

COLECO

ADAM TECHNICAL MANUAL

- I - FUNCTIONAL DESCRIPTION
- II - EOS USER'S MANUAL
- III - OPTIONAL PERIPHERALS

ADAM™
TECHNICAL REFERENCE MANUAL

PRELIMINARY RELEASE

COLECO INDUSTRIES, INC

ACKNOWLEDGEMENTS

Editor: Maria Higginl

Technical Consultants: David K. Hwang
Robert F. Jepson

Cover Design: Laura Shea

COLECO MAKES NO REPRESENTATIONS OR WARRANTIES WHATSOEVER, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, IN CONNECTION WITH THE MATERIALS CONTAINED HEREIN, AND SUCH MATERIALS ARE DISCLOSED AS IS. COLECO SHALL HAVE NO LIABILITY FOR ANY LOSSES CAUSED TO RECIPIENTS OF THESE MATERIALS BY REASON OF ANY CHANGES OR MODIFICATIONS MADE BY COLECO IN THESE MATERIALS AFTER THEIR DISCLOSURE HEREIN. IN ADDITION, COLECO SHALL HAVE NO LIABILITY FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES OR LOSSES WHATSOEVER, INCLUDING LOSS OF PROFITS, IN CONNECTION WITH THE USE OF THE MATERIALS DISCLOSED HEREIN.

© 1984 Coleco Industries, Inc.

All rights reserved

ADAM™, SmartBASIC™, SmartWRITER™, and AdamNET™ are trademarks of Coleco Industries, Inc. ColecoVision® is a registered trademark of Coleco Industries, Inc. Buck Rogers™ indicates a trademark of the Dille Family Trust © 1982 The Dille Family Trust. Planet of Zoom™ and SEGA® are trademarks of SEGA Enterprises, Inc. ©1982 SEGA Enterprises, Inc.

PREFACE

The ADAM Family Computer System Technical Reference Manual is a source of technical information for both hardware and software designers. This preliminary release of the manual includes the most essential information. Future releases will address optional peripherals, additional tools and utilities, and further detailed information on the basic ADAM system.

Operating system source code listings may be requested with the form on the last page of this manual.

Chapter 1 is a general introduction and orientation to ADAM.

Chapter 2, Hardware, describes ADAM's hardware architecture, and discusses the function of each major component.

Chapter 3, Software, includes information on memory configurations, the operating system and the external I/O bus, AdamNet. Application software is also discussed.

Chapter 4, Optional Peripherals, describes Coleco-engineered hardware peripherals available for ADAM. This chapter will be developed as new peripherals become available.

Chapter 5, Development Tools and Utilities, includes information on both software and hardware tools that aid development of Adam software and peripherals.

The Technical Reference Manual is not a general user's manual. For information on setting up and using the system, refer to the manuals provided with ADAM:

Getting Started: the ADAM Set-Up Manual
Typing with ADAM: the ADAM Word Processing Manual
Programming with ADAM: The SmartBASIC Manual

TABLE OF CONTENTS

Preface

Chapter 1: General Introduction

- 1. Hardware Overview
- Fig. 1-1 ADAM Home Computer System Set-Up Diagram
- Fig. 1-2 Expansion Module #3 Set-Up Diagram
- 2. Software Overview
- Fig 1-3 Adam Software Architecture

Chapter 2: Hardware

- 1. Introduction
- Fig. 2-1 System Block Diagram
- Fig. 2-2 System Flow Diagram
- 2. The Memory Console
- 2.1 The Memory and I/O Board
- 2.1.1 Theory of Operation
- Fig. 2-3 Memory and I/O Board Block Diagram
- 2.1.2 Master 6801 Microcomputer
- 2.1.3 Memory Input/Output Controller (MIOC)
- 2.1.4 ROM Circuitry
- 2.1.5 Dynamic RAM Circuitry
- 2.1.6 AdamNet Interface Circuitry
- 2.1.7 Card-Edge Expansion Connectors
- 2.1.7a Connector #1
- 2.1.7b Connector #2
- 2.1.7c Connector #3
- 2.1.8 Expansion Port
- 2.1.9 Interconnects
- 2.2 The CPU Board
- 2.2.1 Theory of Operation
- Fig. 2-4 Block Diagram: CPU Board
- 2.2.2 Z80 Microprocessor
- 2.2.3 ROM Circuitry
- 2.2.4 Video Display Processor
- 2.2.5 Sound Generator
- 2.2.6 RF Circuitry
- 2.2.7 Game Controller Circuitry
- 2.2.8 Clock Generation
- 2.2.9 Interconnects
- 2.3 Data Pack Drive
- 2.3.1 Theory of Operation
- 2.3.2 Servo Printed Circuit Board
- 2.3.3 Read Write Printed Circuit Board

ADAM™ TECHNICAL REFERENCE MANUAL
PRELIMINARY RELEASE

- 2.3.4 Data Pack Specifications
- 2.3.5 I/O Signals between Memory and I/O Board and Data Drive
- 2.4 Differences of Expansion Module 3
- 3. The Keyboard
 - 3.1 Theory of Operation
 - 3.2 Interconnects
- 4. The Printer
 - 4.1 Theory of Operation
 - 4.2 Printer Board
 - 4.3 Interconnects
- 5. Game Controllers
- Fig. 2-5 Game Controller Configuration
- 6. Power Supply
 - 6.1 Power Supply Voltage
 - 6.2 Excessive Current Output Protection
 - 6.3 Printer/Memory Console Interface Cable
 - 6.4 Power Supply Output to CPU (via Printer/Memory Console Interface Cable)
 - 6.5 Power Supply Output to Printer

Chapter 3: Software

- 1. Introduction
- 2. Memory Map and Power/Up Reset Procedure
 - 2.1 Lower Memory Options
 - 2.2 Upper Memory Options
 - 2.3 Power Up/Computer Reset Procedures
 - 2.4 Z80 I/O Port Assignments
 - 2.5 Memory Map Control
 - 2.6 Reset Procedures
- 3. AdamNet
 - 3.1 Introduction
 - Fig. 3-1 Bus Network
 - 3.2 Network Master Concept
 - Fig 3-2 MAC Internals
 - 3.3 Message Philosophy and Definition
 - 3.4 Error Control
 - Fig. 3-3 Error Control
 - 3.5 Class Of Service Concept
 - 3.6 Power Up/Initialization
 - 3.7 Link Speed
 - 3.8 Master Node Interface
 - 3.9 Functional Overview - Z80 and 6801 Master
- 4. Operating System
 - 4.1 EOS

- Fig. 3-4 EOS Memory Map
 - 4.1.1 EOS Overwrite Addresses
 - 4.1.2 EOS Files
 - 4.1.3 File Types and Headers
 - 4.1.4 EOS Executive Calls
 - 4.1.5 EOS Routines Adapted from OS_7
 - 4.1.6 Initializing EOS
 - 4.1.7 EOS Entry Points
 - 4.1.8 EOS Error Codes
- 4.2 OS_7
 - 4.2.1 Introduction
 - 4.2.2 Graphics Generation Software
 - 4.2.3 Sound Generation Software
 - 4.2.4 Interrupt Handling and Write Deferral Software
 - 4.2.5 Timing Software
 - 4.2.6 Controller Interface
 - 4.2.7 Boot-Up Software
 - 4.2.8 Miscellaneous Utilities
 - 4.2.9 Defined Reference Locations
- 5. Tape Format and Other Tape Considerations
- 6. SmartWRITER
 - 6.1 Memory Map
 - 6.2 SmartWRITER-Compatible Files
- 7. SmartBASIC
 - 7.1 Memory Map
- 8. Super Games and Other Programs Using OS7
 - 8.1 Memory Map
- 9. ROM-based Cartridge Programs
 - 9.1 Memory Map

Chapter 4: Optional Peripherals

Chapter 5: Development Tools and Utilities

- 1. Super Game Guidelines

Appendices

- 1. Keyboard Table
- 2. ADAM Emulation Considerations
- 3. The ColecoVision Programmers Manual - Detail Sections
- 4. Schematics and Component Location/Identification Drawings
 - 4.1 Memory and I/O Board Schematic (Sheet 1)
 - 4.1 Memory and I/O Board Schematic (Sheet 2)

ADAM™ TECHNICAL REFERENCE MANUAL
PRELIMINARY RELEASE

- 4.1 Memory and I/O Board Component Location/Identification Drawing
- 4.2 CPU Board Schematic
- 4.2 CPU Board Component Location/Identification Drawing
- 4.3 Interconnect Board Schematic
- 4.3 Interconnect Board Component Location/Identification Drawing
- 4.4 Linear Power Supply Schematic
- 4.4 Linear Power Supply Component Location/Identification Drawing
- 4.4 Linear Power Supply Sub-Assembly

CHAPTER 1: GENERAL INTRODUCTION

1. Hardware Overview

The ADAM Family Computer System consists of three major components: the memory console, the keyboard, and the printer. The consumer provides his own TV or monitor. Other equipment provided with the system are two "joystick" game controllers, various cords and cables to connect the components, and an antenna switch box.

The memory console houses the main memory and CPU of the system, and one data pack drive. Space and connectors are provided for another drive. Two printed circuit boards contain 64K RAM, 16K video RAM, an expansion port, two AdamNet ports, three card connectors and a cartridge slot. Two additional printed circuit boards control the drives.

The system reads from and stores on digital data packs. Digital data packs are a reel to reel magnetic tape encased in a Lexan™ cassette. Each data pack can store up to 256K bytes.

The keyboard has 75 full travel keys, including ten command keys and six programmable function keys. A "power on" LED indicator on the right side of the keyboard shows when the system is on. The keyboard contains one printed circuit board.

The printer is a letter-quality, bi-directional, daisy wheel printer. Paper feeds into the printer through a friction-feed mechanism that accommodates single sheets of paper up to 9½ inches wide. With the addition of an optional tractor-feed mechanism, the printer also accommodates continuous, "fan-fold" paper. Pitch is 10 characters to the inch, and printing speed is 10 characters per second. The printer contains two printed circuit boards, one for the printer and one for the power supply.

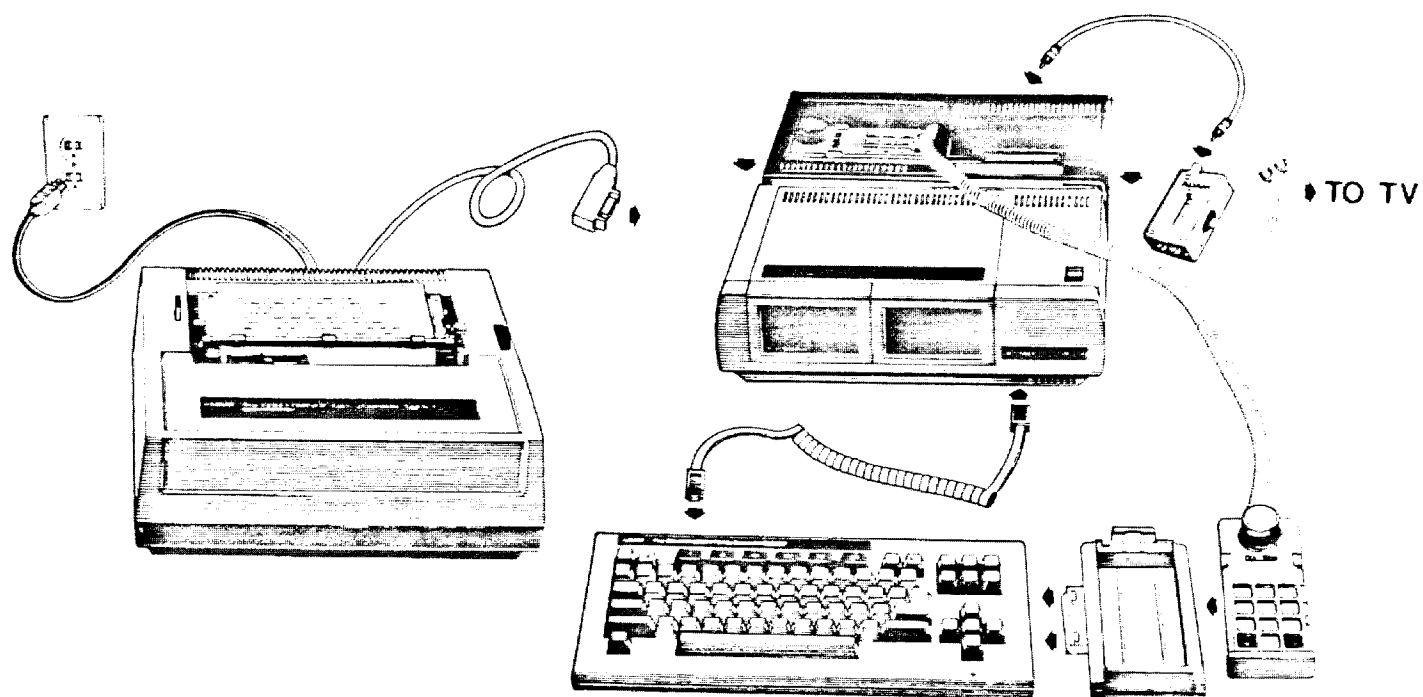
The computer's power supply, which produces 4 regulated DC voltages, is housed in the printer.

ADAM is available in two models, the complete system and Expansion Module #3. When the memory console of Expansion Module #3 is connected to ColecoVision, the two models are essentially identical in function. Figures 1-1 and 1-2 show how the components for each models are connected.

The block diagram in Figure 2-1 represents a high-level view of the system's hardware design. The major elements in the

block diagram is discussed in greater detail in Chapter 2,
Hardware.

FIGURE 1-2: EXPANSION MODULE #3 SET UP DIAGRAM



1.2 Software Overview

ADAM's hardware components are linked together by a 62.5K bps, half-duplex, shared serial bus, known as AdamNet.

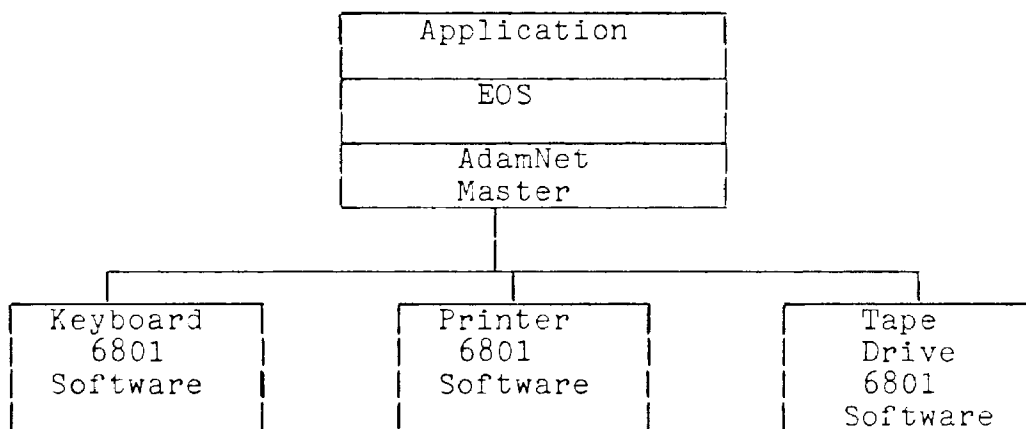
EOS (Elementary Operating System) is a collection of service routines that provides input and output facilities to peripheral devices, in such a way that application programs need not address the physical characteristics of the peripherals or the operation of AdamNet. EOS also provides file management for manipulating data on mass storage devices.

OS_7 is a run-time user's library of software modules that controls graphics, sound, timing, etc. EOS contains many modules equivalent to OS_7 modules, but some have different inputs and outputs.

ADAM contains a ROM-based electric typewriter/word processor/editor called SmartWRITER. SmartBASIC and the Buck Rogers Planet Of Zoom Super Game are included with Adam on data packs.

Each of ADAM's software components is discussed in greater detail in Chapter 3. Figure 1-3 depicts Adam's software architecture.

FIGURE 1-3: ADAM SOFTWARE ARCHITECTURE



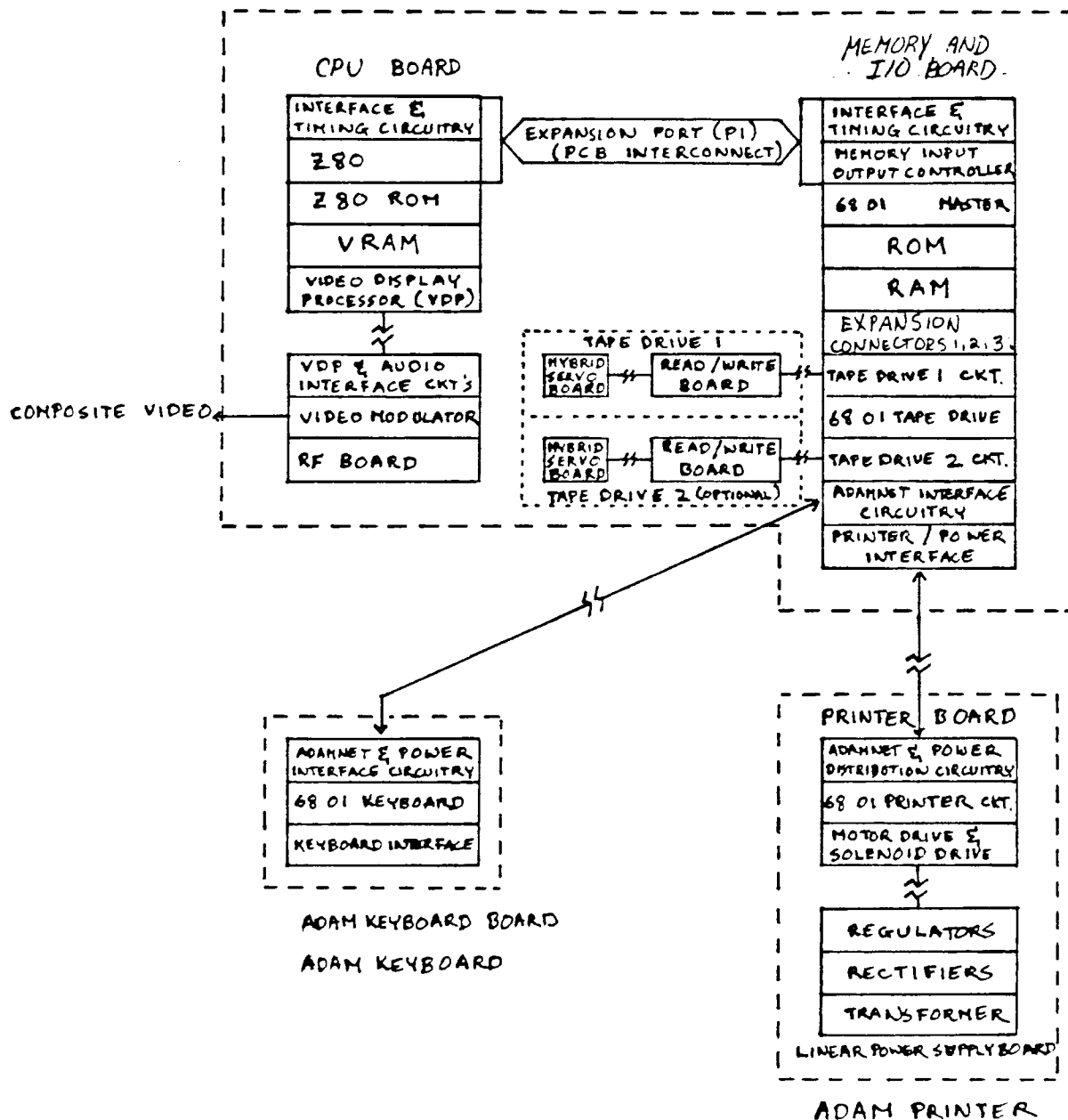
CHAPTER 2: HARDWARE

1. Introduction

This chapter presents technical information on each of the major logical components of ADAM identified in the System Block Diagram, Figure 2-1. The Appendix contains schematics and component location/identification drawings.

For the convenience of hardware developers, pin and signal connections for all expansion connectors are given. The Memory Console provides a total of four expansion connectors. Three female card edge connectors are accessed by removing the top cover of the Memory Console. These are referred to as expansion connectors #1, #2 and #3. One male card edge connector extends from the right side of the Memory Console. This is referred to as the Expansion Port.

FIGURE 2-1: SYSTEM BLOCK DIAGRAM



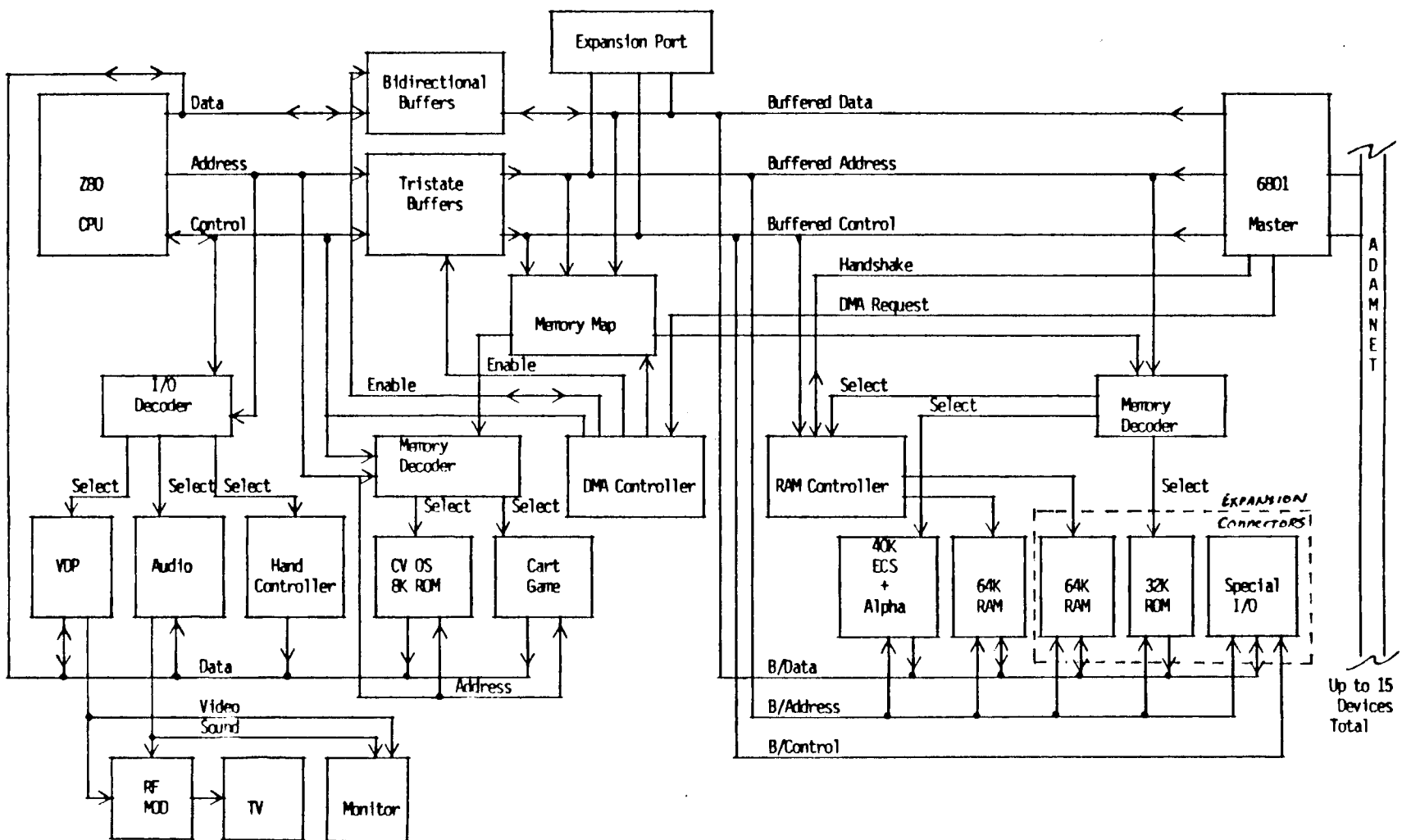


FIGURE 2-2: SYSTEM FLOW DIAGRAM

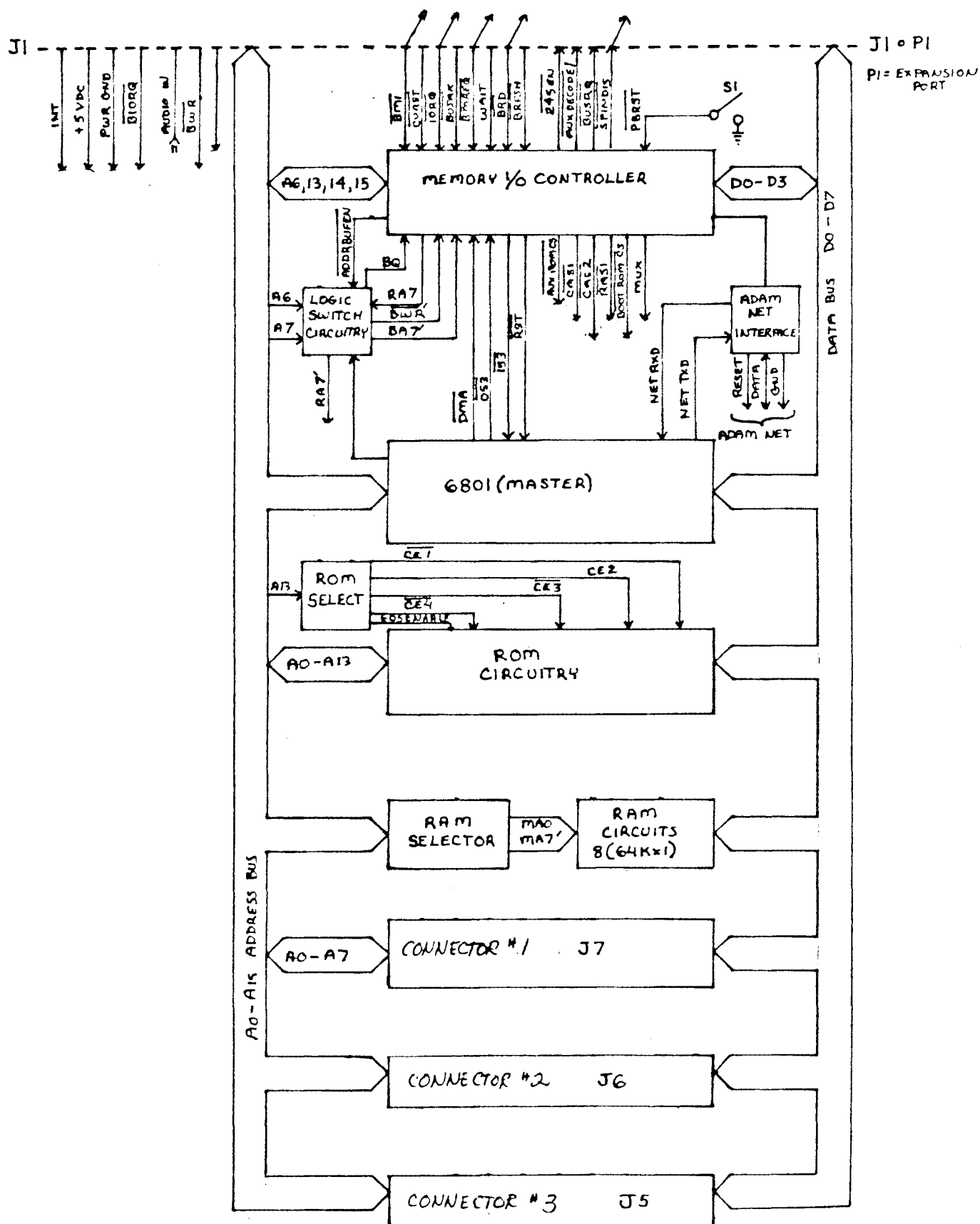
2. THE MEMORY CONSOLE

2.1 The Memory and I/O Printed Circuit Board

2.1.1 Theory of Operation

The Memory and I/O Board contains the 6801 Master microcomputer, 72K bytes of RAM and provisions for up to 72K of ROM/EPROM. This board provides the circuitry required to interface the keyboard, printer, tape drive and future options. Three card-edge connectors provide access for future options. An expansion port provides access for external peripherals. A custom LSI circuit, the Memory Input Output Controller (MIOC), interfaces the 6801 Master microcomputer with the Z80 microprocessor on the CPU Board. Another 6801 microcomputer on the Memory and I/O Board controls the operation of the tape drive interface circuitry.

FIGURE 2-2: THE MEMORY AND I/O BOARD BLOCK DIAGRAM



2.1.2 The Master 6801 Microcomputer

The Master 6801 microcomputer's primary function is to control system access to the keyboard, printer, tape drive and future peripherals. The Master 6801 microcomputer is a front end network processor that supports the Z80. The Master 6801 communicates with the Z80 via the Memory Input Output Controller. The Master 6801 reads and writes information to and from network peripherals on command by the Z80.

The 6801 chip provides 2048 bytes of ROM, 128 bytes of RAM and a UART. The Master 6801 is configured for single chip mode operation and runs, as do Adam's other 6801's, at a 1MHZ rate. This frequency is derived from an external 4MHZ crystal and the 6801's internal divide-by-4 circuitry.

2.1.3 The Memory Input Output Controller (MIOC)

The Memory Input Output Controller is a 40-pin IC that interfaces two dissimilar microprocessors (the Z80 and the Master 6801) by performing the proper decoding and timing functions. It is responsible for selecting the memory configuration (Refer to Chapter 3, Section 2). The MIOC has 22 input signals, 16 output signals, power and ground.

2.1.4 ROM Circuitry

Room is provided on the Memory and I/O Board for up to 40K of ROM. ROM selection is controlled by a decoder circuit which is driven by A13, BOOTROMCS and EOS ENABLE. The Memory and I/O Board ROM circuit contains EOS (Elementary Operating System) software.

2.1.5 Dynamic RAM Circuitry

The dynamic RAM circuit consists of eight 64K RAM chips arranged so that each represents one specific data bus bit. Information written to or read from RAM is controlled by the $\overline{BWR}$, $\overline{RAS}$ and $\overline{CAS}$ signals. The latched Z80 address bus BA0 - BA15 (or BA0 - BA6, RA7, BA8 - BA15) is provided along with MUX from the MIOC to the two data selector multiplexers, which output MA0 - MA7 (or RA7) to the DRAM address bus. The Z80 provides a 7-bit refresh address after each instruction fetch. The MIOC generates an eighth bit for 256-refresh cycle dynamic RAMS. See Appendix 2, Adam Emulation Considerations for further information.

2.1.6 AdamNet Interface Circuitry

A quad comparator circuit provides data to and from the Master 6801 microcomputer via a half-duplex 62.5 kilobaud

serial network called AdamNET. The comparator also can reset all the devices on AdamNET via MIOC control. AdamNET links the tape drive, printer and keyboard to the Master 6801. Each of these peripherals has a 6801 and a quad comparator circuit that control AdamNET. Besides the data signal and reset signal, ground and power are provided as part of the AdamNET bus.

2.1.7 Card Edge Expansion Connectors

Three card edge connectors are provided for future development. Refer to Subsection 2.1.9 for pin connections.

2.1.7a Connector #1

This connector is soldered to the Memory and I/O Board, and is labelled J7.

2.1.7b Connector #2

This connector is designed for expansion ROM and I/O devices and is soldered to the Memory and I/O Board. It is labelled J6.

2.1.7c Connector #3

This connector allows for expansion RAM and/or ROM up to 64K bytes, and is labelled J5.

2.1.8 Expansion Port

The expansion port is connected to the Memory and I/O Board at P1.

