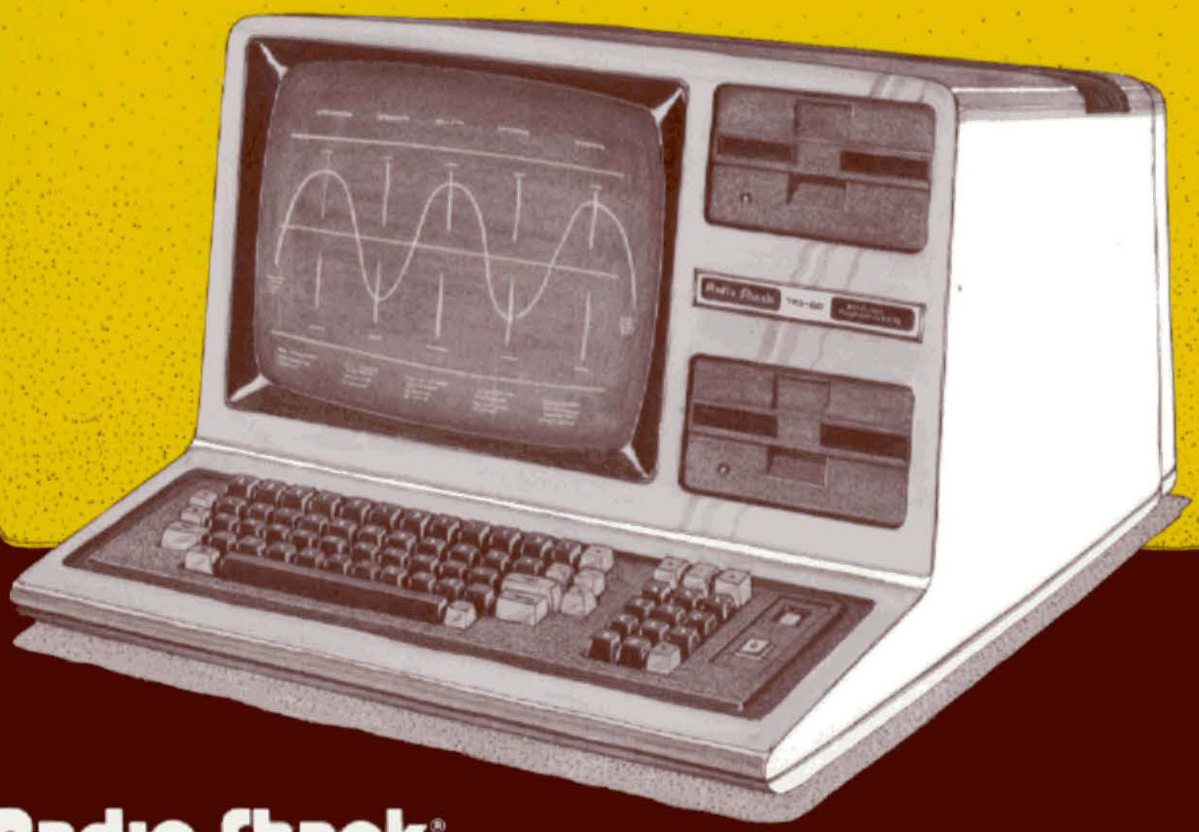
CAT. NO. 26-2110

TRS-80® Model 4

Technical Reference Manual



Radio Shack®

The biggest name in little computers®

CUSTOM MANUFACTURED IN THE USA BY RADIO SHACK, A DIVISION OF TANDY CORPORATION

TERMS AND CONDITIONS OF SALE AND LICENSE OF RADIO SHACK COMPUTER EQUIPMENT AND SOFTWARE PURCHASED FROM A RADIO SHACK COMPANY-OWNED COMPUTER CENTER, RETAIL STORE OR FROM A RADIO SHACK FRANCHISEE OR DEALER AT ITS AUTHORIZED LOCATION

LIMITED WARRANTY

I. CUSTOMER OBLIGATIONS

A CUSTOMER assumes full responsibility that this Radio Shack computer hardware purchased (the "Equipment"), and any copies of Radio Shack software included with the Equipment or licensed separately (the "Software") meets the specifications, capacity, capabilities, versatility, and other requirements of CUSTOMER

B CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation

II. RADIO SHACK LIMITED WARRANTIES AND CONDITIONS OF SALE

A For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment, RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects THIS WARRANTY IS ONLY APPLICABLE TO PURCHASES OF RADIO SHACK EQUIPMENT BY THE ORIGINAL CUSTOMER FROM RADIO SHACK COMPANY-OWNED COMPUTER CENTERS, RETAIL STORES AND FROM RADIO SHACK FRANCHISEES AND DEALERS AT ITS AUTHORIZED LOCATION The warranty is void if the Equipment's case or cabinet has been opened, or if the Equipment or Software has been subjected to improper or abnormal use If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer for repair, along with a copy of the sales document or lease agreement The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair, replacement, or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items.

B RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software, except as provided in this paragraph Software is licensed on an "AS IS" basis, without warranty The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software The defective Software shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer along with the sales document

C Except as provided herein no employee, agent, franchisee, dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK

D Except as provided herein, **RADIO SHACK MAKES NO WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.**

E Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER

III. LIMITATION OF LIABILITY

A EXCEPT AS PROVIDED HEREIN, RADIO SHACK SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "EQUIPMENT" OR "SOFTWARE" SOLD, LEASED, LICENSED OR FURNISHED BY RADIO SHACK, INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THE "EQUIPMENT" OR "SOFTWARE". IN NO EVENT SHALL RADIO SHACK BE LIABLE FOR LOSS OF PROFITS, OR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR IN ANY MANNER ARISING OUT OF OR CONNECTED WITH THE SALE, LEASE, LICENSE, USE OR ANTICIPATED USE OF THE "EQUIPMENT" OR "SOFTWARE"

NOTWITHSTANDING THE ABOVE LIMITATIONS AND WARRANTIES, RADIO SHACK'S LIABILITY HEREUNDER FOR DAMAGES INCURRED BY CUSTOMER OR OTHERS SHALL NOT EXCEED THE AMOUNT PAID BY CUSTOMER FOR THE PARTICULAR "EQUIPMENT" OR "SOFTWARE" INVOLVED.

B RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software

C No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software, whichever first occurs

D Some states do not allow the limitation or exclusion of incidental or consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER

IV. RADIO SHACK SOFTWARE LICENSE

RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the RADIO SHACK Software on **one** computer, subject to the following provisions:

A Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software

B Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER, but not title to the Software

C CUSTOMER may use Software on one host computer and access that Software through one or more terminals if the Software permits this function.

D CUSTOMER shall not use, make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in this Software License Customer is expressly prohibited from disassembling the Software

E CUSTOMER is permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation of **one** computer with the Software, but only to the extent the Software allows a backup copy to be made However, for TRSDOS Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use

F CUSTOMER may resell or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER

G All copyright notices shall be retained on all copies of the Software

V. APPLICABILITY OF WARRANTY

A The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby RADIO SHACK sells or conveys such Equipment to a third party for lease to CUSTOMER

B The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK, the author, owner and/or licensor of the Software and any manufacturer of the Equipment sold by RADIO SHACK

VI. STATE LAW RIGHTS

The warranties granted herein give the **original** CUSTOMER specific legal rights, and the **original** CUSTOMER may have other rights which vary from state to state

TRS-80® MODEL 4 TECHNICAL REFERENCE MANUAL

Radio Shack®
A DIVISION OF TANDY CORPORATION
FORT WORTH, TEXAS 76102

TRSDOS® Version 6 Operating System:
© 1983 Logical Systems.
Licensed to Tandy Corporation.
All Rights Reserved.

Model 4 Technical Reference Manual; Hardware Part:
© 1983 Tandy Corporation.
All Rights Reserved.

Model 4 Technical Reference Manual; Software Part:
© 1983 Tandy Corporation and Logical Systems.
All Rights Reserved.
Revised to include TRSDOS 6.01 enhancements.

Reproduction or use, without express written permission from Tandy Corporation, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information contained herein.

TRSDOS is a registered trademark of Tandy Corporation.

10 9 8 7 6 5 4 3 2

IMPORTANT NOTICE

This Technical Reference Manual is written for owners of the TRS-80 Model 4 Computer, who have a thorough understanding of electronics and computer circuitry. It is not written to the beginner's level of comprehension.

This manual contains detailed schematics and theories of operation for each major part of the Model 4. These tools will aid you with designing interfaces for your computer, repairing your own computer after its warranty has expired, or simply obtaining practical knowledge of your TRS-80 Model 4 operation.

Radio Shack will not be liable for any damage caused, or alleged to be caused, by the customer or any other person using this technical manual to repair, modify, or alter the TRS-80 Model 4 Computer in any manner.

Many parts of the computer electronics are very sensitive and can be easily damaged by improper servicing. We strongly suggest that for proper servicing, the computer be returned to Radio Shack.

Because of the sensitivity of computer equipment and the potential problems which can result from improper servicing, the following limitations apply to services offered by Radio Shack:

1. If any of the warranty seals on any Radio Shack computer product are broken, Radio Shack reserves the right to refuse to service the equipment or to void any remaining warranty on the equipment.

2. If any Radio Shack computer product has been modified so that it is not within manufacturer's specifications, including, but not limited to, the installation of any non-Radio Shack parts, components, or replacement boards, then Radio Shack reserves the right to refuse to service the equipment, void any remaining warranty, remove and replace any non-Radio Shack part found in the equipment, and perform whatever modifications are necessary to return the equipment to original factory specifications.

3. The cost for the labor and parts required to return the Radio Shack computer equipment to its original specifications will be charged to the customer in addition to the normal repair charges.

TABLE OF CONTENTS

Part 1 / Hardware

SECTION I Introduction 1
1.1 System Overview 3
1.2 Block Diagram 4
1.3 Jumper Options 5

SECTION II Disassembly/Assembly 7
2.1 Disassembly 9
2.1.1 Case 9
2.1.2 CPU Board 9
2.1.3 FDC Board (Optional) 9
2.1.4 RS-232 Board (Optional) 9
2.1.5 Main Power Supplies 9
2.1.6 Disk Drives (Optional) 9
2.1.7 Video Monitor (CRT) and Video Board 10

2.2 Assembly 11
2.2.1 RS-232 Board (Optional) 11
2.2.2 FDC Board (Optional) 11
2.2.3 CPU Board 11
2.2.4 Power Supply 11
2.2.5 Disk Drive (Optional) 11
2.2.6 Disk Drive Power Supply (Optional) 11
2.2.7 Video Monitor (CRT) and Video Board 11
2.2.8 Case 11

SECTION III CPU Circuit Board 13
3.1 Model 4 Theory Of Operation 15
3.1.1 Introduction 15
3.1.2 CPU and Timing 15
3.1.3 Buffering 15
3.1.4 Address Decoding 15
3.1.5 ROM 19
3.1.6 RAM 19
3.1.7 Keyboard 19
3.1.8 Video 19
3.1.9 Real Time Clock 21
3.1.10 Cassette Circuitry 21
3.1.11 Printer Circuitry 21
3.1.12 I/O Connectors 21
3.1.13 Sound Option 22

3.2 Model 4 I/O BUS 26

3.3 Port Bits 28

Schematic 8000173, 64K Version CPU PCB 8858090 31
CPU Component Layout 43
Circuit Trace, CPU Component Side 45
Circuit Trace, CPU Solder Side 47
Parts List CPU PCB 6700104AA3 49
Parts List CPU PCB 8858090 53

SECTION IV Floppy Disk Interface 57
4.1 Model 4 FDC PCB #8858Ø6Ø 59
4.1.1 Control and Data Buffering 59
4.1.2 Nonmaskable Interrupt Logic 59
4.1.3 Drive Select Latch and Motor ON Logic 59
4.1.4 Wait State Generation and WAITIMOUT Logic 60
4.1.5 Clock Generation Logic 60
4.1.6 Disk Bus Selector Logic 60
4.1.7 Read/Write Data Pulse Shaping Logic 60
4.1.8 Disk Bus Output Drivers 60
4.1.9 Write Precompensation and Clock Recovery Logic 60
4.1.10 Floppy Disk Controller Chip 61
4.1.11 Adjustments and Jumper Options 61
FDC Board to CPU Board Signal Description 62
Schematic 8ØØØØ95, FDC Controller 8858Ø6Ø 63
Component Location/Circuit Trace, Component Side 65
Circuit Trace, FDC PCB, Solder Side 66
Parts List, FDC PCB #8858Ø6Ø 67

4.2 Model 4 FDC PCB #8858163Ø 69
4.2.1 Control and Data Buffering 69
4.2.2 Nonmaskable Interrupt Logic 69
4.2.3 Drive Select Latch and Motor ON Logic 69
4.2.4 Wait State Generation and WAITIMOUT Logic 69
4.2.5 Clock Generation Logic 70
4.2.6 Disk Bus Selector Logic 70
4.2.7 Disk Bus Output Drivers 70
4.2.8 Write Precompensation and Write Data Pulse Shaping Logic 70
4.2.9 Clock and Read Data Recovery Logic 70
4.2.10 Floppy Disk Controller Chip 72
Schematic 8ØØØ168, FDC PCB #8858163Ø 73
Component Layout, FDC PCB #8858163Ø 75
Circuit Trace, FDC PCB #8858163Ø, Circuit Side 76
Circuit Trace, FDC PCB #8858163Ø, Solder Side 77
Parts List, FDC PCB #885163Ø 78

SECTION V Mini-Disk Drive 79
5.1 Mini-Disk Drive #879Ø112 (TPI) 81
5.1.1 Physical Description 81
5.1.2 Functional Description 81
5.1.3 Interface Connections 81
5.1.4 Physical Checkout 81
5.1.5 Mounting The Disk Drive 83
5.1.6 Flat Ribbon Cable Assembly 83
5.1.7 Resistor Termination 83

5.2 Theory Of Operation
5.2.1 Introduction 84
5.2.2 Organization Of The Disk Drive 84
5.2.3 Functional Block Diagram Description 84
5.2.4 INDEX 84

5.2.5 Write Protect 84
5.2.6 Track ØØ Switch 84
5.2.7 Spindle Drive 84
5.2.8 Positioner Control 84
5.2.9 Data Electronics 85
5.2.10 Data Recording 86
5.2.11 Data Reproduction 86

5.3 Operation 88
5.3.1 Introduction 88
5.3.2 Interface Electronics Specifications 88
5.3.3 Input Control Lines 88
5.3.4 Select Lines (DS1*-DS4*) 88
5.3.5 Drive Motor Enable 88
5.3.6 Direction and Step (DIR*) (STEP*) 88
5.3.7 Compensated Write Data (CWD) 88
5.3.8 Write Gate (WG*) 88
5.3.9 Output Status 90
5.3.10 Index Pulse (IP*) 90
5.3.11 Track ØØ (TRKØ*) 90
5.3.12 Write Protect (WPRT*) 90
5.3.13 Read Data (RD*) 90

5.4 Maintenance 91
5.4.1 Physical Description of the PC Board 91
5.4.2 Circuit Board Test Points 91
5.4.3 Option Select 91
5.4.4 Preventive Maintenance 93
5.4.5 Cleaning the Head 93

5.5 Alignment And Adjustment 93
5.5.1 Drive Motor Maintenance 93
5.5.2 Carriage Movement Check 93
5.5.3 Head Radial Alignment/CE Alignment 94
5.5.4 Track ØØ Alignment 95
5.5.5 Write Protect Switch 95
5.5.6 Index Sector Adjustment 95
5.5.7 Head Amplitude/Compliance Check 96
5.5.8 Final Check 96
Schematic Diagram, Mini Disk Drive #8790112 97
Component Location, Logic/Servo PCB #995050001 99
Parts List, Logic/Servo PCB #995050001 100
Exploded View, Mini Disk Drive #8790112 104
Parts List, Mini Disk Drive #8790112 105

SECTION VI Power Supplies 107
6.1 Power Supply #8790021, 38W (ASTEC #AA11320) 109
6.1.1 General 109
6.1.2 Theory of Operation 109
Schematic Diagram, Power Supply #8790021 110
Parts List, Power Supply #8790021 111

6.2 Power Supply #8790043, 65W (ASTEC #AA12090) . . . 112
6.2.1 Test Set-Up . . . 112
6.2.2 Visual Inspection . . . 112
6.2.3 Start-Up . . . 112
6.2.4 No Output . . . 112
6.2.5 Low Outputs . . . 113
6.2.6 Crowbar . . . 113
6.2.7 Performance Test . . . 114
Schematic Diagram, Power Supply #8790043 . . . 115
Component Layout #1700218, Power Supply #8790043 . . . 116
Circuit Trace, Power Supply #8790043, Solder Side . . . 117
Parts List, Power Supply #8790043 . . . 118

6.3 Power Supply #8790049, 65W (Tandy) . . . 121
6.3.1 System Description . . . 121
6.3.2 Technical Specifications . . . 124
6.3.3 Theory of Operation . . . 125
6.3.4 Troubleshooting Chart . . . 133
6.3.5 Testing and Adjustments . . . 134
Schematic #8000164, Power Supply #8790049 . . . 135
Component Layout, Power Supply #8790049 . . . 137
Circuit Trace, Power Supply #8790049, Solder Side . . . 138
Parts List, Power Supply #8790049 . . . 139

SECTION VII Video Monitor . . . 143
7.1 Video Monitor #8492002 (Manufacturer-RCA) . . . 145
7.1.1 Functional Specifications . . . 145
7.1.2 Service Adjustments . . . 146
Schematic #8000187, RCA Video Monitor . . . 147
Component Location/Circuit Trace . . . 149
Parts List, Video Monitor #8492002 . . . 150

7.2 Video Monitor #8790607 (Manufacturer-TCE) . . . 155
Schematic #8000186, TCE Video Monitor . . . 155
Parts List, Video Monitor #8790607 . . . 157

SECTION VIII Exploded View/Parts List . . . 161
Exploded View, Model 4 Computer, Catalog #26-1067/8/9 . . . 163
Parts List, Model 4 Computer, Catalog #26-1067/8/9 . . . 165

SECTION IX RS-232C Circuit Board . . . 167
9.1 RS-232C Technical Description . . . 169
9.2 Pinout Listing . . . 170
9.3 Port And Bit Assignments . . . 171
RS232 Board to CPU Board Signal Description . . . 173
Schematic #8000076, RS232C Printed Circuit Board . . . 175
Component Location/Circuit Trace, RS232C PC Board, Component Side . . . 177
Circuit Trace, RS232C PC Board, Solder Side . . . 178
Parts List, RS232C PC Board . . . 179

LIST OF FIGURES

Fig. No.	Description	Page No.
1-1	TRS-8Ø Model 4 Interconnection Diagram	4
3-1	Model 4 Block Diagram	16
3-2	RAM Memory	20
3-3	Timing of U3 and CPU	23
3-4	Timing of U4	24
3-5	CPU Video Access Timing	25
3-6	I/O Bus Timing Diagram	27
4-1	Write Precompensation Timing	71
5-1	Cable Assembly—Connector Pin Removal Chart	83
5-2	Resistor Termination	83
5-3	Disk Drive Functional Block Diagram	85
5-4	FM Recording	85
5-5	Write Timing Diagram	87
5-6	Read Timing Diagram	87
5-7	Interface Configuration	88
5-8	Logic/Servo PC Board Test Points/Connector Locations	92
5-9	"Cat Eyes" Pattern	94
5-1Ø	Radial Adjustment	94
5-11	Track ØØ Switch Adjustment	95
5-12	Index Sector Timing	95
5-13	Index Adjustment	95
5-14	Upper Arm and Carriage	96
6-1	Test Set-Up	112
6-2	Q2 Collector Waveform	113
6-3	Q2 Base Waveform	113
6-4	Basic Flyback Converter	121
6-5	Waveforms for Figure 6-4	122
6-6	Block Diagram, Power Supply 8790Ø49	123
6-7	Input AC Supply	125
6-8	Kick-Start Latch	126
6-9	Oscillator, Pulse Generator Waveforms	127
6-1Ø	Control Section	128
6-11	Base Drive Circuit	128
6-12	Q7 Base Voltage Waveform	129
6-13	Primary Side Protection	129
6-14	Feedback Signal Development	131
6-15	Overvoltage Crowbar	132
6-16	Power Chain	132
6-17	Control Chain Simplified Schematic	133
7-1	Deflection Yoke Assembly	146

Part 2 / Software

1/ Disk Organization . . . 183
- Single Density Floppy Diskette . . . 183
- Double Density Floppy Diskette . . . 183
- 5" 5-Meg Hard Disk . . . 184
- Disk Space Available to the User . . . 184
- Unit of Allocation . . . 184

2/ Disk Files . . . 185
- Methods of File Allocation . . . 185
  - Dynamic Allocation . . . 185
  - Pre-Allocation . . . 185
- Record Length . . . 185
- Record Processing Capabilities . . . 186
  - Record Numbers . . . 186

3/ TRSDOS File Descriptions . . . 187
- System Files (/SYS) . . . 187
- Utility Programs . . . 189
- Device Driver Programs . . . 189
- Filter Programs . . . 189
- Creating a Minimum Configuration Disk . . . 189

4/ Device Access . . . 191
- Device Control Block (DCB) . . . 191
- Memory Header . . . 192

5/ Drive Access . . . 193
- Drive Code Table (DCT) . . . 193
- Disk I/O Table . . . 195
- Directory Records . . . 195
- Granule Allocation Table (GAT) . . . 198
- Hash Index Table (HIT) . . . 200

6/ File Control . . . 205
- File Control Block (FCB) . . . 205

7/ TRSDOS Version 6 Programming Guidelines . . . 209
- Converting to TRSDOS Version 6 . . . 209
- Programming With Restart Vectors . . . 211
- KFLAG$ (BREAK)((PAUSE), and (ENTER) Interfacing . . . 211
- Interfacing to @ICNFG . . . 214
- Interfacing to @KITSK . . . 215
- Interfacing to the Task Processor . . . 216
- Interfacing RAM Banks 1 and 2 . . . 218
- Device Driver and FIlter Templates . . . 222
- @CTL Interfacing to Device Drivers . . . 224

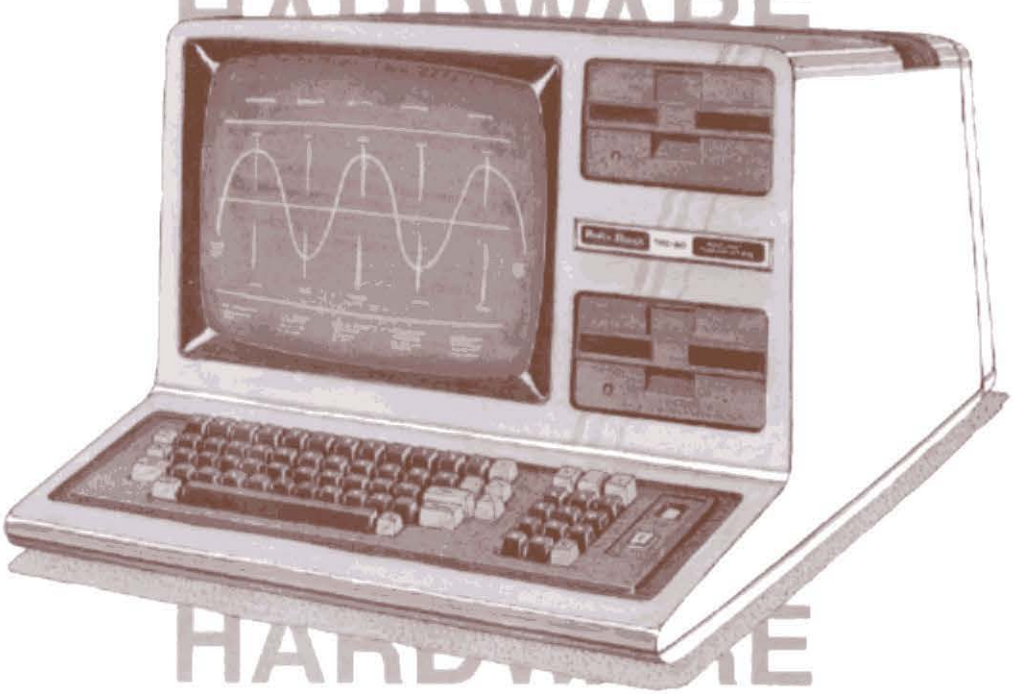
8/ Using the Supervisor Calls 227
Calling Procedure 227
Program Entry and Return Conditions. 227
Supervisor Calls. 228
Numerical List of SVCs 331
Alphabetical List of SVCs 334
Sample Programs. 336

9/ Technical Information on TRSDOS Commands and Utilities 361

Appendix A/ TRSDOS Error Messages. 365
Appendix B/ Memory Map 371
Appendix C/ Character Codes 373
Appendix D/ Keyboard Code Map 383

Index 385

HARDWARE



HARDWARE

PART I
HARDWARE

SECTION I

INTRODUCTION

INTRODUCTION

1.1 SYSTEM OVERVIEW

The Radio Shack TRS-80 Model 4 Microcomputer is an enhanced version of Radio Shack's popular TRS-80 Model III Microcomputer. The TRS-80 Model 4 is software compatible with the Model III so that owners of either system can take advantage of the large number of programs available.

Features of the TRS-80 Model 4 which are common to the TRS-80 Model III include:

- Availability of Level I or Level II BASIC in ROM
- Full size typewriter style keyboard
- A 12-inch video display
- Built-in cassette interface
- Character display of 16 lines of 64 characters
- Graphics under control of BASIC (128 H x 48 V)
- UL recognized construction
- 12-key numeric keypad for rapid entry of numbers
- Rugged cabinet housing keyboard, electronics, video display, and power supply
- Direct drive video monitor for improved resolution
- Internal power supply
- Parallel printer port for use with Radio Shack printers

Other features available when Level II BASIC is used are: real time clock, upper and lower case characters, RAM internally expandable to 128K bytes, I/O port for peripheral expansion, and cassette interface available with 500 and 1500 baud rates.

Optional peripherals for the TRS-80 Model 4 include disk drives (two built-in, two external) with double density for increased storage capacity, and a built-in RS-232 serial interface for communications and peripheral interface.

1.2 BLOCK DIAGRAM

The Block Diagram (Figure 1-1) shows the various internal components and connections of the Model 4 Microcomputer.

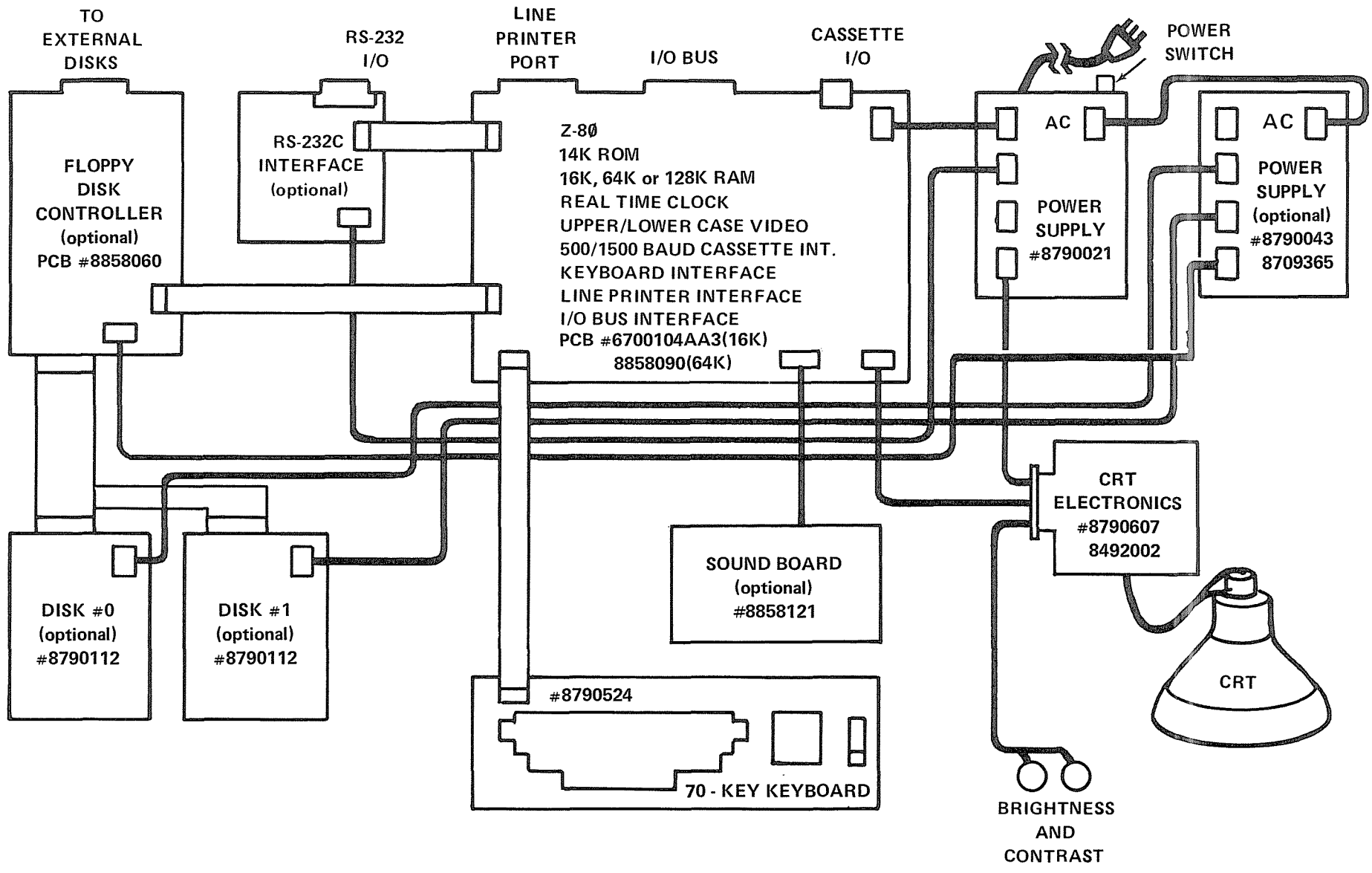


FIGURE 1-1. TRS-80 MODEL 4 INTERCONNECTION DIAGRAM

1.3 JUMPER OPTIONS

1.3.1 16K to 64K

1. Remove U77 - U84

2. Move E5-E6 to E5-E4
   Move E1-E2 to E2-E3
   Move E12-E13 to E12-E11

3. Add E7-E8 and eight Tandy #8040665 ICs in locations U77-U84.

4. Remove capacitors C68, C72, C76, C80, C84, C88, C92 and C96.

1.3.2 64K to 128K

1. Remove shunt at U72 and add IC Tandy #8075468 in its place.

2. Add eight Tandy #8040665 ICs in locations U85-U92.

1.3.3 Graphics Board

1. Remove E14-E15 (and screws near U65 and U71).

2. Plug in Graphics Board

3. Replace mounting screws at U65 and U71.

1.3.4 E9-E10 are not used at this time

SECTION II

DISASSEMBLY/ASSEMBLY

DISASSEMBLY/ASSEMBLY

2.1 DISASSEMBLY

2.1.1 Case

1. Remove all cables from the bottom and rear of the Computer. Position the Computer on its rear panel to provide easy access to the case bottom. Remove the ten screws from the case bottom. Notice the different types and lengths of screws and note their positions. Set them aside in groups.

2. Position the Computer upright and remove the #6 screw and washer from the top of the back panel of the case.

3. Very carefully remove case top, lifting **straight up** and setting it aside to the left (if facing video screen). Be careful not to exceed the length of the video cable.

4. Remove screws from the chassis shield and the ground connectors and remove the shield. **

2.1.2 CPU Board

1. Remove the six screws which attach the RFI Rear Shield to the metal CPU mounting bracket. This bracket is also held in place with tape at the lower part of the bracket. Carefully peel the tape away from the shield so that it may be reused.

2. Remove all cables connecting the CPU Board (power supply cable, video, keyboard, and cassette cables, and if applicable, the RS-232, sound board, and FDC interconnect cables).

3. Remove the eight screws fastening the CPU PCB (three at the top and bottom and one on either side of the board in the middle.

4. Make sure all cables to the CPU Board have been disconnected then remove the Board. (If your unit uses plastic spacer mounts to hold the Board, press the small tabs on the mounts through the mounting holes in the PCB and gently pull the CPU Board off.)

2.1.3 FDC Board (optional)

NOTE: The CPU Board must be removed before removing the FDC Board.

1. Be sure to disconnect all cables to the FDC Board.

2. Remove the screws holding the FDC Board to the metal chassis and remove the Board.

**Not applicable to all units.

2.1.4 RS-232 Board (optional)

NOTE: The CPU Board must be removed before removing the RS-232 Board.

1. Be sure to remove all cables connecting the RS-232 Board.

2. Remove the screws connecting the PC Board to the chassis and remove the Board.

2.1.5 Main Power Supplies

Three different power supplies are used on the Model 4 Computer, depending on the initial configuration of the unit. If the unit is supplied with 16K RAM and cassette input, Power Supply Assembly #8790021 is supplied to provide DC voltages required to power the computer. It is mounted on the front side of the main CPU mounting bracket at the rear of the unit. If disk drive(s) are added to this computer, a second power supply is mounted to the LH disk drive mounting bracket.

If the computer is supplied with 64K RAM and a single or dual disk drive, then power supply #8790043 or #8790049 is supplied to power the unit. This power supply is mounted at the left side of the LH disk drive mounting bracket. It is attached to the CPU mounting bracket by screws through the heat sink bracket of the power supply.

1. Remove all interconnecting cables to the board(s). These include the power supply cable, video, keyboard, and cassette cable. If applicable, also remove the RS-232 and FDC power connectors and the disk ribbon cable.

2. Note the position of the ground tab and remove it.

3. Remove the screws which attach the power supply to its mounting support (four for the 38W power supply and two for the 65W power supplies). Spacers separate the 38W power supply from its mounting bracket.
   The 65W power supply is provided with insulated standoffs to prevent shorting the power supply against the metal disk drive shield bracket.

4. Carefully lift the power supply out of the computer. If the unit is one which has been upgraded to include a disk drive and is equipped with two power supplies, it may be necessary to remove the Main CPU mounting bracket to provide access to the mounting screws of one of the power supplies.

2.1.6 Disk Drives (optional)

1. To remove the Disk Drive in the top position, carefully remove the FDC inter-connect cable connected to the rear of the Drive.

2. Remove the four screws and washers (two on each side) which connect the Drive to the Disk Mounting Bracket. Remove the RFI shield which covers the top drive.

3. Disconnect the power supply connector from the bottom of the top board in the Disk Drive and also remove the ground wire from the rear of the Drive.

4. To remove the Disk Drive in the bottom position, you must first remove the Power Supply attached to the LH Disk Mounting bracket to gain access to the Disk Drive mounting screws.

5. After removing the Power Supply, remove the FDC inter-connect cable from the rear of the Drive.

6. Remove the four screws and washers (two on each side) which connect the Drive to the Disk Mounting Bracket.

7. Disconnect the power supply connector from the bottom of the top board in the Disk Drive and also remove the ground wire from the rear of the Drive.

2.1.7 Video Monitor (CRT) and Video Board

1. Disconnect the four color coded wires with spade lugs from the CRT yoke. (Be sure to note their positions.)

2. Disconnect the connector on the rear of the CRT neck.

WARNING

There may be a high voltage charge on the high voltage anode. To discharge, connect one end of a wire to a known good ground and connect the other end of the wire to the blade of a common screwdriver. Insert the screwdriver blade under the suction cup and touch it to the clip holding the wire to the CRT.

3. Insert a common screwdriver under the suction cup on the high-voltage anode wire on the side of the CRT. Use the screwdriver to compress the clip holding the wire to the tube and pull the wire free.

4. Remove the ground wire fastened directly to the Video Board.

5. Remove the upper right and lower left nuts and washers which hold the CRT in place.

CAUTION

If dropped, the CRT may implode. To avoid this kind of accident, support the CRT while performing the next step.

6. Remove the remaining lower right and upper left nuts, and washers and carefully remove the CRT.

7. Disconnect the CPU cable connector from the Video Board.

8. Remove the two screws fastening the Video Board to the Case Top and carefully lift out Board.

2.2 ASSEMBLY

2.2.1 RS-232 Board (optional)

1. Install the PC Board using #6 x 1/4" screws. If applicable, press the PC Board onto the plastic spacer mounts then fasten with the screws.

2. Reconnect all cables to the RS-232 Board.

2.2.2 FDC Board (optional)

1. Install the PC Board using #6 x 1/4" screws and the plastic spacer mounts, if used.

2. Reconnect all cables to the FDC Board.

2.2.3 CPU Board

1. Make sure good insulating washers are attached to the CPU Board then fasten the Board using #6 x 1/2" screws.
2. Reconnect all cables to the CPU Board (power supply cable, video, keyboard, and cassette cables, and if applicable, RS-232, sound board, and FDC inter-connect cables).
3. Attach the small PCB Mount Bracket (if used) to the metal chassis bracket with two #6 x 1/4" screws.

2.2.4 Power Supply

1. Fasten the Power Supply to the CPU chassis bracket using #6 x 1/4" screws. Be sure the ground tab is fastened back in place.

2. Reconnect all cables (power supply, video, keyboard, and cassette cables, and RS-232 and FDC power cables and Disk ribbon cable if necessary).

2.2.5 Disk Drive (optional)

1. Place the Disk Drive in the bottom position and reconnect the ground wire and power supply connector.

2. Fasten the Drive with four #6-32 x 1/2" screws and four flat washers (two on each side).

3. Reconnect the FDC inter-connect cable to the rear of the Drive.

4. Position the second Disk Drive in the top position and reconnect the ground wire and power supply connector.

5. Fasten with four screws and washers (two on each side).

6. Reconnect the FDC inter-connect cable to the rear of the Drive.

2.2.6 Disk Drive Power Supply (optional)

1. Before installing the Power Supply, be sure that the bottom Disk Drive is mounted in place and the Disk Shield is in position on the Disk Mounting Bracket.

2. Reconnect all cables and wires to the Power Supply.

3. Fasten the Power Supply with four #6 x 3/8" screws. Be sure the ground tab is fastened back in place.

2.2.7 Video Monitor (CRT) and Video Board

1. Position the CRT in the Case Top and install the upper left and lower right #10 washers and nuts.

2. Install the upper right and lower left #10 washers and nuts. Be sure to reconnect the ground wire from the CPU cable. It will require two nuts to fasten it.

3. Install the Video Board into the Case and fasten with two #6 x 3/8" screws.

4. Connect the ground wire with solder lug back to the Video Board.

5. Install the plug on the rear of the CRT neck.

6. Install the four color coded wires with spade lugs to their associated terminals (as determined by a colored dot on the yoke near each terminal).

7. Install the high-voltage anode wire on the side of the CRT. Use a screwdriver to compress the clip and insert it into the CRT. Press down on the suction cup to secure.

2.2.8 Case

1. Double-check to be sure all wires are connected correctly and all Boards are properly fastened.

2. Attach the chassis shield (if used) with #6 x 1/4" screws and reconnect the ground connectors.

3. Carefully place the Case Top over the Case Bottom. **Do not hit the CRT neck. It could implode or break off.**

4. Install the #6 x 3/8" sheet metal screw and flat washer in the top rear panel of the Case.

5. Carefully rest the Computer on its rear panel and replace the ten #8 screws; five 1" sheet metal toward rear, three 7/8" machine head along front, and two 1" machine head in remaining positions.

SECTION III

CPU CIRCUIT BOARD

CPU CIRCUIT BOARD

3.1 MODEL 4 THEORY OF OPERATION

3.1.1 Introduction

The TRS-80 Model 4 Microcomputer is a self-contained desktop microcomputer designed not only to be completely software compatible with the TRS-80 Model III, but to provide many enhancements and features. System distinctions which enable the Model 4 to be Model III compatible include: a Z80 CPU capable of running at a 4 MHz clock rate, BASIC operating system in ROM (14K), memory-mapped keyboard, 64-character by 16-line memory-mapped video display, up to 128K Random Access Memory, cassette circuitry able to operate at 500 or 1500 baud, and the ability to accept a variety of options. These options include: one to four 5-1/4 inch double density floppy disk drives, one to four five megabyte hard disk drives, an RS-232 Serial Communications Interface, and a 640 by 240 pixel high resolution graphics board.

3.1.2 CPU and Timing

The central processing unit of the Model 4 microcomputer is the Z80-A microprocessor — capable of running at either a two (2.02752) or four (4.05504) MHz clock rate. The main CPU timing comes from the 20 MHz (20.2752 MHz) crystal-controlled oscillator, Y1 and Q1. There is an additional 12 MHz (12.672 MHz) oscillator, Y2 and Q2, which is necessary for the 80 by 24 mode of video operation. The oscillator outputs are sent to two Programmable Array Logic (PAL) circuits, U3 and U4, for frequency division and routing of appropriate timing signals.

PAL U3 divides the 20 MHz signal by five for 4 MHz CPU operation, by ten for a 2 MHz rate, and slows the 4 MHz clock for the M1 Cycle (See Figure 3-3). U3 also divides the master clock by four to obtain a 5 MHz clock to be sent to the RS-232 option connector as a reference for the baud rate generator. PAL U4 selects an appropriate 10 MHz or 12 MHz clock for the video shift clock, and using divider U5 provides additional timing signals to the video display circuitry (See Fig. 3-4).

Hex latch U18 is clocked from the 20 MHz clock, and is used to provide MUX and CAS timing for the dynamic memory circuits. Also, with additional gates from U16, U19, U20, U31, and U32, this chip provides the wait circuitry necessary to prevent the CPU from accessing video RAM during the active portion of the display. This is done by latching the data for the video RAM and simultaneously forcing the Z80 CPU into a "WAIT" state and is necessary to eliminate undesirable "hashing" of the video display (See Fig. 3-4).

3.1.3 Buffering

Low level signals from and to the CPU need to be buffered, or current amplified in order to drive many other circuits. The 16 address lines are buffered by U55 and U56, which are unidirectional buffers that are permanently enabled. The eight data lines are buffered by U71. Since data must flow both to and from the CPU, U71 is a bi-directional buffer which can go into a three-state condition when not in use. Both direction and enable controls come from the address decoding section.

The clock signal to the CPU (from PAL U3) is buffered by active pullup circuit Q3 RESET and WAIT inputs to the CPU are buffered by U17 and U46. Control outputs from the Z80 (M1*, RD*, WR*, MREQ*, and IORQ*) are sent to PAL U58, which combines these into other appropriate control signals consistent with Model 4's architecture. Other than MREQ*, which is buffered by part of U38, the raw control signals go to no other components, and hence require no additional buffering.

3.1.4 Address Decoding

The address decoding section is divided into two sub-sections: Port address decoding and Memory address decoding.

In port address decoding, low order address lines (some combined through a portion of U32) are sent to the address and enable inputs of U48, U49, and U50. U48 is also enabled by the IN* signal, which means that is decodes port input signals, while U49 decodes port output signals. A table of the resulting port map is shown below:

Port Addr. (Hex)	Read Function	Write Function
FC - FF	Cassette In, Mode Read	Cassette Out, resets cassette data latch
F8 - FB	Read Printer Status	Output to Printer
(1) F4 - F7	- reserved -	Drive Select latch
(1) F3	FDC Data Reg.	FDC Data Reg.
(1) F2	FDC Sector Reg.	FDC Sector Reg.
(1) F1	FDC Track Reg.	FDC Track Reg.

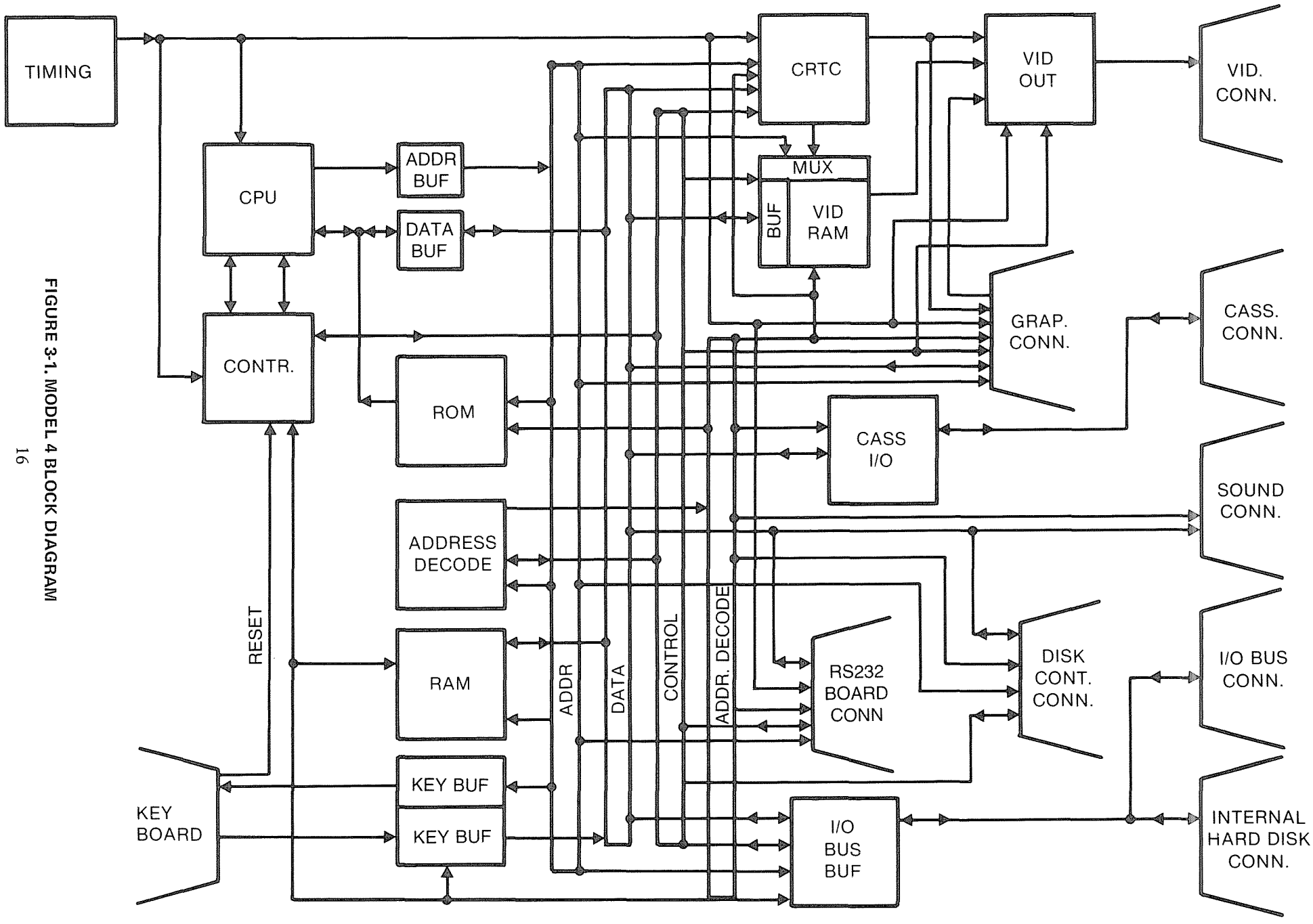


FIGURE 3-1. MODEL 4 BLOCK DIAGRAM

(1)	FØ	FDC Status Reg.	FDC Command Reg.
	EC - EF	Resets RTC Int.	Mode Output latch
(2)	EB	Rcvr Holding Reg.	Xmit Holding Reg.
(2)	EA	UART Status Reg.	UART/Modem control
(2)	E9	- reserved -	Baud Rate Register
(2)	E8	Modem Status	Master Reset/Enable UART control reg.
	E4 - E7	Read NMI Status	Write NMI Mask reg.
	E0 - E3	Read INT Status	Write INT Mask reg.
(3)	CF	HD Status	HD Command
(3)	CE	HD Size/Drv/Hd	HD Size/Drv/Hd
(3)	CD	HD Cylinder high	HD Cylinder high
(3)	CC	HD Cylinder low	HD Cylinder low
(3)	CB	HD Sector Number	HD Sector Number
(3)	CA	HD Sector Count	HD Sector Count
(3)	C9	HD Error Reg.	HD Write Precomp.
(3)	C8	HD Data Reg.	HD Data Reg.
(3)	C7	HD CTC channel 3	HD CTC channel 3
(3)	C6	HD CTC channel 2	HD CTC channel 2
(3)	C5	HD CTC channel 1	HD CTC channel 1
(3)	C4	HD CTC channel 0	HD CTC channel 0
(3)	C2 - C3	HD Device ID Reg.	- reserved -
(3)	C1	HD Control Reg.	HD Control Reg.
(3)	C0	HD Wr. Prot. Reg.	- reserved -
	94 - 9F	- reserved -	- reserved -
(4)	90 - 93	- reserved -	Sound Option
(5)	8C - 8F	Graphics Sel. 2	Graphics Sel. 2
	8B	CRTC Data Reg.	CRTC Data Reg.
	8A	CRTC Control Reg.	CRTC Control Reg.
	89	CRTC Data Reg.	CRTC Data Reg.
	88	CRTC Control Reg.	CRTC Control Reg.
	84 - 87	- reserved -	Options Register
(5)	83	- reserved -	Gra. X Reg. Write
(5)	82	- reserved -	Gra. Y Reg. Write
(5)	81	Graphics Ram Rd.	Graphics Ram Wr.
(5)	80	- reserved -	Gra. Options Reg. Wr

Notes:
(1) Valid only if FDC option is installed
(2) Valid only if RS-232 option is installed
(3) Valid only if Hard Disk option is installed
(4) Valid only if sound option is installed
(5) Valid only if High Resolution Graphics option is installed

Following is a Bit Map of the appropriate ports in the Model 4. Note that this is an "internal" bit map only. For bit maps of the optional devices, refer to the appropriate section of the desired manual.

Model 4 Port Bit Map

Port	D7	D6	D5	D4	D3	D2	D1	D0
FC - FF	Cass							Cassette
(READ)	data 500 bd			(MIRROR of PORT EC)				data 1500 bd
FC - FF			(Note, also resets cassette data latch)				cass.	cassette
(WRITE)	x	x	x	x	x	x	out	data out
F8 - FB (READ)	Prntr BUSY	Prntr Paper	Prntr Select	Prntr Fault	x x	x x	x x	x x
F8 - FB (WRITE)	Prntr D7	Prntr D6	Prntr D5	Prntr D4	Prntr D3	Prntr D2	Prntr D1	Prntr D0
EC - EF			(Any Read causes reset of Real Time Clock Interrupt)					
EC - EF (WRITE)	x x	CPU Fast	x x	Enable EX I/O	Enable Altset	Mode Select	Cass Mot On	x x
E0 - E3 (READ)	x x	Receive Error	Receive Data	Xmit Empty	10 Bus Int	RTC Int	C Fall Int	C Rise Int
E0 - E3 (WRITE)	x x	Enable Rec Err	Enable Rec Data	Enable Xmit Emp	Enable 10 Int	Enable RT Int	Enable CF Int	Enable CR Int
90 - 93 (WRITE)	x x	x x	x x	x x	x x	x x	x x	Sound Bit
84 - 87 (WRITE)	Page	Fix Upr Memory	Memory Bit 1	Memory Bit 0	Invert Video	80/64	Select Bit 1	Select Bit 0

Memory mapping is accomplished by PAL U59 in the Basic 16K or 64K computer. In a 128K system, PAL U72, along with the select and memory bits of the options register, also enter into the memory mapping function.

Four memory maps are listed below. Memory Map I is compatible with the Model III. Note that there are two 32K banks in the 64K system, which can be interchanged with either position of the upper two banks of a 128K system. The 128K system has four moveable 32K banks. Also note, in the Model III mode, that decoding for the printer status read (37E8 and 37E9 hexadecimal) is accomplished by U93 and leftover gates from U40, U46, U51, U54, U60, and U62.

Memory Map I — Model III Mode

	0000 — 1FFF	ROM A (8K)
	2000 — 2FFF	ROM B (4K)
	3000 — 37FF	ROM C (2K) — Less 37E8 - 37E9
	37E8 — 37E9	Printer Status Port
	3800 — 3BFF	Keyboard
	3C00 — 3FFF	Video RAM (Page bit selects 1K of 2K)
*	4000 — 7FFF	RAM (16K system)
*	4000 — FFFF	RAM (64K system)

Memory Map II

0000 – 37FF	RAM (14K)	
3800 – 3BFF	Keyboard	
3C00 – 3FFF	Video RAM	
4000 – 7FFF	RAM (16K)	End of one 32K Bank
8000 – FFFF	RAM (32K)	Second 32K Bank

Memory Map III

0000 – 7FFFF	RAM (32K)	End of One 32K Bank
8000 – F3FF	RAM (29K)	Second 32K Bank
F400 – F7FF	Keyboard	
F800 – FFFF	Video RAM	

Memory Map IV

0000 – 7FFF	RAM (32K)	One 32K Bank
8000 – FFFF	RAM (32K)	Second 32K Bank

(See Figure 3-2 for 128K Maps)

3.1.5 ROM

The Model 4 Microcomputer contains 14K of Read Only Memory (ROM), which is divided into an 8K ROM (U68), a 4K ROM (U69), and a 2K ROM (U70). ROMs used have three-state outputs which are disabled if the ROMs are deselected. As a result, ROM data outputs are connected directly to the CPU data bus and do not use data buffer U71, which is disabled during a ROM access.

ROMs are Model III compatible and contain a BASIC operating system, as well as a floppy disk boot routine. The enable inputs to the ROMs are provided by the address decoding section, and are present only in the Model III mode of operation.

3.1.6 RAM

Three configurations of Random Access Memory are available on the Model 4: 16K, 64K, and 128K. The 16K option uses 4116 type, 16K by 1 dynamic RAMs, which require three supply voltages (+12 volts, +5 volts, and –5 volts). The 64K and 128K options use 6665 type, 64K by 1 dynamic RAMs, which require only a single supply voltage (+5 volts). The proper voltage for each option is provided by jumpers.

Dynamic RAMs require multiplexed incoming address lines. This is accomplished by ICs U63 and U76. Output data from RAMs is buffered by U64. With the 128K option, there are two rows of the 64K by 1 RAM ICs. The proper row is selected by the CAS* signal from PAL U72.

3.1.7 Keyboard

The Model 4 Keyboard is a 70-key sculptured keyboard, scanned by the microprocessor. Each key is identified by its column and row position. Columns are defined by address lines A0 - A7, which are buffered by open-collector drivers U29 and U30. Data lines D0 - D7 define the rows and are buffered by CMOS buffers U44 and U45. Row inputs to the buffers are pulled up by resistor pack RP 1, unless a key in the current column being scanned is depressed. Then, the row for that key goes low.

3.1.8 Video

The heart of the video display circuitry in the Model 4 is the 68045 Cathode Ray Tube Controller. The CRTC allows two screen formats: 64 by 16 and 80 by 24. Since the 80 by 24 screen requires 1,920 screen memory locations, a 2K by 8 static RAM is used for the Video RAM. The 64 by 16 mode has a two-page screen display and a bit in the options register for determining which page is active for the CPU. Offset the start address of the CRTC to gain access to the second page in the 64 by 16 mode.

Addresses to the video RAM are provided by the 68045 when refreshing the screen and by the CPU when updating the data. These two sets of addresses are multiplexed by U33, U34, and U35. Data between the CPU and Video RAM is latched by U6 for a write, and buffered by U7 for a read operation.

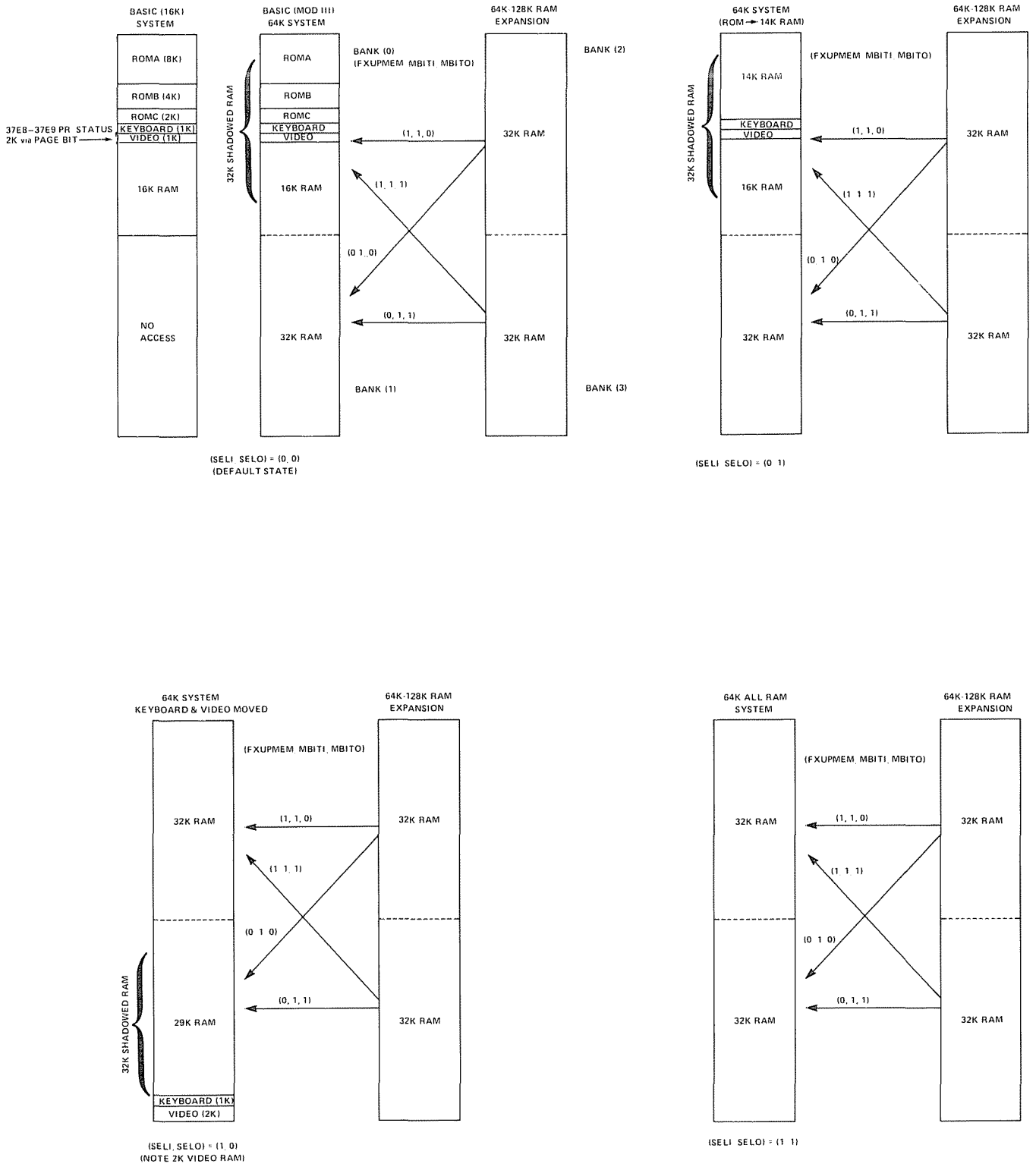


FIGURE 3-2. RAM MEMORY

During screen refresh, the data outputs of the Video RAM (ASCII character codes) are latched by U8 and become the addresses for the character generator ROM (U23). In cases of low resolution graphics, a dual 1 of 4 data selector (U9) is the cell generator, with additional buffering from U10

The shift register U11 inputs are the latched data outputs of the character or cell generator. The shift clock input comes from the PAL U4, and is 10.1376 MHz for the 64 by 16 mode and 12.672 MHz for 80 by 24 operation. The serial output from the shift register later becomes actual video dot information.

Special timing in the video circuit is handled by hex latch U2. This includes blanking (originating from CRTC) and shift register loading (originating from U4). Additional video control and timing functions, such as sync buffering, inversion selection, dot clock chopping, and graphics disable of normal video, are handled by miscellaneous gates in U12, U13, U14, U22, U24, and U26.

3.1.9 Real Time Clock

The Real Time Clock circuit in the Model 4 provides a 30 Hz (in the 2 MHz CPU Mode) or 60 Hz (in the 4 MHz CPU Mode) interrupt to the CPU. By counting the number of interrupts that have occured, the CPU can keep track of the time. The 60 Hz vertical sync signal from the video circuitry is divided by two (2 MHz Mode) by U53, and the 30 Hz at pin 1 of U51 is used to generate the interrupts. In the 4 MHz mode, signal FAST places a logic low at pin 1 of U51, causing signal VSYNC to trigger the interrupts at the 60 Hz rate. Note that any time interrupts are disabled, the accuracy of the clock suffers.

3.1.10 Cassette Circuitry

The cassette write circuitry latches the two LSBs (D0 and D1) for any output to port FF (hex). The outputs of these latches (U27) are then resistor summed to provide three discrete voltage levels (500 Baud only). The firmware toggles the bits to provide an output signal of the desired frequency at the summing node.

There are two types of cassette Read circuits — 500 baud and 1500 baud. The 500 baud circuit is compatible with both Model 1 and III. The input signal is amplified and filtered by Op amps (U43 and U28. Part of U15 then forms a Zero Crossing Detector, the output of which sets the latch U40. A read of Port FF enables buffer U41, which allows the CPU to determine whether the latch has been set, and simultaneously resets the latch. The firmware determines by the timing between settings of the latch whether a logic "one" or "zero" was read in from the tape.

The 1500 baud cassette read circuit is compatible with the Model III cassette system. The incoming signal is compared to a threshold by part of U15. U15's output will then be either high or low and clock about one-half of U39, depending on whether it is a rising edge or a falling edge. If interrupts are enabled, the setting of either latch will generate an interrupt. As in the 500 baud circuit, the firmware decodes the interrupts into the appropriate data.

For any cassette read or write operation, the cassette relay must be closed in order to start the motor of the cassette deck. A write to port EC hex with bit one set will set latch U42, which turns on transistor Q4 and energizes the relay K1. A subsequent write to this port with bit one clear will clear the latch and de-energize the relay.

3.1.11 Printer Circuitry

The printer status lines are read by the CPU by enabling buffer U67. This buffer will be enabled for any input from port F8 or F9, or any memory read from location 37E8 or 37E9 when in the Model III mode. For a listing of bit status, refer to the bit map.

After the printer driver software determines that the printer is ready to receive another character (by reading the status), the character to be printed is output to port F8. This latches the character into U66, and simultaneouly fires the one-shot U65 to provide the appropriate strobe to the printer.

3.1.12 I/O Connectors

Two 20-pin single inline connectors, J7 and J8, are provided for the connection of a Floppy Disk Controller and an RS-232 Communications Interface, respectively. All eight data lines and the two least significant address lines are routed to these connectors. In addition, connections are provided for device or board selection, interrupt enable, interrupt status read, interrupt acknowledge, RESET, and the CPU WAIT signal.

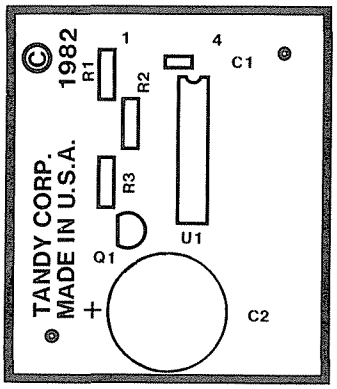
The graphics connector, J10, contains all of the above interface signals, plus CRTCLK, the dotclock signal, a graphics enable input, and other timing clocks which synchronize the graphics board with the CRTC.

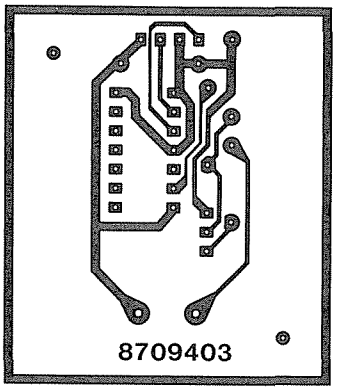
The I/O bus connector, J2, contains connections for all eight data lines (buffered by U74), the low order address lines (buffered by U73), and the control lines (buffered by U75) IN*, OUT*, RESET*, M1*, and IORQ*. In addition, the I/O bus connector has inputs to allow the device(s), connected to generate CPU WAIT states and interrupts.

The sound connector, J11, contains only four connections: sound enable (any output to port 90 hex), data bit D0, Vcc, and ground.

