffer operation

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74LS373	19ns	24mA
74S373	10ns	105mA
74LS374	19ns	27mA
74S374	8ns	116mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ± 5%; T_A = 0°C to + 70°C	MILITARY RANGES V_{CC} = 5V ± 10%; T_A = −55°C to + 125°C
Plastic DIP	N74LS373N • N74S373N N74LS374N • N74S374N	
Plastic SO	N74LS373D • N74S373D N74LS374D • N74S374D	
Ceramic DIP		S54LS373F • S54S373F S54LS374F • S54S374F
LLCC		S54S374G S54LS373G • S54LS374G

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

PINS	DESCRIPTION	54/74S	54/74LS
All	Inputs	1SuI	1LSuI
All	Outputs	10SuI	30LSuI

NOTE
Where a 54/74S unit load (SuI) is 50μA I_{IH} and − 2.0mA I_{IL}, and a 54/74LS unit load (LSuI) is 20μA I_{IH} and − 0.4mA I_{IL}

DESCRIPTION

The '373 is an octal transparent latch coupled to eight 3-State output buffers. The two sections of the device are controlled independently by Latch Enable (E) and Output Enable ($\overline{OE}$) control gates.

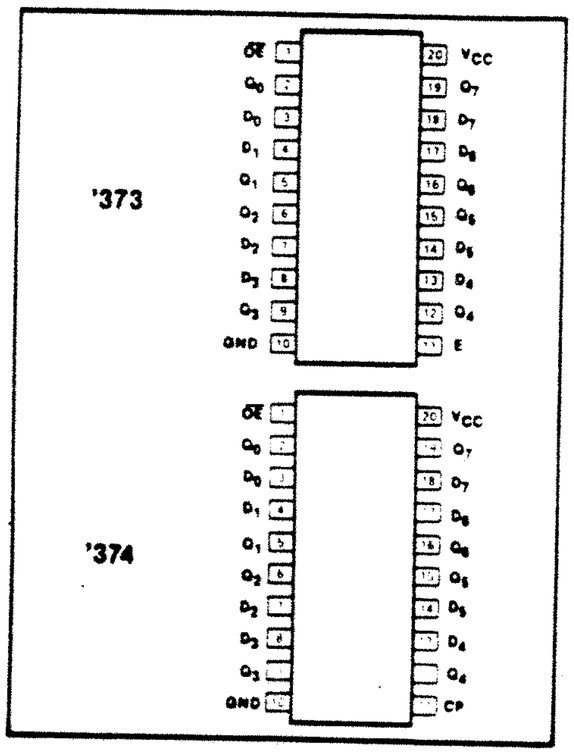
The data on the D inputs are transferred to the latch outputs when the Latch Enable (E) input is HIGH. The latch remains transparent to the data inputs while E is HIGH, and stores the data present one setup time before the HIGH-to-LOW enable transition. The enable gate has about 400mV of hysteresis built in to help minimize problems that signal and ground noise can cause on the latching operation.

The 3-State output buffers are designed to drive heavily loaded 3-State buses, MOS memories, or MOS microprocessors. The active LOW Output Enable ($\overline{OE}$) controls all eight 3-State buffers independent of the latch operation. When $\overline{OE}$ is LOW, the latched or transparent data appears at the outputs. When $\overline{OE}$ is HIGH, the outputs are in the HIGH impedance "off" state, which means they will neither drive nor load the bus.