Pin	Type	Refer To Table	Pin	Type	Refer To Table
1	Ground		31	Audio input	
2	Ground		32	Video input enable + 9VDC	
3	BD3 Tristate, I/O	1, 2	33	NTSC Composite video input, 6VDC, 1.5 VAC	
4	BA14 Tristate output	1, 2	34	GAME MODE RESET output	
5	Y2 LS138 decoder output		35	Sound chip 76489 disable, 0 Volts DC	
6	Y1 LS138 decoder output		36	Not in use	
7	<u>HALT</u> input	1, 2	37	BA11 Tristate output	1, 2
8	<u>BWR</u> Tristate output	1, 2	38	BA12 Tristate output	1, 2
9	<u>NMI</u> input/output	1, 2	39	VDP Sync/Reset input	1
10	<u>SPINNER INT DISABLE</u> input	1			
11	<u>BUSRQ</u> input	1	40	<u>BIORQ</u> Tristate output	
12	<u>BD1</u> Tristate, I/O	1, 2	41	Not used	
13	<u>Z80</u> Reset input	1, 2	42	Not used	1
14	<u>BD0</u> Tristate, I/O	1, 2	43	BA15 Tristate output	1, 2
15	<u>BM1</u> Tristate output	1, 2	44	BA3 Tristate output	1, 2
16	BD7 Tristate, I/O	1, 2	45	B03.58 MHz clock	
17	BD6 Tristate, I/O	1, 2	46	BD2 Tristate, I/O	1, 2
18	BA1 Tristate output	1, 2	47	BA0 Tristate output	1, 2
19	BD4 Tristate, I/O	1, 2	48	<u>BD5</u> Tristate, I/O	1, 2
20	BA2 Tristate	1, 2	49	<u>BRFSH</u> Tristate output	
21	BA4 Tristate output	1, 2	50	<u>WAIT</u> input	1, 2
22	BA13 Tristate output	1, 2	51	<u>INT</u> input	1, 2
23	BA5 Tristate output	1, 2	52	<u>BUSAK</u> output	1, 2
24	BA6 Tristate output	1, 2	53	<u>BRD</u> Tristate output	1, 2
25	BA7 Tristate output	1, 2	54	<u>BMREQ</u> Tristate output	1
26	BA8 Tristate output	1, 2	55	<u>IORQ</u> output	1, 2
27	BA9 Tristate output	1, 2	56	AUDIO 76489 RDY output	
28	<u>BA10</u> Tristate output	1, 2	57	+12V	
29	<u>AUX DECODE 1</u> input	1	58	+5V	
30	<u>AUX DECODE 2</u> input	1	59	+5V	
			60	-5V	

$\bar{X}$ - Denotes active low

TABLE 1: DC CHARACTERISTICS

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
V _{IL}	Input low voltage	-0.3		0.8	V	
V _{IH}	Input high voltage	2.0		V _{CC}	V	
V _{OL}	Output low voltage			0.4	V	I _{OL} = 1.8mA
V _{OH}	Output high voltage	2.4			V	I _{OH} = 250uA
I _{LI}	Input leakage current			±10	uA	V _{IN} = 0 to V _{CC}
I _{LO}	Tri-state output Leakage current in float			±10	uA	V _{OUT} = 0.4V to V _{CC}

TABLE 2: TIMING REFERENCE TABLE

[]* See notes at end of table

SIGNAL	SYMBOL	PARAMETER	MIN (ns)	MAX (ns)
A0-15	$t_D(AD)$	Address output delay		110
	$t_F(AD)$	Delay to float		90
	t_{acm}	Address stable prior to $\overline{MREQ}$ (memory cycle)	[1]*	
	t_{aci}	Address stable prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O cycle)	[2]*	
	t_{ca}	Address stable from $\overline{RD}$, $\overline{WR}$ $\overline{IORQ}$ or $\overline{MREQ}$	[3]*	
	t_{caf}	Address stable from $\overline{RD}$ or $\overline{WR}$ during float	[4]*	
D0-7	$t_D(D)$	Data output delay		150
	$t_F(D)$	Delay to float during write cycle		90
	$t_{SI}(D)$	Data setup time to rising edge of clock during M1 cycle	35	
	$t_{SF}(D)$	Data setup time to falling edge at clock during M2 to M5	50	
	t_{dcn}	Data stable prior to $\overline{WR}$ (memory cycle)	[5]*	
	t_{dci}	Data stable prior to $\overline{WR}$ (I/O cycle)	[6]*	
	t_{dcf}	Data stable from $\overline{WR}$	[7]*	
	t_H	Input hold time	0	
$\overline{MREQ}$	$t_{DL\overline{Q}}(MR)$	$\overline{MREQ}$ delay from falling edge of clock, $\overline{MREQ}$ low	20	85
	$t_{DH\overline{Q}}(MR)$	$\overline{MREQ}$ delay from rising edge of clock, $\overline{MREQ}$ high		85
	$t_{DH\overline{Q}}(MR)$	$\overline{MREQ}$ delay from falling edge of clock $\overline{MREQ}$ high		85
	$t_w(MRL)$	Pulse width, $\overline{MREQ}$ low	[8]*	
	$t_w(MRH)$	Pulse width, $\overline{MREQ}$ high	[9]*	
$\overline{IORQ}$	$t_{DL\overline{Q}}(IR)$	$\overline{IORQ}$ delay from rising edge of clock $\overline{IORQ}$ low		75
	$t_{DL\overline{Q}}(IR)$	$\overline{IORQ}$ delay from falling edge of clock $\overline{IORQ}$ low		85
	$t_{DH\overline{Q}}(IR)$	$\overline{IORQ}$ delay from rising edge of clock, $\overline{IORQ}$ high		85
	$t_{DH\overline{Q}}(IR)$	$\overline{IORQ}$ delay from falling edge of clock, $\overline{IORQ}$ high		85

TABLE 2: TIMING REFERENCE TABLE (Cont'd)

[]* See notes at end of table

SIGNAL	SYMBOL	PARAMETER	MIN (ns)	MAX (ns)
$\overline{\text{RD}}$	$t_{\text{DL}\overline{\Phi}}(\text{RD})$	$\overline{\text{RD}}$ delay from rising edge of clock, $\overline{\text{RD}}$ low	15	85
	$t_{\text{DL}\overline{\Phi}}(\text{RD})$	$\overline{\text{RD}}$ delay from falling edge of clock, $\overline{\text{RD}}$ low		95
	$t_{\text{DH}\overline{\Phi}}(\text{RD})$	$\overline{\text{RD}}$ delay from rising edge of clock, $\overline{\text{RD}}$ high		85
	$t_{\text{DH}\overline{\Phi}}(\text{BD})$	$\overline{\text{RD}}$ delay from falling edge of clock, $\overline{\text{RD}}$ high		85
$\overline{\text{WR}}$	$t_{\text{DL}\overline{\Phi}}(\text{WR})$	$\overline{\text{WR}}$ delay from rising edge of clock, $\overline{\text{WR}}$ low	[10]*	65
	$t_{\text{DL}\overline{\Phi}}(\text{WR})$	$\overline{\text{WR}}$ delay from falling edge of clock, $\overline{\text{WR}}$ low		80
	$t_{\text{DH}\overline{\Phi}}(\text{WR})$	$\overline{\text{WR}}$ delay from falling edge of clock, $\overline{\text{WR}}$ high		80
	$t_{\text{w}}(\overline{\text{WRL}})$	Pulse width, $\overline{\text{WR}}$ low		
$\overline{\text{M1}}$	$t_{\text{DL}}(\text{M1})$	$\overline{\text{M1}}$ delay from rising edge of clock $\overline{\text{M1}}$ low		100
	$t_{\text{DH}}(\text{M1})$	$\overline{\text{M1}}$ delay from rising edge of clock $\overline{\text{M1}}$ high		100
$\overline{\text{RFSH}}$	$t_{\text{DL}}(\text{RF})$	$\overline{\text{RFSH}}$ delay from rising edge of clock, $\overline{\text{RFSH}}$ low		130
	$t_{\text{DH}}(\text{RF})$	$\overline{\text{RFSH}}$ delay from rising edge of clock, $\overline{\text{RFSH}}$ high		120
$\overline{\text{WAIT}}$	$t_{\text{S}}(\text{WT})$	$\overline{\text{WAIT}}$ setup time to falling edge of clock	70	
$\overline{\text{HALT}}$	$t_{\text{D}}(\text{HT})$	$\overline{\text{HALT}}$ delay time from falling edge of clock		300
$\overline{\text{INT}}$	$t_{\text{S}}(\text{IT})$	$\overline{\text{INT}}$ setup time to rising edge of clock	80	

TABLE 2: TIMING REFERENCE TABLE (Cont)

[]* See notes at end of table

SIGNAL	SYMBOL	PARAMETER	MIN (ns)	MAX (ns)
$\overline{\text{NMI}}$	$t_w(\text{NMI})$	Pulse width, $\overline{\text{NMI}}$ low	80	
$\overline{\text{BUSRQ}}$	$t_s(\text{BQ})$	$\overline{\text{BUSRQ}}$ setup time to rising edge of clock	50	
$\overline{\text{BUSAk}}$	$t_{DL}(\text{BA})$	$\overline{\text{BUSAk}}$ delay from rising edge of clock, $\overline{\text{BUSAk}}$ low		100
	$t_{DH}(\text{BA})$	$\overline{\text{BUSAk}}$ delay from falling edge of clock, $\overline{\text{BUSAk}}$ high		100
$\overline{\text{RESET}}$	$t_s(\text{RS})$	$\overline{\text{RESET}}$ setup time to rising edge of clock	60	
	$t_F(\text{C})$	Delay to/from float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$)		80
	t_{mr}	$\overline{\text{MI}}$ stable prior to $\overline{\text{IORQ}}$ (interrupt Ack.)	[11]*	

Timing Reference Table Notes

- [1] $t_{acm} = t_w(\emptyset H) + t_f - 65$
 - [2] $t_{aci} = t_c - 70$
 - [3] $t_{ca} = t_w(\emptyset L) + t_r - 50$
 - [4] $t_{caf} = t_w(\emptyset L) + t_r - 45$
 - [5] $t_{dcn} = t_c - 170$
 - [6] $t_{dci} = t_w(\emptyset L) + t_r - 170$
 - [7] $t_{cdf} = t_w(\emptyset L) + t_r - 70$
 - [8] $t_w(\overline{\text{MRL}}) = t_c - 30$
 - [9] $t_w(\overline{\text{MRH}}) = t_w(\emptyset H) + t_f - 20$
 - [10] $t_w(\overline{\text{WRL}}) = t_c - 30$
 - [11] $t_{mr} = 2t_c + t_w(\emptyset H) + t_f - 65$
- t_c = clock period: 279.36 ns $\pm$.01
 $t_w(\emptyset H)$ = clock pulse width, clock High = 120 ns min
 $t_w(\emptyset L)$ = clock pulse width, clock Low = 120 ns min
 t_f = clock fall time = 15 ns max
 t_r = clock rise time = 15 ns max

2.1.9 Interconnects2.1.9a Memory and I/O Board/CPU Board

The Memory and I/O Board is connected to the CPU Board at J1, with two 30-pin ribbon cables and a dual 30-pin card edge connector.

<u>Signal</u>	<u>Description</u>
BD0-BD7	8 bidirectional data lines. BD0 is least significant, BD7 is most significant.
BA0-BA15	16 address lines to Memory and I/O Board. BA0 is least significant, BA15 is most significant.
<u>BWR</u>	Output of Z80 to Memory and I/O Board; write strobe used to output data during an I/O or memory operation. Indicates a write operation.
<u>BRD</u>	Output of Z80 to Memory and I/O Board; read strobe used to clock data into the Z80 during an I/O or memory operation.
<u>BMREQ</u>	Output of Z80 to Memory and I/O Board; indicates present read or write operation is directed to memory or memory-mapped devices.
<u>IORQ</u> <u>BIORQ</u>	Same as <u>BMREQ</u> , but indicates an I/O operation instead of memory or memory-mapped devices
<u>BRFSH</u>	Output of Z80 to Memory and I/O Board; indicates BA0-BA6 contain a row address for the required dynamic memory refresh. (An eighth row address bit is generated by the MIOC 'RA7'.)
<u>RST</u>	Generated by the MIOC as a result of either a game <u>CVRST</u> or computer <u>PBRST</u> reset. It connects to and resets the Colecovision or CPU Board.
<u>Bφ</u>	System clock generated on Colecovision or CPU Boards. Line connects to Memory and I/O Board.

<u>WAIT</u>	Used to insert extra clock cycles into Z80 timing during opcode fetch cycles and when accessing slow memory or I/O. Excessive use of <u>WAIT</u> causes inadequate dynamic RAM refresh.
<u>ADDRBUFEN</u>	An active low signal enables the address and control signal buffers between the Colecovision or CPU Board, and the Memory and I/O Board. The control signals are <u>BRD</u> , <u>BWR</u> , <u>BRFSH</u> , <u>BMREQ</u> , <u>BMI</u> , and <u>BIORQ</u> . A high level disables these signals from the Z80, and allows them to go tristate (high-impedance). This occurs during a DMA cycle where another device needs to access memory or devices on the Memory and I/O Board. See <u>BUSRQ</u> and <u>BUSAK</u> .
<u>245EN</u>	Same as <u>ADDRBUFEN</u> except <u>245EN</u> controls the buffer for BD0 through BD7 data lines to CPU or Colecovision buffer board.
<u>BUSRQ</u> (unbuffered) <u>BUSAK</u> (buffered)	<u>BUSRQ</u> is generated by the MIOC as the result of a DMA request. The <u>BUSRQ</u> signal requests that the Z80 relinquish the address and data busses and certain control signals at the end of its current cycle. After receiving the <u>BUSRQ</u> the Z80 responds with a <u>BUSAK</u> signal to indicate it has relinquished the bus. The Z80 remains in an inactive state until the controlling device removes the <u>BUSRQ</u> signal. The <u>BUSRQ</u> line connects to the Colecovision or CPU Board. Generally, only the master 6801 may assert a <u>BUSRQ</u> .
<u>BMI</u>	Output of Z80 from CPU or Colecovision Board; indicates the present memory cycle is an opcode fetch (start of next instruction).
<u>CVRST</u>	This signal generates an <u>RST</u> to the Z80 processor. Also reset are the MIOC and master 6801. <u>CVRST</u> initializes the MIOC memory map such that addresses from 0-1FFFH enable the OS-7 ROM; 2000H through 7FFFH enable RAM1; and 8000H through FFFFH enables the game cartridge.
<u>AUXDECODE1</u>	Generated by Memory and I/O Board. Selects or deselects the OS-7 ROM.

INT

Active low, this signal is an input to the Z80 and results in a maskable interrupt which directs the Z80 to respond to some external event.

SPINDIS

Allows disabling of spinner interrupts by the hand controllers. Active low.

Audio Out

Audio In

AUX VID

VID GATE

CLK, RSTDIS

SEL4, SEL2

HALT, NMI

AUXDECODE2

VIDRST

These signals are not used on the Memory and I/O Board but are made available at the expansion connector.

2.1.9b Interconnects for Connector #1 at J7

BD0-BD7	8 bidirectional data lines. BD0 is least significant, BD7 is most significant.
BA0-BA7	Address lines to Memory and I/O Board. BAO is least significant, BA7 is most significant.
$\overline{\text{BWR}}$	Output of Z80 to Memory and I/O Board; write strobe used to output data during an I/O or memory operation. Indicates a write operation.
$\overline{\text{BRD}}$	Output of Z80 to Memory and I/O Board; read strobe used to clock data into the Z80 during an I/O or memory operation.
$\overline{\text{IORQ}}$ $\overline{\text{BIORQ}}$	Same as $\overline{\text{BMREQ}}$, but indicates an I/O operation instead of memory or memory-mapped devices.
$\overline{\text{BMI}}$	Output of Z80 from CPU or Coleco-vision Board; indicates the present memory cycle is an opcode fetch (start of next instruction).
$\overline{\text{INT}}$	Active low, this signal is an input to the Z80 and results in a maskable interrupt which directs the Z80 to respond to some external event.

2.1.9c Interconnects for Connector #2 at J6

BD0-BD7	8 bidirectional data lines. BD0 is least significant, BD7 is most significant.
BA0-BA15	16 address lines to Memory and I/O Board. BAO is least significant, BA15 is most significant.
$\overline{\text{BWR}}$	Output of Z80 to Memory and I/O Board; write strobe used to output data during an I/O or memory operation. Indicates a write operation.
$\overline{\text{BRD}}$	Output of Z80 to Memory and I/O Board; read strobe used to clock data into the Z80 during an I/O or memory operation.
$\overline{\text{BMREQ}}$	Output of Z80 to Memory and I/O Board; indicates present read or write operation is

directed to memory or memory-mapped devices.

IORQ
BIORQ Same as BMREQ, but indicates an I/O operation instead of memory or memory-mapped devices. IORQ is unbuffered; BIORQ is buffered.

BMI Output of Z80 from CPU or Coleco-vision Board; indicates the present memory cycle is an opcode fetch (start of next instruction).

INT Active low, this signal is an input to the Z80 and results in a maskable interrupt which directs the Z80 to respond to some external event.

AUDIO IN

2.1.9d Interconnects for Connector #3 at J5

BD0-BD7 8 bidirectional data lines. BD0 is least significant, BD7 is most significant.

BA0-BA15 Address lines to Memory and I/O Board. BA0 is least significant, BA15 is most significant. RA7 is substituted for BA7.

BWR Output of Z80 to Memory and I/O Board; write strobe used to output data during an I/O or memory operation. Indicates a write operation.

BRD Output of Z80 to Memory and I/O Board; read strobe used to clock data into the Z80 during an I/O or memory operation.

2.1.9e Other Memory and I/O Board Connections

J2 and AdamNet Connections - The following signals are found on the
J8 AdamNet connectors for keyboard and expansion devices.

Data - 62.5K bps serial 'bidirectional' line for data transmission reception by network devices.

Reset - hardware network reset

+5V

Signal Ground

J9 Power Supply/Printer Connector - In addition to containing the signals found on J2 and J8, the necessary power supply voltages of +12V Logic, +12V Inductive, and -5V connect here.

J10 and Data Drive Connectors - For a detailed description of
J12 the signals found on the data drive connectors, refer to Subsection 2.3.5.

J1 Cartridge Connector

<u>Pin</u>	<u>Type</u>	<u>Pin</u>	<u>Type</u>	<u>Pin</u>	<u>Type</u>	<u>Pin</u>	<u>Type</u>	<u>Pin</u>	<u>Type</u>
1	D2	7	A0	13	RF ground	19	A13	25	A7
2	CS3*	8	D5	14	A11	20	A14	26	A9
3	D1	9	A1	15	A3	21	A5	27	CS4*
4	D3	10	D6	16	A10	22	CS2*	28	A8
5	D0	11	A2	17	A4	23	A6		
6	D4	12	D7	18	CS1*	24	A12		

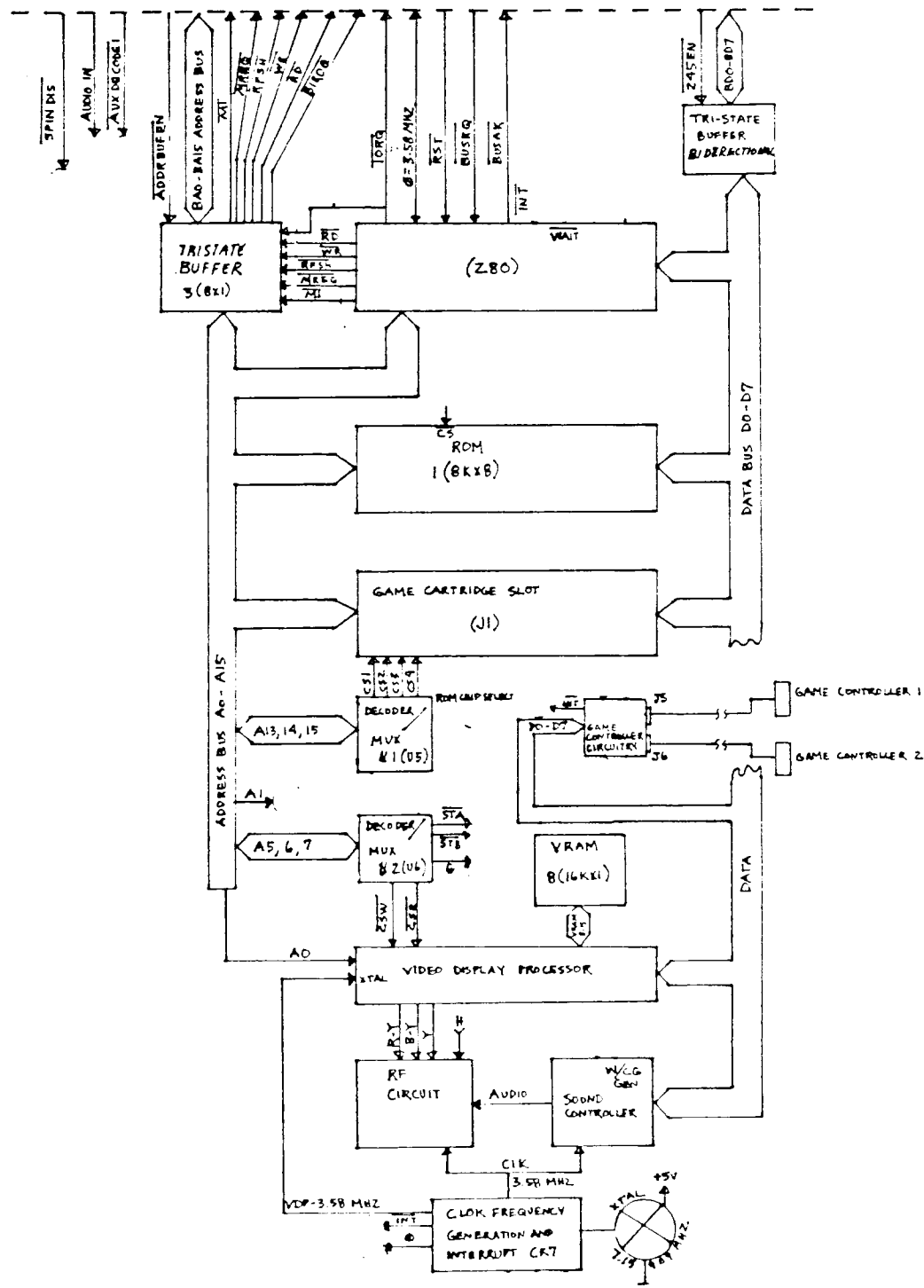
30 +5V Typical available current 0.2A

29 Digital Ground

*LS138 Decoder output. Refer to Table 3.

TABLE 3: CARTRIDGE INTERFACE PARAMETERS

Sym bol	Parameter	Conditions	Min Typ Max	Units
V _{OH}	High level output voltage	V _{CC} = Min, V _{IH} = 2V V _{IL} = V _{IL} Max, I _{OH} = - 400 uA	2.7 3.4	V
V _{OL}	Low level output voltage	V _{CC} = Min, V _{IH} = 2V I _{OL} = 8 mA	0.35 0.5	V
		V _{IL} = V _{IL} Max I _{OL} = 4 mA	0.25 0.4	
V _{IH}	High level input voltage		2	V
V _{IL}	Low level input voltage		0.8	V



2.2.2 Z80 Microprocessor

The Z80 CPU, which consists of a Z80A microprocessor and a clock circuit for synchronization, has control of the Adam computer system. The Z80 configures the memory map and can switch banks of memory. Refer to Chapter 3, Section 2 for details on the memory configuration.

2.2.3 ROM Circuitry

The CPU Board includes an 8K operating system ROM (OS_7) and a connector for up to 32K of cartridge ROM.

2.2.4 Video Display Processor (VDP)

The Video Display Processor, a Texas Instruments (TI) 9928, generates all video, control and synchronization signals and controls the storage, retrieval and refresh of display data in a dynamic memory, VRAM. The 9928 uses a table-driven architecture that allows the programmer to control every pixel in the visual display area, and to define and control 32 "sprites." Sprites may be placed anywhere on the display and moved at will.

The VDP has three major interfaces: CPU, RF modulator, and VRAM. The VDP is addressable in data mode (used when VRAM is being written or read) and register mode (used when control information is being written to and read from one of the VDP's internal registers). The addresses of the ports in the CPU I/O address space are as follows:

Data Port	...OBEH
Register Port	...OBFH

The video RAM circuit consists of 8 (16384 x 1) RAM integrated circuits. The contents of VRAM define the TV image. AO, CSW and CSR are CPU-controlled input signals to the VDP that control when the data is written to or read from VRAM. The VDP output signals R/W, CAS and RAS control the RAM operation.

Data can be transmitted to or from the CPU over the data bus, depending on the state of the Chip Select Write (CSW) and Chip Select Read (CSR) control lines. When CSW is low, data is transmitted from the CPU to the Video Display Processor. When CSR is low, data is transmitted from the Video Display Processor to the CPU. CSR and CSW should not be simultaneously low.

Another control line, address line AO, determines where the VDP retrieves or sends data. If AO is in a high state, the

data is stored into, or retrieved from an internal register. The register used is determined by the data. If A0 is in a low state, the data is stored into or retrieved from the VRAM.

Refer to the Texas Instruments TMS9918A/TMS9928A/TMS9929A Video Display Processors Data Manual for further information.

2.2.5 Sound Generator

The system uses a TI 76489 (6496) sound generator controller to produce sounds. The chip contains three programmable tone generators, a programmable white-noise generator, and programmable attenuation for each of the channels. The chip is addressed through a single write-only port at location OFFH. Wait-request hardware has been included in the system because the sound chip is a slow peripheral requiring data lines to be stable for a relatively long time while it is receiving data.

2.2.6 RF Circuitry

The RF modulator uses the 1889 chip to interface audio, color difference, and luminance signals to the antenna terminals of a TV receiver. It consists of two VHF channels, 3 or 4, selectable by a slide switch with determined LC tank circuits. The Chroma subcarrier is derived from the 3.58 MHz system clock to ensure accuracy and stability. The sound oscillator's frequency modulator is achieved by using a 4.5 MHz tank circuit and deviating the center frequency via a varactor diode. Due to the incompatible signal level between the VDP 9928 and the 1889, a DC restoration circuit ensures the DC level of the video signal.

The $\overline{R-Y}$, $\overline{B-Y}$, and Y signals from the VDP, along with the 3.58 MHz clock and the audio signal from the SN76489 (6496), are provided to the RF modulator to produce the composite video output.

2.2.7 Game Controller Circuitry

The two game controllers are connected to the CPU Board via two "D" type connectors. Each controller is accessed by the system through its own port. See CONT-SCAN in the OS_7 Source Code Listing for details.

For each controller, 18 switches are read on a single 8-bit port. Therefore, once a port has been read, some decoding is required to determine which switches have been depressed.

Two spinner switches that are not wired in the controller are used in some games. To ensure that the spinner switch closures are processed as soon as they happen, they are connected to the CPU maskable interrupt, and the cartridge software determines which switch causes the interrupt.

Controller Connector Pin Out

Pin	Type	Comments
1	Indirect D0 input	Refer to Table 1
2	Indirect D2 input	Refer to Table 1
3	Indirect D3 input	Refer to Table 1
4	Indirect D1 input	Refer to Table 1
5	Strobe signal output, Common 1	
6	Indirect D6 input	Refer to Table 1
7	Indirect D5 input	Refer to Table 1
8	Strobe signal output, Common 0	
9	Indirect INT input	

Strobe signal: typical 350 micro sec pulse width, -0.7V Low, +2.8V high typ.

For further information on the game controllers, refer to Chapter 2, Section 5.

2.2.8 Clock Generation

The system clock is a 3.58 MHz square wave generated by dividing the 7.1 MHz clock by two. The video chip clock (10.7 MHz) drives the Video Display Processor. The video chip clock is obtained from the third multiple, high Q tuned tank circuit on the 3.58 MHz system clock. The 7.1 MHz clock is generated by a crystal controlled oscillator. The output of the oscillator circuit is buffered and divided by two to provide a 50% duty cycle wave form.

2.2.9 Interconnects

The CPU Game Board and the Memory and I/O Board connect via two 30-pin ribbon cables and a dual 30-pin card edge connector, making a pin-for-pin connection between J1 on the Memory and I/O Board and J2 on the CPU Board. Refer to Subsection 2.1.9.

2.3 DATA PACK DRIVE MODULE

2.3.1 Theory of Operation

The data drive assembly provides for two drives: one is included with the system, the other is optional. The data drive is a computer-controlled, digital cassette drive.

The components of the data drive subsystem are located in two places: The Memory and I/O Board contains the tape drive 6801 microcomputer, a quad comparator and RAM. The Read/Write and Servo Boards are located in the data drive assembly.

The data drive 6801 controls the direction of the tape, tape speed, stop, track selection, and the Read/Write operation. In addition, the data drive 6801 monitors the presence of a data pack and transmits and receives data through AdamNet.

The comparator interfaces the data drive 6801 to AdamNet. The RAM circuitry consists of two 1024 x 4 ROM integrated circuits, connected in parallel to provide an 8-bit data bus.

Two printed circuit boards, the Servo Board and the Read/Write Board are located in the data drive assembly.

2.3.2 The Servo Board

The Servo Board controls the direction and speed of the data drive by:

- Controlling the "take-up" motor which pulls tape in the direction of motion (forward or reverse).

- Controlling the "supply" motor which applies a slight pull or drag in the direction opposite to tape motion. This function maintains a stable tape motion.

- Maintaining a constant tape velocity across the head at any position, from the beginning to the end of the tape.

- Providing two different tape speeds; 20 inches per second (slow), and 80 inches per second (fast). Tape speed is controlled by the data drive 6801.

- Providing a brake function that allows the data drive 6801 to stop tape motion in milliseconds. (This function prevents the tape from coasting to a stop.)

- Keeping the tape in tension when the tape is not in motion, preventing slack in the tape.

Providing a signal to the data drive 6801 indicating the status of tape motion.

Providing a signal to the data drive 6801 indicating whether or not a data pack has been properly placed in the data drive.

2.3.3 The Read/Write Board

The Read/Write Board records digital data encoded in bi-phase mark format on two separate tracks on the tape. This board also plays back the data recorded on the two tracks, with data output in the same format as recorded. The data drive 6801 selects the tracks.

The bi-phase mark technique embeds the clock in the data line. When data is returned from the data drive to the data drive 6801, the data line is coded back to the standard binary format.

A "cassette-in-place" switch located on the data drive chassis lets the Servo Board know when the data pack is in place and ready for use. An optical encoding wheel interacts with the Servo Board and the tape for drive speed control.

2.3.4 Data Pack Specifications

The magnetic tape in the data pack is standard two-track digital recording tape, 300 feet in length by .15 inches in width.

Effective data transfer rate	1.4K bytes per second
------------------------------	-----------------------

Tape speed	
Normal	20 inches per second
Fast forward/rewind	80 inches per second

Tape capacity	256K bytes
---------------	------------

	Two tracks, 128 blocks per track 1 block = 1K
--	---

2.3.5 I/O Signals between Memory and I/O Board and Data Drive

<u>Signal Name</u>	<u>Mem/I/O Board</u>	<u>R/W I/O</u>	<u>Description</u>
(input) BRAKE*	J10-1	E26	Brakes tape motion. Logic 1 (active high) applies brake. This signal is passed from the Read/Write (RW) Board to the Servo Board at point E6.
(input) GO REV	J10-2	E25	Commands reverse direction of tape motion when at Logic 0 (active low). This signal is passed from R/W Board to Servo Board at point E8.
(input) GO FWD	J10-3	E27	Commands forward direction of tape motion when at Logic 0 (active low). This signal is passed from R/W Board to Servo Board at point E8.
(input) STOP	J10-4	E28	Prevents tape motion and latches data drive output to Logic 1 when at Logic 1 (active high). Enables data drive output and tape motion when at Logic 0. This signal is used in circuits on both boards. It is passed from R/W Board to Servo Board at point E9.
(input) SPEED SELECT	J10-5	E29	Selects speed of tape motion; Logic 0 = 20 ips (slow speed), Logic 1 = 80 ips (fast speed). This signal is passed from R/W Board to Servo Board at E10.
GND	J10-6	E30	Return path for all logic and analog signals. Connected from R/W Board to Servo Board at point E19.

2.3.5 I/O Signals between Memory/I/O Board and Data Drive (continued)

<u>Signal Name</u>	<u>Mem/I/O Board</u>	<u>R/W I/O</u>	<u>Description</u>
(output) MSENSE	J10-7	E31	Provides sense of tape motion status for tape drive 6801 (active high). When Logic 1, tape is properly in motion. When Logic 0, tape is not moving due to stop/braking action or malfunction. This signal is passed between R/W Board and Servo Board at point E12.
(input) +12VI	J10-8	E32	+12V Inductive Power Supply line, used to power motor circuitry. Passed from R/W to Servo circuit at point E12.
(output) CIP	J10-9	E33	Indicates to tape drive 6801 that a data pack has been properly inserted in drive when Logic 0 (active low). Passed between R/W and Servo Boards at point E14.
(input) DATA IN	J11-1	E17	Data input to drive from tape drive 6801. Data is encoded in bi-phase mark format. Each bit cell is 70 micro-seconds in duration. A Logic 1 is denoted by a flux change in the bit cell. A Logic 0 is denoted by no flux change in the bit cell. This data is input to the drive in a serial stream.
(input) TRACK A/ $\overline{B}$	J11-2	E18	Selects recording track for read or write operation. Logic 1 selects Track A; Logic 0 selects Track B.
GND	J11-3	E19	Return path for all analog and logic signals.

2.3.5 I/O Signals between Memory/I/O Board and Data Drive (continued)

<u>Signal Name</u>	<u>Mem/I/O Board</u>	<u>R/W I/O</u>	<u>Description</u>
(input) +5V	J11-4	E20	+5V is passed to Servo Board at point E3.
(output) DATA OUT	J11-5	E21	Data output from drive to tape drive 6801. Data is encoded in bi-phase mark format as described in DATA IN signal. Jitter in signal is specified at 4% maximum, peak shift at 5% maximum.
(input) +12VL	J11-6	E22	+12V is passed from R/W to Servo Board at point E15.
(input) <u>WRENABLE</u>	J11-7	E23	Selects write mode (Logic 0) or read mode (Logic 1) of R/W circuit operation (active low).
	J11-8	E24	No connection.

2.4 Differences of Expansion Module #3

Expansion Module #3 is designed for the consumer who owns ColecoVision. It consists of the Memory Console, the keyboard, the printer, and various cords and cables to connect the components. The Memory Console attaches to the ColecoVision Console. The consumer provides his own TV. The major difference between Expansion Module #3 and the complete Adam Home Computer System is in the Memory Console. The equivalent of the CPU Board is housed in the ColecoVision Console, not in the Memory Console. Expansion Module #3 does not provide composite video output; therefore a monitor cannot be used.

Expansion Module #3 is not packaged with "joystick" game controllers or the antenna switch box, since these items come with ColecoVision.

The ColecoVision Board and the Memory and I/O Board connect via two 30-pin ribbon cables, a dual 30-pin card edge connector and the Interconnect Board, making a connection between J1 on the Memory and I/O Board and the expansion port of the ColecoVision console. Many of the connections are made pin-for-pin. The exceptions are:

- | | |
|-----------------|---|
| A0 - A15 | These lines are buffered. The input to the buffer is from the ColecoVision and the <u>tristate control of the buffer</u> (<u>ADDRBUFEN</u>) is from the Memory and I/O Board. |
| D0 -D7 | These <u>lines</u> are buffered bidirectionally. The <u>RD</u> signal from the ColecoVision controls the direction (when active, data <u>flows</u> toward the ColecoVision) and the <u>245EN</u> signal controls the tristate function. |
| <u>IORQ</u> | This line is connected directly between the two boards (pin 55) and <u>is also buffered</u> (controlled by <u>ADDRBUFEN</u>) and connected to pin 40 of the Memory and I/O Board. |
| <u>Q</u> 'CLOCK | This line, pin 40 of the ColecoVision Board, is connected to pin 45 of the Memory and I/O Board. |
| <u>Q</u> CLOCK | This line, pin 45 of the ColecoVision, is not connected. |

M1, MREQ,
RFSH, WR
RD

These lines are buffered. The input to the buffer is from the ColecoVision, and the tristate control of the buffer (ADDRBUFEN) is from the Memory and I/O Board.

3. THE KEYBOARD

3.1 Theory Of Operation

The keyboard is the major input device through which the user communicates with the system. The game controller joystick, buttons and keypad can also be used to input information. The keyboard consists of two major subsystems. The external subsystem is an array of keys much like that of a mechanical typewriter. The internal subsystem, located on the Keyboard Printed Circuit Board, includes the keyboard 6801 microcomputer and the AdamNet serial interface.

The keyboard contains 75 full travel keys, including special function keys. Some special function keys are labelled on the key top, for example, STORE/GET, DELETE, BACKSPACE. The function keys on the top row of the keyboard are smart keys, marked I through VI. Corresponding smart key labels shown on the video display identify their functions.

The keyboard 6801 determines which keys the user presses through the keyboard matrix. Every 5 or 8 milliseconds, the keyboard 6801 scans the matrix and stores the characters in a buffer.

The matrix consists of vertical columns which correspond to keyboard 6801 output lines and horizontal rows which correspond to keyboard 6801 input lines.

A debounce function determines if any of the keys registers a low signal; then that row is read horizontally. More than one row in a column is read if more than one key in that column is depressed. The ASCII code for the depressed key or keys is read back to the 6801. The keyboard 6801 compares the code with the code input from the previous scan. If it is the same, the information is recorded; if it is different, the information is not recorded.

The keyboard 6801 responds to the following commands from the master 6801:

- Reset - the keyboard 6801 erases the character buffer and resets shift lock to the unlocked state.

- Send character - the keyboard 6801 sends one character via the RxD/TxD line to the master 6801.

3.2 Interconnects

The lines linking the keyboard 6801 to AdamNet are:

Signal ground
+5V (input)
Reset (input)
RxD/TxD (input/output)

4. THE PRINTER

4.1 Theory of Operation

The printer houses two printed circuit boards: the Printer Board and the Linear Power Supply Board. The Printer Board includes a 6801 microcomputer and parallel drivers that control the printer's electro-mechanical devices. The electro-mechanical devices include the daisy wheel motor, the carriage motor, the platen advance mechanism, the print solenoid, and the ribbon solenoid. The Linear Power Supply Board includes regulators, rectifiers, and a transformer. For more information on the power supply, refer to Chapter 2, Section 6.

4.2 The Printer Board

The printer 6801 communicates with the Master 6801 on the Memory and I/O Board via AdamNet to receive data to be printed, control the motions of the printer's mechanical parts, and ensure that the printer performs the optimum number of motions simultaneously.

Within the printer 6801 RAM is a 16-character buffer for data being sent to the printer over AdamNet. The buffer ensures that time is not lost between characters being printed, and maximizes printing speed.

The printer 6801 responds to some ASCII control codes including carriage return, line feed, backspace, escape, shift-out and shift-in. Shift-out causes the printer 6801 to reverse its left and right directions, allowing printing from right to left.

The printer 6801 controls the printer's electro-mechanical devices. It also ensures that lateral carriage motion, rotation of the daisy wheel and stepping of the platen can be activated simultaneously.