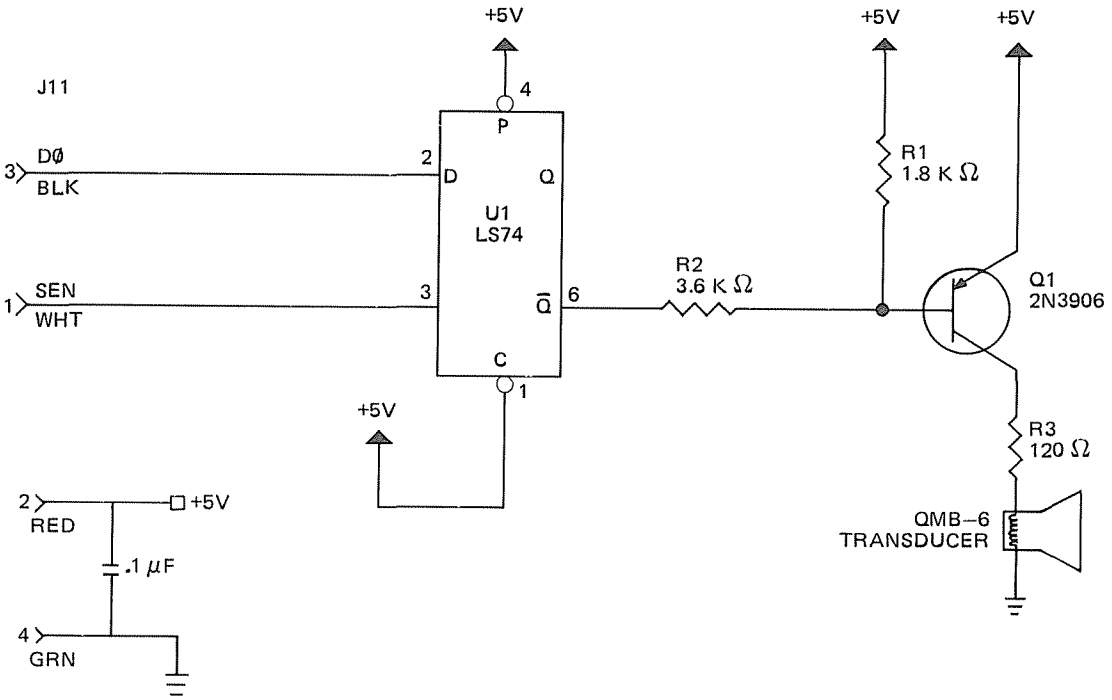
3.1.13 Sound Option

The Model 4 sound option, available as standard equipment on the disk drive versions, is a software intensive device. Data is sent out to port 90H, alternately setting and clearing data bit D0. The state of this bit is latched by sound board U1 and amplified by sound board Q1, which drives a piezoelectric sound transducer. The speed of the software loop determines the frequency, and thus, the pitch of the resulting tone.





COMPONENT LOCATION/CIRCUIT TRACE, SOUND BOARD #8858121



SCHEMATIC 8000188, SOUND BOARD #8858121

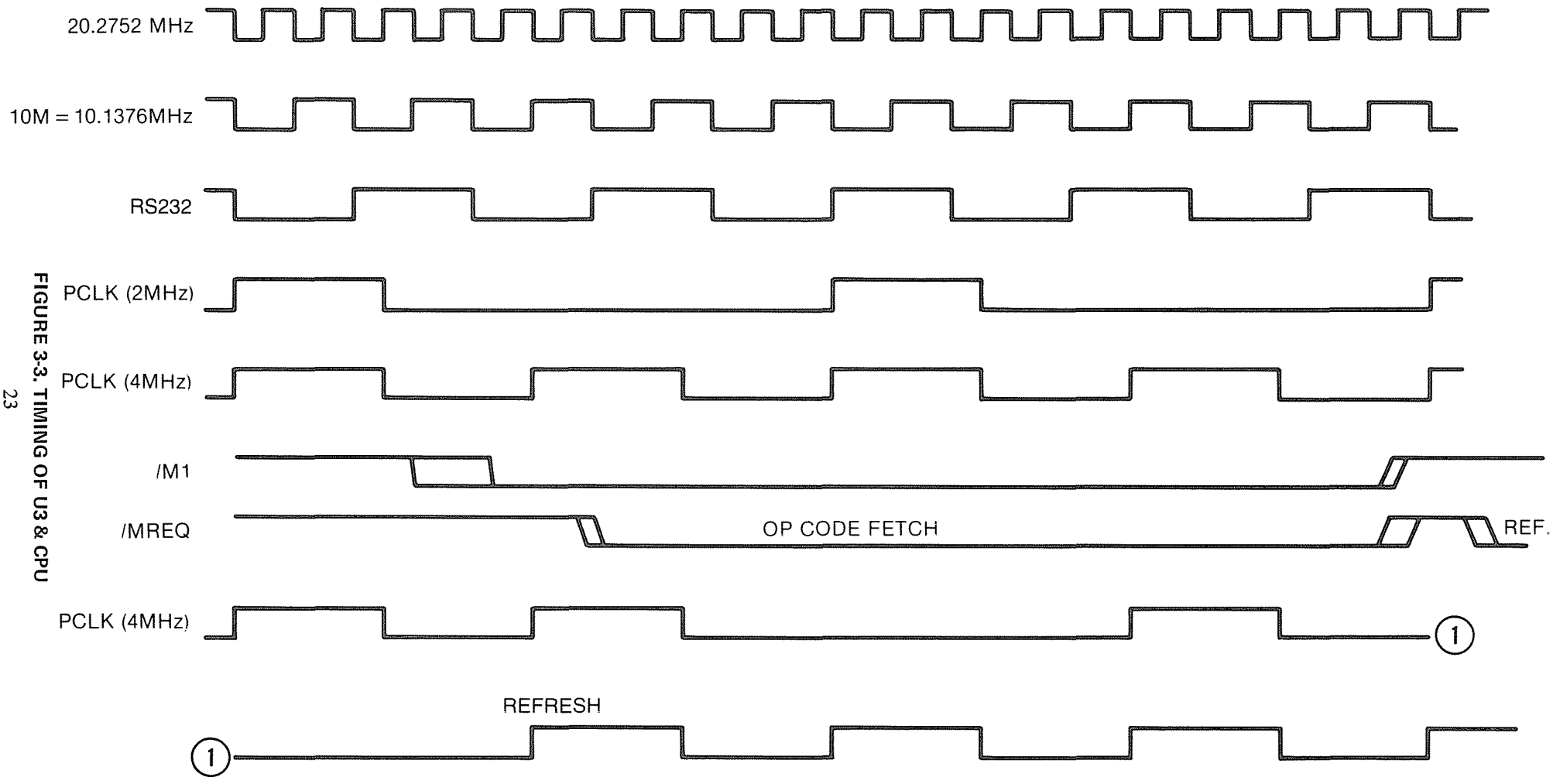


FIGURE 3-3. TIMING OF U3 & CPU

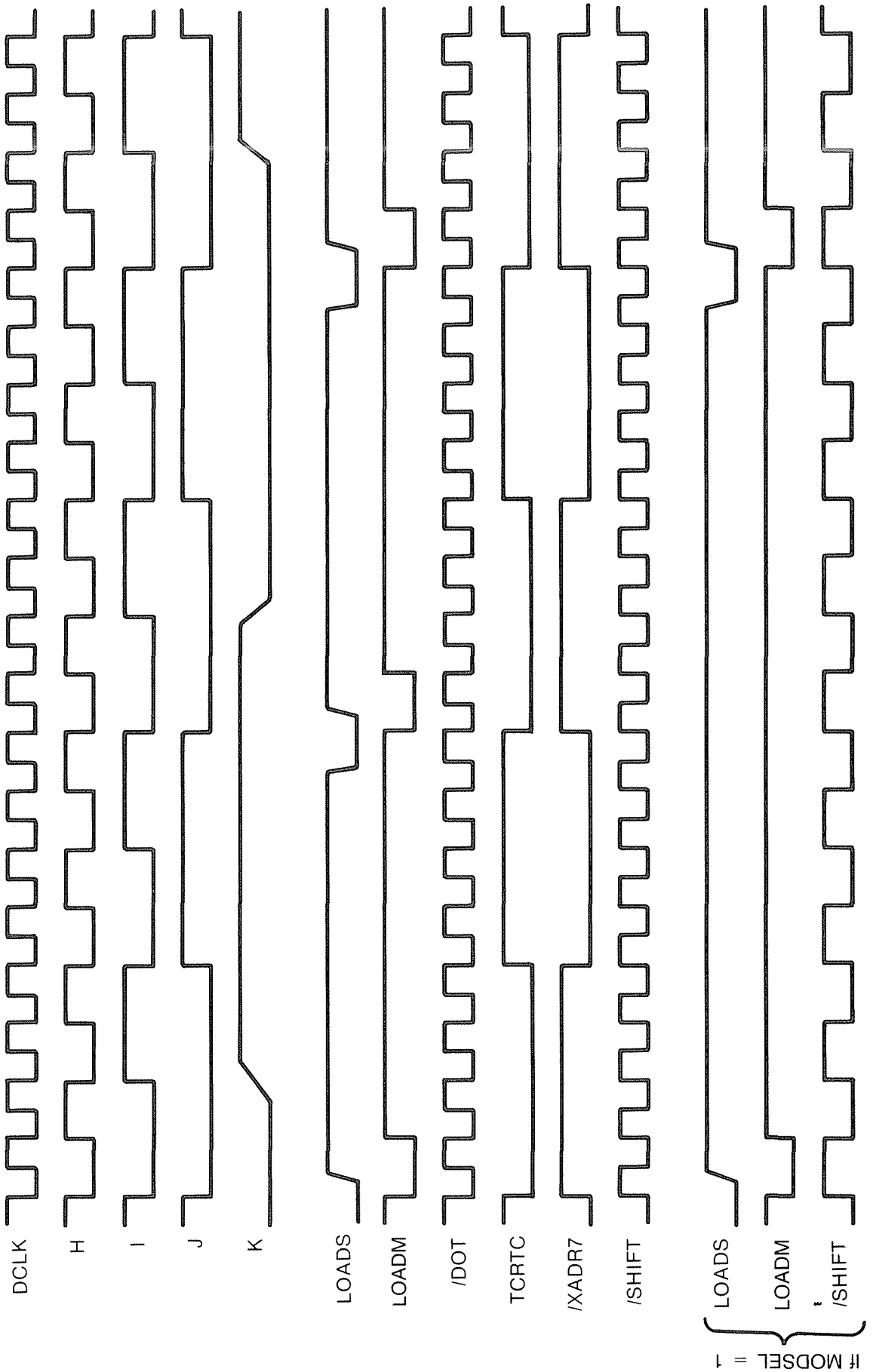


FIGURE 3-4. TIMING OF U4

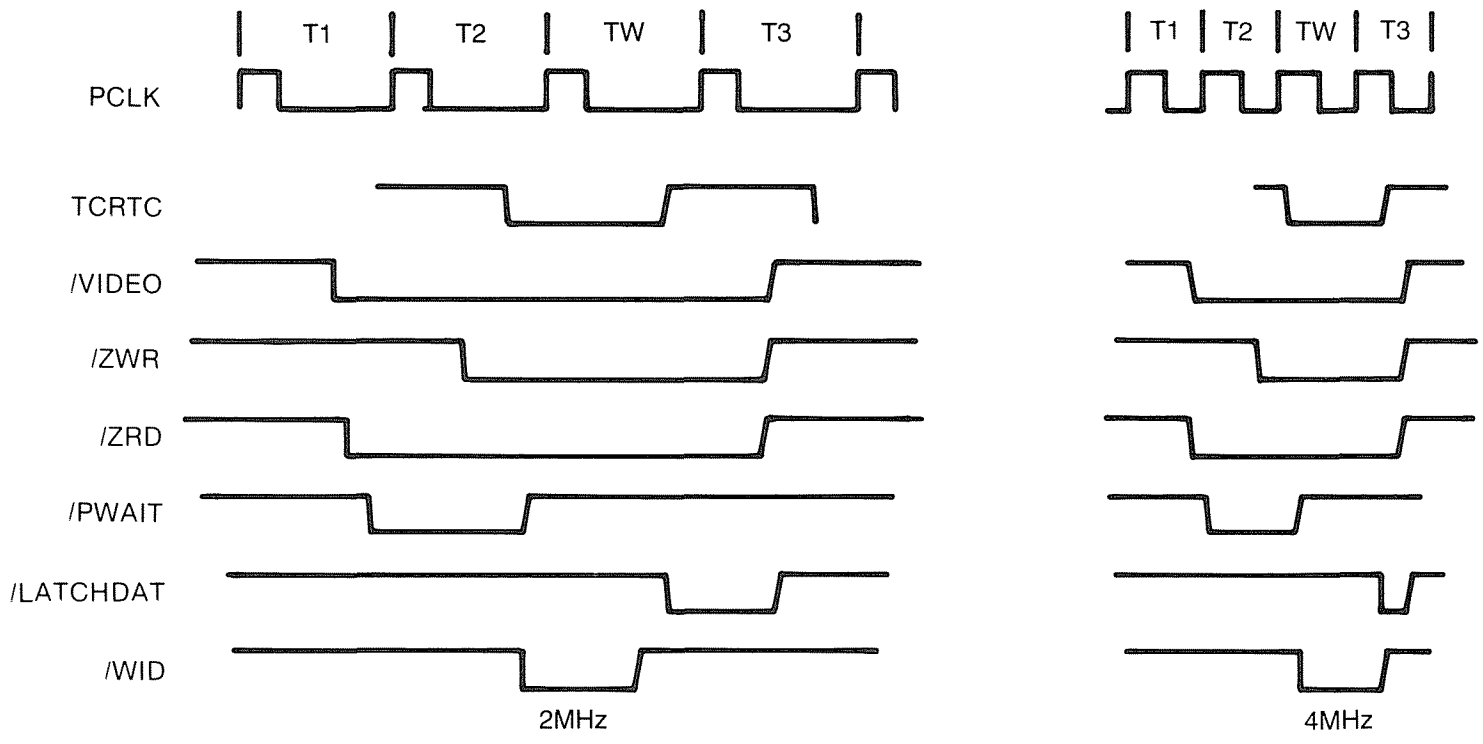


FIGURE 3-5. CPU VIDEO ACCESS TIMING

3.2 MODEL 4 I/O BUS

The Model 4 Bus is designed to allow easy and convenient interfacing of I/O devices to the Model 4. The I/O Bus supports all the signals necessary to implement a device compatible with the Z-80s I/O structure. That is:

Addresses:

A0 to A7 allow selection of up to 256† input and 256 output devices if external I/O is enabled.

†Ports 80H to 0FFH are reserved for System use.

Data:

DB0 to DB7 allow transfer of 8-bit data onto the processor data bus if external I/O is enabled.

Control Lines:

a. IN* – Z-80 signal specifying that an input is in progress. Gated with IORQ.
b. OUT* – Z-80 signal specifying that an output is in progress. Gated with IORQ.
c. RESET* – system reset signal.
d. IOBUSINT* – input to the CPU signaling an interrupt from an I/O Bus device if I/O Bus interrupts are enabled.
e. IOBUSWAIT* – input to the CPU wait line allowing I/O Bus device to force wait states on the Z-80 if external I/O is enabled.
f. EXTIOSEL* – input to CPU which switches the I/O Bus data bus transceiver and allows an INPUT instruction to read I/O Bus data.
g. M1* – and IORQ* – standard Z-80 signals . . . .

The address line, data line, and control lines a to c and e to g are enabled only when the ENEXIO bit in EC is set to a one.

To enable I/O interrupts, the ENIOBUSINT bit in the CPU IOPORT E0 (output port) must be a one. However, even if it is disabled from generating interrupts, the status of the IOBUSINT* line can still read on the appropriate bit of CPU IOPORT E0 (input port).

See Model 4 Port Bit assignment for port 0FF, 0EC, and 0E0 on pages 28 and 29.

The Model 4 CPU board is fully protected from "foreign I/O devices" in that all the I/O Bus signals are buffered and can be disabled under software control. To attach and use an I/O device on the I/O Bus, certain requirements (both hardware and software) must be met.

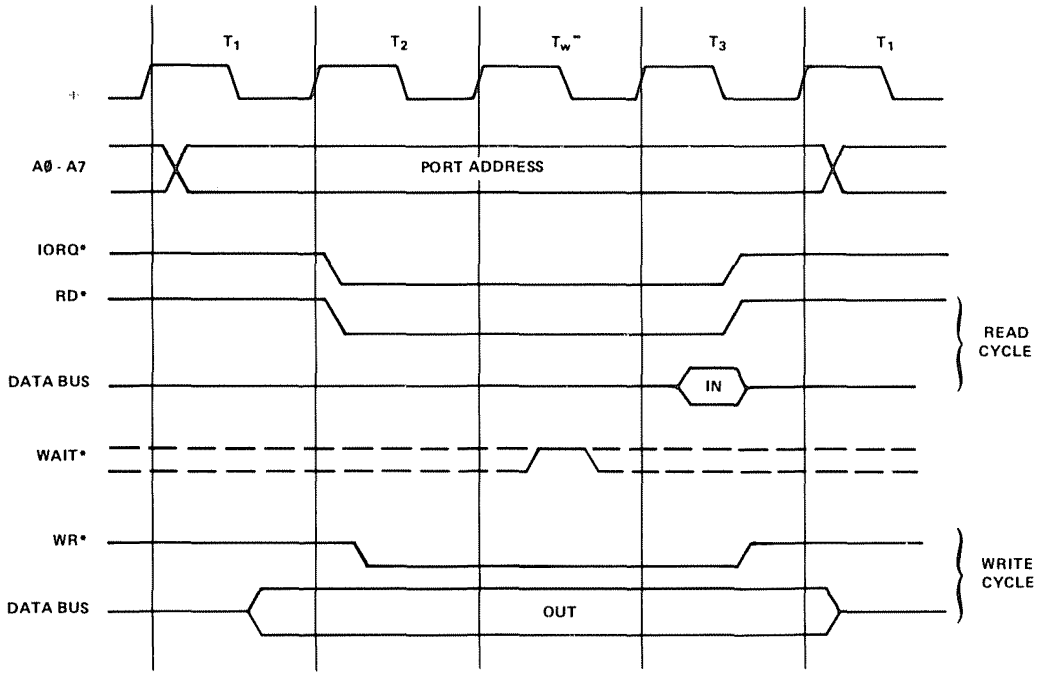
For input port device use, you must enable external I/O devices by writing to port 0ECH with bit 4 on in the user software. This will enable the data bus address lines and control signals to the I/O Bus edge connector. When the input device is selected, the hardware will acknowlege by asserting EXTIOSEL* low. This switches the data bus transceiver and allows the CPU to read the contents of the I/O Bus data lines. See Figure 3.6 for the timing. EXTIOSEL* can be generated by NANDing IN and the I/O port address.

Output port device use is the same as the input port device in use, in that the external I/O devices must be enabled by writing to port 0ECH with bit 4 on in the user software – in the same fashion.

For either input or output devices, the IOBUSWAIT* control line can be used in the normal way for synchronizing slow devices to the CPU. Note that since dynamic memories are used in the Model 4, the wait line should be used with caution. Holding the CPU in a wait state for 2 msec or more may cause loss of memory contents since refresh is inhibited during this time. It is recommended that the IOBUSWAIT* line be held active no more than 500 μsec with a 25% duty cycle.

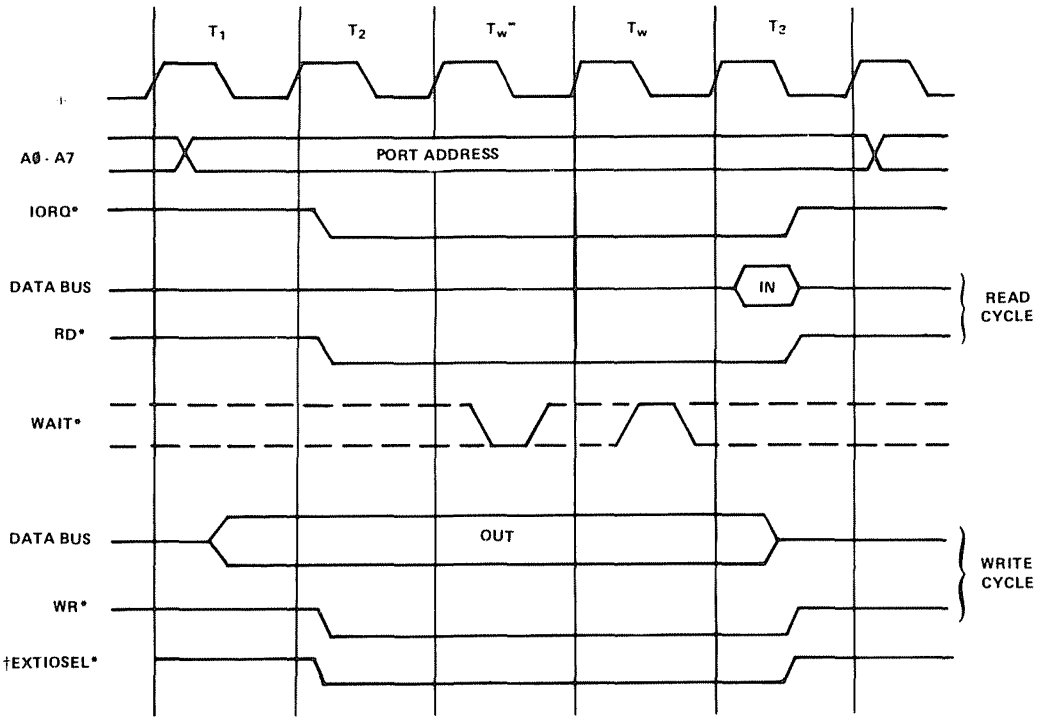
The Model 4 will support Z-80 mode 1 interrupts. A RAM jump table is supported by the LEVEL II BASIC ROMs and the user must supply the address of his interrupt service routine by writing this address to locations 403E and 403F. When an interrupt occurs, the program will be vectored to the user supplied address if I/O Bus interrupts have been enabled. To enable I/O Bus interrupts, the user must set bit 3 of Port 0E0H.

Input or Output Cycles.



"Inserted by Z80 CPU

Input or Output Cycles with Wait States.



"Inserted by Z80 CPU

†Coincident with IORQ* only on INPUT cycle.

FIGURE 3-6. I/O BUS TIMING DIAGRAM

3.3 MODEL 4 PORT BITS

Name: WRNMIMASKREG*
Port Address: ØE4H
Access: WRITE ONLY

Bit 7 = ENINTRQ; Ø disables Disk INTRQ from generating an NMI.
1 enables above.

Bit 6 = ENDRQ; Ø disables Disk DRQ from generating an NMI.
1 enables above.

Name: RDNMISTATUS*
Port Address: ØE4H
Access: READ ONLY

Bit 7 = Status of Disk INTRQ; 1 = False, Ø = True

Bit 6 = Status of Disk DRQ; 1 = False, Ø = True

Bit 5 = Reset* Status; 1 = False, Ø = True

Name: MOD OUT
Port Address: ØECH
Access: WRITE ONLY

Bit 7 = Undefined

Bit 6 = Undefined

Bit 5 = DISWAIT; Ø disables video waits, 1 enables

Bit 4 = ENEXTIO; Ø disables external IO Bus, 1 enables

Bit 3 = ENALTSET; Ø disables alternate character set,
1 enables alternate video character set.

Bit 2 = MODSEL; Ø enables 64 character mode,
1 enables 32 character mode.

Bit 1 = CASMOTORON; Ø turns cassette motor off,
1 turns cassette motor on.

Bit Ø = Undefined

Name: RDINTSTATUS*
Port Address: ØEØH
Access: READ ONLY

NOTE: A Ø indicates the device is interrupting.

Bit 7 = Undefined

Bit 6 = RS-232 ERROR INT

Bit 5 = RS-232 RCV INT

Bit 4 = RS-232 XMIT INT

Bit 3 = IOBUS INT

Bit 2 = RTC INT

Bit 1 = CASSETTE (1500 Baud) INT F

Bit Ø = CASSETTE (1500 Baud) INT R

Name: CASOUT*
Port Address: ØFFH
Access: WRITE ONLY

Bit 7 = Undefined

Bit 6 = Undefined

Bit 5 = Undefined

Bit 4 = Undefined

Bit 3 = Undfined

Bit 2 = Undefined

Bit 1 = Cassette output level

Bit Ø = Cassette output level

Name: WRINTMASKREG*
Port Address: ØEØH
Access: WRITE ONLY

Bit 7 = Undefined

Bit 6 = ENERRORINT; 1 enables RS-232 interrupts on parity error, framing error, or data overrun error.
Ø disable above.

Bit 5 = ENRCVINT; 1 enables RS-232 receive data register full interrupts,
Ø disables above.

Bit 4 = ENXMITINT; 1 enables RS-232 transmitter holding register empty interrupts,
Ø disables above.

Bit 3 = ENIOBUSINT; 1 enables I/O Bus interrupts,
Ø disables the above.

Bit 2 = ENRTC; 1 enables real time clock interrupt,
Ø disables above.

Bit 1 = ENCASINTF; 1 enables 1500 Baud falling edge interrupt,
Ø disables above.

Bit Ø = ENCASINTR; 1 enables 1500 Baud rising edge interrupt,
Ø disables above.

Name: CAS IN*
Port Address: ØFFH
Access: READ ONLY

Bit 7 = 500 Baud Cassette bit

Bit 6 = Undefined

Bit 5 = DISWAIT (See Port ØECH definition)

Bit 4 = ENEXTIO (See Port ØECH definition)

Bit 3 = ENALTSET (See Port ØECH definition)

Bit 2 = MODSEL (See Port ØECH definition)

Bit 1 = CASMOTORON (See Port ØECH definition)

Bit Ø = 1500 Baud Cassette bit

NOTE: Reading Port ØFFH clears the 1500 Baud Cassette interrupts.

Name: DRVSEL*
Port Address: ØF4H
Access: WRITE ONLY

Bit 7 = FM*/MFM; Ø selects single density,
1 selects double density.

Bit 6 = WSGEN; Ø = no wait states generated,
1 = wait states generated.

Bit 5 = PRECOMP; Ø = no write precompensation,
1 = write precompensation enabled.

Bit 4 = SDSEL; Ø selects side Ø of diskette,
1 selects side 1 of diskette.

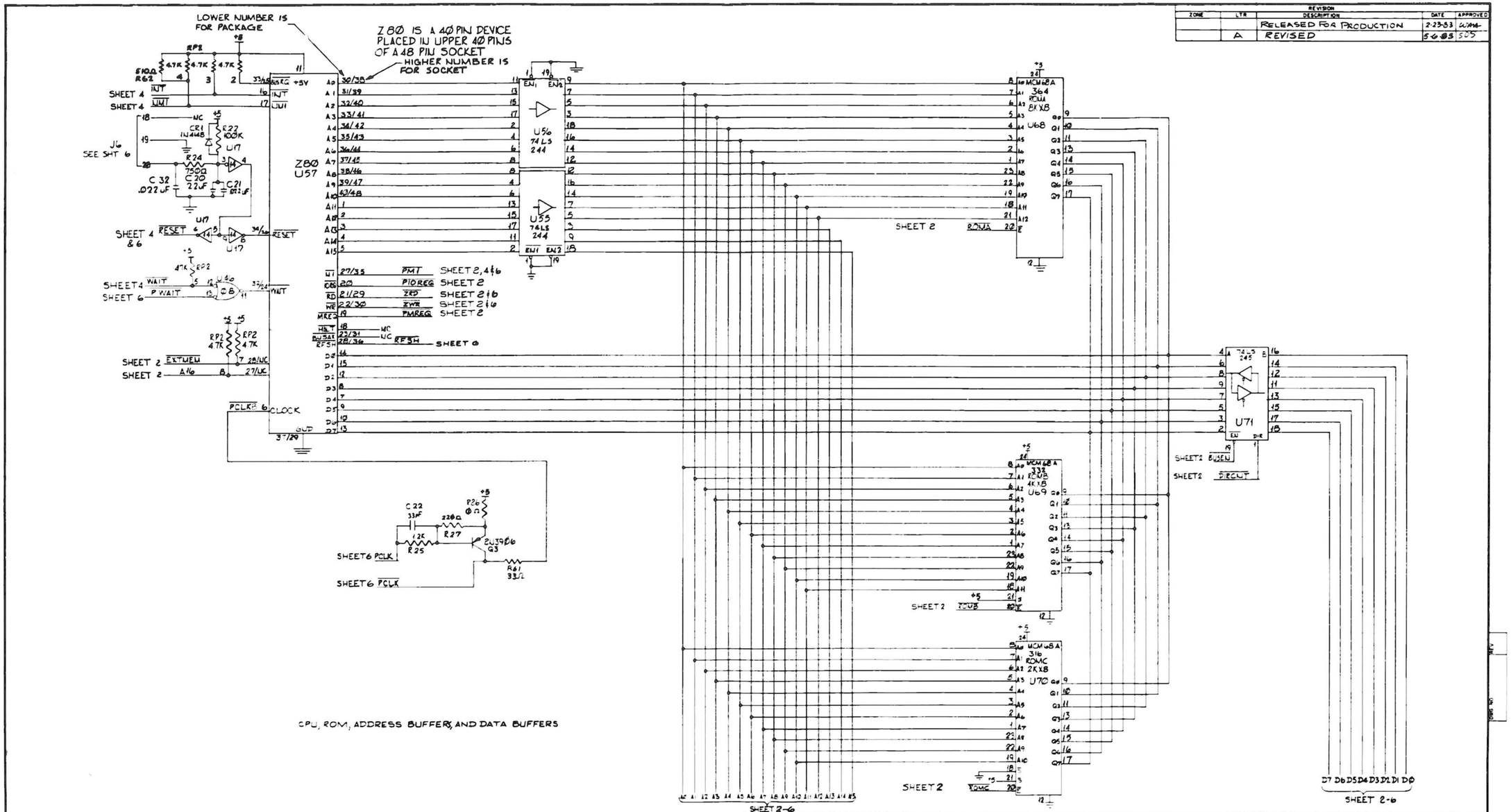
Bit 3 = Drive select 4

Bit 2 = Drive select 3

Bit 1 = Drive select 2

Bit Ø = Drive select 1

ZONE	LTR	REVISION DESCRIPTION	DATE	APPROVED
		RELEASED FOR PRODUCTION	2-23-83	WWH
	A	REVISED	5-6-85	SDS



PROPRIETARY INFORMATION

THE INFORMATION CONTAINED HEREIN IS PROPRIETARY AND IS EXCLUSIVE PROPERTY OF TANDY CORPORATION. NO REPRODUCTION OF ANY KIND MAY BE MADE WITHOUT THE EXPRESS WRITTEN AUTHORIZATION OF TANDY CORPORATION, 1000 TWO TANDY CENTER, FORT WORTH, TX 76102.

MATERIAL	UNLESS OTHERWISE SPECIFIED TOLERANCES XX = ± .010 XXX = ± .005 ANGLES = ± 1° HOLE DIA TOLERANCES .014 - .250 = +.005 -.001 .251 - .750 = +.008 -.001 .751 - UP = +.015 -.001		
FINISH	DIMENSIONS ARE IN INCHES AND APPLY AFTER PLATING		471
	DO NOT SCALE THIS DRAWING	NEXT ASSY	USED ON

DRAFT	S.D. HATFIELD	DATE	11-4-80
CHECK	G E G	DATE	
DESIGN	G E GAULKE	DATE	2/22/83
APPD		DATE	2-23-83
APPD		DATE	2/23/83

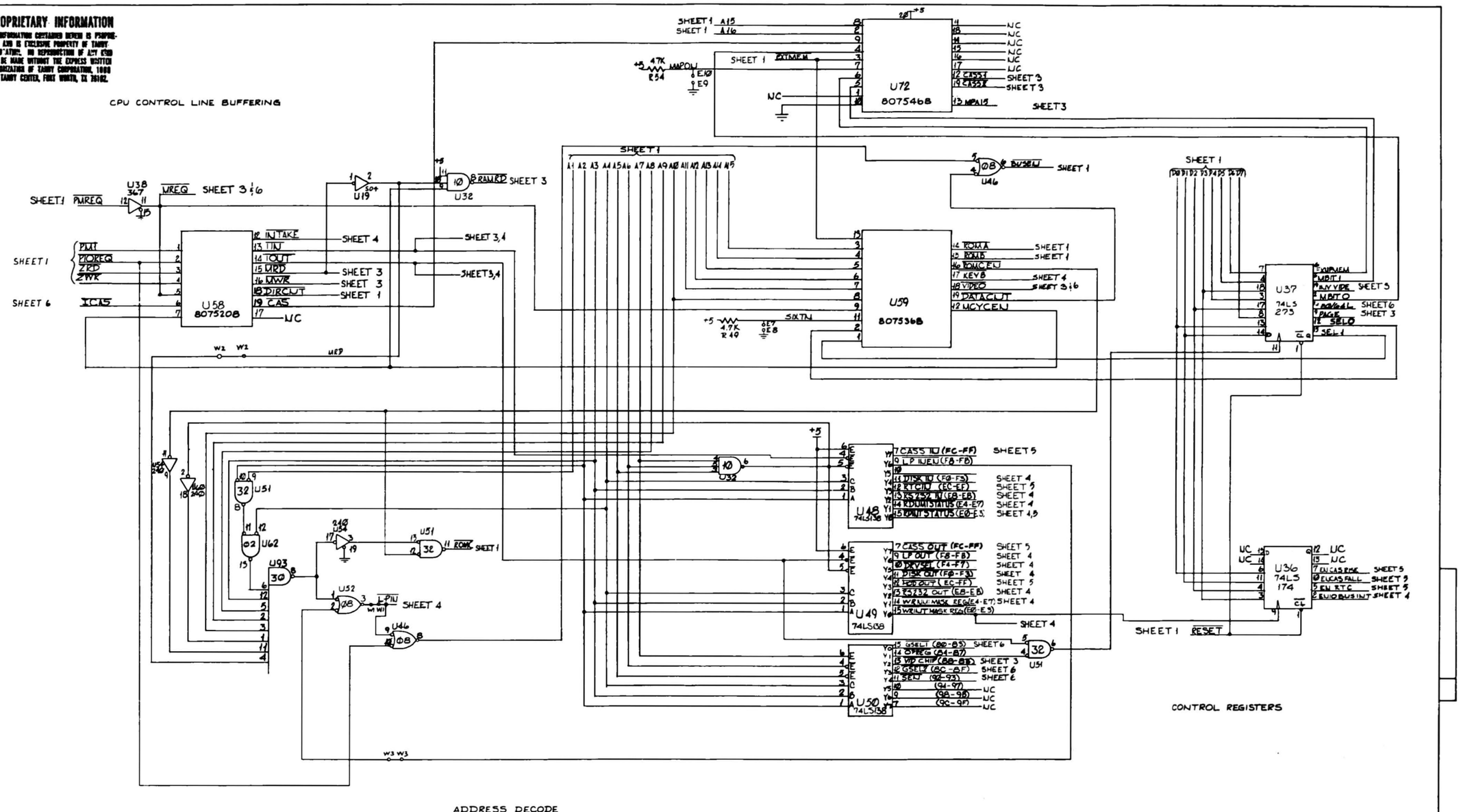
TITLE: SCHEMATIC - PROJECT R #471

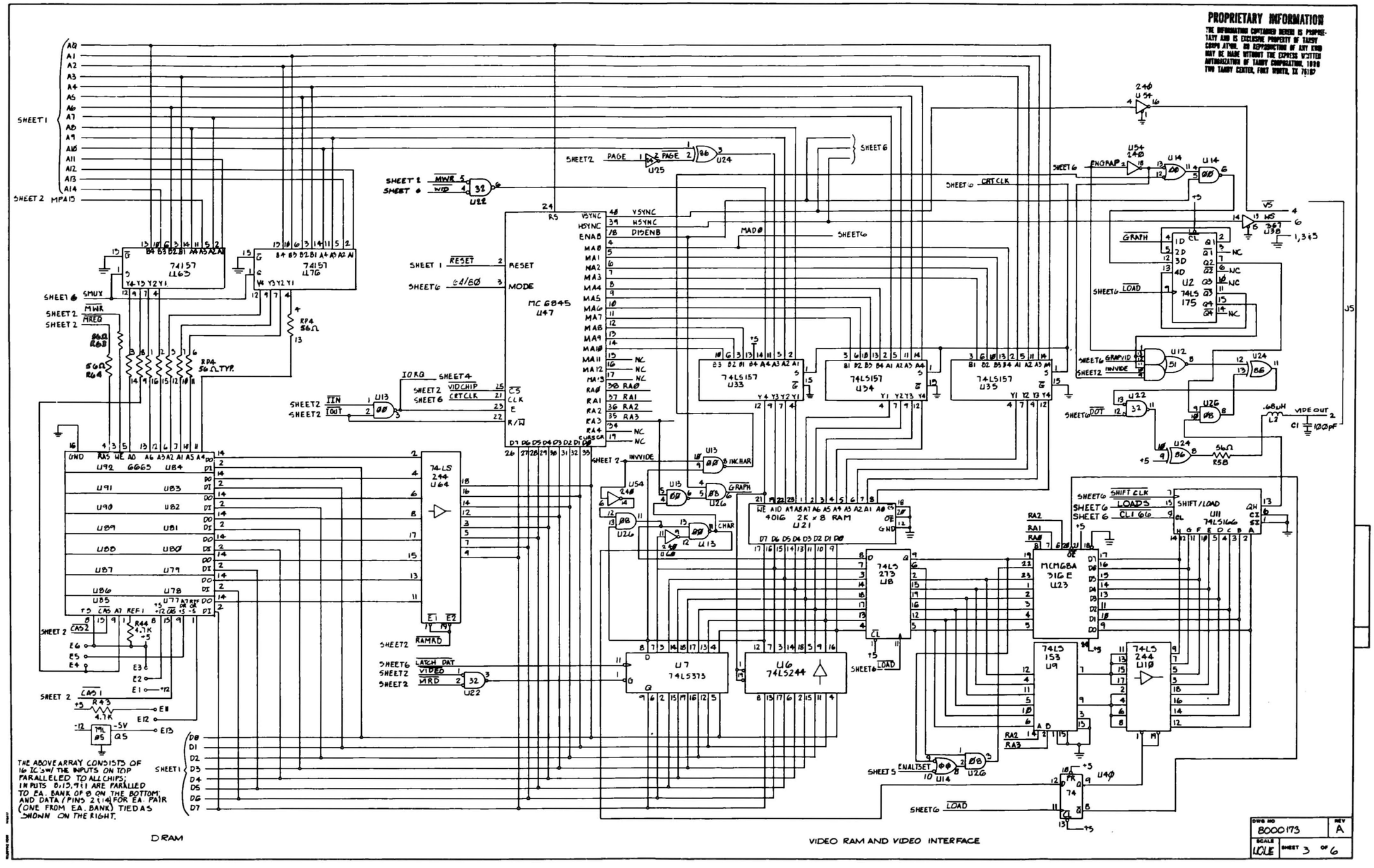
tandy

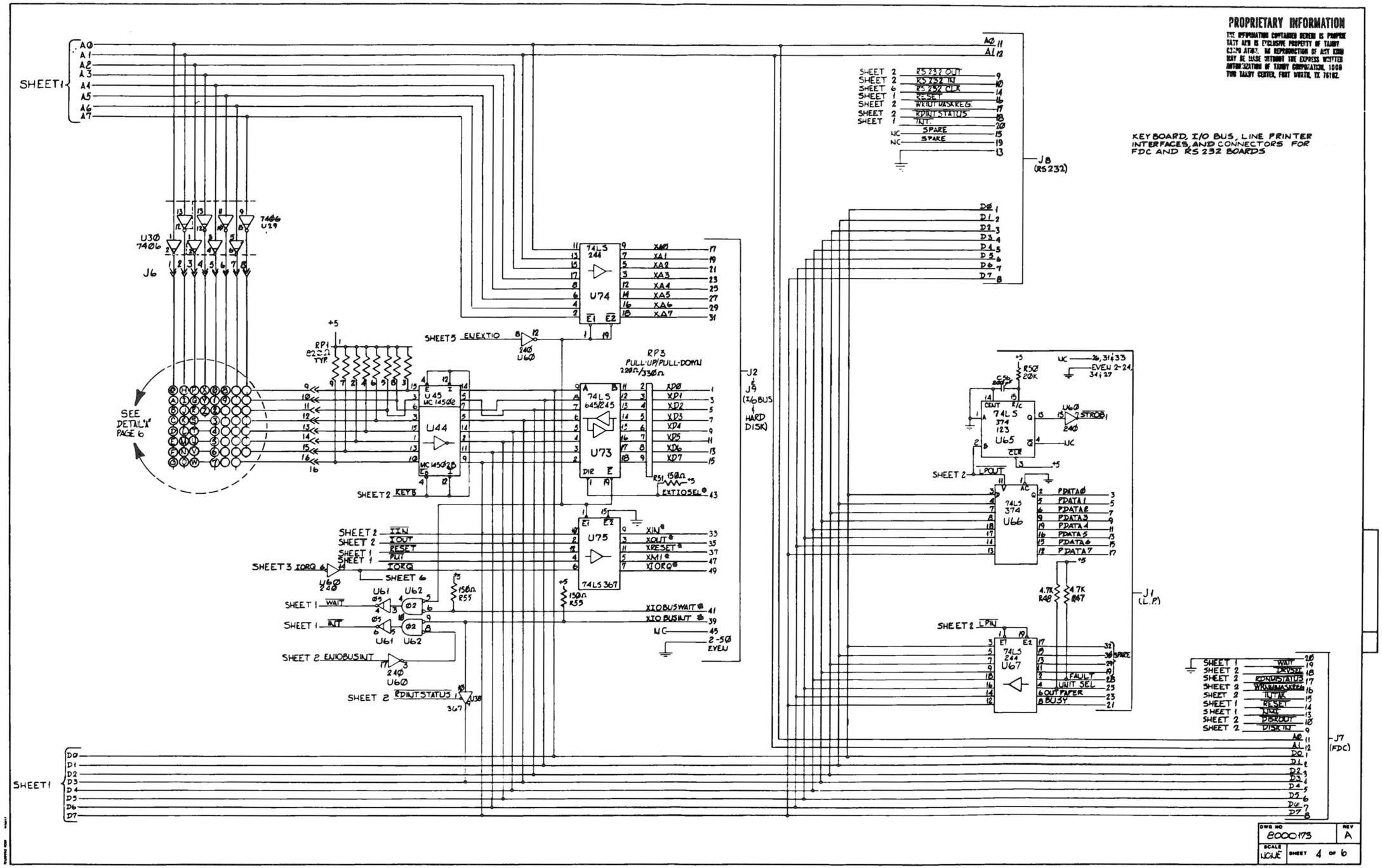
DWG NO	SIZE
8000173	D
SCALE NONE	SHEET 1 OF 6

PROPRIETARY INFORMATION

THE INFORMATION CONTAINED HEREIN IS PROPRIETARY AND IS EXCLUSIVE PROPERTY OF TANDY CORPORATION. NO REPRODUCTION OF ANY KIND MAY BE MADE WITHOUT THE EXPRESS WRITTEN AUTHORIZATION OF TANDY CORPORATION, 1000 TWO TANDY CENTER, FORT WORTH, TX 76102.

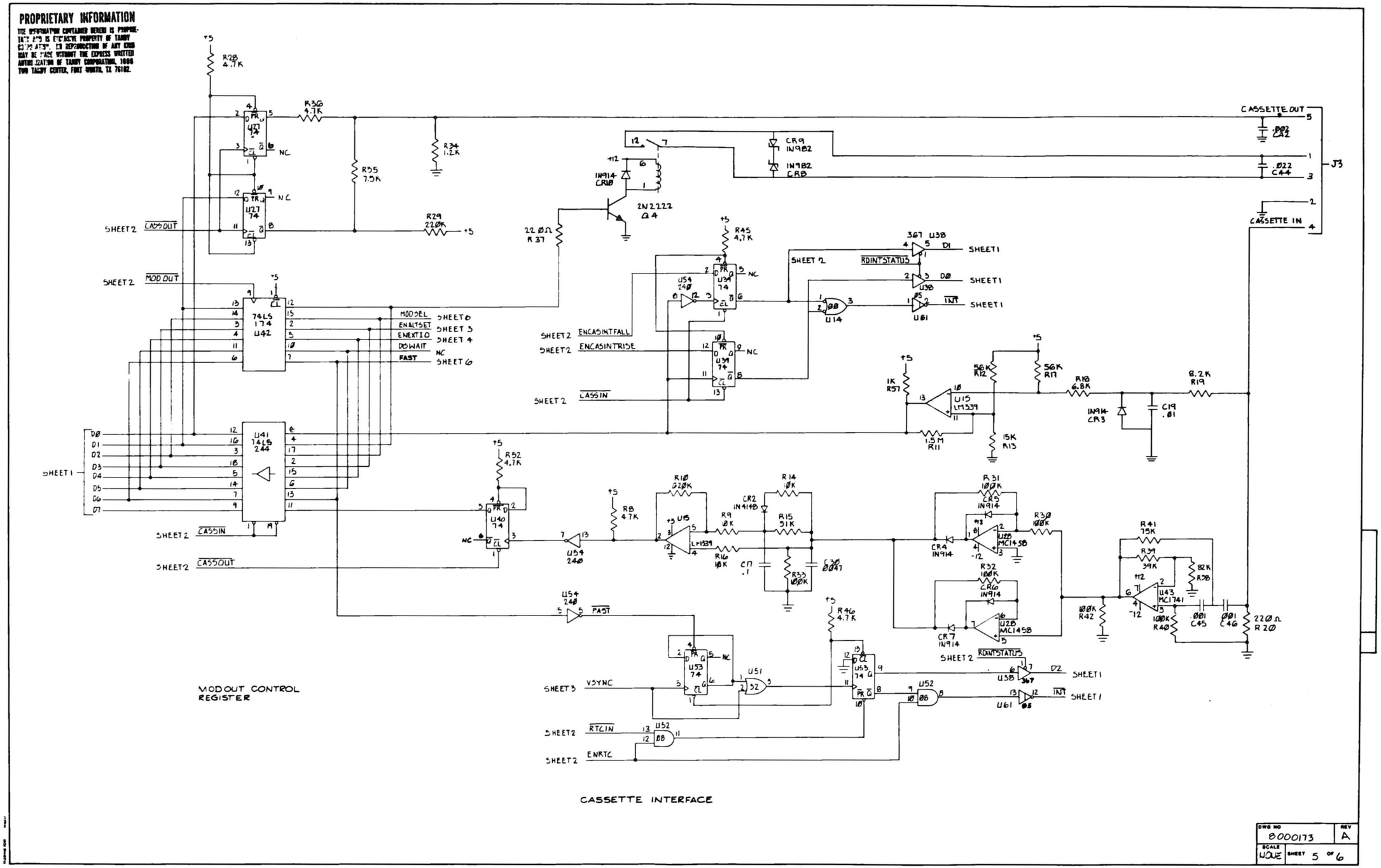



PROPRIETARY INFORMATION
THE INFORMATION CONTAINED HEREIN IS PROPRIETARY AND IS EXCLUSIVE PROPERTY OF TANDY CORPORATION. NO REPRODUCTION OF ANY KIND MAY BE MADE WITHOUT THE EXPRESS WRITTEN AUTHORIZATION OF TANDY CORPORATION, 1980
TWO TANDY CENTER, FORT WORTH, TX 76102
MC 6845
U47
74LS157
U33
U34
U35
4016 2K x 8 RAM
U21
MCM6BA 31GE
U23
74LS 244
U64
74LS 273
U8
U7
74LS373
U6
74LS244
74LS 153
U9
74LS 244
U10
SHIFT/LOAD
U11
74LS166
74LS 175
U2
THE ABOVE ARRAY CONSISTS OF 16 IC'S W/ THE INPUTS ON TOP PARALLELED TO ALL CHIPS; INPUTS 8,13,9 & 11 ARE PARALLED TO EA. BANK OF 8 ON THE BOTTOM; AND DATA (PINS 2 & 14) FOR EA. PAIR (ONE FROM EA. BANK) TIED AS SHOWN ON THE RIGHT.
DRAM
VIDEO RAM AND VIDEO INTERFACE
DWG NO 8000173
REV A
SCALE NONE
SHEET 3 OF 6


PROPRIETARY INFORMATION
THE INFORMATION CONTAINED HEREIN IS PROPRIETARY AND IS EXCLUSIVE PROPERTY OF TANDY CORPORATION. NO REPRODUCTION OF ANY KIND MAY BE MADE WITHOUT THE EXPRESS WRITTEN AUTHORIZATION OF TANDY CORPORATION, 1000 TWO TANDY CENTER, FORT WORTH, TX 76102.
KEYBOARD, I/O BUS, LINE PRINTER INTERFACES, AND CONNECTORS FOR FDC AND RS 232 BOARDS
SHEET 1
U30 7406
7406 U29
J6
SEE DETAIL "A" PAGE 6
RP1 822Ω TYP.
U44 MC 1450 2B
SHEET 2 KEYS
SHEET 5 EWEXTIO
240 U60
U74 74LS 244
RP3 PULL-UP/PULL-DOWN 220Ω/330Ω
U73 74LS 645/245
R51 150Ω
EXTIOSEL
U75 74LS 367
SHEET 2 IIN
SHEET 2 IOUT
SHEET 1 RESET
SHEET 1 PWI
SHEET 3 IORQ
SHEET 6
U61 U62
SHEET 1 WAIT
SHEET 1 INT
SHEET 2 EXIOBUSINT
SHEET 2 RDINTSTATUS
XIOBUSWAIT
XIOBUSINT
NC
2-50 EVEN
J2 J9 (I/O BUS & HARD DISK)
SHEET 2 RS232 OUT
SHEET 2 RS232 IN
SHEET 6 RS232 CLR
SHEET 1 RESET
SHEET 2 WRINTMASKREG
SHEET 2 RDINTSTATUS
SHEET 1 INT
SPARE
SPARE
J8 (RS232)
U65 74LS 374 123
U60 240 STROBE
SHEET 2 LPOUT
U66 74LS 374
PDATA0
PDATA1
PDATA2
PDATA3
PDATA4
PDATA5
PDATA6
PDATA7
J1 (L.P.)
SHEET 2 LPIN
U67 74LS 244
FAULT
UNIT SEL
OUT PAPER
BUSY
SPACE
SHEET 1 WAIT
SHEET 2 DRVSEL
SHEET 2 RDINTSTATUS
SHEET 2 WRINTMASKREG
SHEET 2 INTRK
SHEET 1 RESET
SHEET 1 INT
SHEET 2 DSKROT
SHEET 2 DISKINT
J7 (FDC)
DWG NO 8000173
REV A
SCALE NONE
SHEET 4 OF 6

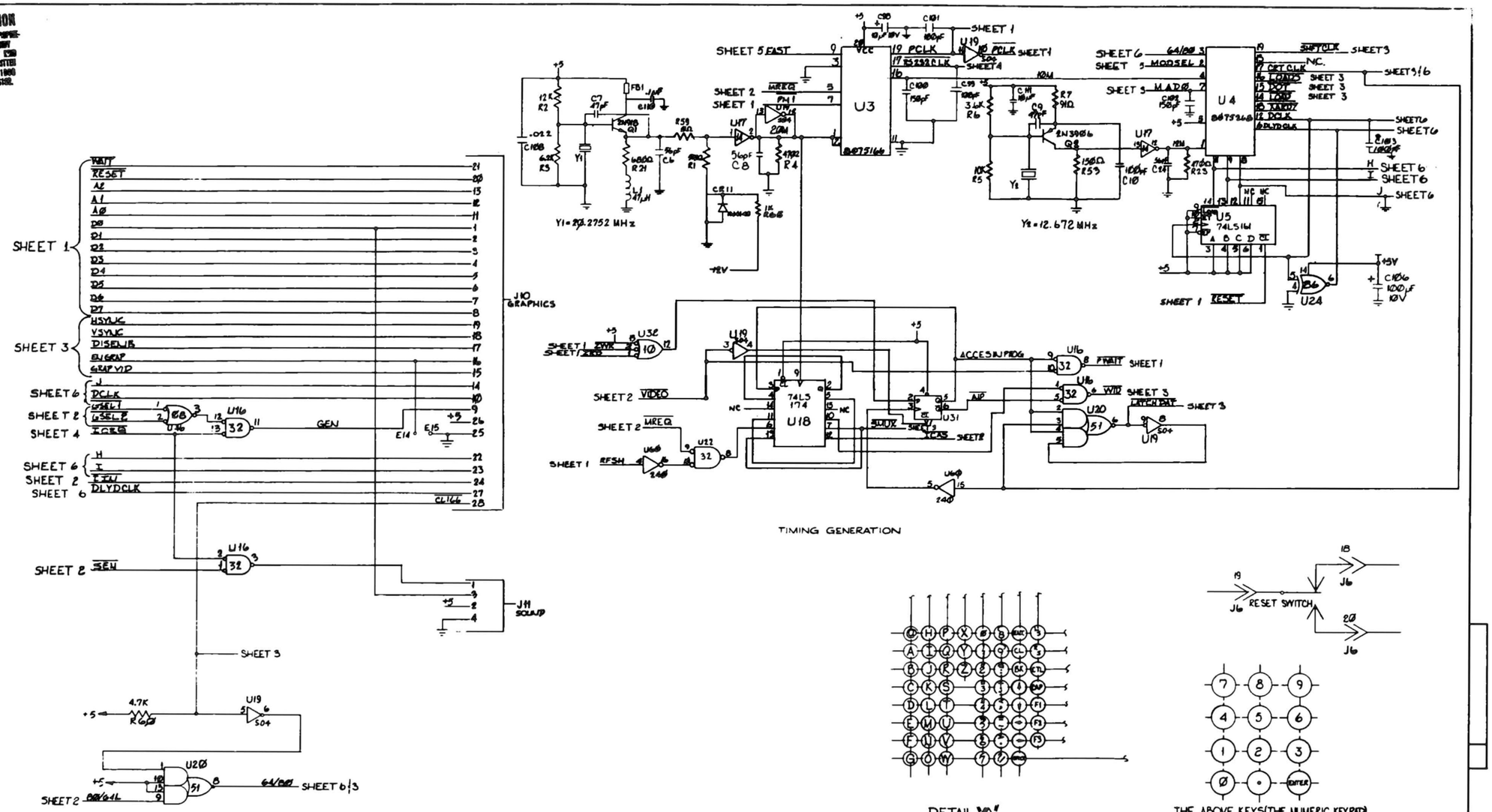
PROPRIETARY INFORMATION

THE INFORMATION CONTAINED HEREIN IS PROPRIETARY AND IS EXCLUSIVE PROPERTY OF TANDY CORPORATION. NO REPRODUCTION OF ANY KIND MAY BE MADE WITHOUT THE EXPRESS WRITTEN AUTHORIZATION OF TANDY CORPORATION, 1800 TWO TANDY CENTER, FORT WORTH, TX 76102.



PROPRIETARY INFORMATION

THE INFORMATION CONTAINED HEREIN IS PROPRIETARY AND IS EXCLUSIVE PROPERTY OF TANDY CORPORATION. NO REPRODUCTION OF ANY KIND MAY BE MADE WITHOUT THE EXPRESS WRITTEN AUTHORIZATION OF TANDY CORPORATION, 1980 TWO TANDY CENTER, FORT WORTH, TX 76102.

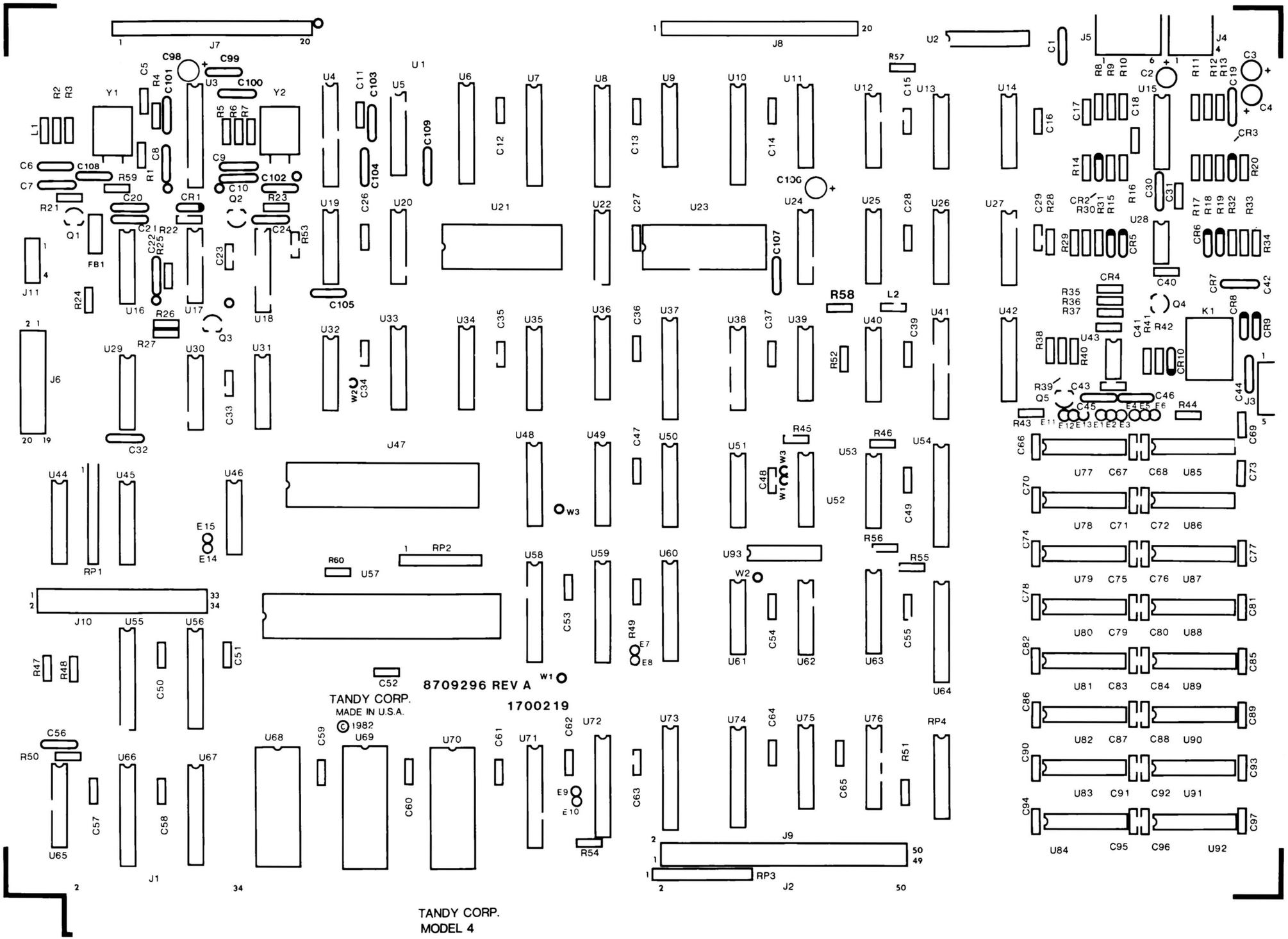


DETAIL "A"
FROM PAGE 4.

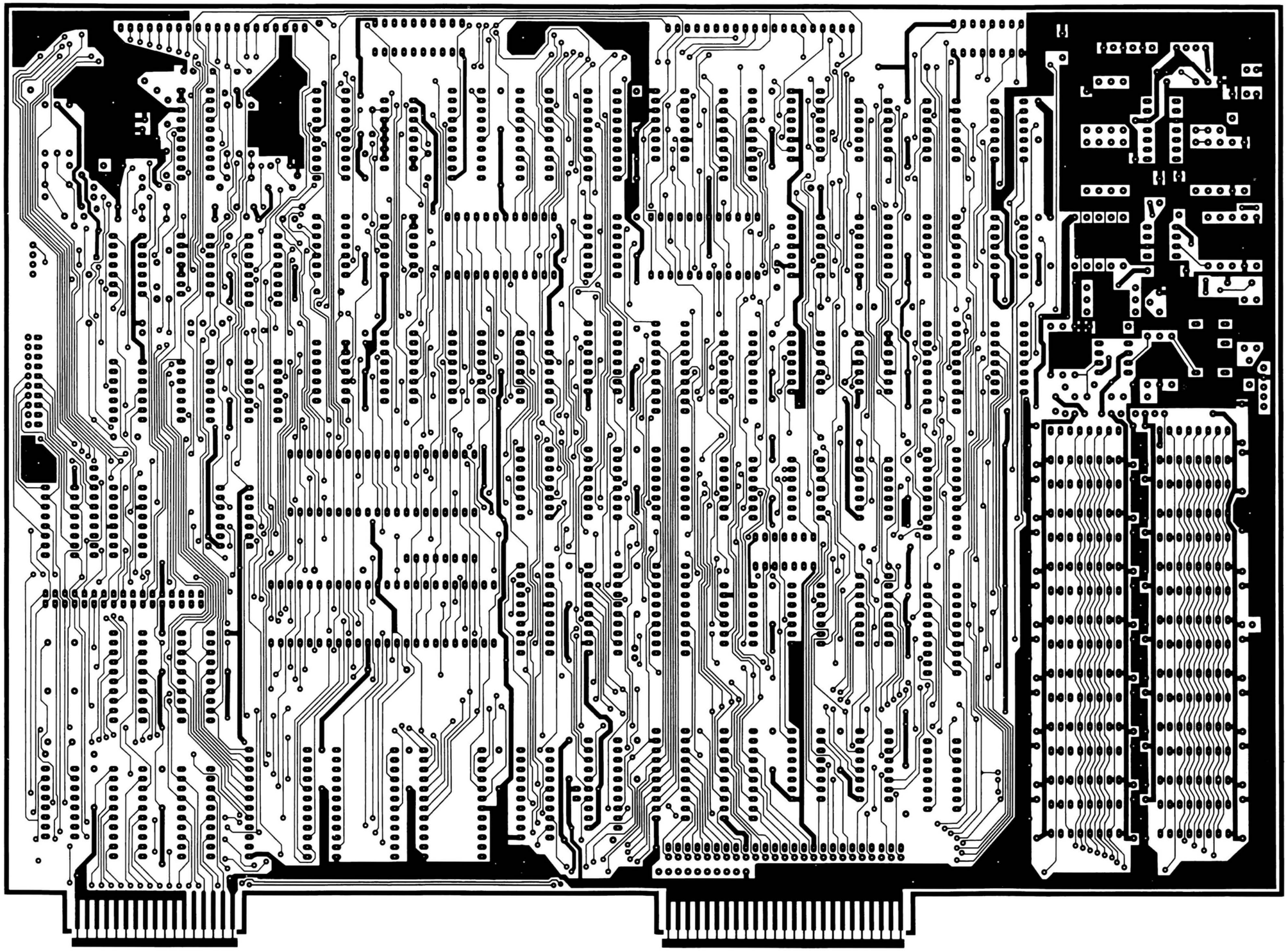
KEYBOARD DETAIL

THE ABOVE KEYS (THE NUMERIC KEYPAD) ARE CONNECTED IN PARALLEL WITH SIMILAR KEYS OF THE KEYBOARD MATRIX.