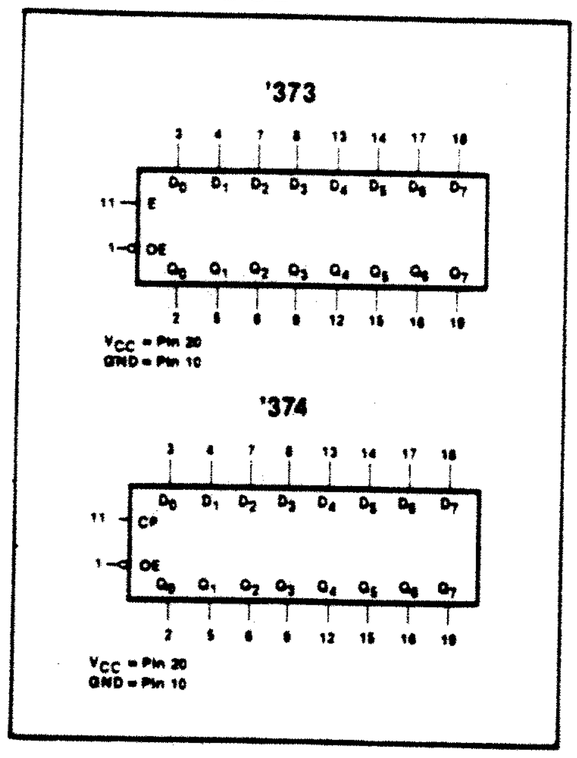
The '374 is an 8-bit, edge-triggered register coupled to eight 3-State output buffers. The two sections of the device are controlled independently by the Clock (CP) and Output Enable ($\overline{OE}$) control gates.

The register is fully edge triggered. The state of each D input, one setup time before the LOW-to-HIGH clock transition, is transferred to the corresponding flip-flop's Q output. The clock buffer has about 400mV of hysteresis built in to help minimize problems that signal and ground noise can cause on the clocking operation.

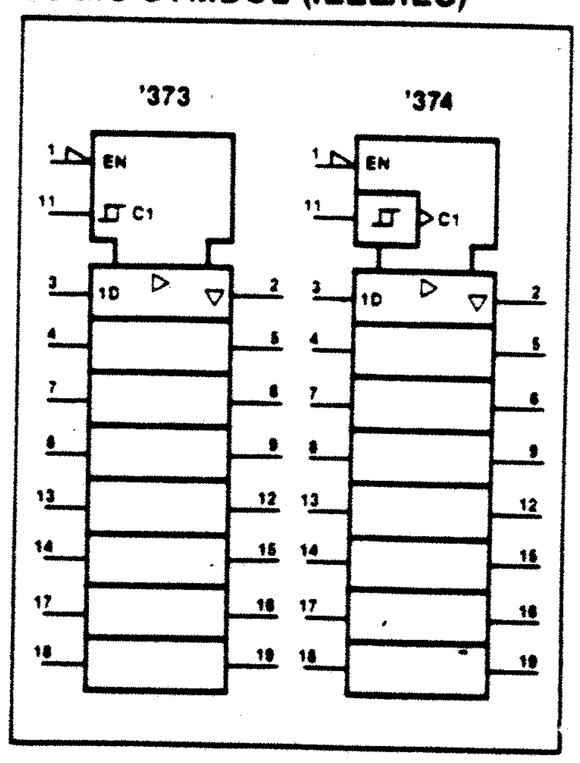
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



TRANSCEIVER

54/74LS645, 74LS645-1

Octal Bus Transceiver (3-State)

- **Octal bidirectional bus interface**
- **3-State buffer outputs**
- **PNP inputs for reduced loading**
- **Hysteresis on all Data inputs**
- **48mA sink capability ('LS645-1)**

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74LS645 & -1	10ns	58mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ± 5%; T_A = 0°C to +70°C	MILITARY RANGES V_{CC} = 5V ± 10%; T_A = −55°C to +125°C
Plastic DIP	N74LS645N N74LS645-1N	
Ceramic DIP		S54LS645F

DESCRIPTION

The 'LS645 is an octal transceiver featuring non-inverting 3-State bus compatible outputs in both send and receive directions. The outputs are all capable of sinking 24mA and sourcing up to 15mA, producing very good capacitive drive characteristics. In addition, the 74LS645-1 features a 48mA sink current capability. The device features a Chip Enable ($\overline{CE}$) input for easy cascading and a Send/Receive (S/R) input for direction control. All Data inputs have hysteresis built in to minimize ac noise effects.