4.3 Interconnects

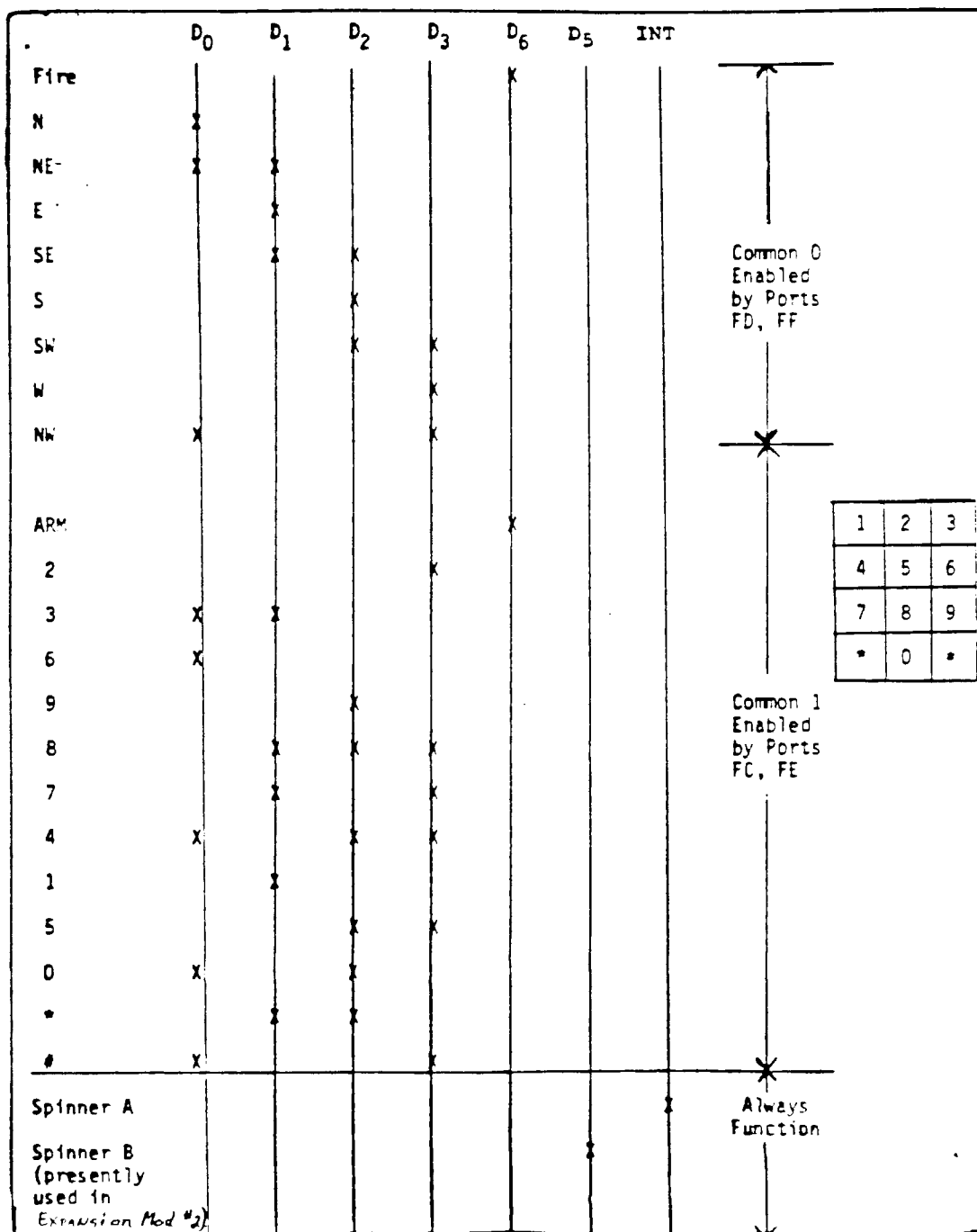
The lines linking the printer 6801 to AdamNet are:

- Signal ground
- +5V (input)
- Reset (input)
- RxD/TxD (input/output)

5. GAME CONTROLLERS

The game controller contains an 8-position joystick, two push buttons and a 12-key keypad. The information from a controller is read by the CPU on eight input lines through a single port. Once a port has been read, the input data must be decoded. See CONT_SCAN in the OS_7 Source Code Listing for details.

FIGURE 2-5: CONTROLLER CONFIGURATION



6. POWER SUPPLY

6.1 Power Supply Voltage

The power supply for the ADAM computer is located in the printer. The power supply converts the incoming line voltage (AC) to one 18V unregulated voltage that powers the ribbon solenoid and four low level, regulated DC voltages as follows:

+5v	Main source of power to the CPU
-5V	Supplies power to the CPU
+12VI	Supplies power to drive the inductive loads such as carriage motor, daisy wheel motor, print solenoid, platen motor and digital data drive.
+12VL	Supplies power to the system logic.

6.2 Excessive Current Output Protection

The power supply uses a variety of methods to protect against excessive current output.

The +5V and the +12VI are fused and use electronic fold-back current limiting.

The +12VL is not fused but uses electronic foldback limiting.

The -5v uses conventional current limiting and thermal protection which halts the current when the regulator gets too hot.

The 18V unregulated uses the same fuse as the +12VI.

A thermal fuse in the power transformer protects against overcurrent at the transformer.

The AC line input may vary from 108VAC to 132VAC. The power supply ensures a constant and quiet source of DC power.

6.3 Printer/Memory Console Interface Cable

The printer/console interface cable consists of 7 insulated wires and one uninsulated drain wire.

<u>Pin</u>	<u>Color</u>	<u>Voltage/Description</u>
1	Brown	+12L VDC +.508V -.6V
2	Red	+12I VDC +.497V -.6V
3	Orange	+5.075 VDC +.079V -.255V
4	Yellow	-5.15VDC +.25V
5	Green	Ground
6	Blue	AdamNet
7	Violet	Reset
8	--	Drain
9	--	No wire

6.4 Power Supply Output to CPU (via Printer/Memory Console Interface Cable)

<u>Voltage</u>	<u>Full Load Current</u>
+5V	2.75A
-5V	0.2A
+12VI	0.6A
+12VL	0.3A

6.5 Power Supply Output to Printer

<u>Voltage</u>	<u>Full Load Current</u>
+5VL	0.25A
+12VI	1.95A
+18V (unreg)	1.0A

CHAPTER 3: SOFTWARE

1. INTRODUCTION

This chapter presents technical information on each of ADAM's software components. Emphasis is placed on AdamNet and the available operating systems. Descriptions of application software are provided as examples to software developers.

Source code listings for EOS and OS_7 can be requested with the form on the last page of this manual.

2. MEMORY MAP AND POWER UP/RESET PROCEDURE

Adam's Z80 microprocessor can address 64K bytes at any one time. The 64K addressable memory space is divided into two 32K sections. Each section may contain one of four memory options. Any one option for lower memory and any one option for upper memory can be selected for the full 64K memory space via port 7FH.

Memory Options for 0 - 7FFFH (lower memory)

SMART WRITER WORD PROCESSOR ROM	32K INTRINSIC RAM	32K EXPANSION RAM	OS 7 24K INTRINSIC RAM
---	-------------------------	-------------------------	---------------------------------

Memory Options for 8000H - FFFFH (upper memory)

32K RAM	32K EXPANSION ROM	32K EXPANSION RAM	32K CARTRIDGE ROM
------------	-------------------------	-------------------------	-------------------------

2.1 Lower Memory Options

SmartWRITER Word Processor ROM - This memory option consists of 32K of SmartWRITER ROM code. A small part of this code, EOS_BOOT, is responsible for system initialization during power up and reset. EOS ROM can also be accessed when this option is selected. See Subsection 4.1, EOS, for further details.

32K Intrinsic RAM - This option is the lower half of the 64K RAM included with every ADAM. DMA transfers to AdamNet can take place only in intrinsic RAM. SmartBASIC and most programs stored on data pack reside in this memory.

32K Expansion RAM - This option is the lower half of the 64K Memory Expander, an optional feature not included with the ADAM system. The 64K Memory Expander increases ADAM's memory to 144K of read/write memory (64K intrinsic, 64K expansion, 16K VRAM).

OS 7 and 24K Intrinsic RAM - This option contains OS_7 and 24K of ADAM's intrinsic RAM. OS_7 is the 8K ROM installed in ColecoVision and ADAM. In Expansion Module #3, this ROM is in the ColecoVision. The description of the 32K Intrinsic RAM also applies to this 24K intrinsic RAM.

2.2 Upper Memory Options

32K Intrinsic RAM - This option is the upper half of the 64K intrinsic RAM included with ADAM. DMA transfers to AdamNet can take place only intrinsic RAM. SmartBASIC and most programs stored on data pack reside in this memory.

Expansion ROM - This memory is provided by an expansion ROM, an optional feature not included in the ADAM system. The expansion ROM is installed in Connector #2 on the Memory and I/O Board. EOS_BOOT checks this connector for valid data before initializing EOS. If valid data is found, the EOS_BOOT code jumps to this ROM.

32K Expansion RAM - This is the upper half of the optional Expansion RAM described for lower memory.

32K Cartridge ROM - This memory option is the cartridge slot on ADAM or ColecoVision, used to execute game cartridges or other cartridge-based software.

2.3 Power Up/Computer Reset Procedure

When Adam powers up or when the computer reset switch is pressed, the MIOC selects SmartWRITER in lower memory and Intrinsic RAM in upper memory. EOS_BOOT executes this procedure:

Check for Expansion ROM. If Expansion ROM exists, jump to Expansion ROM.

Else, initialize EOS and jump to EOS_START

Check for the presence of devices in this order:

Disk Drive 1
Disk Drive 2
Data Pack Drive 1
Data Pack Drive 2

If a device exists, check for valid data in the device.

Boot and execute code from the first valid device found by loading block 0 to address 0C800H then jumping to 0C800H.

If no valid data is found on any device, execute SmartWRITER.

2.4 Z80 I/O Port Assignments

<u>Port</u>	<u>Description</u>
00H through 1DH	Reserved
1EH	Optional Auto Dialer
1FH	Reserved
20H through 3EH	Reserved
3FH*	Network reset; EOS enable
40H through 4EH	Reserved
4FH	Expansion connector #2
50H through 5DH	Reserved
5EH	Optional Modem Data I/O
5FH	Optional Modem Control Status
60H through 7EH	Reserved
7FH	Memory Map Control
80H through FFH	Reserved for ColecoVision use

*Net reset - The net reset function is performed by setting bit 0 and then resetting bit 0.

*EOS enable - Setting bit 1 enables the EOS ROM. Resetting bit 1 disables EOS ROM. The EOS enable function only affects the SmartWRITER ROMs. To access the EOS ROM, the SmartWRITER ROMs must be selected.

For further details on port assignments, see PORT_COLLECTION in the EOS Source Code Listing.

2.5 Memory Map Control

Software can select the memory configuration by writing to Port 7FH. Data bits D0 and D1 select the lower (0 - 7FFFH) memory option. D2 and D3 select the upper (8000H - FFFFH) memory option. D4 through D7 are reserved for future expansion, and should remain as 0. The value to be written to Port 7FH is obtained from the following tables.

Lower Memory Option Selection

0	0	SmartWRITER ROM and EOS ROM
0	1	32K Intrinsic RAM
1	0	32K Expansion RAM
1	1	OS7 + 24K Intrinsic RAM

Upper Memory Option Selection

0	0	32K Intrinsic RAM
0	1	Expansion ROM
1	0	Expansion RAM
1	1	Cartridge ROM

2.6

Reset Procedures

Adam can be reset in either computer mode or in game mode. When the computer reset switch is pressed, Adam resets to computer mode, according to the power up procedure described in Subsection 2.3.

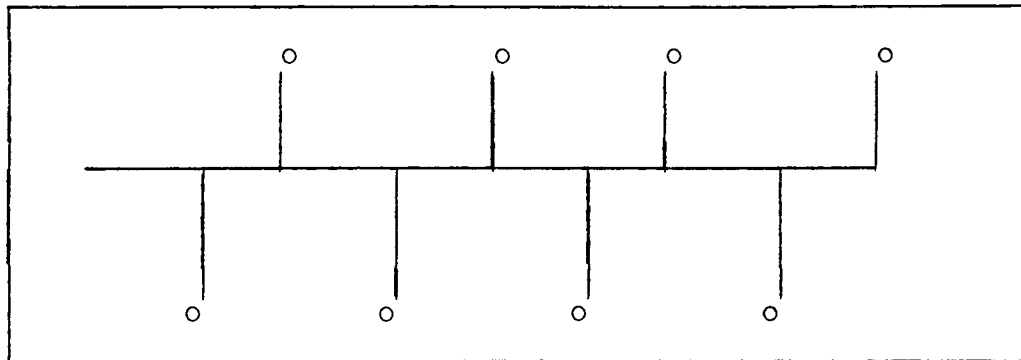
When the cartridge (or ColecoVision) reset switch is pressed, Adam resets to game mode. 32K of Cartridge ROM are switched into upper memory. OS-7 plus 24K of intrinsic RAM are switched into the lower bank of memory. Refer to Subsection 9.1 for the game mode memory map.

3. ADAMNET

3.1 Introduction

AdamNet is based physically on a shared bus, single master topology, although logically it resembles a token-passing network. The physical design of the network is depicted in Figure 3-1.

FIGURE 3-1: BUS NETWORK



The circles represent nodes (for example, printers or keyboards), which connect to a shared bus, represented by the horizontal line. The network is a four wire bus. The wire definitions are:

- Data
- Reset
- Signal ground
- Power

The network resides in all Adam components. The reset line behaves as a master reset signal to every node attached to the network. A transition on this line causes all attached devices to enter the power-on state, allowing the master to bring sanity to the network during periods of erratic behavior.

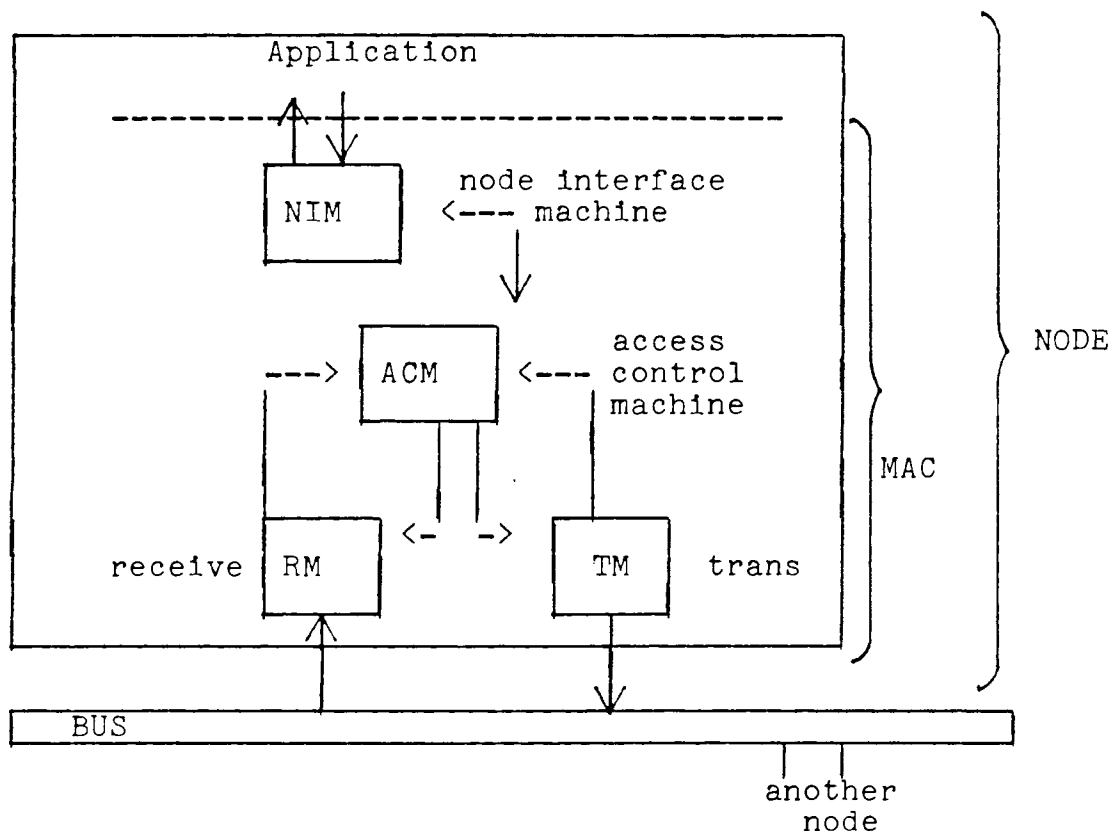
Logical Design

Logically, AdamNet resembles a token-passing network. In a token-passing network, the right to talk on the bus is passed from node to node. This right, or "token", allows the node to access the bus and send messages. One node functions as the master of the bus. The master gives the token to other nodes with a regulated frequency.

3.2 Network Master Concept

Conceptually the master is like any other node on the network. Each node contains an IEEE defined Medium Access Control (MAC) layer of software that enables the node to gain access to the network. The MACs in non-master nodes contain a subset of the functions in the master's MAC. Internally all MACs are composed of four loosely coupled logical machines: Node Interface Machine (NIM), Access Control Machine (ACM), Receive Machine (RM) and Transmit Machine (TM).

FIGURE 3-2: MAC Internals



The MAC's version of the Access Control Machine is significantly different from that of the non-master nodes. The NIM, RM and TM components in the master and non-master nodes are basically equivalent. The RM and TM are physical level I/O routines that accept and send data frames.

3.3 Message Philosophy and Definition

AdamNet messages can be divided into two types: commands and responses. Each type can be subdivided into either data or

control sub-types. Commands are messages initiated by the master, while responses are initiated by non-master nodes. The following four possibilities exist:

- command.control
- command.data
- response.control
- response.data

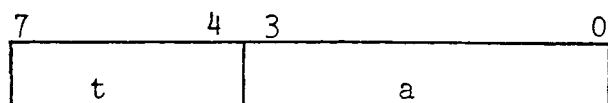
The format of a basic message follows:



Where: a = device address
s = subtype (indicates the nature of the message)

The messages are defined in the following pages.

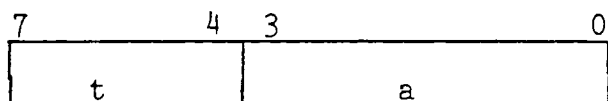
3.3a. COMMAND.CONTROL (RESET)



a = target address (1..15)

t = RESET (0)

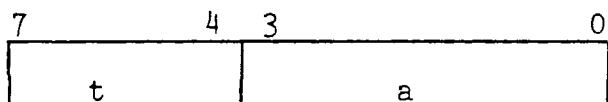
3.3b. COMMAND.CONTROL (STATUS)



a = target address (1..15)

t = STATUS (1)

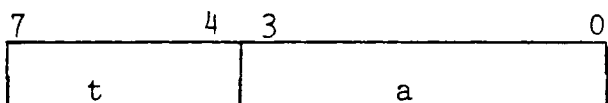
3.3c. COMMAND.CONTROL (ACK)



a = target address (1..15)

t = ACK (2)

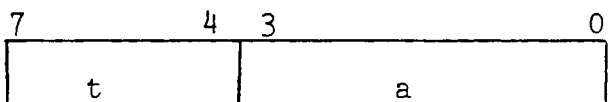
3.3d. COMMAND.CONTROL (CLR)



a = target address (1..15)

t = CLEAR (3)

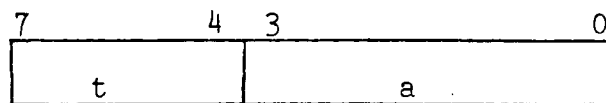
3.3e. COMMAND.CONTROL (RECEIVE)



a = target address (1..15)

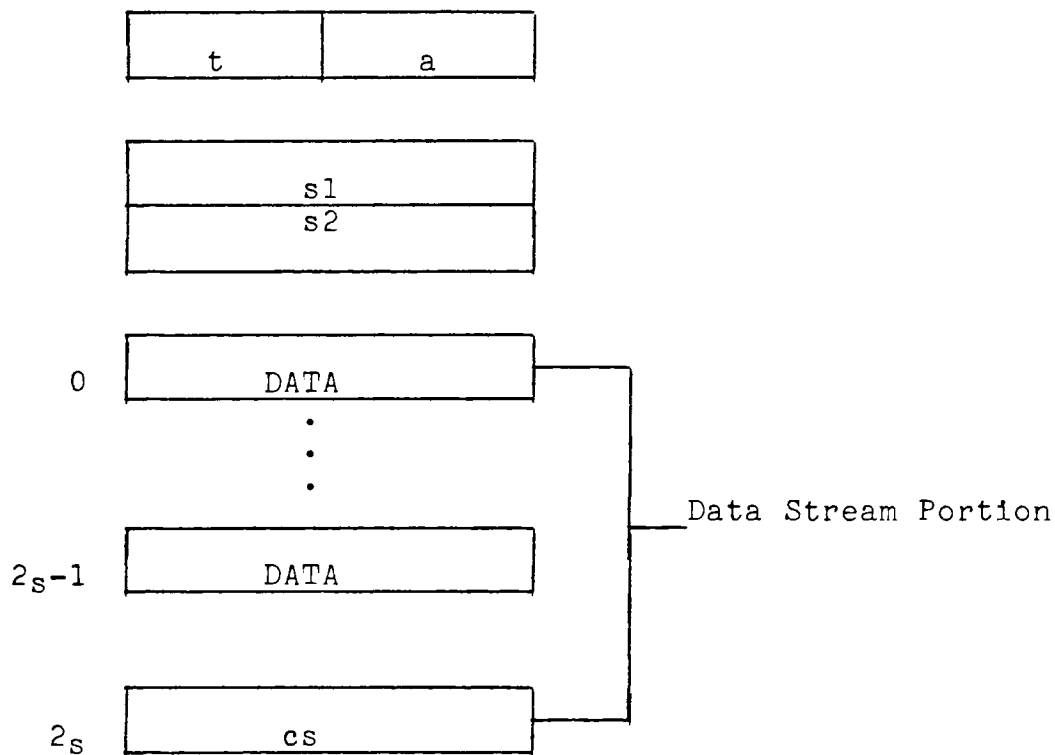
t = RECEIVE (4)

3.3f. COMMAND.CONTROL (CANCEL)



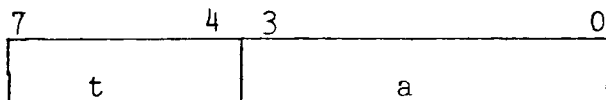
a = target address (1..15)
t = CANCEL (5)

3.3g. COMMAND.DATA (SEND)



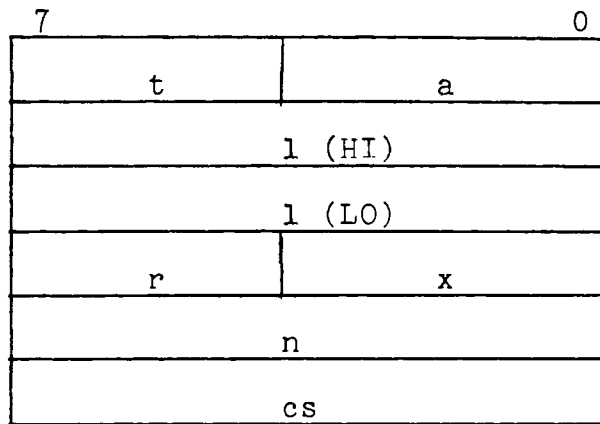
a = target address (1..15)
t = SEND (6)
s1,2 = MAX message size.
DATA = data byte
cs = check sum

3.3h. COMMAND.CONTROL (NACK)



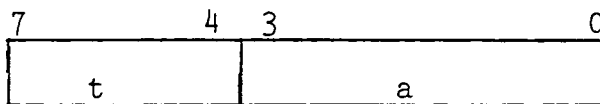
a = target address (1..15)
t = NACK (7)

3.3i. RESPONSE.CONTROL (STATUS)



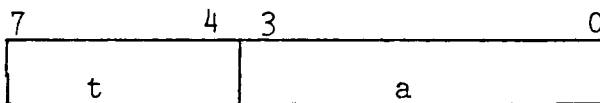
a = source address (1..15)
 t = STATUS (8)
 x = transmit code: 0 - character mode (0-255)
 1 - block mode
 r = reserved
 n = device deependent status
 l = MAX message size
 cs = check sum of bytes 1 - 4

3.3j. RESPONSE.CONTROL (ACK)



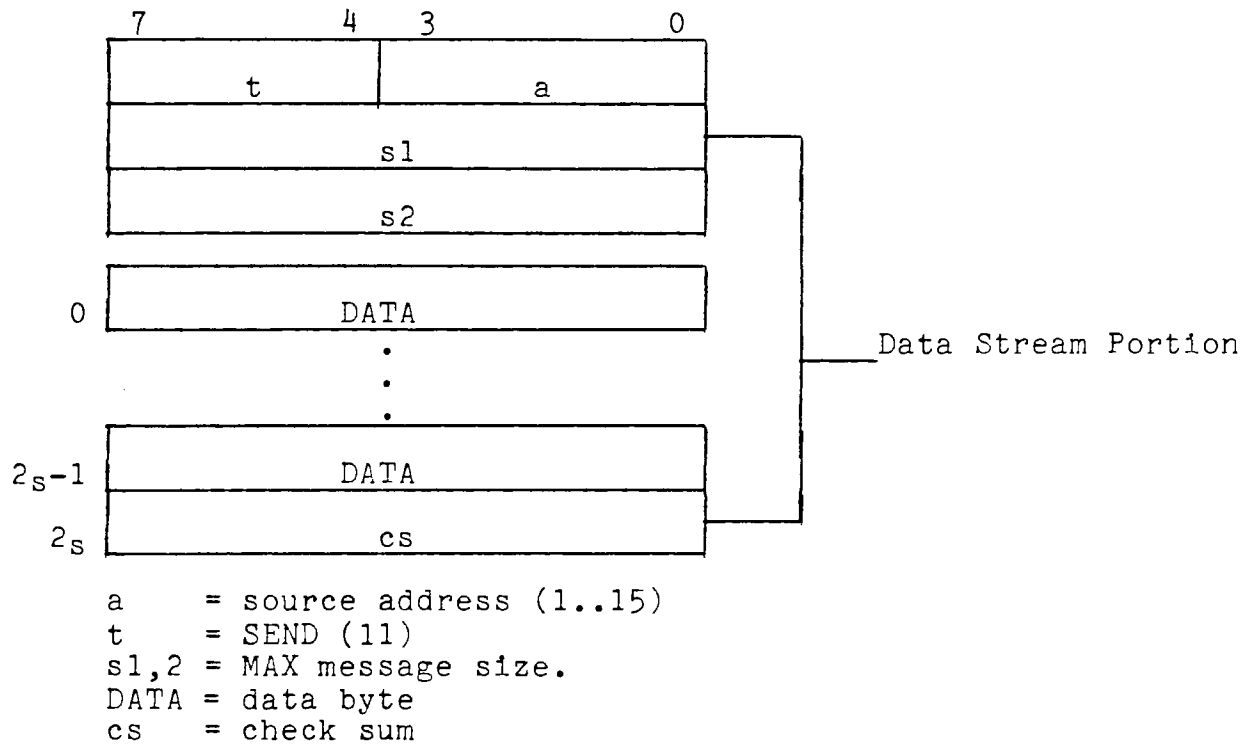
a = source address (1..15)
 t = ACK (9)

3.3k. RESPONSE.CONTROL (CANCEL)

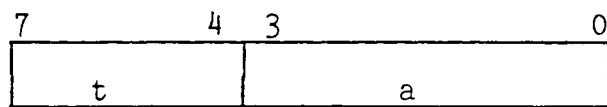


a = source address (1..15)
 t = CANCEL (10)

3.31. RESPONSE.DATA (SEND)



3.3m. RESPONSE.CONTROL (NACK)

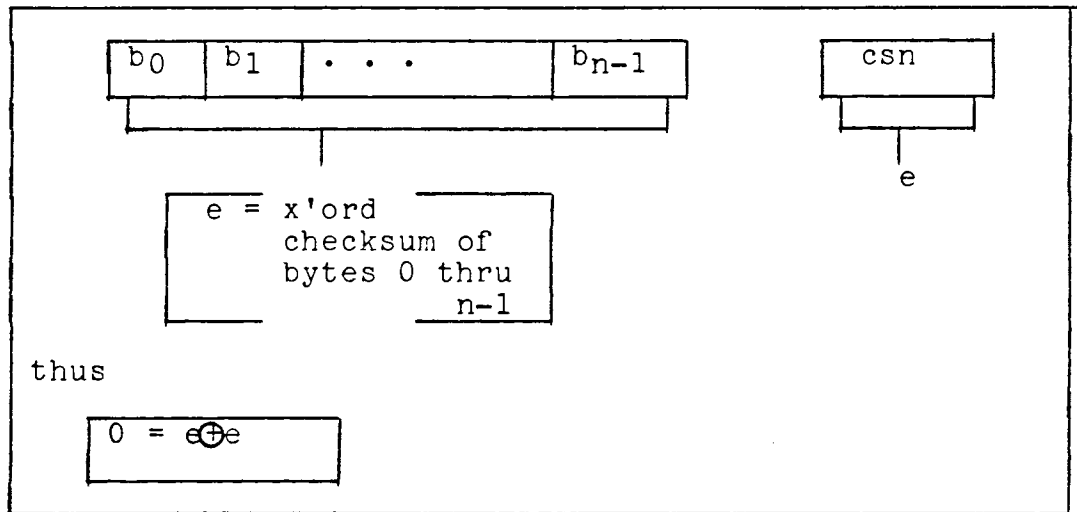


a = source address (1..15)
t = NACK (12)

3.4 Error Control

Data messages use an exclusive-or checksum to verify the integrity of the data. More specifically, for a $n-1$ byte data stream, an n th byte is added which is equal to the exclusive or'd checksum of the zero'th through the $n-1$ byte of the data stream. This insures that the checksum (exclusive or'd) of the zero'th through the n th byte equals zero.

FIGURE 3-3: ERROR CONTROL



A retry method is implemented by the Master to determine whether the lack of a response from a node is due to either a busy or failure condition.

3.5 Class of Service Concept

Each node is assigned a class of service. Class of service is defined as:

maximum message length
node type

To accept a device onto the network the CPU queries the device with a "COMMAND.CONTROL (STATUS)." In turn, the target device responds with a "RESPONSE.CONTROL (STATUS)" which entails the delivery to the host of a multi-byte package. This package defines to the host the nature and attributes of the responding device.

3.6 Power Up/Initialization

The CPU asks each attached device for its class of service message at power up. The CPU is responsible for associating

a logical unit address to each physical port on the bus. See EOS for device assignments.

3.7 Link Speed

The baud rate is 62.5k. This translates to a bit cell time of 16us. Thus, the time to transmit one byte (including a start and stop bit) is 160us.

3.8 Master Node Interface

AdamNet communicates with the Z80 through a message passing system. The Z80 RAM contains a group of data structures called DCBs (Device Communication Blocks). Every device on AdamNet is associated with a DCB. The structure of a DCB follows:

DCB_CMD_STAT	EQU	0
DCB_BA_LO	EQU	1
DCB_BA_HI	EQU	2
DCB_BUF_LEN_LO	EQU	3
DCB_BUF_LEN_HI	EQU	4
DCB_SEC_NUM_0	EQU	5
DCB_SEC_NUM_1	EQU	6
DCB_SEC_NUM_2	EQU	7
DCB_SEC_NUM_3	EQU	8
DCB_DEV_NUM	EQU	9
*reserved	EQU	10
*reserved	EQU	11
*reserved	EQU	12
*reserved	EQU	13
DCB_RETRY_LO	EQU	14
DCB_RETRY_HI	EQU	15
DCB_ADD_CODE	EQU	16
DCB_MAXL_LO	EQU	17
DCB_MAXL_HI	EQU	18
DCB_DEV_TYPE	EQU	19
DCB_NODE_TYPE	EQU	20

A DMA mechanism allows the 6801 master to scan the DCBs for pending work. The 6801 master writes completion messages and data to the DCBs or data areas as indicated in DCB messages. The Z80 either deposits commands into the DCBs, or inspects the DCBs for the completion status of previously issued commands.

A single data structure (PCB) allows the Z80 to communicate with the 6801 master.

3.9 Functional Overview - Z80 and 6801 Master

The following table provides a functional overview of the Z80 and 6801 master. The three pages following the table provide further details.

<u>Layer</u>	<u>Responsibilities</u>
Z80	Administer PCB, DCB, and data blocks Signal 6801_Master_App Scheduling and prioritizing
6801_Master_App	Scan DCBs (Events from Z80) Post I/O to 6801Mac Send control signals Collect signals Interpret PCB and DCBs
6801_Master_Mac	Control network Write/read data to/from Z80. MEM based on NIM_BLK

Z80

Inputs

1. Processor_Control_Block (to function #3)
2. Device_Control_Block (to function #3)
3. Memory_Blocks (to function #1)

Functions

1. To manage data for PCB, DCBs and Memory Blocks.
2. To send signals to the 6801_MASTER_APP.
 - a. Processor Commands
 - b. Device Commands
3. To receive signals from the 6801_MASTER_APP.
 - a. Process_Status
 - b. Device_Status
4. To administer the priority of scheduling devices by regulating the frequency within which the Z80_MASTER posts commands to the applicable DEVICE_CONTROL_BLOCK_COMMAND.

Outputs

1. PCB (from functions #4, 2)
2. DCB (from functions #4, 2)
3. MEM BLOCKS (from function #4)

Note: Z80_MASTER must allocate the MAX message length buffer for the sending/receiving node's message.

6801 Master App

Inputs

1. PCB (to functions #2, 3)
2. DCBs (to function #1, 3)
3. NIM_BLKs (to function #4)

Functions

1. To scan the DCBs for changes of state to the DCB(1).
DEVICE_COMMAND field.
2. To scan the PCB.
3. To interpret the scanned information.
4. To receive I/O completion signals from the MAC.
(Deposited in M_SIG).
5. To post I/O requests to the 6801_MASTER_MAC.
6. To post processor status information to the Z80_MASTER.
7. To post DEVICE_STATUS information to the Z80_MASTER.

Outputs

1. PCB (from function #6)
2. DCB (from function #7)
3. NIM_BLK (from function #4)

6801 Master Mac

Inputs

1. NET_IN MESSAGES (to functions #1, 3)
2. NIM_BLK (to functions #1, 2, 5)

Functions

1. To sequence the I/O of the network.
2. To send messages to devices.
3. To receive messages from devices.
4. To signal status information to the 6801_MASTER_APP.
5. To execute 6801_MASTER_APP signals.

Outputs

1. NET_OUT MESSAGES (to functions #1, 2)
2. NIM_BLK (to functions #1, 4)

4. OPERATING SYSTEM4.1 EOS (Elementary Operating System)

This section provides information to allow designers to write programs that operate in an Adam/EOS environment. It describes the organization of the Elementary Operating System (EOS) including file management and executive calls.

The Elementary Operating System is a collection of service routines that provide input/output facilities to peripheral devices. Programs accessing these devices need not be concerned with the physical characteristics of the devices, because EOS logically resides between the physical devices and application programs. EOS shields application programs from the details of AdamNet.

Entries to EOS are defined in Subsection 4.1.7. Application programs should use only these entry points. Access to EOS in any other manner is dangerous and may cause program malfunction. EOS error codes are listed in Subsection 4.1.8.

FIGURE 3-4: EOS MEMORY MAP

FILE CONTROL BLOCK HEADERS	D390H
FCB BUFFERS (3X1KB)	D400H
EOS CODE	E000H
ADAMNET DEVICE DRIVERS	F400H
EOS DATA TABLES	FBFFH
EOS JUMP TABLES	FC30H
GLOBAL RAM AREA	FD50H
PROCESSOR CONTROL BLOCK	FEC0H
DEVICE CONTROL BLOCK	FEC4H
DMA RESERVED BYTE	FFFFH

4.1.1 EOS Overwrite Addresses

Some applications may not require all of the routines provided by EOS. To accommodate this situation, EOS has three defined starting locations. Applications should always access the routines from the defined jump table locations.