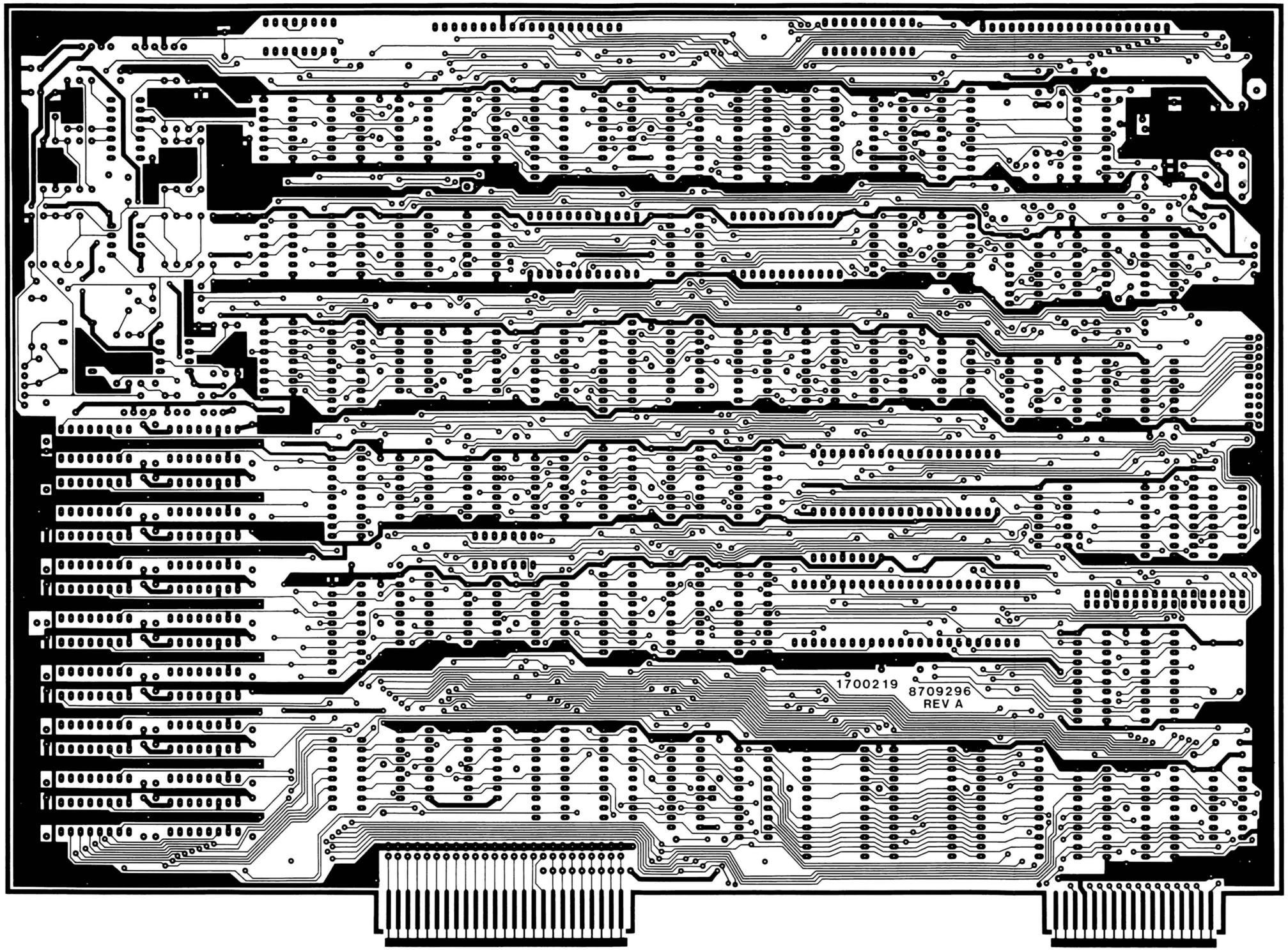
DWG NO	REV
8000173	A
SCALE NONE	SHEET 6 OF 6



COMPONENT LAYOUT, CPU PCB #8858090



CIRCUIT TRACE, CPU PCB #8858090, COMPONENT SIDE



CIRCUIT TRACE, CPU PCB #8858090, SOLDER SIDE

Parts List, CPU PCB #6700104AA3
Model 4 16K, Cassette Input
Catalog number 26-1067

Item	Qty	Description	Mfgr's Part No.
1	4	Cap, 100 pfd 50V C. Disk (C1,99,101, 103)	8301104
2	3	Cap, 10 ufd 35V Elec. Rad (C2-4)	8326103
3	74	Cap, 0.1 ufd 50V Mono Axial (C5,11-18,23,26-29,31,33-37,39-41, 43,47-55,57-97)	8374104
4	1	Cap, 56 pfd 50V C. Disk NPO (C6)	8300563
5	2	Cap, 47 pfd 50V C. Disk NPO (C7,9)	8300472
6	2	Cap, 56 pfd 50V C. Disk (C8,24)	8300564
7	1	Cap, 100 pfd 50V C. Disk (C10)	8301103
8	1	Cap, .01 ufd 50V C. Disk (C19)	8303104
9	1	Cap, 22 ufd 16V Elec. Rad(C20)	8326221
10	4	Cap, .022 ufd 50V C. Disk (C21,32, 44,108)	8303224
11	1	Cap, 33 pfd 50V C. Disk (C22)	8300334
12	1	Cap, .0047 ufd 50V C. Disk (C30)	8302474
13	1	Cap, .0022 ufd 50V C. Disk (C42)	8302224
14	2	Cap, .001 ufd 50V Mono Axial (C45,46)	8372104
15	1	Cap, 200 pfd 50V (C56)	8301204
16	1	Cap, 10 ufd 10V Tant. (C98)	8336101
17	5	Cap, 150 pfd 50V C. Disk (C100,102 104,107,109)	8301154
18	1	Cap, 100 ufd 10V Tant. (C106)	8337101
19	2	Connector, 4-Pin Right Angle (J4,J11)	8519079
20	1	Connector, 5-Pin Right Angle (J3)	8519091
21	1	Connector, 6-Pin Right Angle (J5)	8519103
22	1	Connector, Dual 10 Rt. Angle Header (J6)	8519107
23	2	Connector, 20-Pin Flat Flex Cable (J7,8)	8519101
24	1	Connector, 17-Pin Dual Header (J10)	8519169
25	8	Diode, 1N4148 Zener 75V (CR1-7,10)	8150148
26	1	Ferrite Bead (FB1)	8419014
27	1	Inductor, 47uH (L1)	8419028
28	1	Inductor, .68uH (L2)	8419029
29	1	IC, 74LS175 Quad Flip-Flop (U2)	8020175
30	1	IC, PAL16R6A (U3)	8075166
31	1	IC, PAL16L8 (U4)	8075268
32	1	IC, 74LS161 Binary Counter (U5)	8020161

Parts List, CPU PCB #6700104AA3
Model 4 16K, Cassette Input
Catalog number 26-1067

Item	Qty	Description	Mfgr's Part No.
33	8	IC, 74LS244 Quad Tranceiver (U6,10, 41,55,56,64,67,74)	8020244
34	1	IC, 74LS373 Octal Latch (U7)	8020373
35	2	IC, 74LS273 Octal Flip-Flop (U8,37)	8020273
36	1	IC, 74LS153 Dual Multiplexer (U9)	8020153
37	1	IC, 74LS166 Shift Register (U11)	8020166
38	2	IC, 74LS51 AND OR Invert (U12,20)	8020051
39	2	IC, 74LS00 Quad 2-IN NAND (U13,14)	8020000
40	1	IC, LM339 Comparator (U15)	8050339
41	3	IC, 74LS32 Quad 2-IN OR (U16,22,51)	8020032
42	1	IC, 74LS14 Hex Inverter (U17)	8020014
43	3	IC, 74LS174 Flip-Flop (U18,36,42)	8020174
44	1	IC, 74SO4 (U19)	8010004
45	1	IC, 4016 200NS RAM 2K X 8 Static (U21)	8040116
46	1	IC, MCM68A316E Character Generator (U23)	8044316
47	1	IC, 74LS86 Quad 2-IN OR (U24)	8020086
48	1	IC, 74LS04 Hex Inverter (U25)	8020004
49	3	IC, 74LS08 Quad 2-IN AND (U26,46,52)	8020008
50	5	IC, 74LS74 Dual Flip-Flop (U27,31,39 40,53)	8020074
51	1	IC, MC1458 OP-AMP (U28)	8050458
52	2	IC, 7406 Hex Inverter (U29,30)	8000006
53	1	IC, 74LS10 Triple 3-IN NAND (U32)	8020010
54	3	IC, 74LS157 Quad Multiplexer (U33-35)	8020157
55	2	IC, 74LS367 Memory (U38,75)	8020367
56	1	IC, MC1741 OP-AMP (U43)	8050741
57	2	IC, MC14502 B CMOS Driver (U44,45)	8030502
58	1	IC, SY68045 CTC 60HZ (U47)	8040045
59	3	IC, 74LS138 Decoder (U48-50)	8020138
60	2	IC, 74LS240 Octal Buffer (U54,60)	8020240
61	1	IC, Z80A CPU (U57)	8047880
62	1	IC, PAL10L8 (U58)	8075208
63	1	IC, PAL16L8 (U59)	8075368
64	1	IC, 7405 O.C. Buffer (U61)	8000005
65	1	IC, 74LS02 Quad 2-IN NOR (U62)	8020002
66	2	IC, 74157 Quad Multiplexer (U63,76)	8000157
67	1	IC, 74LS123 Dual Multivibrator (U65)	8020123
68	1	IC, 74LS374 Octal Flip-Flop (U66)	8020374
69	1	IC, MCM68A364 ROM A (U68)	8041364
70	1	IC, MCM68A332 ROM B (U69)	8040332
71	1	IC, MCM68A316 ROM C (U70)	8048316
72	2	IC, 74LS245 Octal Tranceiver (U71,73)	8020245

Parts List, CPU PCB #6700104AA3
Model 4 16K, Cassette Input
Catalog number 26-1067

Item	Qty	Description	Mfgr's Part No.
73	1	IC, DIP Shunt 4-POS. (U72)	8489057
74	8	IC, MCM4116 16K RAM 200NS (U77-84)	8042016
75	1	IC, 74LS30 Positive NAND (U93)	8020030
76	1	Relay, 12V 2 AMP (K1)	8429105
77	2	Res, 510 ohm, 5% 1/4W (R1,59)	8207151
78	1	Res, 12K ohm, 5% 1/4W (R2)	8207312
79	1	Res, 6.2K ohm, 1/4W (R3)	8207262
80	2	Res, 470 ohm, 5% 1/4W (R4,23)	8207147
81	4	Res, 10K ohm, 5% 1/4W (R5,9,14,16)	8207310
82	1	Res, 3.6K ohm, 5% 1/4W (R6)	8207236
83	1	Res, 91 ohm, 5% (R7)	8207091
84	13	Res, 4.7K ohm, 5% 1/4W (R8,28,36, 43-49,52,54,60)	8207247
85	1	Res, 620K ohm, 5% 1/4W (R10)	8207462
86	1	Res, 1.5M ohm, 5% 1/4W (R11)	8207515
87	2	Res, 56K ohm, 5% 1/4W (R12,17)	8207356
88	2	Res, 15K ohm, 5% 1/4W (R13)	8207315
89	1	Res, 51K ohm, 5% 1/4W (R15)	8207351
90	1	Res, 6.8K ohm, 5% 1/4W (R18)	8207268
91	1	Res, 8.2K ohm, 5% 1/4W (R19)	8207282
92	3	Res, 220 ohm, 5% 1/4W (R20,27,37)	8207122
93	1	Res, 680 ohm, 5% 1/4W (R21)	8207168
94	7	Res, 100K ohm, 5% 1/4W (R22,30-33, 40,42)	8207410
95	1	Res, 750 ohm, 5% 1/4W (R24)	8207175
96	2	Res, 1.2K ohm, 5% 1/4W (R25,34)	8207212
97	1	Res, 22 ohm, 5% 1/4W (R26)	8207022
98	1	Res, 220K ohm, 5% 1/4W (R29)	8207422
99	1	Res, 7.5K ohm, 5% 1/4W (R35)	8207275
100	1	Res, 82K ohm, 5% 1/4W (R38)	8207382
101	1	Res, 39K ohm, 5% 1/4W (R39)	8207339
102	1	Res, 75K ohm, 5% 1/4W (R41)	8207375
103	1	Res, 20K ohm, 5% 1/4W (R50)	8207320
104	4	Res, 150 ohm, 5% 1/4W (R51,53,55,56)	8207150
105	1	Res, 1K ohm, 5% 1/4W (R57)	8207210
106	1	Res, 56 ohm, 5% 1/4W (R58)	8207056
107	1	Res Pak, 820 ohm, SIP 10-PIN (RP1)	8290182
108	1	Res Pak, 4.7K ohm, SIP 8-PIN (RP2)	8292246
109	1	Res Pak, 27 ohm, DIP 16-PIN (RP4)	8290027

Parts List, CPU PCB #6700104AA3
Model 4 16K, Cassette Input
Catalog number 26-1067

Item	Qty	Description	Mfgr's Part No.
110	1	Transistor, 2N918 (Q1)	8110918
111	2	Transistor, 2N3906 PNP (Q2,3)	8100906
112	1	Transistor, 2N2222 (Q4)	8110222
		MISCELLANEOUS	
113	1	Crystal, 20.2752 MHz (Y1)	8409031
114	1	Crystal, 12.672 MHz (Y2)	8409030
115	3	Jumper Wire 20 Gauge (W1-3)	*NOTE
116	1	PCB Logic Board, Rev. PP3	8709296
117	1	Regulator, 79LO5, -5V (Q5)	8051905
118	7	Socket, 20-Pin DIP (U3,4,58,59,71-73)	8509009
119	5	Socket, 24-Pin DIP (U21,23,68-70)	8509001
120	2	Socket, 40-Pin DIP (U47,57)	8509002
121	16	Socket, 16-Pin DIP (U77-84,85-92)	8509003
122	13	Staking Pin (E1-8,11-15)	8529014

Note: W1,W3 are 4-1/2" long, W2 is 6" long

Parts List, CPU PCB 8858090
Model 4 64K, Single or Double Drive
Catalog Number 26-1068 or 26-1069

Item	Qty	Description	Mfgr's Part No.
1	4	Cap, 100 PFD 50V C. Disk (C1,99,101, 103)	8301104
2	3	Cap, 10 MFD 35V ELEC. RAD (C2-4)	8326103
3	58	Cap, 0.1MFD 50V MONO AXIAL (C5,11-18,23,26-29,31,33-37,39-41, 43,47-55,57-65,67,69,71,73,75,77,79 81,83,85,87,89,91,93,95,97)	8374104
4	1	Cap, 56 PFD 50V C. DISK NPO (C6)	8300563
5	2	Cap, 47 PFD 50V C. DISK NPO (C7,9)	8300472
6	2	Cap, 56 PFD 50V C. DISK (C8,24)	8300564
7	1	Cap, 100 PFD 50V C. DISK (C10)	8301103
8	1	Cap, .01 MFD 50V C. DISK (C19)	8303104
9	1	Cap, 22 MFD 16V ELEC. RAD(C20)	8326221
10	4	Cap, .022 MFD 50V C. DISK (C21,32, 44,108)	8303224
11	1	Cap, 33 PFD 50V C. DISK (C22)	8300334
12	1	Cap, .0047 MFD 50V C. DISK (C30)	8302474
13	1	Cap, .0022 MFD 50V C. DISK (C42)	8302224
14	2	Cap, .001 MFD 50V MONO AXIAL (C45,46)	8372104
15	1	Cap, 200 PFD 50V (C56)	8301204
16	1	Cap, 10 MFD 10V TANT. (C98)	8336101
17	5	Cap, 150 PFD 50V C. DISK (C100,102 104,107,109)	8301154
18	1	Cap, 100 MFD 10V TANT. (C106)	8337101
19	2	Connector, 4-Pin Right Angle (J4,J11)	8519079
20	1	Connector, 5-Pin Right Angle (J3)	8519091
21	1	Connector, 6-Pin Right Angle (J5)	8519103
22	1	Connector, Dual 10 Rt. Angl. Header (J6)	8519107
23	2	Connector, 20-Pin Flat Flex Cable (J7,8)	8519101
24	1	Connector, 17-Pin Dual Header (J10)	8519169
25	8	Diode, 1N4148 Zener 75V (CR1-7,10)	8150148
26	1	Ferrite Bead (FB1)	8419014
27	1	Inductor, 47uH (L1)	8419028
28	1	Inductor, .68uH (L2)	8419029
29	1	IC, 74LS175 Quad Flip-Flop (U2)	8020175
30	1	IC, PAL16R6A (U3)	8075166
31	1	IC, PAL16L8 (U4)	8075268

Parts List, CPU PCB 8858090
Model 4 64K, Single or Double Drive
Catalog number 26-1068 or 26-1069

Item	Qty	Description	Mfgr's Part No.
32	1	IC, 74LS161 Binary Counter (U5)	8020161
33	8	IC, 74LS244 Quad Tranceiver (U6,10, 41,55,56,64,67,74)	8020244
34	1	IC, 74LS373 Octal Latch (U7)	8020373
35	2	IC, 74LS273 Octal Flip-Flop (U8,37)	8020273
36	1	IC, 74LS153 Dual Multiplexer (U9)	8020153
37	1	IC, 74LS166 Shift Register (U11)	8020166
38	2	IC, 74LS51 AND OR Invert (U12,20)	8020051
39	2	IC, 74LS00 Quad 2-In NAND (U13,14)	8020000
40	1	IC, LM339 Comparator (U15)	8050339
41	3	IC, 74LS32 Quad 2-In OR (U16,22,51)	8020032
42	1	IC, 74LS14 Hex Inverter (U17)	8020014
43	3	IC, 74LS174 Flip-Flop (U18,36,42)	8020174
44	1	IC, 74SO4 (U19)	8010004
45	1	IC, 4016 200NS RAM 2K X 8 Static (U21)	8040116
46	1	IC, MCM68A316E Character Generator (U23)	8044316
47	1	IC, 74LS86 Quad 2-In OR (U24)	8020086
48	1	IC, 74LS Hex Inverter (U25)	8020004
49	3	IC, 74LS08 Quad 2-In AND (U26,46,52)	8020008
50	5	IC, 74LS74 Dual Flip-Flop (U27,31,39 40,53)	8020074
51	1	IC, MC1458 OP-AMP (U28)	8050458
52	2	IC, 7406 Hex Inverter (U29,30)	8000006
53	1	IC, 74LS10 Triple 3-In NAND (U32)	8020010
54	3	IC, 74LS157 Quad Multiplexer (U33-35)	8020157
55	2	IC, 74LS367 Memory (U38,75)	8020367
56	1	IC, MC1741 OP-AMP (U43)	8050741
57	2	IC, MC14502 B CMOS Driver (U44,45)	8030502
58	1	IC, SY68045 CTC 50Hz Version (U47)	8041045
59	3	IC, 74LS138 Decoder (U48-50)	8020138
60	2	IC, 74LS240 Octal Buffer (U54,60)	8020240
61	1	IC, Z80A CPU (U57)	8047880
62	1	IC, PAL10L8 (U58)	8075208
63	1	IC, PAL16L8 (U59)	8075368
64	1	IC, 7405 O.C. Buffer (U61)	8000005
65	1	IC, 74LS02 Quad 2-In NOR (U62)	8020002
66	2	IC, 74157 Quad Multiplexer (U63,76)	8000157
67	1	IC, 74LS123 Dual Multivibrator (U65)	8020123
68	1	IC, 74LS374 Octal Flip-Flop (U66)	8020374
69	1	IC, MCM68A364 ROM A (U68)	8048364
70	1	IC, MCM68A332 ROM B (U69)	8040332
71	1	IC, MCM68A316 ROM C (U70)	8042316

Parts List CPU PCB 8858090
Model 4 64K, Single or Double Drive
Catalog number 26-1068 and 26-1069

Item	Qty	Description	Mfgr's Part No.
72	2	IC, 74LS245 Octal Tranceiver (U71,73)	8020245
73	1	IC, DIP Shunt 4-Pos. (U72)	8489057
74	8	IC, MCM6665 64K RAM 200NS (U85-92)	8040665
75	1	IC, 74LS30 Positive NAND (U93)	8020030
76	1	Relay, 12V 2 Amp (K1)	8429105
77	2	Res, 510 ohm, 5% 1/4W (R1,59)	8207151
78	1	Res, 12K ohm, 5% 1/4W (R2)	8207312
79	1	Res, 6.2K ohm, 1/4W (R3)	8207262
80	2	Res, 470 ohm, 5% 1/4W (R4,23)	8207147
81	4	Res, 10K ohm, 5% 1/4W (R5,9,14,16)	8207310
82	1	Res, 3.6K ohm, 5% 1/4W (R6)	8207236
83	1	Res, 91 ohm, 5% (R7)	8207091
84	13	Res, 4.7K ohm, 5% 1/4W (R8,28,36, 43-49,52,54,60)	8207247
85	1	Res, 620K ohm, 5% 1/4W (R10)	8207462
86	1	Res, 1.5Meg ohm, 5% 1/4W (R11)	8207515
87	2	Res, 56K ohm, 5% 1/4W (R12,17)	8207356
88	2	Res, 15K ohm, 5% 1/4W (R13)	8207315
89	1	Res, 51K ohm, 5% 1/4W (R15)	8207351
90	1	Res, 6.8K ohm, 5% 1/4W (R18)	8207268
91	1	Res, 8.2K ohm, 5% 1/4W (R19)	8207282
92	3	Res, 220 ohm, 5% 1/4W (R20,27,37)	8207122
93	1	Res, 680 ohm, 5% 1/4W (R21)	8207168
94	7	Res, 100K ohm, 5% 1/4W (R22,30-33, 40,42)	8207410
95	1	Res, 750 ohm, 5% 1/4W (R24)	8207175
96	2	Res, 1.2K ohm, 5% 1/4W (R25,34)	8207212
97	1	Res, 22 ohm, 5% 1/4W (R26)	8207022
98	1	Res, 220K ohm, 5% 1/4W (R29)	8207422
99	1	Res, 7.5K ohm, 5% 1/4W (R35)	8207275
100	1	Res, 82K ohm, 5% 1/4W (R38)	8207382
101	1	Res, 39K ohm, 5% 1/4W (R39)	8207339
102	1	Res, 75K ohm, 5% 1/4W (R41)	8207375
103	1	Res, 20K ohm, 5% 1/4W (R50)	8207320
104	4	Res, 150 ohm, 5% 1/4W (R51,53,55,56)	8207150
105	1	Res, 1K ohm, 5% 1/4W (R57)	8207210
106	1	Res, 56 ohm, 5% 1/4W (R58)	8207056
107	1	Res Pak, 820 ohm, SIP 10-Pin (RP1)	8290182
108	1	Res Pak, 4.7K ohm, SIP 8-Pin (RP2)	8292246
109	1	Res Pak, 27 ohm, DIP 16-Pin (RP4)	8290027

Parts List CPU PCB 8858090
Model 4 64K, Single or Double Drive
Catalog number 26-1068 and 26-1069

Item	Qty	Description	Mfgr's Part No.
110	1	Transistor, 2N918 (Q1)	8110918
111	2	Transistor, 2N3906 PNP (Q2,3)	8100906
112	1	Transistor, 2N2222 (Q4)	8110222
		MISCELLANEOUS	
113	1	Crystal, 20.2752 MHz (Y1)	8409031
114	1	Crystal, 12.672 MHz (Y2)	8409030
115	3	Jumper Wire, 20 Gauge (W1-3)	*NOTE
116	1	PCB, Logic Board Rev. PP3	8709296
117	7	Socket, 20-Pin DIP (U3,4,58,59,71-73)	8509009
118	5	Socket, 24-Pin DIP (U21,23,68-70)	8509001
119	2	Socket, 40-Pin DIP (U47,57)	8509002
120	16	Socket, 16-Pin DIP (U77-84,85,-92)	8509003
121	10	Staking Pin (E1-8,11-15)	8529014

SECTION IV

FLOPPY DISK INTERFACE

FLOPPY DISK INTERFACE

4.1 MODEL 4 FDC PCB #8858060

The TRS-80 Model 4 Floppy Disk interface Board is an optional board which if incorporated provides a standard five inch floppy disk controller. The Floppy Disk Interface Board supports both single and double density encoding schemes. This feature, along with a special software package, allows the transfer of Model I disk files to the Model 4 system. Write precompensation can be software enabled or disabled beginning at any track, although the system software enables write precompensation for all tracks greater than twenty-one. The amount of write precompensation is continuously variable from Ø nsec to more than 500 nsec. The write precompensation is factory adjusted to 200 nsec. The data clock recovery logic incorporates a phaselocked loop oscillator which achieves state-of-the-art reliability. One to four drives may be controlled by the interface (two internal drives and two external). All data transfers are accomplished by CPU data requests. In double density operation, data transfers are synchronized to the CPU by forcing a wait to the CPU and clearing the wait by a data request from the FDC chip. The end of the data transfer is indicated by generating a non-maskable interrupt from the interrupt request output of the FDC chip. A hardware watchdog timer insures that error conditions will not hang the wait line to the CPU for a period long enough to destroy RAM contents.

4.1.1 Control and Data Buffering

Refer to the Schematic Diagram 8000095.

The Floppy Disk Controller Board is an I/O port mapped device which utilizes ports E4H, F0H, F1H, F2H, F3H, and F4H. The decoding logic is implemented on the CPU board. (See the Decoding Logic section of the CPU discussion.) U4 of the Floppy Disk Controller Board is a non-inverting octal buffer which isolates and buffers the required control signals. Table 4-1 and Table 4-2 summarize the port and bit allocation for the Floppy Controller Board. U2 of the Floppy Disk Controller Board is a bi-directional, 8-bit transceiver used to buffer data to and from the Floppy Controller Board. The direction of data transfer is controlled by the combination of control signals DISKIN* and RDNMIMASKREG*. If either signal is active (logic low), U2 is enabled to drive data onto the CPU board data bus. If both signals are inactive (logic high), U2 is enabled to receive data from the CPU data bus.

4.1.2 Nonmaskable Interrupt Logic

A dual "D" flip-flop (U5) is used to latch data bits D6 and D7 on the rising edge of the control signal WRNMIMASREG*. The outputs of U5 control the conditions which will generate a non-maskable interrupt to the CPU. The NMI interrupt conditions are programmed by doing an OUT instruction to port E4H with the appropriate bits set. If data bit 7 is set, an NMI will be generated by an FDC interrupt request. If data bit 7 is reset, interrupt requests from the FDC are disabled. If data bit 6 is set, an NMI will be generated by Motor Time Out. If data bit 6 is reset, interrupts on Motor Time Out are disabled. An IN instruction from port E4H enables the CPU to question the Floppy Disk Controller Board to determine the source of the non-maskable interrupt. Data bit 7 indicates the status of FDC interrupt request (0 = true, 1 = false). Data bit 6 indicates the status of Motor Time Out (0 = true, 1 = false). Data bit 5 indicates the status of the front panel reset (0 = true, 1 = false). The control signal RDNMIMASKREG* when active (logic 0), gates this status onto the CPU data bus.

4.1.3 Drive Select Latch and Motor On Logic

Selecting a drive prior to a disk I/O operation is accomplished by doing an OUT instruction to port F4H with the proper bit set. The following table describes the bit allocation of the Drive Select Latch.

DATA BIT	FUNCTION
D0	Selects Drive 0 when set *
D1	Selects Drive 1 when set *
D2	Selects Drive 2 when set *
D3	Selects Drive 3 when set *
D4	Side 0 selected when reset, Side 1 selected if set
D5	Write Precom. engaged when set, disabled if reset
D6	Generate waits if set, no waits if reset
D7	Selects MFM mode if set, FM mode if reset

*Only one of these bits should be set per output.

Table 4.1. Port F4H Bit Allocation

A hex "D" flip-flop (U6) latches the drive select bits, side select and FM*/MFM bits on the rising edge of the control signal IDRVSEL*. A dual "D" flip-flop (U18) is used to latch the Wait Enable and Precompensation enable bits on the rising edge of IDRVSEL*. The rising edge of IDRVSEL* also triggers a one-shot (1/2 of U15) which produces a Motor On to the disk drives. The duration of the Motor On signal is approximately two seconds. The spindle motors are not designed for continuous operation, therefore the inactive state of the Motor On signal is used to clear the Drive Select Latch, which de-selects any drives which were previously selected. The Motor On one-shot is retriggerable by simply executing an OUT instruction to the Drive Select Latch.

4.1.4 Wait State Generation and WAITIMOUT Logic

As previously mentioned, a wait state to the CPU can be initiated by an output to the Drive Select Latch with D6 set. Pin 5 of U18 will go high after this operation. This signal is inverted by 1/6 of U1 and is routed to the CPU board where it forces the Z-80 into a wait state. The Z-80 will remain in the wait state as long as WAIT* is low. Once initiated, the wait state will remain until one of four conditions are satisfied. One half of U10 (a five input NOR gate) is used to perform this function. INTRQ, DRQ, RESET, and WAITIMOUT are the inputs to the NOR gate. If any one of these inputs are active (logic high), the output of the NOR gate (U10 pin 6) will go low. This output is tied to the clear input of the wait latch. This signal, when low, will clear the Q output (U18 pin 5) and set the Q* output (U18 pin 6). This condition causes WAIT* to go high and allows the Z-80 to exit the wait state. U20 is a 12-bit binary counter which serves as a watchdog timer to insure that a wait condition will not persist long enough to destroy dynamic RAM contents. The counter is clocked by a 1MHz signal and is enabled to count when its reset pin is low (U20 pin 11). A logic high on U20 pin 11 resets the counter ouputs. U20 pin 15 is the divide by 1024 output and is used to generate the signal WAITIMOUT. This watchdog timer logic will limit the duration of a wait to 1024μsec, even if the FDC chip fails to generate a data request or an interrupt request.

4.1.5 Clock Generation Logic

A 4MHz crystal oscillator and a divide by 2 and divide by 4 counter generate the clock signals required by the FDC board. The basic 4MHz oscillator is implemented with two invertors (1/3 of U25) and a quartz crystal (Y1). One half of U24 is used to divide the basic 4MHz clock by 2 to produce a 2MHz output at U24 pin 6. This output is again divided by 2 using the remaining half of U24 to produce a 1MHz output at U24 pin 8. The 1MHz clock is used to drive the clock input of the 1793 FDC chip and the clock input of the watchdog time (U20).

4.1.6 Disk Bus Selector Logic

As mentioned previously, the Model 4 Floppy Disk Board supports up to four drives (two internal, two external). This function is implemented by using two disk drive interface buses, one for the internal drives and one for the external drives. J4 is the edge connector used for the internal drives and J1 is the edge connector for the external drives. U22 (a quad 2 to 1 data selector) is used to select which set of inputs from the disk drive buses are routed to the 1793 FDC chip. U22 pin 1 is the control pin for the data selector. If U22 pin 1 is low, the external inputs are selected, otherwise the internal inputs are selected. This control signal (labeled EXTSEL*) is derived from the outputs of the Drive Select Latch. If Drive 2 or Drive 3 is selected, U17 pin 1 will go low indicating that an external drive is selected. One half of U10 (a five input NOR gate) is used to detect when one of the four drives is selected. The output of this NOR gate (U10 pin 5) is inverted and is used as the head load timing and ready signal for the 1793 FDC chip. Therefore if any drive is selected, the head is assumed to be loaded and the selected drive is assumed to be ready.

4.1.7 Read/Write Data Pulse Shaping Logic

Two one-shots (1/2 of U15 and 1/2 of U23) are used to insure that the read and write data pulses are approximately 450nsec in duration.

4.1.8 Disk Bus Output Drivers

High current open collector drivers (U21, U9, and U1) are used to buffer the output signals from the Drive Select Latch and the FDC chip to the floppy disk drives. Note from the schematic that each output signal to the drives has two buffers associated with each signal, one set is used for the internal drive bus and the other set is used for the external bus. No select logic is required for these output signals since the drive select bits define which drive is active.

4.1.9 Write Precompensation and Clock Recovery Logic

The Write Precompensation and Read Clock Recovery logic is comprised of U11 (WD1691), U13 (WD2143) and U14 (LS629), along with a few passive components. The WD1691 is an LSI device which minimizes the external logic required to interface the 1793 FDC chip to a disk drive. With the use of an external VCO, U14, the WD1691 will derive the RCLK signal for the 1793, while providing an adjustment signal for the VCO, to keep the RCLK synchronous with the read data from the drive. Write precompensation control signals are also provided by the WD1691 to interface directly to the WD2143 (U13) clock generator. The Read Clock Recovery section of the WD1691 has five inputs: DDEN*, VCO, RDD*, WG, and VFOE*/WF. It also has three outputs: PU, PD*, and RCLK. The inputs VFOE*/WF and WG when both are low, enable the Clock Recovery logic. When WG is high, a write operation is in progress and the Clock Recovery circuits are disabled regardless of the state of any other inputs.

The Write Precompensation section of the WD1691 was designed to be used with the WD2143 clock generator. Write Precompensation is not used in single density mode and the signal DDEN* when high indicates this condition. In double density mode (DDEN* = 0), the signals EARLY and LATE are used to select a phase input (01 - 04) on the leading edge of WDIN. The STB line is latched high when this occurs, causing the WD2143 to start its pulse generation.

02 is used as the write data pulse on nominal (EARLY = 0 LATE = 0), 01 is used for the early, and 03 is used for the late. The leading edge of 04 resets the STB line in anticipation of the next write data pulse. When TG43 = 0 or DDEN* = 1, precompensation is disabled and any transitions on the WDIN line will appear on the WDOUT line.

When VFOE*/WF and WG are low, the Clock Recovery circuits are enabled. When the RDD* line goes low, the PU or PD* signals will become active. If the RDD* has made its transition in the beginning of the RCLK window, PU will go from a high impedence state to a logic one, requesting an increase in VCO frequency. If the RDD* line has made it transition at the end of the RCLK window, PU will remain in the high impedence state while PD* will go to a logic zero, requesting a decrease in the VCO frequency. When the leading edge of RDD* occurs in the center of the RCLK window, both PU and PD* will remain in the high impedence state, indicating that no adjustment of the VCO frequency is required. By tying PU and PD* together, an adjustment signal is created which will be forced low for a decrease in VCO frequency and forced high for an increase in VCO frequency. To speed up rise times and stabilize the output voltage, a resistor divider using R7, R10, and R9 is used to adjust the tri-state level at approximately 1.4V. This adjustment results in a worst case voltage swing of plus or minus 1V, which is acceptable for the frequency control input of the VCO (U14). This signal derived from the combination of PU and PD* will eventually correct the VCO input to exactly the same frequency multiple as the FDD* signal. The leading edge of the RDD* signal will then occur in the exact center of the RCLK window, an ideal condition for the 1793 internal recovery circuits.

4.1.10 Floppy Disk Controller Chip

The 1793 is an MOS LSI device which performs the functions of a floppy disk formatter/controller in a single chip implementation. The 1793 is functionally identical to the 1791 used on the Model II FDC Printer Interface Board, except that the data bus is true as opposed to inverted. Refer to the appendix section for more information on the FD1793. The Model II Technical Reference Manual also contains a good presentation of the 1791 FDC chip as well as a discussion on Write Precompensation. The following port addresses are assigned to the internal registers of the 1793 FDC chip.

PORT #	FUNCTION
F0H	Command/Status Register
F1H	Track Register
F2H	Sector Register
F3H	Data Register

Table 4.2 Port Allocation

4.1.11 Adjustments and Jumper Options

The Data Separator must be adjusted with the 1793 in an idle condition (no command currently in operation). Adjust R7 potentiometer for a 1.4V level on pin 2 of U14. Then adjust R6 potentiometer to yield a 2MHz square wave at pin 16 of U11.

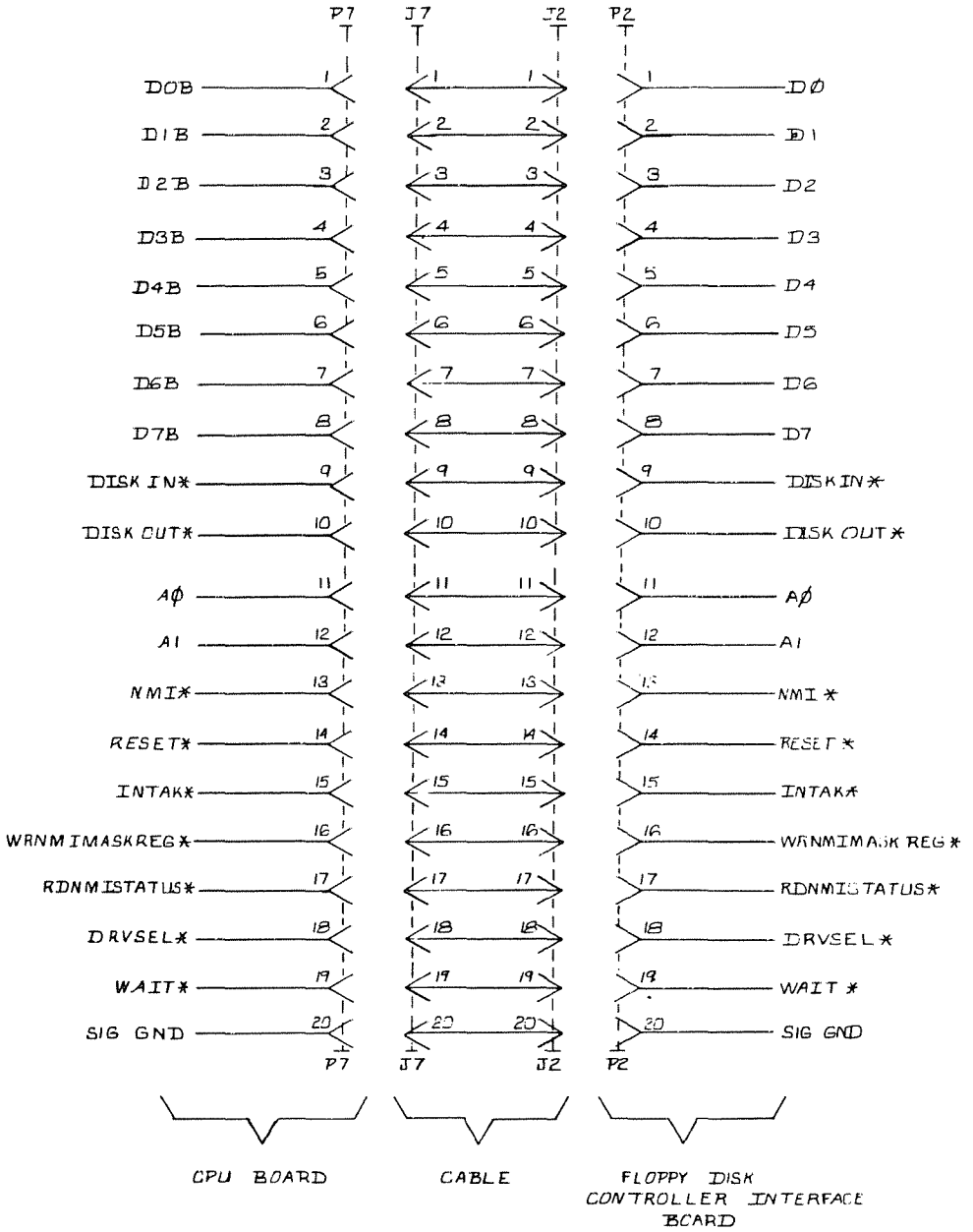
The Write Precompensation must be adjusted while executing a continuous write command on a track greater than twenty-one. Adjust R5 potentiometer to yield 200nsec wide pulses at pin 4 of U11. This results in a write precompensation value of 200nsec.

There are four jumper options on the Floppy Disk Controller Board. They are designated on the PC Board silkscreen and are referenced on the Schematic Diagram. The jumpers should be installed as described below.

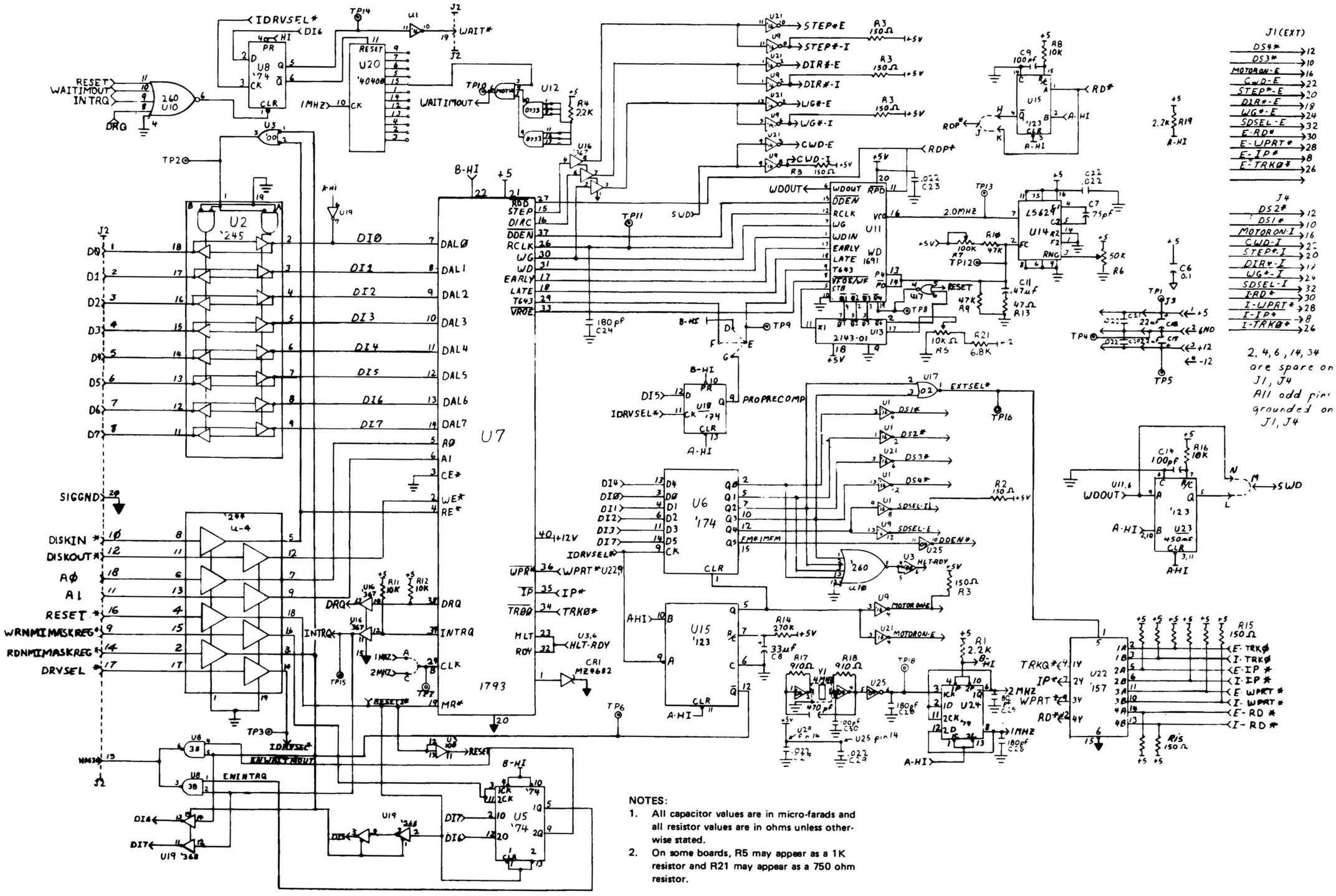
JUMPER CONNECTIONS

A to B
E to G
L to M
H to J

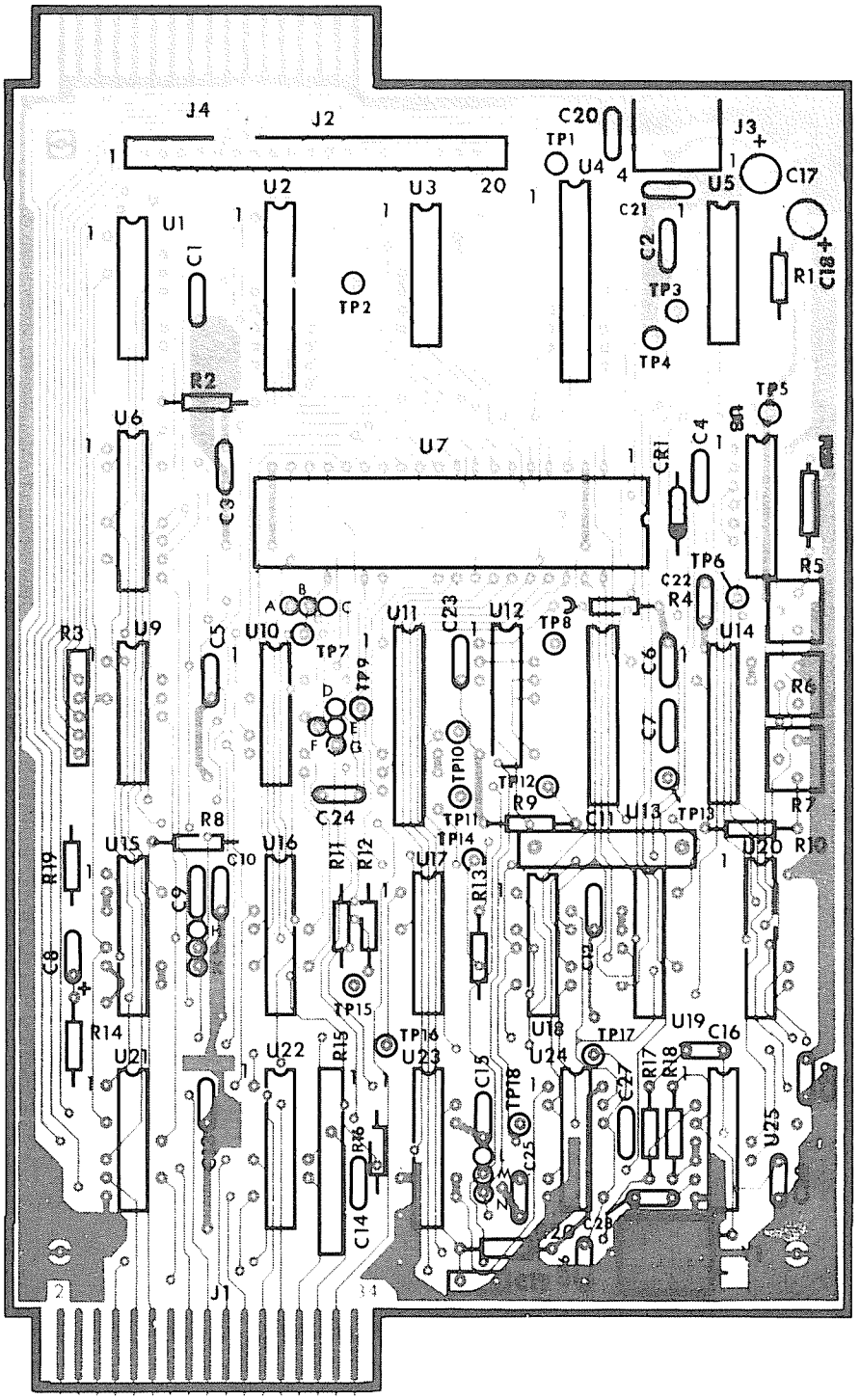
FLOPPY DISK CONTROLLER INTERFACE CONNECTOR



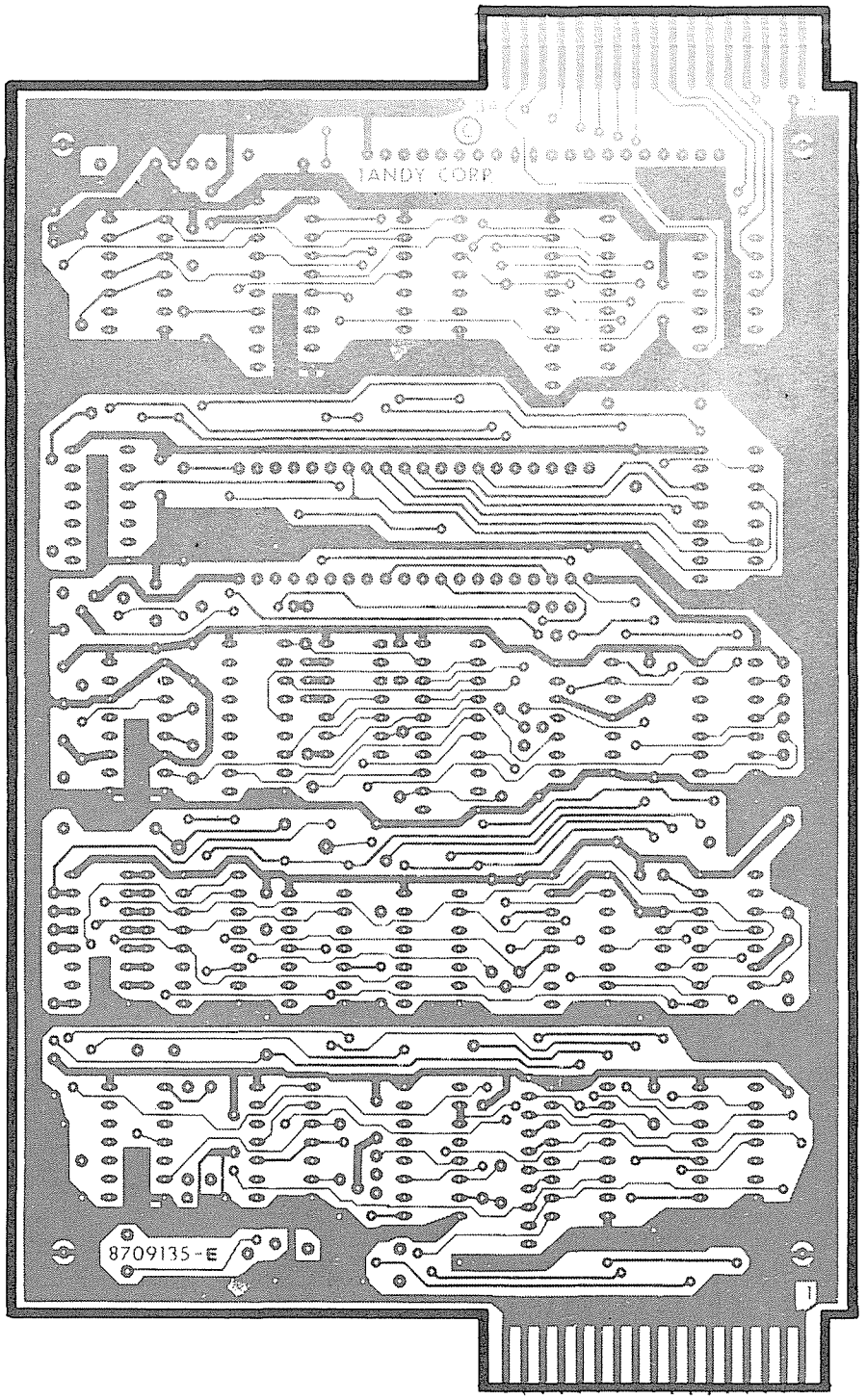
FDC BOARD TO CPU BOARD SIGNAL DESCRIPTION



SCHEMATIC 8000095E, FLOPPY DISK CONTROLLER #8858060



COMPONENT LOCATION/CIRCUIT TRACE
FLOPPY DISK INTERFACE PC BOARD – COMPONENT SIDE



CIRCUIT TRACE, FLOPPY DISK INTERFACE PC BOARD – SOLDER SIDE

PARTS LIST, FLOPPY DISK INTERFACE PC BOARD

#8858060

SYMBOL	DESCRIPTION	MANUFACTURER'S PART NUMBER	RADIO SHACK PART NUMBER
	INTEGRATED CIRCUITS (cont'd)		
U6	74LS174, Quad "D" Flip-Flop	802-0174	AMX3565
U7	WD1793	850-9002	AXX3041
U8	74LS38, NAND Buffer	802-0038	AMX4328
U9	7416, Hex Inverter/Buffer	800-0016	----------
U10	74LS260, Dual NOR gate	802-0260	----------
U11	WD1691	850-9009	AMX4471
U12	MC140733, AND gate	803-0073	----------
U13	WD2143-01	850-9006	AMX4472
U14	74LS629. VCO	802-0629	AMX4663
U15	74LS123, Mono Multivibrator	802-0123	AMX3803
U16	74LS367, Hex Buffer	802-0367	AMX3567
U17	74LS02, NAND gate	802-0002	AMX3551
U18	74LS74, Dual "D" Flip-Flop	802-0074	AMX3558
U19	74LS368, Hex Inverter/Buffer	802-0368	AMX3568
U20	MC14040B, Binary Counter	803-0040	AMX4666
U21	7416, Hex Inverter/Buffer	800-0016	----------
U22	74LS157, Quad Multiplexer	802-0157	AMX3563
U23	74LS123, Mono Multiplexer	802-0123	AMX3803
U24	74LS74, Dual "D" Flip-Flop	802-0074	AMX3558
U25	74LS04, Hex Inverter	802-0004	AMX3552
	RESISTORS		
R1	2.2K, 1/4W, 5%	820-7222	AN0216EEC
R2	150 ohm, 1/4W, 5%	820-7115	AN0142EEC
R3	150 ohm, 6 pin resistor network	829-0012	ARX0241
R4	2.2K, 1/4W, 5%	820-7222	AN0216EEC
R5	1K, Trim Pot	827-9210	AP0835
R6	50K, Trim Pot	827-9350	AP7168
R7	100K, Trim Pot	827-9410	----------
R8	10K, 1/4W, 5%	820-7310	AN0281EEC
R9	47K, 1/4W, 5%	820-7347	AN0340EEC
R10	47K, 1/4W, 5%	820-7347	AN0340EEC
R11	10K, 1/4W, 5%	820-7310	AN0281EEC
R12	10K, 1/4W, 5%	820-7310	AN0281EEC
R13	47 ohm, 1/4W, 5%	820-7047	AN0099EEC
R14	270K, 1/4W, 5%	820-7427	----------
R15	150 ohm, 10 pin resistor network	829-0013	ARX0242
R16	10K, 1/4W, 5%	820-7310	AN0281EEC
R17	910 ohm, 1/4W, 5%	820-7191	AN0192EEC
R18	910 ohm, 1/4W, 5%	820-7191	AN0192EEC
R19	2.2K, 1/4W, 5%	820-7222	AN0216EEC
R20	22K, 1/4W, 5%	820-7322	----------
	MISCELLANEOUS		
	Socket, 18 pin	850-9006	AJ6701
	Socket, 20 pin	850-9009	AJ6760
	Socket, 40 pin	850-9002	AJ6580
	FDC Board, complete assembly		AXX0510
	FDC Board, without major chips		AXX0509

PARTS LIST, FLOPPY DISK INTERFACE PC BOARD
#8858060

SYMBOL	DESCRIPTION	MANUFACTURER'S PART NUMBER	RADIO SHACK PART NUMBER
	CAPACITORS		
C1	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C2	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C3	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C4	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C5	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C6	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C7	75pF, 50V, ceramic disc	830-0754	----------
C8	33μF, 16V, electrolytic, radial	839-6331	ACC336QDAP
C9	100pF, 50V, ceramic disc	830-1104	ACC101QJCP
C10	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C11	0.47μF, 16V, mylar	835-4471	----------
C12	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C13	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C14	100pF, 50V, ceramic disc	830-1104	ACC101QJCP
C15	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C16	470pF, 50V, ceramic disc	830-1474	ACC471QJCP
C17	10μF, 16V, electrolytic, radial	832-6101	ACC106QDAP
C18	10μF, 16V, electrolytic, radial	832-6101	ACC106QDAP
C19	0.01μF, 16V, ceramic disc	830-3104	ACC103QJCP
C20	0.022μF, 50V, ceramic disc	830-3224	ACC223QJCP
↓	↓	↓	↓
C23	0.022μF, 50V, ceramic disc	830-3224	ACC223QJCP
C24	180pF, ceramic disc	830-1184	----------
C25	180pF, ceramic disc	830-1184	----------
C26	180pF, ceramic disc	830-1184	----------
C27	0.022μF, ceramic disc	830-3224	ACC223QJCP
C28	180pF, ceramic disc	830-1184	----------
C29	0.022μF, ceramic disc	830-3224	ACC223QJCP
C30	100pF, ceramic disc	830-1104	ACC101QJCP
	CONNECTORS		
J2	20 pos. right angle	851-9078	----------
J3	4 pin right angle header	851-9079	AJ6977
	CRYSTAL		
Y1	4 MHz	840-9010	AMX2804
	DIODES		
CR1	MZ4682	815-0682	ADX1518
	INTEGRATED CIRCUITS		
U1	7416, Hex Inverter/Buffer	800-0016	----------
U2	74LS245	802-0245	AMX4470
U3	74LS00, NAND gate	802-0000	AMX3550
U4	74LS244, Octal Buffer	802-0244	AMX3864
U5	74LS74, Dual "D" Flip-Flop	802-0074	AMX3558

4.2 MODEL 4 FDC PCB #8858160

The TRS-80 Model III/4 Floppy Disk Interface Board is an optional board which, if incorporated, provides a standard 5-1/4" floppy disk controller. The Floppy Disk Interface Board supports both single and double density encoding schemes. Write precompensation can be software enabled or disabled beginning at any track, although the system software enables write precompensation for all tracks greater than twenty-one. The amount of write precompensation provided is 250 nsec and is not adjustable. The data clock recovery logic incoporates a digital data separator which achieves state-of-the-art reliability. One to four drives may be controlled by the interface (two internal drives and two external). All data transfers are accomplished by CPU data requests. In double density operation, data transfers are synchronized to the CPU by forcing a wait to the CPU and clearing the wait by a data request from the FDC chip. The end of the data transfer is indicated by generation of a non-maskable interrupt from the interrupt request output of the FDC chip. A hardware watchdog timer insures that any error condition will not hang the wait line to the CPU for a period long enough to destroy RAM contents.

4.2.1 Control and Data Buffering

Refer to Schematic Diagram 8000168

The Floppy Disk Controller Board is an I/O port mapped device which utilizes ports E4H, F0H, F1H, F2H, F3H, and F4H. The decoding logic is implemented on the CPU board. (Refer to Paragraph 3.1.4 Decoding Logic of the CPU operation). U4 is a non-inverting octal buffer which isolates and buffers the required control signals from the CPU board to the FDC board. U2 is a bi-directional, 8-bit transceiver used to buffer data to and from the FDC board. The direction of data transfer is controlled by the combination of control signals DISKIN* and RDNMISTATUS*. If either signal is active (logic low), U2 is enabled to drive data onto the CPU board data bus. If both signals are inactive (logic high), U2 is enabled to receive data from the CPU board data bus.

4.2.2 Nonmaskable Interrupt Logic

A dual D flip-flop (U12) is used to latch data bits D6 and D7 on the rising edge of the control signal WRNMIMASKREG*. The outputs of U12 enable the conditions which will generate a non-maskable interrupt to the CPU. The NMI interrupt conditions are programmed by doing an OUT instruction to port E4H with the appropriate bits set. If data bit 7 is set, an FDC interrupt request is enabled to generate an NMI interrupt. If data bit 7 is reset, interrupt requests from the FDC are disabled. If data bit 6 is set, a Motor Time Out is enabled to generate a NMI interrupt. If data bit 6 is reset, interrupts on Motor Time Out are disabled. An IN instruction from port E4H enables the CPU to check the FDC board to determine the source of the non-maskable interrupt. Data bit 7 indicates the status of FDC interrupt request (0 = true, 1 = false). Data bit 6 indicates the status of Motor Time Out (0 = true, 1 = false). Data bit 5 indicates the status of the Reset signal from the CPU board (0 = true, 1 = false). The control signal RDNMISTATUS* gates this status onto the CPU data bus when active (logic low).

4.2.3 Drive Select Latch and Motor ON Logic

Selecting a drive prior to a disk I/O operation is accomplished by doing an OUT instruction to port F4H with the proper bit set. The following table described the bit allocation of the Drive Select Latch:

Data Bit	Function
D0	Selects Drive 0 when set*
D1	Selects Drive 1 when set*
D2	Selects Drive 2 when set*
D3	Selects Drive 3 when set*
D4	Selects Side 0 when reset Selects Side 1 when set
D5	Write precompensation enabled when set, disabled when reset
D6	Generates WAIT if set
D7	Selects MFM mode if set Selects FM mode if reset

*Only one of these bits should be set per output

A hex D flip-flop (U5) latches the drive select bits, side select and FM*/MFM bits on the rising edge of the control signal IDRVSEL*. A dual D flip-flop (U15) is used to latch the Wait Enable and Write precompensation enable bits on the rising edge of IDRVSEL*. The rising edge of IDRVSEL* also triggers a one-shot (1/2 of U13) which produces a Motor On to the disk drives. The duration of the Motor On signal is approximately two seconds. The spindle motors are not designed for continuous operation, therefore the inactive state of the Motor On signal is used to clear the Drive Select Latch, which de-selects any drives which were previously selected. The Motor On one-shot is retriggerable by simply executing another OUT instruction to the Drive Select Latch.

4.2.4 Wait State Generation and WAITIMOUT Logic

As previously mentioned, a wait state to the CPU can be initiated by an OUT to the Drive Select Latch with D6 set. Pin 5 of U15 will go high after this operation. This signal is inverted by 1/6 of U1 and is routed to the CPU board where it forces the Z-80 into a wait state. The Z-80 will remain in the wait state as long as WAIT* is low. Once initiated, the WAIT* will remain low until one of four

conditions is satisfied. One half of U9 (a five input NOR gate) is used to perform this function. INTQ, DRQ, RESET, and WAITIMOUT are the inputs to the NOR gate. If any one of these inputs is active (logic high), the output of the NOR gate (U9 pin 6) will go low. This output is tied to the clear input of the wait latch. When this signal goes low, it will clear the Q output (U18 pin 5) and set the Q* output (U15 pin 6). This condition causes WAIT* to go high which allows the Z-80 to exit the wait state. U3 is a 12-bit binary counter which serves as a watchdog timer to insure that a wait condition will not persist long enough to destroy dynamic RAM contents. The counter is clocked by a 1 MHz clock and is enabled to count when its reset pin is low (U3 pin 11). A logic high on U3 pin 11 resets the counter outputs. U3 pin 15 is a divide-by-1024 output and is used to generate the signal WAITIMOUT. This watchdog timer logic will limit the duration of a wait to 1024 μsec, even if the FDC chip should fail to generate a DRQ or a INTRQ.

4.2.5 Clock Generation Logic

A 4 MHz crystal oscillator and a 4-bit binary counter are used to generate the clock signals required by the FDC board. The 4 MHz oscillator is implemented with two inverters (1/3 of U22) and a quartz crystal (Y1). The output of the oscillator is inverted and buffered by 1/6 of U22 to generate a TTL level square wave signal. U21 is a 4-bit binary counter which is divided into a divide-by-2 and a divide-by-8 section. The divide-by-2 section is used to generate the 2MHz output at pin 12. The 2 MHz is NANDED with a MHz by 1/4 of U17 and the output is used to clock the divide-by-8 section of U21. A 1 MHz clock is generated at pin 9 of U21 which is 90 degrees phase-shifted from the 2 MHz clock. This phase relationship is used to gate the guaranty Write Data Pulse (WD) to the Write precompensation circuit. The 4 MHz is used to clock the digital data separator U11 and the Write precompensation shift register U10. The 1 MHz clock is used to drive the clock input of the FDC chip (U6) and the clock input of the watchdog timer (U3).

4.2.6 Disk Bus Selector Logic

As mentioned previously, the Floppy Disk Controller board supports up to four drives (two internal and two external). This function is implemented by using two disk drive interface buses, one for the internal drives and one for the external drives. J1 is the edge connector used to drive the internal disk drives and J4 is the edge connector used to drive the external drives. U19 (a quad 2 to 1 data selector) is used to select which set of inputs is routed from the disk drive buses to the FDC chip. U19 pin 1 is the control pin for the data selector. If pin 1 is low, the external inputs are selected, otherwise the internal inputs are selected. This control signal EXTSEL* is generated from the outputs of the Drive Select Latch. If Drive 2 or 3 is selected, U20 pin 1 will go low indicating that an external drive is selected. One half of U9 (a five-input NOR gate) is used to detect when any of the four drives is selected.

The output of the NOR gate (U9 pin 5) is inverted and is used as the head load timing (HLT) and ready (RDY) signal for the FDC chip. Therefore, if any drive is selected, the head is assumed to be loaded and the selected drive is assumed to be ready.

4.2.7 Disk Bus Output Drivers

High current open collector drivers (U18, U8, and U1) are used to buffer the output signals from the FDC board to the disk drives. Note from the schematic that each output signal to the drives has two buffers associated with each signal. One set is used for the internal drive bus and the other set is used for the external drive bus. No select logic is required for these output signals since the drive select bits define which drive is active.

4.2.8 Write Precompensation and Write Data Pulse Shaping Logic

The Write Precompensation logic is comprised of U10 (74LS195), 1/4 of U17, 1/4 of U20, and 1/2 of U15. U10 is a parallel in, serial out shift register and is clocked by 4 MHz which generates a precompensation value of 250 nsec. The output signals EARLY and LATE of the FDC chip (U6) are input to P0 and P2 of the shift register. A third signal is generated by 1/4 of U20 when neither EARLY nor LATE is active low and is input to P1 of U10. WD of the FDC chip is NANDed with 2 MHz to gate the guaranteed Write Data Pulse to U10 for the parallel load signal SHFT/LD. When U10 pin 9 is active low, the signals preset at P1-P3 are clocked in on the rising edge of the 4 MHz clock. After U10 pin 9 goes high, the data is shifted out at a 250 nsec rate. EARLY will generate a 250 nsec delay, NOT EARLY **AND** NOT LATE will generate a 500 nsec delay, and LATE will generate a 750 nsec delay. This provides the necessary precompensation for the write data. As mentioned previously, Write Precompensation is enabled through software by an OUT to the Drive Select Latch with bit 5 set. This sets the Q output of the 74LS74 (U15 pin 9) which disables the shift register U10. This signal also enables U20 to allow the write data (WD) to bypass the Write Precompensation circuit. The Write Date (WD) pulse is shaped by a one-shot (1/2 of U3) which stretches the data pulses to approximately 500 nsec.