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

PINS	DESCRIPTION	54/74LS & -1
All	Inputs	1LSul
All	Outputs	30LSul

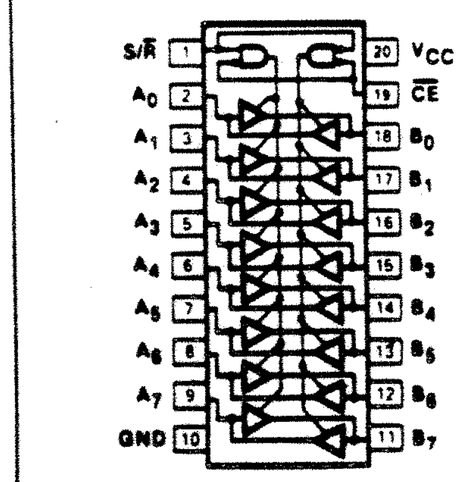
NOTE
A 54/74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

FUNCTION TABLE

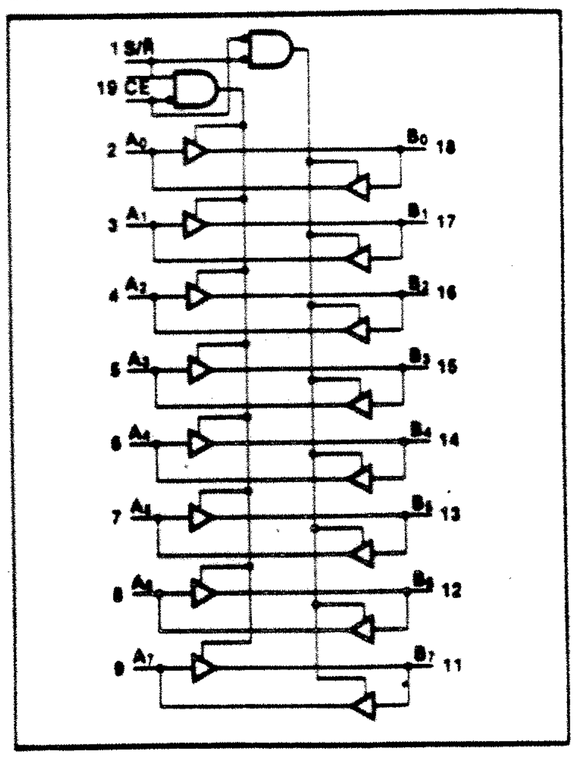
INPUTS		INPUTS/OUTPUTS	
$\overline{CE}$	S/$\overline{R}$	A_n	B_n
L	L	A = B	INPUTS
L	H	INPUTS	B = A
H	X	(Z)	(Z)

H = HIGH voltage level
L = LOW voltage level
X = Don't care
(Z) = HIGH impedance "off" state

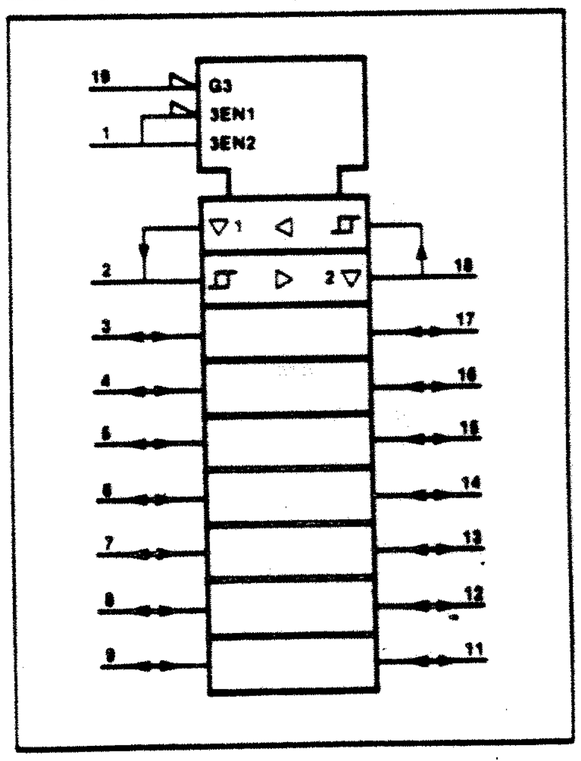
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