Location 0D390H is the lowest EOS starting address. File manager software is located in this part of EOS. If the application program uses file manager software, the application should not overwrite 0D390H or above.

Location 0E000H. If the application does not use the file manager software, it can overwrite EOS up to 0DFFFFH. Routines that access the lower level network device drivers, and the device drivers for the VDP, controllers and sound generator are available.

Locations 0F400H to 0FC30H contain the lowest level device drivers for AdamNet. The application can extend up to 0F3FFH if it is using OS_7 or its own drivers for the VDP, controllers and sound generator.

4.1.2 EOS Files

The file structure takes advantage of the sequential design of the data pack. (For more information on tapes and tape formats, see Chapter 3, Section 5.) A file directory keeps track of the location of files on tape, and other pertinent information such as size, creation date, and protection attributes. Some programs, such as super games, do not require directories.

Data is stored on tape in blocks of 1K (1024 bytes). An EOS file occupies a number of contiguous blocks, allocated upon file creation. EOS files can be any number of blocks but; are limited by the physical storage space available.

Block 0 is reserved for a cold start loader. Block 1 and up, depending on the directory size, contain the directory.

The EOS file manager accesses and controls a file through a File Control Block (FCB) maintained in RAM. The FCB contains static and dynamic information about the file. The FCB is created when a file is opened and destroyed when the file is closed. Application programs are limited to two FCBs at any one time. Each FCB is 36 bytes long. A data buffer, 1024 bytes long, is associated with each FCB.

The Directory

The directory contains two types of records: Volume and File Records. The first 26 bytes of the first block in the directory is the Volume Record. The Volume Record is followed by a number of File Records, also 26 bytes in length. The 13th byte of the Volume Record determines the size of the directory. Up to 127 blocks may be allocated for the directory, allowing a maximum of 4953 file entries, and 39 directory entries per 1K block.

Size (bytes)	(12)	(1)	(4)	(4)	(2)	(3)
	VOL_NAME	DIRECTORY SIZE	DIR_CHECK	VOL_SIZE	reserved	CREATION DATE

Volume Record

Size (bytes)	(12)	(1)	(4)	(2)	(2)	(2)	(3)
	DIR_NAME	DIR ATTRIB	DIR_START BLOCK	DIR_MAX LENGTH	DIR_USED LENGTH	DIR_LAST COUNT	CREATION DATE

File Record

<u>Name</u>	<u>Length</u>	<u>Explanation</u>						
VOL_NAME	12 bytes	Logical Volume Name						
DIRECTORY SIZE	1 byte							
VOL_ATTR	Bit 7	Delete protection if set to 1						
VOL_DIRSIZE	Bits 6-0	Directory size in blocks						
DIR_CHECK	4 bytes	A unique code that indicates the presence of a directory.						
VOL_SIZE	4 bytes	Total number of blocks allocated						
CREATION DATE	3 bytes							
		<table><tr><td>(1)</td><td>(1)</td><td>(1)</td></tr><tr><td>VOL_YEAR</td><td>VOL_MONTH</td><td>VOL_DAY</td></tr></table>	(1)	(1)	(1)	VOL_YEAR	VOL_MONTH	VOL_DAY
(1)	(1)	(1)						
VOL_YEAR	VOL_MONTH	VOL_DAY						
DIR_NAME	12 bytes	file name. 12th byte denotes file type (A, a, H or h). Terminated by ETX (03H). (See 4.1.3)						

```

Bits:  7  6  5  4 | 3  2  1  0
       p  w  r  u | s  a  e  f

```

A typical user-defined file would have a DIR-ATTRIB of 10H. An executable file, which is loaded and run from 100H, has a DIR ATTRIB of 0C8H.

DIR_YR	DIR_MNTH	DIR_DAY
--------	----------	---------

File Control Block (FCB) Organization

bytes

12	FCB_NAME	Up to eleven characters, free format
1	FCB_ATTR	Same as attribute byte of File Record
4	FCB_START_BLOCK	File starting block number on tape
2	FCB_MAX_LENGTH	Total number of blocks allocated
2	FCB_USED_LENGTH	Number of blocks used within the file
2	FCB_LAST_COUNT	Number of bytes in last block of the file
1	FCB_DEVICE	Device containing the file. See below.
1	FCB_MODE	File open modes. See below.
4	FCB_BLOCK	Block number currently in FCB_BUFFER
4	FCB_LAST_BLOCK	Last block number of the file
2	FCB_POINTER	FCB_BUFFER pointer
1	FCB_LENGTH	Data block size of the current FCB_DEVICE
1024	FCB_BUFFER	1K buffer allocated for each file currently opened. Matches the block size on the digital data pack. Controlled by the EOS file manager to transfer data between the user program and mass storage devices.

FCB_DEVICE

A unique number assigned to all devices (present and future):

00H - Master
01H - Keyboard
02H - Printer
03H - Reserved
04H - Floppy Disk Drive - 1
05H - Floppy Disk Drive - 2
06H - Reserved
07H - Reserved
08H - Data Pack Drive - 1
18H - Data Pack Drive - 2
09H - Reserved
0AH - Reserved
0BH - Reserved
0CH - Reserved
0DH - Parallel Interface Module

0EH - RS-232C Interface Module
0FH - Gateway

4.1.3 File Types and Headers

The 12th character of DIR_NAME indicates the file type. Valid file types are A, a, h or H. The lower case file types indicate backup versions of a file. These files have the user file attribute bit set. The first three bytes of an "H" file define the length of the header and the application code. Following these bytes are the header and then ASCII information. Refer to Subsection 6.2 for an example of an "H" file.

The application code defines the format of the header. The following application codes have been assigned:

1	SmartWRITER
2	SmartBASIC
16	Electronic FlashCard Maker

"A" files have the user bit set, but do not contain headers. ASCII information begins in the first byte of the file.

4.1.4 EOS Executive Calls

This subsection summarizes EOS executive subroutines that can be called from application programs. The executive calls fall into three categories: system operations, simple device I/O and mass storage file I/O.

System Operations

_EOS_START	Starts EOS by going through initialization steps
_HARD_INIT	Initializes the system at powerup
_SOFT_INIT	Initializes the system anytime but powerup
_HARD_RESET_NET	Applies hardware reset (38H) to AdamNet
_SYNC	Synchronizes Z80 with Master 6801
_SCAN_ACTIVE	Scans devices on AdamNet and establishes the Device Control Blocks (DCB)
_RELOC_PCB	Relocates PCB and DCBs to different addresses after powerup if necessary
_FIND_DCB	Finds the DCB address for the assigned device
_GET_DCB_ADDR	Same as _FIND_DCB
_GET_PCB_ADDR	Finds the current PCB address of the system
_REQUEST_STATUS	Issues a status request command to a device

Simple Device Operations

<u>_CONS_INIT</u>	Initializes the system console
<u>_CONS_OUT</u>	Displays a character or performs screen control on the system console
<u>_REQ_KBD_STAT</u>	Requests keyboard status
<u>_REQ_PR_STAT</u>	Requests printer status
<u>_RD_CH_DEV</u>	Indicates a read command to a character device
<u>_RD_KBD</u>	Reads a character from the keyboard
<u>_START_RD_CH_DEV</u>	Sets up a character device DCB to issue read command (concurrent operation)
<u>_END_RD_CH_DEV</u>	Checks the status of character device DCB after command has been sent to read (concurrent)
<u>_START_RD_KBD</u>	Starts a keyboard read command (concurrent)
<u>_END_RD_KBD</u>	Checks the keyboard status (concurrent)
<u>_WR_CH_DEV</u>	Initiates a write command to a character device
<u>_PR_CH</u>	Prints a character on the printer
<u>_PR_BUFF</u>	Prints string of ASCII characters terminated by ETX (03)
<u>_START_WR_CH_DEV</u>	Sets up a character device DCB to issue write command (concurrent)
<u>_END_WR_CH_DEV</u>	Checks the status of character device DCB after command has been sent to write (concurrent)
<u>_START_PR_CH</u>	Starts a printer print command (concurrent)
<u>_END_PR_CH</u>	Checks the printer status (concurrent)
<u>_START_PR_BUFF</u>	Starts the command of printing a string of characters (concurrent)
<u>_END_PR_BUFF</u>	Checks the printer status (concurrent)

Mass Storage Device Operations

File Manager Section

<u>_SET_DATE</u>	Sets the current date
<u>_GET_DATE</u>	Gets the current date
<u>_QUERY_FILE</u>	Reads directory entry of a file
<u>_SET_FILE</u>	Sets the file directory entry
<u>_MAKE_FILE</u>	Creates a file in the directory
<u>_OPEN_FILE</u>	Opens a file by setting up the FCB
<u>_CLOSE_FILE</u>	Closes a file by marking the FCB
<u>_RESET_FILE</u>	Resets the file by pointing back to the first byte
<u>_READ_FILE</u>	Reads data from a file into user's buffer
<u>_WRITE_FILE</u>	Writes data to a file
<u>_FMGR_INIT</u>	Initializes the file manager
<u>_SCAN_FOR_FILE</u>	Scans the directory block(s) for a file

Device Driver Section

<u>_RD_1_BLOCK</u>	Reads a block of data (1024 bytes) from mass storage device
<u>_WR_1_BLOCK</u>	Writes a block of data to mass storage device. I/O buffers containing data to be transferred to tape can reside anywhere in Z80 RAM.
<u>_REQ_TAPE_STAT</u>	Requests status of the data pack drive
<u>_START_RD_1_BLOCK</u>	Sends read command to read a block of data from a block device (concurrent)
<u>_END_RD_1_BLOCK</u>	Checks status after <u>_START_RD_1_BLOCK</u>
<u>_START_WR_1_BLOCK</u>	Sends write command to write a block of data to a block device (concurrent)
<u>_END_WR_1_BLOCK</u>	Checks status after <u>_END_WR_1_BLOCK</u>

4.1.5 EOS Routines Adapted from OS 7

A section of EOS contains device drivers for the video processor, sound generator and controllers. Many of these routines are functionally the same as routines in the OS_7 ROM. The inputs and outputs for the EOS version of these routines is not necessarily the same as those for equivalent routines in OS_7. Entry points are defined in Subsection 4.1.7. The file A_uOS_00 in the EOS Source Code specifies parameter passing.

Routines for the Video Processor

WRITE_VRAM	READ_VRAM
WRITE_REGISTER	READ_REGISTER
FILL_VRAM	INIT_TABLE
PUT_VRAM	GET_VRAM
CALC_OFFSET	PX_TO_PTRN_POS
LOAD_ASCII	PUT_ASCII
WR_SPR_ATTRIBUTE	

Routines for the Controllers

DECODER	POLLER
SPINNER	

Routines for the Sound Processor

DECLSN	DECMSN
MSNTOLSN	ADD8TO16
SOUND_INIT	TURN_OFF_SOUND
PLAY_IT	SOUNDS
EFFECT_OVER	

Adam Routines for OS_7 Access

SWITCH_MEM	PORT_COLLECTION
------------	-----------------

4.1.6 Initializing EOS

No initialization of EOS is necessary if an application program boots from disk or data pack. If a program boots from the cartridge slot or a connector, EOS must be loaded from ROM to its executable location, according to the following procedure. This procedure must be executed from address range 8000H through 0DFFFH in intrinsic RAM.

Step 1: Select the SmartWRITER option in lower memory.

out 7FH,00H

Step 2: Strobe the EOS_ENABLE line by writing the value 02H to port 3FH.

```
out 3FH,02H
```

Step 3 Copy EOS from location 6000H to location 0E000H. Do not overwrite the DCBs.

```
LD HL,6000H
LD DE,0E000H
LD BC,0FEC0H - 0E000H
LDIR
```

Step 4: Deselect EOS ROM. EOS ROM should be deselected if the SmartWRITER ROMS will be accessed by the application program. LOAD_ASCII is an example of an EOS routine which accesses the EOS ROM.

```
out 3FH,0
```

Step 5: Initialize EOS tables. All RAM should be cleared to 0. See Subsection 4.1.7 for the equate values.

```
LD BC,CLEAR_RAM_SIZE
LD DE,CLEAR_RAM_START+1
LD HL,CLEAR_RAM_START
XDR A
LD [HL],A
LDIR
```

Step 6: Initialize I/O ports. EOS collects port values from the OS_7 ROM. EOS routines use these ports when they access the video processor, controllers and sound generator.

```
CALL PORT_COLLECTION
```

Step 7: Initialize AdamNet. _HARD_INIT performs the 6801/Z80 synchronization, performs a roll call poll on the network, and establishes the DCBs.

```
CALL _HARD_INIT
```

4.1.7 EOS Entry Points

```

ADD816          EQU 0FD4DH :P
BLK_STRT_PTR    EQU 0FDDCH :D
BLOCKS_REQ      EQU 0FE0CH :D
BUF_END         EQU 0FE0AH :D
BUF_START       EQU 0FE08H :D
BYTES_REQ       EQU 0FE02H :D
BYTES_TO_GO     EQU 0FE04H :D
CALC_OFFSET     EQU 0FD32H :P
CLEAR_RAM_SIZE  EQU 00147H :A
CLEAR_RAM_START EQU 0FD60H :D
COLORTABLE      EQU 0FD6CH :D
CONTROLLER_0_PD EQU 0FC28H :A
CONTROLLER_1_PD EQU 0FC2CH :A
CURRENT_DEV     EQU 0FD6FH :D
CURRENT_PCB     EQU 0FD70H :D
CURSOR          EQU 0FEA5H :D
CUR_BANK        EQU 0FD6EH :D
DCB_IMAGE       EQU 0FD8BH :D
DECLSN          EQU 0FD44H :P
DECSN           EQU 0FD47H :P
DEFAULT_BT_DEV  EQU 0FD6FH :D
DEVICE_ID       EQU 0FD72H :D
DIR_BLOCK_NO    EQU 0FDD9H :D
EFFECT_OVER     EQU 0FD5CH :P
EOS_DAY         EQU 0FDE2H :D
EOS_MONTH       EQU 0FDE1H :D
EOS_STACK       EQU 0FE58H :D
EOS_YEAR        EQU 0FDE0H :D
PCB_BUFFER      EQU 0FDBAH :D
PCB_DATA_ADDR   EQU 0FDFFH :D
PCB_HEAD_ADDR   EQU 0FD0FH :D
FILENAME_CMPS   EQU 0FDD8H :D
FILE_COUNT      EQU 0FDD4H :D
FILE_NAME_ADDR  EQU 0FD73H :D
FILE_NUMBR      EQU 0FDD7H :D
FILL_VRAM       EQU 0FD26H :P
FMGR_DIR_ENT    EQU 0FDE3H :D
FNUM            EQU 0FE01H :D
FOUND_AVAIL_ENT EQU 0FDD8H :D
GET_VRAM        EQU 0FD2FH :P
INIT_TABLE      EQU 0FD29H :P
INT_VCTR_TBL    EQU 0FBFFH :A
KEYBOARD_BUFFER EQU 0FD75H :D
LINEBUFFER      EQU 0FE7EH :D
LOAD_ASCII      EQU 0FD38H :P
MEM_CNFG00      EQU 0FC17H :A
MEM_CNFG01      EQU 0FC18H :A
MEM_CNFG02      EQU 0FC19H :A
MEM_CNFG03      EQU 0FC1AH :A
MEM_CNFG04      EQU 0FC1BH :A
MEM_CNFG05      EQU 0FC1CH :A
MEM_CNFG06      EQU 0FC1DH :A
MEM_CNFG07      EQU 0FC1EH :A
MEM_CNFG08      EQU 0FC1FH :A
MEM_CNFG09      EQU 0FC20H :A
MEM_CNFG0A      EQU 0FC21H :A
MEM_CNFG0B      EQU 0FC22H :A
MEM_CNFG0C      EQU 0FC23H :A
MEM_CNFG0D      EQU 0FC24H :A
MEM_CNFG0E      EQU 0FC25H :A

```

```

MEM_CNFG0F      EQU 0FC26H :A
MEM_SWITCH_PORT EQU 0FC27H :A
MOD_FILE_COUNT  EQU 0FDD5H :D
MSNTOLSN        EQU 0FD4AH :P
NET_RESET_PORT  EQU 0FC28H :A
NEW_HOLE_SIZE   EQU 0FE1AH :D
NEW_HOLE_START  EQU 0FE16H :D
NUM_COLUMNS     EQU 0FEA0H :D
NUM_LINES       EQU 0FE9FH :D
JLDCCHAR        EQU 0FE79H :D
PATRNGENTBL     EQU 0FD6AH :D
PATRNNAMETBL    EQU 0FD68H :D
PCB              EQU 0FEC0H :A
PERSONAL_DEBOUN EQU 0FE5AH :D
PLAY_IT         EQU 0FD56H :P
POLLER          EQU 0FD3EH :P
PORT_COLLECTION EQU 0FD11H :P
PORT_TABLE      EQU 0FC27H :A
PRINT_BUFFER     EQU 0FD76H :D
PTRN_NAME_TBL    EQU 0FEA3H :D
PTR_TO_LST_DF_S  EQU 0FE6EH :D
PTR_TO_S_ON_0    EQU 0FE70H :D
PTR_TO_S_ON_1    EQU 0FE72H :D
PTR_TO_S_ON_2    EQU 0FE74H :D
PTR_TO_S_ON_3    EQU 0FE76H :D
PUT_ASCII        EQU 0FD17H :P
PUT_VRAM         EQU 0FD2CH :P
PX_TO_PTRN_POS   EQU 0FD35H :P
QUERY_BUFFER     EQU 0FD0AH :D
READ_REGISTER    EQU 0FD23H :P
READ_VRAM        EQU 0FD1DH :P
RETRY_COUNT      EQU 0FDD6H :D
REV_NUM          EQU 0FD60H :D
SAVE_CTRL        EQU 0FE78H :D
SECTORS_TO_INIT  EQU 0FD86H :D
SECTOR_NO        EQU 0FD87H :D
SCUNDPRT        EQU 0FC2FH :A
SOUNDS           EQU 0FD59H :P
SDUND_INIT       EQU 0FD50H :P
SPIN_SW0_CT      EQU 0FE58H :D
SPIN_SW1_CT      EQU 0FE59H :D
SPRITEATTRTBL    EQU 0FD64H :D
SPRITEGENTBL     EQU 0FD66H :D
START_BLOCK      EQU 0FE12H :D
STROBE_RESET_PD  EQU 0FC2EH :A
STROBE_SET_PORT  EQU 0FC2DH :A
SWITCH_MEM       EQU 0FD14H :P
SWITCH_TABLE     EQU 0FC17H :A
TEMP_STACK       EQU 0FE6EH :D
TURN_OFF_SOUND   EQU 0FD53H :P
UPDATE_SPINNER   EQU 0FD41H :P
UPPER_LEFT       EQU 0FEA1H :D
USER_BUF         EQU 0FE06H :D
USER_NAME        EQU 0FE10H :D
VDP_CTRL_PORT    EQU 0FC29H :A
VDP_DATA_PORT    EQU 0FC2AH :A
VDP_MODE_WORD    EQU 0FD61H :D
VDP_STATUS_BYTE  EQU 0FD63H :D
VECTOR_08H       EQU 0FBFFH :A
VECTOR_10H       EQU 0FC02H :A
VECTOR_18H       EQU 0FC05H :A
VECTOR_20H       EQU 0FC08H :A
VECTOR_28H       EQU 0FC0BH :A

```

4.1.7 EOS Entry Points

VECTOR_30H	EQU 0FC0EH	:A	_REQ_TAPE_STAT	EQU 0FC87H	:P
VECTOR_38H	EQU 0FC11H	:A	_RESET_FILE	EQU 0FCC6H	:P
VECTOR_66H	EQU 0FC14H	:A	_SCAN_ACTIVE	EQU 0FC8AH	:P
VOL_BLK_SZ	EQU 0FDDCH	:D	_SCAN_FOR_FILE	EQU 0FCFCH	:P
VRAM_ADDR_TABLE	EQU 0FD64H	:D	_SET_DATE	EQU 0FCD8H	:P
WRITE_REGISTER	EQU 0FD20H	:P	_SET_FILE	EQU 0FCCFH	:P
WRITE_VRAM	EQU 0FD1AH	:P	_SOFT_INIT	EQU 0FC8DH	:P
WR_SPR_ATTRIBUT	EQU 0FD38H	:P	_SOFT_RES_DEV	EQU 0FC90H	:P
X_MAX	EQU 0FE78H	:D	_SOFT_RES_KBD	EQU 0FC93H	:P
X_MIN	EQU 0FE7AH	:D	_SOFT_RES_PR	EQU 0FC96H	:P
Y_MAX	EQU 0FE7DH	:D	_SOFT_RES_TAPE	EQU 0FC99H	:P
Y_MIN	EQU 0FE7CH	:D	_START_PR_BUFF	EQU 0FC9CH	:P
_CHECK_FCB	EQU 0FCF0H	:P	_START_PR_CH	EQU 0FC9FH	:P
_CLOSE_FILE	EQU 0FCC3H	:P	_START_RD_1_BLK	EQU 0FCA2H	:P
_CONS_DISP	EQU 0FC33H	:P	_START_RD_CH_DE	EQU 0FCA5H	:P
_CONS_INIT	EQU 0FC36H	:P	_START_RD_KBD	EQU 0FCA8H	:P
_CONS_OUT	EQU 0FC39H	:P	_START_WR_1_BLK	EQU 0FCA3H	:P
_CV_A	EQU 0FD0EH	:P	_START_WR_CH_DE	EQU 0FCAEH	:P
_DELETE_FILE	EQU 0FCE1H	:P	_SYNC	EQU 0FCB1H	:P
_DLY_AFT_HRD_RE	EQU 0FC3CH	:P	_TRIM_FILE	EQU 0FCE0H	:P
_END_PR_BUFF	EQU 0FC3FH	:P	_WRITE_BLOCK	EQU 0FCF6H	:P
_END_PR_CH	EQU 0FC42H	:P	_WRITE_FILE	EQU 0FCD5H	:P
_END_RD_1_BLOCK	EQU 0FC45H	:P	_WR_1_BLOCK	EQU 0FCB4H	:P
_END_RD_CH_DEV	EQU 0FC48H	:P	_WR_CH_DEV	EQU 0FCB7H	:P
_END_RD_KBD	EQU 0FC4BH	:P			
_END_WR_1_BLOCK	EQU 0FC4EH	:P			
_END_WR_CH_DEV	EQU 0FC51H	:P			
_EOS_1	EQU 0FD05H	:P			
_EOS_2	EQU 0FD08H	:P			
_EOS_3	EQU 0FD0BH	:P			
_EOS_START	EQU 0FC30H	:P			
_FILE_QUERY	EQU 0FCFFH	:P			
_FIND_DCB	EQU 0FC54H	:P			
_FMGR_INIT	EQU 0FCBAH	:P			
_GET_DATE	EQU 0FCDBH	:P			
_GET_DCB_ADDR	EQU 0FC57H	:P			
_GET_PCB_ADDR	EQU 0FC5AH	:P			
_GOTO_WP	EQU 0FCE7H	:P			
_HARD_INIT	EQU 0FC50H	:P			
_HARD_RESET_NET	EQU 0FC60H	:P			
_INIT_TAPE_DIR	EQU 0FCB0H	:P			
_MAKE_FILE	EQU 0FCC9H	:P			
_MODE_CHECK	EQU 0FCF9H	:P			
_OPEN_FILE	EQU 0FCC0H	:P			
_POSIT_FILE	EQU 0FD02H	:P			
_PR_BUFF	EQU 0FC63H	:P			
_PR_CH	EQU 0FC66H	:P			
_QUERY_FILE	EQU 0FCCC	:P			
_RD_1_BLOCK	EQU 0FC69H	:P			
_RD_DEV_DEP_STA	EQU 0FCE4H	:P			
_RD_KBD	EQU 0FC6CH	:P			
_RD_KBD_RET_COD	EQU 0FC6FH	:P			
_RD_PR_RET_CODE	EQU 0FC72H	:P			
_RD_RET_CODE	EQU 0FC75H	:P			
_RD_TAPE_RET_CO	EQU 0FC78H	:P			
_READ_BLOCK	EQU 0FCF3H	:P			
_READ_EDS	EQU 0FCEAH	:P			
_READ_FILE	EQU 0FCD2H	:P			
_RELOC_PCB	EQU 0FC7BH	:P			
_RENAME_FILE	EQU 0FCDEH	:P			
_REQUEST_STATUS	EQU 0FC7EH	:P			
_REQ_KBD_STAT	EQU 0FCB1H	:P			
_REQ_PR_STAT	EQU 0FCB4H	:P			

4.1.8 EOS Error Codes

DCB_NOT_FOUND	EQU	1
DCB_BUSY	EQU	2
DCB_IDLE_ERR	EQU	3
NO_DATE_ERR	EQU	4
NO_FILE_ERR	EQU	5
FILE_EXISTS_ERR	EQU	6
NO_FCB_ERR	EQU	7
MATCH_ERR	EQU	8
BAD_FNUM_ERR	EQU	9
EOF_ERR	EQU	10
TOO_BIG_ERR	EQU	11
FULL_DIR_ERR	EQU	12
FULL_TAPE_ERR	EQU	13
FILE_NM_ERR	EQU	14
RENAME_ERR	EQU	15
DELETE_ERR	EQU	16
RANGE_ERR	EQU	17
CANT_SYNC1	EQU	18
CANT_SYNC2	EQU	19
PRT_ERR	EQU	20
RQ_TP_STAT_ERR	EQU	21
DEVICE_DEPD_ERR	EQU	22
PROG_NON_EXIST	EQU	23
NO_DIR_ERR	EQU	24

4.2 OS 7

4.2.1 Introduction

This subsection presents an overview of the ColecoVision Operating System, referred to as the OS or OS_7. In contrast to the typical definition of an operating system as a run-time executive, the ColecoVision OS is a run-time user's library that provides access to modules which control events related to graphics, sounds, timing, etc.

Appendix 3 contains the detail sections of the ColecoVision Programmer's Manual, which document each module. Included are the calling sequence, input/output parameters, side effects, and and calls to other OS routines. The documentation assumes that the programmer is familiar with the Z80 instruction set.

The ColecoVision OS provides power-up procedures and system-defined entry points for the user. It handles input/output operations and housekeeping functions. The OS provides the displayable ASCII character set and vectors into the cartridge memory space that correspond to the Z80 hardware restart and interrupt vectors.

The OS software can be divided functionally as follows:

- Graphics Generation
- Sound Generation
- Interrupt Handling and Write Deferral
- Timing
- Controller Interface
- Boot-up
- Miscellaneous Utilities
- Defined Reference Locations

4.2.2 Graphics Generation Software

Software in this area is divided into the chip driver level, the table level and the object level. First, there are chip drivers for the TMS 9928 level. Software to initialize and manipulate tables belongs to the table level. Video graphics are generated by objects on the object level. Each object has its own definition of tables, frames and display locations called out by the user.

4.2.3 Sound Generation Software

Sound generation software produces tones and music by table "look-up." The software also has provisions to produce special effects. A prioritization scheme allows important

sounds to overlay less important background sounds without destroying their continuity.

4.2.4 Interrupt Handling and Write Deferral Software

When the OS is accessing the VDP register or VRAM, protection may be needed when a VDP interrupt occurs. Interrupt deferral routines handle this situation when top-level graphics software is in use. Bulletin No. 10 in Appendix D of the ColecoVision Programmer's Manual lists additional interrupt deferral software to fix the problem at low-level VDP access.

4.2.5 Timing Software

In a real-time application, timing is essential to the system operation. The OS timing software manages the software timers allocated by the user. It allows the user to start the timer, update the timer, check if it is timed out and then relinquish the timer. The number of software timers that can be supported depends on available cartridge RAM.

4.2.6 Controller Interface

The controller interface has several features:

The process of scanning, debouncing and decoding controller inputs is automatic upon invoking the OS routines.

The software only debounces and decodes those inputs that the user wishes to access.

The user has the option of bypassing automatic scanning and decoding and accessing the raw controller data, if necessary.

4.2.7 Boot-Up Software

The OS performs certain initialization tasks, such as turning off the sound chip, initializing buffers and flags, and display of the standard logo screen. If no game cartridge is present, it warns the user to turn off the system before attempting to insert a cartridge or expansion module.

If a cartridge is present, the OS reads the cartridge title from a predefined location and displays it along with copyright information. This screen is displayed for a short time before the OS relinquishes control to the cartridge software.

4.2.8 Miscellaneous Utilities

This software includes some low-level utilities, evolved in conjunction with the graphics and sound packages. It inclu-

des a nibble arithmetic package and a routine that displays a standard game-option screen.

4.2.9 Defined Reference Locations

OS_7 has a number of system-defined reference locations in the areas of cartridge RAM, OS ROM and Cartridge ROM. The user-accessible locations are listed in the file, OS_SYMBOLS, Rev. 2 (Appendix F of the ColecoVision Programmer's Manual). It is IMPORTANT for the user to call the OS subroutines through those authorized entry points (Jump Table). The user also must be aware of the limitation of using cartridge RAM as a scratch pad due to the predefined OS data areas and stack memory. Locations at the beginning of the cartridge program may be accessed for pointers to the tables and buffers in CRAM. Also in this area, restart and interrupt vectors are defined for the user. Section X of the ColecoVision Programmers Manual lists all the system defined locations in all memory areas.

5. TAPE FORMAT AND OTHER TAPE CONSIDERATIONS

There are two types of tape format for ADAM data packs. Type GW tapes have block 0 at the end. Type HE tapes have block 0 in the middle. An example of a type GW is The Buck Rogers™ Plaent of Zoom™ Super Game. Both SmartBASIC and the blank data pack are examples of type HE tapes.

The capacity of the tape is 256K. Blocks are defined as 1K in length. There are two tracks, 128 blocks per track. Block numbers refer to the block of data on a data pack referenced by application software. The following illustration shows physical block positions.

FORMAT GW (Block 0 at the end)

←--- tape motion

0	40	7F	block no.
=====			track 0
=====			
80	C0	FF	track 1
=====			block no.

Format HE (Block 0 in the middle)

←--- tape motion

40	7F 0	3F	block no.
=====			track 0
=====			
80	BF C0	FF	track 1
=====			block no.

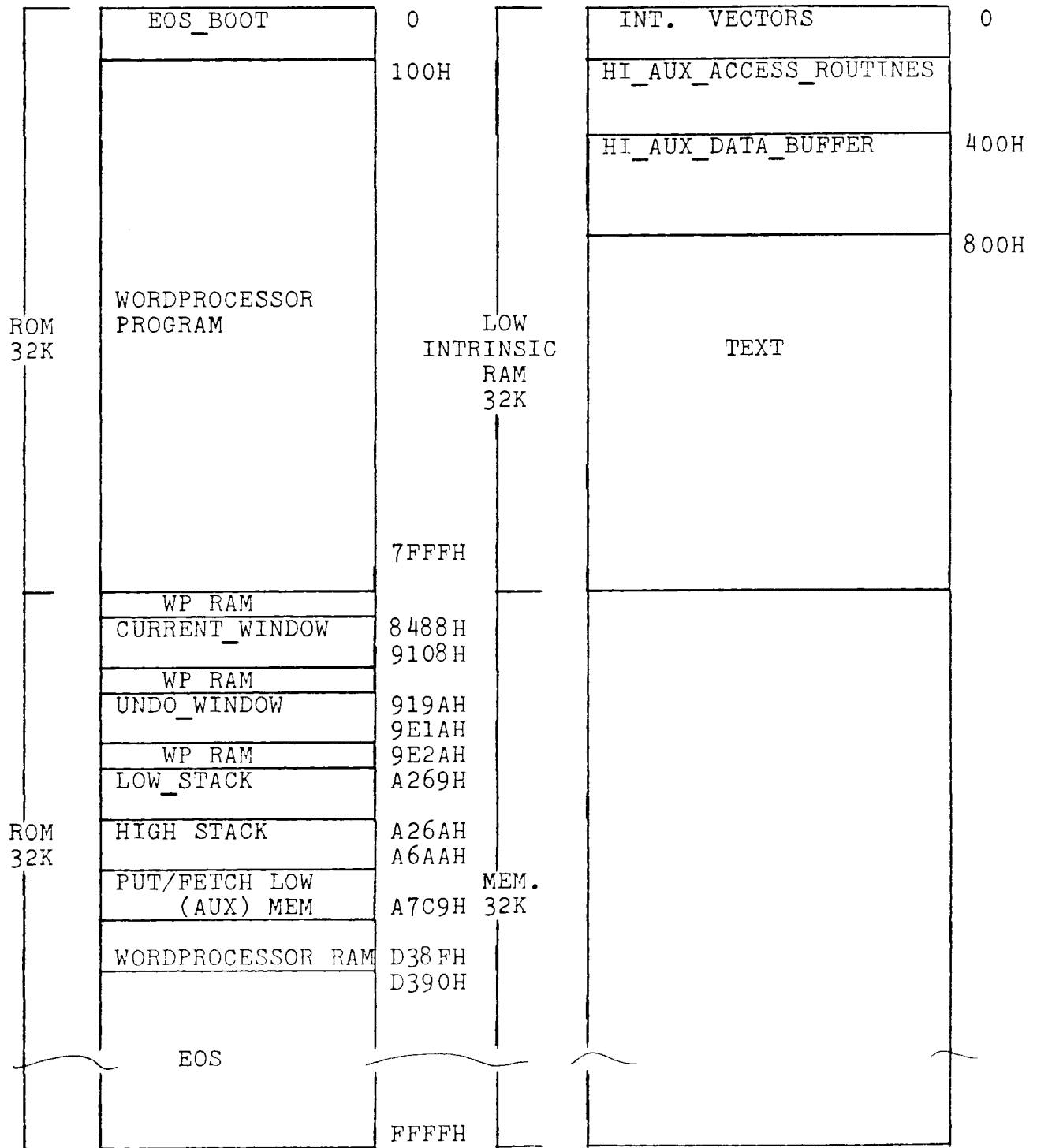
Considerations in Choosing Tape Format

Type GW tapes do not contain a directory. Type GW tapes cannot be readily copied, so users are not able to easily copy one GW tape to another GW tape.

Type HE tapes are compatible with other type HE tapes. If a program needs to store information on a separate data pack, access or be accessed by other software, then a type HE tape is suggested.

6. SmartWRITER

6.1 Memory Map



6.2 SmartWRITER-Compatible Files

Files must meet certain criteria to be compatible with SmartWRITER. SmartWRITER files have the user file attribute bit set, and the last character of the file name is H. SmartWRITER files start with a header and ASCII information begins after the header. The first two bytes of the file define the length of the header. The third byte contains the application code (1 for SmartWRITER). The format of the header is determined by the application code. Backup versions of SmartWRITER files have a lower-case "h" as the file type.