4.2.9 Clock and Read Data Recovery Logic

The Clock and Read Data Recovery Logic is comprised of

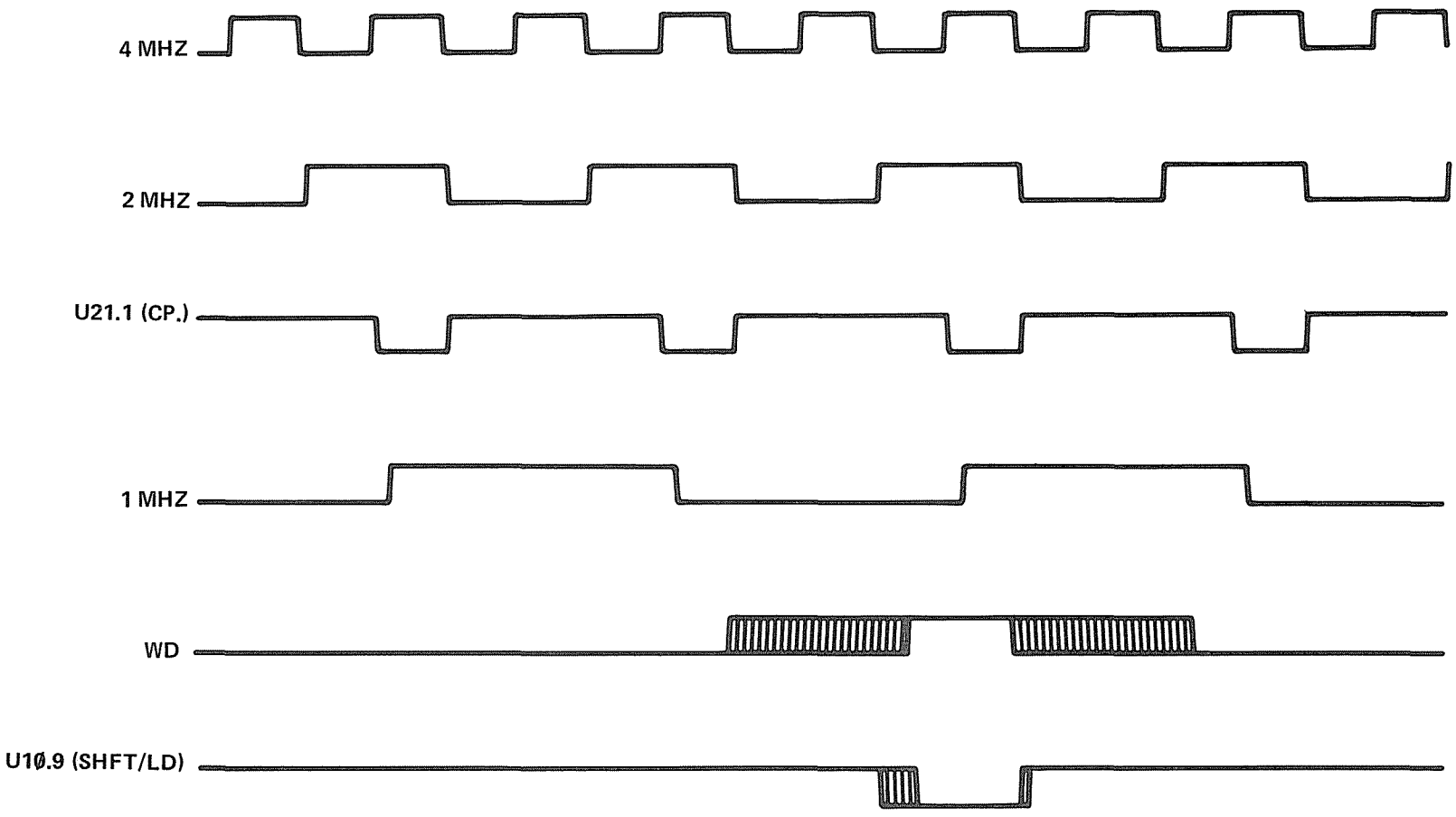


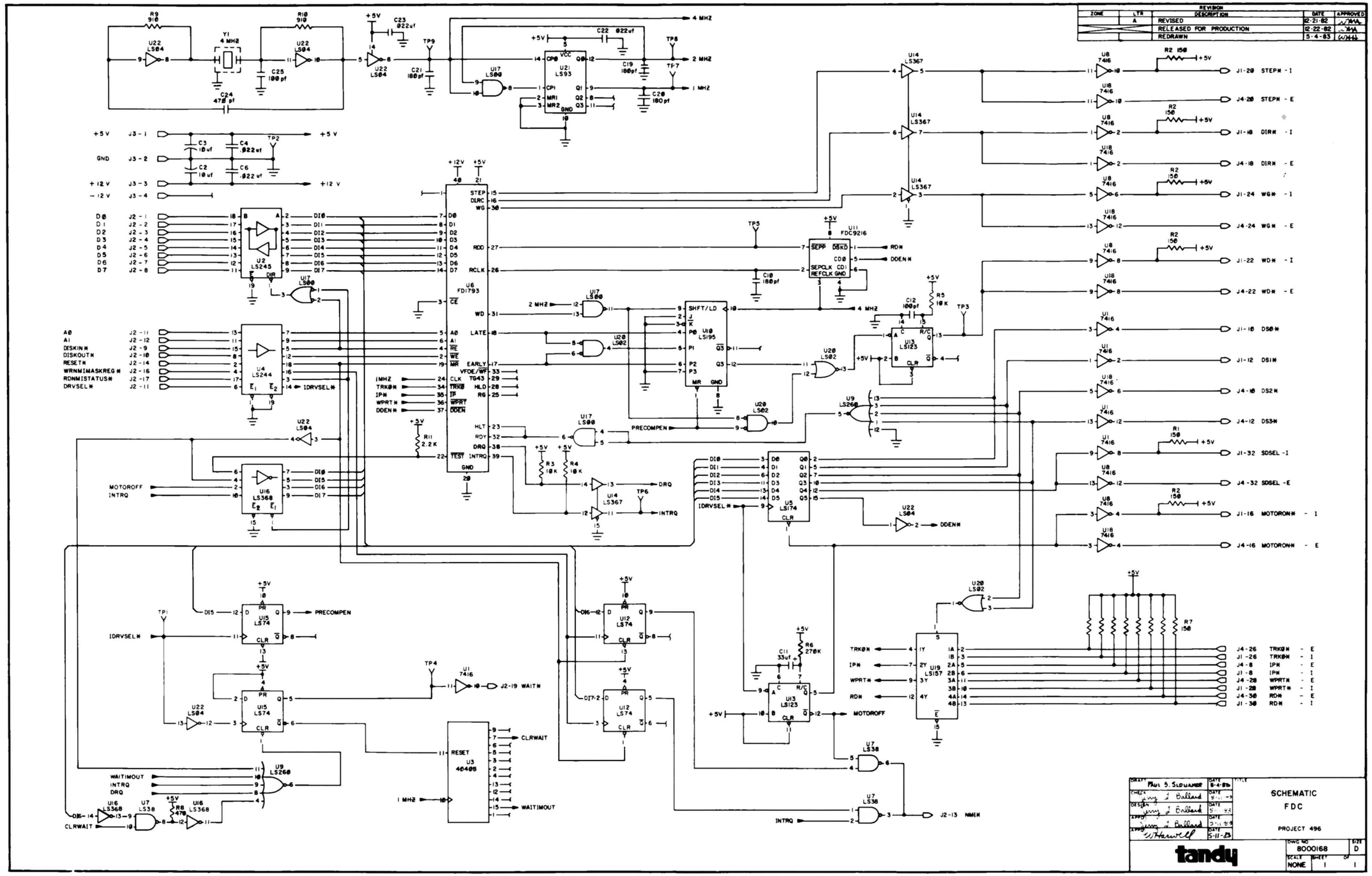
FIGURE 4-1. WRITE PRECOMPENSATION TIMING

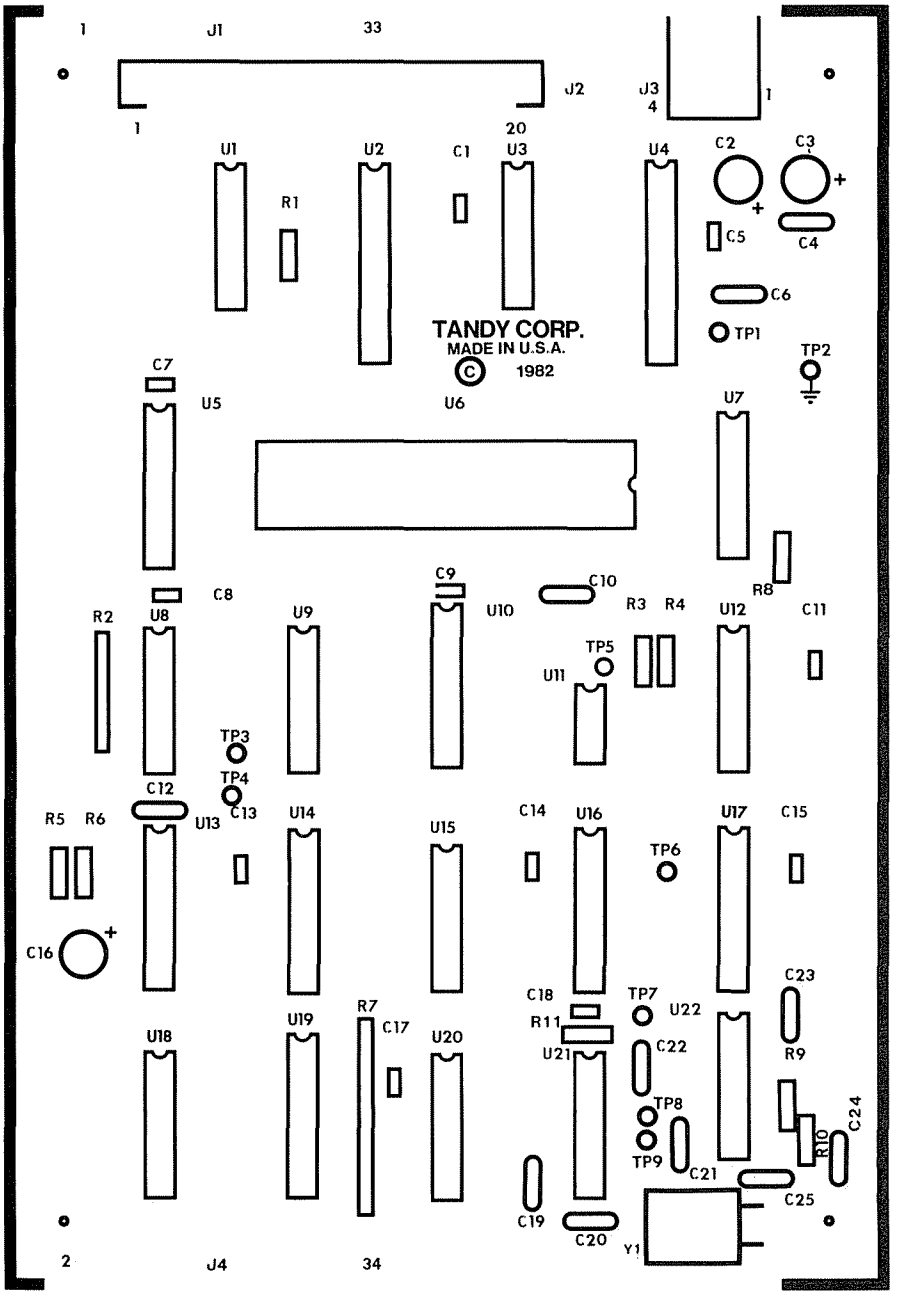
one chip, the FDC9216. The FDC9216 is a Floppy Disk Data Separator (FDDS) which converts a single stream of pulses from the disk drive into separate clock and data pulses for input to the FDC chip. The FDDS consists of a clock divider, a long-term timing corrector, a short-term timing corrector, and reclocking circuitry. The reference clock (REFCLK) is 4 MHz and is divided by the internal clock divider. CD0 and CD1 of the FDDS chip control the divisor which divides REFCLK. With CD1 grounded (logic low), CD0 (when a logic low) generates a divide-by-1 for MFM mode and when logic high generates a divide-by-2 for FM mode. CD0 is controlled by the signal DDEN* which is Double Density Enable or MFM enable. The FDDS detects the leading edges of RD* pulses and adjusts the phase of the internal clock to genearte the separated clock (SEPCLK) to the FDC chip. The separate long and short term timing correctors assure the clock separation to be accurate. The separated Data (SEPD*) is used as the RDD* input to the FDC chip.

4.2.10 Floppy Disk Controller Chip

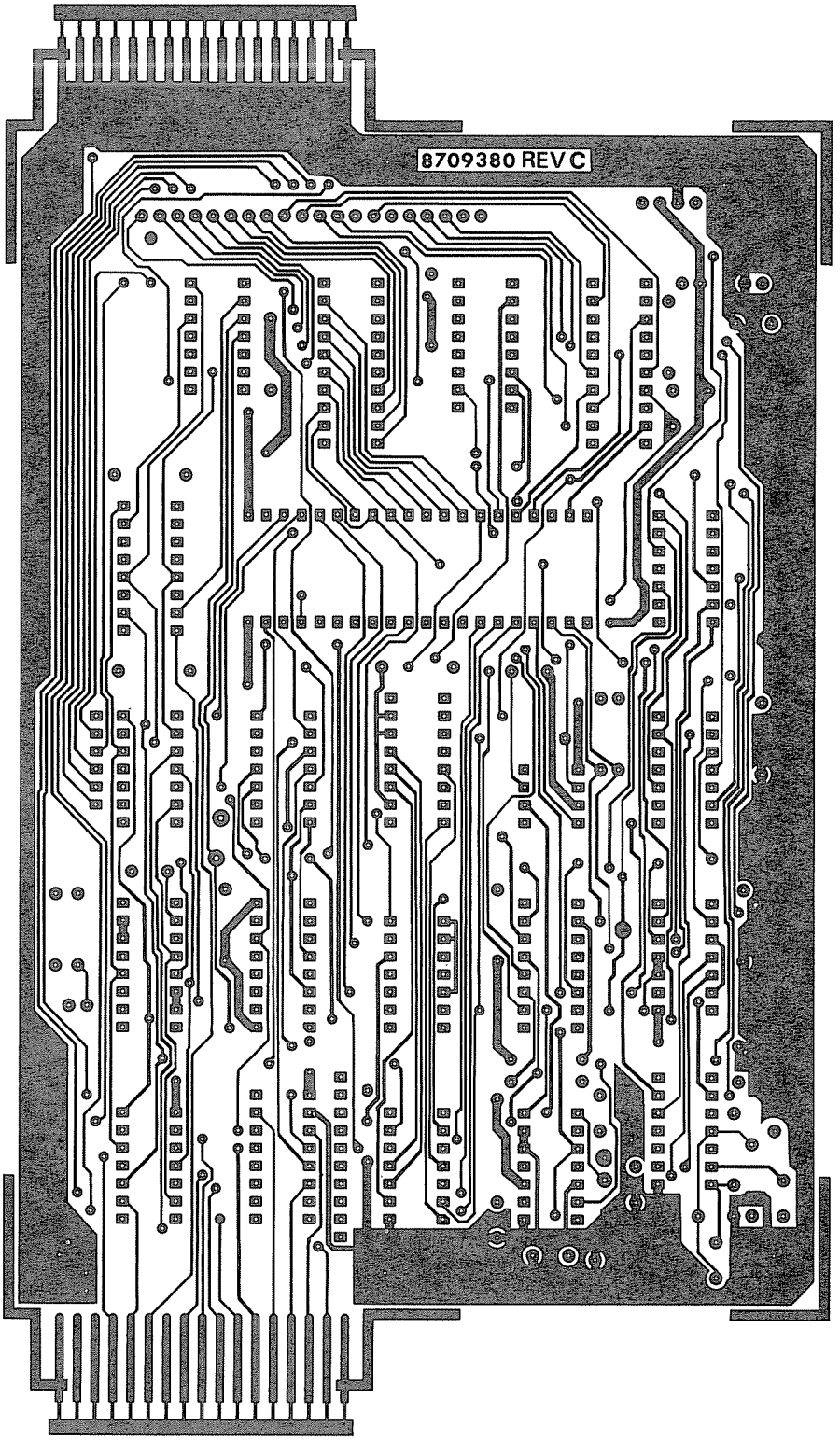
The 1793 is an MOS LSI device which performs the functions of a floppy disk formatter/controller in a single chip implementation. The 1793 is functionally identical to the 1791 used on the Model II FDC Printer Interface Board except that the data bus is true as opposed to inverted. Refer to the appendix section for more information on the FD1793. The Model II Technical Reference Manual also contains a good presentation of the 1791 FDC chip as well as a discussion on Write precompensation. The following port addresses are assigned to the internal registers of the 1793 FDC chip:

Port #	Function
F0H	Command/Status Register
F1H	Track Register
F2H	Sector Register
F3H	Data Register

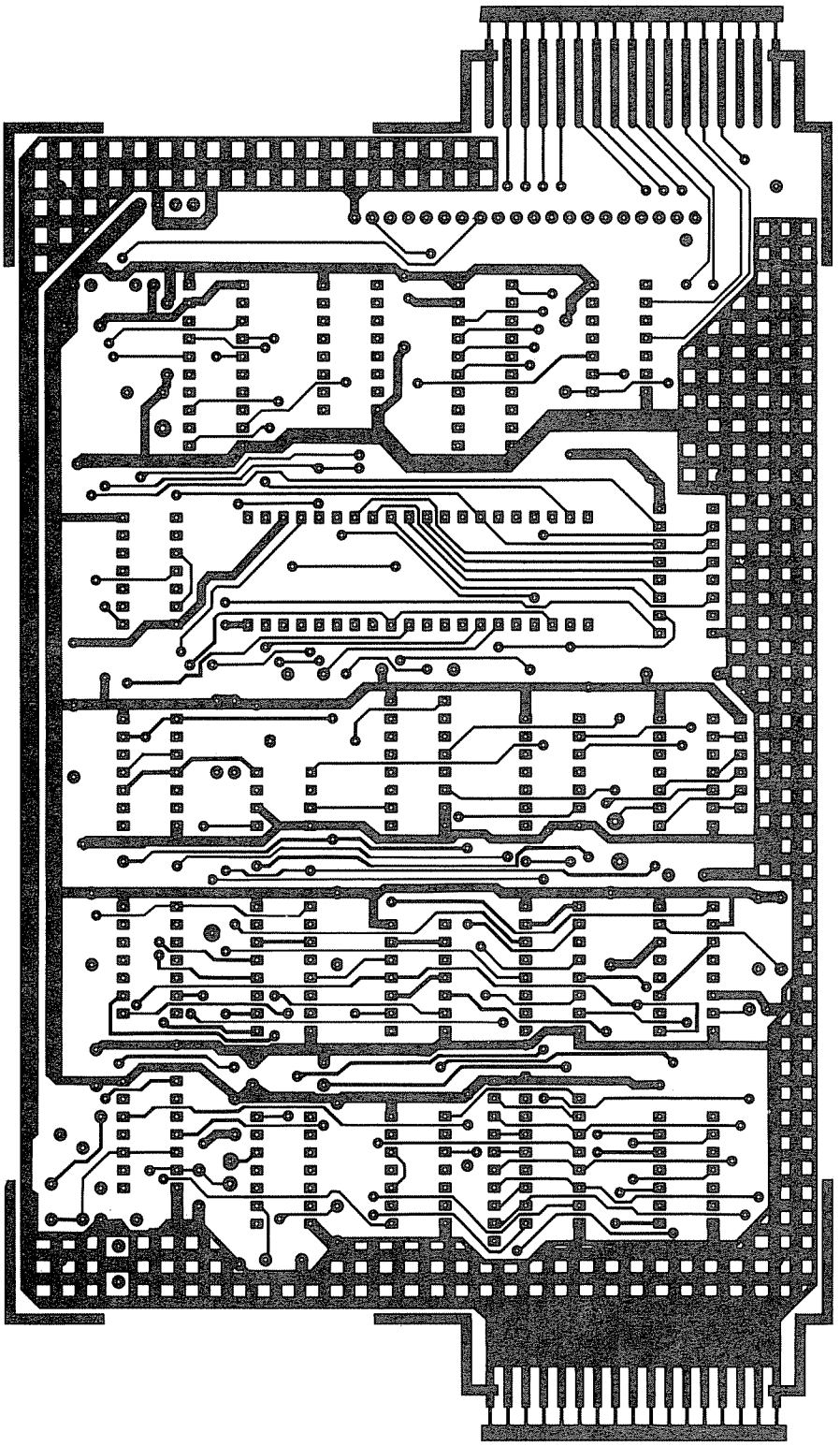

REVISION
ZONE | LTR | DESCRIPTION | DATE | APPROVED
A | REVISED | 12-21-82
RELEASED FOR PRODUCTION | 12-22-82
REDRAWN | 5-4-83
SCHEMATIC
FDC
PROJECT 496
tandy
DWG NO 8000168
SIZE D
SCALE NONE
SHEET 1 OF 1



COMPONENT LAYOUT #1700223C, FLOPPY DISK CONTROLLER PCB #8858160



CIRCUIT TRACE #1700223C, FLOPPY DISK CONTROLLER 8858160 CIRCUIT SIDE



CIRCUIT TRACE #1700223C, FLOPPY DISK CONTROLLER #8858160 SOLDER SIDE

Parts List, FDC PCB #8858160

Item	Qty	Description	Mfgr's Part No.
1	1	Printed Circuit Board	8709380
2	9	Staking Pin (TP1-9)	8529014
3	1	Connector, 20-Pin Rt. Angle (J2)	8519078
4	1	Connector, 4-Pin Rt. Angle (J3)	8519079
5	1	Socket, 8-Pin (U11)	8509011
6	1	Socket, 40-Pin (U6)	8509002
7	1	Resistor, 150 ohm 1/4W 5% (R1)	8207115
8	1	Resistor Pak, 150 ohm 6-Pin (R2)	8290012
9	3	Resistor, 10K ohm 1/4W 5% (R3-5)	8207310
10	1	Resistor, 270K ohm 1/4W 5% (R6)	8207427
11	1	Resistor Pak, 150 ohm 10-Pin (R7)	8290013
12	1	Resistor, 470 ohm 1/4W 5% (R8)	8207147
13	2	Resistor, 910 ohm 1/4W 5% (R9,10)	8207191
14	1	Resistor, 2.2K ohm 1/4W 5% (R11)	8297222
15	11	Capacitor, 0.1 ufd 50V (C1,5,7-9,11,13-15, C17,18)	8374104
16	2	Capacitor, 10 ufd 16V Elec. (C2,3)	8326101
17	4	Capacitor, .022 ufd 50V (C4,6,22,23)	8303224
18	4	Capacitor, 180 pfd 50V C. Disk (C10,19-21)	8301184
19	2	Capacitor, 100 pfd 50V C. Disk (C12,25)	8301104
20	1	Capacitor, 33 ufd 16V Elec. (C16)	8396331
21	1	Capacitor, 470 pfd 50V C. Disk (C24)	8301474
22	1	Crystal, 4.000 MHz (Y1)	8409010
23	3	IC, 7416 Hex Inverter Buffer (U1,8,18)	8000016
24	1	IC, 74LS245 Octal Bus Tranceiver (U2)	8020245
25	1	IC, MC14040 Binary Counter (U3)	8030040
26	1	IC, 74LS244 Octal Buffer (U4)	8020244
27	1	IC, 74LS174 Flip Flop (U5)	8020174
28	1	IC, 1793 FDC (U6)	8030793
29	1	IC, 74LS38 NAND Buffer (U7)	8020038
30	1	IC, 74LS260 5-In NOR Gate (U9)	8020260
31	1	IC, 74LS195 Shift Register (U10)	8020195
32	1	IC, 9216 Data Separator (U11)	8040216
33	2	IC, 74LS74 Flip Flop (U12,15)	8020074
34	1	IC, 74LS123 Mono Multiplier (U13)	8020123
35	1	IC, 74LS367 Hex Buffer (U14)	8020367
36	1	IC, 74LS368 Hex Inv. Buffer (U16)	8020368
37	1	IC, 74LS00 NAND Gate (U17)	8020000
38	1	IC, 74LS157 Quad Multiplier (U19)	8020157
39	1	IC, 74LS02 NOR Gate (U20)	8020002
40	1	IC, 74LS93 4-Bit Counter (U21)	8020093
41	1	IC, 74LS04 Hex Inverter (U22)	8020004

SECTION V

MINI DISK DRIVE

MINI-DISK DRIVES

5.1 MINI-DISK DRIVE #8790112 (T.P.I 01-0053-001)

5.1.1 Physical Description

The electronic components of the Mini-Disk Drive are mounted on the Logic/Servo Board. This board is located above the chassis, and the power and interface signals are connected directly to it.

The spindle is belt driven by a DC motor with an integral tachometer. The servo control circuit, pulleys and the tachometer control the speed of the spindle. The read/write/erase head assembly is positioned by means of a stepper motor, split band and a pulley.

5.1.2 Functional Description

The Disk Drive is fully self-contained. It consists of a spindle drive system, a head positioning system, and read/write/erase system.

When the front latch is opened, access is provided for the insertion of a 5¼ inch (133.4 mm) standard diskette. The diskette is positioned in place by plastic guides, the front latch and a back stop.

Closing the front latch activates the cone/clamp system which centers and clamps the diskette to the drive hub. The drive hub is driven at a constant speed of 300 rpm by a servo controlled DC motor. In operation, the magnetic head is loaded into contact with the recording medium whenever the front latch is closed.

A 4-phase stepper motor/band assembly and its associated electronics position the magnetic head over the desired track. This positioner employs a one-step rotation to cause a 1-track linear movement. When a write-protected diskette is inserted into the Drive, the write-protect sensor disables the write electronics of the Drive and an appropriate signal is applied to the interface. (When performing a write operation, a 0.013-inch (0.33 mm) [nominal] data track is recorded.)

Data recovery electronics include a low-level read amplifier, differentiator, zero-crossing detector and digitizing circuits. No data decoding facilities are provided in the basic Drive.

The Drive is also supplied with the following sensor systems;

1. A track ØØ switch which senses when the Head/Carriage assembly is positioned at Track ØØ.
2. The index sensor (an LED light source and a phototransistor) is positioned so that when an index hole is detected, a digital signal is generated. The index sensor used is a high resolution device which can distinguish holes placed close together, i.e., index-sector holes in a hard sectored diskette.
3. The write-protect sensor disables the Disk Drive electronics whenever a write-protect tab is applied to the diskette.

5.1.3 Interface Connections

Signal connections for the external Disk Drive are made via a user-supplied 34-pin flat ribbon connector. The internal Drive uses a Radio Shack connector, Part Number AW2535. Both of these connectors mate directly with the PC Board connector J2 at the rear of the Drive. The DC power connector is a four - pin connector J4 on the PC Board on the top rear of the Drive.

The signal connector harness should be of the flat ribbon or twisted pair type with the following characteristics:

1. Maximum length of 10 feet (3 M).
2. 22 - 24 gauge conductor compatible with the connector to be used.

Power connections for external Drives should be made with 18 AWG cable. Internal Drives use the Disk DC Power Harness, Radio Shack Part Number AW2532.

5.1.4 Physical Checkout

Before applying power to the unit, the following inspection should be performed:

1 Front latch. Check that the front latch opens and closes. Note that when the door is opened, the head arm raises.
2. Ensure that the front panel is secure.
3. Manually rotate the drive hub. The hub should rotate freely.
4. Check that the PC board is secure. Check that the connectors are firmly seated.

Media	Industry-compatible 5-1/4 inch (133.4 mm) diskette
Tracks per inch	48
Number of Tracks	40
Read/Write Track Width	.020 inches (5.08 mm)
Dimensions	
Height	3.38 inches (85.85 mm)
Width	5.87 inches (149.10 mm)
Depth	8.0 inches (203.2 mm)
Weight	4.5 lbs (2.04 Kg
Temperature (Exclusive of Media)	
Operating	10°C to 43°C (50°F to 110°F)
Non-operating	−40°C to 71°C (−40°F to 160°F)
Relative Humidity (Exclusive of Media)	
Operating	20% to 80%
Non-operating	5% to 95% (non-condensing)
Vibration	6 to 600 Hz 0.5g peak
Seek Time	5 msec track to track
Head Settling Time	15 msec (last track addressed)
Error Rate	1 per 10^9 recoverable 1 per 10^{12} non-recoverable
Head Life	20,000 hours (normal use)
Media Life	3 million passes on a single track
Disk Speed	300 rpm ±1.5% (long term)
Instantaneous Speed Variation	±3.0%
Start/Stop Time	250 msec (maximum)
Transfer Rate	125/250K bits/sec
Bits/Disk (unformatted)	1.75 million (FM)
Recording Modes (typical)	FM, MFM, MMFM
Power	+12 V dc ±0.6 V, 900 ma maximum 5 V dc ±0.25 V, 600 ma maximum

TABLE 5-1. MINI-DISK DRIVE MECHANICAL AND ELECTRICAL SPECIFICATIONS.

5. Check for debris or foreign material between the heads and remove same.

6. Check plastics for broken or cracked pieces, e.g., write protect lever, guide rails, etc.

NOTE: To ensure proper operation of the Drive, the chassis should be connected to earth ground. The 3/16-inch (4.76 mm) male QC lug, located at the rear of the chassis, is provided for this connection.

5.1.5 Mounting The Disk Drive

The Drive can be mounted in any plane, i.e., upright, horizontal or vertical. However, when it is mounted horizontally, the Logic PC board side of the chassis must be the uppermost side. Tapped holes are provided in various locations for the attachment of user-supplied hardware.

5.1.6 Flat Ribbon Cable Assembly

The Flat Ribbon Cable Assembly that is used is the same one used in the Model I and III Computers. Pins must be removed from Drive connectors on the Cable Assembly as noted below:

1. Connector for the first internal drive (Drive 0) and the first external drive (Drive 2) pins 12, 14 and 32.

2. Connector for the second internal drive (Drive 1) and the second external drive (Drive 3), pins 10, 14 and 32.

INTERNAL DRIVES

DRIVE NUMBER ZERO	DRIVE NUMBER ONE
12	10
14	14
32	32

EXTERNAL DRIVES

DRIVE NUMBER TWO	DRIVE NUMBER THREE
12	10
14	14
32	32

FIGURE 5-1. CABLE ASSEMBLY – CONNECTOR PIN REMOVAL CHART

5.1.7 Resistor Termination

The Resistor Termination in the SIP socket (R51) is used only in the last drive in the system. Remove all other Resistor Terminations.

NOTE: The internal Drives in the Model 4 Computer are not terminated.

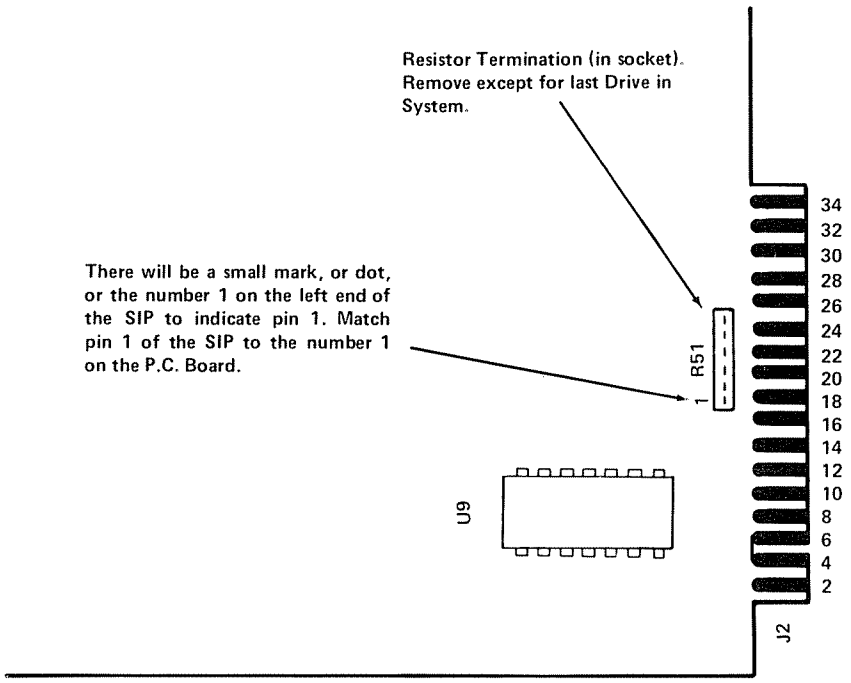


FIGURE 5-2. RESISTOR TERMINATION

5.2 THEORY OF OPERATION

5.2.1 Introduction

The Disk Drive consists of the mechanical and electrical components necessary to record and read digital data on a diskette. DC power at ±12V and +5V (provided by the user for the external drives) is required for operation.

5.2.2 Organization of The Disk Drive

All electrical subassemblies in the Disk Drive are constructed with leads which terminate in 4 or 5 pin connectors, enabling the individual assemblies to be removed.

The magnetic head is connected to the PC board via a cable terminating in a 5-pin female connector and its associated male socket which is located in close proximity to the read/write data electronics.

Interface signals and power are provided via connectors at the rear of the Drive. A detailed description of these signals is presented in Section 5.3 of this manual.

5.2.3 Functional Block Diagram Description

Figure 5-3 is a functional block diagram of the Disk Drive and should be referred to in conjunction with the following discussion:

The Disk Drive consists of the following functional groups:

- ★ Index Pulse Shaper
- ★ Write Protect Sensor
- ★ Track ØØ Sensor
- ★ Spindle Drive Control
- ★ Carriage Position Control
- ★ Write/Erase Control
- ★ Read Amplifier and Digitizer

5.2.4 INDEX

An index pulse is provided to the user system via the INDEX PULSE interface line. The index circuitry consists of an Index LED, Index Phototransistor and a Pulse Shaping Network. As the index hole in the disk passes the Index LED/Phototransistor combination, light from the LED strikes the Index Phototransistor, causing it to conduct. The signal from the Index Phototransistor is then passed to the Pulse Shaping Network which produces a pulse for each hole detected. This pulse is presented to the user on the INDEX PULSE interface line.

5.2.5 Write Protect

A Write Protect signal is provided to the user system via the WRITE PROTECT interface line. The write protect circuitry consists of a Write Protect Sensor and circuitry to route the signal produced.

When a write protected diskette is inserted in the drive, the sensor is activated and the logic disables the write electronics and supplies the status signal to the Interface.

5.2.6 Track ØØ Switch

The level on the TRACK ØØ interface line is a function of the position of the magnetic head assembly. When the head is positioned at Track ØØ and the stepper motor is at the home position, a true level is generated and sent to the user.

5.2.7 Spindle Drive

The Spindle Drive system consists of a spindle assembly driven by a DC motor-tachometer combination through a drive belt.

Associated with the spindle drive motor are the servo electronics required for control.

The control circuitry also includes a current limiter and an interface control line. When the DRIVE MOTOR ENABLE interface line is true, the drive motor is allowed to come up to speed. When the current through the drive motor exceeds 1.3 A, the current limit circuitry disables the motor drive.

5.2.8 Positioner Control

The Head Positioning system utilizes a four-phase stepper motor drive which changes one phase for each track advancement of the Read/Write carriage. In addition to the logic necessary for motion control, a gate is provided as an element for inhibiting positioner motion during a write operation.

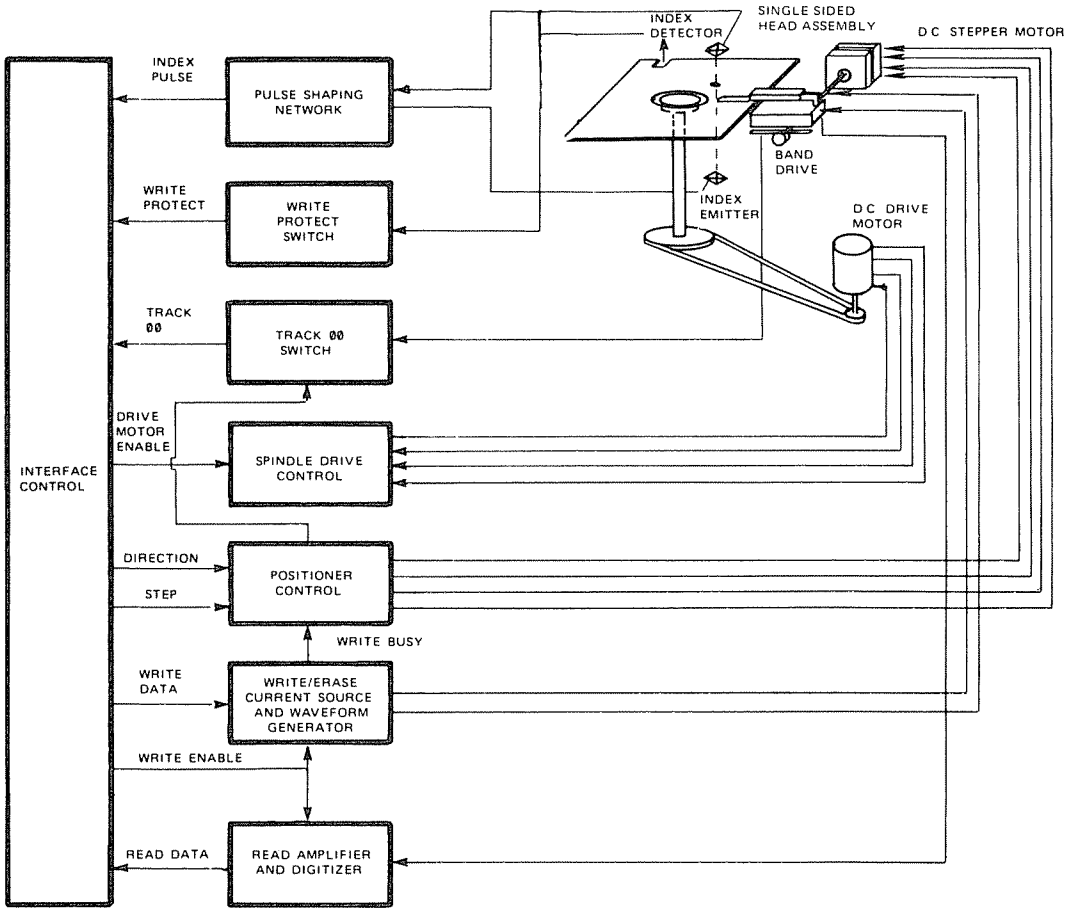


FIGURE 5-3. FUNCTIONAL BLOCK DIAGRAM

5.2.9 Data Electronics

Information can be recorded on the diskette using a double-frequency code. Figure 5-4 illustrates the magnetization profiles in each bit cell for the number sequence shown.

The erase gaps provide an erased guard band on either side of the recorded track. This accommodates the tolerances in track positioning.

All signals required to control the data electronics are provided by the user system and are shown in the Block Diagram (Figure 5-3). These control signals are:

- ★ SELECT
- ★ WRITE ENABLE
- ★ WRITE DATA

The READ DATA composite signal is sent to the user system via the interface.

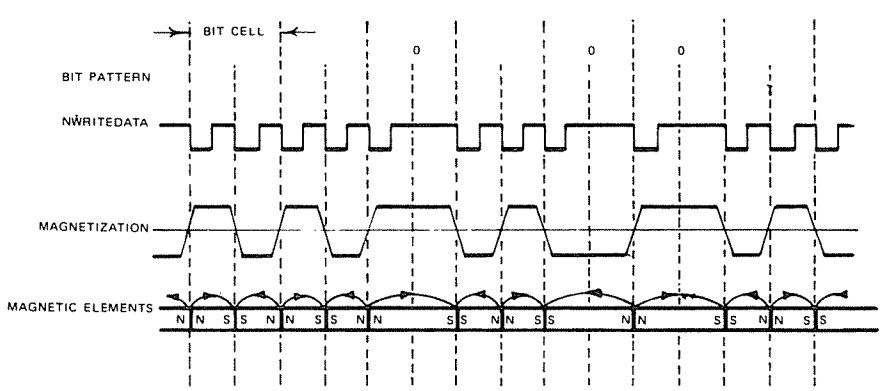


FIGURE 5-4. FM RECORDING

5.2.10 Data Recording

Referring to Drawing 8790112 (page 97) it can be seen that the Write Electronics consists of a Write/Erase Current Source and Write Waveform Generator, Erase Current Source and Trim Erase Control Logic.

The read/write winding on the magnetic head is center-tapped. During a write operation, current from the Write Current Source flows in alternate halves of the winding under control of the Write Waveform Generator.

Before recording can begin, certain conditions must be satisfied. The conditions required for recording (i.e., unit ready) must be established by the user system as follows:

(1) Drive speed stabilization. This condition will exist 250 mSec after starting the drive motor.

(2) Subsequent to any step operation, the positioner must be allowed to settle. This requires 20 mSec total after the last step pulse is initiated, i.e., 5 mSec for the step motion and 15 mSec for settling.

NOTE: All of the foregoing operations can be overlapped, if required.

Figure 5-5 shows the relevant timing diagram for a write operation. When t=Ø , the unit is ready and the WRITE ENABLE interface line goes true, thus enabling the Write Current Source.

Since the trim erase gaps are behind the read/write gap, TRIM ERASE control goes true 390μSec after the WRITE ENABLE interface line. It should be noted that this value is optimized between the requirements at Track ØØ and Track 40 so that the effect of the trim erase gaps on previous information is minimized.

Figure 5-5 also shows the information on the WRITE DATA interface line and the output of the Write Waveform Generator which toggles on the leading edge of every WRITE DATA pulse.

Note that a minimum of 4 μSec and a maximum of 8 μSec between WRITE ENABLE going true and the first WRITE DATA pulse is only required if faithful reproduction of the first WRITE DATA transition is significant.

At the end of recording, at least one additional pulse on the WRITE DATA line must be inserted after the last significant WRITE DATA pulse to avoid excessive peak shift effects.

The TRIM ERASE signal must remain true for 800μSec after the termination of WRITE ENABLE to ensure that all recorded data are trim erased. This value is again optimized between the requirements at Tracks ØØ and 40.

The duration of a write operation is from the true-going edge of WRITE ENABLE to the false-going edge of TRIM ERASE. This is indicated by the internal WRITE BUSY waveform shown.

5.2.11 Data Reproduction

The Read Electronics consists of the following:

- ★ Read Switch
- ★ Read Amplifier
- ★ Filter
- ★ Differentiator
- ★ Comparator and Digitizer

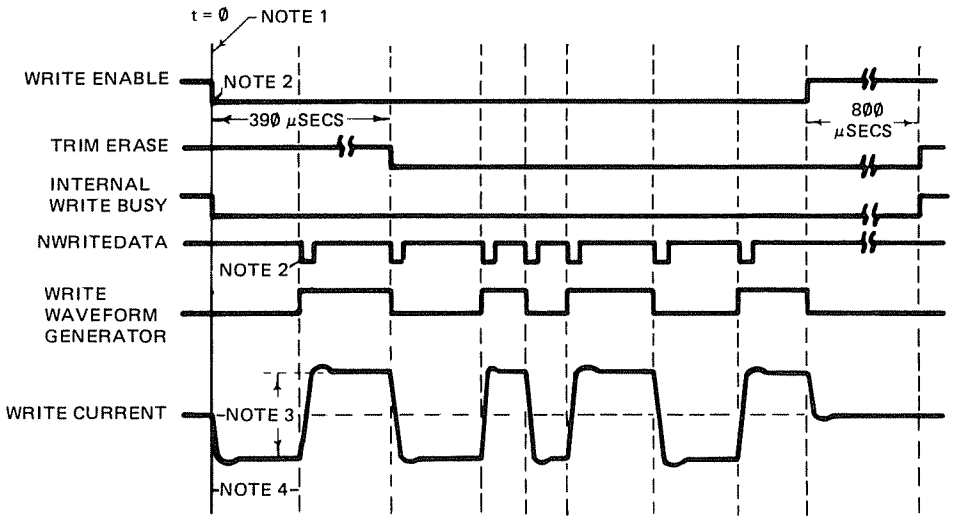
The Read Switch consists of two FETs that are used to isolate the Read Amplifier from electrical signals across the magnetic head during a Write operation.

The remaining functions of the Read circuitry are contained in the MC3470 IC.

Before reading can begin, the Drive must be in a ready condition. As with the data recording operation, this ready condition must be established by the user system. In addition to the requirements established in Paragraph 5.2.10, DATA RECORDING, a 100 μSec delay must exist from the trailing edge of the TRIM ERASE signal to allow the Read Amplifier to settle after the transient caused by the Read Switch returning to the Read mode.

Referring to Figure 5-6, the output signal from the read/write head is amplified by a read amplifier and filtered to remove noise by a Linear Phase Filter. The linear output from the Filter is passed to the Differentiator which generates a waveform whose zero crossovers correspond to the peaks of the read signal. This signal is then fed to the Comparator and Digitizer circuit. Variable resistor R22 is used to balance the Comparator and Differentiator by adjusting out any circuit imbalances and reducing peak shift.

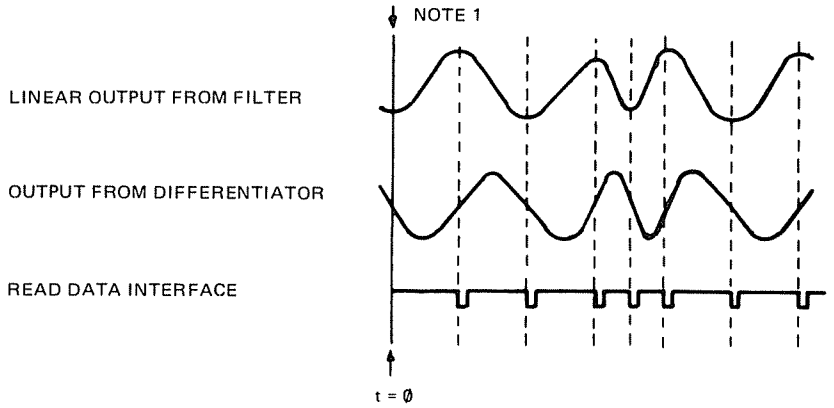
The Comparator and Digitizer circuitry generates a 1 μSec READ DATA pulse corresponding to each peak of the read signal. This Composite Read Data signal is then sent to the user system via the READ DATA interface line.



NOTES:

1. t = Ø = 250 MILLISECONDS AFTER DRIVE MOTOR STARTS, OR 20 MILLISECONDS AFTER LAST STEP PULSE(WHICHEVER IS THE LATEST TIME).
2. UNSYNCHRONIZED
3. 8.5 MA PEAK TO PEAK
4. 4 μSECONDS MINIMUM. 8 μSECONDS MAXIMUM

FIGURE 5-5. WRITE TIMING DIAGRAM



NOTES:
t = Ø = 250 MILLISECONDS AFTER DRIVE MOTOR STARTS, OR 20 MILLISECONDS AFTER STEP COMMAND, OR 100 μSECONDS AFTER TERMINATION OF WRITE BUSY (WHICHEVER IS THE LATEST TIME)

FIGURE 5-6. READ TIMING DIAGRAM

5.3 OPERATION

5.3.1 Introduction

This section contains the interface description and the electrical adjustments necessary for the Disk Drive.

5.3.2 Interface Electronics Specifications

All interface signals are TTL compatible. Logic true (low) is +0.4 V (maximum). Logic false (high) is +2.4 V (minimum). Figure 5-7 illustrates the interface configuration.

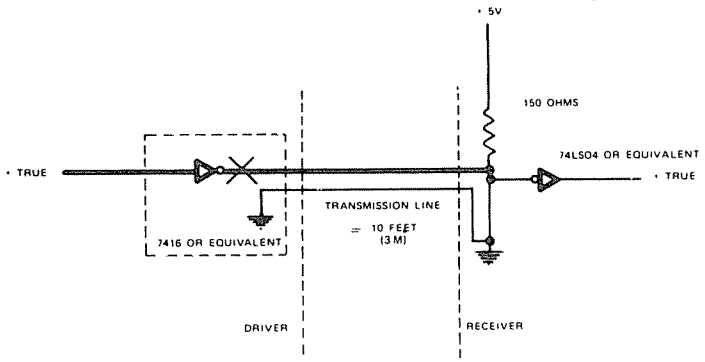


FIGURE 5-7. INTERFACE CONFIGURATION

It is recommended that the interface cable be a flat ribbon cable, with a characteristic impedance of 100 ohms (or equivalent twisted pairs). Maximum interface cable length is 10 feet (3 M).

5.3.3 Input Control Lines

Interface connector pin assignments and power connector pin assignments are given in Table 5-2 and Table 5-3.

5.3.4 Select Lines (DSI* - DS4*)

The SELECT lines provide a means of selecting and deselecting a Disk Drive. These four lines (DS1* - DS4*) select one of the four Disk Drives attached to the controller. When the signal logic level is true (low), the Disk Drive electronics are activated and the Drive is conditioned to respond to step or read/write commands. When the logic level is false (high) the input control lines and output status lines are disabled.

A SELECT line must remain stable in the true (low) state until the execution of a step or read/write command is completed.

The Disk Drive address is determined by SELECT lines 1 through 4 on the PC board. These lines provide a means of daisy-chaining a maximum of four Disk Drives to a controller. Only one line can be true (low) at a time. An undefined operation might result if two or more units are assigned the same address or if two or more SELECT lines are in the true (low) state simultaneously.

5.3.5 Drive Motor Enable

When this signal line logic level goes true (low), the drive motor accelerates to its nominal speed of 300 rpm and stabilizes in less than 250 mSec. When the logic level goes false (high), the Disk Drive stops.

5.3.6 Direction and Step (DIR*) (STEP*)

When the Disk Drive is selected, a true (low) pulse with a time duration greater than 200 nSec on the STEP line initiates the access motion. The direction of motion is determined by the logic state of the DIRECTION line when a STEP pulse is issued. The motion is towards the center of the disk if the DIRECTION line is in the true (low) state when a STEP pulse is issued. The direction of motion is away from the center of the disk if the DIRECTION line is in the false (high) state when a STEP pulse is issued. To ensure proper positioning, the DIRECTION line should be stable 0.1 μSec (minimum) before the trailing edge of the corresponding STEP pulse and remain stable until 0.1 μSec after the trailing edge of the STEP pulse. The access motion is initiated on the trailing edge of the STEP pulse.

5.3.7 Compensated Write Data (CWD)

When the Disk Drive is selected, this interface line provides the bit-serial WRITE DATA pulses that control the switching of the write current in the head. The write electronics must be conditioned for writing by the WRITE ENABLE line (refer to the WRITE ENABLE paragraph below).

For each high-to-low transition on the WRITE DATA line, a flux change is produced at the head write gap. This causes a flux change to be stored on the disk.

When the double-frequency type encoding technique is used (in which data and clock form the combined Write Data signal), it is recommended that when writing all zeroes, the repetition rate of the high-to-low transitions be equal to the nominal data rate, ±0.1%. The repetition rate of the high-to-low transitions, when writing all ones, should be equal to twice the nominal data rate, ±0.1%.

5.3.8 Write Gate (WG*)

When this signal is true (low), the write electronics are prepared for writing data (read electronics disabled). This signal turns on write current in the read/write head. Data is written under control of the WRITE DATA input line. It is generally recommended that changes of state on the WRITE ENABLE line occur before the first WRITE DATA pulse. However, the separation between the leading edge of WRITE ENABLE and the first significant WRITE DATA pulse should not be less than 4 μSec and not greater than 8 μSec. The same restrictions exist for the relationship between the least significant WRITE DATA pulse and the

Controller-to-Disk Drive		
Ground	Signal	Description (Mnemonic)
1	2	Connector clamp
3	4	(Spare)
5	6	(Spare)
9	10	SELECT 1 (NDS1)
11	12	SELECT 2 (NDS2)
13	14	SELECT 3 (NDS3)
15	16	DRIVE-MOTOR ENABLE (NMOTORON)
17	18	DIRECTION
19	20	STEP (NSTEP)
21	22	WRITE DATA (NWRITEDATA)
23	24	WRITE GATE (NWRITEGATE)
31	32	SELECT 4 (NDS4)

Disk Drive-to-Controller		
Ground	Signal	Description (Mnemonic)
7	8	INDEX (NINDEX/SECTOR)
25	26	TRACK ØØ (NTRKØØ)
27	28	WRITE PROTECT (NWRITEPROTECT)
29	30	READ DATA (NREADDATA)
33	34	Connector Clamp

TABLE 5-2. DISK DRIVE EDGE CARD PIN ASSIGNMENTS – J2.

Pin	Supply Voltage
1	+12 V DC
2	Return (+12 V DC)
3	Return (+5 V DC)
4	+5 V DC

TABLE 5-3. POWER CONNECTOR PIN ASSIGNMENTS

termination of the WRITE ENABLE signal. When the WRITE ENABLE line is false (high), all write electronics are disabled.

When a write protected diskette is installed in a Disk Drive, the write electronics are disabled regardless of the state of the WRITE ENABLE line.

5.3.9 Output Status
(See Table 5-2)

5.3.10 Index Pulse (IP*)

The INDEX PULSE signal is provided once each revolution (200 mSec, nominal) to indicate to the controller the beginning of a track. The INDEX PULSE line remains in the true (low) state for the duration of the INDEX PULSE (The duration of an INDEX PULSE is nominally 4.0 mSec).

The leading edge of an INDEX PULSE must always be used to ensure diskette interchangeability between Disk Drives.

5.3.11 Track ØØ (TRKØ*)

When the Disk Drive is selected, the TRACK ØØ interface signal indicates to the controller that the read/write head is positioned at Track ØØ. The Track ØØ remains true (low) until the head is moved away from Track ØØ.

5.3.12 Write Protect (WPRT*)

When the Disk Drive is selected, this signal line logic level goes true (low) when the diskette is write protected. The write electronics are internally disabled when the diskette is write protected.

NOTE: It is recommended that the write data line be inactive whenever Write Enable is false (i.e., read state).

When the level on this line is false (high), the write electronics are enabled and the write operation can be performed. It is recommended that the controller not issue a write command when the WRITE PROTECT signal is true (low).

5.3.13 Read Data (RD*)

This interface line transmits the readback data to the controller when the Drive is selected. It provides a pulse for each flux transition recorded on the medium. The READ DATA output line goes true (low) for a duration of 1 μSec for each flux change recorded.

The leading edge of the READ DATA output pulse represents the true positions for the flux transitions on the diskette surface.

5.4. MAINTENANCE

5.4.1 Physical Description Of The PC Board

The Logic PC board is approximately 6 inches (152 mm) long by 5.5 inches (139 mm) wide. Figure 5-8 illustrates the placement of test points and connectors.

INSTALLATION

To replace the Logic P.C. Board and the Servo P.C. Board with the newer Logic/Servo P.C. Board, the following steps should be taken:

1. Discard the jumper cable that goes from P13 on the Logic P.C. Board to P20 on the Servo P.C. Board.

2. Disconnect all cables that connect the two Boards.

3. Remove both the Logic and the Servo P.C. Boards and the Servo P.C. Board mounting hardware.

4. Connect all cables except the Drive Motor Cable that was connected to P21 on the Servo P.C. Board.

5. Connect the Drive Motor Cable to P13 on the Logic/Servo P.C. Board.

6. Go through the maintenance checks again. Pay particular attention to 5.5.1, DRIVE MOTOR MAINTENANCE and 5.5.8, PEAK SHIFT COMPENSATION ADJUSTMENT.

5.4.2 Circuit Board Test Points

The following test point description assumes that the Logic and Servo P.C. boards are installed in the Drive and that the Drive is in an operational mode with a diskette installed:

Analog Ground (TP1)
The analog ground reference point is provided for measuring read/write waveforms.

Amplified Read Signal (TP2, TP3)
These test points are provided to observe the differential output of the first stage of read signal amplification.

Logic Ground (TP4)
Digital Logic ground is referenced at TP4.

Track ØØ (TP5)
This signal is low (true) when the carriage is positioned at track ØØ and the step motor phase is correct.

Step Pulse (TP6)
When stepping in or out, the signal is a high - going pulse for each step of the carriage.

Read Data One Shot (TP7)
The output of the one shot used in the read section is nominally 1.0 μsec for each flux transition detected.

Write Protect Switch (TP8)
When a write protected diskette is installed in the Drive, the signal is low (true).

Index Pulse (TP9)
With a standard soft sectored diskette installed, the signal is a high going - pulse, nominally 4.0 msec in duration every 200 msec.

Motor On (TP11)
This signal is low (true) for the Motor On condition.

5.4.3 Option Select

Input Line Terminations

The Disk Drive has been provided with the capability of terminating the input lines listed below:

- ★ Motor On
- ★ Direction Select
- ★ Step
- ★ Write Data
- ★ Write Gate

These lines are terminated through a 150 ohm resistor pack that is installed in a SIP socket (R51), located adjacent to J2.

In a multiple drive system, only the last drive on the interface is to be terminated. All other drives on the interface must have the resistor pack removed. Catalog Numbers 26-1160 and 26-1164 are shipped with the termination resistor installed. Catalog Numbers 26-1161/1162/1163 are shipped without a resistor pack.

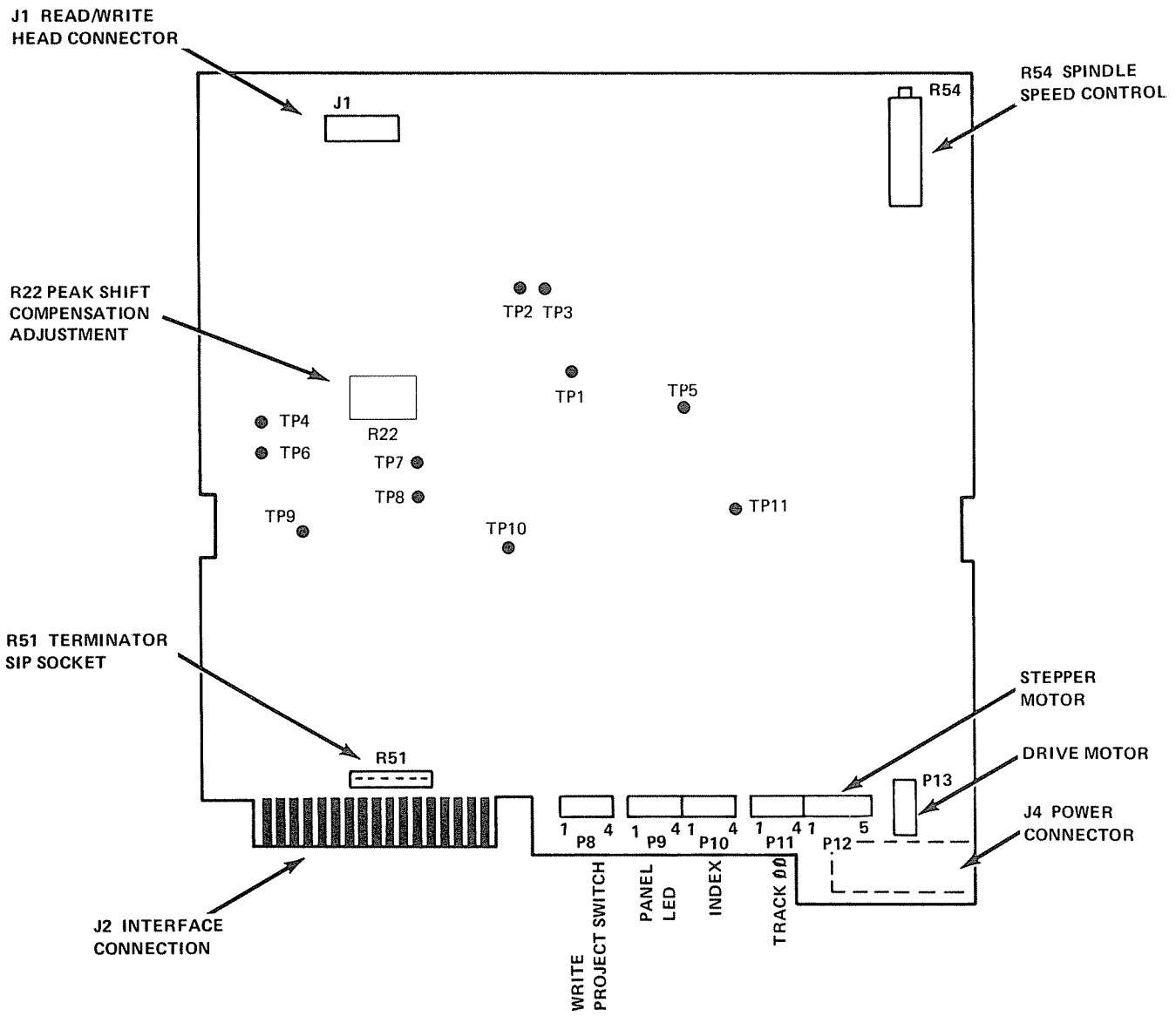


FIGURE 5-8. LOGIC/SERVO PC BOARD TEST POINTS AND CONNECTOR LOCATIONS

5.4.4 Preventive Maintenance

To ensure that the Disk Drive operates at its design potential, the only scheduled preventive maintenance required is periodic cleaning of the magnetic recording head.

Mechanical and electrical adjustment details are provided for further service as a result of disassembly or repair.

5.4.5 Cleaning The Head

To clean the magnetic head, we recommend using Radio Shack's Universal Disk Drive Head Cleaning Kit, catalog number 26-407. This kit comes with complete instructions for safely cleaning the disk drive heads.

CAUTION: Rough or abrasive cloth should not be used to clean the magnetic recording head. Use of cleaning solvents other than 91% Isopropyl alcohol may damage the head.

Extreme care must be exercised to prevent the heads from being damaged (i.e., scratching, banging together, etc.).

5.5 ALIGNMENT AND ADJUSTMENT

To perform the alignment and adjustment procedures, you will require the following equipment:

- ★ 30 MHz Dual Trace Triggered Sweep Scope
- ★ 5 1/4" Alignment Diskette
- ★ 5 1/4" Blank "scratch" Diskette

5.5.1 Drive Motor Maintenance

Long Term Motor Speed Adjustment

1. Check power to unit. (+12 VDC ± Ø.6V, and +5 VDC ± Ø.25V.)

2. Insert a blank diskette and activate the drive.

3. Under fluorescent lighting, observe the speed disk on the spindle pulley. For 6Ø Hz operation, the outer ring on the speed disk should appear stationary. For 5Ø Hz operation, the inner ring should appear stationary.

4. If the speed disk does not seem to be motionless, adjust R54 (located in the upper right corner of the PC Board) for the desired effect.

5. If motor speed cannot be adjusted, repair Servo PCB or motor as required.

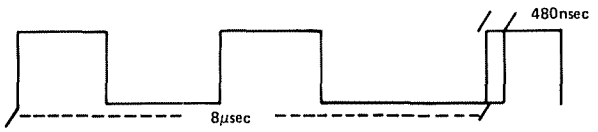
Instantaneous Speed Variation Check/RAW Data Check

1. With a blank diskette inserted, write a 2F pattern on any track.

2. Connect scope to TP 7 with:

   Vert. to 2 volts/div.
   Time Base to 1 μsec/div.
   Trigger internal/positive

3. Observe the waveform. Measure jitter on the leading edge of the third pulse. The leading edge of the pulse should start 8 μsec ±24Ø nsec from the trigger pulse. The third pulse jitter should be less than 48Ø nsec (edge to edge).



4. If jitter is excessive, repair or replace the drive belt, the motor, the Servo PC Board, or the spindle as necessary.

5.5.2 Carriage Movement Check

1. Step between tracks ØØ and 34/4Ø.

2. Check carriage movement. Be certain that carriage is moving freely and not binding at any point. Repair if necessary.

5.5.3 Head Radial Alignment/CE Alignment

1. Insert a 5 1/4" alignment diskette.

2. Connect Scope:
   Channel A to TP 2
   Channel B to TP 3
   Ground to TP 1
   Ext. Sync to TP 9

   Set mode to add (A + B) Invert B
   Vert. to 50 mV/div
   Trigger external - loosely.

3. Read track 16 and verify the "cat eyes" pattern as shown in Figure 5-9. The smaller of the two "eyes" should not be less than 75% amplitude of the other. % = (small lobe/larger lobe) x 100.