intel

8751H
8751H-11/8751H-10/8751H-8
SINGLE-COMPONENT 8-BIT MICROCOMPUTERS

- **8751H: 12 MHz Operation**
- **8751H-8: 8 MHz Operation**
- **Program Memory Security**
- **4K × 8 EPROM**
- **128 × 8 RAM**
- **32 I/O Lines (Four 8-Bit Ports)**
- **Two 16-Bit Timer/Counters**
- **Programmable Full-Duplex Serial Channel**
- **128K Accessible External Memory**
- **Boolean Processor**
- **4 μs Multiply and Divide**
- **218 User Bit-Addressable Locations**

The 8751H is a pin compatible EPROM version of the 8051AH. Intel's advanced +5 volt, depletion load, N-channel, HMOS technology allows the 8751H to remain fully compatible with its 8751/8751-8 predecessor in addition to incorporating a new program memory security feature. This allows the 8751H to be a full-speed MCS®-51 prototyping tool and provides for an effective single component solution for highly sensitive controller applications requiring code modification flexibility.

Specifically, the 8751H features: 4K byte program memory space; 32 I/O lines; two 16-bit timer/event counters; a 5-source; 2-level interrupt structure; a full duplex serial channel; a Boolean processor; and on-chip oscillator and clock circuitry. Standard TTL and most byte-oriented MCS-80 and MCS-85 peripherals can be used for I/O and memory expansion.

The 8751H is available in a hermetically sealed, ceramic, 40-lead dual in-line package which includes a window that allows for EPROM erasure when exposed to ultraviolet light (see Erasure Characteristics). During normal operation, ambient light may adversely affect the functionality of the chip. Therefore, applications which expose the 8751H to ambient light may require an opaque label over the window.