An example of a SmartWRITER file follows:

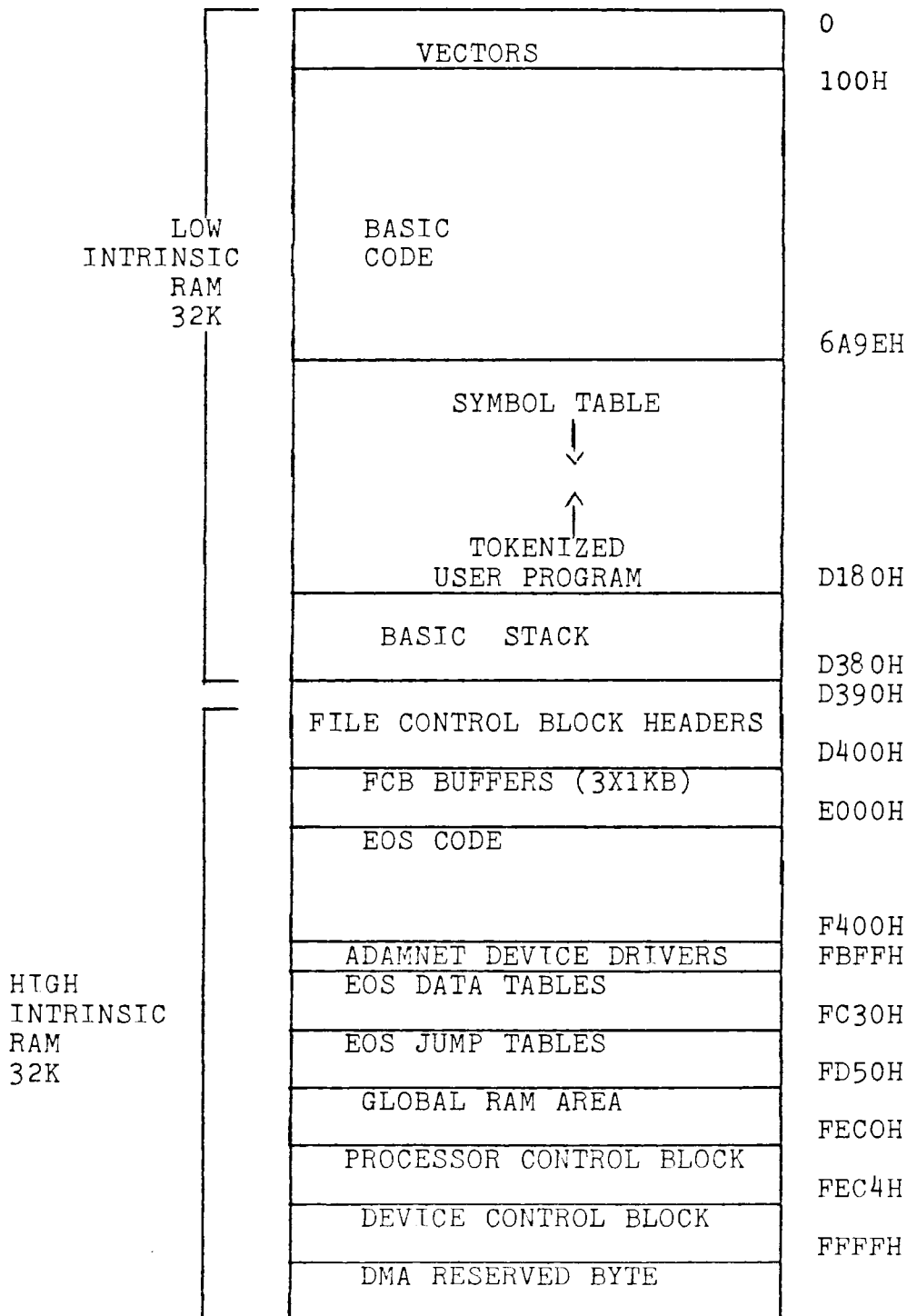
<u>BYTE</u>	<u>DESCRIPTION</u>
0	HEADER_SIZE_LOW (=256)
1	HEADER_SIZE_HIGH (=0)
2	FILE_TYPE_CODE (=1)
3	TOP_MARGIN
4	BOTTOM_MARGIN
5	LEFT_MARGIN
6	RIGHT_MARGIN
7	LINE_SPACING
8	TAB_ARRAY (1) [0 = NO MRG, 1 = MRG]
89	TAB_ARRAY (80) [0 = NO MRG, 1 = MRG]
90	UNUSED
258	UNUSED
259	FIRST ASCII DATA BYTE
n	ASCII DATA

7. SmartBASIC

SmartBASIC is generally source-code compatible with Applesoft BASIC. PEEKS and POKES, and other machine-dependent features of Applesoft BASIC are different in SmartBASIC. SmartBASIC graphics feature four modes:

- Text mode - 24 lines of 31 characters
- low resolution graphics - 40 x 40, with four lines of text
- high resolution graphics - 256 x 160 with four lines of text
- Pure high resolution graphics - 256 x 192

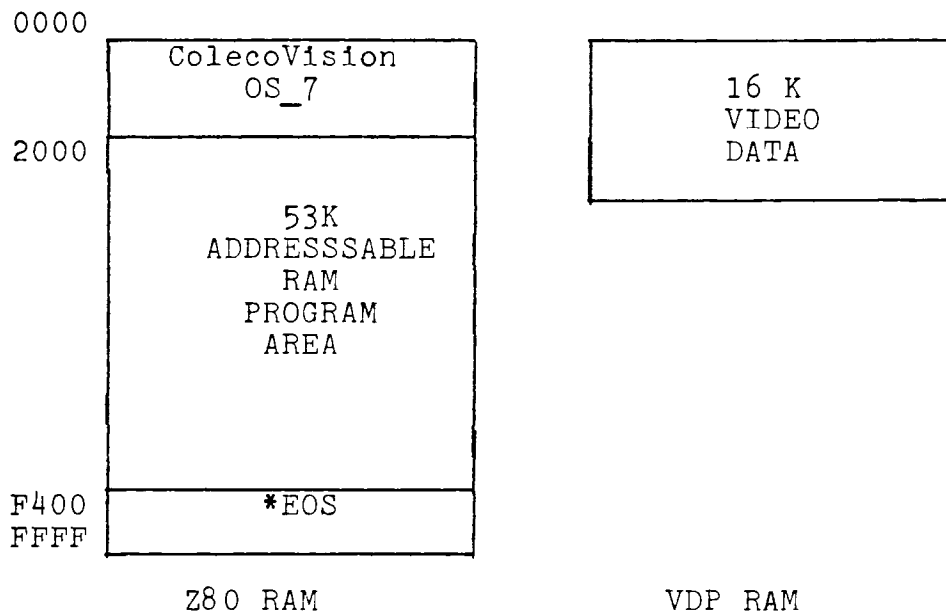
7.1 Memory Map



8. SUPER GAMES AND OTHER PROGRAMS USING OS7

Refer to Chapter 5, Section 2 for a detailed explanation of the Buck Rogers™ Planet of Zoom™ Super Game. The Buck Rogers game serves as an example for developers of super games.

8.1 MEMORY MAP



* ADAM EOS starts at D390. In this example of a super game, only the AdamNet device drivers in EOS are needed. Refer to Chapter 3, Subsection 4.1.1, EOS Overwrite Addresses.

Address 0000 through 1FFF

OS_7 ROM is available for program use and requires 8K.

Address F400 through FFFF

EOS is used to access peripherals and files, and requires 3K.

Address 2000 through F3FF

The remaining 53K of Z80 RAM is available for program and data storage. OS_7 uses RAM from 7000H through 73FFH.

Address 0000 through 3FFF VDP RAM

Unused portions of the 16K VDP RAM may be used as temporary storage. VDP RAM may not be loaded directly from tape, but must be read into intrinsic RAM, then transferred using one of the VDP access routines.

9. ROM-BASED CARTRIDGE PROGRAMS

9.1 Memory Map

0000

OS_7

2000

7000

73FF

CARTRIDGE RAM

8000

CARTRIDGE ROM
(UP TO 32K)

FFFF

CHAPTER 4: OPTIONAL PERIPHERALS

This chapter will be developed as optional peripherals become available.

CHAPTER 5: DEVELOPMENT TOOLS AND UTILITIES

1. SUPER GAMES

This section explains background loading, tape mapping and playability of super games and other programs that use OS_7. Familiarity with OS_7 is assumed. The information and examples in this section should help programmers who are accustomed to cartridge software adapt to tape-based software more easily.

1.1 Background Loading

The background loading software (see Subsection 1.7) is designed to load overlays from data pack to RAM while the program is executing. Care must be taken that data being loaded during execution does not destroy data that is controlling execution. A good approach is to use two buffers. One buffer is loaded while the other is controlling program execution.

1.2 Timing Considerations

Each background block read takes about one second assuming no retries and no repositioning. Retries take about 1.2 to 1.4 seconds and only happen if the checksum fails to compare on read. There are a maximum of three attempts to read a block on the tape, then the checksum failed code is sent. Repositioning takes as long as one second to find the current position on tape (approximately one second for every 80 inches of tape travel). Repositioning is automatically handled by the drive. For the NMI driven tape manager, additional overhead in transfer time averages 8 milliseconds ($\frac{1}{2}$ 60hz clock tick). Buffering data for transfer to VDP-RAM results in Z80 CPU usage assuming use of WRITE VRAM. The following table shows the Z80 CPU time used in VDP-RAM writes.

TABLE 4: Z-80 CPU TIME FOR VDP-RAM WRITES

Number of bytes to transfer	Milliseconds
1	0.077
10	0.175
100	1.16
1000	11.25
2000	22.4
4000	44.8
8000	89.6

1.3 Mapping

The programmer must lay out the data to minimize load time, through mapping. Minimizing load time is the most crucial aspect of designing super games.

Since the data on tape is a memory image, it may be read directly into Z80 RAM and immediately executed. The programmer should diagram the tape blocks on a time line, following the timing considerations in Subsection 1.2. Some action on the screen must hide the loading process. Rewinding and positioning time must be included in calculating load size and load time.

An example of a time line diagram follows. It shows the screen actions and what is loaded in background while the action is taking place.

TIME IN SECONDS	SCREEN	LOADING	ADDRESS	BLOCK NUMBER
0	BLANK	COLD START MAIN	C800H	0
1			8000H	1
2			8400	2
3			8800	3
4	TITLE SCREEN	OVERLAY 3	8C00	4
5			9000	5
6			9400	6
7			9800	7
8	LOGO SCREEN		9C00	8
9			A000	9
10			A400	10
11			A800	11
12	SELECT OPTIONS		AC00	12
13			B000	13
14			B400	14
15			B800	15
16	GAME START	[*REWIND TO TRACK 1 SECTOR 1 5 SECONDS OF REWIND TIME FOLLOW]	BC00	16
17				128
18				
19				
20		OVERLAY 4		
21				
22			C000	129
23			C400	130
24		ETC...	C800	131
25			CC00	132

TIME LINE DIAGRAM

Overlay Control Blocks

To read memory images into RAM, three pieces of information are required. They are:

Start of memory buffer of transfer address	2 bytes
Start of memory image on tape or block number	2 bytes
Number of 1K Blocks in the memory image	1 byte

This information is organized into a 5-byte block called an Overlay Control Block (OCB). The overlay control blocks are organized into an Overlay Control Table (OCT). The OCT controls the loading of data from tape using the tape interface software described in Subsection 1.6. The table should be pointed to by a two byte pointer and it should be terminated by a byte set to FF. The Overlay Control Table should be loaded immediately after the cold start loader.

1.4 Start of Game

EOS loads Block 0 (the cold start loader) to C800H and passes control to it. The cold start loader then initializes the system and loads enough of the main program to allow some user interaction to begin. The following rules define the interface to the cold start loader.

1. The main program is loaded to location 8000H.
2. Immediately following the ColecoVision OS vectors at 8000H, and the game name, is a pointer to the OCT. The first entry in the OCT describes the main program.
3. Control is passed to the main program by the vector at address 800AH as defined in the OS_7 PRIME.
4. The main program must contain the OCT in the first 1K block.
5. The main program must contain the background loading routines described in Subsection 1.7. They are required to reside in the first 3K.
6. The main program must immediately display graphics or allow some user interaction.
7. Once control is passed to the main program, game play must start as soon as possible.
8. When control is transferred to the main program, Register B contains the boot device ID. Register B should be stored at the globally defined address DEVICE_ID.

Subsection 1.5 is an example of the 8000H area code which interfaces to the cold start loader. Compare this to the area defined in OS_7.

1.5 8000H Area Code- Interface to Cold Start Loader

FILE: CART:ROB

HEWLETT-PACKARD: CART (c) Coleco 1984

Fri, 18 May 1984, 14:59

PAGE 1

LOCATION OBJECT CODE LINE SOURCE LINE

```
1 *280*
2
3
4 ; This is an example of the DCB pointer following the game name
5 ; section and how to store the boot device number for future use.
6 ; The RAM at 8000H is defined in greater detail in the ColecoVision
7 ; Programmers Manual.
8
9          ORG      8000H
10         EQU      0000
11 ; ** CARTRIDGE SOFTWARE POINTERS 8000H **
12 ; -----
13
14         HEX      55,AA      ;Cartridge present: Colecovision logo
15         DEFW     -----   ;Pointer to the sprite name table
16         DEFW     -----   ;Pointer to the sprite order table
17         DEFW     -----   ;Pointer to the working buffer for WR_SPR_NN_TBL
18         DEFW     -----   ;Pointer to the hand controller input areas
19         DEFW     MAIN_PROG  ;Entry point to the user program
20
21 *****
22
23         JP      RST      ;RST 08
24         JP      RST      ;RST 10
25         JP      RST      ;RST 18
26         JP      RST      ;RST 20
27         JP      RST      ;RST 28
28         JP      RST      ;RST 30
29
30 * THIS IS THE MASKABLE INTERRUPT SOFT VECTOR
31         JP      MASK_INT
32
33 * THIS IS THE NMI SOFT VECTOR.
34         JP      VDPINT
35
36 *****
37
38         ** Game name section **
39
40         DEFB     "SUPER"
41         DEFB     "/GAMES ",IDH,"/1984"
42
43 ;         IDH = Copyright symbol
44 ;         IEH,IFH = Trade mark symbol
45
46 *****
47
48         EXT      DCB
49
50         DEFW     DCB      ; A pointer to the DCB must follow the
51                          ; date in the game name.
52
53         DEV_ID   DEFB      8      ; This is the default DEVICE_ID. See MAIN_PROG (below) for applications
54                          ; booted from a different device.
55 *****
```

8000 55AA
8002 0000
8004 0000
8006 0000
8008 0000
800A 803E

800C C38039
800F C38039
8012 C38039
8015 C38039
8018 C38039
801B C38039

801E C3803C
8021 C3803A

8024 5355504552
8029 2F47414045
802E 5320102F31
8033 393834

8036 0000
8038 08

LOCATION OBJECT CODE LINE SOURCE LINE

```
57 *****
58
8039 C9 59 RST      RET      ;280 restarts to be defined
60
61 ;-----
803A ED45 62 VOPINT    RETN      ;Non maskable interrupt processing-
63              ;Normally used for critical timing:
64              ;      music
65              ;      processing timers
66              ;      sprite motion processing
67
68 ;-----
803C ED4D 69 MASK_INT  RETI      ;280 maskable interrupt vector-
70              ;Normally used with spinner switch:
71              ;      steering wheel
72              ;      sports controller
73              ;      roller controller
74
75              EXT      _HARD_INIT,COLDSTART
803E      76 MAIN_PROG
803E 78      77      LD      A,B              ; Main prog is entered with the Device ID in the B registe
78              ;If it is not saved, the DDP manager or application will o
79              ;run from Drive A.
803F 328038 80      LD      [DEV_ID],A
81
82 ; The rest of the application follows...
```

LINE#	SYMBOL	TYPE	REFERENCES
75	COLDSTART	E	
53	DEV_ID	A	80
76	MAIN_PROG	A	19
69	MASK_INT	A	31
48	DCB	E	50
59	RST	A	23, 24, 25, 26, 27, 28
62	VDPINT	A	34
75	_HARD_INIT	E	
10	----	A	15, 16, 17, 18

1.6 Tape Interface Software

The tape interface software loads RAM from tape in background using the OCT structure defined in Subsection 1.7. The entry points for these modules are described in the following subsections. The tape manager programs are interrupt driven and should be called on every clock cycle (60Hz) to drive background loading. The OCT, DDP_MANAGER (Subsection 1.10) and TAPE_INTERFACE (Subsection 1.9) or DDP_INTERFACE (Subsection 1.11) must be linked into the program.

1.6.1 Tape Manager Programs

The tape manager consists of two interchangeable parts dependent upon environment.

TAPE_MANAGER

The program TAPE_MANAGER (shown in Subsection 1.8) is designed for use on the HP64000. This program allows for simulated tape I/O via the HP disk, and should be used in conjunction with TAPE_INTERFACE.

DDP_MANAGER

This program, shown in Subsection 1.10, replaces the tape manager in working games. Entry points and interface are identical to the TAPE_MANAGER except that this module uses EOS calls to manipulate the tape. DDP_MANAGER should be used in conjunction with DDP_INTERFACE.

The final version of the game should have the DDP_MANAGER installed. The tape managers are fully interchangeable.

1.6.2 TAPE INTERFACE

TAPE_INTERFACE consists of a set of entry points and data passed in the accumulator. DDP_INTERFACE is similar to TAPE_INTERFACE, except that some labels are different.

<u>Entry Points</u>	<u>Accumulator</u>	<u>Comments</u>
LOAD_OVERLAY	OVERLAY_NUMBER	The code uses the overlay number to look up OCT information. The overlay is loaded from tape to Z80 RAM.
WRITE_OVERLAY	OVERLAY_NUMBER	As in LOAD_OVERLAY, but data flow is from RAM to tape.
ABORT_TAPE	-----	Aborts all tape functions. Resets tape. Makes tape ready to immediately receive new commands. Does not reposition tape.
TEST_TAPE	-----	Returns status of tape.

Status from TEST-TAPE may show the following conditions.

0	OK
1	Checksum failed
2	Block not found
3	Tape not present
4	Device not present

The checksum shows that after three retries, the tape block was not read correctly and the data transferred by the read command is not valid.

1.7 Background Loading Software

FILE: OCB:TDS

HONEYWELL-PACKARD: OCB (c) Coleco 1982 Confidential

Tue, 15 May 1984, 20:30

PAGE 1

LOCATION OBJECT CODE LINE

SOURCE LINE

```
1 *ZROM*
3 NAME *Rev 0 - DTT*
4
5 DESCR_OCB      MACRO
6                .GOTO ENDESCR_OCB
7
8 Author:        DTT
9 Project:       WAFER, A132
10 Starting date: 25mar83
11
12 *****
13 * OCB          DTT *
14 *****
15
16      Rev.  Date      Name      Change
17      1
18      0   25mar83   DTT      Initial Pseudo code
19
20 NAME: OCB (OVERLAY CONTROL BLOCK TABLE)
21 THE OCB DESCRIBED BELOW IS AN EXAMPLE ONLY AND DOES NOT DESCRIBE
22 ANY GAME. I AM SHOWING A MAIN PROGRAM STARTING AT 8000H.
23 THE BACKUP COPY OF THE COLD START LOADER ALSO GETS LOADED AT 8000H.
24 THERE ARE 18 OVERLAYS DESCRIBED IN THIS DCT. THE LAST TWO ARE
25 SPARE. TWO DESCRIBE VANITY SCREEN AND DATA
26
27 ENDESCR_OCB:
28 MEND
29
30 :EXTERNAL DATA AREAS USED:
31 : EXT
32
33 :GLOBAL DATA AREAS DEFINED:
34 GLB OCB
35 EXT OCB_PTR
36
37 :LOCAL EQUATES
38
39 :GLOBAL EQUATES
40
41 :-----
42 OCB MACRO LP1,LP2,LP3 :LOAD ADDRESS,BLOCK NUMBER,NUMBER OF BLOCKS
43 * REV 0 DTT. 7/12/83 coded and tested
44 DEFW LP1 :TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
45 DEFW LP2 :BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
46 DEFB LP3 :NUMBER OF 1K BLOCKS TO TRANSFER
47 MEND
48
49 :-----
50 PROG
51
52 OCB:
53 OCB 08000H,1,16 :MAIN PROGRAM
54 * REV 0 DTT. 7/12/83 coded and tested
55 * DEFW 08000H :TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
56 * DEFW 1 :BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
57 * DEFB 16 :NUMBER OF 1K BLOCKS TO TRANSFER
58 OCB 03000H,1+16,16 :BACKUP MAIN PROGRAM
```

0000
0000

0000 8000
0002 0001
0004 10
0005

LOCATION OBJECT CODE LINE SOURCE LINE

```

+ * REV 0 DTT. 7/12/83 coded and tested
0005 8000 + DEFW 08000H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0007 0011 + DEFW 1+16 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0009 10 + DEFB 16 ;NUMBER OF 1K BLOCKS TO TRANSFER
000A 55 OCB 02400H,1+16+16,7 ;OVERLAY 3
+ * REV 0 DTT. 7/12/83 coded and tested
000A 2400 + DEFW 02400H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
000C 0021 + DEFW 1+16+16 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
000E 07 + DEFB 7 ;NUMBER OF 1K BLOCKS TO TRANSFER
000F 56 OCB 0C000H,1+16+16+7,8 ;OVERLAY 4
+ * REV 0 DTT. 7/12/83 coded and tested
000F C000 + DEFW 0C000H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0011 0028 + DEFW 1+16+16+7 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0013 08 + DEFB 8 ;NUMBER OF 1K BLOCKS TO TRANSFER
0014 57 OCB 02400H,1+16+16+7+8,6 ;OVERLAY 5
+ * REV 0 DTT. 7/12/83 coded and tested
0014 2400 + DEFW 02400H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0016 0030 + DEFW 1+16+16+7+8 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0018 06 + DEFB 6 ;NUMBER OF 1K BLOCKS TO TRANSFER
0019 58 OCB 0C000H,1+16+16+7+8+6,8 ;OVERLAY 6
+ * REV 0 DTT. 7/12/83 coded and tested
0019 C000 + DEFW 0C000H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0018 0036 + DEFW 1+16+16+7+8+6 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0010 08 + DEFB 8 ;NUMBER OF 1K BLOCKS TO TRANSFER
001E 59 OCB 02400H,1+16+16+7+8+6+8,6 ;OVERLAY 7
+ * REV 0 DTT. 7/12/83 coded and tested
001E 2400 + DEFW 02400H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0020 003E + DEFW 1+16+16+7+8+6+8 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0022 06 + DEFB 6 ;NUMBER OF 1K BLOCKS TO TRANSFER
0023 60 OCB 0C000H,1+16+16+7+8+6+8+6,6 ;OVERLAY 8
+ * REV 0 DTT. 7/12/83 coded and tested
0023 C000 + DEFW 0C000H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0025 0044 + DEFW 1+16+16+7+8+6+8+6 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0027 06 + DEFB 6 ;NUMBER OF 1K BLOCKS TO TRANSFER
0028 61 OCB 02400H,1+16+16+7+8+6+8+6+6,4 ;OVERLAY 9
+ * REV 0 DTT. 7/12/83 coded and tested
0028 2400 + DEFW 02400H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
002A 004A + DEFW 1+16+16+7+8+6+8+6+6 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
002C 04 + DEFB 4 ;NUMBER OF 1K BLOCKS TO TRANSFER
002D 62 OCB 07C00H,128+1,1 ;REWIND (NEVER ACTUAL EXECUTEABLE CODE)
+ * REV 0 DTT. 7/12/83 coded and tested
002D 7C00 + DEFW 07C00H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
002F 0081 + DEFW 128+1 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0031 01 + DEFB 1 ;NUMBER OF 1K BLOCKS TO TRANSFER
0032 63 OCB 0C000H,128+1+13,14 ;OVERLAY 11
+ * REV 0 DTT. 7/12/83 coded and tested
0032 C000 + DEFW 0C000H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0034 008E + DEFW 128+1+13 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0036 0E + DEFB 14 ;NUMBER OF 1K BLOCKS TO TRANSFER
0037 64 OCB 02400H,128+1+13+14,7 ;OVERLAY 12
+ * REV 0 DTT. 7/12/83 coded and tested
0037 2400 + DEFW 02400H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0039 009C + DEFW 128+1+13+14 ;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
003B 07 + DEFB 7 ;NUMBER OF 1K BLOCKS TO TRANSFER
003C 65 OCB 0C000H,128+1+13+14+7,1 ;OVERLAY 13
+ * REV 0 DTT. 7/12/83 coded and tested
003C C000 + DEFW 0C000H ;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)

```

LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
003E	00A3	+	DEFW	128+1+13+14+7	;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0040	01	+	DEFB	1	;NUMBER OF 1K BLOCKS TO TRANSFER
0041		66	DCB	03000H,128+1,10	;OVERLAY 14 VANITY SCREEN
		+	* REV 0 DTT. 7/12/83 coded and tested		
0041	3000	+	DEFW	03000H	;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0043	0081	+	DEFW	128+1	;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0045	0A	+	DEFB	10	;NUMBER OF 1K BLOCKS TO TRANSFER
0046		67	DCB	05800H,128+13,1	;OVERLAY 15 VANITY DATA OVERLAY
		+	* REV 0 DTT. 7/12/83 coded and tested		
0046	5800	+	DEFW	05800H	;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0048	008D	+	DEFW	128+13	;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
004A	01	+	DEFB	1	;NUMBER OF 1K BLOCKS TO TRANSFER
0048		68	DCB	0C000H,128+1+13+14+7+1,6	;OVERLAY 16
		+	* REV 0 DTT. 7/12/83 coded and tested		
0048	C000	+	DEFW	0C000H	;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
004D	00A4	+	DEFW	128+1+13+14+7+1	;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
004F	06	+	DEFB	6	;NUMBER OF 1K BLOCKS TO TRANSFER
0050		69	DCB	0FFFFH,0FFFFH,0FFH	;SPARE
		+	* REV 0 DTT. 7/12/83 coded and tested		
0050	FFFF	+	DEFW	0FFFFH	;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0052	FFFF	+	DEFW	0FFFFH	;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0054	FF	+	DEFB	0FFH	;NUMBER OF 1K BLOCKS TO TRANSFER
0055		70	DCB	0FFFFH,0FFFFH,0FFH	;SPARE
		+	* REV 0 DTT. 7/12/83 coded and tested		
0055	FFFF	+	DEFW	0FFFFH	;TRANSFER ADDRESS OF THE OVERLAY (WHERE DOES IT GET LOADED IN RAM)
0057	FFFF	+	DEFW	0FFFFH	;BLOCK NUMBER OF THE FIRST BLOCK IN THE OVERLAY
0059	FF	+	DEFB	0FFH	;NUMBER OF 1K BLOCKS TO TRANSFER
005A	FF	71	HEX	FF	;DEFINES THE TERMINATOR

FILE: DCB:TDS

CROSS REFERENCE TABLE

PAGE 4

LINE#	SYMBOL	TYPE	REFERENCES
52	OCB	P	34
35	DCB_PTR	E	

1.8 TAPE MANAGER

FILE: TAPE_MANAGER.TOS HEWLETT-PACKARD: TAPE_MANAGER (c) Coleco 1983 Confidential Tue, 15 May 1984, 20:30 PAGE 1

LOCATION OBJECT CODE LINE SOURCE LINE

```
1  "ZROM"
3  NAME "Rev 6 - DTT"
4
5  DESCR_          MACRO
6                  .GOTO ENDESCR_1
7
8  Author:         DTT
9  Project:        -----* ---
10 Starting date: 10feb83
11
12  Prev release Date:
13  Prev release Rev:
14
15  Header Rev: 3
16
17  *****
18  *
19  * TAPE_MANAGER   DTT
20  *
21  *****
22
23      Rev History (one line note indicating the change)
24
25      Rev.  Date      Name      Change
26      6    11/16/83   DTT      ADDED DEVICE ID TO SIMULATE DEVICE INDEPENDENCE
27      5    12sep83   DTT      MODIFIED ERROR SYSTEM TO WRITE ERRORS TO CSA AREA
28      4    08aug83   DTT      SIMULATES WRITES!
29      3    07jul83   DTT      ADDED KILL_TAPE/CSA
30      2    30jun83   DTT      STATE MACHINE FOR MULTI TASKING
31      1    05apr83   DTT      BINK ON I/O ERROR, REMOVE TIMING STUFF, EI AND DI ADDED.
32      0    10feb83   DTT      Initial Pseudo code
33
34  NAME: TAPE_MANAGER (OVERLAY CONTROL)
35
36
37  FUNCTION: (LOAD OVERLAYS VIA SIMULATED I/O WITH THE HP64000)
38
39
40  INPUTS:  (ACCUMULATOR = OVERLAY NUMBER)
41
42
43  OUTPUTS: (OVERLAY IS LOADED TO RAM ADDRESS)
44
45
46  PSEUDOCODE (PASCAL type pseudocode of procedure.)
47
48
49
50      ----- overlay request? Y -----
51      :                               :
52      :-->: IDLE                       : RENAME
53      :   :                               :
54      :   :                               :
55      :   :                               :
56      :   : overlay request? N         :
57      :   :                               :
58      :   :                               :
```

LOCATION OBJECT CODE LINE SOURCE LINE

```

59 :
60 :
61 :
62 :
63 :
64 : \:/
65 :
66 : :
67 : : READ : <-----: OPEN :
68 : : FIRST : :
69 : : RECORD : :
70 : :-----:
71 : :
72 : :
73 : \:/
74 :
75 : :-----> :
76 : : READ : : MOVE DATA :
77 : : DATA : : end of file? N : FROM BUFFER :
78 : : RECORD : : <-----: TO RAM :
79 : :-----:
80 : :
81 : : end of file? Y :
82 : : \:/ :
83 : :-----:
84 : : :
85 : : PRE IDLE STATE : <-----: CLOSE :
86 : : (WAIT FOR CLOSE): :
87 : : :
88 : :-----:
89 ENDESCR_1 MEND

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

91 DESC_a      MACRO
92 .GOTO      DESC_a
93 *****
94 Each transition state looks like this:
95
96
97 from previous
98 =====:
99 state      :
100           \:/
101
102           : I/O errors? Y      :
103           : TRANSITION         :-----> : ERROR STATE      :
104           :==>:                 : (terminate) :-----:
105           : :                   :=====: :
106           : -----:             :-----: :
107           : :                   :         : /:\         :
108           : :                   :         : :           :
109           : sim i/o :           \:/       :-----:
110           -----:             ----- to state 6
111           function : :=====>
112           complete? N : TEST FOR :
113           : ABORT :to state 7
114           : REQUEST :=====>
115           -----
116           no :
117           abort : to next
118           : state
119           \:/
120
121 DESC_a      MEND

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

123 DESC_2          MACRO
124 .GOTO          DESC_2
125 *
126 *****
127 *
128
129 COMMON ATTRIBUTE AREAS (COMMUNICATION BUFFER WITH HP)
130 ALL VALUES ARE HEXIDECIMAL
131
132 1) ASSIGN FILENAME TO CA
133      CA      CA+1      CA+2      CA+n
134 -----
135 : 8A :LENGTH :FILENAME -----> : USER ID      : NOT
136 :      :BYTE   :UP TO 9 BYTES   : UP TO 5 BYTES : USED----->
137 -----
138 LENGTH_BYTE := (((LENGTH OF FILE NAME)+1)/2)-1)*32) + ((LENGTH OF USER ID)/2)*8)
139 FILENAME MUST BE ODD NUMBER OF BYTES LONG MAY BE PADDED WITH ONE SPACE.
140 USER ID MUST BE EVEN NUMBER OF BYTES LONG MAY BE PADDED WITH ONE SPACE.
141
142 2) OPEN
143      CA      CA+1      CA+2
144 -----
145 : 81 : 04 : 00 : NOT
146 :      :   :   : USED----->
147 -----
148 CA+1 MUST BE 04HEX DESIGNATES ABSOLUTE FILE WILL BE OPENED.
149 CA+2 MUST BE 00HEX DESIGNATES DISK NUMBER.
150
151 3) CLOSE
152      CA
153 -----
154 : 82 : NOT
155 :      : USED ----->
156 -----
157
158 4) READ
159      CA      CA+1      CA+2      CA+4      CA+6
160 -----
161 : 87 : 80 : #BYTES TO LOAD : N/A : N/A : LOAD ADDRESS
162 :      :   : MSB : LSB :      : MSB : LSB
163 -----
164 CA+1 DEFINES THE BUFFER LENGTH IN WORDS - 1 MUST BE LOADED BEFORE CALLING SIM I/O
165 CA+2 DEFINES NUMBER OF BYTES TO MOVE TO RAM FROM THE SIM I/O BUFFER
166 CA+6 DEFINES THE RAM ADDRESS TO LOAD TO
167 CA+8 (NOT SHOWN) IS THE START OF THE SIM I/O BUFFER
168 NOTE: CA+2 AND CA+4 ARE ONE WORD LONG BUT NOT STORED AS 280 WORDS.
169 THE 280 EXPECTS WORD VALUES TO BE STORED LSB/MSB.
170
171 DESC_2
172 MEND
173
174

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

176 NEXT_STATE      MACRO &P1
177                LD      A,&P1                      :SOMETHING IN THE COMMAND BUFFER!
178                LD      [TAPE_STATE],A
179                LD      HL,[STATE_VECTORS+&P1+&P1]
180                LD      [NEXT_STATE_ADDRESS],HL
181                JP      END_OF_STATE_MACHINE
182 DESC_8           MEND
183 :SUBROUTINES CALLED:
184 :      EXT
185
186 :OPERATING SYSTEM CALLS:
187
188 :EXTERNAL DATA AREAS USED:
189                EXT KILL_TAPE
190                EXT WRITE_TAPE
191                EXT CSA
192                EXT TAPE_STATE
193                EXT OVERLAY_NUMBER
194
195 :GLOBAL DATA AREAS DEFINED:
196                GLB      DEVICE_ID
197
198 :LOCAL EQUATES
<7400> 199 CA      EQU      07400H                      :COMMON ATTRIBUTES AREA FOR SIM I/O
<7401> 200 BUF_LEN EQU      CA+1
<7401> 201 FILETYPE EQU     CA+1                      :FILETYPE ADDRESS IN C_A_
<7402> 202 DISC_NUM EQU     FILETYPE+1                :DISK NUMBER ADDRESS IN C_A_
<7402> 203 REC_LEN EQU      CA+2                      :RECORD LENGTH ADDRESS IN C_A_
<7408> 204 BUFFER EQU      CA+8                      :I_O_BUFFER
<7401> 205 RNAME_BUF EQU    CA+1                      :RENAME BUFFER AREA
<7404> 206 LOADADDR EQU     CA+4                      :ADDRESS OF MEM ADDRESS OF OVERLAY
207
<0080> 208 MAXBUFLN EQU      128                      :UP TO 128 WORDS MAY BE READ IN
<0004> 209 ABSOLUTE EQU      004H                      :FILE TYPE IS ABSOLUTE
<0081> 210 OPEN EQU          081H
<0082> 211 CLOSE EQU         082H
<0087> 212 READ EQU          087H
<0089> 213 WRITE EQU         089H
<008A> 214 RENAME EQU        08AH
215
216 :GLOBAL EQUATES
217 :      INCLUDE equate file name
218