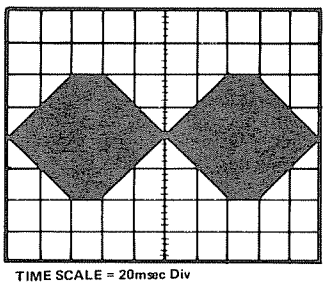


FIGURE 5-9. "CAT EYES" PATTERN

4. Step to track ØØ. Return to track 16 and re-verify the pattern.

5. Step to track 34/4Ø and return to track 16. Re-verify the "cat eyes" pattern.

If Radial Needs Adjustment:

a. Loosen the two retaining screws on back of the unit and the retaining screw on the carriage assembly.

b. Turn the Adjustment Cam until the "cat eyes" pattern is within 75%. If the pattern does not come within spec, loosen the hex head screw on the collar of the stepper motor and rotate the stepper motor shaft until the "eyes" are to the 75% level. Tighten the hex head screw and return to step a.

c. Tighten all screws and recheck adjustment.

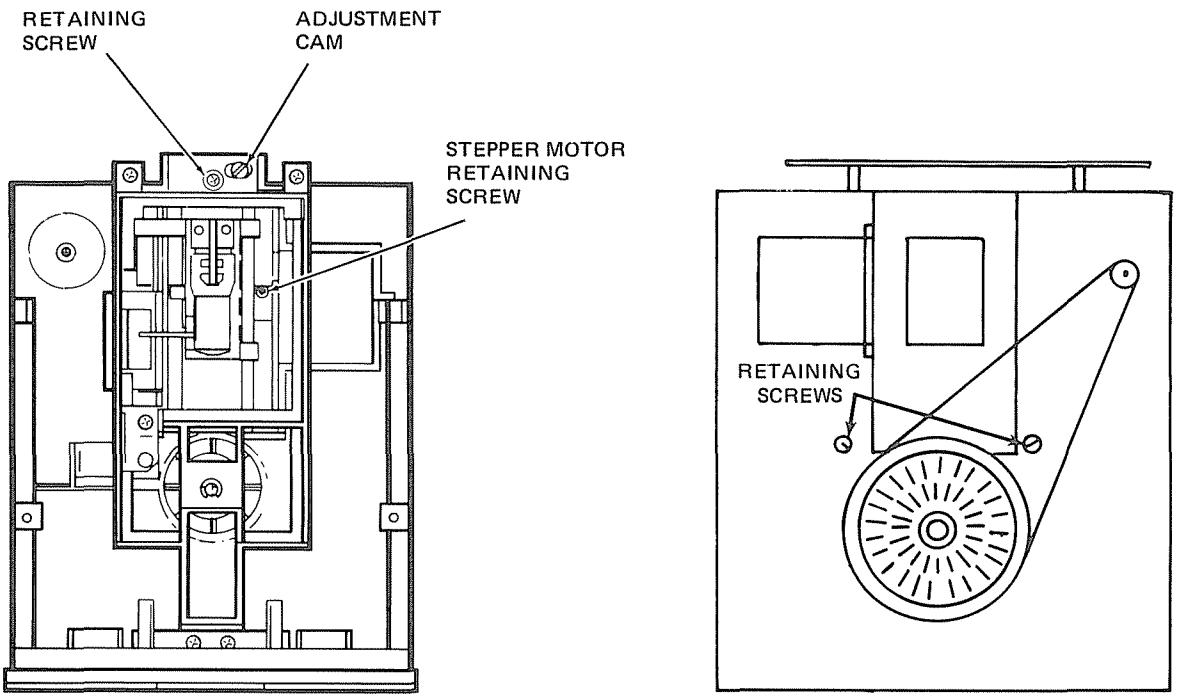


FIGURE 5-10. RADIAL ADJUSTMENT

5.5.4 Track ØØ Alignment

Track ØØ Switch Setting

This switch is set at the factory and glued into position so this will not be a routine adjustment. It will require adjustment only after switch replacement. The switch should be positioned at the front of the adjustment slot and the front screw tightened. Position the head at Track Ø 2. Raise the rear of the switch until the switch contact is closed, then tighten the rear screw. Add a drop of Locktight 42Ø to lock the adjustment into place. Check it for the following conditions: the switch should make (signal low) at Track Ø1, and be off (signal high) at track Ø3. This may be checked at Plug 11, pin 1.

Track ØØ Stop

A Track Ø Ø stop is incorporated into the bracket which attaches the guide rods at the rear of the Disk Drive assembly. This stop is not adjustable.

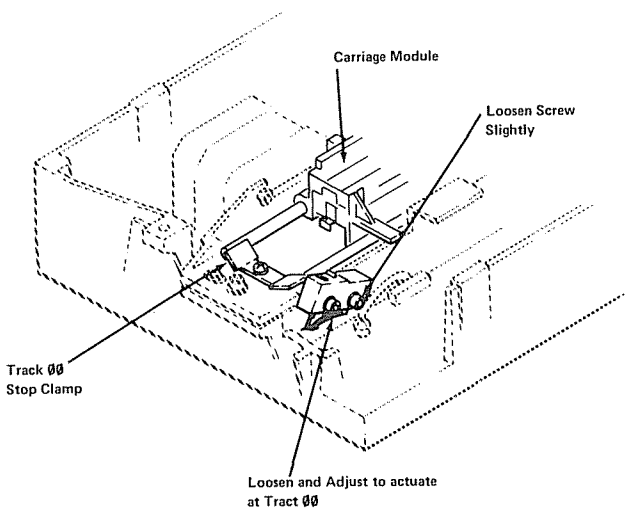


FIGURE 5-11. TRACK ØØ SWITCH ADJUSTMENT

5.5.5 Write Protect Switch

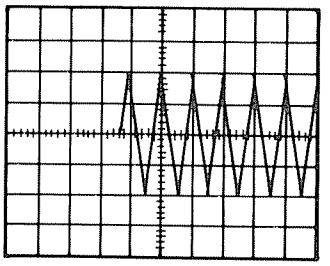
The write protect switch is a field-adjustable setting. The switch is located on the front right-hand corner of the drive. The two screws holding this switch are located on the outside of the chassis. To adjust the switch, loosen the rear screw and place a .Ø 1Ø" shim at the top and inside of the rail slot. Lift the switch until contact is made with the shim, then tighten the rear screw. Check with a write-protected diskette for write protect action, and a non write-protected diskette to verify that the write protect switch is properly functioning. Attach the meter to Plug 8, pin 1. The level should be low for a write-protected diskette and high for a non write-protected diskette.

5.5.6 Index Sector Adjustment

1. Set Scope:
   Vert. at Ø2 volts/div.
   Time Base 5Ø μsec/div.

2. Insert alignment diskette and read track Ø1. Scope display should show Ø volts AC trace for 2ØØ μsec ±5Ø μsec. If the waveform is outside of spec., loosen the index sector light mounting screw and adjust to spec.

3. Open and close the Drive door and re-verify the index sector timing.



TIME SCALE = 50μsec/DIV

FIGURE 5-12. INDEX SECTOR TIMING

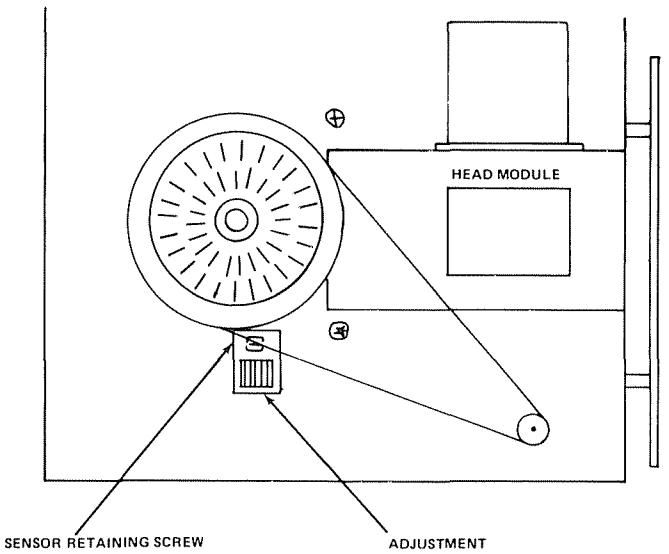


FIGURE 5-13. INDEX ADJUSTMENT

5.5.7 Head Amplitude/Compliance Check

1. Insert a blank diskette and write a pattern to track 34.

2. Set Scope:
   Time Base to 10μsec/div.
   Read the amplitude.

3. Apply an additional 15 grams of pressure to the head load arm. (15 grams is approximately equal to the weight of a 25 cent piece.)

4. Observe the amplitude. If it increases more than 10%, the compliance needs to be adjusted.

5. To adjust the compliance, loosen the two nuts that hold the head load arm in place. While monitoring the amplitude, move the arm until output is highest. Hold the arm in this position and tighten the nuts.

6. Re-verify compliance. If compliance cannot be adjusted properly, replace the head load arm.

PEAK SHIFT COMPENSATION ADJUSTMENT

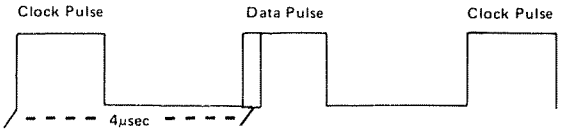
R22 is adjusted as follows:

1. With a blank diskette inserted, write a 2F pattern on the innermost track (track 39).

2. Connect the 'scope to TP7 with:

   Vert. to 2 volts/div.
   Time Base to 1 μsec/div.
   Trigger internal/positive

3. The waveform should look like this:



4. Adjust R22 to center the second pulse between the first and third pulses with a maximum jitter of 240 nanoseconds.

5.5.8 Final Check

1. Use the drive to format and backup a blank diskette.

2. Check the diskette in a known good drive.

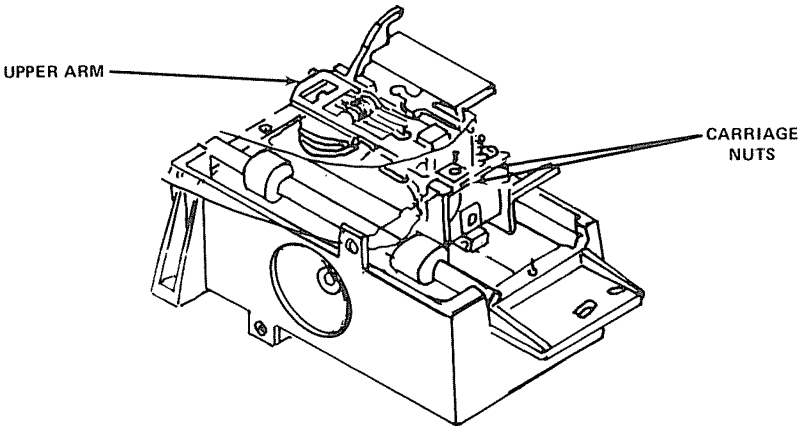
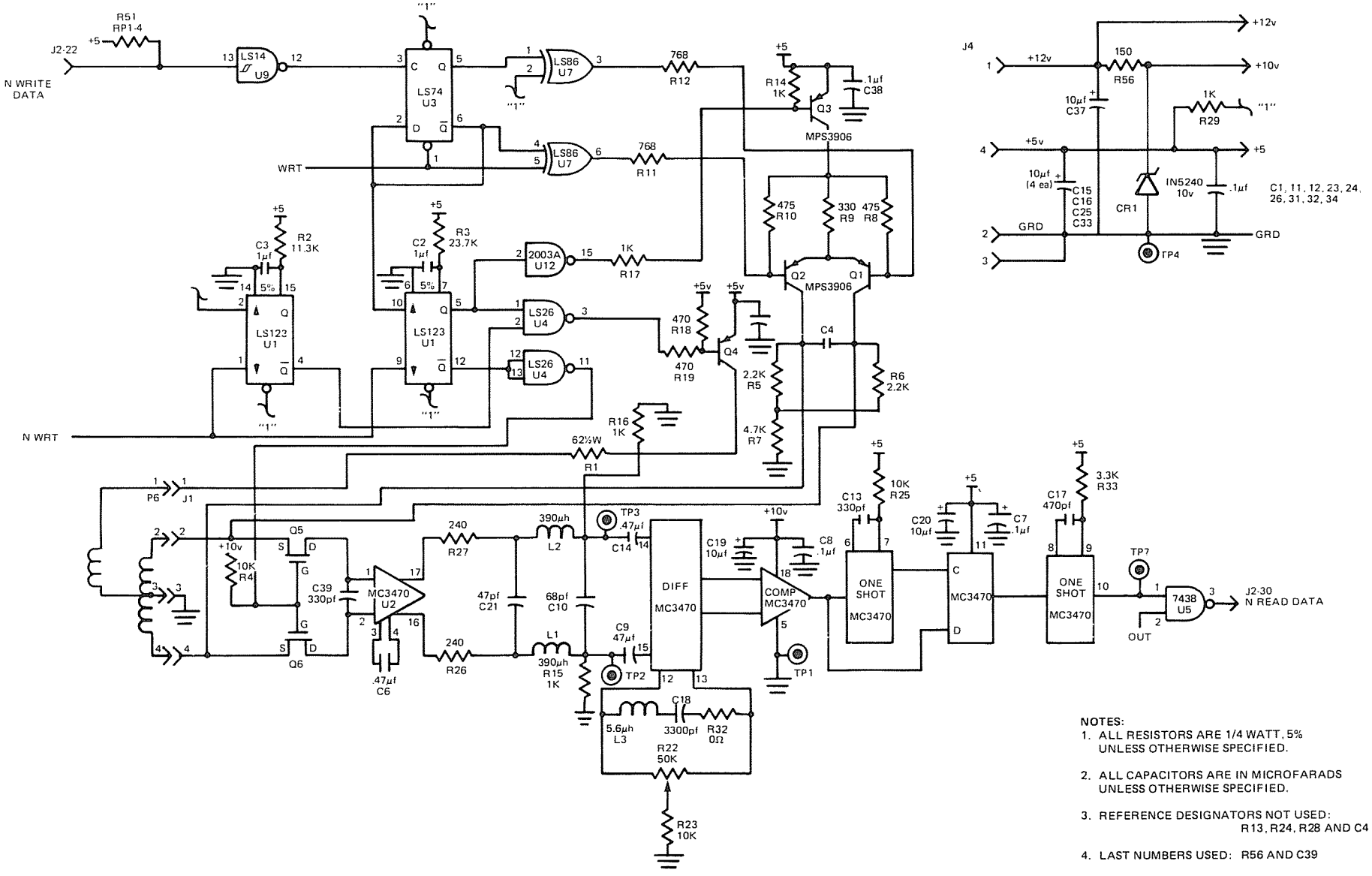


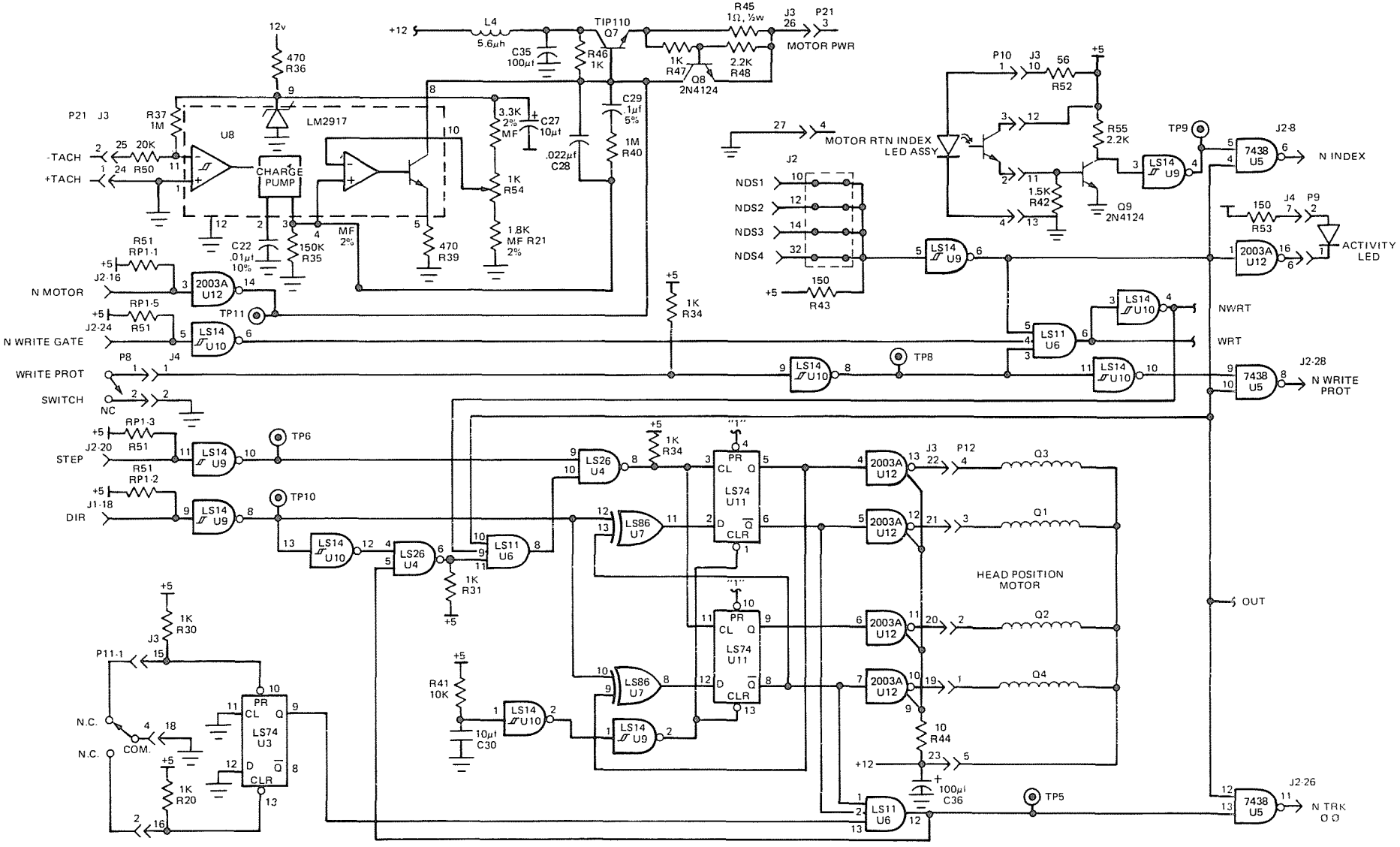
FIGURE 5-14. UPPER ARM AND CARRIAGE



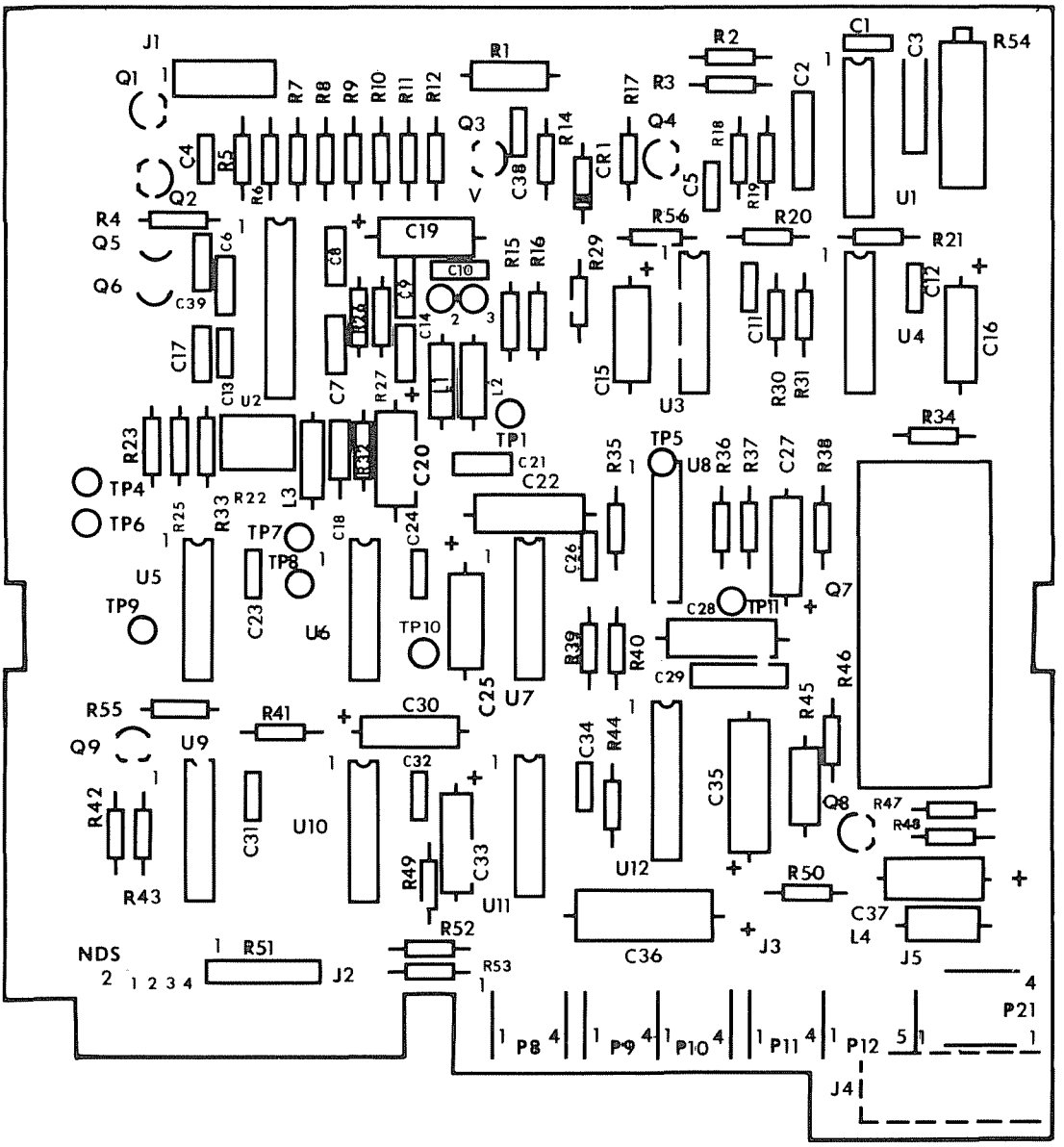
NOTES:

1. ALL RESISTORS ARE 1/4 WATT, 5% UNLESS OTHERWISE SPECIFIED.
2. ALL CAPACITORS ARE IN MICROFARADS UNLESS OTHERWISE SPECIFIED.
3. REFERENCE DESIGNATORS NOT USED: R13, R24, R28 AND C4
4. LAST NUMBERS USED: R56 AND C39

SCHEMATIC DIAGRAM, MINI-DISK DRIVE #8790112 (Sheet 1 of 2)



SCHEMATIC DIAGRAM, MINI-DISK DRIVE #8790112 (Sheet 2 of 2)



LOGIC BOARD (COMPONENT SIDE)

COMPONENT LOCATION, LOGIC/SERVO PC BOARD #995050001

PARTS LIST
Logic/Servo Controller Board #995050001

Symbol	Description	Manufacturer's Part Number	Radio Shack Part Number
	PC Board (Rev C)	————	105050-001
	CAPACITORS		
C1	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C2	0.1 μF, 100 VDC, Polyester	ACC302KLGP	250001-233
C3	0.1 μF, 100 VDC, Polyester	ACC302KLGP	250001-233
C4	Not Used	————	————
C5	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C6	0.47 μF, 50 V, Z5U, Radial	ACC474ZJCP	250001-647
C7	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C8	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C9	0.47 μF, 50 V, Z5U, Radial	ACC474ZJCP	250001-647
C10	68 pF, 50 V, Radial	ACC680KJCP	250001-068
C11	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C12	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C13	330 pF, 50 V, Radial	ACC331KJCP	250001-133
C14	0.47 μF, 50 V, Z5U, Radial	ACC474ZJCP	250001-647
C15	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C16	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C17	470 pF, 50 V, Radial	ACC471KJCP	250001-147
C18	3300 pF, 100 V, Polyester	ACC302KLGP	250001-233
C19	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C20	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	251002-410
C21	470 pF, 50 V, Radial	ACC471KJCP	250001-147
C22	0.01 μF, 63 V, Axial	ACC103KJHA	250102-210
C23	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C24	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C25	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C26	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C27	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C28	0.022 μF, 50 V, Axial	ACC223KJCA	250002-722
C29	0.1 μF, 100 VDC, Polyester	ACC104JLGP	251002-610
C30	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C31	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C32	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C33	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C34	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C35	100 μF, 16 V, Electrolytic, Axial	ACC107WDAA	261002-310
C36	100 μF, 16 V, Electrolytic, Axial	ACC107WDAA	261002-310
C37	10 μF, 16 V, Electrolytic, Axial	ACC106WDAA	261002-410
C38	0.1 μF, 50 V, Z5U, Radial	ACC104ZJCF	250002-610
C39	330 pF, 50 V, Radial	ACC331KJCP	250001-133
	CONNECTORS		
J1	Header, 5-pin, Right Angle	AJ6815	312000-005
J2	On PC Board	————	————
J3	Header, 23-pin Right Angle	AJ7049	312000-023
J4	Housing, 4-pin, PC Mount, Power	ART2738	313000-004
J5	Header, 5-pin, Right Angle	AJ6815	312000-005

PARTS LIST (Cont'd)
Logic/Servo Controller Board #995050001

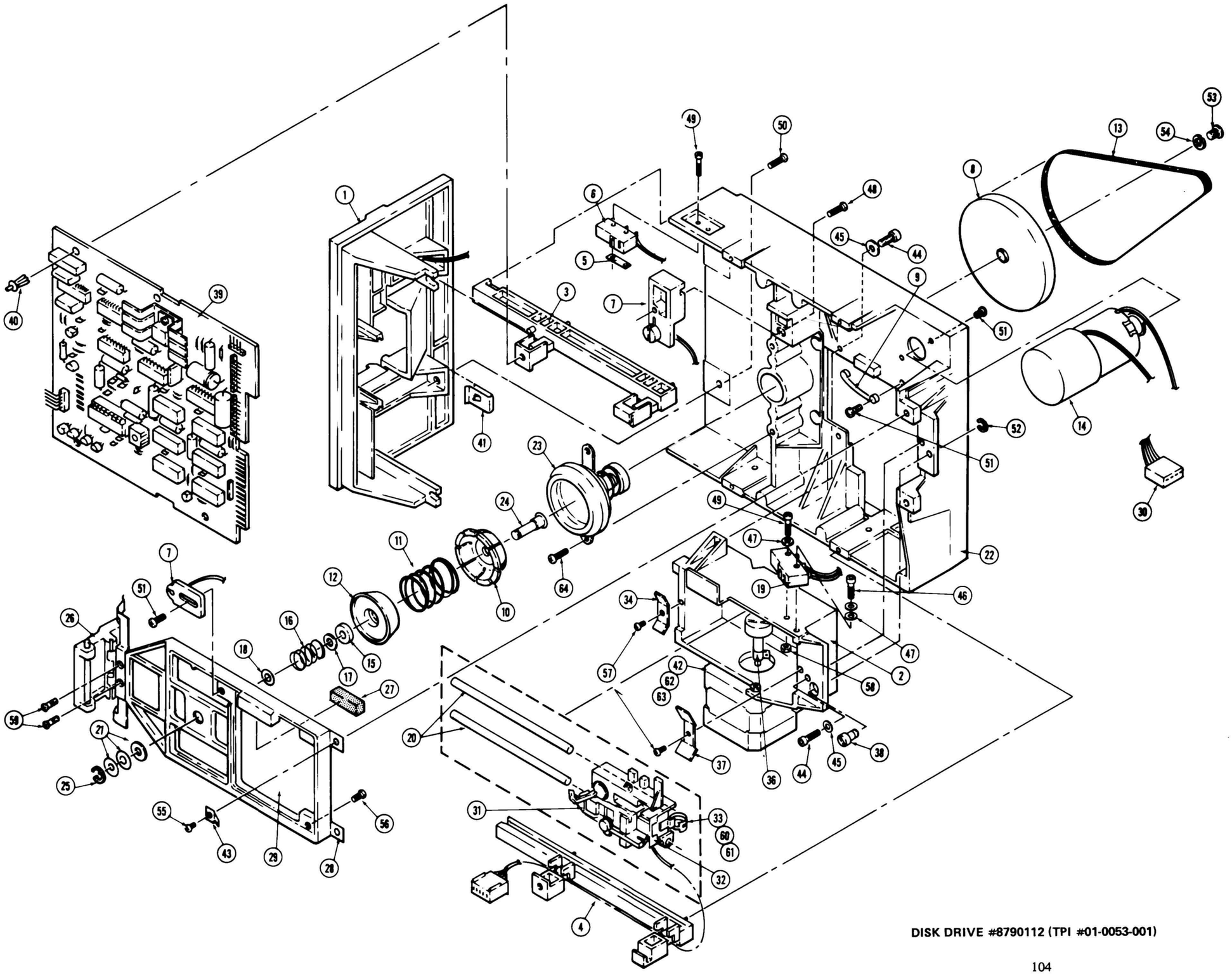
Symbol	Description	Manufacturer's Part Number	Radio Shack Part Number
	DIODE		
CR1	1N5240B, 10%, Zener	ADX1582	600008-001
	INDUCTORS		
L1	390 μH, 10%, Axial	ACA8058	461391-010
L2	390 μH, 10%, Axial	ACA8058	461391-010
L3	5.6 μH, 10%, Axial	ACA8194	461506-010
L4	5.6 μH, Axial	ACA8194	461506-010
	INTEGRATED CIRCUITS		
U1	74LS123	AMX3803	610002-123
U2	MC3470	AMX4467	610010-470
U3	74LS74	AMX3558	610002-074
U4	74LS26	AMX4522	610002-026
U5	7438	AMX3683	610002-038
U6	74LS11	AMX3554	610002-011
U7	74LS86	AMX3701	610002-086
U8	LM2917	AMX4181	610020-917
U9	74LS14	AMX3716	610002-014
U10	74LS14	AMX3716	610002-014
U11	74LS74	AMX3558	610002-074
U12	ULN2003A	AMX4513	610010-413
	RESISTORS		
R1	62 ohms, 1/2 W, 5%	AN0110EFC	550003-062
R2	11 K, 1/4 W, 2%	AN0285CEE	540300-113
R3	24 K, 1/4 W, 2%	AN0526CEE	540300-243
R4	10 K, 1/4 W, 5%	AN0281EEC	550001-310
R5	2.2 K, 1/4 W, 5%	AN0216EEC	550001-222
R6	2.2 K, 1/4 W, 5%	AN0216EEC	550001-222
R7	4.7 K, 1/4 W, 5%	AN0247EEC	550001-247
R8	475 ohms, 1/4 W, 1%	AN0627BEE	540104-750
R9	330 ohms, 1/4 W, 2%	AN0159CEE	540303-303
R10	475 ohms, 1/4 W, 1%	AN0627BEE	540104-750
R11	768 ohms, 1/4 W, 1%	AN0572BEE	540107-680
R12	768 ohms, 1/4 W, 1%	AN0572BEE	540107-680
R13	Not Used	———	———
R14	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R17	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R18	470 ohms, 1/4 W, 5%	AN0169EEC	550001-147
R19	470 ohms, 1/4 W, 5%	AN0169EEC	550001-147
R20	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R21	1.8 K, 1/4 W, 2%	AN0210CEE	540300-182
R22	50 K, Trimmer, 20%	AP7211	560150-302
R23	10 K, 1/4 W, 5%	AN0281EEC	550001-310

PARTS LIST (Cont'd)
Logic/Servo Controller Board #995050001

Symbol	Description	Manufacturer's Part Number	Radio Shack Part Number
R24	Not Used	———	———
R25	10 K, 1/4 W, 5%	AN0281EEC	550001-310
R26	240 ohms, 1/4 W, 5%	AN0151EEC	550001-124
R27	240 ohms, 1/4 W, 5%	AN0151EEC	550001-124
R28	Not Used	———	———
R29	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R30	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R31	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R32	0 ohms	AJ6814	550000-001
R33	3.3 K, 1/4 W, 2%	AN0230CEE	540300-332
R34	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R35	150 K, 1/4 W, 2%	AN0384CEE	540301-503
R36	470 ohms, 1/4 W, 5%	AN0169EEC	550001-147
R37	1 M, 1/4 W, 5%	AN0445EEC	550001-510
R38	3.3 K, 1/4 W, 2%	AN0230CEE	540300-332
R39	470 ohms, 1/4 W, 5%	AN0169EEC	550001-147
R40	1 M, 1/4 W, 5%	AN0445EEC	550001-510
R41	10 K, 1/4 W, 5%	AN0281EEC	550001-310
R42	1.5 K, 1/4 W, 5%	AN0206EEC	550001-215
R43	150 ohms, 1/4 W, 5%	AN0384CEE	540301-503
R44	10 ohms, 1/4 W, 5%	AN0063EEC	550001-010
R45	1 ohm, 1/2 W, 5%	AN0022EFC	550003-001
R46	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R47	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R48	2.2 K, 1/4 W, 5%	AN0216EEC	550001-222
R49	1 K, 1/4 W, 5%	AN0196EEC	550001-210
R50	20 K, 1/4 W, 5%	AN0306EEC	550001-320
R51	150 ohms, 6-pin SIP	ARX0241	570004-001
R52	56 ohms, 1/4 W, 5%	AN0013EEC	550001-056
R53	150 ohms, 1/4 W, 5%	AN0142EEC	550001-151
R54	1 K, Trimmer, 10%	AP7210	561501-301
R55	2.2 K, 1/4 W, 5%	AN0216EEC	550001-222
R56	150 ohms, 1/4 W, 5%	AN0142EEC	550001-151
	TRANSISTORS		
Q1	MPS3906	AMX3584	630007-001
Q2	MPS3906	AMX3584	630007-001
Q3	MPS3906	AMX3584	630007-001
Q4	MPS2907	AMX4187	630008-907
Q5	2N5460, FET	AMX4782	630006-001
Q6	2N5460, FET	AMX4782	630006-001
Q7	TIP110, Power	AMX4331	630003-110
Q8	2N4124	AMX4178	630002-001
Q9	2N4124	AMX4178	630002-001

PARTS LIST (Cont'd)
Logic/Servo Controller Board #995050001

Symbol	Description	Manufacturer's Part Number	Radio Shack Part Number
	MISCELLANEOUS		
TP1	Staking Pin	AHB9682	310024-001
TP2	Staking Pin	AHB9682	310024-001
TP3	Not Used	———	———
TP4	Not Used	———	———
TP5	Staking Pin	AHB9682	310024-001
TP11	Staking Pin	AHB9682	310024-001
(Q7)	Heat Sink, TO220	———	405351-001
(Q7)	Screw, 4-40 x 3/8", Phillips	AHD2249	411351-406
(Q7)	Nut, 4-40	AHD7227	410020-001
(R51)	Socket, 6-pin, SIP	AJ7048	311500-006



DISK DRIVE #8790112 (TPI #01-0053-001)

Parts List, Mini-Disk Drive Assembly #8790112
(TPI #01-0053-001)

Item	Quan	Mfg.Part No.	Description	RS PN
1	1	995010603	Front Panel Assembly	
2	1	105310001	Carriage Module	
3	1	995015003	Left Hand Guide Rail Assembly	
4	1	105157001	Right Hand Guide	8852023
5	1	400007001	Nut Plate, #2-56	
6	1	995223003	Write Protect Switch Assembly	8852017
7	1	995103011	Index Assembly (two parts)	
8	1	995111003	Timing Pulley Assembly	
9	1	105324001	Spring, Module Bias	
10	1	105166001	Cone, Thrust	
11	1	105626001	Spring, Cone Release	
12	1	105155001	Expander	
13	1	450007001	Belt, Drive	8852018
14	1	995102003	Drive Motor Assembly	8852015
15	1	450004001	Bearing, Cone	
16	1	105626002	Spring, Cone	
17	1	105304001	Spacer, Cone	
18	1	400009001	Washer, Shoulder	
19	1	995222003	Track 00 Switch Assembly	8852016
20	2	105333001	Shaft, Carriage	8852065
21	AR	400014001	Washer, Nylon	
22	1	105355001	Chassis	
23	1	995101003	Hub & Shaft Assembly	
24	1	105331001	Cone, Shaft	
25	1	400006001	E-Ring, .147" Shaft	8852050
26	1		Latch Plate Assembly	
	1	105002001	. Latch Plate	
	1	105027001	. Hinge, Latch Plate	
	1	105320001	. Latch Inhibitor	
	1	105360001	. Brass Pin	
27	1	105901001	Foam Tape Strip	8852063
28	2	105322001	Spring, Cone Lever	
29	1	105356001	Lever, Cone	
30	5	310016001	Connector, 4-Hole Molex	
	1	310015001	Connector, 5-Hole Molex	
		310017001	. Terminal Pin	8852046
31	1	995324001	Upper Arm Assembly	8852044
32	1	995234001	Head Carriage Assembly	8852031
33	1	105330001	Drive Band	8852032
34	1	105336001	Clamp, Shaft	8852064
35	1	105312001	Pulley, Stepper	
36	1	105313001	Collar	8852047
37	1	105335001	Track 00 Stop Clamp	
38	1	105314001	Cam Screw, Carriage Module	8852058
39	1	995050001	Logic/Servo PCB Assembly	

Parts List, Mini-Disk Drive Assembly #8790112
(TPI #01-0053-001)

Item	Quan	Mfg.Part No.	Description	RS PN
40	2	325500001	Rivet, Plastic	
41	2	409545632	Clip, Tinnerman	
42	1	995304001	Stepper Motor Assembly	
43	1	310018001	Ground Lug	
44	3	410002007	Screw, #6-32 x .375"	
45	3	410024001	Washer, #6 Flat	
46	1	410038001	Screw, #2-56 x 7/16"	
47	4	410024001	Washer, #2 Flat	
48	1	410009001	Screw, #6-32 x 3/8"	
49	3	417323208	Screw, #2-56 x 1/2"	
50	2	400001004	Screw, #6 Hex Head	
51	4	410008001	Screw, #6-32 x 1/4"	
52	1	400011001	E-Ring, Cam Screw	
53	1	410012001	Screw, #8-32 x 1/4"	8852056
54	1	410006042	Washer, #8 Split Lock	
55	2	410008001	Screw, #6-32 x 1/4"	
56	2	419351605	Screw, #6-32 x 5/16"	
57	2	410013001	Screw, #4-40 x 3/16"	
58	1	410021001	Nut, #2-56	
59	2	414310605	Screw, #6-32 x .312"	
60	2	400188225	Washer, #2 Flat	
61	2	410001007	Screw, #2-56 x .312"	
62	2	404351001	Washer, #4 Starlock	
63	2	410020001	Nut, #4 x .245" AF	
64	2	410011001	Screw, #6-32 x 3/16"	

SECTION VI

POWER SUPPLY

6.1 POWER SUPPLY #8790021, 38W (ASTEC #AA11320)

6.1.1 General

One of the power supplies used in the Model 4 is a 38W switching mode power supply utilizing the flyback method of power conversion. This supply has built-in EMI filter and has over-voltage and over-current protection as well as regulation over line and load variations.

6.1.2 Theory Of Operation

PRIMARY

Power is taken from a 115 volt AC line. A UL approved fusable resistor, R25, limits inrush current as well as acting as a system fuse. The AC current is taken through an EMI filter which suppresses power supply and system noise that would otherwise be reflected back into the power lines.

After the EMI filter, the signal goes to a full-wave bridge rectifier (DB1) then into a capacitive input filter. The combination of these last two items produces a 165 volt DC B+ line from which a DC-DC converter is run.

The DC-DC converter consists of a transistor (Q2) which is used to chop the B+ line at approximately 20 kHz. D2, R9, C8, C9, and D3 are all part of a snubber network designed to protect Q2 as well as suppress ringing which would contribute to line conducted EMI.

SECONDARY

When the power transistor (Q2) is 'ON', energy is stored in the core of the power transformer (T2). The secondary windings are polarized so that the output rectifiers do not conduct while Q2 is ON. When Q2 turns 'OFF', it causes the transformer to 'flyback' which causes the polarity on the output windings of T2 to reverse, allowing the output rectifiers to conduct and deliver energy to the output and output capacitors.

A pi filter consisting of capacitors on either side of a series filter choke is used to smooth the output waveform, deliver energy during the ON time, and reduce ripple.

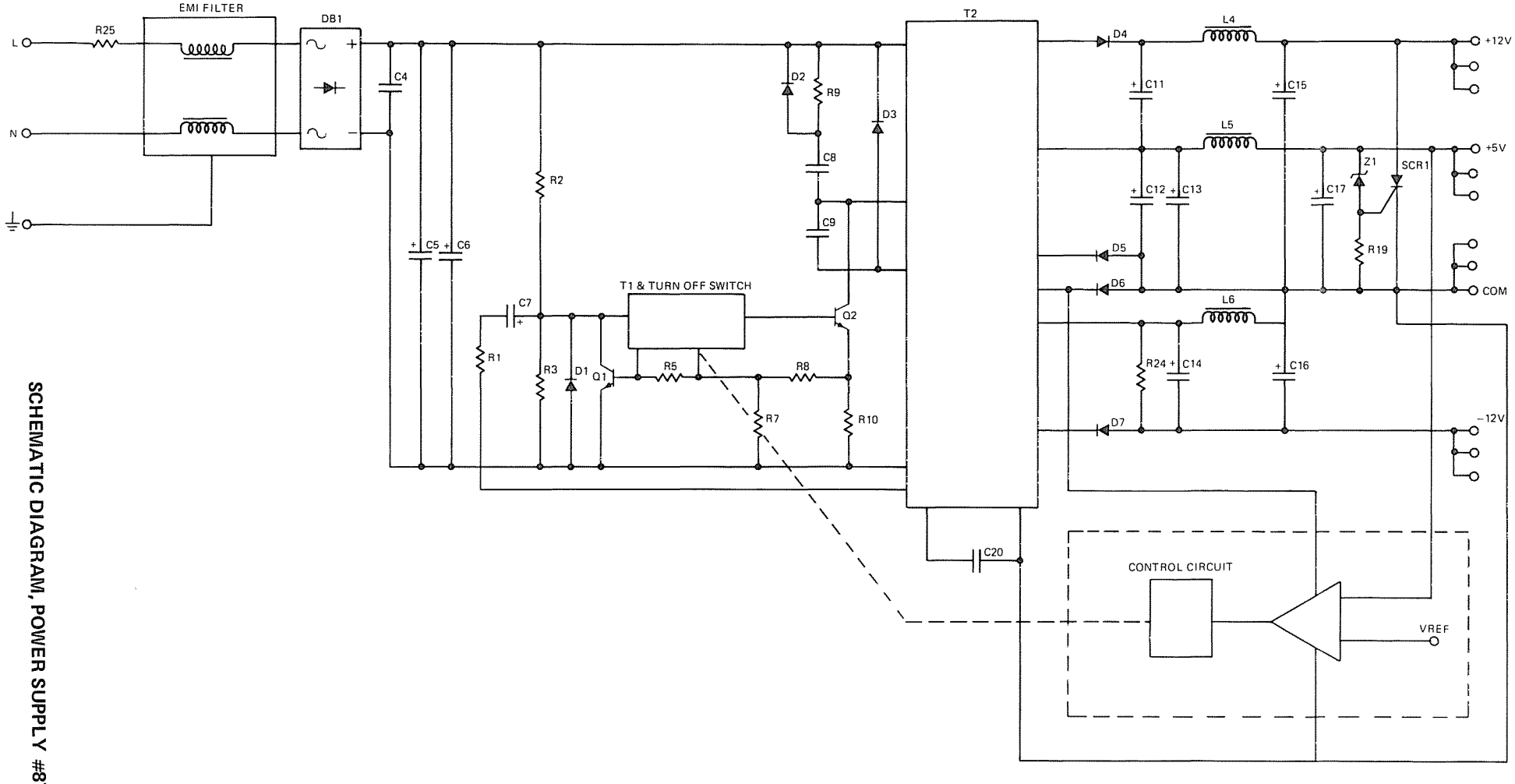
FEEDBACK AND CONTROL

On this power supply the only regulated voltage is the +5 volt output. The +12 volt and -12 volt outputs are simply additional secondary windings stacked on top of the +5 volt winding. They track the +5 voltage close enough to provide sufficient regulation to run the circuitry requiring +12 or -12 volts.

The +5 volt output is fed into an error amplifier which compares it to a reference voltage. The output of this amplifier is fed into a control circuit which is magnetically coupled through T1 to the turn-off circuit. This output also controls the ON time of Q2 providing voltage regulation by variations of the duty-cycle.

OVERVOLTAGE PROTECTION

If a failure in the power supply causes the outputs to rise uncontrolled past a specified voltage limit, the power supply will automatically shut down. This is done by sensing the +5 volt output via a Zener diode. When the voltage limit is reached, the SCR is triggered and shorts the +12 volt output to ground. A short on any output will cause the supply to foldback and shut down.



SCHEMATIC DIAGRAM, POWER SUPPLY #8790021

PARTS LIST, POWER SUPPLY #8790021 (ASTEC #AA11320)

SYMBOL	DESCRIPTION
	CAPACITORS
C1	2200pF, 250VAC, metallized paper
C2	0.01μF, 1KV, ceramic
C3	2200pF, 250VAC, metallized paper
C4	0.22μF, 100V, polyester
C5	47μF, 250V, electrolytic
C6	100μF, 250V, electrolytic
C7	220μF, 10V, electrolytic
C8	4500pF, 1KV, ceramic
C9	0.01μF, 1KV, ceramic
C10	0.22μF, 100V, polyester
C11	1000μF, 25V, electrolytic
C12	1000μF, 25V, electrolytic
C13	1000μF, 25V, electrolytic
C14	330μF, 16V, electrolytic
C15	2200μF, 16V, electrolytic
C16	330μF, 16V, electrolytic
C17	470μF, 25V, electrolytic
C18	0.022μF, 50V, polyester
C19	0.22μF, 100V, polyester
C20	3900pF, 400VAC, ceramic
C21	0.1μF, 250VAC, metallized
	DIODES
D1	RGP10A, rectifier
D2	RGP10D, rectifier
D3	RGP10J, rectifier
D4	Rectifier assembly
D5	Rectifier assembly
D6	Rectifier assembly
D7	RGP15B, rectifier
D8	1N4606, silicon
D9	1N4606, silicon
D10	1N4606, silicon
DB1	W06, bridge rectifier
Z1	5.6V, 5%, 1W, zener
	INDUCTORS
L1	Common mode choke assembly
L2	Base choke
L3	1.5mH, choke
L4	Choke coil assembly
L5	Choke coil assembly
L6	Choke coil
	INTEGRATED CIRCUIT
IC1	Regulator
	RESISTORS
R1	47 ohm, 5%, metal oxide film
R2	150K, 5%, carbon film
R3	1K, 5%, carbon film
R4	Not used
R5	82 ohm, 5%, carbon film
R6	27 ohm, 5%, carbon film
R7	3.3 ohm, 5%, carbon film
R8	10 ohm, 5%, carbon film
R9	27 ohm, 5%, metal oxide film
R10	75 ohm, 5%, 1W, metal film
R11	270 ohm, 5%, carbon film
R12	82 ohm, 5%, carbon film
R13	270 ohm, 5%, carbon film
R14	8.2 ohm, 5%, carbon film
R15	560 ohm, 5%, carbon film
R16	56 ohm, 5%, carbon film
R17	56 ohm, 5%, carbon film
R18	12K, 5%, carbon film
R19	12 ohm, 5%, carbon film
R20	470 ohm, 5%, carbon film
R21	2.7K, 2%, metal film
R22	2.7K, 2%, metal film
R23	68K, 5%, carbon film
R24	220 ohm, 5%, 1W, metal oxide film
R25	2 ohm, 5%, fusing metal film
R26	22K, 5%, carbon film
SCR1	SCR C122F
	TRANSFORMERS
T1	Control assembly
T2	Power assembly
	TRANSISTORS
Q1	PE8050B
Q2	Transistor assembly
Q3	PE8550B

6.2 Power Supply #8790043, 65 Watt (Astec AA12090)

6.2.1 Test Set-Up

A. Equipment Needed

1. Isolation Transformer (minimum of 500 VA rating). Dangerously high voltages are present in this power supply. So, for the safety of the individual doing the testing, please use an isolation transformer. The 500 VA rating is needed to keep the AC waveform from being clipped off at the peaks. These power supplies have peak charging capacitors and draw full power at the peak of the AC waveform.

2. 0-140 V Variable Transformer (Variac). Used to vary input voltage. Recommend 10 amp. 1.4 KVA rating minimum.

3. Voltage meter – Needed to measure DC voltages to 50 VDC and AC voltages to 200 VAC. Recommend two digital multimeters.

4. Oscilloscope – Need x10 and x100 probes.

5. Load board with connectors – See Table 6-1 for values of loads required. The entry on the table for safe load power is the minimum power rating for the load resistors used.

6. Ohmmeter.

B. Set-Up Procedure

Set up test equipment as shown in Figure 1. You will want to monitor the input voltage and the output voltage of the regulated bus, which is the +5V output, with DVM's. Also monitor the +5V output with the oscilloscope using 50mv/div sensitivity. The DVM monitoring the +5V output can also be used to check the other outputs. See the **No Output** section for test points within power supply.

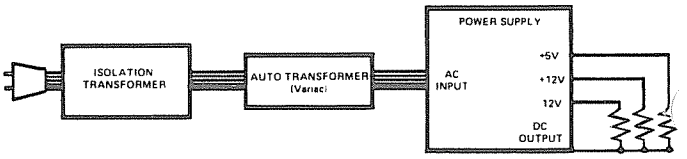


Figure 6-1. Test Set-Up

6.2.2 Visual Inspection

Check power supply for any broken, burned, or obviously damaged components. Visually check fuse, if any question check with ohmmeter.

6.2.3 Start-Up

Load power supply with minimum load as specified in Table I. Bring power up slowly with variable Transformer while monitoring +5V output with scope and DVM. Supply should start with approximately 40-60 VAC applied and should regulate when 90 VAC is reached. If output has reached +5 volts, refer to the **Performance Test** section. If there is no output, refer to the **No Output** section.

6.2.4 No Output

General

If the power supply does not produce an output with the AC input applied to the L and N connections and the power switch ON, one or more components have failed. A no-output-fault condition is most likely caused by a shorted/open component on the primary side but may also be caused by a short on the secondary. To determine this, follow the steps below.

A. Check Fuse:

If fuse is blown, replace but do not apply power until cause of failure is found.

B. Preliminary Check on Major Primary Components:

Check Diode Bridge (DB1), Power Transistor (Q2) and Catch Diode (D3) for shorted junctions. If any component is found shorted, replace.

C. Primary Check on Major Secondary Components

Using Ohmmeter from output common to each output, with output loads disconnected, check for shorted rectifiers or capacitors. If +12V output is shorted, also check crowbar SCR (SCR1).

D. Check B+ with the Fuse Intact

Connect power supply as in Figure 1 and attach x100 scope probe ground to the anode of D1. Slowly turn up power and check for B+ on end of R14 nearest the transformer. With input at 95 VAC, this point should be between 260 and 270 VDC. If this is not correct, check resistor and DB1.

If R14 is open it was most likely caused by a shorted component that is fed power by R14. Check the following components for proper operation (Q2, Q1, D1, D3).

E. Check Q2 Waveforms

Using x100 probe on Q2 heat sink, check collector waveform. Transistor should be switching, correct waveform is shown in Figure 2.

If this is not present check for open junctions on Q2. If Q2 is ok, check to see if base voltage is being supplied to Q2, it should be 0.7 volts. If it is not present, check components (L3, Q1, D1, and R4).

6.2.5 Low Outputs

A. All outputs are Low

If all outputs are low at the same time, check to ensure that the voltage selection jumper is in the proper position.

B. +5V and +12V (V3) Outputs

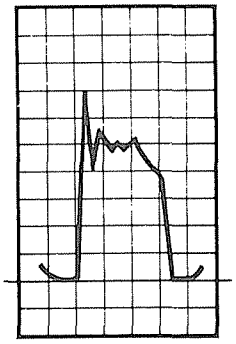
The power supply regulates off of the +5V and +12V (V3) outputs. If these outputs are low, it could cause the others to be low. If so, adjust +5V and +12V (V3) outputs by removing or adding R27 and R28.

C. Rectifier Check

If any one output is not present, first check the rectifier associated with that output and then the rest of the components in the circuit and the solder joints on the PCB.

6.2.6 Crowbar

If the crowbar is not operating, check Z1 and SCR1. If the crowbar is not triggering within the specified limits, change Z1.



50 V/DIV
5 μsec/DIV

Input – 120VAC
Loads – +5 @ 2A
+12 @ 1A
-12 @ 0.1A

Figure 6-2. Q2 Collector Waveform

1.0 V/DIV
5 μsec/DIV

Input and Loads same as above.

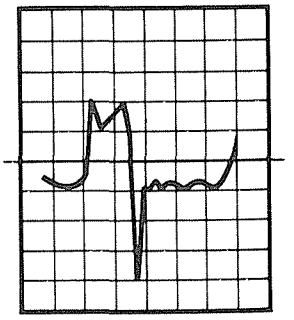


Figure 6-3. Q2 Base Waveform

6.2.7 Performance Test

Each of these test conditions should be set up and noted to be within the limits specified in Table 6-2.

Step	Input	+5V Load	+12V(V2)	+12V(V3)	–12V
1	90VAC	Max	Max	Max	Max
2	132	Max	Max	Max	Max
3	120	Max	Min	Min	Min
4	120	Min	Min	Max	Min
5	132	Min	Min	Min	Min
6	Test Crowbar Limits.				

If the power supply does not pass the above tests, refer to Section 6.2.4 and 6.2.5.

Table 6-1. Load Board Values

Output	Min Load	Load Resistance	Safe Load Power	Max Load	Load Resistance	Load Power
+5V	1.35A	3.7 ohms	12.5W	4.0A	1.25 ohms	50W
+12V-V2	0.40A	30 ohms	10 W	2.1A	5.7 ohms	50W
+12V-V3	0.60A	20 ohms	15 W	1.5A	8 ohms	35W
–12V	O A	1K ohms	1 W	0.1A	120 ohms	3W

Table 6-2. Voltage and Ripple Specifications

Output	Min	Max	Ripple (Max RP)
+5V	5V	5.25	50 MV
+12V	11.40V	12.60V	120 MV
+12V	11.40V	12.60V	120 MV
–12V	–11.40V	–12.60V	120 MV

Pin Assignments

AC Input:

TB1

Pin 1	Line
Pin 2	Neutral

DC Output:

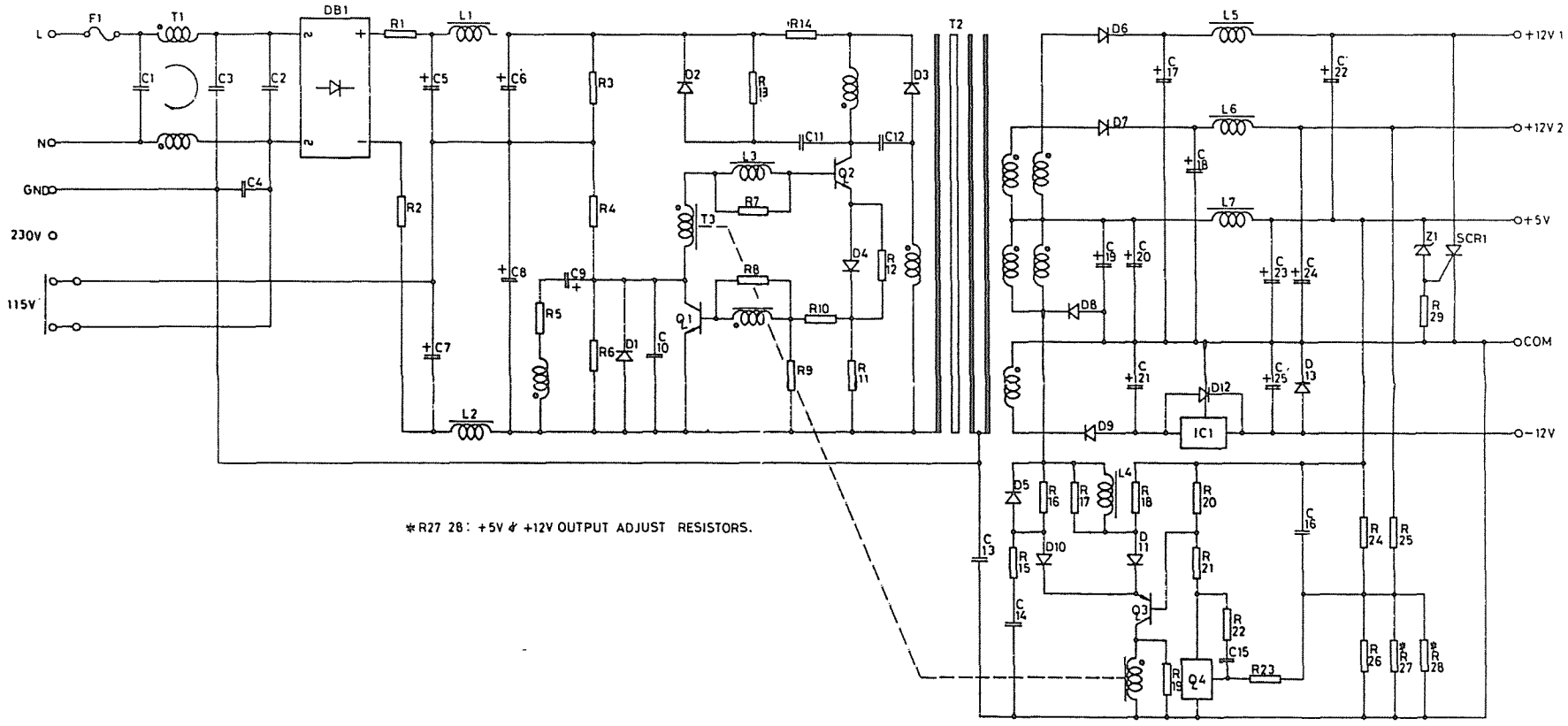
TB2

Pin 1	Common		Pin 7	V3	+12V
Pin 2	V2	+12V	Pin 8	V1	+5V
Pin 3	Key		Pin 9	V1	+5V
Pin 4	V4	– 12V	Pin 10	V1	+5V
Pin 5	V3	+12V	Pin 11	Common	
Pin 6	V3	+12V	Pin 12	Common	
			Pin 13	Common	

Mating Connectors are:

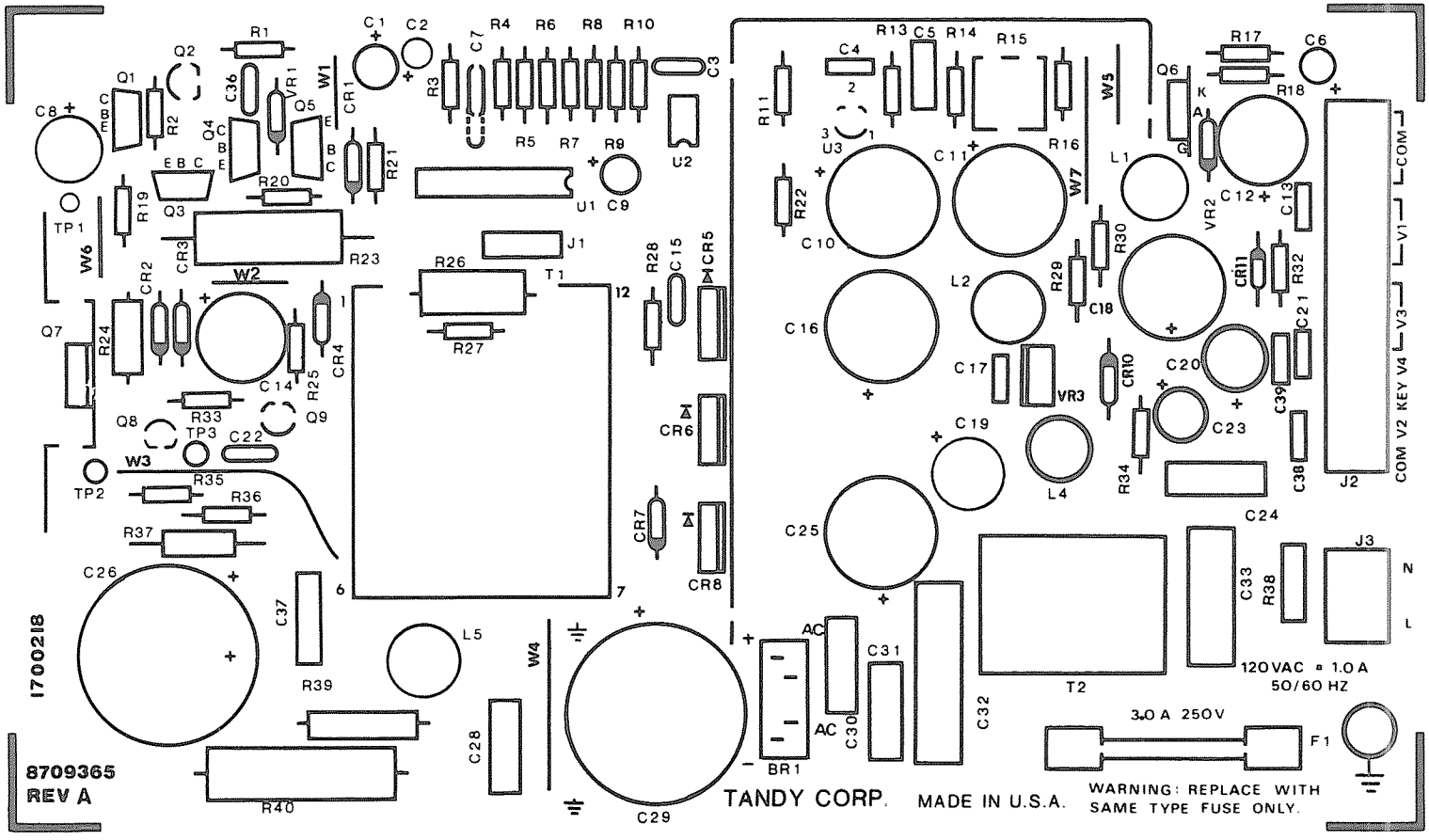
Molex Japan Co. Ltd.
AC Input (housing) 5239-09
DC Output (housing) 5265-12
Pin 5167

R	R1,2 R5 R3,4,6 R13,7,8,14,9,10,11 R12 R15,16,17 R18 R19,20,21,22 R23 R24,26,25,27,29,28
C	C1 C3 C4 C2 C5,7 C6,8 C9 C10 C11 C12 C13 C14 C19,20,21 C17 C18 C15,23,25,24,16,22
T/L	T1 L1 L2 T3 L3 T2 L4 L5 L6 L7
D/Q	D1 D2 Q1 Q2 D4 D3 D5,10,8,6,7,9 D11 Q3 Q4 D12 D13
MISC	F1 DB1 IC1 Z1 SCR1

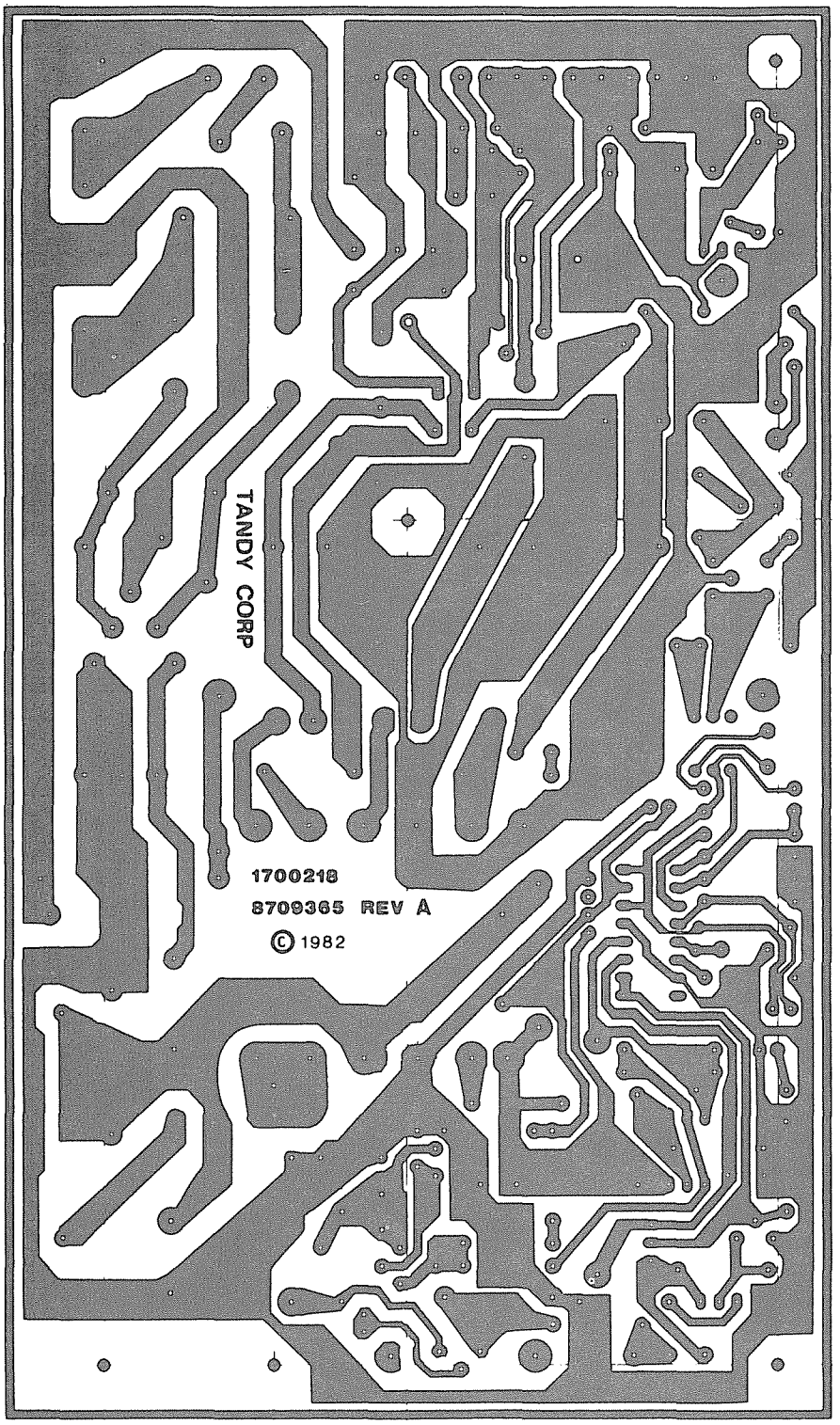


*R27 28: +5V & +12V OUTPUT ADJUST RESISTORS.