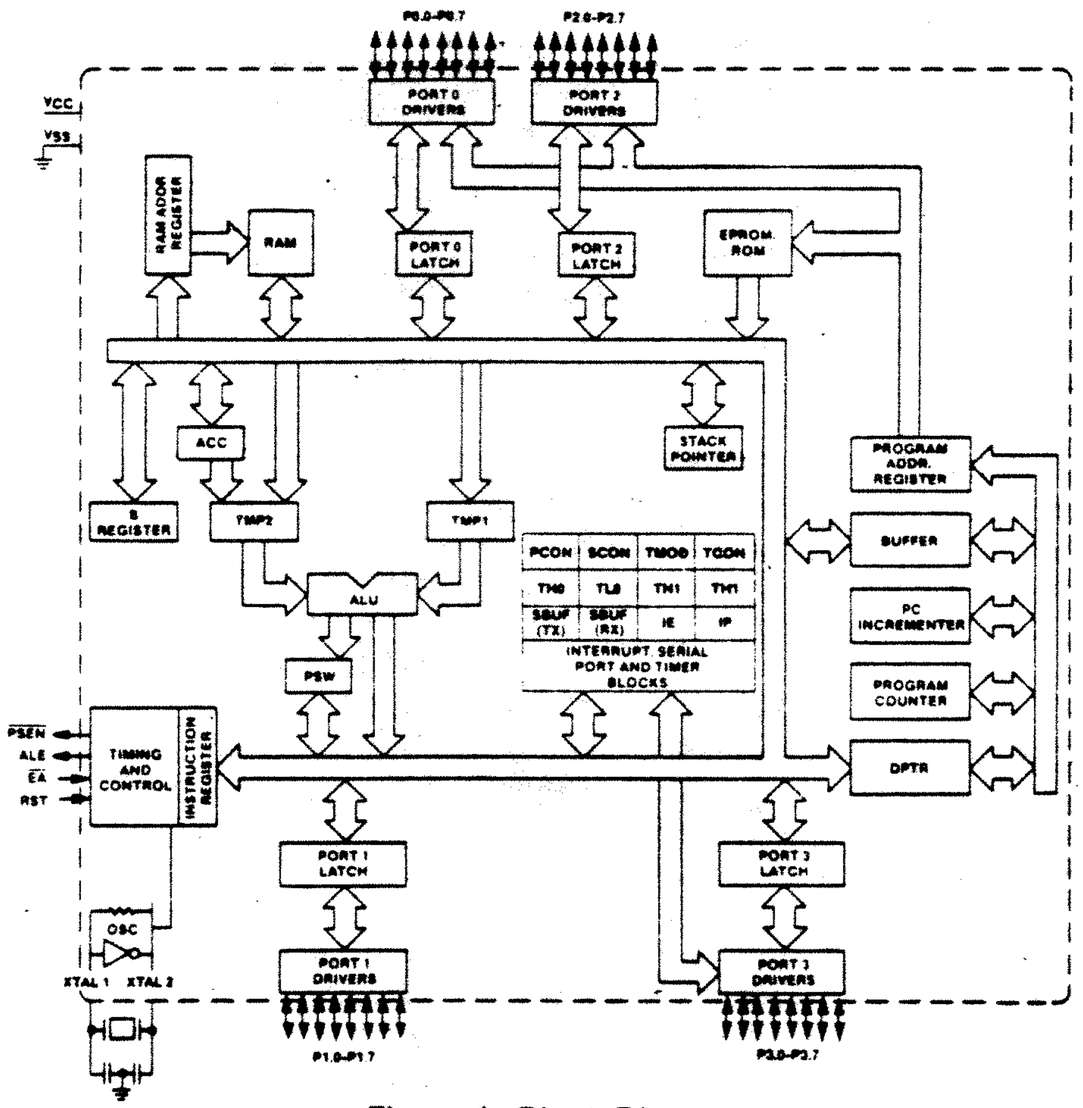


Figure 1. Block Diagram

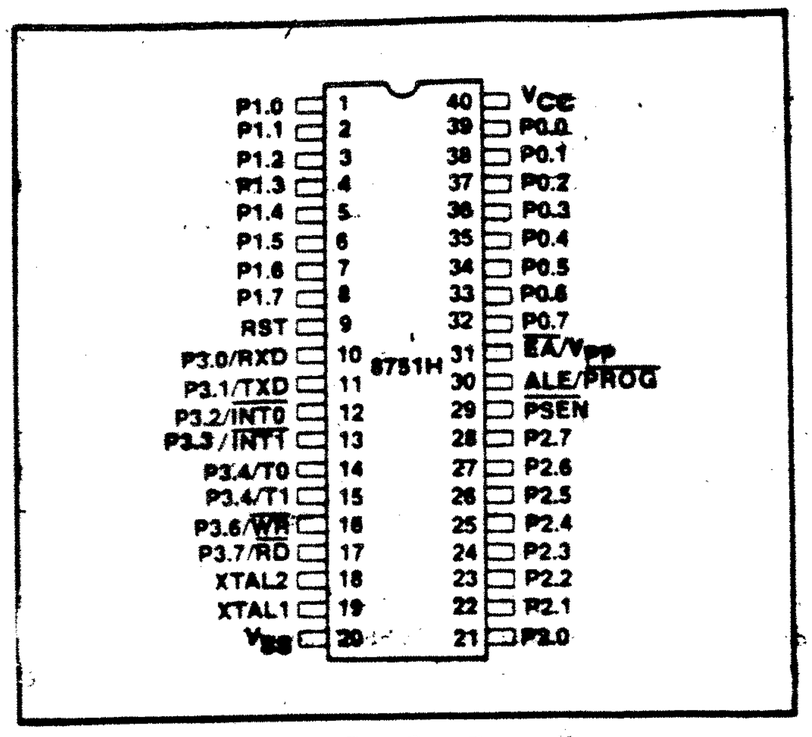


Figure 2. Pin Configuration

8751H PIN DESCRIPTIONS

V_{SS}
Circuit ground potential.