```

LOCATION	OBJECT CODE	LINE	SOURCE LINE
		220	PROG
		221	GLB TAPE_MANAGER
		222	
		223	GLB INITIALIZE_TAPE
0000		224	INITIALIZE_TAPE:
		225	GLB INIT_TAPE
0000		226	INIT_TAPE:
0000 3E00		227	LD A,0 ;CLEAR THE HP64000 AREA
0002 327400		228	LD [CA],A
0005 320000		229	LD [CSA],A ;AND THE TAPE CONTROL STATUS AREA
0008 30		230	DEC A
0009 320000		231	LD [OVERLAY_NUMBER],A ;MAKE THE OVERLAY NUMBER -1 (INVALID)
000C		232	NEXT_STATE 0 ;SET THE IDLE STATE!
000C 3E00		*	LD A,0 ;SOMETHING IN THE COMMAND BUFFER!
000E 320000		*	LD [TAPE_STATE],A
0011 2A0041		*	HL,[STATE_VECTORS+0+0]
0014 220000		*	LD [NEXT_STATE_ADDRESS],HL
0017 C3016A		*	JP END_OF_STATE_MACHINE
001A C9		233	RET
		234	
001B		235	TAPE_MANAGER:
		236	; BEGIN (Ordinarily registers are restored; retain only the pushes and pops you need.)
		237	
001B 3A7400		238	LD A,[CA] ;TEST THE STATUS OF THE FILE
001E 87		239	OR A
001F FA016A		240	JP M,END_OF_STATE_MACHINE ;IF THERE IS A COMMAND IN THE BUFFER
0022 2805		241	JR Z,AB_REQ
0024 FE01		242	CP 1 ;END OF FILE ON READ FROM HP64000
0026 C2016C		243	JP NZ,ERROR
		244	*
		245	*
		246	*
		247	GLB AB_REQ
0029 3A0000		248	AB_REQ: LD A,[CSA] ;IF COMMAND IS TO KILL TAPE COMMAND
002C FE00		249	CP KILL_TAPE
002E 2000		250	JR NZ,CASE_STATE
0030 3A0000		251	LD A,[TAPE_STATE] ;CHECK THE STATE OF THE TAPE
		252	*
0033 FE03		253	CP 3 ;STATE 0,1,2
0035 DA013D		254	JP C,STATE_PRE_IDLE ;FILE NOT OPENED
		255	*
0038 FE06		256	CP 6 ;STATE 3,4,5
003A DA012A		257	JP C,STATE_CLOSE ;FILE OPENED TRY TO CLOSE IT
		258	*
		259	*
		260	*
		261	*
		262	*
		263	* CASE TAPE_STATE,(IDLE,RENAME,OPEN,READ1,READ2,MOVE2VRAM,CLOSE,PRE_IDLE)
		264	*
003D		265	CASE_STATE:
		266	*
		267	*
003D 2A0000		268	LD HL,[NEXT_STATE_ADDRESS]
0040 E3		269	JP [HL]

LOCATION OBJECT CODE LINE SOURCE LINE

```

0043 0068      272      DEFW STATE_RENAME          ; 1
0045 00A2      273      DEFW STATE_OPEN           ; 2
0047 00C2      274      DEFW STATE_READ_1         ; 3
0049 00DA      275      DEFW STATE_READ_2         ; 4
004B 00F2      276      DEFW STATE_MOVE2RAM       ; 5
004D 012A      277      DEFW STATE_CLOSE          ; 6
004F 013D      278      DEFW STATE_PRE_IDLE       ; 7
0051 015A      279      DEFW WRITE_1
280 *
281 *          IF THE MACHINE IS IDLE IT'S OK TO TEST FOR ANOTHER READ REQUEST
282 *
0053      283 STATE_IDLE:                          ;STATE 0
0053 3A0000      284          LD A,[CSA]              ;TEST THE COMMAND STATUS AREA
0056 B7          285          OR A
0057 CA016A      286          JP Z,END_OF_STATE_MACHINE
287 *
005A      288 NEXT_STATE 1
005A 3E01      +          LD A,1                      ;SOMETHING IN THE COMMAND BUFFER!
005C 320000      +          LD [TAPE_STATE],A
005F 2A0043      +          LD HL,[STATE_VECTORS+1+1]
0062 220000      +          LD [NEXT_STATE_ADDRESS],HL
0065 C3016A      +          JP END_OF_STATE_MACHINE
289 *
290 *          ASSIGN THE SIM I/O FILE TO THE CURRENT OVERLAY NAME
291 *
0068      292 STATE_RENAME:                          ;STATE 1
0068 3A0000      293          LD A,[OVERLAY_NUMBER]
006B 210187      294          LD HL,SAMPLE_NAME
006E 117401      295          LD DE,RNAM_BUF              ;POINT TO THE RENAME BUFFER IN THE COMMON ATTRIBUTES AREA
0071 010004      296          LD BC,SAMPLE_NAME_LEN      ;NO NAME COULD BE MORE THAN TEN BYTES LONG IT?
0074 ED80        297          LDIR                      ;MOVE THE OVERLAY DATA TO THE RENAME BUFFER
298 *
299 *
300 *
0076 010A30      301          LD BC,10*256+"0"          ;GET THE OVERLAY NUMBER INTO ASCII
0079      302 OV_2_ASCII_1:
0079 B7          303          OR A
007A 98          304          SBC A,B                  ;DIVIDE OVERLAY NUMBER BY 10
007B 0C          305          INC C                      ;SET THE NUMBER OF TENS IN C (DOEN'T RESET CARRY)
007C 30FB        306          JR NC,OV_2_ASCII_1          ;CARRY WAS SET BY SUBTRACT IF B>A
007E 0D          307          DEC C                      ;SAVE THE TENS BYTE
007F 80          308          ADD A,B                  ;ADD TEN TO GET THE REMAINDER
0080 0630        309          LD B,"0"                  ;SAVE THE ONES BYTE
0082 80          310          ADD A,B
0083 47          311          LD B,A
0084 79          312          LD A,C
0085 FE30        313          CP "0"
0087 2002        314          JR NZ,OV_2_ASCII_2          ;CONVERT TO ASCII
0089 3E5F        315          LD A,"_"                  ;SPECIAL CASE
008B      316 OV_2_ASCII_2:
008B 12          317          LD [OE],A                  ;OE IS POINTING TO THE FILENAME FROM THE LDIR ABOVE
008C 13          318          INC DE
008D 78          319          LD A,B
008E 12          320          LD [OE],A
321 *
322 *          ASSIGN THE FILE
323 *

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

00BF 3E84      324      LD      A,RENAME
0091 327400    325      LD      [CA],A
                                326 *
0094          327      NEXT_STATE 2                                ;NEXT OPEN
0094 3E02      +      LD      A,2                                ;SOMETHING IN THE COMMAND BUFFER!
0096 320000    +      LD      [TAPE_STATE],A
0099 2A0045    +      LD      HL,[STATE_VECTORS+2+2]
009C 220000    +      LD      [NEXT_STATE_ADDRESS],HL
009F C3016A    +      JP      END_OF_STATE_MACHINE

                                328 *
                                329 *
00A2          330 STATE_OPEN:                                ;STATE 2
                                331 *
                                332 *      OPEN THE FILE
                                333 *
00A2 3E04      334      LD      A,ABSOLUTE                                ;LOAD THE FILE TYPE
00A4 327401    335      LD      [FILETYPE],A
00A7 3E00      336      LD      A,0                                ;DISK NUMBER
00A9 327402    337      LD      [DISC_NUM],A
00AC 3E81      338      LD      A,OPEN                                ;OPEN THE CURRENT FILE
00AE 327400    339      LD      [CA],A
00B1 3A0000    340      LD      A,[CSA]                                ;NEXT STATE READ OR WRITE?
00B4          341      NEXT_STATE 3                                ;NEXT STATE = READ_1
00B4 3E03      +      LD      A,3                                ;SOMETHING IN THE COMMAND BUFFER!
00B6 320000    +      LD      [TAPE_STATE],A
00B9 2A0047    +      LD      HL,[STATE_VECTORS+3+3]
00BC 220000    +      LD      [NEXT_STATE_ADDRESS],HL
00BF C3016A    +      JP      END_OF_STATE_MACHINE
00C2          342 STATE_READ_1:                                ;STATE 3
                                343 *
                                344 *      READ THE FIRST RECORD TO SKIP OVER IT
                                345 *
00C2 3E80      346      LD      A,MAXBUFLN                                ;SET THE INPUT BUFFER LENGTH
00C4 327401    347      LD      [BUF_LEN],A                                ;
00C7 3E87      348      LD      A,READ
00C9 327400    349      LD      [CA],A                                ;READ THE FIRST RECORD
                                350
00CC          351      NEXT_STATE 4
00CC 3E04      +      LD      A,4                                ;SOMETHING IN THE COMMAND BUFFER!
00CE 320000    +      LD      [TAPE_STATE],A
00D1 2A0049    +      LD      HL,[STATE_VECTORS+4+4]
00D4 220000    +      LD      [NEXT_STATE_ADDRESS],HL
00D7 C3016A    +      JP      END_OF_STATE_MACHINE

                                352
                                353 *
                                354 *      READ REMAINING RECORDS AND MOVE TO RAM
                                355 *
00DA          356 STATE_READ_2:                                ;STATE 4
00DA 3E80      357      LD      A,MAXBUFLN                                ;SET THE INPUT BUFFER LENGTH
00DC 327401    358      LD      [BUF_LEN],A                                ;
00DF 3E87      359      LD      A,READ                                ;
00E1 327400    360      LD      [CA],A
                                361                                ;NEXT STATE = READ A DATA RECORD
00E4          362      NEXT_STATE 5
00E4 3E05      +      LD      A,5                                ;SOMETHING IN THE COMMAND BUFFER!
00E6 320000    +      LD      [TAPE_STATE],A
00E9 2A0043    +      LD      HL,[STATE_VECTORS+5+5]

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

00EC 220000      +      LD      [NEXT_STATE_ADDRESS],HL
00EF C3016A      +      JP      END_OF_STATE_MACHINE
363
364 *
365 *      MOVE THE LAST BUFFER TO RAM
366 *
00F2            367 STATE_MOVE2RAM:
00F2 3A7400      368      LD      A,[CA]
00F5 87          369      OR      A
00F6 280E        370      JR      Z,NO_EOF
371 *
372 *
373
374            NEXT_STATE 6
00F8            375 NO_EOF:
00F8 3E06        376      LD      HL,[CLOC_LEN]
00FA 320000      377      LD      C,H
00FD 2A004D      378      LD      B,L
0100 220000      379 *
0103 C3016A      380      LD      HL,[LOADADDR]
0106            381      LD      D,L
0106 2A7402      382      LD      E,H
0109 4C          383      LD      HL,BUFFER
010A 45          384
0108 2A7404      385      LD      A,[CSA]
010E 55          386      CP      WRITE_TAPE
010F 5C          387      JR      Z,WRITE_1
0110 217408      388
0113 3A0000      389      LDIR
0116 FE00        390 *
0118 283A        391
011A EDB0        392            NEXT_STATE 4
011C            393 *
011C 3E04        394 *      CLOSE OVERLAY FILE
011E 320000      395 *
0121 2A0049      396 STATE_CLOSE:
0124 220000      397      LD      A,CLOSE
0127 C3016A      398      LD      [CA],A
399
400            NEXT_STATE 7
012F            401 *
012F 3E07        402 *
0131 320000      +      LD      A,7
0134 2A004F      +      LD      [TAPE_STATE],A
0137 220000      +      LD      HL,[STATE_VECTORS+7*7]
013A C3016A      +      LD      [NEXT_STATE_ADDRESS],HL
+      JP      END_OF_STATE_MACHINE

```

```

;STATE 5
;CHECK THE RETURN STATUS
;END OF FILE?

```

```

;NEXT STATE = CLOSE FILE

```

```

;SOMETHING IN THE COMMAND BUFFER!

```

```

;MOVE THE CURRENT OVERLAY TO MEMORY
;GET THE NUMBER OF BYTES TO MOVE TO RAM

```

```

;GET THE LOAD ADDRESS AND CHECK TO SEE IF IT IS IN VRAM
;FROM ADDRESS

```

```

;NEXT STATE = READ A DATA RECORD
;SOMETHING IN THE COMMAND BUFFER!

```

```

;STATE 6
;CLOSE THE FILES
;NEXT STATE = PRE-IDLE

```

```

;SOMETHING IN THE COMMAND BUFFER!

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

403 *      NEXT STATE IS IDLE
404 *
0130      405 STATE_PRE_IDLE:      ;STATE 7
013D 3E00      406      LD      A,0      ;NEXT STATE = PRE-IDLE
013F 320000     407      LD      [CSA],A      ;CLEAR THE COMMAND STATUS BUFFER
0142 3D      408      DEC      A
0143 320000     409      LD      [OVERLAY_NUMBER],A      ;SET THE OVERLAY NUMBER TO OUT OF RANGE
0146      410 NEXT_STATE 0
0146 3E00      +      LD      A,0      ;SOMETHING IN THE COMMAND BUFFER!
0148 320000     +      LD      [TAPE_STATE],A
0148 2A0041     +      LD      HL,[STATE_VECTORS+0+0]
014E 220000     +      LD      [NEXT_STATE_ADDRESS],HL
0151 C3016A     +      JP      END_OF_STATE_MACHINE

411 *
412 *      WRITE OUT 250 DATA BYTES TO AN OVERLAY! OH BJJ!
413 *
0154      414 WRITE_1:      ;GET HERE FROM MOVE2RAM!
0154 EB      415      EX      DE,HL
0155 ED80     416      LDIR
0157 3E89     417      LD      A,WRITE
0159 327400   418      LD      [CA],A
015C      419 NEXT_STATE 6
015C 3E06     +      LD      A,6      ;SOMETHING IN THE COMMAND BUFFER!
015E 320000     +      LD      [TAPE_STATE],A
0161 2A004D     +      LD      HL,[STATE_VECTORS+6+6]
0164 220000     +      LD      [NEXT_STATE_ADDRESS],HL
0167 C3016A     +      JP      END_OF_STATE_MACHINE

420 *
421 *
422 *
423
424 ; END (TAPE_MANAGER)
425
016A      426 END_OF_STATE_MACHINE:
016A AF      427      XOR      A
016B C9      428      RET

429 *****
430 *****
016C      431 ERROR:
432 ; BEGIN      (Ordinarily registers are restored; retain only the pushes and pops you need.)
433
016C 4F      434      LD      C,A
016D 3A0000   435      LD      A,[OVERLAY_NUMBER]      ;SAVE THE OVERLAY NUMBER IN A
0170 47      436      LD      B,A
0171 3A0000   437      LD      A,[TAPE_STATE]
0174 210000   438      LD      HL,0      ;B HAS THE FUNCTION CODE
0177 77      439      LD      [HL],A      ;C HAS THE ERROR CODE
440 ;
441 ;      LD [HL] HAS JUST CAUSED A GINK TO OCCUR ON THE HP64000
442 ;      THIS IS TO LET THE USER KNOW THERE HAS BEEN AN ERROR
443 ;
0178 AF      444      XOR      A
0179 327400   445      LD      [CA],A      ;CLEAR THE ERROR FROM THE HP64000
017C 3C      446      INC      A      ;PRETEND THE ERROR WAS A CRC CHECK FROM ADAMS TAPE
017D 320000   447      LD      [CSA],A
0180 2A0C41   448      LD      HL,[STATE_VECTORS+0000]
0183 220000   449      LD      [NEXT_STATE_ADDRESS],HL

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

0186 C9      450      RET
              451
              452 :   END (TAPE_ERR)
              453
              454 *****
              455 *****
              456 :
              457 :   OVERLAY NAMES
              458 :
              459 :   -----
              460 :   :LENGTH :FILENAME -----> : USER ID      :
              461 :   :BYTE   :UP TO 9 BYTES   : UP TO 5 BYTES :
              462 :   -----
              463 :   LENGTH_BYTE := (((((LENGTH OF FILE NAME)+1)/2)-1)*32) + ((LENGTH OF USER ID)/2)*8)
              464 :   FILENAME MUST BE ODD NUMBER OF BYTES LONG MAY BE PADDED WITH ONE SPACE.
              465 :   USER ID MUST BE EVEN NUMBER OF BYTES LONG MAY BE PADDED WITH ONE SPACE.
              466 :
              467 *****
              468 *****
              469 *
              470 *   OVERLAY NAME 0
              471 *
0187          472 SAMPLE_NAME
0187 40        473 OVLAY0  DEFB  FL_NM_LNO*32+US_ID_LNO*8      :LENGTH DESCRIPTION BYTE
0188 4F564C5F31 474 NAM_0   ASCII "OVL_1"                      :MUST BE ODD NUMBER OF LETTERS
              475 USID0    ASCII ""                          :MUST BE EVEN NUMBER OF LETTERS
              <0002> 476 FL_NM_LNO EQU ((USID0-NAM_0+1)/2)-1      :LENGTH OF FILENAME IN WORDS
              <0000> 477 US_ID_LNO EQU ((S-USID0)/2)           :LENGTH OF USER ID IN WORDS
              <0004> 478 SAMPLE_NAME_LEN EQU S-SAMPLE_NAME-2    :NUMBER OF BYTES TO MOVE = 2
              479 :*****
              480 DATA
              481 GLB NEXT_STATE_ADDRESS
              0000 482 NEXT_STATE_ADDRESS DEFS 2                :POINTER TO NEXT ENTRY STATE
              0002 483 DEVICE_ID      DEFS 1                  :DEVICE INDEPENDENCE SIMULATED

```

LINE#	SYMBOL	TYPE	REFERENCES
209	ABSOLUTE	A	334
248	AB_REQ	P	241,247
204	BUFFER	A	383
200	BUF_LEN	A	347,358
199	CA	A	200,201,203,204,205,206,228,238,325,339,349,360,368,378,418,445
265	CASE_STATE	P	250
211	CLOSE	A	397
191	CSA	E	229,248,284,340,385,407,447
483	DEVICE_ID	D	196
202	DISC_NUM	A	337
426	END_OF_STATE_MA	P	181,240,286
431	ERROR	P	243
201	FILETYPE	A	202,335
476	FL_VM_LNO	A	473
224	INITIALIZE_TAPE	P	223
226	INIT_TAPE	P	225
189	KILL_TAPE	E	249
206	LOADADDR	A	380
208	MAXBUFLN	A	346,357
474	NAM_0	P	476
482	NEXT_STATE_ADDR	D	180,268,449,481
375	NO_EOF	P	370
210	OPEN	A	338
193	OVERLAY_NUMBER	E	231,293,409,435
473	OVLAY0	P	
302	OY_2_ASCII_1	P	306
316	OY_2_ASCII_2	P	314
212	READ	A	348,359
203	REC_LEN	A	376
214	RENAME	A	324
205	RNAM_BUF	A	295
472	SAMPLE_NAME	P	294,478
479	SAMPLE_NAME_LEN	A	296
396	STATE_CLOSE	P	257,277
283	STATE_IDLE	P	271
367	STATE_MOVE2RAM	P	276
330	STATE_OPEN	P	273
405	STATE_PRE_IDLE	P	254,278
342	STATE_READ_1	P	274
356	STATE_READ_2	P	275
292	STATE_RENAME	P	272
270	STATE_VECTORS	P	179,448
235	TAPE_MANAGER	P	221
192	TAPE_STATE	E	178,251,437
475	USID0	P	476,477
477	US_ID_LNO	A	473
213	WRITE	A	417
414	WRITE_1	P	279,387
190	WRITE_TAPE	E	386

PAGE 1

FILE: TAPE_INTE:TOS HEWLETT-PACKARD: TAPE_INTERFACE (c) Coleco 1983 Confidential

LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
----------	--------	------	------	--------	------

```

1  *Z80*
3  NAME ^Rev 02 - GPB^
4
5  De_TAPE_INTERFACE MACRO                                ;Header Rev. 5
6      .GOTO Ede_TAPE_INTERFACE
7
8  Project:        TAPE, C101
9
10 *****
11 *                                *
12 *  TAPE_INTERFACE      DTT  *
13 *                                *
14 *****
15
16      Rev History
17      Rev.  Date      Name      Change
18      2      11/2/83   GPB      CHANGED RANGE TO RANGE_ ( duplicate symbol problem)
19      1      9/13/83   DTT      CHANGED TO ALLOW ERROR RETIRES
20      0      7/5/83   DTT      Initial Pseudo code
21
22  Function:
23      REQUEST READS AND WRITES AS DEFINED IN OCB.
24      REQUESTS ABORT_TAPE.
25      TEST STATUS OF TAPE REQUEST.
26
27
28  Ede_TAPE_INTERFACE MEND
29  Pseudo_code_TAPE_INTERFACE MACRO ;Pseudocode macro area
30      BEGIN;
31          STORE OVERLAY_NUMBER
32          HL := POINTER TO OCB := OVERLAY_NUMBER*5 + OVERLAY_TABLE_POINTER
33          MOVE OCB TO CSA
34          IF WRITE THEN
35              SEND WRITE_COMMAND
36          ELSE
37              SEND READ_COMMAND
38          ENDF
39      END
40      .GOTO Ed_TAPE_INTERFACE
41
42
43
44  Ed_TAPE_INTERFACE MEND

```

LOCATION OBJECT CODE LINE SOURCE LINE

```
46 ;Subroutines called
47 :     EXT
48
49 ;Subroutines defined
50     GLB     TEST_TAPE
51     GLB     ABORT_TAPE
52     GLB     LOAD_OVERLAY
53     GLB     WRITE_OVERLAY
54     GLB     CALC_OCB_ADDR
55     GLB     WRITE_BLOCKS
56     GLB     LOAD_BLOCKS
57     GLB     BLOCK_IO
58 :     GLB
59
60 ;Operating system calls
61 :     EXT
62
63 ;Inputs/Outputs passed in registers
64 :     A = OVERLAY NUMBER 1 thru N
65 :     A <> 0 = ERROR
66
67 ;External data areas used
68     EXT     OCB_PTR                                ;POINTER TO THE OVERLAY CONTROL TABLE
69
70 ;Global data areas defined
71     GLB OVERLAY_NUMBER
72     GLB TAPE_STATE
73     GLB CSA
74     GLB XFER_ADDR
75     GLB BLOCK_NUM
76     GLB RANGE_
77 :     GLB
78
79 ;Local equates
80 :     EQU
81
82 ;Global equates
83     GLB READ_TAPE
84     GLB WRITE_TAPE
85     GLB KILL_TAPE
86 READ_TAPE EQU 81H
87 WRITE_TAPE EQU 82H
88 KILL_TAPE EQU 87H
89 :     INCLUDE File_name:userid
90
```

<0081>

<0082>

<0097>

LOCATION OBJECT CODE LINE SOURCE LINE

```

33
34 ;Inputs/Outputs passed in registers
35 ; CALLED EVERY 60th SECOND BY THE NMI
36
37
<0008> 38 TAPE1          EQU 08H
<0001> 39 READ_TAPE      EQU 81H
<0002> 40 WRITE_TAPE     EQU 82H
<0007> 41 KILL_TAPE      EQU 87H
42 INCLUDE P_DCB_EQU:EOS
+ ;THESE OUR EQUATES THAT ARE USED BY THE EOS PROGRAMS TO REFERENCE
+ ;PCB AND DCB INFORMATION
+
+
+ ;PCB EQUATES
+
<0000> + P_CMD_STAT      EQU 0          ; THIS IS THE COMMAND/STATUS BYTE
+
<0001> + P_REL_ADDR      EQU 1          ; THIS IS THE RELOCATION ADDRESS
<0001> + P_REL_ADDR_LO    EQU P_REL_ADDR+0
<0002> + P_REL_ADDR_HI    EQU P_REL_ADDR+1
+
<0003> + P_NUM_DCBS     EQU 3          ; THIS IS THE NUMBER OF DCBS DEFINED
+
+
<0004> + P_SIZE         EQU 4          ; THE NUMBER OF BYTES IN THE PCB
+
+
+ ;DCB EQUATES
+
<0000> + D_CMD_STAT      EQU 0          ; THE COMMAND STATUS BYTE
+
<0001> + D_BUF_ADR      EQU 1          ; ADDRESS OF THE DATA BUFFER
<0001> + D_BUF_ADR_LO    EQU D_BUF_ADR+0
<0002> + D_BUF_ADR_HI    EQU D_BUF_ADR+1
+
<0003> + D_BUF_LEN      EQU 3          ; THE LENGTH OF THE DATA BUFFER
<0003> + D_BUF_LEN_LO    EQU D_BUF_LEN+0
<0004> + D_BUF_LEN_HI    EQU D_BUF_LEN+1
+
<0005> + D_SECT_NUM     EQU 5          ; THE BLOCK DEVICE SECTOR NUMBER
+
<0009> + D_SEC_DEV_ID    EQU 9          ; SECONDARY DEVICE ID
+
<000E> + D_RET_COUNT     EQU 14         ; THE NUMBER OF TIMES A COMMAND WILL
+                                     ; BE RETRIED.
<000E> + D_RET_COUNT_LO    EQU D_RET_COUNT+0
<000F> + D_RET_COUNT_HI    EQU D_RET_COUNT+1
+
<0010> + D_DEV_ADDR     EQU 16         ; THE DEVICE ADDRESS (ID)
+
<0011> + D_MAX_MSG_LEN    EQU 17         ; THE MAX LENGTH OF A DATA STRING
+                                     ; FOR THE DEVICE
<0011> + D_MAX_MSG_LEN_LO    EQU D_MAX_MSG_LEN+0
<0012> + D_MAX_MSG_LEN_HI    EQU D_MAX_MSG_LEN+1
+

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

          92      PRG
          93
          94 :      TEST FOR COMPLETION OF IO REQUEST
0000      95 TEST_TAPE
0000 3A0002 96      LD A,[CSA]
0003 B7     97      OR A
0004 C3003D 98      JP      EXIT_TAPE
          99 *
         100 *
         101 *
0007      102 ABORT_TAPE
0007 3E87   103      LD      A,KILL_TAPE
0009 320002 104      LD      [CSA],A
000C C3003D 105      JP      EXIT_TAPE
          106 *
          107 *
          108
000F      109 CALC_OCB_ADDR:
000F 3D     110      DEC      A
          111      LD      C,A
          112      LD      B,0
          113      LD      HL,[OCB_PTR]
          114      ADD     HL,3C
          115      ADD     HL,8C
          116      ADD     HL,8C
          117      ADD     HL,8C
          118      ADD     HL,8C
          119      RET
          120 *
001C      121 WRITE_OVERLAY:
001C 37     122      SCF
          123      JR      LD_1
          124 *
          125 *
001F      126 LOAD_OVERLAY:
001F B7     127      OR      A
          128 *
          129 *
0020      130 LD_1:
          131 :      BEGIN      {Ordinarily registers are restored; retain only the pushes and pops you need.}
          132 *
          133 *
          134 *
0020 320000 135      LD      [OVERLAY_NUMBER],A
          136 *
          137 *
0023 F5     138      PUSH     AF *****;
0024 C0000F 139      CALL     CALC_OCB_ADDR
          140      POP      AF -----;
          141 *
          142      JR      NC,LOAD_BLOCKS
          143
002A      144 WRITE_BLOCKS:
002A 3E82   145      LD      A,WRITE_TAPE
          146      JR      BLOCK_IO
          147      LD      A,READ_TAPE
002E 1802   148
002E 3E81   148

```

;ADDR = OCB_TABLE(OV_NUM-1*5)

;GET THE OVERLAY NUMBER IN C

;POINTER TO THE OCB TABLE

;OCB_ADDR = OVERLAY_NUM*5 + START_OF_TABLE

;SET CARRY FLAG IF WRITE INSTRUCTION!

;RESET CARRY FLAG IF READ

;FOR DEBUGGING PURPOSES

;SAVE THE CARRY FLAG IF SET

;WRITE INSTRUCTIONS GO HERE

;READ INSTRUCTIONS GO HERE!

LOCATION OBJECT CODE LINE

SOURCE LINE

```
0030          149 BLOCK_ID:
0030 110003    150          LD      DE,CSA+1      ;      ;POINT TO THE COMMAND STATUS AREA
0033 010005    151          LD      BC,5          ;      ;NUMBER OF BYTES TO MOVE
0036 ED80      152          LDIR             ;
0038 110002    153          LD      DE,CSA
0038 12        154          LD      [DE],A      ;SET THE COMMAND IN THE CSA BUFFER
003C AF        155          XOR      A
          156          GLB      EXIT_TAPE
003D          157 EXIT_TAPE:
003D C9        158          RET
          159 *****
          160 DATA
0000          161 OVERLAY_NUMBER DEFS 1
0001          162 TAPE_STATE      DEFS 1
0002          163 CSA              DEFS 6
          <0003> 164 XFER_ADDR      EQU CSA+1
          <0005> 165 BLOCK_NUM      EQU XFER_ADDR+2
          <0007> 166 RANGE_         EQU BLOCK_NUM+2
```

Errors= 0

LINE#	SYMBOL	TYPE	REFERENCES
102	ABORT_TAPE	P	51
143	BLOCK_ID	P	57,146
165	BLOCK_NUM	D	75,166
109	CALC_DCB_ADDR	P	54,139
163	CSA	D	73,96,104,150,153,164
157	EXIT_TAPE	P	98,105,156
88	KILL_TAPE	A	85,103
147	LOAD_BLOCKS	P	56,142
126	LOAD_OVERLAY	P	52
130	LO_1	P	123
68	DCB_PTR	E	113
161	OVERLAY_NUMBER	D	71,135
166	RANGE_	D	76
86	READ_TAPE	A	83,148
162	TAPE_STATE	D	72
95	TEST_TAPE	P	50
***	WRITE	U	34
144	WRITE_BLOCKS	P	55
121	WRITE_OVERLAY	P	53
87	WRITE_TAPE	A	84,145
164	XFER_ADDR	D	74,165

1.10 DDP MANAGER

Tue, 15 May 1984, 20:29 PAGE 1

FILE: DDP_MANAG:IOS HEWLETT-PACKARD: DDP_MANAGER (c) Coleco 1983 Confidential

LOCATION OBJECT CODE LINE SOURCE LINE

```
1 ^Z80^
3 NAME ^Rev 01 - DTT^
4
5 De_DDP_MANAGER MACRO ;Header Rev. 5
6 .GOTO Ede_DDP_MANAGER
7
8 Project: M132, VS
9
10 *****
11 *
12 * DDP_MANAGER DTT *
13 *
14 *****
15
16 Rev History
17 Rev. Date Name Change
18 1 DTT DEVICE_ID --> DEV_ID
19 0 9/9/83 DTT Initial Pseudo code
20
21 Function:
22
23 CONTRDLS THE DIGITAL DATA PACK FOR READS AND WRITES SETUP BY TAPE_INTERFACE
24
25 Ede_DDP_MANAGER MEND
26 Pseudo_code_DDP_MANAGER MACRO ;Pseudocode macro area
27 .GOTO Ep_DDP_MANAGER
28
29
30
31 Ep_DDP_MANAGER MEND
```