SCHEMATIC, POWER SUPPLY ASSEMBLY #8790043 (ASTEC AA12090)



COMPONENT LAYOUT #1700218, POWER SUPPLY ASSEMBLY #8790043



CIRCUIT TRACE #1700218, POWER SUPPLY ASSEMBLY #8790043

Parts List, Power Supply #8790043, (Astec AA12090)

Item	Sym	Description	Mfgr's Part No.
		Capacitors	
1	C1	0.01 ufd, +/- 20%, 250VAC, MPR	068-10300010-220
2	C2	0.1 ufd, +/- 20%, 250VAC, MPR	068-10400010-220
3	C3	4700 pfd, +/- 20%, 400VAC, Ceramic	055-47220001-189
4	C4	4700 pfd, +/- 20%, 400VAC, Ceramic	055-47220001-189
5	C5	100 ufd, +/- 20%, 250V, Electro.	057-10120170-141
6	C6	47 ufd, +/- 20%, 250V, Electro.	057-47020040-192
7	C7	100 ufd, +/- 20%, 250V, Electro.	007-10120170-141
8	C8	47 ufd, +/- 20%, 250V, Electro.	057-47020040-192
9	C9	220 ufd, +/- 20%, 10V, Electro.	057-22110300-190
10	C10	0.01 ufd, +/- 20%, 100V, Ceramic	055-10382125-111
11	C11	470 pfd, +/- 20%, 3kV, Ceramic	055-47167728-111
12	C12	0.01 ufd, +/-20%, 1kV, Ceramic	055-10368925-111
13	C13	0.01 ufd, +/- 20%, 250VAC, MPR	068-10300010-220
14	C14	0.22 ufd, +/- 10%, 100V, Polyester	058-22400240-189
15	C15	0.022 ufd, +/- 20%, 100V, Polyester	058-22300080-220
16	C16	0.22 ufd, +/- 10%, 100V, Polyester	058-22400240-189
17	C17	1000 ufd, +/- 20%, 16V, Electro.	057-10220100-192
18	C18	1000 ufd, +/-20%, 16V, Electro.	057-10220100-192
19	C19	1000 ufd, +/-20%, 16V, Electro.	057-10220100-192
20	C20	1000 ufd, +/- 20%, 16V, Electro.	057-10220100-192
21	C21	330 ufd, +/- 20%, 16V, Electro.	057-33120120-192
22	C22	2200 ufd, +/-20%, 16V, Electro.	057-22220070-192
23	C23	2200 ufd, +/-20%, 16V, Electro.	057-22220070-192
24	C24	2200 ufd, +/- 20%, 16V, Electro.	057-22220070-192
25	C25	330 ufd, +/-20%, 16V, Electro.	057-33120120-192
		Diodes	
26	D1	RGP10B	226-10400070-118
27	D2	RGP10J	226-10400060-118
28	D3	RGP10M	226-10400100-118
29	D4	RGP15B	226-10100040-118
30	D5	1N4606	212-10700210-158
31	D6	Heatsink Assembly	853-60200650-000
32	D7	Heatsink Assembly	853-60200650-000
33	D8	Heatsink Assembly	853-60200650-000
34	D9	RGP10B	226-10400070-118
35	D10	1N4606	212-10700210-158
36	D11	1N4606	212-10700210-158
37	D12	1N4001GP	226-10400080-118
38	Z1	Zener, 5.6V, +/-5% @ 40mA	222-56086002-148
39	DB1	Bridge Rectifier KBP10	226-30500010-118
		Fuse	
40	F1	Fuse 2.5A, 250V, 3AG	084-00200060-217

Parts List, Power Supply #8790043, (Astec AA12090)

Item	Sym	Description	Mfgr's Part No.
		Inductors	
41	L1	Toroid	024-00000110-484
42	L2	Toroid	124-00000110-484
43	L3	Base Choke 2.2 uH	328-00100030-124
44	L4	Choke 1.5 mH	328-00100010-124
45	L5	Choke Coil	852-20100180-264
46	L6	Choke Coil	852-20100180-264
47	L7	Filter Choke Coil	852-10100370-264
		Resistors	
48	R1	4 ohm, +/- 10%, Thermister	258-40970015-152
49	R2	4 ohm, +/- 10%, Thermister	258-40970015-152
50	R3	100K ohm, +/- 5%, 1W, Metal Film	247-10036054-156
51	R4	100K ohm, +/-5%, 1W, Metal Film	247-10036054-156
52	R5	33 ohm, +/-5%, 2W, Metal Oxide Film	248-33006063-189
53	R6	820 ohm, +/-5%, 1/4W, Carbon Film	240-82106022-152
54	R7	5.6 ohm, +/- 5%, 1/4W, Carbon Film	240-56906022-152
55	R8	47 ohm, +/- 5%, 1/4W, Carbon Film	240-47006022-152
56	R9	5.6 ohm, +/- 5%, 1/4W, Carbon Film	240-56906022-152
57	R10	10 ohm, +/- 5%, 1/4W, Carbon Film	240-10006022-152
58	R11	0.47 ohm, +/-5%, 1W, Metal Film	247-04786054-156
59	R12	5.6 ohm, +/- 5%, 1/4W, Carbon Film	240-56906022-152
60	R13	120 ohm, +/-5%, 1W, Metal Oxide Film	248-12106052-189
61	R14	1 ohm, +/- 5%, 1W, Metal Film	247-10086054-156
62	R15	39 ohm, +/-5%, 1/4W, Carbon Film	240-39006022-152
63	R16	270 ohm, +/-5%, 1/2W, Carbon Film	240-27106033-152
64	R17	270 ohm, +/-5%, 1/2W, Carbon Film	240-27106033-152
65	R18	8.2 ohm, +/-5%, 1/4W, Carbon Film	240-82906022-152
66	R19	330 ohm, +/-5%, 1/4W, Carbon Film	240-33106022-152
67	R20	56 ohm, +/-5%, 1/4W, Carbon Film	240-56006022-152
68	R21	56 ohm, +/-5%, 1/4W, Carbon Film	240-56006022-152
69	R22	12K ohm, +/-5%, 1/4W, Carbon Film	240-12306022-152
70	R23	470 ohm, +/-5%, 1/4W, Carbon Film	240-47106022-152
71	R24	4.7K ohm, +/-2%, 1/4W, Metal Film	247-47015022-189
72	R25	22K ohm, +/-2%, 1/4W, Metal Film	247-22025022-189
73	R26	2.7K ohm, +/-1%, 1/4W, Metal Film	247-27014022-189
74	R27	100K ohm, +/-5%, 1/4W, Carbon Film	240-10406022-152
75	R28	100K ohm, +/-5%, 1/4W, Carbon Film	240-10406022-152
76	R29	12 ohm, +/-5%, 1/4W, Carbon Film	240-12006022-152
		Transformers	
77	T1	Common Mode	852-20200120-264
78	T2	Power	852-10201340-000
79	T3	Control	852-10201510-000

Parts List, Power Supply #8790043, (Astec AA12090)

Item	Sym	Description	Mfgr's Part No.
		Transistors	
80	Q1	NPN, SD467	209-11700460-120
81	Q2	NPN, 2SC1358	209-30200020-143
82	Q3	PNP, SB561	210-11700350-120
83	Q4	Intergrated Circuit, TL431CLP	211-10800100-176

6.3 Power Supply #8790049, 65 Watt

6.3.1 System Description

Basic Principle

A switching power supply circuit employs a high-speed semiconductor switch to control the storage and release of electrical energy in an inductor and provide regulated DC output voltages with a minimum loss of energy in heat-dissipating elements. There are several schemes for achieving this result which differ primarily in the arrangement of the basic circuit elements. These elements include a switch, an inductor, a rectifier, a capacitor and a DC voltage source.

An arrangement well-suited for economical power supplies with rated power outputs under 100 watts is the FLYBACK CONVERTER shown in Figure 6-4. The waveforms in Figure 6-5 are used to describe the operation of the Flyback Converter circuit. For the purpose of this discussion we will assume that the duration of the "ON" time equals the duration of the "OFF" time.

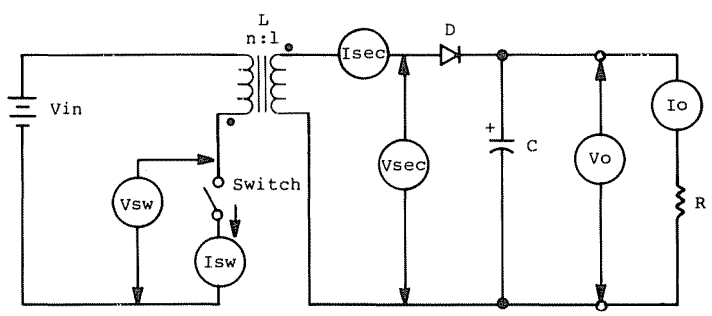


Figure 6-4. Basic Flyback Converter

When the switch is closed (ON) at time ta, Vin is impressed across the primary winding of inductor L and the current Isw increases linearly from zero until the switch opens (OFF) at time tb. Note that Isec is zero while the switch is closed. This is because Vsec is negative with respect to Vo thus reverse-biasing diode D. Note that Vsw is also zero while the switch is closed.

When the switch opens at time tb, the magnetic field of L instantly collapses and reverses polarity. At this moment, Vsw is equal to Vin plus the voltage across L just before the switch opened (also equal to Vin). Therefore, at the instant the magnetic field reverses polarity, Vsw = 2Vin.

During the interval when the switch is open (tb to tc), the secondary voltage, Vsec, is a replica of the primary voltage Vsw. Diode D is now forward biased due to the polarity of the inductor windings and because the turns ratio, n, is such that:

$$V_{sec} \times n > V_o$$

This biasing replenishes the charge in capacitor C that was delivered to the load R during the ta-tb interval. This is the "flyback" interval and is so named because the inductor releases the energy stored in its magnetic field while the switch is OFF.

Several other facts are illustrated by the waveforms of Figure 6-5. First, the voltage across the switch Vsw decays exponentially from 2Vin to Vin during the "OFF" interval. This is because the inductor and the switch timing are adjusted to transfer all of the energy that was stored in the inductor while the switch was ON, into the secondary while the switch is OFF. (Observe that Isec DECREASES linearly with time to zero at the end of the "OFF" time period.) This is known as resetting the core. Thus, at time tc when the switch is ready to turn on again, the DC input voltage Vin is again available to charge the inductor. Also at this time, all currents in the inductor are zero.

Second, since we have assumed that Isw increases linearly with time and that the ON and OFF time periods are equal (50% duty cycle), the average current in the primary, Isw (av), is 1/4 the peak current Isw. Also, the average current in the secondary, which is equal to the load current Io, is 1/4 the peak current in the secondary.

Third, the turns ratio is set by the ratio of the average primary voltage (Vsw) over a full cycle at its lowest value to the maximum permissible output voltage, Vo. The lowest Vsw value occurs at low AC line and maximum output load. In practice, the actual turns ratio, the ratio of peak-to-average voltages and currents, and the duty cycle may be adjusted to compensate for circuit losses.

Fourth, notice the ringing or oscillation that appears on the peak portion of Vsw and Vsec. This oscillation occurs at the resonant frequency of the leakage inductance of the inductor L and the parasitic capacitance of the circuit. The parasitic capacitance includes the interwinding capacitance of the inductor and stray capacitance of the switch. If this oscillation is not damped by a suitable means, the peak voltages may easily exceed the breakdown rating of the switch or the insulation in the inductor.

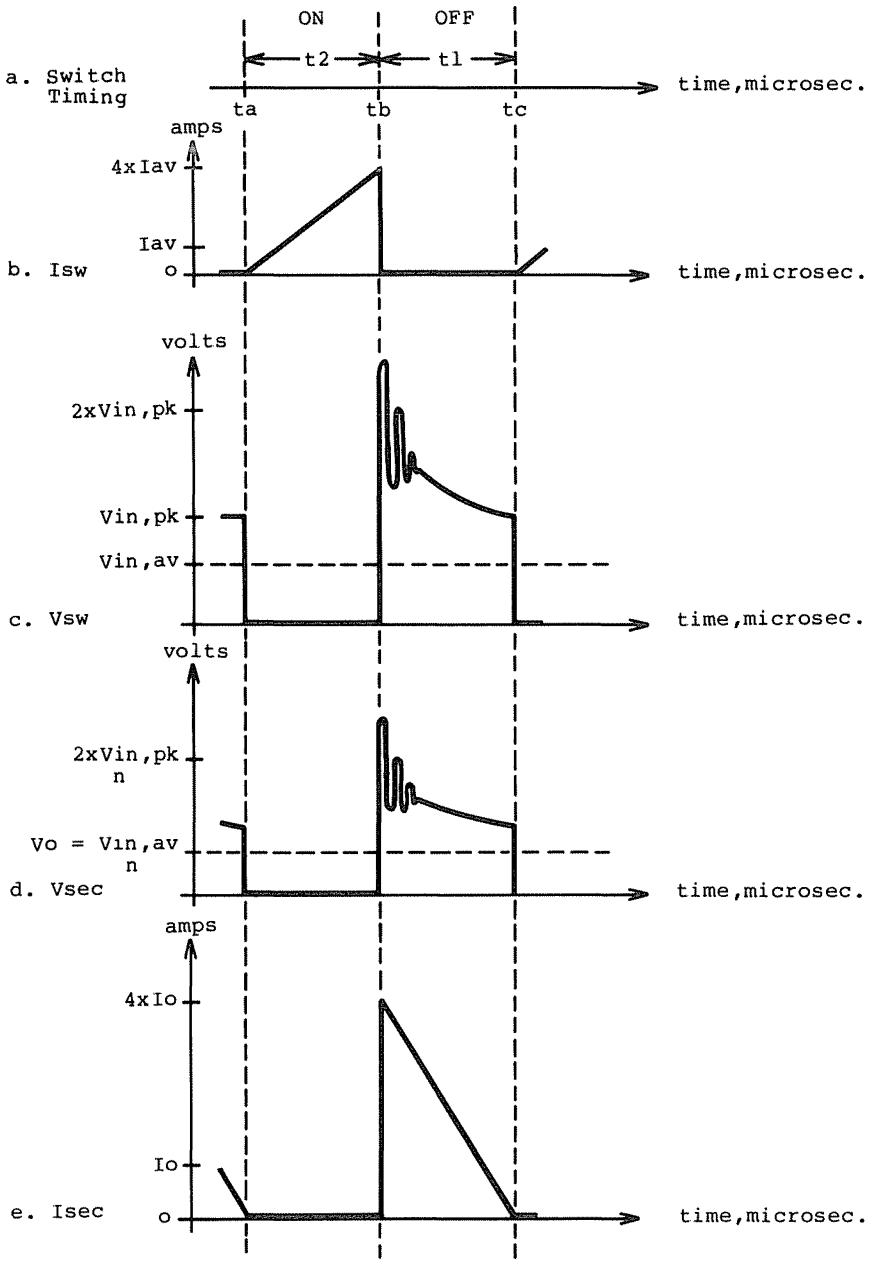


Figure 6-5. Waveforms for Figure 6-4.

Block Diagram

The basic circuit illustrated in Figure 6-4 can be divided into three functional blocks: Input DC supply, primary, and secondary. To make use of this model, we need to expand it to provide control for the switch timing and to include sufficient circuitry to satisfy performance and reliability specifications. The complete block diagram is shown in Figure 6-6.

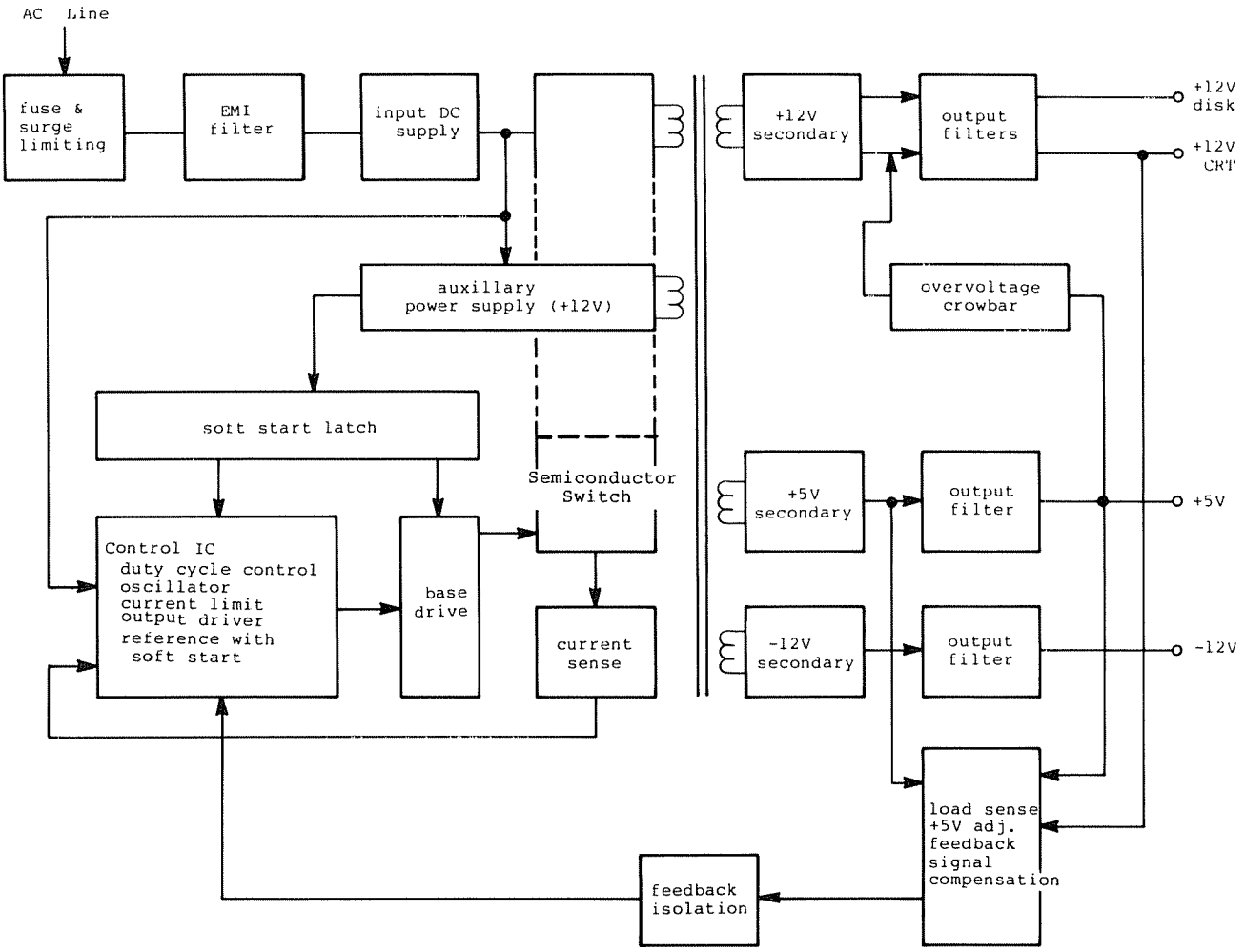


Figure 6-6. Block Diagram

The other blocks provide additional output voltages, add safety or protective features, reduce circuit noise, and develop signals for use by the control section. The control section continuously operates the bipolar transistor switch and varies the proportion of ON time to OFF time in response to changes in AC input line voltage or output load current. This is accomplished by feeding back a signal from the output terminals that instructs the control section to increase or decrease the ON time to compensate for a change in the output voltage.

The DC voltage supply to the control section is controlled by the latch circuit when AC power is first applied to the power supply. A built-in timing circuit allows the input DC supply filter capacitor to become fully charged before power is applied to the control section. After the control section circuit starts and secondary voltages reach their regulated output levels, the auxiliary power supply provides the required DC voltage to operate the control section. The latch is reset when the current limit or under-voltage sensors operate, thus removing DC voltage to the Control IC.

There are four secondary or output voltages in addition to the auxiliary supply: +5.05 volt, +12 volt CRT, +12 volt Disk, and −12 volt. The +5.05 and +12 DISK voltages are regulated by the control circuit response to the frequency compensated feedback control signal which comes from the load sense section. Since the load sensing occurs on the secondary side, an optional coupler circuit is necessary to provide safety isolation between the primary side common ground and the secondary side common ground.

All the secondary voltages, including the auxiliary +12 voltage, share the same magnetic flux linkage in the transformer core and are controlled by the flyback inductor. Any change in secondary load currents cause a change in the shared magnetic flux. This change in the flux of the inductor sets up an EMF (electromotive force) which causes a flux in opposition to the one which resulted from the change in load current. Thus, the original change tends to be counteracted and the current delivered to the load remains constant.

The output filters reduce the remaining ripple voltage components of the AC line and switching frequencies to levels low enough to prevent interference with the circuits operated by the supply. Switching frequency components that could be conducted out the AC input terminals are suppressed by the EMI filter to avoid interference with other equipment connected to the power line.

The overvoltage crowbar senses an abnormal rise in the +5.1 volt output and short-circuits the voltage line to the common secondary ground, thus tripping the current limiting circuit which finally shuts down the supply.

The surge limiter at the AC line input prevents the input filter capacitor in-rush current surge from exceeding component ratings or unnecessarily tripping external fuses.

6.3.2 Technical Specifications

Environment

Temperature; Operating	0° to 50° C (32° to 122° F)
Storage	–40° to 85° C (–40° to 185° F)
Humidity; Operating	85% r.h. @ 35 C (95° F) max.
Storage	95% r.h. @ 55 C (131° F) max.

Input Voltage:
90 to 135 VAC rms, 47 to 63 Hz

Input Surge Current:
48 amps max.

Efficiency:
70% min. at full load with 115 VAC rms input

Output Voltages:
V1, +5.05 VDC
V2, +12 VDC CRT
V3, +12 VDC DISK
V4, –12 VDC

Output Power:
continuous 65 watts max.

Output Current:

	Output	Load Min.	Load Max.
Condition 1 (Model III use)	V1	1.35 A	4.0 A
	V2	0.60 A	1.5 A
	V3	0.40 A	2.1 A
	V4	0.005 A	0.10 A
Condition 2 (Hard Disk use)	V1	2.5 A	5.0 A
	V3	0.75 A	2.0 A*
	V4	0.005 A	0.10 A

*NOTE: V2 and V3 connect in parallel to provide the V3 output. The V3 output will support a 5.0 A peak load which decays to 1.0 A in approx. 8 seconds. V1 and V3 must be within specified regulation when this surge decays to 4.0 A.

Output Ripple Voltage:

V1	(5.05 VDC)	50mV p-p
V2	(+12 VDC)	150mV p-p
V3	(+12 VDC)	150mV p-p
V4	(–12 VDC)	150mV p-p

NOTE: Ripple is the composite 100/120 Hz ripple due to the line, plus the high frequency ripple due to the power oscillator. Common mode noise which may be observed due to oscilloscope connections should be ignored.

Output Voltage Regulation:
After initially setting V1, output voltage tolerances under all conditions of rated line, load, and temperature should remain within the following limits:

V1	(+5.05 VDC)	+/– 3%
V2	(+12 VDC)	see *NOTE
V3	(+12 VDC)	+/– 5%
V4	(–12 VDC)	+25%, –8.3%

*NOTE:
a) The initial value of V2 must not change by more than +/– 100mV under the following load conditions of V3:
– A step increase in output current from 0.6 A (initial condition) to 2.4 A, decaying within 60 msec to 2.1 A.
– A step decrease in output current from 2.1 A (initial condition) to 0.6 A.
b) V2 output voltage may vary +/– 5% under all other conditions of rated line, load, and temperature as defined in the specification.

Over-Current Protection:
Power supply will shut down before total power exceeds the point where damage would result. No damage will result when any output is short circuited continuously with 100 milliohms or less.

Over-Voltage Protection:
The +5.05 VDC circuit is protected with a "crowbar" circuit with a trip range of 5.8 to 6.8 VDC.

Hold-Up Time at Continuous Max Load:

Nominal Line	16 mSec minimum
Low Line	10 mSec minimum

6.3.3 Theory of Operation

The basic operating principles of a flyback converter and the necessary functional blocks to form a complete power supply were reviewed in the System Description section. In this part, the operation of each section of the circuit will be analyzed and later these sections will be connected to illustrate the signal flow in the power supply.

AC INPUT

A conventional bridge rectifier and a filter capacitor are connected directly across the AC line to provide the DC input voltage to the power supply.

An EMI filter consisting of capacitors C30-C33 and choke T2 are inserted at the input to the rectifier. This filter circuit keeps the high frequency signals generated in the power supply from being conducted into the AC power line. C30 and C31 provide a low impedance to the earth ground terminal for signals common to both hot and neutral sides of the AC line. C32 provides a low impedance dissipative path for the RF signal energy which appears across the line. T2 blocks RF signals common to both sides of the line and reflects them back toward the lower impedance elements near the rectifier. T2 also helps block differential (across-the-line) signals by using the EMF set up by the signal current on one side of the line to oppose the signal current flowing in the other side. C33 serves as a transient bypass capacitor to protect the power supply from large transient voltages that appear on the AC power line. C33 also improves the efficiency of the RFI filter choke T2 by terminating the line in a low impedance to absorb and dissipate any remaining differential RF energy.

R38 is a negative-temperature-coefficient-themistor which limits the turn-on surge current of the power supply filter capacitor C29. The resistance of this thermistor when "cold" is approximately 10 ohms. As the filter capacitor charges toward the peak value of the AC input voltage, it draws less current from the line. At the same time, the heating effect of the current flowing in the thermistor causes its resistance to decrease until it reaches its rated "hot" resistance of less than 1 ohm. As you can see, the thermistor dissipates very little power when the power supply is in operation. The thermistor is designed to cool rapidly enough, during power loss or turn-off, to limit the turn-on surge after only a few seconds cool-down.

The fuse, a fast acting 3.0 amp unit, is selected to ignore the short term turn-on surges, but open quickly in the event of an abnormally high current that would result from a component failure in the DC input supply or current limiting circuits.

Auxiliary Power Supply

The auxiliary power supply is operational when the main supply is on and not in a shut-down condition. This power supply consists of winding 2-3 on T1, half-wave rectifier CR4, and filter capacitor C14. The voltage output is approximately +15 volts under normal conditions but momentarily reaches about +31 volts during start-up.

Kick Start Latch

Start up of the circuit is initiated by the kick start latch. This latch is shown in simplified form in Figure 5a along with the accompanying waveforms in Figure 5b. When power is applied, C14 charges toward Vin = +160 volts through R26 with a time constant of approximately RC or 37.5 seconds. However, as we'll see, the kick start latch turns on in 2 or 3 seconds, the time required for the voltage across C14 to reach 30 + Vbe4 = 30.7 volts. At this point Q4 turns on and develops a bias across R21 which turns on Q5.

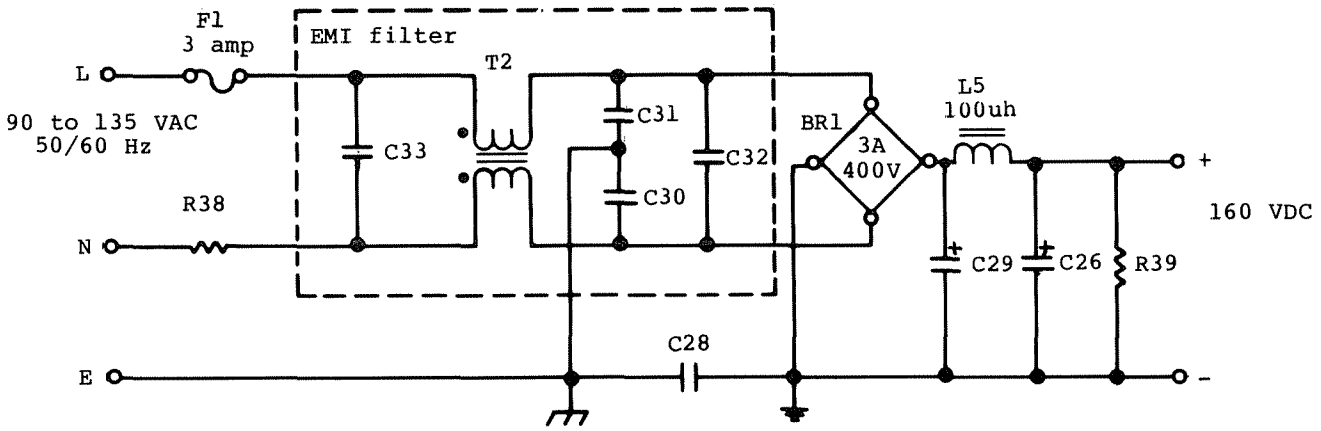


Figure 6-7. Input AC Supply

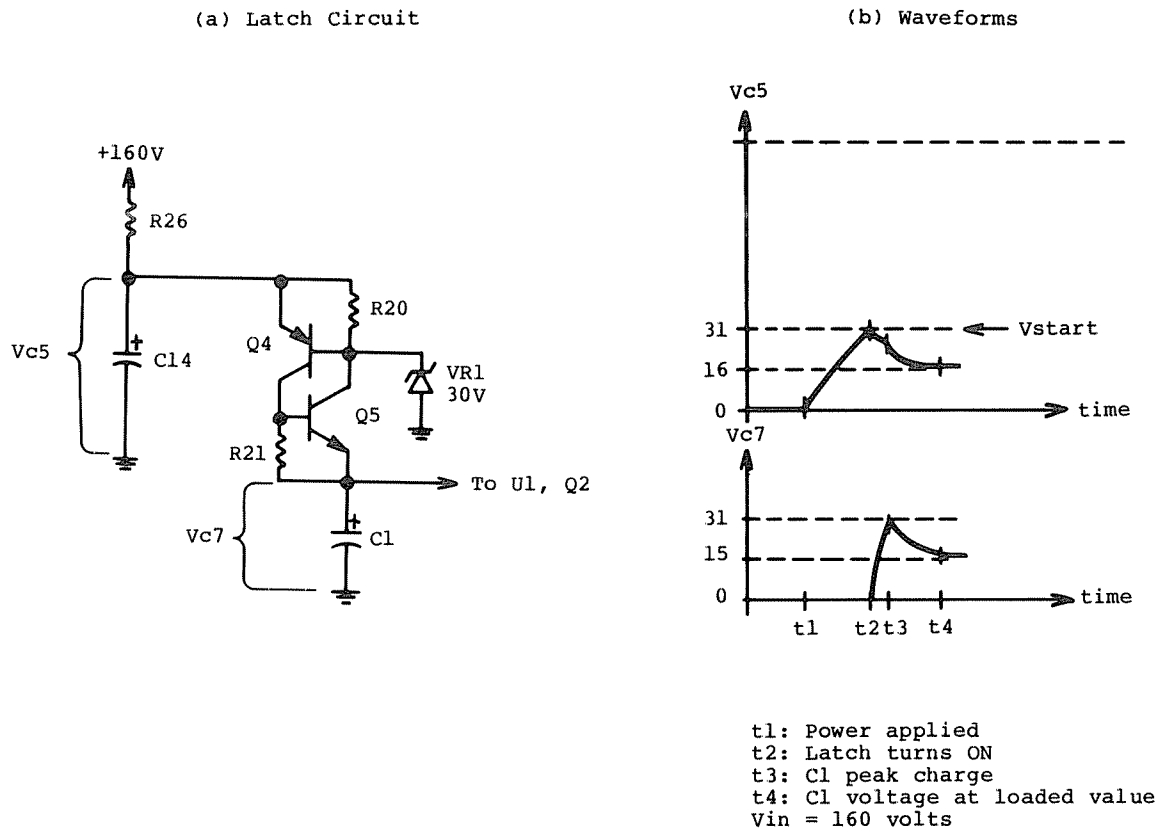


Figure 6-8. Kick-Start Latch

Referring to Figure 6-8b, as C14 dumps its charge into C1 beginning at time t2, the voltage across C14 starts to decrease toward a level that will be determined by the load composed of U1 and the base drive circuit. Notice that the voltage across C1 momentarily approaches the full 31 volts at time t3 before it drops down under load to about +15 volts at time t4.

With C1 charging rapidly through the low resistance of a saturated Q4 via Vbe5, the reference supply inside U1 develops its 5.0 volt output when the voltage across C1 exceeds about 8 volts. At this point, the supply has not quite yet started, but U1 has a DC supply at pin 10. All that remains is to start up the pulse generator so that the supply operates and replenishes the charge in C14 on each cycle, thus maintaining a DC source for U1 of about +15 volts. Completion of the start-up sequence occurs when the soft start circuit, described in the next section, has started the pulse generator.

Control Section

The control section consists of the control IC, the primary half of the feedback optocoupler U2, and the base drive circuit for the switching transistor. The control circuit IC has three major parts: an internal regulator, a pulse generator, and an error amplifier section.

The internal reference is a regulated +5.0 DC voltage. This voltage provides the reference voltages for the comparators used in the pulse generator as well as the DC supply voltage for the feedback optical coupler and the internal circuits of U1 except for its output transistors.

The pulse generator section of the control IC has four major parts: (a) sawtooth oscillator; (b) wave-shaping and output circuit; (c) regulating comparator; (d) dead-time comparator. Figure 6 illustrates the sawtooth oscillator and output circuit waveforms and the approximate levels of the DC control voltages applied by the comparators to the wave-shaping logic. The oscillator frequency is set by the values of R3 and C7 shown in Figure 7.

The amplitude of the sawtooth is set at 3.0 volts (approximately 60% of the 5.0 volt reference voltage). Whenever the sawtooth voltage, Vosc, exceeds both of the DC control voltages, Vreg and Vdt, the output circuit will be in the ON condition.

The DC control voltage, Vreg, set at a quiescent value by R6 and R9, varies in response to changes in the supply's DC output voltages as sensed by U3 and coupled through U2. Notice that these voltages will vary because of changes in output loading, AC input voltage, and also because of the residual 120 Hz ripple component from the main DC supply.

The dead-time control voltage, Vdt, is set at a constant value by R4 and R5 and ensures that the pulse generator "OFF" time will be at least 50% of the sawtooth period. This allows adequate time for the complete transfer of stored energy from the primary to the secondary of transformer T1 as discussed in the section on basic principles.

A concept known as duty cycle was introduced in earlier paragraphs. Duty cycle is defined as the ratio of the "ON" time of the sawtooth cycle to the total length of the sawtooth period. Since the sawtooth has a linear ramp characteristic, the duty cycle is also equal to:

$$\text{duty cycle } d = \frac{V_{osc,pk} - V_{reg}}{V_{osc,pk}} = \frac{t_{on}}{T_{period}}$$

There are three possible conditions of the duty cycle:

- $d = 0$ which occurs when either control voltage Vreg or Vdt exceeds the peak value of the sawtooth waveform Vosc
- $d = 50\%$ which occurs when Vreg is less than Vdt. This happens when the loading on the output of the supply is heaviest and the AC input voltage is at its lowest permitted level (see specifications)
- $0 < d < 50\%$ which occurs during normal operation.

The dead-time control voltage is used in one other important way. Notice the 4.7 μfd capacitor, C2, connected across R4 in Figure 6-10. When power is first applied to the supply, the voltage across the capacitor is zero. Therefore, Vdt = Vref = 5.0 volts and no pulses appear at the output because Vdt is greater than Vosc,pk. As C2 charges, Vdt decreases toward 1/2 (Vosc,pk) in a time determined by R5 and C2 as t = 5x15k ohm X 4.7 μfd = 1/3 second. As Vdt decreases past Vosc, pk, very narrow pulses begin appearing at pin 8 of U1. The pulses become successively wider until Vdt is less than Vreg. C2 continues charging until Vdt reaches the final correct value of about 1.5 volts. This action provides the soft start feature of the power supply and allows sufficient time for the DC input supply and latch to reach normal operating conditions before the supply is started. In effect, the load is connected to the supply gradually by the soft start circuit.

Frequency stability of the sawtooth oscillator is provided by the 2% tolerance and polyester construction of the timing capacitor, C7, and the 100 parts-per-million temperature stability and 1% tolerance of R3. Voltage stability of the DC control voltages is provided by the +/– 2 1/2 percent stability of the 5.0 volt reference.

The control section consists of two error amplifiers in U1, the primary half of U2, and associated circuitry shown in Figure 6-10. One of the error amplifiers serves as a regulator or pulse-width modulator which derives the DC control voltage, Vreg, from the signal voltage developed across R7 by the current in U2.

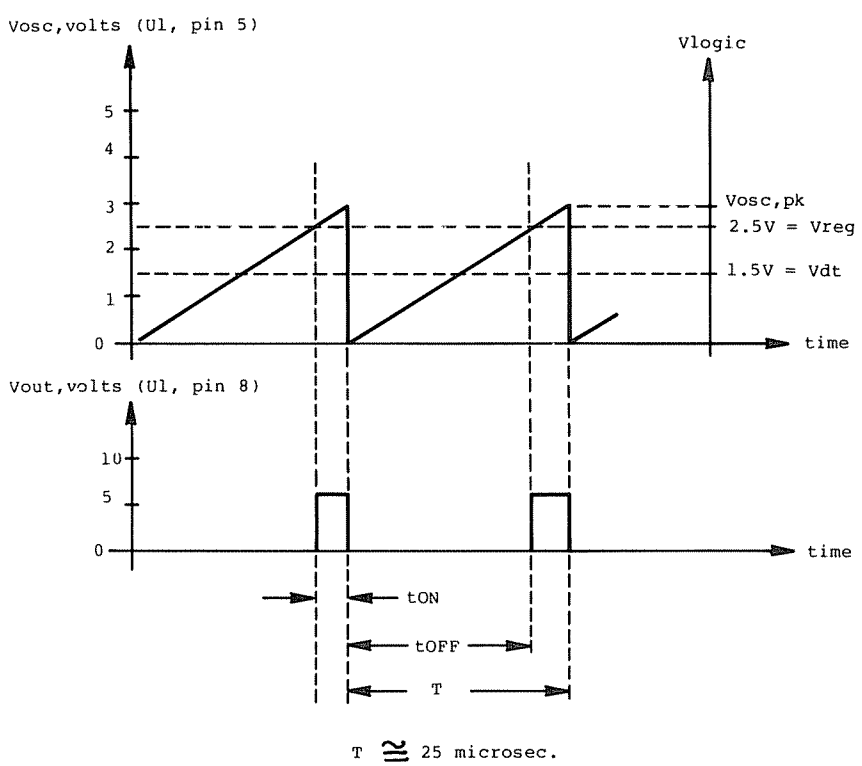


Figure 6-9. Oscillator, Pulse Generator Waveforms

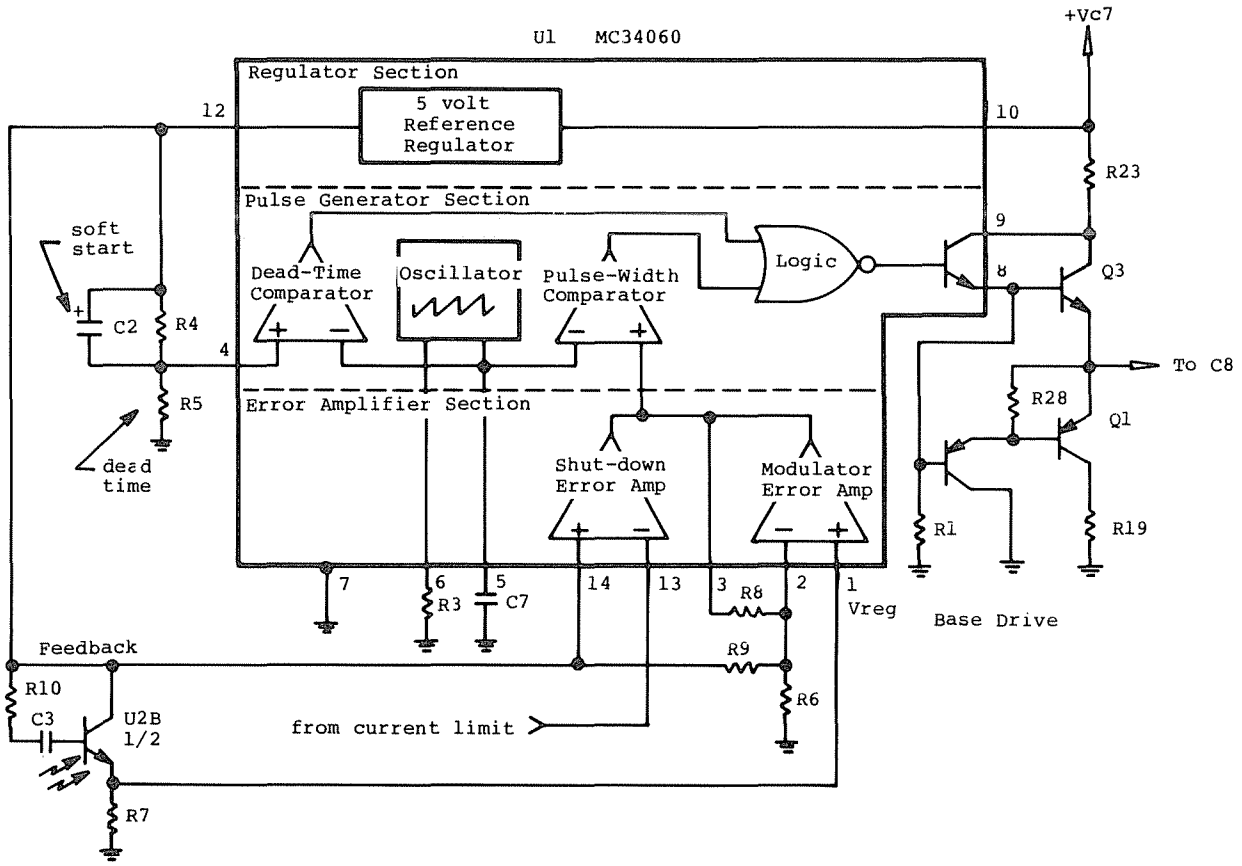


Figure 6-10. Control Section

This current is a replica of the current developed by U3 in response to the condition of the output voltage at the +5.1v and +12v outputs. This amplifier has a gain of about 10 determined by:

$$A = \frac{R8}{R9 \parallel R6} = \frac{22k\ ohm}{2.35k\ ohm} = 10$$

The other error amplifier in U1 serves as a shut-down comparator. The positive terminal, pin 14, is set at the +5.0 volt reference and pin 13, the negative terminal or shut-down pin, is tied to the current limit latch. The output of this error amplifier (equal to Vreg since both error amplifier outputs are tied to the wave-shaping logic) will rapidly increase toward the +5.0 volt reference when pin 13 drops below 5.0 volts. Recall that if Vreg exceeds the peak sawtooth voltage, pulses are inhibited and the power supply shuts-down.

Base Drive

Figure 6-11 illustrates the BASE DRIVE circuitry which turns switching transistor Q7 on and off in response to the output of the pulse generator portion of U1. The "ON" circuit is shown in Figure 6-11a and the "OFF" circuit is shown in Figure 6-11b. Waveforms for these circuits appear in Figure 6-12.

The output transistor of U1 combined with Q3 forms a Darlington pair. This circuit provides the relatively large current necessary (through coupling capacitor C8) to turn on Q7. R23 limits this base current to a value large enough to turn on Q7 quickly, but not so large that it will exceed the ratings of Q3, C8, or the base emitter junction of Q7, or so large that the turn-off time of Q7 is excessive.

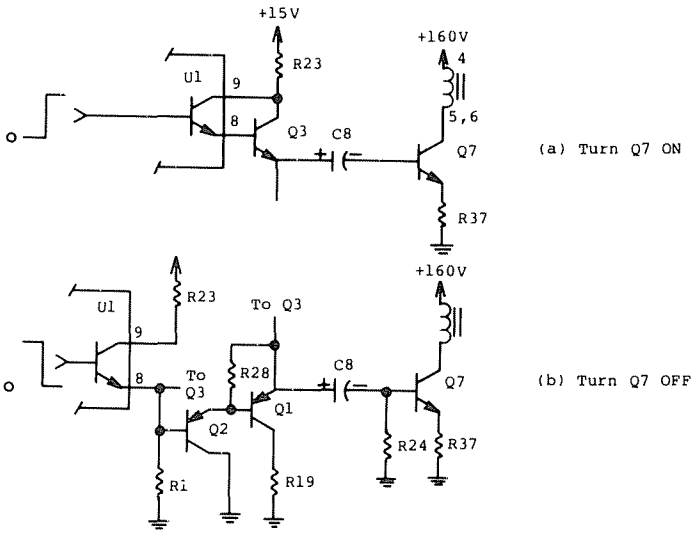


Figure 6-11. Base Drive Circuit

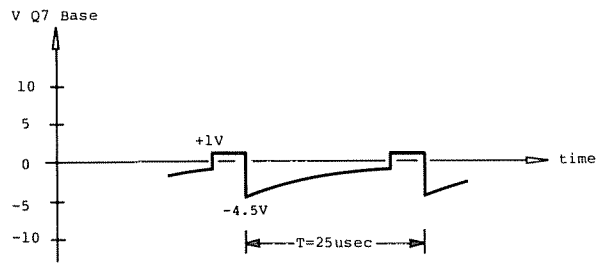


Figure 6-12. Q7 Base Voltage Waveform

As Q3 turns on, C8 charges to approximately +5 volts and Q7 is driven into saturation. Energy is stored in the primary winding of T1 as the collector current of Q7 increases or "ramps up" at a rate determined by the inductance of the transformer primary winding.

When the output transistor of U1 turns off, the emitters of Q1 and Q2 are initially at the +6 volt level determined by the charge on C8, the Vbe drop of Q7, and the drop across R37. Both base-emitter junctions of the Q1-Q2 Darlington pair are biased ON and the positive terminal of C8 is clamped to near-ground by the saturating Q1. At this point, C8 still has most of its charge and the base voltage of Q7 is approximately –4.5 volts with respect to ground.

With the strong reverse polarity provided by C8 across the base emitter junction of Q7, the "forward" charge stored in the junction capacitance is quickly swept out and Q7 is turned off. C8 continues to discharge through R24 to prepare for the next "ON" cycle. R19 limits the initial discharge of C8 while Q7 is turning off.

Notice the symmetry in the base drive circuit and the key role played by C8 in both the turn-on and turn-off sequences. Because of this crucial role in the circuit, this capacitor is specified as a high temperature, low-equivalent-series-resistance component.

Primary Circuit and Current Limit Shutdown

The Primary Circuit

The Primary circuit, shown in Figure 6-13a functions exactly as described earlier in the "Basic Principle" section. That is, the switch (Q7) is controlled by the base drive waveform developed by the control section.

The Snubber Circuit

Practical transformers cannot couple 100% of the stored energy from the primary to the secondary since all of the flux from the primary fails to link all the secondary turns. A circuit using this practical transformer behaves as though a small fraction of the primary inductance was not wound on the core of the transformer, but instead placed apart from the primary and in series with it. This small, separately-acting inductance does not participate in the transformer action and is called the leakage inductance.

If the resonant circuit, consisting of this leakage inductance and the stray capacitance in the adjacent circuit, has sufficient Q (relatively low resistance losses), a damped oscillation will occur in this resonant circuit when the transistor switch opens. The peak value of this oscillation will add to the Vce = 2 x Vin which appears across the transistor switch just after turn-off.

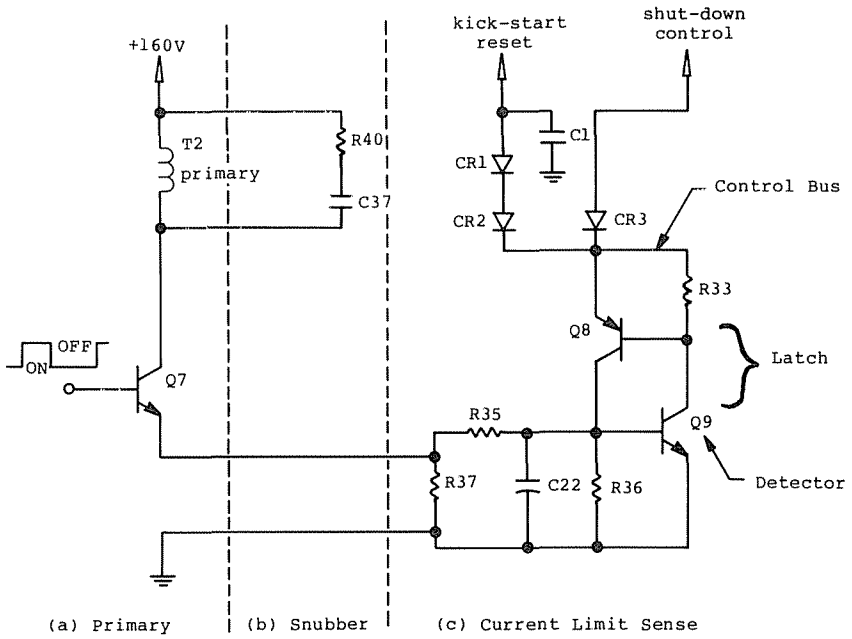


Figure 6-13. Primary Side Protection

The combined peak Vce may exceed the transistor breakdown rating if not damped out by the action of a snubber circuit.

When Q7 turns off, the energy stored in the leakage inductance is transferred to the electric field of the total capacitance of C37 plus stray capacitance. (Since C37 capacitance is much larger than the strays, it dominates in this action and tends to limit the peak value of the Q7 turn-off voltage.) If there were no resistance in this series connection of C37-plus-parasitics and leakage inductance, they would exchange this energy back and forth indefinitely. R40 is used to damp this oscillation without excessively slowing the turn-off of Q7, thus effectively snubbing the turn-off voltage spike at the collector of Q7.

Current Limit Circuit and the Shut-Down Sequence

The current limit circuit forces the voltage level at a control pin of U1 to change to a near-zero value very quickly when the current in the transistor switch exceeds a predetermined point. It also removes the supply voltage from the control circuit and resets the kick start latch and soft-start circuits.

The current limit circuit shown in Figure 6-13c has three parts: a control bus, a detector, and a latch. The control bus supplies the operating DC voltage to the current limit circuit. It also conducts the current limit signal to control pin 13 and to the reset-point in the kick start latch circuit. Diodes CR2 and CR3 steer this signal.

The normal maximum peak current in switching transistor Q7 is 3 amps. The detector transistor Q8 is biased to turn on by the divider action of R35 and R36 whenever the Q7 peak current through R37 exceeds 4 amps. A low-pass filter, formed by R35 and C22, prevents false detections on transient signals that don't represent an over-current condition.

As soon as Q9 turns on, its collector current develops the turn-on bias for Q8 across R33, and the Q8-Q9 pair "latches" in the "ON" state until the DC source for the latch is removed. Removal of this DC source occurs when C1 discharges through CR1, thus removing DC voltage from the control IC. Notice also that the kick start latch, Q4 and Q5, is still in the "ON" state and thus provides a discharge path for C14. When the decreasing voltage across C14 is less than approximately one volt, the Q4-Q5 latch also switches off.

At this point in time, all circuits are in an OFF condition except the input DC supply. C14 now begins to re-charge toward the input DC supply to restart the power supply. If a fault remains, the kick start and current limit circuits will continue to shut-down and re-start the power supply several times per second until the fault is removed or AC power to the supply is turned-off.

Under-Voltage Lockout

The Under-Voltage Lockout, UVL, shuts down the supply whenever the AC input voltage drops below about 90 volts. This occurs when the voltage at pin 13, set by the divider action of R27 and R25, diminishes to a level below the internal reference supply of the control IC. Pulses are inhibited immediately and because the DC supply to the Control IC is no longer replenished by the auxiliary supply, it discharges toward zero.

Why is it important to shut down the supply if the input AC line drops below 90 volts? The answer will become clear when an inherent characteristic of the circuit is discussed, namely, its negative input resistance.

Imagine the situation where the supply is delivering full power to its load and the AC input voltage drops five or ten volts. The supply control circuit responds by increasing the "ON" time of the switching transistor thus increasing the average current in the primary winding. The only way the DC supply can deliver more current is to draw it from the AC line. So the negative change in AC input voltage was accompanied by a positive change in AC input current.

Another way to describe this characteristic is that the supply is a constant power device, that is:

$$P_{in} = V_{in} \times I_{in} = \text{constant.}$$

Thus if V decreases, I will increase, and vice versa. The supply will thus draw more and more current from the AC line if the AC voltage continues to decrease. In order to limit the average current to a safe value, the control circuit senses the input voltage and shuts down the supply before the AC voltage level becomes too low or the AC current input becomes too high.

Secondary Outputs

Each of the secondary windings consist of a half-wave rectifier followed by a pi filter. The input capacitor of the filter stores the charge delivered to it when the rectifier is biased ON by the polarity of the transformer winding. The inductor and the output capacitor form a low-pass filter which removes the switching frequency ripple component.

The current output of the −12 volt supply is much smaller than that of the positive voltage outputs. Because of this, the current limit circuit response is not sufficiently effective to prevent damage to the −12 volt circuit. Therefore, a three terminal regulator with its own current limiting circuit is used to protect the −12 volt output.

All of the 12 volt rectifiers are fast recovery types and the +5 volt rectifier is a Schottky type. These diodes feature high switching speeds during turn-off. Their low forward voltage drop minimizes dissipation resulting in maximum efficiency. Each of the positive outputs has a bleeder resistor.

The reason for two separate +12 volt outputs is to provide sufficient isolation between different types of loads. It is easier to regulate the +12 volts if the load which contains the DC motors in the disk drives is separated from the rest of the loads. In addition, the +12 volt "Disk" output (V3) is included in the load sense network in order to minimize the load transients which occur when the disk drives turn on and off. The supply is then better able to regulate the other +12 volt output (V2) during the severe V3 transitions.

Load Sense and Feedback Signal Development

The circuit of Figure 6-14 has three parts. In part (a), the IC's U2 and U3 are biased ON by resistors R11 and R22. These resistors also sense the changes in AC line input voltage to provide line regulation. U2A is the LED half of an optocoupler which serves to isolate the DC ground circuits of primary and secondary while coupling the AC feedback signal via optical coupling. U3 serves as both a stable DC reference voltage which the output voltages are compared against and as an error amplifier which provides the gain necessary for adequate sensitivity of the control IC to load changes.

Each of the passive components in the load sensing network is a high stability (+/– 100ppm) part to assure stability of the network over the operating temperature range of the power supply.

Part (b) of Figure 6-14 includes the network which tailors the frequency response of the error amplifier so that it responds to low frequency change only. This network, consisting of R14/C5 and R13/C4, also determines the stability of the power supply by ensuring that the power supply control circuit has no tendency to oscillate.

Part (c) illustrates the load sensing network. Equal currents through R15 are supplied from the +12V DISK and +5.05V outputs by R29 and R30. In addition, a portion of the transient signal occurring on the +12V CRT output (when the motors turn on or off) is fed to R15 by C17. The wiper of R15 feeds a control signal which represents the status of the current loads to the error amplifier U3. U3 amplifies and compensates it then U2 couples that control signal to U1 where it is used to vary the switching transistor (Q7) ON time to adjust the output voltages as necessary. R15 is adjustable to provide the initial set-up of the +5.05V output when it is installed in a computer.

Overvoltage Crowbar

Some of the circuits supplied by the +5 volt output are quite sensitive to voltages in excess of 7 volts. Since some circuits require both +5 and +12 volts, a failure in those circuits could apply +12 volts to the +5 volt bus and thus damage some of the +5 volt circuits. To prevent the +5 volt bus from exceeding a safe level, an SCR, Q6, is used to "crowbar" or short-circuit the +5.05-volt output to the secondary ground bus. This short circuit triggers the current limiting circuit and the supply shuts down until it tries to restart.

Referring to Figure 6-15, VR2 sets the turn-on point of the SCR and R17 develops the gate signal when VR2's Zener breakdown voltage of 5.6 volts is exceeded. C6 and R17 provide current limiting for VR2 and filter the gate signal so Q6 won't respond to transient signals.

Power Chain

In a sense we have already analyzed the power chain in the section on basic principle of operation. The base drive causes the switching transistor to turn on and off at a prescribed rate. This action alternately stores energy from the DC input in the primary inductance and releases it into the secondary through the flyback transformer action. The energy is then stored in the input filter capacitor at a voltage determined by the

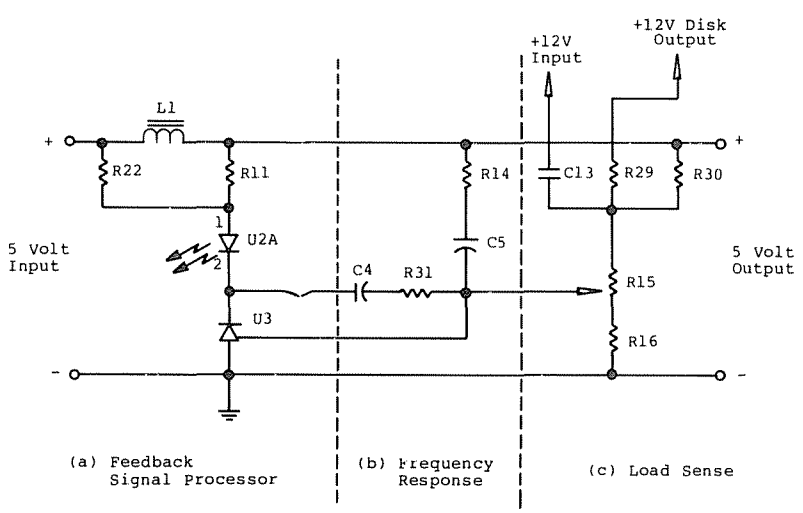


Figure 6-14. Feedback Signal Development

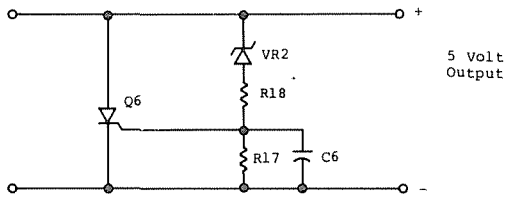


Figure 6-15 Overvoltage Crowbar

transformer turns ratio. Notice that the turns ratio determines the ratio of collector voltage to secondary voltage, both of which are alternating voltages. The ratio of input-to-output DC voltage is determined by the duty cycle and the turns ratio together.

For example, lets look at the +5 volt output of Figure 6-16 at normal loading and approximately 120 VAC input. Under these conditions, the DC input voltage is 168 VDC and the duty cycle is approximately 40%. Thus, our average DC voltage at the switching transistor collector (or across the primary) is 40% of 168 or 67.5 volts. Dividing this average DC voltage by the turns ratio for the 5 volt secondary (54 : 4 = 13.5) gives us 5.0 volts.

$$\frac{67.5V}{13.5} = 5V$$

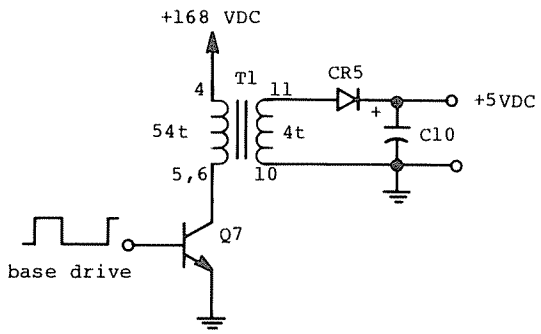


Figure 6-16. Power Chain

Control Chain

Imagine the load end of the feedback path disconnected from the +5.1 volt output terminal and unfolded so that the load sense network is now at the "input". The secondary rectifier (CR5) and filter (C10/C11, L1, C12) remain as the output. The circuit as it now appears, redrawn in simplified form in Figure 6-17 is known as the control chain. To see how the regulation action occurs, assume a small negative voltage change at the "input" of the feedback network and follow it through the control chain.

This negative voltage change, which would correspond to a slightly heavier load current, appears at pin 1 of U3 as a decreasing voltage. The error amplifier in U3 inverts and amplifies this signal. The positive-going output voltage of U3 at pin 3 causes less current to flow in the internal LED of U2A. A replica of this smaller current, optically coupled and induced in the phototransistor of U2B, develops a reduced voltage across R7 at the non-inverting input of the regulator error amplifier in U1.

The regulator error amplifier in U1 does not invert the signal, but further amplifies it, improving the sensitivity of the control chain to small changes at the power supply output. The regulator error amplifier output is Vreg. Since we established earlier that a negative-going Vreg increases the length of the base drive pulse, Q7 is turned on a little sooner so that it can store more energy from the AC line in the primary inductance. Finally, this increased energy is stored in the filter capacitor C10/C11 during the flyback interval and supplies the increased demand for current that resulted in the original reduction in the output voltage.

More simply stated, the control chain uses an amplified version of the output voltage CHANGE to adjust the width of the base drive pulse through the action of a control voltage at a comparator input.