V_{CC}
Supply voltage during programming, verification, and normal operation.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus during accesses to external memory. It also receives the instruction bytes during EPROM programming, and outputs instruction bytes during program verification. (External pullups are required during program verification.) Port 0 can sink (and in bus operations can source) eight LS TTL inputs.

Port 1
Port 1 is an 8-bit bidirectional I/O port with internal pullups. It receives the low-order address byte during EPROM programming and program verification. Port 1 can sink/source four LS TTL inputs.

Port 2
Port 2 is an 8-bit bidirectional I/O port with internal pullups. It emits the high-order address byte during accesses to external memory. It also receives the high-order address bits during EPROM programming and program verification. Port 2 can sink/source four LS TTL inputs.

Port 3
Port 3 is an 8-bit bidirectional I/O port with internal pullups. It also serves the functions of various special features of the MCS®-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/counter 0 external input)
P3.5	T1 (Timer/counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

Port 3 can sink/source four LS TTL inputs.

RST
A high on this pin for two machine cycles while the oscillator is running resets the device. A small external pulldown resistor (≈8.2kΩ) from RST to V_{SS} permits power-on reset when a capacitor (≈10 μf) is also connected from this pin to V_{CC}.

ALE/$\overline{PROG}$
Address Latch Enable output for latching the low byte of the address during accesses to external memory. ALE is activated at a constant rate of 1/6 the oscillator frequency except during an external data memory access at which time one ALE pulse is skipped. ALE can sink/source 8 LS TTL inputs. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.

$\overline{PSEN}$
Program Store Enable output is the read strobe to external Program Memory. $\overline{PSEN}$ is activated twice each machine cycle during fetches from external Program Memory. (However, even when executing out of external Program Memory two activations of $\overline{PSEN}$ are skipped during each access to external Data Memory.) $\overline{PSEN}$ is not activated during fetches from internal Program Memory. $\overline{PSEN}$ can sink/source 8 LS TTL inputs.

$\overline{EA}$/VPP
When $\overline{EA}$ is held high, the 8751H executes out of internal Program Memory (unless the Program Counter exceeds 0FFFH). When $\overline{EA}$ is held low, the 8751H executes only out of external Program Memory. This pin also receives the 21V programming supply voltage (VPP) during EPROM programming. This pin should not be floated during normal operation.

XTAL1
Input to the inverting amplifier that forms the oscillator. XTAL1 should be grounded when an external oscillator is used.

XTAL2
Output of the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external oscillator signal when an external oscillator is used.

The newest MCS-51 members, the 8032 and 8052, have more on-chip memory and an additional 16-bit timer/counter. The new timer can be used as a timer, a counter, or to generate baud rates for the serial port. As a timer/counter, it operates in either a 16-bit auto-reload mode or a 16-bit "capture" mode. This new feature is described in Section 6.6.2.

Pinouts are shown in the individual data sheets and on the inside back cover of this handbook.

Table 1. MCS®-51 Family Members

PART	TECHNOLOGY	ON-CHIP PROGRAM MEMORY	ON-CHIP DATA MEMORY
8051AH	HMOS II	4K — ROM	128
8031AH	HMOS II	NONE	128
8751H	HMOS I	4K — EPROM	128
80C51	CHMOS	4K — ROM	128
80C31	CHMOS	NONE	128
8052	HMOS II	8K — ROM	256
8032	HMOS II	NONE	256

The major MCS®-51 features are:
- 8-Bit CPU
- On-Chip oscillator and clock circuitry
- 32 I/O lines
- 64K address space for external data memory
- 64K address space for external program memory
- Two 16-bit timer/counters (three on 8032/8052)
- A five-source interrupt structure (six sources on 8032/8052) with two priority levels
- Full duplex serial port
- Boolean processor

6.1 MEMORY ORGANIZATION

The 8051 has separate address spaces for Program Memory and Data Memory. The Program Memory can be up to 64K bytes long. The lower 4K (8K for 8052) may reside on-chip. The Data Memory can consist of up to 64K bytes of off-chip RAM, in addition to which, it includes 128 bytes of on-chip RAM (256 bytes for the 8052), plus a number of "SFRs" (Special Function Registers) as listed below.