LOCATION OBJECT CODE LINE SOURCE LINE

```

<0013>  + D_DEV_TYPE          EQU    19      ; THE DEVICE TYPE, BLOCKED OR CHARACTER
<0014>  + D_STATUS_FLAGS      EQU    20      ; DEVICE DEPENDENT STATUS FLAGS
<0015>  + D_SIZE              EQU    21      ; THE NUMBER OF BYTES IN THE DCB
      +
      + ;DEVICE ID'S FOR THE KEYBOARD, PRINTER, AND TAPE DRIVE
      +
<0001>  + KEYBOARD_ID          EQU     1      ; KYBD ID
<0002>  + PRINTER_ID          EQU     2      ; PRINTER ID
<0008>  + TAPE_ID              EQU     8      ; TAPE DRIVE ID
<0002>  + ERROR_RETRY         EQU     2      ; MAX RETRYS ON ERRORS, READ_BLOCK AND WRITE_BLOCK
<000F>  + MAX_DEV_ADDR        EQU    15      ; HIGEST POSSIBLE DEVICE ADDRESS
      + ; ON NETWORK
      +
      + ;PCB COMMAND EQUATES
      +
<0000>  + PCB_IDLE             EQU     0      ; THIS IS AN IDLE STATE
<0001>  + PCB_SYNC1           EQU     1      ; SYNC BYTE 1
<0081>  + PCB_SYNC1_ACK       EQU    PCB_SYNC1+80H
<0002>  + PCB_SYNC2           EQU     2      ; SYNC BYTE 2
<0082>  + PCB_SYNC2_ACK       EQU    PCB_SYNC2+80H
<0003>  + PCB_SNA             EQU     3      ; SET NEW PCB ADDRESS
<0083>  + PCB_SNA_ACK         EQU    PCB_SNA+80H
<0004>  + PCB_RESET           EQU     4      ; RESET ALL NODES
<0084>  + PCB_RESET_ACK       EQU    PCB_RESET+80H
<0005>  + PCB_WAIT            EQU     5      ;
<0085>  + PCB_WAIT_ACK        EQU    PCB_WAIT+80H
      +
      + ;DCB COMMAND EQUATES
      +
<0000>  + DCB_IDLE            EQU     00     ;
<0001>  + DCB_STATUS          EQU     01     ; REQUEST STATUS
<0002>  + DCB_RESET           EQU     02     ; RESET NODE
<0003>  + DCB_WR              EQU     03     ; WRITE DATA TO DEVICE
<0004>  + DCB_RD              EQU     04     ; READ DATA FROM DEVICE
      +
      +
<FEC0>  + INIT_PCB_ADDR        EQU    OFEC0H  ; INITIAL ADDRESS OF THE PCB
      +
      +

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

      + ;GENERAL USAGE EQUATES FOR USE WITH DCB INFO
      +
<0007> + CMND_COMPLETE_BIT      EQU      7      : THIS IS THE BIT THAT INDICATES THE
      +                      : COMMAND HAS BEEN PROCESSED.
<0080> + CMND_FIN_STATUS        EQU      80H     : THIS IS THE STATUS OF A COMMAND
      +                      : THAT COMPLETED WITH NO ERRORS
<008C> + KBD_NAK                EQU      8CH     : INDICATES NO KEY READY
      +
<0086> + PR_NAK                 EQU      86H     : INDICATES THE PRINTER IS BUSY
      +
<0003> + ETX                    EQU      03H     : END OF DATA STRING INDICATOR
      +
<0098> + TIMEOUT                EQU      98H     : DEVICE TIMED OUT
      +
43  INCLUDE EOS_ERRS:EOS
      +
<0001> + DCB_NOT_FOUND          EQU      1      : THERE WAS NO DCB FOR THE DEVICE REQUESTED.
<0002> + DCB_BUSY               EQU      2      : DCB IS BUSY
<0003> + DCB_IDLE_ERR           EQU      3      : DCB IS IDLE
      +
<0004> + NO_DATE_ERR            EQU      4
<0005> + NO_FILE_ERR            EQU      5
<0006> + FILE_EXISTS_ERR        EQU      6
<0007> + NO_FCB_ERR             EQU      7
<0008> + MATCH_ERR              EQU      8
<0009> + BAD_FNUM_ERR           EQU      9
<000A> + EOF_ERR                EQU      10
<000B> + TOO_BIG_ERR            EQU      11
<000C> + FULL_DIR_ERR           EQU      12
<000D> + FULL_TAPE_ERR          EQU      13
<000E> + FILE_NM_ERR            EQU      14      :DLS(8/28/83)
<000F> + RENAME_ERR             EQU      15      :DLS(8/30/83)
<0010> + DELETE_ERR             EQU      16      :DLS(8/30/83)
<0011> + RANGE_ERR              EQU      17      :DLS(8/31/83)
      +
<0012> + CANT_SYNC1             EQU      18
<0013> + CANT_SYNC2             EQU      19
<0014> + PRT_ERR                EQU      20
      +
<0015> + RQ_TP_STAT_ERR         EQU      21
<0016> + DEVICE_DEPD_ERR        EQU      22
44
45 *EXTERNAL DATA AREAS USED:
46
47          EXT _START_RD_1_BLOCK
48          EXT _FIND_DCB
49          EXT _START_WR_1_BLOCK
50          EXT _END_RD_1_BLOCK
51
52 NEXT_STATE      MACRO &P1
53          LD      A,&P1                      :SOMETHING IN THE COMMAND BUFFER!
54          LD      EDI,TAPE_STATE],A
55          LD      HL,ESTATE_VECTORS+&P1+&P1]
56          LD      ENEXT_STATE_ADDRESS],HL
57          JP      END_OF_STATE_MACHINE
58          MEND
59

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

60
61
62      GLB      DEV_ID
63      GLB      INITIALIZE_DDP,INITIALIZE_TAPE,INIT_TAPE,INIT_DDP
0000      64 INITIALIZE_DDP:
0000      65 INITIALIZE_TAPE:
0000      66 INIT_TAPE:
0000      67 INIT_DDP:
0000      68 INIT_CODE:
0000 AF      69      XOR      A
0001 320002   70      LD      [D_CSA],A
0004 320001   71      LD      [D_TAPE_STATE],A
0007 30       72      DEC      A
0008 320000   73      LD      [D_OVERLAY_NUMBER],A
0008 2A0012   74      LD      HL,[STATE_VECTORS+0000]      ;IDLE STATE
000E 2200D2   75      LD      [NEXT_STATE_ADDRESS],HL
0011 C9      76      RET
0012          77 STATE_VECTORS:
0012 001F      78      DEFW     STATE_IDLE      ;STATE 0
0014 0026      79      DEFW     STATE_1      :      1  REQUEST I/O FOR 1 BLOCK
0016 0058      80      DEFW     STATE_2      :      2  TEST FOR COMPLETE AND REQUEST STATUS
0018 0078      81      DEFW     STATE_3      :      3  TEST STATUS
          <001A> 82 LEN_INIT EQU $-INIT_CODE
83 *
84 *
85
001A          86      DEFS     1BH-LEN_INIT      MAKE SURE THE MANAGER VECTOR IS AT THE SAME LOC AS THE S/I/O_MANAGER
87
88
89      GLB      DDP_MANAGER
90      GLB      TAPE_MANAGER
0018          91 TAPE_MANAGER:
0018          92 DDP_MANAGER:
93 * BEGIN      {Ordinarily registers are restored; retain only the pushes and pops you need.}
94 *
95 *
96 *
97 *          FALL THRU TO CASE STATEMENT      ;IF STATE = 6,7
98 *          ;FILE IS TRYING TO CLOSE
99 *
100 *
101 *          CASE      D_TAPE_STATE,(IDLE,STATE_1,STATE_2,STATE_3,INIT_DDP)
102 *
0018          103 CASE_STATE:
104 *
105 *
0018 2A00D2   106      LD      HL,[NEXT_STATE_ADDRESS]
001E E9      107      JP      [HL]
108 *
109 *          IF THE MACHINE IS IDLE IT'S OK TO TEST FOR ANOTHER I/O REQUEST
110 *
001F          111 STATE_IDLE:      ;STATE 0
001F 3A0002   112      LD      A,[D_CSA]      ;TEST THE COMMAND STATUS AREA
0022 87      113      OR      A
0023 F200CF   114      JP      P,END_OF_STATE_MACHINE      ;IF THE CSA CONTAINS AN ERROR CODE DON'T PROCESS IT
115 *
116 *          FALL THROUGH TO STATE 1

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

117 *
0026 118 STATE_1:
0026 3A0002 119 AB_REQ: LD A,ED_CSAJ ;IF COMMAND IS TO KILL DDP COMMAND
0029 FE87 120 CP KILL_TAPE
002B CA0000 121 JP Z,INIT_ODP
122 *
123 * REQUEST TO WRITE/READ A RECORD
124 *
125 *
002E 2A0003 126 LD HL,EXFER_ADDRJ
0031 ED580005 127 LD DE,CBLOCK_NUMJ
0035 010000 128 LD BC,0000H
0038 3A0002 129 LD A,ED_CSAJ
003B FE81 130 CP READ_TAPE
003D 3A0001 131 LD A,CDEV_IDJ
0040 2005 132 JR NZ,EOS_TAPE_WRITE
0042 133 EOS_TAPE_READ:
0042 CD0000 134 CALL _START_RD_1_BLOCK
0045 1803 135 JR RET_ADDR
0047 136 EOS_TAPE_WRITE:
0047 CD0000 137 CALL _START_WR_1_BLOCK
138
139
004A 140 RET_ADDR:
004A 141 NEXT_STATE 2
004A 3E02 + LD A,2 ;SOMETHING IN THE COMMAND BUFFER!
004C 320001 + LD ED_TAPE_STATEJ,A
004F 2A0016 + HL,CSTATE_VECTORS+2+2J
0052 2200D2 + LD CNEXT_STATE_ADDRESSJ,HL
0055 C300CF + JP END_OF_STATE_MACHINE
142
143 *
144 * TEST FOR ACCEPTANCE/COMPEITION OF I/O REQUEST
145 *
0058 146 STATE_2:
0058 3A00D1 147 LD A,CDEV_IDJ ;TEST THE STATUS OF THE FILE
005B CD0000 148 CALL _END_RD_1_BLOCK
005E D200CF 149 JP NC,END_OF_STATE_MACHINE ;BETTER RETRY STATE 2
150 *
151 * AT THIS POINT THE COMMAND HAS BEEN ACCEPTED BY THE NETWORK
152 * IF THE ZERO FLAG IS NOT SET THERE HAVE BEEN ERRORS (09BH=TIMEOUT)
153 *
154 *
0061 20C3 155 JR NZ,STATE_1 ;BETTER RETRY STATE 1
156 *
157 * IF NO ERRORS THEN SET UP REQUEST STATUS OF THE TAPE DRIVE
158 * THIS TESTS THE CHECK SUM (CRC) OF THE DATA XMITTED BY THE TAPE
159 *
0063 160 STATE_2_OK:
0063 161 *
0063 3A00D1 162 LD A,CDEV_IDJ ;STATUS COMMAND
0066 CD0000 163 CALL _FIND_DCB
0069 FD360001 164 LD C[Y+D_COM_STATJ],DCB_STATUS
006D 165 NEXT_STATE 3
006D 3E03 + LD A,3 ;SOMETHING IN THE COMMAND BUFFER!
006F 320001 + LD ED_TAPE_STATEJ,A
0072 2A0018 + HL,CSTATE_VECTORS+3+3J

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

0075 2200D2      *      LD      [NEXT_STATE_ADDRESS],HL
0078 C300CF      *      JP      END_OF_STATE_MACHINE

166 *
167 *      IN STATE 3 TEST FOR ACCEPTANCE OF THE REQUEST FOR STATUS COMMAND
168 *
0078          169 STATE_3:
0078 3A00D1      170      LD      A,[DEV_ID]
007E C00000      171      CALL _FIND_DEV      ;WAS THE STATUS REQUEST ACKNOWLEDGED???
0081 FDC9007E      172      BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT]
0085 CA00CF      173      JP      Z,END_OF_STATE_MACHINE
174 *
175 *      IF THE COMMAND IS NOT YET ACCEPTED, COME BACK TO STATE 3 NEXT PASS
176 *
0088 FD7E00      177      LD      A,[IY+D_COM_STAT]      ;STATUS COMMAND ACCEPTED
0088 FE80      178      CP      080H
0080 20D4      179      JR      NZ,STATE_2_OK      ;COULD NOT GET STATUS?
180 *
181 *
182 *
183          ;SAVE THE STATUS IN REGISTER B
008F 3A00D1      184      LD      A,[DEV_ID]      ;CHECK FOR SECONDARY DEVICE ID
0092 E6F0      185      AND     0F0H
0094 FD7E14      186      LD      A,[IY+D_STATUS_FLAGS]
0097 2808      187      JR      Z,DEV_0_CHECK      ;SECONDARY_DEV_ID =0
0099 CB1F      188      RR      A      ;GET STATUS INTO THE LOW NIBBLE
009B CB1F      189      RR      A
009D CB1F      190      RR      A
009F CB1F      191      RR      A
00A1          192 DEV_0_CHECK:
00A1 E60F      193      AND     0FH
00A3 C200C6      194      JP      NZ,ERROR      ;A BAD STATUS
195 *
196 *      IF THE COMMAND IS ACCEPTED AND THERE ARE NO ERRORS INCREMENT THE CSA DATA
197 *
00A6 2A00D3      198      LD      HL,[XFER_ADDR]      ; A BLOCK HAS BEEN WRITTEN OR READ
00A9 ED5B0005      199      LD      DE,[BLOCK_NUM]      ;UPDATE THE D_CSA AREA!
00AD 010400      200      LD      BC,0400H
00B0 3A00D7      201      LD      A,[RANGE]
0033 3D      202      DEC     A
00B4 CA00D0      203      JP      Z,INIT_DDP      ;DONE WITH THE OVERLAY?
204 *
205 *      IF THE RANGE WAS DECREMENTED TO ZERO WE ARE DONE ELSE (WE ARE NOT?)
206 *
00B7 3200D7      207      LD      [RANGE],A
00BA 13      208      INC     DE
003B ED530005      209      LD      [BLOCK_NUM],DE
00BF 09      210      ADD     HL,BC
00C0 2200D3      211      LD      [XFER_ADDR],HL
212
00C3 C30026      213      JP      STATE_1
214
215 *
216 *      ERROR PROCESSING FROM REQUEST STATUS
217 *
00C6          218 ERROR:
219 *      Errors occur after request status
220 *      Possible errors are 1=CRC check (bad data on tape)

```

LOCATION OBJECT CODE LINE SOURCE LINE

```

221 *          2=block not found  3=no tape in drive  4=no drive
222 *          The user program is expected to test for errors via
223 *          TEST_TAPE in the TAPE_INTERFACE module.
224 *
225 *
00C6 320002 226          LD      [D_CSA],A          ;SAVE THE ERROR CODE IN THE CSA
00C9 2A0012 227          LD      HL,[STATE_VECTORS+0] ;NEXT STATE = IDLE
00CC 220002 228          LD      [NEXT_STATE_ADDRESS],HL
229
230
231 *  END (DDP_MANAGER)
232
00CF          233  END_OF_STATE_MACHINE:
00CF AF      234          XOR  A
00D0 C9      235          RET
236 *****
237 *****          DEV_ID IS DEFAULT 08 FOR TAPE DRIVE 0          *****
238 *****
239
00D1 08      241  DEV_ID          DEFB  TAPE1
242
243
244 *****
245 *****
246 *****
247          GLB  D_OVERLAY_NUMBER
248          GLB  D_TAPE_STATE
249          GLB  D_CSA
250 *-----
251 *THE NEXT_STATE_ADDRESS MAY BE PUT IN DATA,PROG,OR COMN (IT'S ALL RAM TO ADAM)
252 *
00D2          253  NEXT_STATE_ADDRESS DEFS 2
254 *-----
255 *          BY MAKING THE CSA COMMON IT IS EASILY LINKED
256 *          INTO EXISTING CODE
257  DATA
00D0          258  D_OVERLAY_NUMBER DEFS 1
00D1          259  D_TAPE_STATE      DEFS 1
00D2          260  D_CSA          DEFS 6
<0003>      261  XFER_ADDR EQU D_CSA+1
<0005>      262  BLOCK_NUM EQU XFER_ADDR+2
<0007>      263  RANGE      EQU BLOCK_NUM+2

```

Errors= 0

LINE#	SYMBOL	TYPE	REFERENCES
119	AB_REQ	P	
43	BAD_FNUM_ERR	A	
262	BLOCK_NUM	D	127,199,209,263
43	CANT_SYNC1	A	
43	CANT_SYNC2	A	
103	CASE_STATE	P	
42	CMND_COMPLETE_B	A	172
42	CMND_FIN_STATUS	A	
43	DCB_BUSY	A	
42	DCB_IDLE	A	
43	DCB_IDLE_ERR	A	
43	DCB_NOT_FOUND	A	
42	DCB_RD	A	
42	DCB_RESET	A	
42	DCB_STATUS	A	164
42	DCB_WR	A	
92	DDP_MANAGER	P	89
43	DELETE_ERR	A	
43	DEVICE_DEPD_ERR	A	
192	DEV_O_CHECK	P	187
241	DEV_ID	P	62,131,147,162,170,184
42	D_BUF_ADR	A	42,42
42	D_BUF_ADR_HI	A	
42	D_BUF_ADR_LO	A	
42	D_BUF_LEN	A	42,42
42	D_BUF_LEN_HI	A	
42	D_BUF_LEN_LO	A	
42	D_COM_STAT	A	164,172,177
260	D_CSA	D	70,112,119,129,226,249,261
42	D_DEV_ADDR	A	
42	D_DEV_TYPE	A	
42	D_MAX_MSG_LEN	A	42,42
42	D_MAX_MSG_LEN_HI	A	
42	D_MAX_MSG_LEN_LO	A	
258	D_OVERLAY_NUMBE	D	73,247
42	D_RET_COUNT	A	42,42
42	D_RET_COUNT_HI	A	
42	D_RET_COUNT_LO	A	
42	D_SECT_NUM	A	
42	D_SEC_DEV_ID	A	
42	D_SIZE	A	
42	D_STATUS_FLAGS	A	186
259	D_TAPE_STATE	D	54,71,248
233	END_OF_STATE_MA	P	57,114,149,173
43	EOF_ERR	A	
133	EOS_TAPE_READ	P	
136	EOS_TAPE_WRITE	P	132
218	ERROR	P	194
42	ERROR_RETRY	A	
42	ETX	A	
43	FILE_EXISTS_ERR	A	
43	FILE_NM_ERR	A	
43	FULL_DIR_ERR	A	
43	FULL_TAPE_ERR	A	
64	INITIALIZE_DDP	P	63
65	INITIALIZE_TAPE	P	63
68	INIT_CODE	P	82

LINE#	SYMBOL	TYPE	REFERENCES
67	INIT_DCP	P	63,121,203
42	INIT_PCB_ADDR	A	
66	INIT_TAPE	P	63
42	KBD_NAK	A	
42	KETBOARD_ID	A	
41	KILL_TAPE	A	120
82	LEN_INIT	A	86
43	MATCH_ERR	A	
42	MAX_DEV_ADDR	A	
253	NEXT_STATE_ADDR	P	56,75,106,228
43	NO_DATE_ERR	A	
43	NO_FCB_ERR	A	
43	NO_FILE_ERR	A	
42	PCB_IDLE	A	
42	PCB_RESET	A	42
42	PCB_RESET_ACK	A	
42	PCB_SNA	A	42
42	PCB_SNA_ACK	A	
42	PCB_SYNC1	A	42
42	PCB_SYNC1_ACK	A	
42	PCB_SYNC2	A	42
42	PCB_SYNC2_ACK	A	
42	PCB_WAIT	A	42
42	PCB_WAIT_ACK	A	
42	PRINTER_ID	A	
43	PRT_ERR	A	
42	PR_NAK	A	
42	P_COM_STAT	A	
42	P_NUM_DCBS	A	
42	P_REL_ADDR	A	42,42
42	P_REL_ADDR_HI	A	
42	P_REL_ADDR_LO	A	
42	P_SIZE	A	
263	RANGE	D	201,207
43	RANGE_ERR	A	
39	READ_TAPE	A	130
43	RENAME_ERR	A	
140	RET_ADDR	P	135
43	RQ_TP_STAT_ERR	A	
118	STATE_1	P	79,155,213
146	STATE_2	P	80
160	STATE_2_OK	P	179
169	STATE_3	P	81
111	STATE_IDLE	P	78
77	STATE_VECTORS	P	55,74,227
38	TAPE1	A	241
42	TAPE_ID	A	
91	TAPE_MANAGER	P	90
42	TIMEOUT	A	
43	TOO_BIG_ERR	A	
40	WRITE_TAPE	A	
261	XFER_ADDR	D	126,198,211,262
50	_END_RD_1_BLOCK	E	149
48	_FIND_DCB	E	163,171
47	_START_RD_1_BLO	E	134
49	_START_WR_1_BLO	E	137

1.11

DDP INTERFACE

FILE: DDP_INTER.TDS HEWLETT-PACKARD: DDP_INTERFACE (c) Coleco 1984 Confidential

Mon, 21 May 1984, 16135

PAGE 1

LOCATION OBJECT CODE LINE SOURCE LINE

```
1 ^Z80^
3 NAME ^Rev 01 - DDT^
4
5 De_DDP_INTERFACE MACRO           ;Header Rev. 5
6     .GOTO Ed_DDP_INTERFACE
7
8 Project:      TAPE, C101
9
10 *****
11 *
12 * DDP_INTERFACE      DDT      *
13 *
14 *****
15
16 Rev History
17 Rev.  Date      Name      Change
18 1      9/13/83  DDT      CHANGED TO ALLOW ERROR RETIRES
19 0      7/5/83  DDT      Initial Pseudo code
20
21 Function:
22 REQUEST READS AND WRITES AS DEFINED IN OCB.
23 REQUESTS ABORT_TAPE.
24 TEST STATUS OF TAPE REQUEST.
25
26
27 Ed_DDP_INTERFACE MEND
28 Pseudo_code_DDP_INTERFACE MACRO ;Pseudocode macro area
29 BEGIN:
30     STORE D_OVERLAY_NUMBER
31     HL := POINTER TO OCB := D_OVERLAY_NUMBER*5 + OVERLAY_TABLE_POINTER
32     MOVE OCB TO O_CSA
33     IF WRITE THEN
34         SEND WRITE_COMMAND
35     ELSE
36         SEND READ_COMMAND
37     ENDIF
38 END
39     .GOTO Ep_DDP_INTERFACE
40
41
42
43 Ep_DDP_INTERFACE MEND
```

LOCATION OBJECT CODE LINE

SOURCE LINE

```

45 ;Subroutines called
46 ;     EXT
47
48 ;Subroutines defined
49     GL3     TEST_TAPE
50     GLB     ABJRT_TAPE
51     GLB     LOAD_OVERLAY
52     GLB     WRITE_OVERLAY
53     GLB     CALC_DCB_ADDR
54     GLB     WRITE_BLOCKS
55     GL3     LOAD_BLOCKS
56     GL3     BLOCK_ID
57 ;     GLB
58
59 ;Operating system calls
60 ;     EXT
61
62 ;Inputs/Outputs passed in registers
63 ;     A = OVERLAY NUMBER 1 thru N
64 ;     A <> 0 = ERROR
65
66 ;External data areas used
67     EXT     DCB_PTR
68
69 ;Global data areas defined
70     EXT     D_OVERLAY_NUMBER
71     EXT     D_TAPE_STATE
72     EXT     D_CSA
73 ;     GL3
74
75 ;Local equates
76 ;     EQU
77
78 ;Global equates
79     GLB     READ_TAPE
80     GLB     WRITE_TAPE
81     GLB     KILL_TAPE
<0081> 82 READ_TAPE EQU B1H
<0082> 83 WRITE_TAPE EQU B2H
<0087> 84 KILL_TAPE EQU B7H
85 ;     INCLUDE File_name:user_id
86

```

;POINTER TO THE OVERLAY CONTROL TABLE

```

0000      89      PRG
0000      90      ;      TEST FOR COMPLETION OF IO REQUEST
0000      91      TEST_TAPE
0000      92      LD  A,[D_CSA]
0000      93      OR  A
0000      94      JP      EXIT_TAPE
0000      95      *
0000      96      *
0000      97      *
0000      98      ABORT_TAPE
0000      99      LD      A,KILL_TAPE
0000     100      LD      [D_CSA],A
0000     101      JP      EXIT_TAPE
0000     102      *
0000     103      *
0000     104
0000     105  CALC_OCB_ADDR:
0000     106      DEC      A
0000     107      LD      C,A
0000     108      LD      B,0
0000     109      LD      HL,[OCB_PTR]
0000     110      ADD     HL,BC
0000     111      ADD     HL,BC
0000     112      ADD     HL,BC
0000     113      ADD     HL,BC
0000     114      ADD     HL,BC
0000     115      RET
0000     116      *
0000     117  WRITE_OVERLAY:
0000     118      SCF
0000     119      JR      LO_1
0000     120      *
0000     121      *
0000     122  LOAD_OVERLAY:
0000     123      OR      A
0000     124      *
0000     125      *
0000     126  LO_1:
0000     127      ;      BEGIN      (Ordinarily registers are restored; retain only the pushes and pops you need.)
0000     128      *
0000     129      *
0000     130      *
0000     131      LD      [D_OVERLAY_NUMBER],A
0000     132      *
0000     133      *
0000     134      PUSH     AF  ++++++++++;
0000     135      CALL     CALC_OCB_ADDR
0000     136      POP      AF  -----;
0000     137      *
0000     138      JR      NC,LOAD_BLOCKS
0000     139
0000     140  WRITE_BLOCKS:
0000     141      LD      A,WRITE_TAPE
0000     142      JR      BLOCK_ID
0000     143  LOAD_BLOCKS:
0000     144      LD      A,READ_TAPE

```

LOCATION OBJECT CODE LINE

SOURCE LINE

```
0030          145 BLOCK_ID:
0030 110001    146          LD      DE,D_CSA+1          ;      ;POINT TO THE COMMAND STATUS AREA
0033 010005    147          LD      BC,5                ;      ;NUMBER OF BYTES TO MOVE
0036 E080      148          LDIR                     ;
0038 110000    149          LD      DE,D_CSA
0038 12        150          LD      [DE],A              ;SET THE COMMAND IN THE D_CSA BUFFER
003C AF        151          XCR      A
          152          GLB      EXIT_TAPE
003D          153 EXIT_TAPE:
003D C9        154          RET
          155 *****
```

Errors = 0

LINE#	SYMBOL	TYPE	REFERENCES
98	ABORT_TAPE	D	50
145	BLOCK_ID	P	56,142
105	CALC_OCB_ACOR	P	53,135
72	D_CSA	E	92,100,146,149
70	D_OVERLAY_NUM3E	E	131
71	D_TAPE_STATE	E	
153	EXIT_TAPE	P	94,101,152
84	KILL_TAPE	A	81,99
143	LOAD_BLOCKS	P	55,138
122	LOAD_OVERLAY	P	51
126	LG_1	P	119
67	OCB_PTR	E	109
82	READ_TAPE	A	79,144
91	TEST_TAPE	P	47
***	WRITE	J	33
140	WRITE_BLOCKS	P	54
117	WRITE_OVERLAY	P	52
83	WRITE_TAPE	A	80,141

1. Keyboard Table

**** KEYCODES GENERATED BY ADAM COMPUTER KEYBOARD ****

DEC	HEX	ASCII DATA	CV KEY(S)	REPEAT	COMMENTS
0	00	NUL	cntrl 2	N	Null (substitute for cntrl @)
1	01	SOH	cntrl A	N	Start of Heading
2	02	STX	cntrl B	N	Start of Text
3	03	ETX	cntrl C	N	End of Text
4	04	EOT	cntrl D	N	End of Transmission
5	05	ENO	cntrl E	N	Enquiry
6	06	ACK	cntrl F	N	Acknowledge
7	07	BEL	cntrl G	N	Bell
8	08	BS	cntrl H or BACKSPACE	Y	Backspace (see NOTE 5)
9	09	HT	cntrl I or TAB	N	Horizontal Tabulation (see NOTE 6)
10	0A	LF	cntrl J	N	Line Feed
11	0B	VT	cntrl K	N	Vertical Tabulation
12	0C	FF	cntrl L	N	Form Feed
13	0D	CR	cntrl M or RETURN	N	Carriage Return
14	0E	SO	cntrl N	N	Shift Out
15	0F	SI	cntrl O	N	Shift In
16	10	DLE	cntrl P	N	Data Link Escape
17	11	DC1	cntrl Q	N	Device Control 1
18	12	DC2	cntrl R	N	Device Control 2
19	13	DC3	cntrl S	N	Device Control 3
20	14	DC4	cntrl T	N	Device Control 4
21	15	NAK	cntrl U	N	Negative Acknowledge
22	16	SYN	cntrl V	N	Synchronous Idle
23	17	ETB	cntrl W	N	End of Transmission Block
24	18	CAN	cntrl X	N	Cancel
25	19	EM	cntrl Y	N	End of Medium
26	1A	SUB	cntrl Z	N	Substitute
27	1B	ESC	cntrl [or WP/ESCAPE	N	Escape
28	1C	FS	cntrl \	N	File Separator
29	1D	GS	cntrl]	N	Group Separator
30	1E	RS	cntrl ^	N	Record Separator
31	1F	US	cntrl 6	N	Unit Separator (substitute for cntrl _)
32	20	SP	space bar	Y	Space
33	21	!	shift 1	Y	Exclamation Point
34	22	"	shift 2	Y	Quotation Marks (or double quotes)
35	23	#	shift 3	Y	Number Sign
36	24	\$	shift 4	Y	Dollar Sign
37	25	%	shift 5	Y	Percent
38	26	&	shift 7	Y	Amperсанд
39	27	'		Y	Apostrophe (or single quotes)
40	28	(	shift 9	Y	Opening Parenthesis
41	29	)	shift 0	Y	Closing Parenthesis
42	2A	*	shift 8	Y	Asterisk
43	2B	+	+	Y	Plus
44	2C	,	,	Y	Comma
45	2D	-	-	Y	Hyphen (Minus)
46	2E	.	.	Y	Period (Decimal Point)
47	2F	/	/	Y	Slant

D	HEX	ASCII DATA	CV KEY(S)	REPE	COMMENTS
48	30	0	0	Y	
49	31	1	1	Y	
50	32	2	2	Y	
51	33	3	3	Y	
52	34	4	4	Y	
53	35	5	5	Y	
54	36	6	6	Y	
55	37	7	7	Y	
56	38	8	8	Y	
57	39	9	9	Y	
58	3A	:	shift :	Y	Colon
59	3B	;	:	Y	Semicolon
60	3C	<	shift ,	Y	Less Than
61	3D	=	shift +	Y	Equals
62	3E	>	shift .	Y	Greater Than
63	3F	?	shift /	Y	Question Mark
64	40	@	shift 2	Y	Commercial At
65	41	A	shift A	Y	upper case
66	42	B	shift B	Y	" "
67	43	C	shift C	Y	" "
68	44	D	shift D	Y	" "
69	45	E	shift E	Y	" "
70	46	F	shift F	Y	" "
71	47	G	shift G	Y	" "
72	48	H	shift H	Y	" "
73	49	I	shift I	Y	" "
74	4A	J	shift J	Y	" "
75	4B	K	shift K	Y	" "
76	4C	L	shift L	Y	" "
77	4D	M	shift M	Y	" "
78	4E	N	shift N	Y	" "
79	4F	O	shift O	Y	" "
80	50	P	shift P	Y	" "
81	51	Q	shift Q	Y	" "
82	52	R	shift R	Y	" "
83	53	S	shift S	Y	" "
84	54	T	shift T	Y	" "
85	55	U	shift U	Y	" "
86	56	V	shift V	Y	" "
87	57	W	shift W	Y	" "
88	58	X	shift X	Y	" "
89	59	Y	shift Y	Y	" "
90	5A	Z	shift Z	Y	upper case
91	5B	[	[	Y	Opening Bracket
92	5C	\	\	Y	Reverse Slant
93	5D	]	]	Y	Closing Bracket
94	5E	^	^	Y	Circumflex
95	5F	_	shift 6	Y	Underline

C	HEX	ASCII DATA	CV KEY(S)	RE	COMMENTS
96	60	`	shift -	Y	Grave Accent
97	61	a	A	Y	lower case
98	62	b	B	Y	" "
99	63	c	C	Y	" "
100	64	d	D	Y	" "
101	65	e	E	Y	" "
102	66	f	F	Y	" "
103	67	g	G	Y	" "
104	68	h	H	Y	" "
105	69	i	I	Y	" "
106	6A	j	J	Y	" "
107	6B	k	K	Y	" "
108	6C	l	L	Y	" "
109	6D	m	M	Y	" "
110	6E	n	N	Y	" "
111	6F	o	O	Y	" "
112	70	p	P	Y	" "
113	71	q	Q	Y	" "
114	72	r	R	Y	" "
115	73	s	S	Y	" "
116	74	t	T	Y	" "
117	75	u	U	Y	" "
118	76	v	V	Y	" "
119	77	w	W	Y	" "
120	78	x	X	Y	" "
121	79	y	Y	Y	" "
122	7A	z	Z	Y	lower case
123	7B	{	shift [	Y	Opening Brace
124	7C		shift \	Y	Vertical Line
125	7D	}	shift]	Y	Closing Brace
126	7E	~	shift ^	Y	Tilde
127	7F	DEL	cntrl DELETE	Y	Delete (substitute for DEL)

** end of ASCII codes **

** s t of COLECO special codes (i.e. non-ASCII) defined by g **

a) SUFTKEY GROUP

DEC	HEX	CV KEY(S)	REPEAT	COMMENTS
128	80	HOME	N	group exception
129	81	I	N	softkey 1
130	82	II	N	softkey 2
131	83	III	N	softkey 3
132	84	IV	N	softkey 4
133	85	V	N	softkey 5
134	86	VI	N	softkey 6
135	87			unused code
136	88			unused code
137	89	shift I	N	
138	8A	shift II	N	
139	8B	shift III	N	
140	8C	shift IV	N	
141	8D	shift V	N	
142	8E	shift VI	N	
143	8F			unused code

b) WORD PROCESSOR "hard key" GROUP

DEC	HEX	CV KEY(S)	COMMENTS
144	90	WILD CARD	N
145	91	UNDO	N
146	92	MOVE	N
147	93	STORE	N
148	94	INSERT	N
149	95	PRINT	N
150	96	CLEAR	N
151	97	DELETE	N independent function from ASCII delete (DEL)
152	98	shift WILD CARD	N
153	99	shift UNDO	N
154	9A	shift MOVE	N (COPY)
155	9B	shift STORE	N (FETCH)
156	9C	shift INSERT	N
157	9D	shift PRINT	N
158	9E	shift CLEAR	N
159	9F	shift DELETE	N independent function from ASCII delete (DEL)

c) CURSOR CONTROL GROUP

DEC	HEX	CV KEY(S)	REPEAT	COMMENTS
160	A0	up arrow	Y	north
161	A1	right arrow	Y	east
162	A2	down arrow	Y	south
163	A3	left arrow	Y	west
164	A4	cntrl up arrow	Y	
165	A5	cntrl right arrow	Y	
166	A6	cntrl down arrow	Y	
167	A7	cntrl left arrow	Y	
168	A8	up arrow + right arrow	Y	northeast - sequence independent, time critical
169	A9	right arrow + down arrow	Y	southeast - " " " "
170	AA	down arrow + left arrow	Y	southwest - " " " "
171	AB	left arrow + up arrow	Y	northwest - sequence independent, time critical
172	AC	HOME + up arrow	N	sequence independent, time critical
173	AD	HOME + right arrow	N	" " " "
174	AE	HOME + down arrow	N	" " " "
175	AF	HOME + left arrow	N	sequence independent, time critical

d) GENERAL KEY GROUP

DEC	HEX	CV KEY(S)	COMMENTS
176	B0		unused code
177	B1		" "
178	B2		" "
179	B3		" "
180	B4		" "
181	B5		" "
182	B6		" "
183	B7		unused code
184	B8	shift BACKSPACE	Y (see NOTE 5)
185	B9	shift TAB	N (see NOTE 6)
186	BA		unused code
187	BB		" "
188	BC		" "
189	BD		" "
190	BE		" "
191	BF		unused code

** end of COLECO special codes defined by group **

** N 5 **

NOTE 1: The lock key will act as a "shift lock function," i.e., when in the active state (on) all keys will behave as if they were produced by their shifted key versions. Of course, when the lock key is inactive (off) all key depressions will be treated as normal unshifted key depressions. The foregoing description is analogous in operation to that of a standard keyboard.

NOTE 2: The remaining codes 0C0H thru 0EFH are unused codes.

NOTE 3: Codes 0F0H thru 0FFH are reserved for internal use by the keyboard software.

NOTE 4: The following keys have no code assigned to them, they are used internally by the keyboard software to calculate the key value, CNTRL, SHIFT and LOCK. No serial transmission occurs for these keys.

NOTE 5: Codes 008H and 0B8H are provided for purposes of non-destructive and destructive BACKSPACE. The interpretation of these codes are application dependent. It is recommended that the following convention be used:

008H = BACKSPACE (as defined by ASCII)
0B8H = destructive BACKSPACE

NOTE 6: Codes 009H and 0B9H are provided for purposes of right TAB and left TAB. The interpretation of these codes are application dependent. It is recommended that the following convention be used:

009H = right TAB (as defined by ASCII)
0B9H = left TAB

2. ADAM Emulation Considerations

ADAM hardware characteristics affect the selection and interface of an emulator for ADAM.

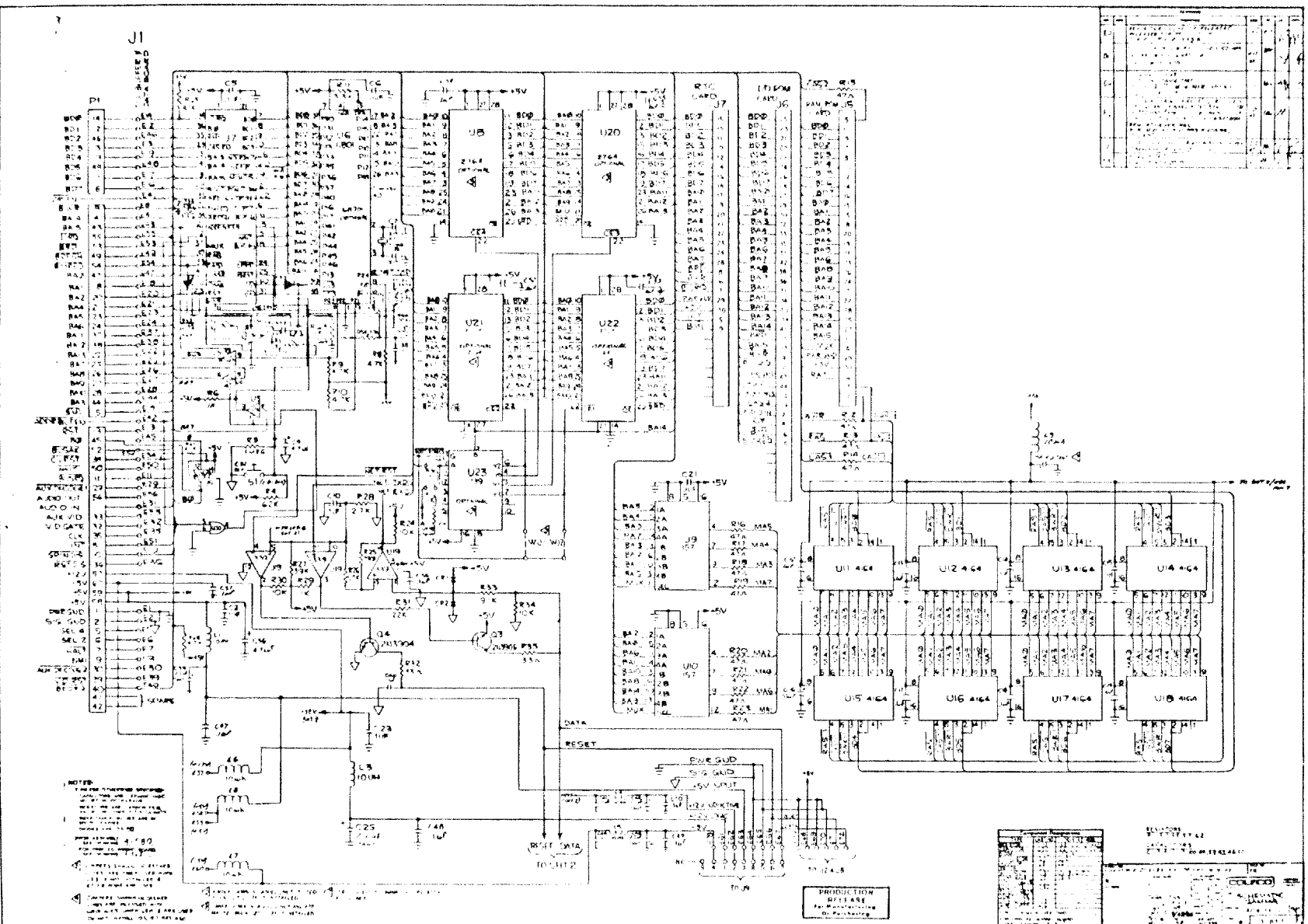
ADAM has dynamic RAMs that must be refreshed to maintain integrity. The RAMs require an 8-bit refresh. Since the Z80 performs a 7-bit refresh, the eighth bit is manufactured by the MIOC, using other signals from the Z80. In general, the signals are:

MREQ
M1
WAIT
REFRESH (RA0 - RA6)
A7

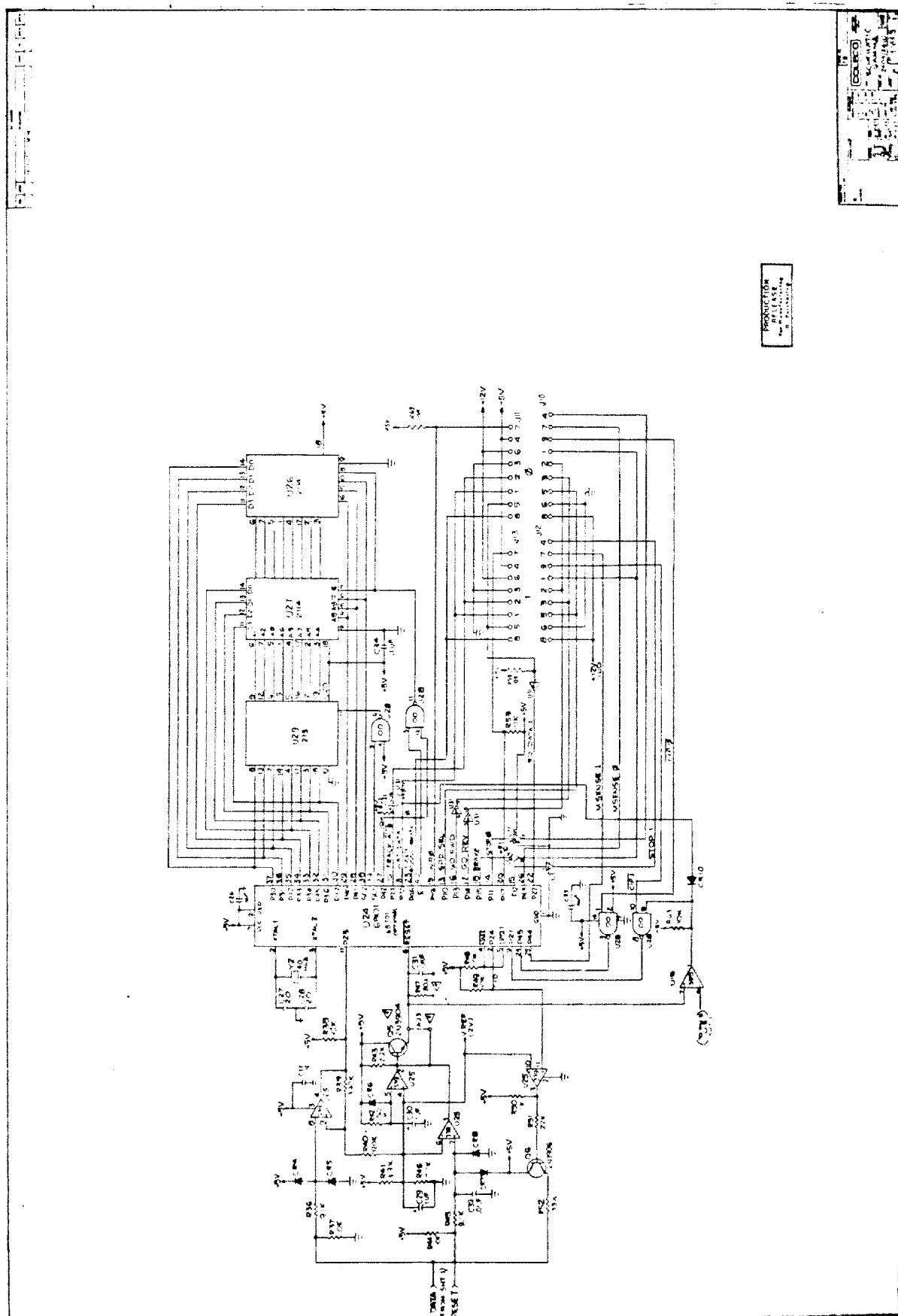
The Master 6801 performs a direct memory access into the 64K intrinsic RAM addressed by the Z80. The MIOC is responsible for the setup and execution of DMA by the Master 6801.