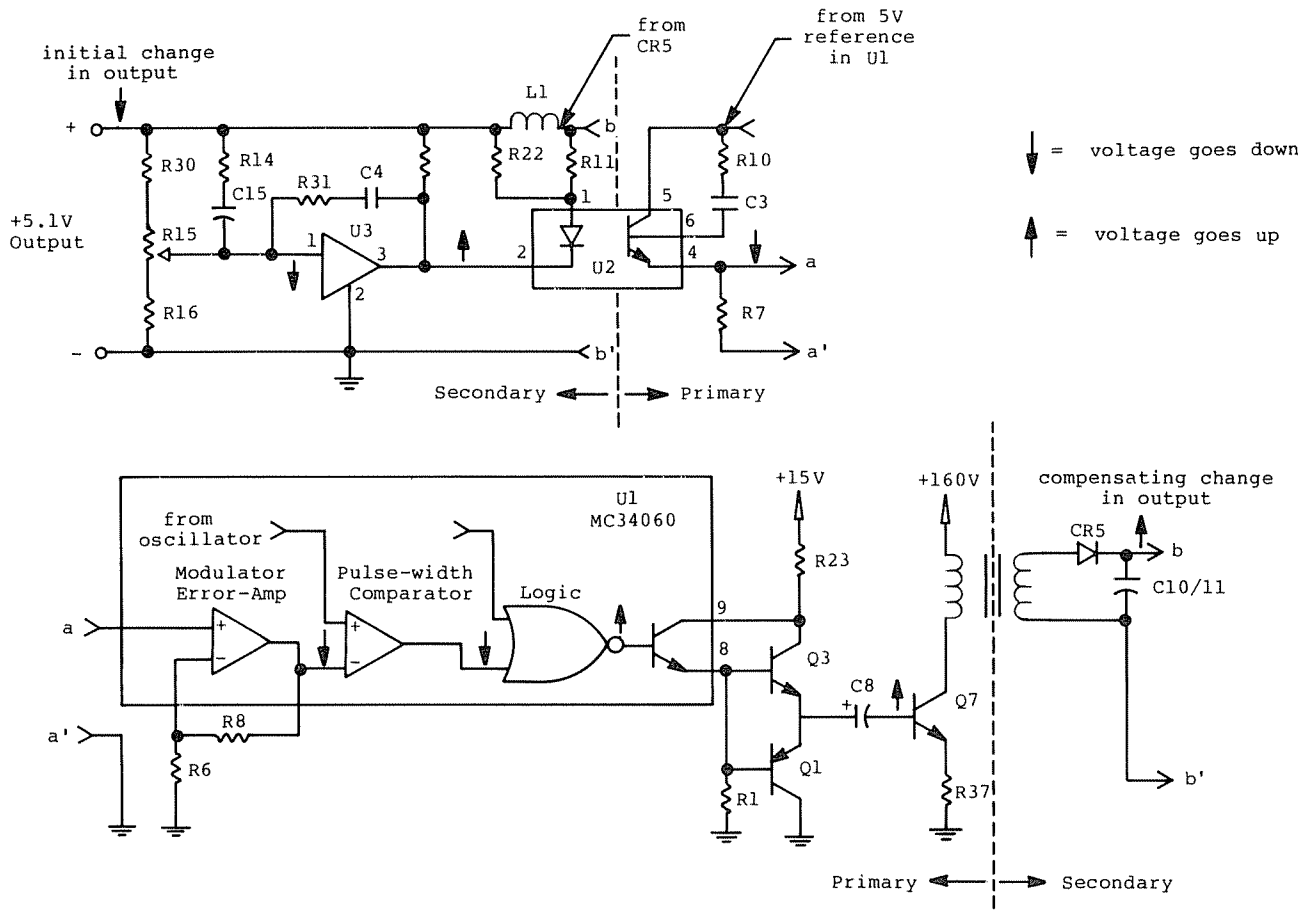


Figure 6-17. Control Chain Simplified Schematic

6.3.4 Troubleshooting Chart for Power Supply #8790049, 65 Watt

Trouble	Cause	Remedy
open fuse	shorted line input filter capacitor	check and/or replace C33, C32, C31, C30
	shorted bridge	check BR1
	shorted filter capacitor	check C29, C26, R39
	shorted switching transistor	check Q7, C37, R40, C26, T1 pri., Q3, Q1, R37
Current limit cycle	single rectifier open in bridge	check and/or replace BR1
	open filter capacitor	check C29
	shorted snubber capacitor or resistor	check C37, R40
	open opto-coupler	check U2
	shorted supply output	check computer for short on +5V, +12V CRT, +12V DISK, –12V outputs and clear shorted condition
	shorted output rectifier	check CR5, CR6, CR7, CR8
	open or shorted output filter capacitor	check C16, C18, C25, C23, C10, C11, C12, C19, C20
	defective crowbar	check Q6

no pulses at pin 8 of U1, (i.e., supply shut down)	no aux. DC supply	check and/or replace CR4, C14, T1 aux.
	no "kick start"	check R26, Q4, Q5, VR1, CR1, C1
	no base drive	check U1, Q3, R23, C8, R24
	dead-time control divider malfunction	check C2, R4, R5, U1 (for V ref.)
	under-voltage protect divider malfunction	check R27, R25, C9, Q9
	PWM feedback malfunction	check and/or replace U1, U2, C3

6.3.5 Testing and Adjustments

The following tests should be performed to guarantee correct operation of the power supply after repairs have been made. The first test checks the primary circuits and is to be made without AC power applied. The second test is a complete operational test with AC power applied.

Primary, Checks T2, U1

NO AC POWER APPLIED

1. Apply +35 volts DC via 3900 ohm resistor from Q5 emitter to primary side ground. Observe the voltage across C5 as it charges. As it reaches a value near +31 volts (about 2 seconds), it should drop to near +15 volts as Q5 turns on.

2. Check U1 pin 8 and/or Q8 base for a base drive pulse: a 40 kHz square wave of 8 volts/4 volts amplitude respectively.

3. Apply +35 volts DC through a 120K ohm resistor and a normally closed SPST switch to U1 pin 13. Operate the switch and observe Q8 base (TP1) for loss of base drive pulses.

Operational, Checks T2, U1, U2

APPLY AC POWER

1. Apply rated maximum loading for condition 1 (Model III use) or condition 2 (5 1/4" Hard Disk use).

2. Apply 120 VAC input voltage and observe Q7 current (via loop on PCB) and voltage (at TP2). Supply should start in two to four seconds.

3. Observe the +5.05 volt output and adjust R15 until the output is exactly +5.05 volts DC.

4. Measure +12V and –12V outputs.

5. Check all outputs at Vin = 90 VAC and 135 VAC at:
   (a) minimum and maximum loads
   (b) check +12V CRT when +12V DISK varies in transient test.

6. Measure ripple. See Measurement Techniques below.

7. Measure efficiency. See Measurement Techniques below.

8. Test operation of current limit and over-voltage protection circuits by applying +7.0 volts to the +5 volt output.

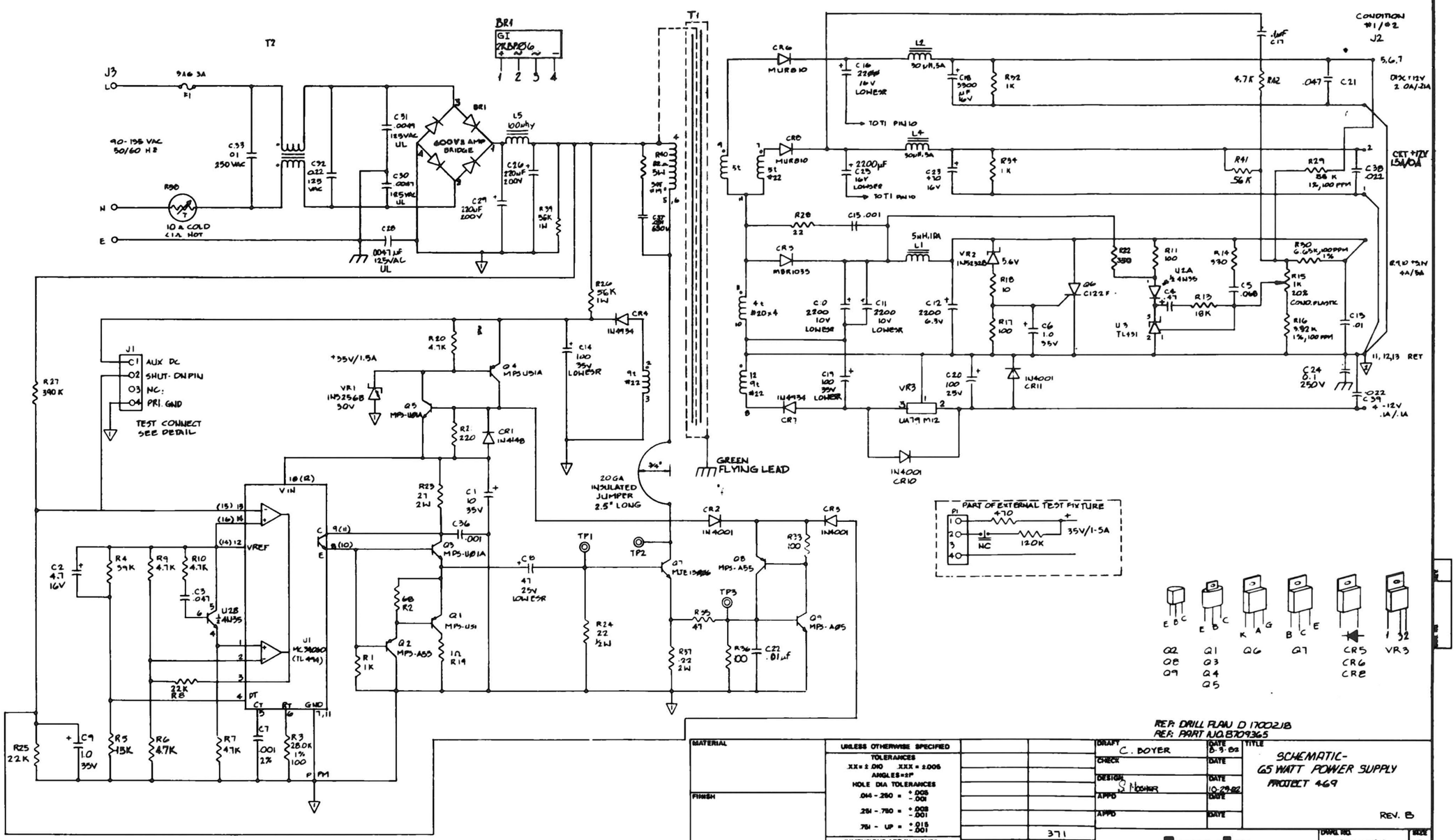
Measurement Techniques

1. Ripple – Unit connected to full load at low line. One end of 50 ohm coaxial cable connected to output terminals. Other end of cable (terminated with 0.01uF ceramic cap in series with 51 ohm resistor) connected to scope using BNC T-fitting. Two components at 120 Hz and 40 kHz.

2. Efficiency – Use Diego Systems Series 200 power monitor. Efficiency = $\frac{\text{Power Out}}{\text{Power In}}$

LTR		REVISION DESCRIPTION	DATE	APPROVED
(D)	A+	REMOVED C27 & CR9 / MOVED C37/ Q R40 82 Ω WAS 6.8K / ADDED TRACE FROM R 29 TO PIN 3,6,7	12-27-82	12-28-82
B		R 29 WAS 28 K1 POLARIZED C 91 ADDED R41, R42	5-18-83	

ZONE	LTR		REVISION DESCRIPTION	DATE	APPROVED
	(A)	A1	R22 390Ω WAS 1000 / DELETED R12	10-27-82	
	(B)	A2	C17, .1uF WAS .01 / ADDED TO C28 100/50V	10-29-82	
	(C)	A3	MODIFIED EXTERNAL TEST FIXTURE, ADDED C37 (R38, 10Ω COLD WAS 16Ω COLD TCO #1338	11-23-82	



REF: DRILL PLAN D 170021B
REF: PART NO. 8709365

MATERIAL	UNLESS OTHERWISE SPECIFIED TOLERANCES .XX = ± .010 .XXX = ± .005 ANGLES = ±1° HOLE DIA TOLERANCES .014 - .250 = +.005 -.001; .251 - .750 = +.008 -.001; .751 - UP = +.015 -.001		
FINISH	DIMENSIONS ARE IN INCHES AND APPLY AFTER PLATING	NEXT ASSY.	USED ON
	DO NOT SCALE THIS DRAWING	371	469

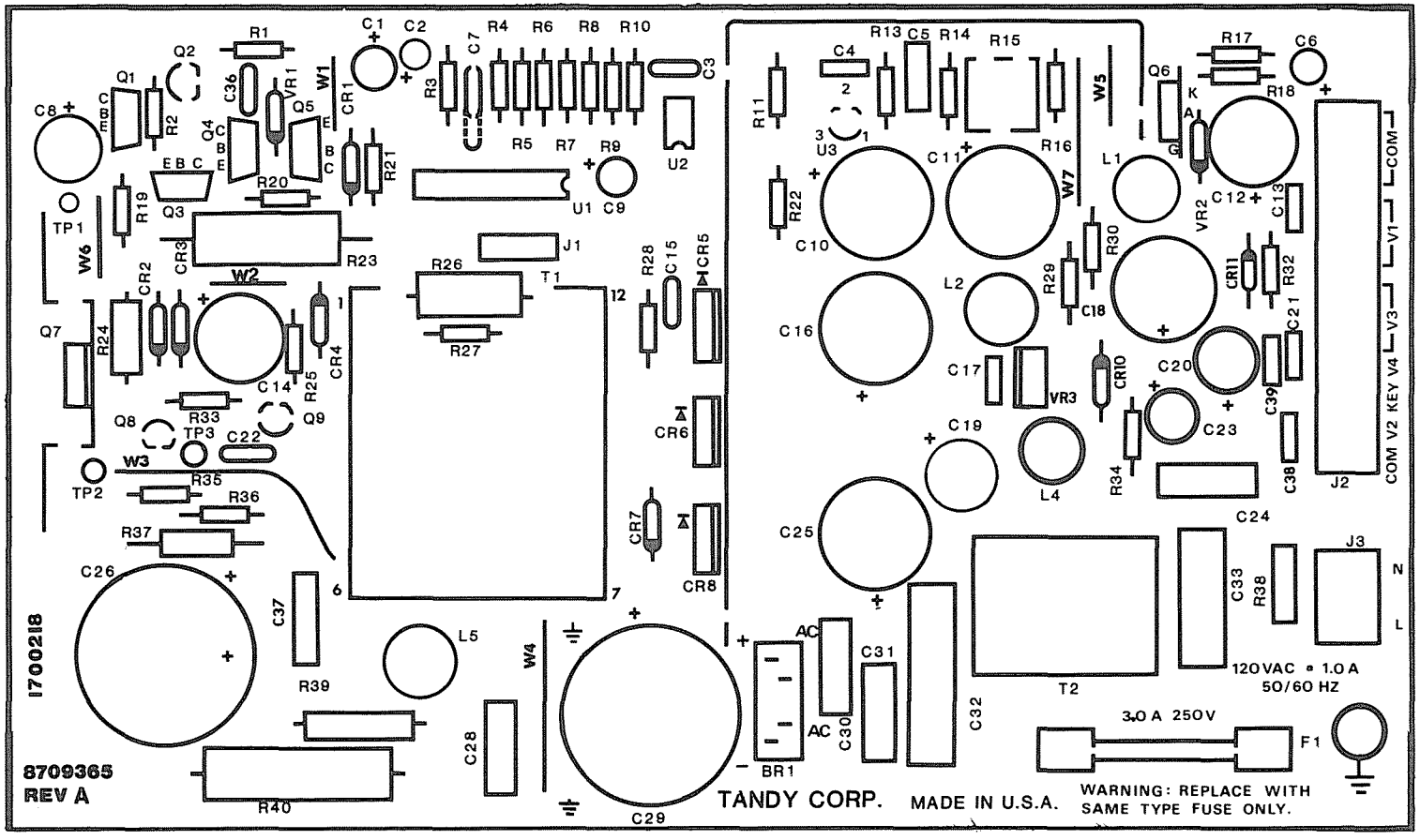
DRAFT: C. BOYER — DATE 8-9-82
CHECK — DATE
DESIGN: S. MOSHER — DATE 10-29-82
APPD — DATE
APPD — DATE

TITLE: SCHEMATIC- G5 WATT POWER SUPPLY PROJECT 469

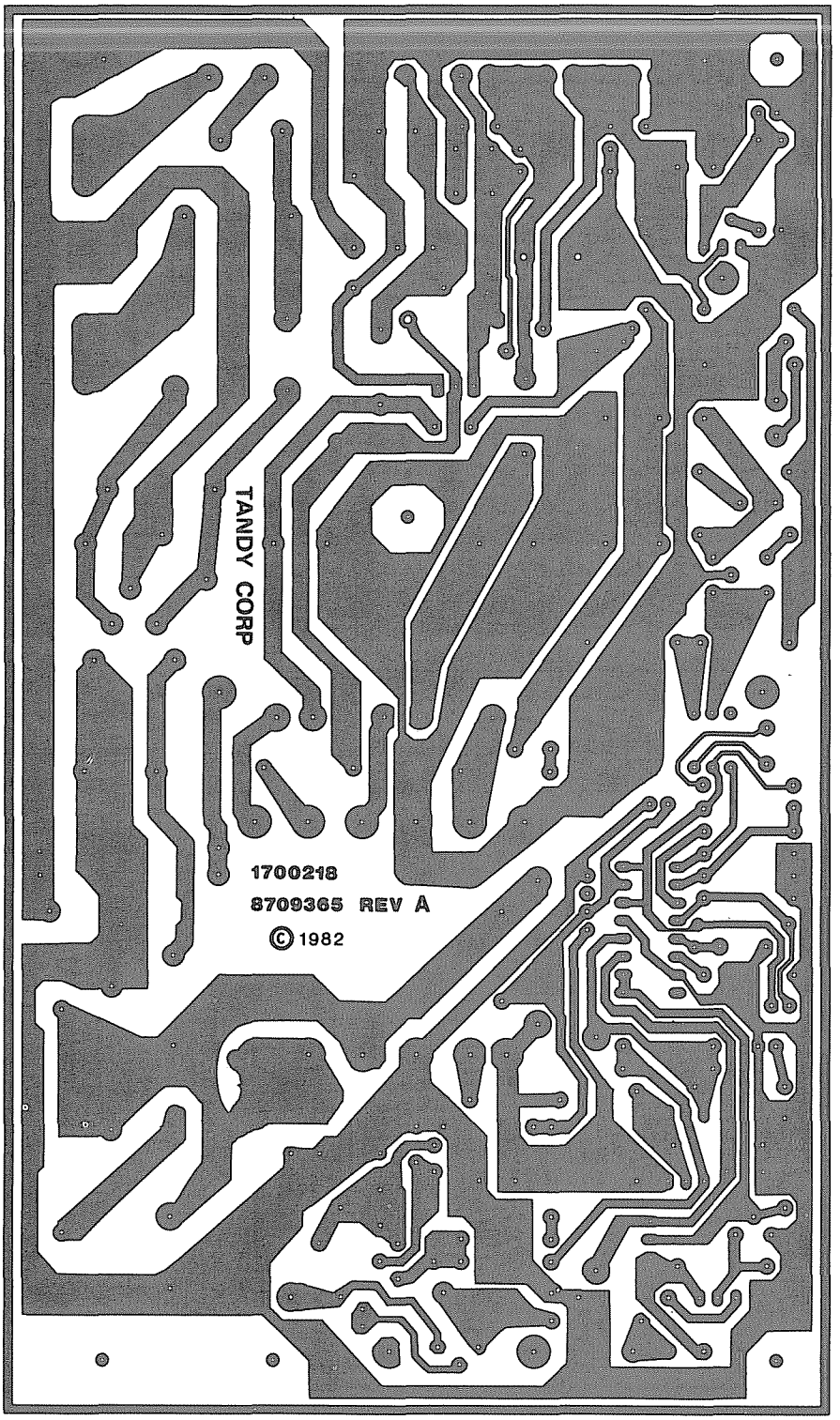
REV. B

tandy

DWG NO. 8000164 — SIZE D — SCALE NONE — SHEET 1 OF 1



COMPONENT LOCATION, POWER SUPPLY PCB #8790049 65W (TANDY)



CIRCUIT TRACE, POWER SUPPLY PCB #8790049, SOLDER SIDE 65W (TANDY)

Parts List, Power Supply #8790049 (Tandy 65W)

Item	Sym	Description	Mfgr's Part No.
1	C1	Capacitor, 10 ufd, 35V Elect Radial	8326103
2	C2	Capacitor, 4.7 ufd, 25V Elect Radial	8325470
3	C3	Capacitor, 0.047 ufd, 50/63V	8393474
4	C4	Capacitor, 0.47 ufd, 50/63V	8394474
5	C5	Capacitor, 0.068 ufd, 50/63V	8393684
6	C6	Capacitor, 1 ufd, 50V, Elect Radial	8325014
7	C7	Capacitor, 0.001 ufd, 63V	8392104
8	C8	Capacitor, 47 ufd, 25V, Elect Radial	8326472
9	C9	Capacitor, 1 ufd, 50V, Elect Radial	8325014
10	C10	Capacitor, 2200 ufd, 10V, Elect Radial	
11	C11	Capacitor, 2200 ufd, 10V, Elect Radial	
12	C12	Capacitor, 2200 ufd, 6.3V, Elect Radial	8328220
13	C13	Capacitor, 0.01 ufd, 50/63V	8393104
14	C14	Capacitor, 100 ufd, 35V, Elect Radial	8327103
15	C15	Capacitor, 1000 pfd, 100V, Ceramic Disk	8302106
16	C16	Capacitor, 2200 ufd, 16V, Elect Radial	8328221
17	C17	Capacitor, 0.1 ufd, 50/63V	8304104
18	C18	Capacitor, 3300 ufd, 16V, Elect Radial	8328331
19	C19	Capacitor, 100 ufd, 25V, Elect Radial	8327102
20	C20	Capacitor, 100 ufd, 25V, Elect Radial	8327102
21	C21	Capacitor, 0.01 ufd, 50/63V	8393104
22	C22	Capacitor, 0.01 ufd, 50/63V	8393104
23	C23	Capacitor, 470 ufd, 16V, Elect Radial	8327461
24	C24	Capacitor, 0.1 ufd, 250V	
25	C25	Capacitor, 2200 ufd, 16V, Elect Radial	8328221
26	C26	Capacitor, 220 ufd, 200V, Elect Radial	8327226
27	C27	Not Used	
28	C28	Capacitor, 0.01 ufd, 250 VAC	8393106
29	C29	Capacitor, 220 ufd, 200V, Elect Radial	8327226
30	C30	Capacitor, 4700 pfd, 125VAC, Ceramic Disk	8303475
31	C31	Capacitor, 4700 pfd, 125VAC, Ceramic Disk	8303475
32	C32	Capacitor, 0.1 ufd, 250VAC	8394106
33	CR1	Diode, 1N4148, Switching	8150148
34	CR2	Diode, 1N4001, 1A/50PIV	8150001
35	CR3	Diode, 1N4001, 1A/50PIC	8150001
36	CR4	Diode, 1N4939, 1A/100PIC	8150934
37	CR5	Diode, MBR1035, 8/10A, 35V, TO-220	8150035
38	CR6	Diode, MUR810, 8A/100PIV, TO-220	8150810
39	CR7	Diode, 1N4934, 1A/100PIV	8150934
40	CR8	Diode, MUR810, 8A/100PIV, TO-220	8150810
41	BR1	Dioide Bridge, 2A,600PIV	8160402
42	VR1	Zener Diode, 1N5232B, 5.6V	8150232
43	VR2	Zener Diode, 1N5256B, 30V	8150256
44	VR3	Diode, A79M12	

Parts List, Power Supply #8790049 (Tandy 65W)

Item	Sym	Description	Mfgr's Part No.
45	F1	Fuse, 3A, AGC	8479104
46	L1	Inductor, 5.0h,m 10A	8419006
47	L2	Inductor, 30h, 5A	8419008
48	L3	Not Used	
49	L4	Inductor, 30h, 5A	8419008
50	L5	Inductor, 100h, 2A	8419009
51	U1	IC, MC34060, Switching Regulator	8060060
52		uA/TL494 Switching Regulator	8060494
53	U2	IC, 4N35, Opto-Isolator	8170428
54	R1	Resistor, 1 kohm, 1/4W 5%	8207210
55	R2	Resistor, 68 ohm, 1/4W 5%	8207068
56	R3	Resistor, 28 kohm, 1/4W 1%	8200328
57	R4	Resistor, 39 kohm, 1/4W 5%	8207339
58	R5	Resistor, 15 kohm, 1/4W 5%	8207315
59	R6	Resistor, 4.7 kohm, 1/4W 5%	8207247
60	R7	Resistor, 4.7 kohm, 1/4W 5%	8207247
61	R8	Resistor, 22 kohm, 1/4W 5%	8207322
62	R9	Resistor, 4.7 kohm, 1/4W 5%	8207247
63	R10	Resistor, 4.7 kohm, 1/4W 5%	8207247
64	R11	Resistor, 100 ohm, 1/4W 5%	8207110
65	R12	Resistor, 620 ohm, 1/4W 5%	
66	R13	Resistor, 18 kohm, 1/4Q 5%	8207318
67	R14	Resistor, 330 ohm, 1/4W 5%	8207133
68	R15	Potentiometer, 1 kohm, 20%, Linear	8275211
69	R16	Resistor, 3.31 kohm, 1/4W 1%	8200232
70	R17	Resistor, 100 ohm, 1/4W 5%	8207110
71	R18	Resistor, 10 ohm, 1/4W 5%	8207010
72	R19	Resistor, 1 ohm, 1/4W 5%	8207001
73	R20	Resistor, 4.7 kohm, 1/4W 5%	8207247
74	R21	Resistor, 220 ohm, 1/4W 5%	8207122
75	R22	Resistor, 330 ohm, 1/4W 5%	8207133
76	R23	Resistor, 27 ohm, 2W 10%	8248127
77	R24	Resistor, 22 ohm, 1/2W 5%	8217022
78	R25	Resistor, 22 kohm, 1/4W 5%	8207322
79	R26	Resistor, 75 kohm, 1W 10%	8248127
80	R27	Resistor, 430 kohm, 1/4W 5%	8207443
81	R28	Resistor, 22 ohm, 1/4W 5%	8207022
82	R29	Resistor, 28 kohm, 1/4W 1%	8200328
83	R30	Resistor, 6.65 kohm, 1/4W 1%	8200266
84	R31	Resistor, 1 kohm, 1/4W 5%	8207210
85	R32	Resistor, 1 kohm, 1/4W 5%	8207210
86	R33	Resistor, 470 ohm, 1/4W 5%	8207470
87	R34	Resistor, 1 kohm, 1/4W 5%	8207210
88	R35	Resistor, 470 ohm, 1/4W 5%	8207470

Parts List, Power Supply #8790049 (Tandy 65W)

Item	Sym	Description	Mfgr's Part No.
89	R36	Resistor, 1 kohm, 1/4W 5%	8207210
90	R37	Resistor, 0.22 ohm, 2W 10%	8248022
91	R38	Thermistor	8298016
92	R39	Resistor, 75 kohm, 1W 10%	8248375
93	R40	Resistor, 82 kohm, 5W 10%	8248268
94	Q1	Transistor, MPSU51A, PNP, TO-202	8100051
95	Q2	Transistor, MPSA55, PNP, TO-92	8100055
96	Q3	Transistor, MPSU01A, NPN, TO-202	8111001
97	Q4	Transistor, MPSU51A,PNP, TO-202	8100051
98	Q5	Transistor, MPSU01A, NPN, TO-202	8111001
99	Q6	SCR, 8A/50PIV, TO-220	8140122
100	Q7	Transistor, MJE13006, NPN, 8A, 400V	8110006
101	Q8	Transistor, MPSA55, PNP, TO-92	8100055
102	Q9	Transistor, MPSA05, NPN, TO-92	8110005
103	T1	Transformer, Power, Ferrite Core	8790046
104	T2	Transformer, Line Choke, 5.5 mH/side, 2A	8790045

SECTION VII

VIDEO MONITOR

VIDEO MONITOR

7.1 VIDEO MONITOR #8492002 (RCA VERSION)

7.1.1 Functional Specifications

The video monitor is a 12-inch solid state monitor designed for display of alphanumeric dot characters. The monitor is designed for a 12-volt DC power input with an average power consumption of 12 watts. The monitor accepts separate video, and vertical drive TTL level inputs.

SPECIFICATIONS

Cathode Ray Tube:	12" diagonal, 90° deflection angle, 4 x 5 aspect ratio, P4 phosphor, integral implosion protection.
Environment:	Operating Temperature: 41°F to 131°F (5°C to 55°C) ambient Humidity: 95% non-condensing at 41°F to 104°F (5°C to 40°C) Operating Altitude: 10,000 ft. (3046 meters) maximum
Power Input:	+12 VDC at 1 amp nominal
TTL Level Input Signals:	4 volts ± 1.5 volts Horizontal: 4 to 25μsec, positive going Vertical: 100 to 1400μsec, negative going Video: positive white
Video Response:	Bandwidth: 15 MHz, 3 dB Pulse rise time less than 30nsec
Scanning Frequency:	Horizontal: 15,600 Hz ± 500 Hz Vertical: 50/60 Hz
Horizontal Retrace:	10.5μsec maximum
Vertical Retrace:	850μsec maximum

7.1.2 Service Adjustments

NOTE: Measurements should be made using 12.0 VDC input. Measurements with kine (CRT) attached will require the ground strap from kine be connected to chassis to prevent transistor failures in the event of kine arcing.

FOCUS

Adjust focus control F524 for best overall focus.

VERTICAL SIZE

Adjust vertical size control R617 to produce vertical scan of approximately 6 inches.

HORIZONTAL LINEARITY

Loosen deflection yoke clamp and slide linearity sleeve forward or backward to equalize character spacing on left side to match character spacing on right side of screen (See Figure 7-1 for location of linearity sleeve.)

WIDTH

Note: Check horizontal linearity prior to width adjustment. Adjust width control to produce horizontal scan of approximately 8 inches.

CENTERING

Adjust centering rings on deflection yoke assembly to center display on screen top to bottom and left to right.

HORIZONTAL HOLD

Horizontal Hold is accomplished by adjustment of the horizontal oscillator coil.

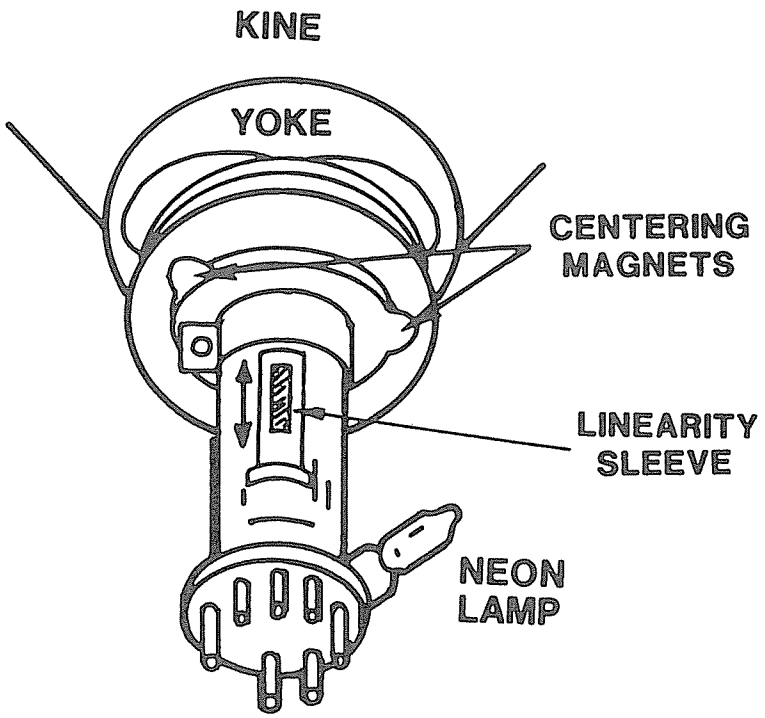
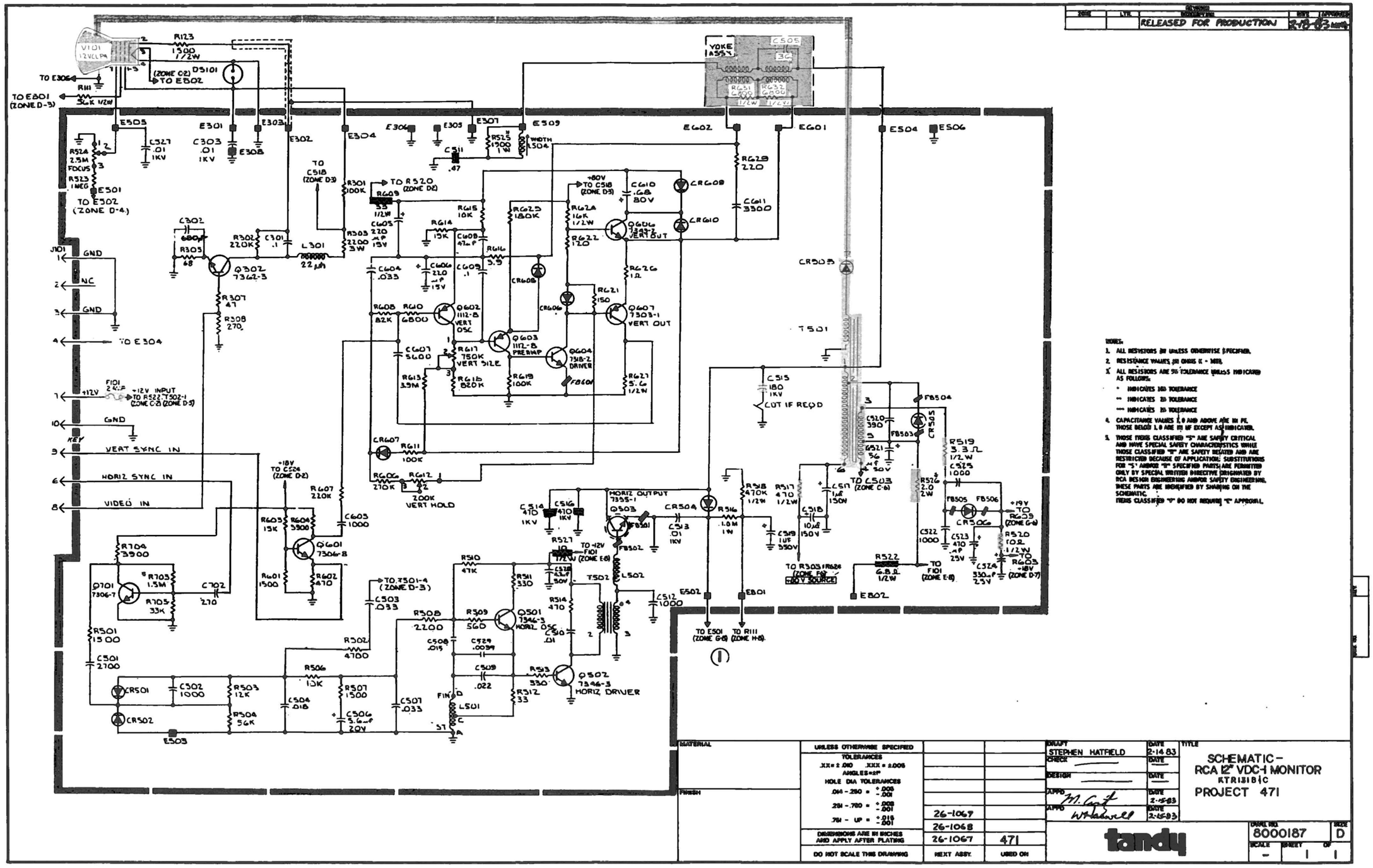


FIGURE 7-1. DEFLECTION YOKE ASSEMBLY



NOTES:

1. ALL RESISTORS 1W UNLESS OTHERWISE SPECIFIED.
2. RESISTANCE VALUES 1W OHMS K = 1000.
3. ALL RESISTORS ARE 5% TOLERANCE UNLESS INDICATED AS FOLLOWS:
   - * INDICATES 10% TOLERANCE
   - ** INDICATES 2% TOLERANCE
   - *** INDICATES 1% TOLERANCE
4. CAPACITANCE VALUES 1.0 AND ABOVE ARE IN PF. THOSE BELOW 1.0 ARE IN UF EXCEPT AS INDICATED.
5. THOSE ITEMS CLASSIFIED "S" ARE SAFETY CRITICAL AND HAVE SPECIAL SAFETY CHARACTERISTICS WHILE THOSE CLASSIFIED "D" ARE SAFETY DERATED AND ARE RESTRICTED BECAUSE OF APPLICATION. SUBSTITUTIONS FOR "S" AND/OR "D" SPECIFIED PARTS ARE PERMITTED ONLY BY SPECIAL WRITTEN DIRECTIVE ORIGINATED BY RCA DESIGN ENGINEERING AND/OR SAFETY ENGINEERING. THESE PARTS ARE IDENTIFIED BY SHADING ON THE SCHEMATIC.
   ITEMS CLASSIFIED "D" DO NOT REQUIRE "E" APPROVAL.

MATERIAL	
FINISH	
UNLESS OTHERWISE SPECIFIED TOLERANCES .XX = ± .010 .XXX = ± .005 ANGLES = ±1° HOLE DIA TOLERANCES .014 - .250 = +.005 -.001; .251 - .750 = +.008 -.001; .751 - UP = +.016 -.001	
DIMENSIONS ARE IN INCHES AND APPLY AFTER PLATING	
DO NOT SCALE THIS DRAWING	

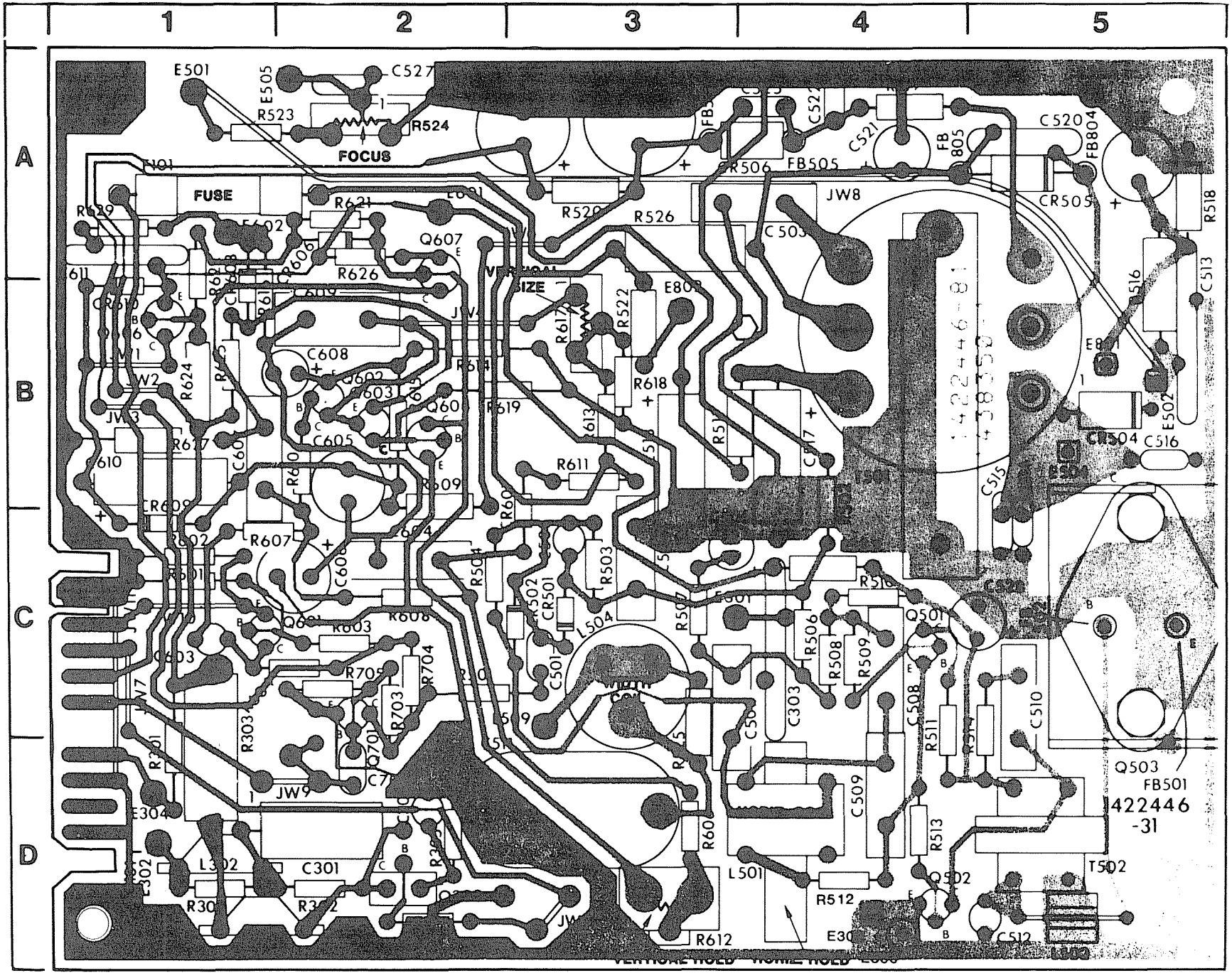
NEXT ASSY.	USED ON
26-1069	
26-1068	
26-1067	471

DRAFT: STEPHEN HATFIELD — DATE: 2-14-83
CHECK — DATE
DESIGN — DATE
APPD: M. Cook — DATE: 2-15-83
APPD: W. Hadwell — DATE: 2-15-83

TITLE: SCHEMATIC – RCA 12" VDC-1 MONITOR KTR1318C PROJECT 471

tandy

DWG. NO. 8000187 — SIZE D — SCALE – — SHEET 1 OF 1



COMPONENT LAYOUT/CIRCUIT TRACE, VIDEO MONITOR PCB #8492002 (RCA)

Video Monitor Assembly #8492002 (RCA #KTR131B & C)

Item	Qty	Description	Mfgr's Part No.
1	1	Board, Mounting	
2	1	Kine Socket Assembly	
3	7	Bead, Ferrite (FB501-506,601)	
4	1	Bead Choke Assembly (L502)	
5	1	Coil, (Peaking) 39uH (L301)	
6	1	Coil, (Peaking) 22uH (L302)	
7	1	Coil, Horiz. Hold (L501)	
8	1	Coil, (L504)	
9	1	Cap, .1 ufd 100V 10% Polyester (C301)	
10	1	Cap, 680 pfd 50V 10% Cer (C302)	
11	3	Cap, .01 ufd 1000V 20% Cer (C303,513,527)	
12	1	Cap, 2700 pfd 50V 10% Cer (C501)	
13	3	Cap, 1000 pfd 50V 20% Cer (C502,512,603)	
14	3	Cap, .033 ufd 100V 10% Polyester (C503,507,604)	
15	1	Cap, .018 ufd 200V 10% Polyester (C504)	
16	1	Cap, 5.6 ufd 20V +100-10% Elec (C506)	
17	1	Cap, .015 ufd 100V 10% Polyester (C508)	
18	1	Cap, .022 ufd 200V 10% Polyester (C509)	
19	1	Cap, .01 ufd 200V 10% Polyester (C510)	
20	1	Cap, .47 ufd 200V 10% Polyester (C511)	
21	2	Cap, 470 pfd 1000V 10% Cer (C514,516)	
22	1	Cap, 180 pfd 10% 1000V Cer (C515)	
23	1	Cap, 1 ufd 150V Elec (C517)	
24	1	Cap, 10 ufd 150V Elec (C518)	
25	1	Cap, 1 ufd 350V Elec (C519)	
26	1	Cap, 390 pfd 500V 10% Cer (C520)	
27	1	Cap, 56 ufd 50V +100-10% 50V Elec (C521)	
28	1	Cap, 1000 pfd 500V 10% Cer (C522,525)	
29	1	Cap, 470 ufd 25V +100-10% Elec (C523)	
30	1	Cap, 330 ufd 25V +100-10% Elec (C524)	
31	1	Cap, 4.7 ufd 50V +100-10% Elec (C528)	
32	1	Cap, .0039 ufd 5600 250V Cer (C529)	
33	2	Cap, 220 ufd 15V +100-10% Elec (C605,606)	
34	1	Cap, 5600 pfd 200V 10% Polyester (C607)	
35	1	Cap, 47 ufd 6V +100-10% Elec (C608)	
36	1	Cap, .1 ufd 100V 10% Polyester (C609)	
37	1	Cap, .68 ufd 80V 20% Elec (C610)	
38	1	Cap, 3300 pfd 500V 10% Cer (C611)	
39	1	Cap, 270 pfd 50V 10% Cer (C702)	

Video Monitor Assembly #8492002 (RCA #KTR131B & C)

Item	Qty	Description	Mfgr's Part No.
40	2	Diode (CR501,502)	
41	3	Diode (CR504-506)	
42	4	Diode (CR607-610)	
43	2	Res, 100K ohm 1/4W 5% (R301,619)	
44	1	Res, 220K ohm 1/4W 5% (R302)	
45	1	Res, 2.2K ohm 3W 5% Fixed Metal Film PRF (R303)	
46	1	Res, 68 ohm 1/4W 5% (R305)	
47	1	Res, 680 ohm 1/4W 5% (R306)	
48	1	Res, 47 ohm 1/4W 5% (R307)	
49	1	Res, 270 ohm 1/4W 5% (R308)	
50	1	Res, 1500 ohm 1/4W 5% (R501)	
51	1	Res, 4700 ohm 1/4W 5% (R502)	
52	1	Res, 12K ohm 1/4W 5% (R503)	
53	1	Res, 56K ohm 1/4W 5% (R504)	
54	2	Res, 10K ohm 1/4W 5% (R506,615)	
55	1	Res, 1500 ohm 1/4W 5% (R507)	
56	1	Res, 2200 ohm 1/4W 5% (R508)	
57	1	Res, 560 ohm 1/4W 5% (R509)	
58	1	Res, 47K ohm 1/4W 5% (R510)	
59	2	Res, 330 ohm 1/4W 5% (R511,513)	
60	1	Res, 33 ohm 1/4W 5% (R512)	
61	1	Res, 470 ohm 1/4W 5% (R514)	
62	1	Res, 1.0M ohm 1W 5% (R516)	
63	1	Res, 470 ohm 1/2W 5% (R517)	
64	1	Res, 470K ohm 1/2W 5% (R518)	
65	1	Res, 3.3 ohm 1/2W 5% Flameproof (R519)	
66	2	Res, 10 ohm 1/2W 5% Flameproof (R520,R527)	
67	1	Res, 6.8 ohm 1/2W 5% Flameproof (R522)	
68	1	Res, 1.0M ohm 1/4W 5% (R523)	
69	1	Res, 2.5M ohm Focus Cont. Variable (R524)	
70	1	Res, 1500 ohm 1W 5% (R525)	
71	1	Res, 2.0 ohm 2W 5% (R526)	
72	1	Res, 1.5K ohm 1/4W 5% (R601)	
73	1	Res, 470 ohm 1/4W 5% (R602)	
74	1	Res, 15K ohm 1/4W 5% (R603)	
75	2	Res, 3900 ohm 1/4W 5% (R604,704)	
76	1	Res, 270K ohm 1/4W 5% (R606)	
77	1	Res, 220K ohm 1/4W 5% (R607)	
78	1	Res, 82K ohm 1/4W 5% (R608)	
79	1	Res, 33 ohm 1/2W 5% Flameproof (R609)	
80	1	Res, 6800 ohm 1/4W 5% (R610)	

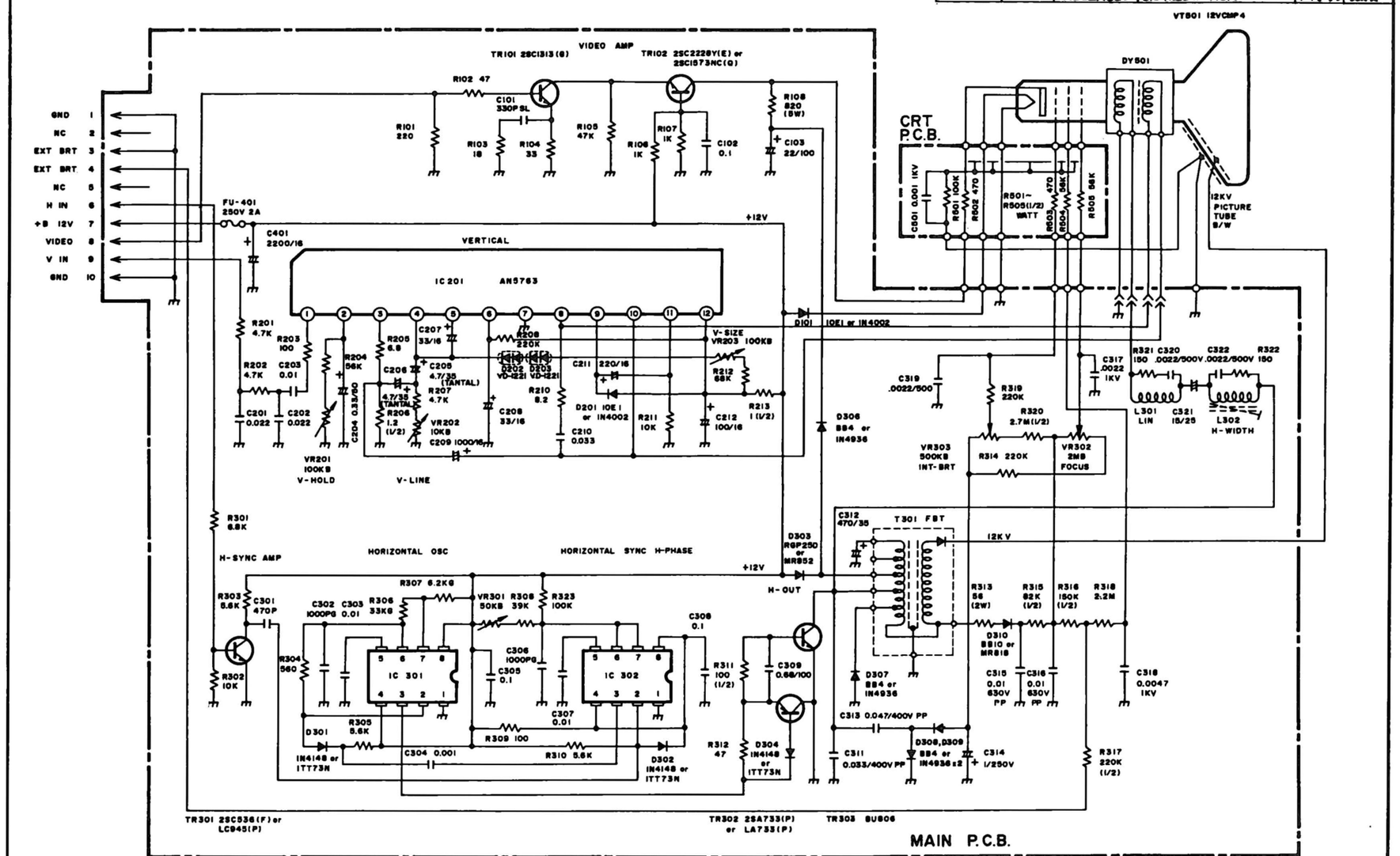
Video Monitor Assembly #8492002 (RCA #KTR131B & C)

Item	Qty	Description	Mfgr's Part No.
81	1	Res, 82K ohm 1/4W 5% (R611)	
82	1	Res, 200K ohm Vert. Hold Variable (R612)	
83	2	Res, 3.9M ohm 1/4W 5% (R613,616)	
84	1	Res, 15K ohm 1/4W 5% (R614)	
85	1	Res, 750K ohm Variable (R617)	
86	1	Res, 820K ohm 1/4W 5% (R618)	
87	1	Res, 150 ohm 1/4W 5% (R621)	
88	1	Res, 120 ohm 1/4W 5% (R622)	
89	1	Res, 16K ohm 1/2W 5% (R624)	
90	1	Res, 180K ohm 1/4W 5% (R625)	
91	1	Res, 1.0M ohm 1/2W 5% (R626)	
92	1	Res, 5.6 ohm 1/2W 5% (R627)	
93	1	Res, 220 ohm 1/4W 5% (R629)	
94	1	Res, 1.5M ohm 1/4W 10% (R703)	
95	1	Res, 33K ohm 1/4W 5% (R705)	
96	1	Transformer, Horiz drive (T502)	
97	1	Transformer Rectifier Assy (___)	
98	1	Transistor, 7362-3, Video Out (Q302)	
99	1	Transistor, 7346-3, Horiz Osc (Q501)	
100	1	Transistor, 7346-3, Horiz Driver (Q502)	
101	1	Transistor, 7335-1, Horiz Out (Q503)	
102	1	Transistor, 7306-8, Vert Int (Q601)	
103	1	Transistor, 1112-8, Vert Osc (Q602)	
104	1	Transistor, 1112-8, Pre Amp (Q603)	
105	1	Transistor, 7318-2, Vert Driver (Q604)	
106	1	Transistor, 7349-2, NPN Vert Out (Q606)	
107	1	Transistor, 7303-1, PNP Vert Out (Q607)	
108	1	Transistor, 7306-7, Horiz Sep (Q701)	
109	2	Buss Wire (.025 x 1.00) (JW1,7)	
110	3	Buss Wire (.025 x .800) (JW,2,3,5)	
111	1	Buss Wire (.025 x 1.200) (JW4)	
112	2	Buss Wire (.025 x .900) (JW6,9)	
113	1	Wire, Electrical Hook-up (RED) 7"LG (JW8)	
114	1	Buss Wire (.025 x 1.100) (JW10)	
115	1	Wire, Electrical Hook-up (BLK) 1.65LG (JW11)	
116	1	Wire, Electrical Hook-up (BRN) 5"LG (JW12)	
117	1	Buss Wire (.025 x 1.000) (JW13)	
118	1	Wire, Electrical Hook-up (BLK) 2.75LG (JW14)	
119	1	Wire, Electrical Hook-up (BLK) 3.00LG (JW15)	
120	1	Wire, Electrical Hook-up (BLK) 4.50LG (JW16)	
121	2	Wire, Electrical Hook-up (BLK) 1.87LG (___,__)	

Video Monitor Assembly #8492002 (RCA #KTR131B & C)

Item	Qty	Description	Mfgr's Part No.
		Miscellaneous	
122	1	Cable Assy (BRN)	
123	1	Cable Assy (GRN)	
124	1	Cable Assy (RED)	
125	1	Cable Assy (YEL)	
126	3	Eyelet - Rolled Flange	
127	1	Fuse, 2 Ampere (F101)	

REVISION				
ZONE	LTR.	DESCRIPTION	DATE	APPROVED
		RELEASED FOR PRODUCTION	1-18-83	WJH-44



NOTE: (1) ALL RESISTANCE VALUES ARE INDICATED IN "OHM" ($K=10^3$ OHM, $M=10^6$ OHM)
(2) ALL CAPACITANCE VALUES ARE INDICATED IN "μF" ($P=10^{-6}\mu F$)
(3) IC301, 302 NE555P or MC1455P1 or M51848P x2.

NEXT ASSY	USED ON
	471

DRAFT	W. HARWELL	DATE	1-15-83
CHECK		DATE	
DESIGN		DATE	
APPD		DATE	1-17-83
APPD		DATE	1-17-83

TITLE: SCHEMATIC- TCE 12" VIDEO MONITOR 8790607 PROJECT R-471

tandy

DWG NO 8000186 SIZE C
SCALE — SHEET 1 OF 1

Video Monitor Assembly #8790607 (TCE)

Item	Qty	Description	Mfgr's Part No.
		Deflection PCB unit w/o Trans Flyback, Heat Sink and Fuse	U-30001
1	1	Cap, 390 pfd 50V 10% Cer (C101)	CC45SL1H391KY
2	3	Cap, 0.1 ufd 50V 10% Mylar (C102,305, 308)	CQ92M1H104K
3	1	Cap, 22 ufd 100V 20% Elec (C103)	CE04C226M
4	2	Cap, 0.022 ufd 50V 10% Mylar (C201,202)	CQ92M1H223K
5	3	Cap, 0.01 ufd 50V 10% Mylar (C203,303 307)	CQ92M1H103K
6	1	Cap, 0.33 ufd 50V 10% Elec (C204)	CE04(RB)334K
7	2	Cap, 4.7 ufd 50V 20% Elec (C205,206) - used up to 3000 unit -	CE04(RB)475M
8	2	Cap, 4.7 ufd 35V 20% Tant (C205,206) - from 3001 unit-	CS15E475M
9	2	Cap, 33 ufd 16V 20% Elec (C207,208)	CE04C336M
10	1	Cap, 1000 ufd 16V 20% Elec (C209)	CE04C108M
11	1	Cap, 0.033 ufd 50V 10% Mylar (C210)	CQ92M1H333K
12	1	Cap, 220 ufd 16V 20% Elec (C211)	CE04C227M
13	1	Cap, 100 ufd 16V 20% Elec (C212)	CE04C107M
14	1	Cap, 470 pfd 50V 10% Cer (C301)	CC45SL1H471KY
15	2	Cap, 1000 pfd 50V 2% Polyst (C302,306)	CQ09S1H102G
16	1	Cap, 0.001 ufd 50V 10% Mylar (C304)	CQ92M1H102K
17	1	Cap, 0.68 ufd 100V 10% Met Plas (C309)	CF92N2A 684K
48	1	Cap, 0.033 ufd 400V 5% Poly (C311)	CQ92P2G 333J
19	1	Cap, 470 ufd 35V 20% Elec (C312)	CE04C477M
20	1	Cap, 0.047 ufd 400V 10% Poly (C313)	CQ92P2G 473K
21	1	Cap, 1 ufd 250V 20% Elec (C314)	CE04C105M
22	2	Cap, 0.01 ufd 630V 10% Poly (C315,316)	CQ92P2J 103K
23	1	Cap, 2200 pfd 1000V +100-0% Cer (C317)	CK45E3A222PY
24	1	Cap, 4700 pfd 1000V +100-0% Cer (C318)	CK45E3A472PY
25	3	Cap, 2200 pfd 500V +100-0% Cer (C319, 320,322)	CK45E2H222PY
26	1	Cap, 15 ufd 25V 20% Elec (C321)	CE04(RP)159M
27	1	Cap, 220 ufd 16V 20% Elec (C401)	CE04C228M
28	1	Coil, Linearity (L301)	143410020A
29	1	Coil, Width (L302)	143310140A
30	2	Diode, 1N4002 (D101,201)	101-E
31	2	Diode, VD1221 (D202,203)	VD1221
32	3	Diode, 1N4148 (D301,302,304)	1N4148
33	1	Diode, RGP250 (D303)	RGP25D
34	4	Diode, 1N4936 (D306-309)	1N4936
35	1	Diode, MR818 (D310)	MR818

Video Monitor Assembly #8790607 (TCE)

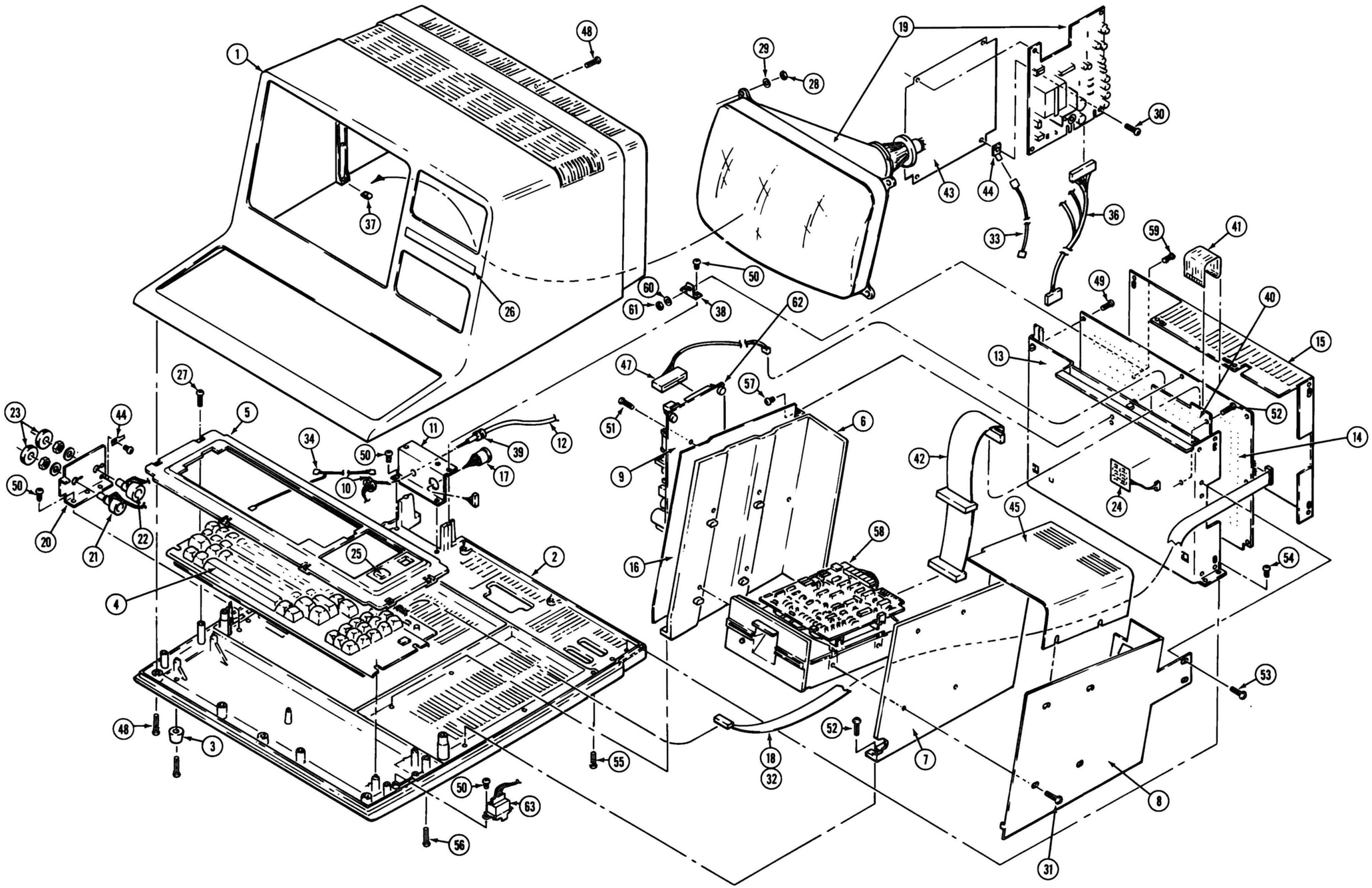
Item	Qty	Description	Mfgr's Part No.
36	1	IC, AN5763 V-Process (IC201)	AN5763
37	2	IC, NE555P Timer (IC301,302)	NE555P
38	1	Res, 220 ohm 1/4W 5% Car (R101)	RD1/4MZ(S)221J
39	2	Res, 47 ohm 1/4W 5% Car (R102,312)	RD1/4MZ(S)470J
40	1	Res, 18 ohm 1/4W 5% Car (R103)	RD1/4MZ(S)180J
41	1	Res, 33 ohm 1/4W 5% Car (R104)	RD1/4MZ(S)330J
42	1	Res, 47K ohm 1/4W 5% Car (R105)	RD1/4MZ(S)473J
43	2	Res, 1K ohm 1/4W 5% Car (R106,107)	RD1/4MZ(S)102J
44	1	Res, 820 ohm 5W 5% Cement (R108)	RT5P 821J
45	3	Res, 4.7K ohm 1/4W 5% Car (R201,202 207)	RD1/4MZ(S)472J
46	2	Res, 100 ohm 1/4W 5% Car (203,309)	RD1/4MZ(S)101J
47	2	Res, 56K ohm 1/4W 5% Car (R204,212)	RD1/4MZ(S)563J
48	1	Res, 6.8 ohm 1/4W 5% Car (R205)	RD1/4MZ(S)6R8J
49	1	Res, 1.2 ohm 1/2W 5% Car (R206)	RD1/2MZ(S)1R2J
50	3	Res, 220K ohm 1/4W 5% Car (208,314 319)	RD1/4MZ(S)224J
51	1	Res, 8.2 ohm 1/4W 5% Car (R210)	RD1/4MZ(S)8R2J
52	2	Res, 10K ohm 1/4W 5% Car (R211,302)	RD1/4MZ(S)103JZ
53	1	Res, 68K ohm 1/4W 5% Car (R212) - Used from 3001 unit -	RD1/4MZ(S)683J
54	1	Res, 1 ohm 1/2W 5% Car (R213)	RD1/2MZ(S)1R0J
55	1	Res, 6.8K ohm 1/4W 5% Car (R301)	RD1/4MZ(S)682J
56	3	Res, 5.6K ohm 1/4W 5% Car (R303,305 310)	RD1/4MZ(S)562J
57	1	Res, 560 ohm 1/4W 5% Car (R304)	RD1/4MZ(S)561J
58	1	Res, 33K ohm 1/2W 2% Car (R306)	RD1/2MZ(S)333G
59	1	Res, 6.2K ohm 1/2W 2% Car (R307)	RD1/2MZ(S)622G
60	1	Res, 39K ohm 1/4W 5% Car (R308)	RD1/4MZ(S)393J
61	1	Res, 100 ohm 1/2W 5% Car (R311)	RD1/2MZ(S)101J
62	1	Res, 56 ohm 2W 5% Metal Oxide (R313)	RSM2P560J
63	1	Res, 82K ohm 1/2W 5% Car (R315)	RD1/2MZ(S)823J
64	1	Res, 150K ohm 1/2W 5% Car (R316)	RD1/2MZ(S)154J
65	1	Res, 220K ohm 1/2W 5% Car (R317)	RD1/2MZ(S)224J
66	1	Res, 1M ohm 1/4W 5% Car (R318)	RD1/4MZ(S)105J
67	1	Res, 2.7M ohm 1/2W 5% Car (R320)	RD1/2MZ(S)275J
68	2	Res, 150 ohm 1/4W 5% Car (R321,322)	RD1/4MZ(S)151J
69	1	Res, 100K ohm 1/4W 5% Car (R323)	RD1/4MZ(S)104J
70	1	Transistor, NPN (TR101)	2SC1313(G)
71	1	Transistor, NPN (TR102)	2SC2228Y(E)
72	1	Transistor, NPN (TR301)	2SC536(F)
73	1	Transistor, PNP (TR302)	2SA733(P)
74	1	Transistor, ___ (TR303)	BU806

Video Monitor Assembly #8790607 (TCE)

Item	Qty	Description	Mfgr's Part No.
75	2	Variable Res, 100K ohm (VR201,203)	176910250A
76	1	Variable Res, 10K ohm (VR202)	176910220A
77	1	Variable Res, 200K ohm (VR203)	176910260A
78	1	Variable Res, 2M ohm (VR302)	176910400A
79	1	Variable Res, 500K ohm (VR303)	176910270A
		Miscellaneous	
80	2	Clip, Fuse	197303080A
81	1	Connector 4 Pin	194110780A
		End of Deflection Unit	
82	1	Cap, 1000 pfd 1000V +100-0% Cer (C501)	CK45E3A102PY
83	1	Cathode Ray Tube (VT501)	559010020A
84	1	Fuse, 2A @ 250V (FU401)	251000790A
85	2	Res, 470 ohm 1/2W 10% (R502,503)	RC1/2GF471K
86	2	Res, 100K ohm 1/2W 10% (R504,505)	RC1/2GF104K
87	1	Transformer, Flyback (T301)	10801001YA
		Miscellaneous	
88	1	Socket, Cylindrical	196310010A
89	1	Socket, 4 Pin	194010370A
90	1	Spring Tension, Wire Ground	434010020A
91	1	Wire, Ground Terminal	3160100608

SECTION VIII

ILLUSTRATED PARTS CATALOG


1
48
29
28
19
30
43
44
37
36
33
59
41
26
50
60
61
38
62
49
40
15
13
47
57
52
27
5
51
6
14
23
44
11
34
50
39
12
17
10
9
42
24
50
45
58
20
21
22
25
2
16
54
4
53
48
3
55
18
32
52
7
50
63
8
56
31

Parts List, Model 4 Computer, #26-1067/8/9

Item	Qty	Description	Mfgr's PN	RS Part No
1	1	Case Top	8719104	
2	1	Base	8719265	
3	4	Feet, Case	8590098	
4	1	Keyboard	8790524	
5	1	Keyboard Bezel	8719164	
6*	1	Bracket, Disk Mounting (LH)	8719106	
7*	1	Bracket, Disk Mounting (RH)	8719105	
8*	1	Shield, Disk (RH side)	8729093	
9	1	Power Supply Assy.(on 26-1067)	8790021	AXX5019
		(on 26-1068/1069)	8790043 or	
			8790049	
10	1	Toroid	8419030	
11	1	Bracket, Connector	8729039	ART3082
12	1	Cord, Power	8709412	
13	1	Chassis	8858073	
14	1	Main Logic PCB Assembly	6700104	
		(on 26-1067)		
		(on 26-1068/1069)	8858090	
15	1	Shield, Chassis	8729049	
16*	1	Shield, Disk (LH side)	8729041	
17	1	Cable Assembly, Cassette	8709372	
18	1	Cable, Flat	8709381	
19	1	Video Display, CRT (RCA)	8492002	AXX8010
		Video Display, CRT (TCE)	8790607	
20	1	Bracket, Pot Mounting	8729155	
21	1	Pot, 500 ohms (Contrast)	_____	
		(part of Item 36)		
22	1	Pot, 500K ohms (Brightness)	_____	
		(part of Item 36)		
23	2	Knob, Thumbwheel	8719112	AK4298
24*	1	Sound PCB Assembly	8858121	
		(26-1068/1069 only)		
25	1	Label, RAM size (16K)	8789261	AHC0321
		(64K)	8789800	
26	1	Name Plate	8719266	
27	6	Screw, #6 x 3/8"	8569047	
28	4	Nut, #10-24 Hex	8579021	
29	4	Washer, #8 Flat	8589016	
30	2	Screw, #6 x 3/8"	8569047	
31	8	Screw, #6-32 x 1/2"	8569046	
32	1	Cable, Keyboard Ground	8709275	

*Not used on Model #26-1067

Parts List, Model 4 Computer, #26-1067/8/9

Item	Qty	Description	Mfgr's PN	RS Part No
33	1	Cable, Ground	8709193	
34	1	Ground, Main	8709161	
35	1	Clip, Tinnerman	8559031	
36	1	Cable Assembly, CRT	8709369	
		for Models 26-1068/1069		
		for Model 26-1067	8709286	
37	5	Clip, Tinnerman		
38	1	Bracket, Support	8729055	
39	1	Strain Relief, Power Cord	8559014	
40*	1	FDC PCB Assembly	8858060	
			or 8858160	
41*	1	Flat Cable Assembly	8459020	
42*	1	Cable Assembly, Disk Drive	8709154	
43	1	Shield, CRT PCB Assembly	8539014	
44	2	Tab, Grounding	8529020	
45	1	Shield, Disk Drive Top	8729175	
		(used on 26-1069 only)		
46	1	Shield, Mylar	8539015	
47	1	Cable Assembly, DC Power	8709367	
		for 26-1068/9		
		for 26-1067	8709178	
48	3	Screw, #8-32 x 3/4" PPH Blk	8569047	
49	8	Screw, #6-32 x 1/2" PPH	8569046	
50	4	Screw, #6 x 1/4" Plastite	8569077	
51	6	Screw, #8-32 x 1/2"	8569140	
52	4	Screw, #6-32 x 1/4"	8569098	
53*	12	Screw, #6 x 3/8" Washer	8569128	
54	4	Screw, Chassis Mounting	8569077	
55	5	Screw, #8 x 1"	8569095	
56	2	Screw, #8-32 x 1"	8569084	
57	2	Screw, #6 x 3/8"	8569047	
58*	2**	Disk Drive Assembly	8790112	
59	2	Screw, #6-32 x 1/4"	8569098	
60	1	Washer, #6 Internal Star	8589043	
61	2	Nut, #6 Hex	8579014	
62*	3	Standoff, Power Supply	8589079	
63	1	Switch, DPST (ON-OFF)	8489030	

* Not Used On 26-1067 Computer
** Qty Of 1 On 26-1068 Computer

SECTION IX

RS-232C CIRCUIT BOARD

RS-232C CIRCUIT BOARD

9.1 RS-232C TECHNICAL DESCRIPTION

The RS-232C option board for the Model 4 computer supports asynchronous serial transmissions and conforms to the EIA RS-232C standards at the input - output interface (P1). The heart of the board is the TR1602 Asynchronous Receiver/Transmitter. It performs the job of converting the parallel byte date from the CPU to a serial data stream including start, stop, and parity bits. For a more detailed description of how this LSI circuit performs these functions, refer to the TR1602 data sheets and application notes. The transmit and receive clock rates that the TR1602 needs are supplied by the Baud rate generator (BR19411). This circuit takes the 5.0688 MHz supplied by the CPU board and the programmed information received from the CPU over the data bus and divides the basic clock rate to provide two clocks. The rates available from the BRG go from 50 Baud to 19200 Baud. See the BRG table for the complete list.