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer (consisting of DPH and DPL	83H 82H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
+*T2CON	Timer/Counter Control	88H
TCON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 (high byte)	8CH
TL0	Timer/Counter 0 (low byte)	8AH
TH1	Timer/Counter 1 (high byte)	8DH
TL1	Timer/Counter 1 (low byte)	8BH
+TH2	Timer/Counter 2 (high byte)	0CDH
+TL2	Timer/Counter 2 (low byte)	0CCH
+RCAP2H	Timer/Counter 2 Capture Register (high byte)	0CBH
+RCAP2L	Timer/Counter 2 Capture Register (low byte)	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buff	99H
PCON	Power Control	97H

The SFRs marked with an asterisk (*) are both bit- and byte-addressable. The SFRs marked with a plus sign (+) are present in the 8052 only. The functions of the SFRs are described as follows.

ACCUMULATOR

ACC is the Accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

B REGISTER

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

PROGRAM STATUS WORD

The PSW register contains program status information as detailed in Figure 6-2.

STACK POINTER

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM,

the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 to 3

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER REGISTERS

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit counting registers for Timer/Counters 0, 1, and 2, respectively.

CAPTURE REGISTERS

The register pair (RCAP2H, RCAP2L) are the capture registers for the Timer 2 "capture mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode. More about Timer 2's features in Section 6.6.2.

CONTROL REGISTERS

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the timer/counters, and the serial port. They are described in later sections.

6.2 OSCILLATOR AND CLOCK CIRCUIT

XTAL1 and XTAL2 are the input and output of a single-stage on-chip inverter, which can be configured with off-chip components as a Pierce oscillator, as shown in Figure 6-3. The on-chip circuitry, and selection of off-chip components to configure the oscillator are discussed in Section

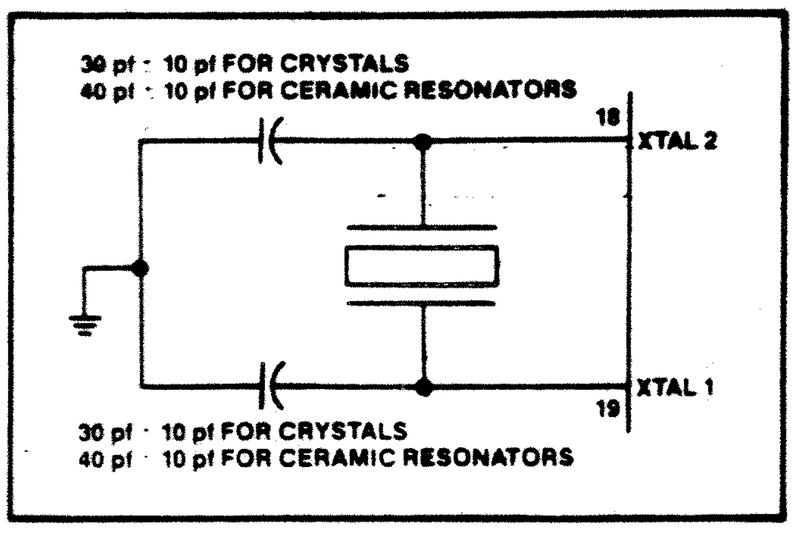


Figure 6-3. Crystal/Ceramic Resonator Oscillator

(MSB) (LSB)

CY	AC	F0	RS1	RS0	OV	—	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank Select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	
OV	PSW.2	Overflow flag.
—	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the accumulator, i.e., even parity.

Note— the contents of (RS1, RS0) enable the working register banks as follows:

(0.0)—Bank 0 (00H-07H)
(0.1)—Bank 1 (08H-0FH)
(1.0)—Bank 2 (10H-17H)
(1.1)—Bank 3 (18H-1FH)

Figure 6-2. PSW: Program Status Word Register