Some emulators place a load on the clock circuit that drives the Z80. Problems with an interface to an emulator may require a check of the clock and a modification to R60 on the CPU Board.

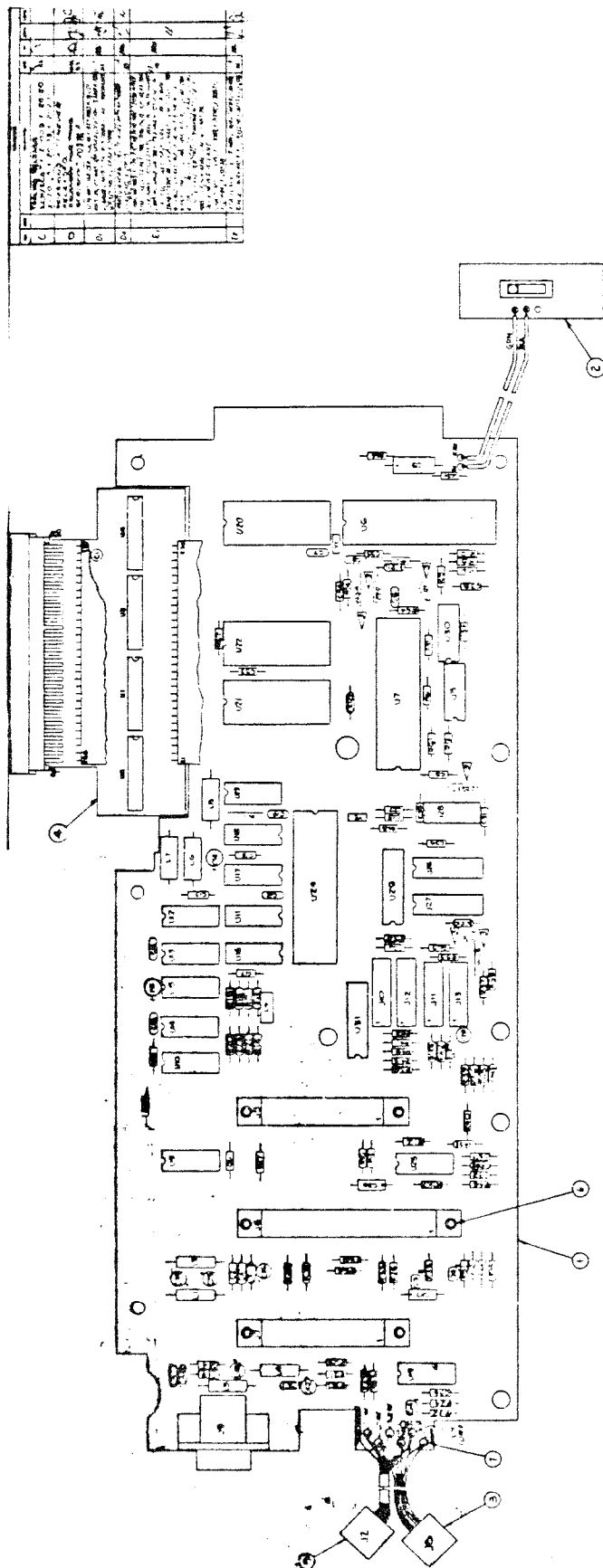
4. Schematics and Component Location/Identification Drawings



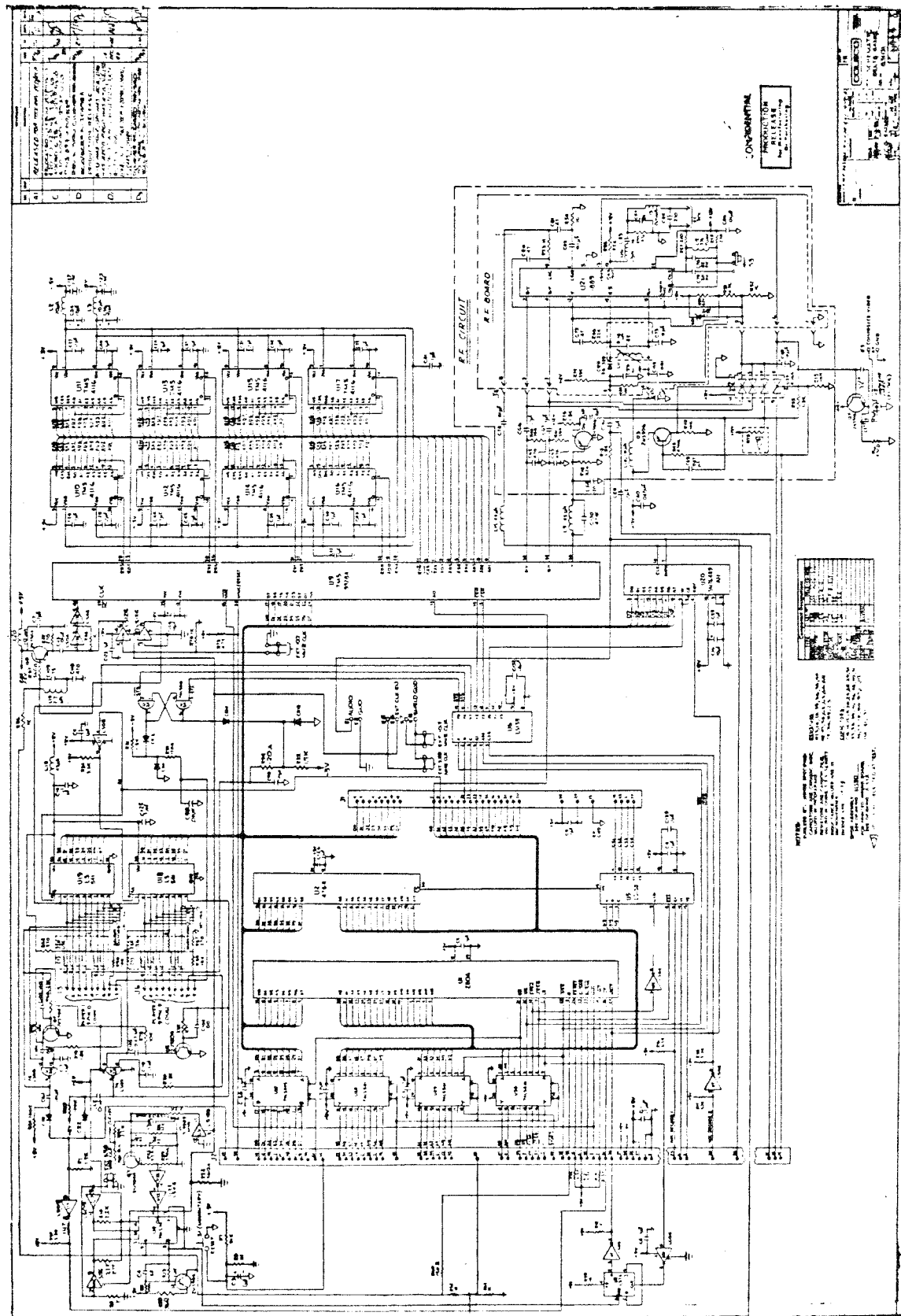
PRODUCTION
RELEASE
For Official Use Only



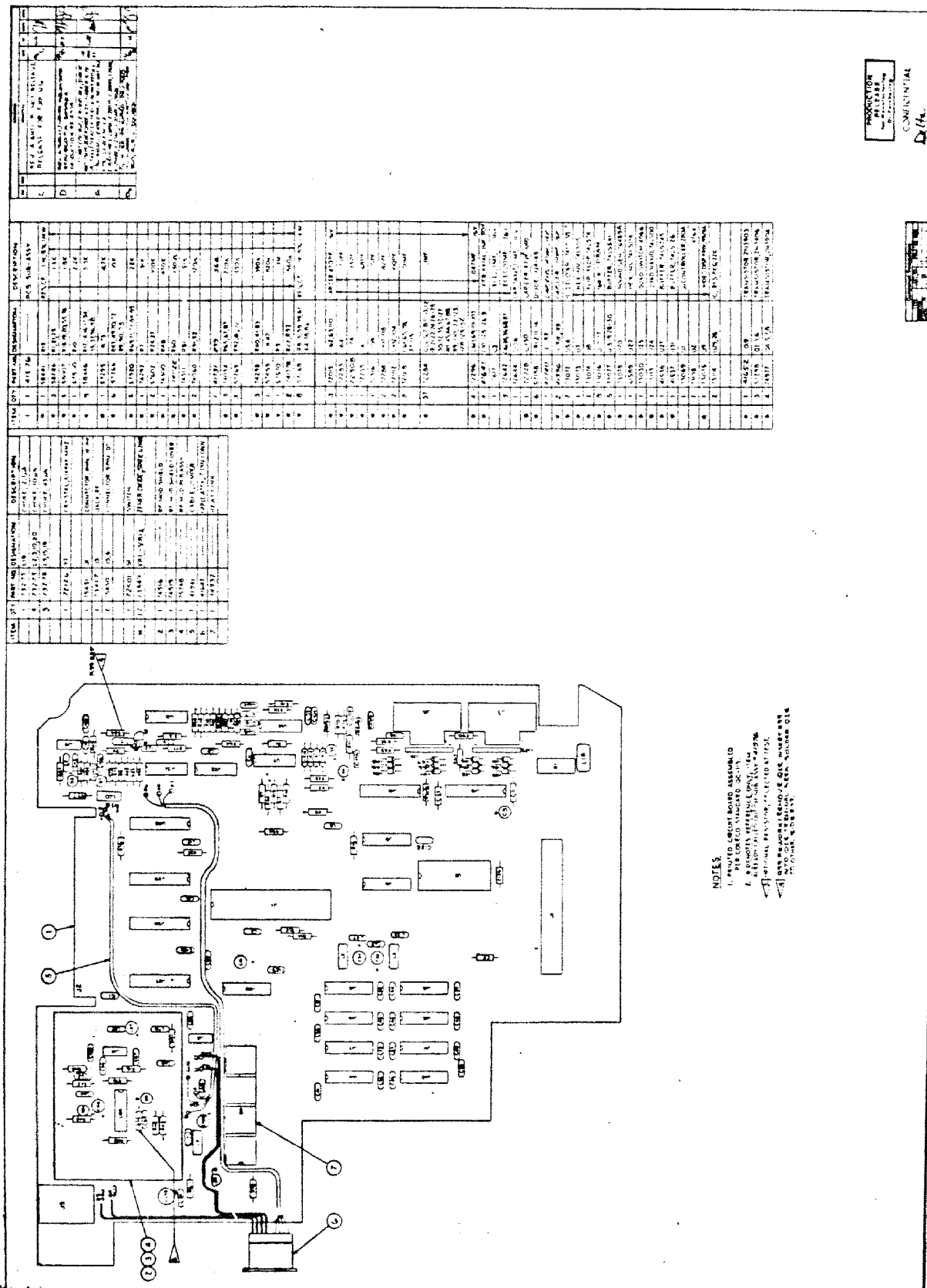
4.1 Memory and I/O Board Component Location/Identification Drawing



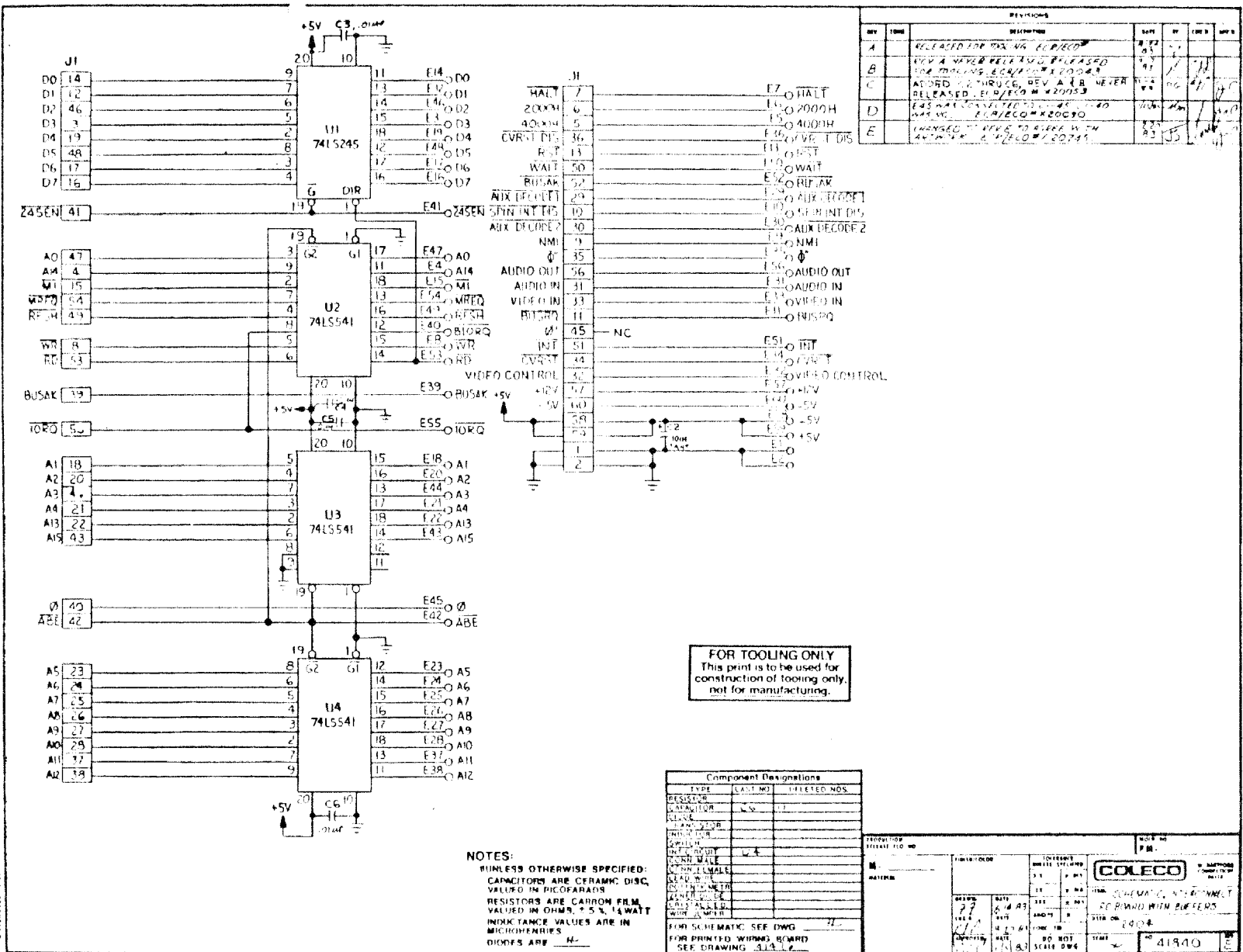
4.2 CPU Board Schematic



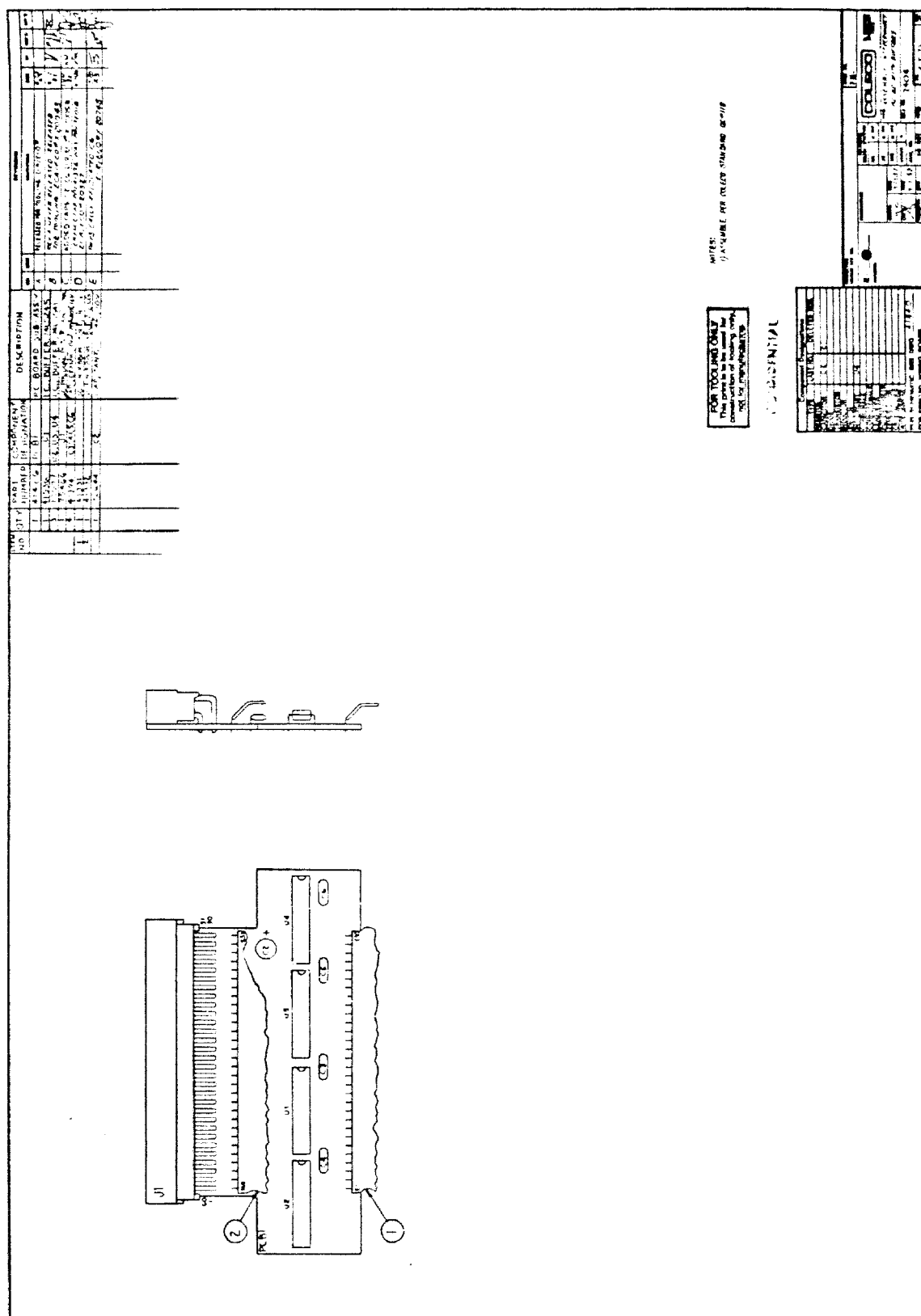
4.2 CPU Board Component Location/Identification Drawing



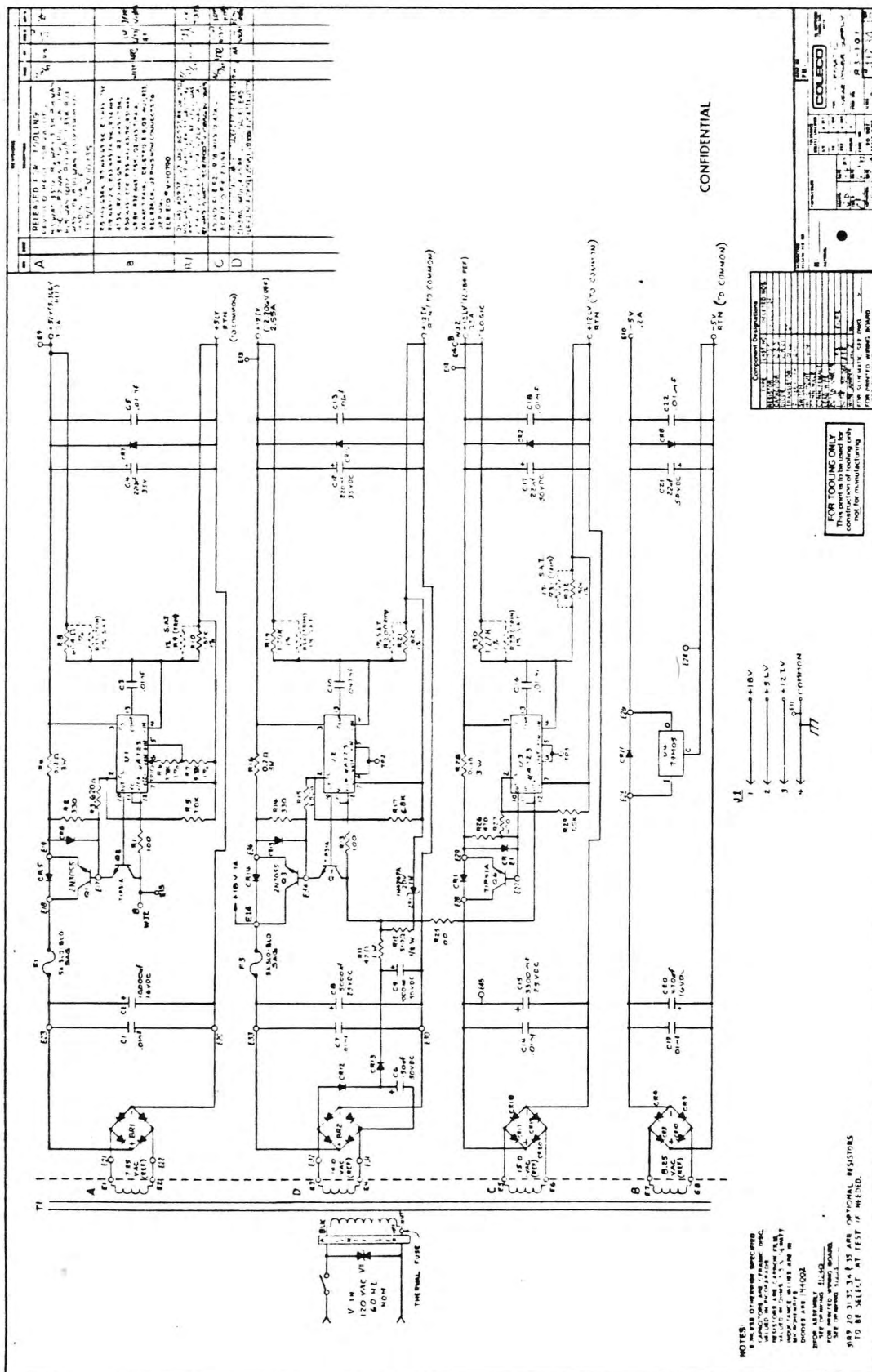
4.3 Interconnect Board Schematic



4.3 Interconnect Board Component Location/Identification Drawing



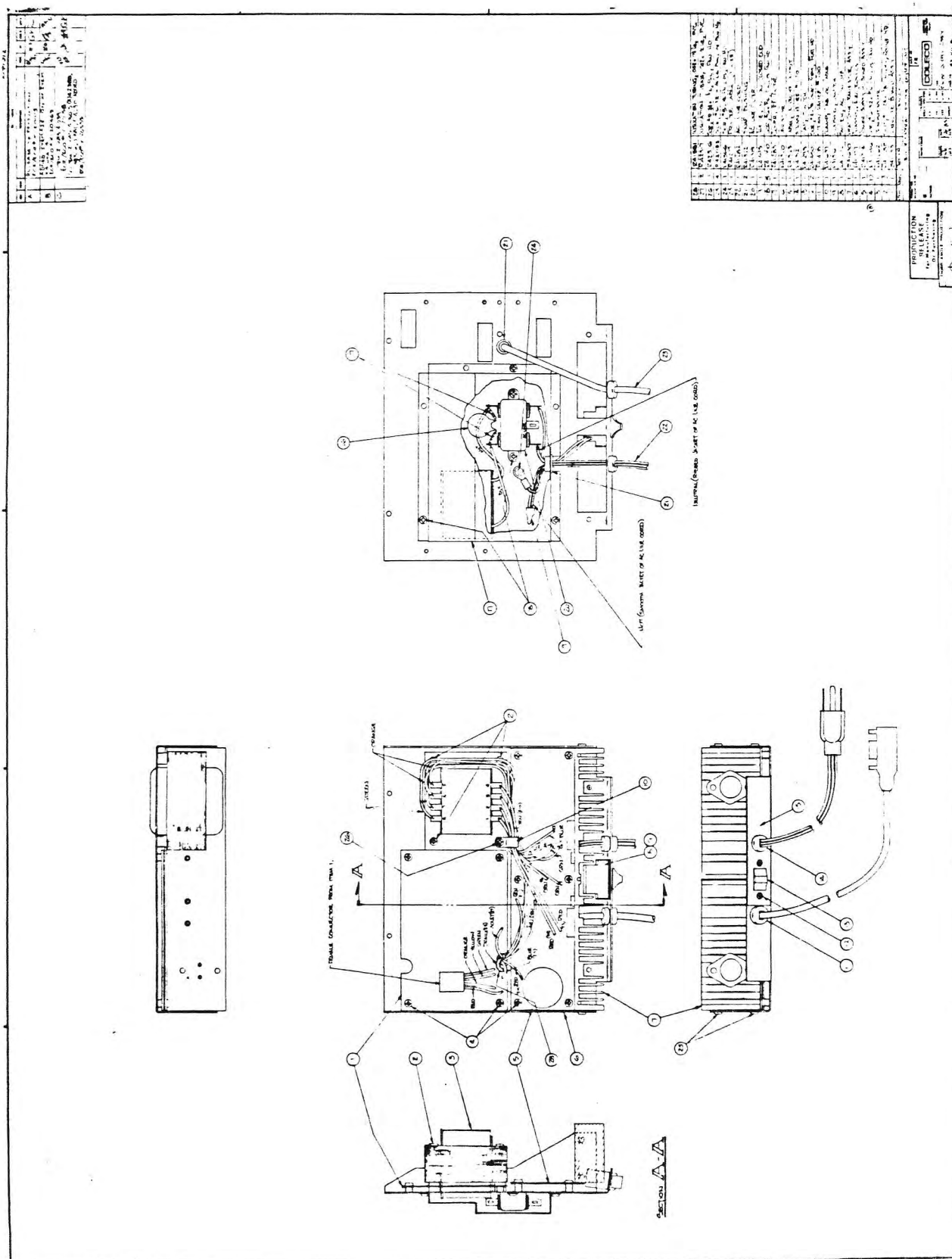
4.4 Linear Power Supply Schematic



CONFIDENTIAL

NOTES
1. VALUES GIVEN ARE APPROXIMATE
2. VALUES GIVEN ARE APPROXIMATE
3. VALUES GIVEN ARE APPROXIMATE
4. VALUES GIVEN ARE APPROXIMATE
5. VALUES GIVEN ARE APPROXIMATE
6. VALUES GIVEN ARE APPROXIMATE
7. VALUES GIVEN ARE APPROXIMATE
8. VALUES GIVEN ARE APPROXIMATE
9. VALUES GIVEN ARE APPROXIMATE
10. VALUES GIVEN ARE APPROXIMATE
11. VALUES GIVEN ARE APPROXIMATE
12. VALUES GIVEN ARE APPROXIMATE
13. VALUES GIVEN ARE APPROXIMATE
14. VALUES GIVEN ARE APPROXIMATE
15. VALUES GIVEN ARE APPROXIMATE
16. VALUES GIVEN ARE APPROXIMATE
17. VALUES GIVEN ARE APPROXIMATE
18. VALUES GIVEN ARE APPROXIMATE
19. VALUES GIVEN ARE APPROXIMATE
20. VALUES GIVEN ARE APPROXIMATE
21. VALUES GIVEN ARE APPROXIMATE
22. VALUES GIVEN ARE APPROXIMATE
23. VALUES GIVEN ARE APPROXIMATE
24. VALUES GIVEN ARE APPROXIMATE
25. VALUES GIVEN ARE APPROXIMATE
26. VALUES GIVEN ARE APPROXIMATE
27. VALUES GIVEN ARE APPROXIMATE
28. VALUES GIVEN ARE APPROXIMATE
29. VALUES GIVEN ARE APPROXIMATE
30. VALUES GIVEN ARE APPROXIMATE
31. VALUES GIVEN ARE APPROXIMATE
32. VALUES GIVEN ARE APPROXIMATE
33. VALUES GIVEN ARE APPROXIMATE
34. VALUES GIVEN ARE APPROXIMATE
35. VALUES GIVEN ARE APPROXIMATE
36. VALUES GIVEN ARE APPROXIMATE
37. VALUES GIVEN ARE APPROXIMATE
38. VALUES GIVEN ARE APPROXIMATE
39. VALUES GIVEN ARE APPROXIMATE
40. VALUES GIVEN ARE APPROXIMATE
41. VALUES GIVEN ARE APPROXIMATE
42. VALUES GIVEN ARE APPROXIMATE
43. VALUES GIVEN ARE APPROXIMATE
44. VALUES GIVEN ARE APPROXIMATE
45. VALUES GIVEN ARE APPROXIMATE
46. VALUES GIVEN ARE APPROXIMATE
47. VALUES GIVEN ARE APPROXIMATE
48. VALUES GIVEN ARE APPROXIMATE
49. VALUES GIVEN ARE APPROXIMATE
50. VALUES GIVEN ARE APPROXIMATE
51. VALUES GIVEN ARE APPROXIMATE
52. VALUES GIVEN ARE APPROXIMATE
53. VALUES GIVEN ARE APPROXIMATE
54. VALUES GIVEN ARE APPROXIMATE
55. VALUES GIVEN ARE APPROXIMATE
56. VALUES GIVEN ARE APPROXIMATE
57. VALUES GIVEN ARE APPROXIMATE
58. VALUES GIVEN ARE APPROXIMATE
59. VALUES GIVEN ARE APPROXIMATE
60. VALUES GIVEN ARE APPROXIMATE
61. VALUES GIVEN ARE APPROXIMATE
62. VALUES GIVEN ARE APPROXIMATE
63. VALUES GIVEN ARE APPROXIMATE
64. VALUES GIVEN ARE APPROXIMATE
65. VALUES GIVEN ARE APPROXIMATE
66. VALUES GIVEN ARE APPROXIMATE
67. VALUES GIVEN ARE APPROXIMATE
68. VALUES GIVEN ARE APPROXIMATE
69. VALUES GIVEN ARE APPROXIMATE
70. VALUES GIVEN ARE APPROXIMATE
71. VALUES GIVEN ARE APPROXIMATE
72. VALUES GIVEN ARE APPROXIMATE
73. VALUES GIVEN ARE APPROXIMATE
74. VALUES GIVEN ARE APPROXIMATE
75. VALUES GIVEN ARE APPROXIMATE
76. VALUES GIVEN ARE APPROXIMATE
77. VALUES GIVEN ARE APPROXIMATE
78. VALUES GIVEN ARE APPROXIMATE
79. VALUES GIVEN ARE APPROXIMATE
80. VALUES GIVEN ARE APPROXIMATE
81. VALUES GIVEN ARE APPROXIMATE
82. VALUES GIVEN ARE APPROXIMATE
83. VALUES GIVEN ARE APPROXIMATE
84. VALUES GIVEN ARE APPROXIMATE
85. VALUES GIVEN ARE APPROXIMATE
86. VALUES GIVEN ARE APPROXIMATE
87. VALUES GIVEN ARE APPROXIMATE
88. VALUES GIVEN ARE APPROXIMATE
89. VALUES GIVEN ARE APPROXIMATE
90. VALUES GIVEN ARE APPROXIMATE
91. VALUES GIVEN ARE APPROXIMATE
92. VALUES GIVEN ARE APPROXIMATE
93. VALUES GIVEN ARE APPROXIMATE
94. VALUES GIVEN ARE APPROXIMATE
95. VALUES GIVEN ARE APPROXIMATE
96. VALUES GIVEN ARE APPROXIMATE
97. VALUES GIVEN ARE APPROXIMATE
98. VALUES GIVEN ARE APPROXIMATE
99. VALUES GIVEN ARE APPROXIMATE
100. VALUES GIVEN ARE APPROXIMATE

4.4 Linear Power Supply Component Location/Identification Drawing



4.4 Linear Power Supply Sub-Assembly