BRG PROGRAMMING TABLE

NIBBLE LOADED	TRANSMIT OR RECEIVE BAUD RATE	16X CLOCK FREQUENCY	SUPPORTED BY SETCOM
ØH	50	0.8 kHz	yes
1H	75	1.2 kHz	yes
2H	110	1.76 kHz	yes
3H	134.5	2.1523 kHz	yes
4H	150	2.4 kHz	yes
5H	300	4.8 kHz	yes
6H	600	9.6 kHz	yes
7H	1200	19.2 kHz	yes
8H	1800	28.8 kHz	yes
9H	2000	32.081 kHz	yes
AH	2400	38.4 kHz	yes
BH	3600	57.6 kHz	yes
CH	4800	76.8 kHz	yes
DH	7200	115.2 kHz	yes
EH	9600	153.6 kHz	yes
FH	19,200	307.2 kHz	yes

The RS-232C board is a port mapped device and the ports used are E8 to EB. Following is a description of each port on both input and output.

PORT	INPUT	OUTPUT
E8	Modem status	Master Reset, enables UART control register load
EA	UART status	UART control register load and modem control
E9	Not Used	Baud rate register load enable bit
EB	Receiver Holding register	Transmitter Holding register

Interrupts are supported on the RS-232C option board by the Interrupt mask register (U10) and the Status register (U9) which allows the CPU to see which kind of interrupt has occurred. Interrupts can be generated on receiver data register full, transmitter register empty, and any one of the errors – parity, framing, or data overun. This allows a minimum of CPU overhead in transferring data to or from the UART. The interrupt mask register is port EØ (write) and the interrupt status register is port EØ (read). Refer to the IO Port description for a full breakdown of all interrupts and their bit positions.

The Model 4 RS-232C board is functionally identical to the Model I RS-232 board with the following exceptions: Interrupts are supported, there are no sense switches for configuring the interface, there is no COM/TERM switch for reversing the function of pins 2 and 3 on the DB-25, and the DC to DC converter is not required since +12V and -12V are provided by the internal power supply. Other differences include three additional interface outputs and no crystal for the BRG. All Model I software written for the RS-232 interface is compatable with the Model 4 RS-232C option board, provided that the software does not use the sense switches to configure the interface. The programmer can get around this problem by directly programming the BRG and UART for the desired configuration or by using the SETCOM command of the disk operating system to configure the interface. The TRS-80 RS-232C Interface hardware manual has a good discussion of the RS-232C standard and specific programming examples (Catalog Number 26-1145).

9.2 PINOUT LISTING

The following list is a pinout description of the DB-25 connector (P1).

PIN#	SIGNAL
1	PGND (Protective Ground)
2	TD (Transmit Data)
3	RD (Receive Data)
4	RTS (Request To Send)
5	CTS (Clear To Send)
6	DSR (Data Set Ready)
7	SGND (Signal Ground)
8	CD (Carrier Detect)
20	DTR (Data Terminal Ready)
22	RI (Ring Indicate)

9.3 PORT AND BIT ASSIGNMENTS

PORT E8H
OUTPUT: MASTER RESET
INPUT: MODEM STATUS REGISTER

An output to this port (and data), performs a master reset to the UART and enables the control register load enable bit. The following table details the bit definitions for an input from port E8H.

DATA BIT	FUNCTION
D7	Clear To Send, Pin 5 DB-25
D6	Data Set Ready, Pin 6 DB-25
D5	Carrier Detect, Pin 8 DB-25
D4	Ring Indicator, Pin 22 DB-25
D3	Not Used
D2	Not Used
D1	Not Used
DØ	Receiver Input, UART Pin 20 DB-25

PORT E9H
OUTPUT: BAUD RATE LOAD
INPUT: NOT USED

An output to this port loads the Baud rate generator with a code which corresponds to the desired receive and transmit Baud rate as outlined in the BRG Programming Table. The low order nibble of the data output to this port determines the receiver Baud rate, while the high order nibble determines the transmit Baud rate.

PORT EAH
OUTPUT: UART AND MODEM CONTROL
INPUT: UART STATUS

An output to this port loads the UART Control register if the enable bit for this function is set (D1 port E8H = 1). The UART Control register is five bits wide (D7 - D3) leaving three bits for modem control (D2 - DØ). Three more modem control bits were added by allowing software to enable or disable the UART Control register. The tables below summarize the bit allocations with the UART Control register enabled and disabled.

PORT EAH OUTPUT BITS WITH UART CONTROL REGISTER ENABLED

DATA BIT	FUNCTION
D7	Even Parity Enable, 1 = even, Ø = odd
D6	Word Length Select 1
D5	Word Length Select 2
D4	Stop Bit Select, 1 = two stop bits, Ø = one stop bit
D3	Parity Inhibit, 1 = disable parity
D2	Break Ø = disable transmit data (continuous space)
D1	Data Terminal Ready, Pin 20 DB-25
DØ	Request To Send, Pin 4 DB-25

PORT EAH OUTPUT BITS WITH UART CONTROL REGISTER DISABLED

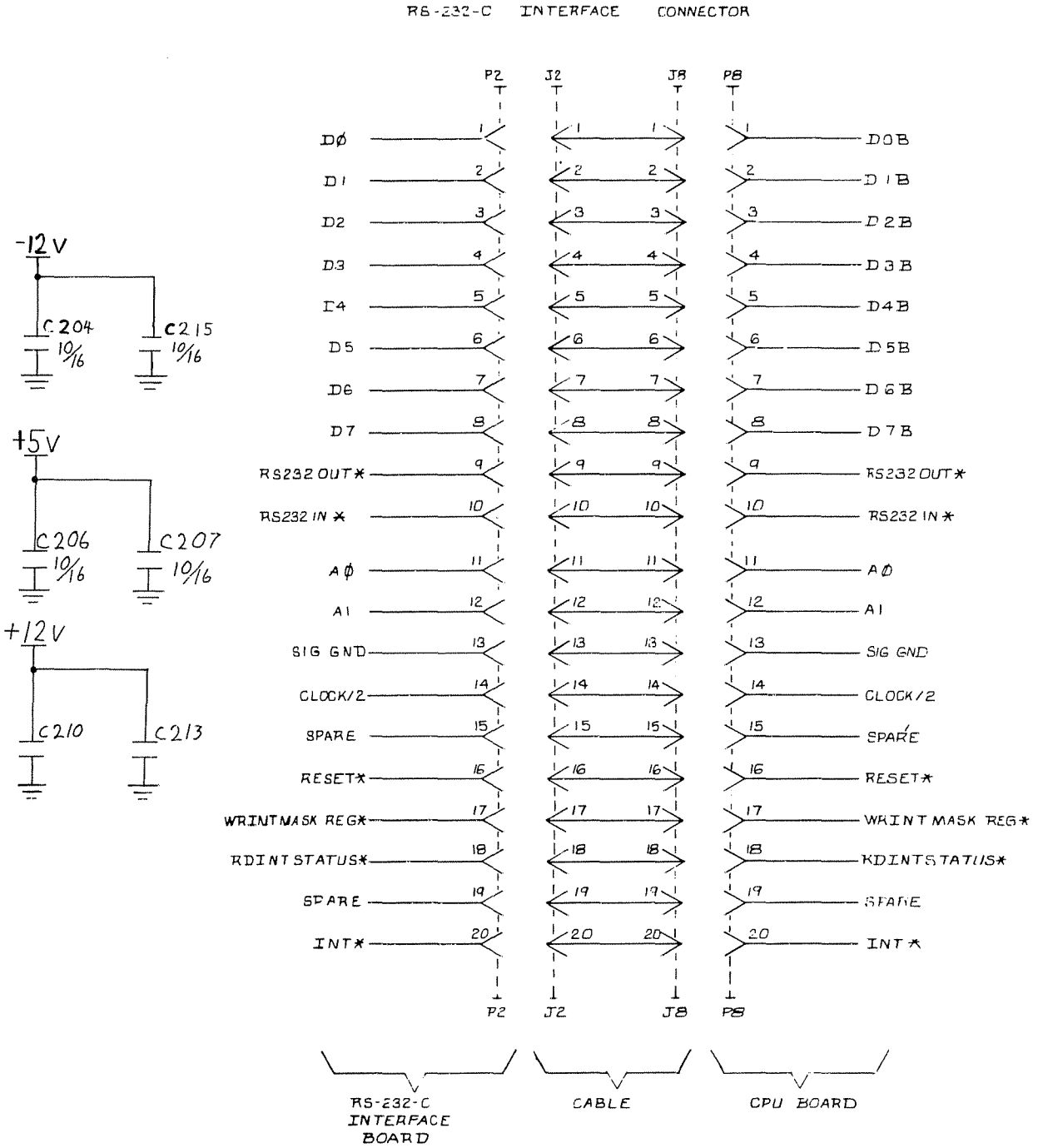
DATA BIT	FUNCTION
D7	Not Used
D6	Not Used
D5	Secondary unassigned, Pin 18 DB-25
D4	Secondary Transmit Data, Pin 14 DB-25
D3	Secondary Request To Send, Pin 19 DB-25
D2	Break Ø = disable Transmit Data (continuous space)
D1	Data Terminal Ready, Pin 20 DB-25
DØ	Request To Send, Pin 4 DB-25

PORT EAH INPUT BITS

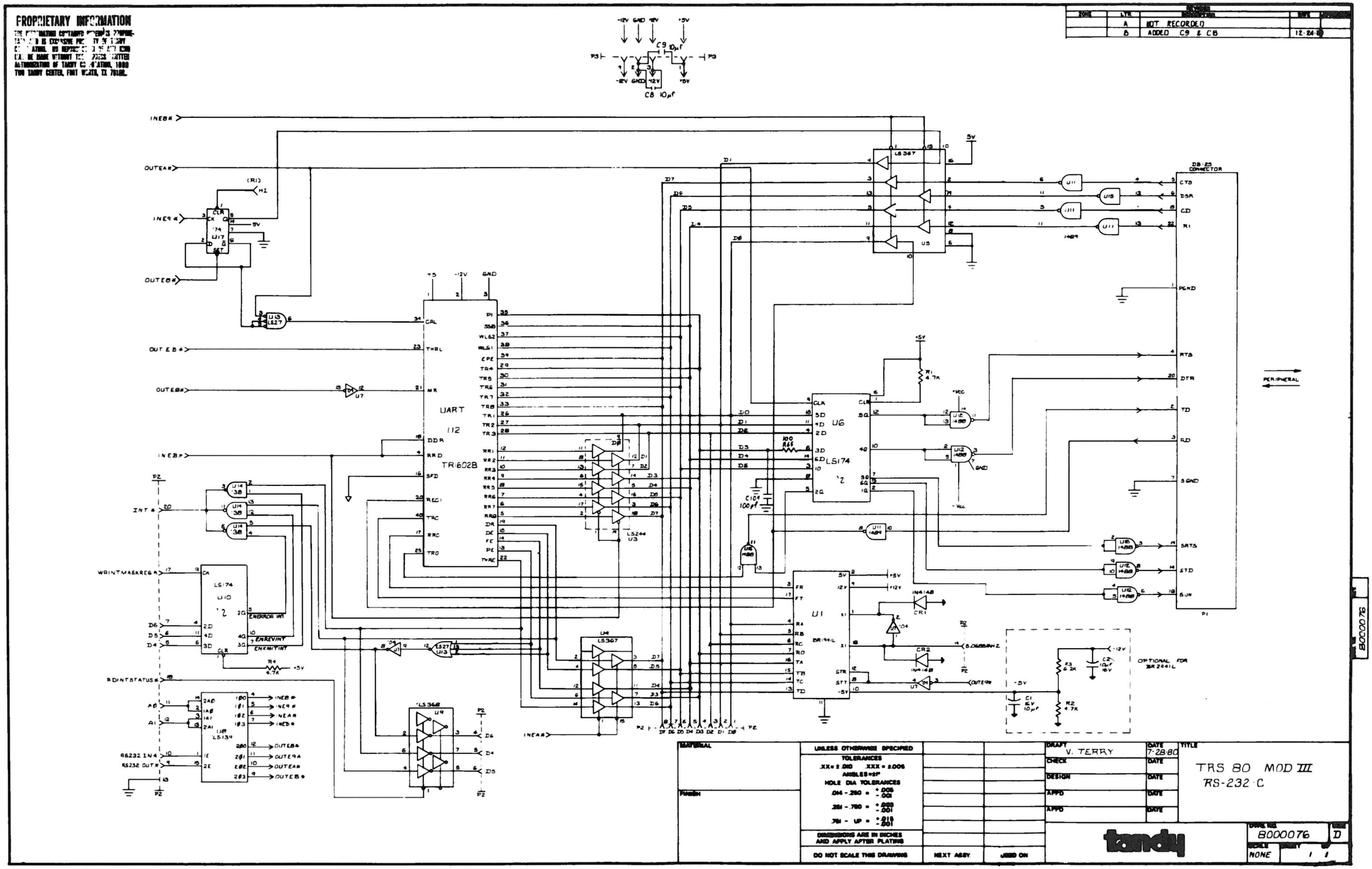
DATA BITS	FUNCTION
D7	Data Received, 1 = condition true
D6	Transmitter Holding register empty, 1 = condition true
D5	Overrun error, 1 = condition true
D4	Framing error, 1 = condition true
D3	Parity error, 1 = condition true
D2	Not Used
D1	Not Used
DØ	Not Used

PORT EBH
OUTPUT: TRANSMITTER HOLDING REGISTER
INPUT: RECEIVER HOLDING REGISTER

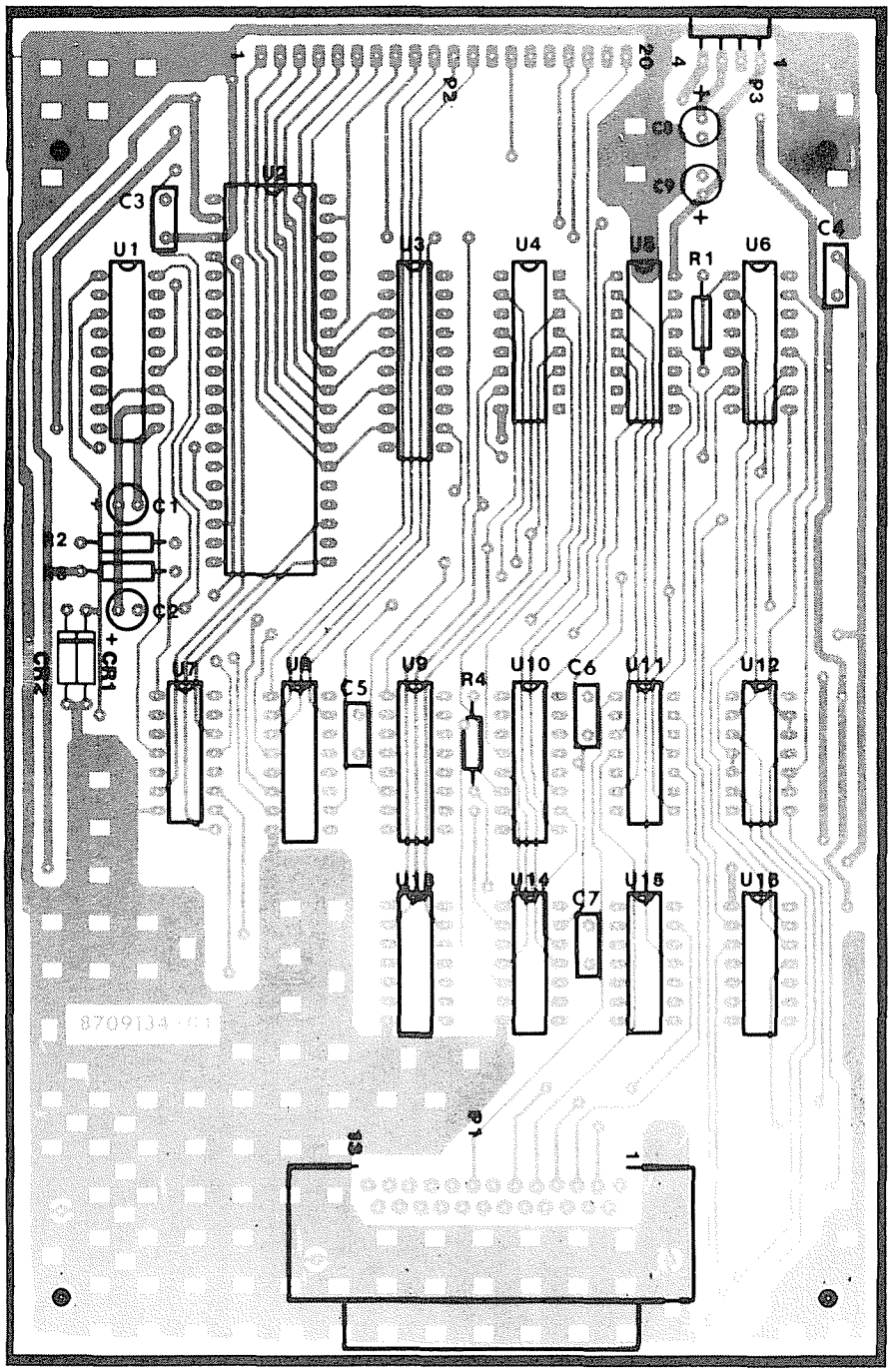
An output to this port loads the UART Transmitter Holding register with a word to be transmitted, as soon as the last word loaded in the holding register is transmitted. This register should never be loaded until the Transmitter Holding register empty bit (port EAH) is true. An input from this port reads the last word received from the UART received data holding register. This register should not be read until the data received bit (port EAH) is true.



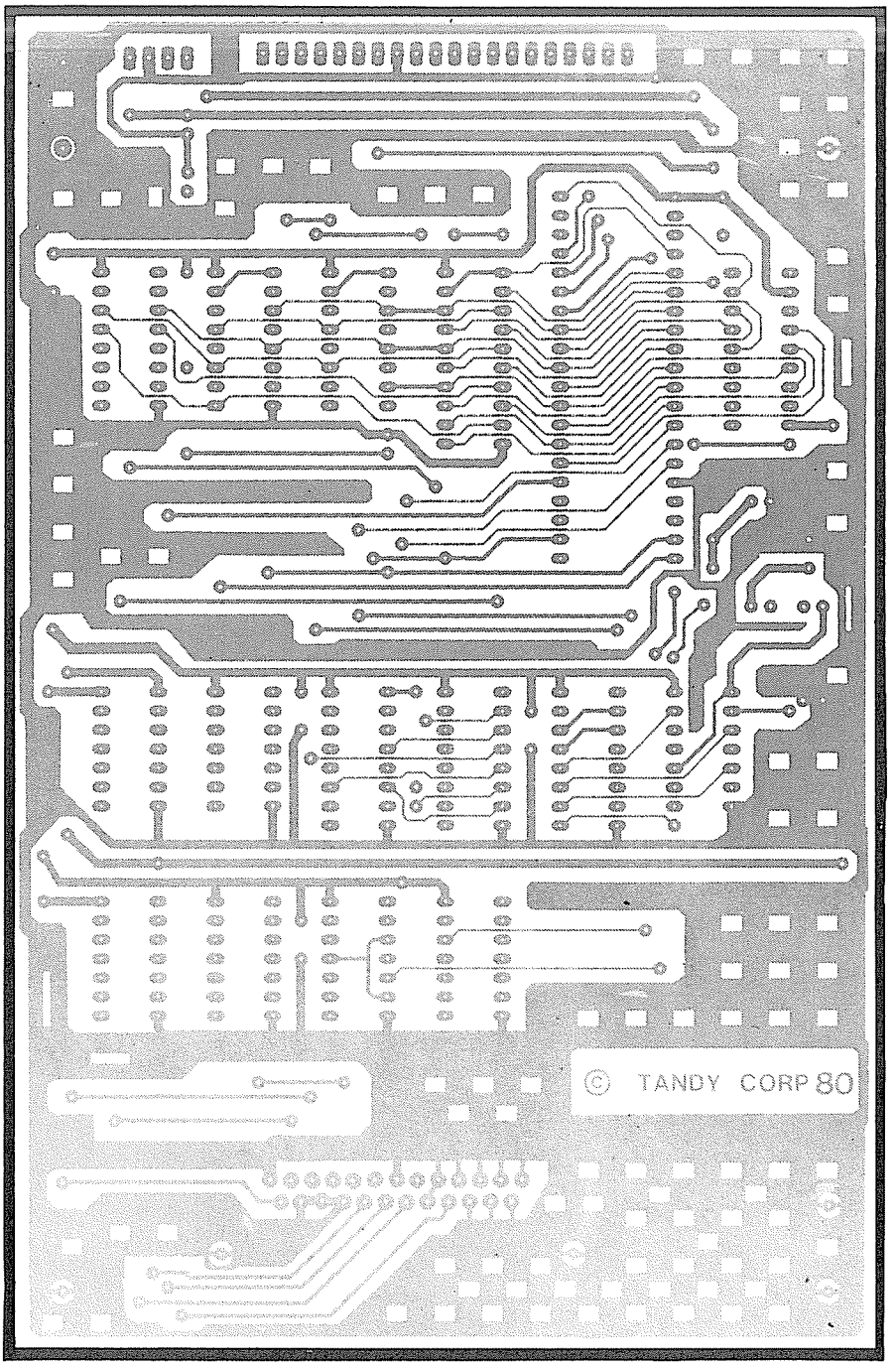
RS-232 BOARD TO CPU BOARD SIGNAL DESCRIPTION



SCHEMATIC DIAGRAM #8000076, RS232C, CHNG B



COMPONENT LOCATION/CIRCUIT TRACE, RS-232C PC BOARD – COMPONENT SIDE

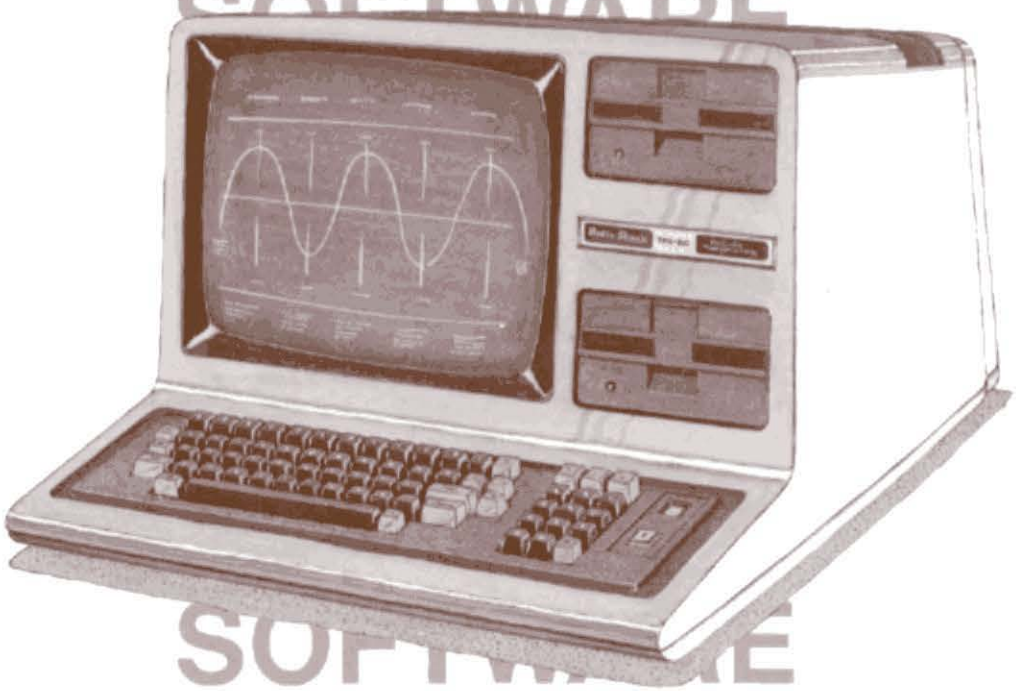


CIRCUIT TRACE, RS-232C PC BOARD — SOLDER SIDE

PARTS LIST RS-232C PC BOARD

SYMBOL	DESCRIPTION	MANUFACTURER'S PART NUMBER	RADIO SHACK PART NUMBER
	CAPACITORS		
C1	10μF, 16V, radial (optional)	832-6101	ACC106QDAP
C2	10μF, 16V, radial (optional)	832-6101	ACC106QDAP
C3	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C4	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C5	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C6	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C7	0.1μF, 50V, monolithic, radial	838-4104	ACC104QJAP
C8	10μF, 16V, radial	832-6101	ACC106QDAP
C9	10μF, 16V, radial	832-6101	ACC106QDAP
	CONNECTORS		
P1	DB-25 Connector	851-9030	----------
P2	20 pos. right angle	851-9078	----------
P3	4 pos. right angle	851-9079	AJ6977
	INTEGRATED CIRCUITS		
U1	BR1941-L, Dual Baud C	804-6941	AMX3921
U2	TR1602B, UART	804-5602	AMX3865
U3	74LS244, Octal Buffer	802-0244	AMX3864
U4	74LS367, Hex Buffer	802-0367	AMX3567
U5	74LS367, Hex Buffer	802-0367	AMX3567
U6	74LS174, Quad "D" Flip-Flop	802-0174	AMX3565
U7	7404, Hex Inverter	-----	AMX3655
U8	74LS139, Dual Decoder	802-0139	AMX3800
U9	74LS368, Hex Buffer	802-0368	AMX3568
U10	74LS174, Quad "D" Flip-Flop	802-0174	AMX3565
U11	MC1489, Quad Line Driver	805-0189	AMX3868
U12	MC1488, Quad Line Driver	805-0188	AMX3867
U13	74LS27, NOR gate	802-0027	----------
U14	74LS38, NAND Buffer	802-0038	AMX4328
U15	MC1489, Quad Line Driver	805-0189	AMX3868
U16	MC1488, Quad Line Driver	805-0188	AMX3867
	RESISTORS		
R1	4.7K, 1/4W, 5%	820-7247	AN0247EEC
R2	4.7K, 1/4W, 5% (optional)	820-7247	AN0247EEC
R3	6.2K, 1/4W, 5% (optional)	-----	AMX4658
R4	4.7K, 1/4W, 5%	820-7247	AN0247EEC
	MISCELLANEOUS		
	Cable, 20 pos., 4.5", flat	845-9020	AW2631
	Socket, 18 pin	850-9006	AJ6701
	Socket, 40 pin	850-9002	AJ6580

SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE



SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE
SOFTWARE

PART II
SOFTWARE

1/Disk Organization

TRSDOS Version 6 can be used with 5¼″ single-sided floppy diskettes and with hard disk. Floppy diskettes can be either single- or double-density. See the charts below for the number of sectors per track, number of cylinders, and so on for each type of disk. (Sectors and cylinders are numbered starting with 0.)

Single-Density Floppy Diskette

Bytes per Sector	Sectors per Granule	Sectors per Track*	Granules per Track	Tracks per Cylinder	Cylinders per Drive	Total Bytes
256						256
	5					1,280
		(10)	2			2,560
				1		2,560
					40	102,400
256	5	(10)	2	1	40	102,400 (100K)**

Double-Density Floppy Diskette

Bytes per Sector	Sectors per Granule	Sectors per Track*	Granules per Track	Tracks per Cylinder	Cylinders per Drive	Total Bytes
256						256
	6					1,536
		(18)	3			4,608
				1		4,608
					40	184,320
256	6	(18)	3	1	40	184,320 (180K)**

*The number of sectors per track is not included in the calculation because it is equal to the number of sectors per granule times the number of granules per track. ($5 \times 2 = 10$ for single density, $6 \times 3 = 18$ for double density, and $16 \times 2 = 32$ for hard disk.)

**Note that this figure is the total amount of space in the given format. Keep in mind that an entire cylinder is used for the directory and at least one granule is used for the bootstrap code. This leaves 96.25K available for use on a single-density data disk and 174K on a double-density data disk.

5″ 5-Meg Hard Disk

Note: Because of continual advancements in hard disk technology, the number of tracks and the number of tracks per cylinder may change. Therfore, any information that comes with your hard disk drive(s) supersedes the information in the table below.

Bytes per Sector	Sectors per Granule	Sectors per Track*	Granules per Track	Tracks per Cylinder	Cylinders per Drive	Total Bytes
256						256
	16					4,096
		(32)	2			8,192
				4		32,768
					153	5,013,504
256	16	(32)	2	4	153	5,013,504 (4,896K)

*The number of sectors per track is not included in the calculation because it is equal to the number of sectors per granule times the number of granules per track. ($5 \times 2 = 10$ for single density, $6 \times 3 = 18$ for double density, and $16 \times 2 = 32$ for hard disk.)

Disk Space Available to the User

One granule on cylinder 0 of each disk is reserved for the system. It contains information about where the directory is located on that disk. If the disk contains an operating system, then all of cylinder 0 is reserved. This area contains information used to load TRSDOS when you press the reset button.

One complete cylinder is reserved for the directory, the granule allocation table (GAT), and the hash index table (HIT). (On single-sided diskettes, one cylinder is the same as one track.) The number of this cylinder varies, depending on the size and type of disk. Also, if any portion of the cylinder normally used for the directory is flawed, TRSDOS uses another cylinder for the directory. You can find out where the FORMAT utility has placed the directory by using the Free *:drive* command.

On hard disks, an additional cylinder (cylinder 1) is reserved for use in case your disk drive requires service. This provides an area for the technician to write on the disk without harming any data. (If you bring your hard disk in for service, you should try to back up the contents of the disk first, just to be safe.)

Unit of Allocation

The smallest unit of disk space that the system can allocate to a file is a granule. A granule is made up of a set of sectors that are adjacent to one another on the disk. The number of sectors in a granule depends on the type and size of the disk. See the charts on the previous two pages for some typical sizes.

2/Disk Files

Methods of File Allocation

TRSDOS provides two ways to allocate disk space for files: dynamic allocation and pre-allocation.

Dynamic Allocation

With dynamic allocation, TRSDOS allocates granules only at the time of write. For example, when a file is first opened for output, no space is allocated. The first allocation of space is done at the first write. Additional space is added as required by further writes.

With dynamically allocated files, unused granules are de-allocated (recovered) when the file is closed.

Unless you execute the CREATE system command, TRSDOS uses dynamic allocation.

Pre-Allocation

With pre-allocation, the file is allocated a specified number of granules when it is created. Pre-allocated files can be created only by the system command CREATE. (See the *Disk System Owner's Manual* for more information on CREATE.)

TRSDOS automatically extends a pre-allocated file as needed. However, it does not de-allocate unused granules when a pre-allocated file is closed. To reduce the size of a pre-allocated file, you must copy it to a dynamically allocated file. The COPY (CLONE = N) system command does this automatically.

Files that have been pre-allocated have a 'C' by their names in a directory listing.

Record Length

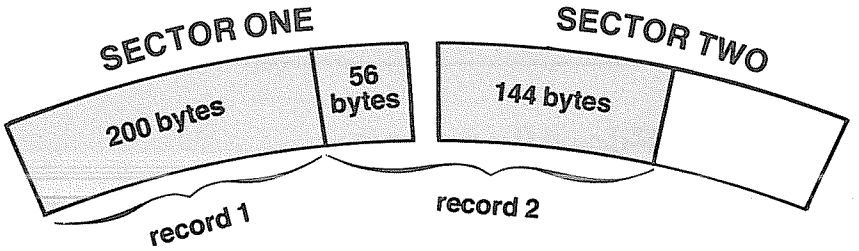
TRSDOS transfers data to and from disks one sector at a time. These sectors are 256-byte blocks, and are also called the system's "physical" records.

You deal with records that are 256 bytes in length or smaller, depending on what size record you want to work with. These are known as "logical" records.

You set the size of the logical records in a file when you open the file for the first time. The size is the number of bytes to be kept in each record. There may be from 1 to 256 bytes per logical record.

The operating system automatically accumulates your logical records and stores them in physical records. Since physical records are always 256 bytes in length, there may be one or more logical records stored in each physical record. When the records are read back from disk, the system automatically returns one logical record at a time. These actions are known as "blocking" and "de-blocking," or "spanning."

For example, if the logical record length is 200, sectors 1 and 2 look like this:



Since they are completely handled by the operating system, you do not need to concern yourself with physical records, sectors, granules, tracks, and so on. This is to your benefit, as the number of sectors per granule varies from disk to disk. Also, physical record lengths may change in future versions of TRSDOS, but the concept of logical records will not.

Note: All files are fixed-length record files with TRSDOS Version 6.

Record Processing Capabilities

TRSDOS allows both direct and sequential file access.

Direct access (sometimes called "random access") lets you process records in any sequence you specify.

Sequential access allows you to process records in sequence: record n, n+1, n+2, and so on. With sequential access, you do not specify a record number. Instead, TRSDOS accesses the record that follows the last record processed, starting with record Ø.

With sequential access files, use the @READ supervisor call to read the next record, and the @WRITE or @VER supervisor call to write the next record. (When the file is first opened, processing starts at record Ø. You can use @PEOF to position to the end of file.)

To read or write to a direct access file, use the @POSN supervisor call to position to a specified record. Then use @READ, @WRITE, or @VER as desired. Once @POSN has been used, the End of File (EOF) marker will not move, unless the file is extended by writing past the current EOF position.

Record Numbers

Using direct (random) access, you can access up to 65,536 records. Record numbers start at Ø and go to 65535.

Using a file sequentially, you can access up to 16,777,216 bytes. To calculate the number of records you can access sequentially, use the formula:

16,777,216 ÷ logical record length = number of sequential records allowed

Below are some examples.

If the LRL = 256, then:
16,777,216 ÷ 256 = 65,536 records

If the LRL = 128, then:
16,777,216 ÷ 128 = 131,Ø72 records

If the LRL = 5Ø, then:
16,777,216 ÷ 5Ø = 335,544 records

If the LRL = 1, then:
16,777,216 ÷ 1 = 16,777,216 records

3/TRSDOS File Descriptions

This section describes four types of files found on your TRSDOS master diskette (system files, utilities, driver programs, and filter programs) and explains their functions. It also describes how to construct a minimum system disk for running applications packages.

System Files (/SYS)

TRSDOS Version 6 would occupy considerable memory space if all of it were resident in memory at any one time. To minimize the amount of memory reserved for system use, TRSDOS uses overlays.

Using an overlay-driven system involves some compromise. While a user's application is in progress, different overlays may need to be loaded to perform certain activities requested of the system. This could cause the system to run slightly slower than a system which has more of its file access routines always resident in memory.

The use of overlays also requires that a SYSTEM disk usually be available in Drive Ø (the system drive). Since the disk containing the operating system and its utilities leaves little space available to the user, you may want to remove certain parts of the system software not needed while a particular application is running. You may in fact discover that your day-to-day operations need only a minimal TRSDOS configuration. The greater the number of system functions unnecessary for your application, the more space you can have available for a "working" system disk. Use the PURGE or REMOVE library command to eliminate unneeded system files from the disk.

The following paragraphs describe the functions performed by each system overlay. (In the display produced by the DIR (SYS) library command, the system overlays are identified by the file extension /SYS.)

Note: Two system files are put on the disk during formatting. They are DIR/SYS and BOOT/SYS. These files should *never* be copied from one disk to another or REMOVEd. TRSDOS automatically updates any information necessary when performing a backup.

SYSØ/SYS

This is not an overlay. It contains the resident part of the operating system (SYSRES). It is also needed to dynamically allocate file space used when writing files. Any disk used for booting the system *must* contain SYSØ. It can be purged from disks not used for booting.

SYS1/SYS

This overlay contains the TRSDOS command interpreter and the routines for processing the @CMNDI, @CMNDR, @FEXT, @FSPEC, and @PARAM system vectors. This overlay must be available on all SYSTEM disks.

SYS2/SYS

This overlay is used for opening or initializing disk files and logical devices. It also contains routines for processing the @CKDRV, @GTDCB, and @RENAM system vectors, and routines for hashing file specifications and passwords. This overlay must be available on all SYSTEM disks.

SYS3/SYS

This overlay contains all of the system routines needed to close files and logical devices. It also contains the routines needed to service the @FNAME system vector. This overlay must not be removed from the disk.

SYS4/SYS

This overlay contains the system error dictionary. It is needed to issue such messages as "File not found," "Directory read error," etc. If you decide to remove this overlay from your working SYSTEM disk, all system errors will produce the error message "SYS ERROR." It is recommended that you not remove this overlay, especially since it occupies only one granule of space.

SYS5/SYS

This is the "ghost" debugger. It is needed if you intend to test out machine language application software by using the TRSDOS DEBUG library command. If your operation will not require this debugging tool, you may purge this overlay.

SYS6/SYS

This overlay contains all of the routines necessary to service the library commands identified as "Library A" by the LIB command. This represents the primary library functions. Only very limited use can be made of TRSDOS if this overlay is removed from your working SYSTEM disk.

SYS7/SYS

This overlay contains all of the routines necessary to service the library commands identified as "Library B" by the LIB command. A great deal of use can be made of TRSDOS even without this overlay. It performs specialized functions that may not be needed in the operation of specific applications. You can purge this overlay if you decide it is not needed on a working SYSTEM disk.

SYS8/SYS

This overlay contains all of the routines necessary to service the library commands identified as "Library C" by the LIB command. A great deal of use can be made of TRSDOS even without this overlay. It performs specialized functions that may not be needed in the operation of specific applications. You can purge this overlay if you decide it is not needed on a working SYSTEM disk.

SYS9/SYS

This overlay contains the routines necessary to service the extended DEBUG commands available after a DEBUG (EXT) is performed. This overlay may be purged if you will not need the extended DEBUG commands while running your application. If you remove SYS5/SYS, then you may as well remove SYS9/SYS, as it would serve no useful purpose.

SYS10/SYS

This system overlay contains the procedures necessary to service the request to remove a file. It should remain on your working SYSTEM disks.

SYS11/SYS

This overlay contains all of the procedures necessary to perform the Job Control Language execution phase. You may remove this overlay from your working disks if you do not intend to execute any JCL functions. If SYS6/SYS (which contains the DO command) has been removed, keeping this overlay would serve no purpose.

SYS12/SYS

This system overlay contains the routines that service the @DODIR, @GTMOD, and @RAMDIR system vectors. It should remain on your disks.

SYS13/SYS

This overlay is reserved for future system use. It contains no code and takes up no space on the disk. You may remove this overlay if you wish to free up its directory slot.

Utility Programs

BACKUP — Used to duplicate data from one disk to another.

COMM — A communications package for use with the RS-232C hardware.

CONV — Used to copy files from Model III TRSDOS to TRSDOS Version 6.

FORMAT — Used to put track, sector, and directory information on a disk.

LOG — Used to log in a double-sided diskette in Drive 0. Also updates the Drive Code Table information as with the DEVICE library command.

PATCH — Used to make changes to existing files.

REPAIR — Used to correct certain information on non-TRSDOS formatted diskettes.

TAPE100 — A disk/tape, tape/disk utility for cassette tape operations with the TRS-80 Model 100.

Device Driver Programs

COM/DVR — The RS-232C communications driver.

FLOPPY/DCT — Configures floppy drives in the system. Not needed with a floppy-only system.

JL/DVR — The Joblog driver program.

MEMDISK/DCT — Used to establish a pseudo floppy drive in memory.

Filter Programs

CLICK/FLT — Produces a short tone as each key is pressed.

FORMS/FLT — Used to select printer parameters and perform character translation.

KSM/FLT — The Keystroke Multiply feature, which allows the assigning of user-determined phrases to alphabetic keys.

Creating a Minimum Configuration Disk

All files except certain /SYS files may be purged from your Drive 0 disk. Additionally, if you place the needed /SYS files in high memory with the SYSTEM (SYSRES) command, it will be possible to run with a minimum configuration disk in Drive 0 after booting the system. Keep the following points in mind when purging system files:

- For operation, SYS files 1, 2, 3, 4, 10, and 12 should remain on the Drive 0 disk or be resident in memory.
- SYS2 must be on the system disk if a configuration file is to be loaded.

- SYS11 must be present only if any JCL files will be used.
- All three libraries (SYS files 6, 7, and 8) may be purged if no library command will be used.
- SYS5 and SYS9 may be purged if the system DEBUG package is not needed.
- SYS0 may be removed from any disk not used for booting.
- SYS11 (the JCL processor) and SYS6 (containing the DO library command) must both be on the disk if the DO command is to be used. Also, if you remove SYS6, you may as well remove SYS11.
- SYS13 (reserved for future use) may be removed to free up an additional directory slot.

The presence of any utility, driver, or filter program is dependent upon your individual needs. You can save most of the TRSDOS features in a configuration file using the SYSTEM (SYSGEN) command, so the driver and filter programs will not be needed in run time applications.

The owner (update) passwords for TRSDOS files are as follows:

File Type	Extension	Owner Password
System files	(/SYS)	LSIDOS
Filter files	(/FLT)	FILTER
Driver files	(/DVR)	DRIVER
Utility files	(/CMD)	UTILITY
BASIC		BASIC
BASIC overlays	(/OV$)	BASIC
CONFIG/SYS		CCC
Drive Code Table Initializer	(/DCT)	UTILITY

Device Control Block (DCB)

The Device Control Block (DCB) is an area of memory that contains information used to interface the operating system with various logical devices. These devices include the keyboard (*KI), the video display (*DO), a printer (*PR), a communications line (*CL), and other devices that you may define.

The following information describes each assigned DCB byte.

DCB + 0 (TYPE Byte)

Bit 7 — If set to "1," the Device Control Block is actually a File Control Block (FCB) with the file open. Since DCBs and FCBs are similar, and devices may be routed to files, a "device" with this bit set indicates a routing to a file.

Bit 6 — If set to "1," the device defined by the DCB is filtered or is a device filter.

Bit 5 — If set to "1," the device defined by the DCB is linked.

Bit 4 — If set to "1," the device defined by the DCB is routed.

Bit 3 — If set to "1," the device defined by the DCB is a NIL device. Any output directed to the device is discarded. For any input request, the character returned is a null (ASCII value 0).

Bit 2 — If set to "1," the device defined by the DCB can handle requests generated by the @CTL supervisor call. See the section on Supervisor Calls for more information.

Bit 1 — If set to "1," the device defined by the DCB can handle output requests which normally come from the @PUT supervisor call.

Bit 0 — If set to "1," the device defined by the DCB can handle requests for input which normally come from the @GET supervisor call.

DCB + 1 and DCB + 2

Contain the address of the driver routine that supports the hardware assigned to this DCB. (In the case of a routed or linked device, the vector may point to another DCB.)

DCB + 3 through DCB + 5

Reserved for system use.

DCB + 6 and DCB + 7

These locations normally contain the two alphabetic characters of the devspec. The system uses the devspec as a reference in searching the device control block tables.

Memory Header

Modules that TRSDOS loads into memory (filters, drivers, and other memory modules such as a SPOOL buffer or the extended DEBUG code) are identified by a standard front-end header:

```
BEGIN:  JR   START          ;Go to actual code
                            ;beginning
        DEFW END-1          ;Contains the highest byte
                            ;of memory
                            ;used by the module
        DEFB 10             ;Length of name, 1-15
                            ;characters;
                            ;bits 4-7 reserved for
                            ;system use
        DEFM 'NAMESTRING'   ;Up to 15 alphanumeric
                            ;characters, with the first
                            ;character A-Z. This should
                            ;be a unique name to
                            ;positively identify the
                            ;module.
MODDCB: DEFW $-$            ;DCB pointing to this
                            ;module (if applicable)
        DEFW 0              ;Spare system pointer _
                            ;RESERVED
;
;       Any additional data storage goes here
;
START:  Start of actual program code
        .
END:    EQU $
```

As explained under the @GTMOD SVC in the "Supervisor Call" section, the location of a specific header can be found provided all modules that are put into memory use this header structure. You can locate the data area for a module by using @GTMOD to find the start of the header and then indexing in to the data area.

5/Drive Access

Drive Code Table (DCT)

TRSDOS uses a Drive Code Table (DCT) to interface the operating system with specific disk driver routines. Note especially the fields that specify the allocation scheme for a given drive. This data is essential in the allocation and accessibility of file records.

The DCT contains eight 10-byte positions — one for each logical drive designated 0-7. TRSDOS supports a standard configuration of four floppy drives. This is the default initialization when TRSDOS is loaded.

Here is the Drive Code Table layout:

DCT+0

This is the first byte of a 3-byte vector to the disk I/O driver routines. This byte is normally X'C3.' If the drive is disabled or has not been configured (see the SYSTEM command in the *Disk System Owner's Manual*), this byte is a RET instruction (X'C9').

DCT+1 and DCT+2

Contain the entry address of the routines that drive the physical hardware.

DCT+3

Contains a series of flags for drive specifications.

Bit 7 — Set to "1" if the drive is software write protected, "0" if it is not. (See the SYSTEM command in the *Disk System Owner's Manual.*)

Bit 6 — Set to "1" for DDEN (double density), or "0" for SDEN (single density).

Bit 5 — Set to "1" if the drive is an 8" drive. Set to "0" if it is a 5¼" drive.

Bit 4 — A "1" causes the selection of the disk's second side. The first side is selected if this bit is "0." This bit value matches the side indicator bit in the sector header written by the Floppy Disk Controller (FDC).

Bit 3 — A "1" indicates a hard drive (Winchester). A "0" denotes a floppy drive (5¼" or 8").

Bit 2 — Indicates the time delay between selection of a 5¼" drive and the first poll of the status register. A "1" value indicates 0.5 second and a "0" indicates 1.0 second. See the SYSTEM command in the *Disk System Owner's Manual* for more details.

If the drive is a hard drive, this bit indicates either a fixed or removable disk: "1" = fixed, "0" = removable.

Bits 1 and 0 — Contain the step rate specification for the Floppy Disk Controller. (See the SYSTEM command in the *Disk System Owner's Manual.*) In the case of a hard drive, this field may indicate the drive address (0-3).

DCT+4

Contains additional drive specifications.

Bit 7 — Reserved for future use. In order to maintain compatibility with future releases of TRSDOS, do not use this bit.

Bit 6 — If "1," the controller is capable of double-density mode.

Bit 5 — "1" indicates that this is a 2-sided floppy diskette; "0" indicates a 1-sided floppy disk. Do not confuse this bit with Bit 4 of DCT + 3. This bit shows if the disk is double-sided; Bit 4 of DCT + 3 tells the controller what side the current I/O is to be on.

If the hard drive bit (DCT + 3, Bit 3) is set, a "1" denotes double the cylinder count stored in DCT + 6. (This implies that a logical cylinder is made up of two physical cylinders.)

Bit 4 — If "1," indicates an alien (non-standard) disk controller.

Bits 0-3 — Contain the physical drive address by bit selection (0001, 0010, 0100, and 1000 equal logical Drives 0, 1, 2, and 3, respectively, in a default system). The system supports a translation only where no more than one bit can be set.

If the alien bit (Bit 4) is set, these bits may indicate the starting head number.

DCT + 5

Contains the current cylinder position of the drive. It normally stores a copy of the Floppy Disk Controller's track register contents whenever the FDC is selected for access to this drive. It can then be used to reload the track register whenever the FDC is reselected.

If the alien bit (DCT + 4, Bit 4) is set, DCT + 5 may contain the drive select code for the alien controller.

DCT + 6

Contains the highest numbered cylinder on the drive. Since cylinders are numbered from zero, a 35-track drive is recorded as X'22,' a 40-track drive as X'27,' and an 80-track drive as X'4F.' If the hard drive bit (DCT + 3, Bit 3) is set, the true cylinder count depends on DCT + 4, Bit 5. If that bit is a "1," DCT + 6 contains only half of the true cylinder count.

DCT + 7

Contains allocation information.

Bits 5-7 — Contain the number of heads for a hard drive.

Bits 0-4 — Contain the highest numbered sector relative to zero. A 10-sector-per-track drive would show X'09.' If DCT + 4, Bit 5 indicates 2-sided operation, the sectors per cylinder equals twice this number.

DCT + 8

Contains additional allocation information.

Bits 5-7 — Contain the number of granules per track allocated in the formatting process. If DCT + 4, Bit 5 indicates 2-sided operation, the granules per cylinder equals twice this number. For a hard drive, this number is the total granules per cylinder.

Bits 0-4 — Contain the number of sectors per granule that was used in the formatting operation.

DCT + 9

Contains the number of the cylinder where the directory is located. For any directory access, the system first attempts to use this value to read the directory. If this operation is unsuccessful, the system examines the BOOT granule (cylinder 0) directory address byte.

Bytes DCT + 6, DCT + 7, and DCT + 8 must relate without conflicts. That is, the highest numbered sector (+ 1) divided by the number of sectors per granule (+ 1) must equal the number of granules per track (+ 1).

Disk I/O Table

TRSDOS interfaces with hardware peripherals by means of software drivers. The drivers are, in general, coupled to the operating system through data parameters stored in the system's many tables. In this way, hardware not currently supported by TRSDOS can easily be supported by generating driver software and updating the system tables.

Disk drive sub-systems (such as controllers for 5¼″ drives, 8″ drives, and hard disk drives) have many parameters addressed in the Drive Code Table (DCT). Besides those operating parameters, controllers also require various commands (SELECT, SECTOR READ, SECTOR WRITE, and so on) to control the physical devices. TRSDOS has defined command conventions to deal with most commands available on standard Disk Controllers.

The function value (hexadecimal or decimal) you wish to pass to the driver should go in register B. The available functions are:

Hex	Dec	Function	Operation Performed
X'00'	0	DCSTAT	Test to see if drive is assigned in DCT
X'01'	1	SELECT	Select a new drive and return status
X'02'	2	DCINIT	Set to cylinder 0, restore, set side 0
X'03'	3	DCRES	Reset the Floppy Disk Controller
X'04'	4	RSTOR	Issue FDC RESTORE command
X'05'	5	STEPI	Issue FDC STEP IN command
X'06'	6	SEEK	Seek a cylinder
X'07'	7	TSTBSY	Test to see if requested drive is busy
X'08'	8	RDHDR	Read sector header information
X'09'	9	RDSEC	Read sector
X'0A'	10	VRSEC	Verify if the sector is readable
X'0B'	11	RDTRK	Issue an FDC track read command
X'0C'	12	HDFMT	Format the device
X'0D'	13	WRSEC	Write a sector
X'0E'	14	WRSYS	Write a system sector (for example, directory)
X'0F'	15	WRTRK	Issue an FDC track write command

Function codes X'10' to X'FF' are reserved for future use.

Directory Records (DIREC)

The directory contains information needed to access all files on the disk. The directory records section is limited to a maximum of 32 sectors because of physical limitations in the Hash Index Table. Two additional sectors in the directory cylinder are used by the system for the Granule Allocation Table and the Hash Index Table. The directory is contained on one cylinder. Thus, a 10-sector-per-cylinder formatted disk has, at most, eight directory sectors. See the section on the Hash Index Table for the formula to calculate the number of directory sectors.

A directory record is 32 bytes in length. Each directory sector contains eight directory records (256/32 = 8). On system disks, the first two directory records of the first eight directory sectors are reserved for system overlays. The total

number of files possible on a disk equals the number of directory sectors times eight (since 256/32 = 8). The number available for use is reduced by 16 on system disks to account for those record slots reserved for the operating system. The following table shows the directory record capacity (file capacity) of each format type. The dash suffix (-1 or -2) on the items in the density column represents the number of sides formatted (for example, SDEN-1 means single density, 1-sided).

	Sectors per Cylinder	Directory Sectors	User Files on Data Disk**	User Files on SYS Disk
5″ SDEN-1	10	8	62	48
5″ SDEN-2	20	18	142	128
5″ DDEN-1	18	16	126	112
5″ DDEN-2	36	32	254	240
8″ SDEN-1	16	14	110	96
8″ SDEN-2	32	30	238	224
8″ DDEN-1	30	28	222	208
8″ DDEN-2	60	32	254	240
5-MEG HARD*				

*Hard drive format depends on the drive size and type, as well as the user's division of the physical drive into logical drives. After setting up and formatting the drive, you can use the FREE library command to see the available files.

**Note: Two directory records are reserved for BOOT/SYS and DIR/SYS, and are not included in the figures for this column.

TRSDOS Version 6 is upward compatible with other TRSDOS 2.3 compatible operating systems in its directory format. The data contained in the directory has been extended. An SVC is included to either display an abbreviated directory or place its data in a user-defined buffer area. For detailed information, see the @DODIR and @RAMDIR SVCs.

The following information describes the contents of each directory field:

DIR + 0

Contains all attributes of the designated file.

Bit 7 — If "0," this flag indicates that the directory record is the file's primary directory entry (FPDE). If "1," the directory record is one of the file's extended directory entries (FXDE). Since a directory entry can contain information on up to four extents (see notes on the extent fields, beginning with DIR + 22), a file that is fractured into more than four extents requires additional directory records.

Bit 6 — Specifies a SYStem file if "1," a nonsystem file if "0."

Bit 5 — If set to "1," indicates a Partition Data Set (PDS) file.

Bit 4 — Indicates whether the directory record is in use or not. If set to "1," the record is in use. If "0," the directory record is not active, although it may appear to contain directory information. In contrast to some operating systems that zero out the directory record when you remove a file, TRSDOS only resets this bit to zero.

Bit 3 — Specifies the visibility. If "1," the file is INVisible to a directory display or other library function where visibility is a parameter. If a "0," then the file is VISible. (The file can be referenced if specified by name by an @INIT or @OPEN SVC.)

Bits 0-2 — Contain the USER protection level of the file. The 3-bit binary value is one of the following:

0 = FULL	2 = RENAME	4 = UPDATE	6 = EXECUTE
1 = REMOVE	3 = WRITE	5 = READ	7 = NO ACCESS

DIR + 1

Contains various file flags and the month field of the packed date of last modification.

Bit 7 — Set to "1" if the file was "CREATEd" (see CREATE library command in the *Disk System Owner's Manual*). Since the CREATE command can reference a file that is currently existing but non-CREATEd, it can turn a non-CREATEd file into a CREATEd one. You can achieve the same effect by changing this bit to a "1."

Bit 6 — If set to "1," the file has not been backed up since its last modification. The BACKUP utility is the only TRSDOS facility that resets this flag. It is set during the close operation if the File Control Block (FCB + Ø, Bit 2) shows a modification of file data.

Bit 5 — If set to "1," indicates a file in an open condition with UPDATE access or greater.

Bit 4 — If the file was modified during a session where the system date was not maintained, this bit is set to "1." This specifies that the packed date of modification (if any) stored in the next three fields is not the actual date the modification occurred. If this bit is "1," the directory command displays plus signs (+) between the date fields if you request the (A) option.

Bits Ø-3 — Contain the binary month of the last modification date. If this field is a zero, DATE was not set when the file was established or since if it was updated.

DIR + 2

Contains the remaining date of modification fields.

Bits 3-7 — Contain the binary day of last modification.

Bits Ø-2 — Contain the binary year minus 8Ø. For example, 198Ø is coded as ØØØ, 1981 as ØØ1, 1982 as Ø1Ø, and so on.

DIR + 3

Contains the end-of-file offset byte. This byte and the ending record number (ERN) form a pointer to the byte position that follows the last byte written. This assumes that programmers, interfacing in machine language, properly maintain the next record number (NRN) offset pointer when the file is closed.

DIR + 4

Contains the logical record length (LRL) specified when the file was generated or when it was later changed with a CLONE parameter.

DIR + 5 through DIR + 12

Contain the name field of the filespec. The filename is left justified and padded with trailing blanks.

DIR + 13 through DIR + 15

Contain the extension field of the filespec. It is left justified and padded with trailing blanks.

DIR + 16 and DIR + 17

Contain the OWNER password hash code.

DIR + 18 and DIR + 19

Contain the USER password hash code. The protection level in DIR + Ø is associated with this password.

DIR + 20 and DIR + 21

Contain the ending record number (ERN), which is based on full sectors. If the ERN is zero, it indicates that no writing has taken place (or that the file was not closed properly). If the LRL is not 256, the ERN represents the sector where the EOF occurs. You should use ERN minus 1 to account for a value relative to sector 0 of the file.

DIR + 22 and DIR + 23

This is the first extent field. Its contents indicate which cylinder stores the first granule of the extent, which relative granule it is, and how many contiguous grans are in use in the extent.

DIR + 22 — Contains the cylinder value for the starting gran of that extent.

DIR + 23, Bits 5-7 — Contain the number of the granule in the cylinder indicated by DIR + 22 which is the first granule of the file for that extent. This value is relative to zero ("0" denotes the first gran, "1" denotes the second, and so on).

DIR + 23, Bits 0-4 — Contain the number of contiguous granules, relative to 0 ("0" denotes one gran, "1" denotes two, and so on). Since the field is five bits, it contains a maximum of X'1F' or 31, which represents 32 contiguous grans.

DIR + 24 and DIR + 25

Contain the fields for the second extent. The format is identical to that for Extent 1.

DIR + 26 and DIR + 27

Contain the fields for the third extent. The format is identical to that for Extent 1.

DIR + 28 and DIR + 29

Contain the fields for the fourth extent. The format is identical to that for Extent 1.

DIR + 30

This is a flag noting whether or not a link exists to an extended directory record. If no further directory records are linked, the byte contains X'FF.' A value of X'FE' in this byte establishes a link to an extended directory entry. (See "Extended Directory Records" below.)

DIR + 31

This is the link to the extended directory entry noted by the previous byte. The link code is the Directory Entry Code (DEC) of the extended directory record. The DEC is actually the position of the Hash Index Table byte mapped to the directory record. For more information, see the section "Hash Index Table."

Extended Directory Records

Extended directory records (FXDE) have the same format as primary directory records, except that only Bytes 0, 1, and 21-31 are utilized. Within Byte 0, only Bits 4 and 7 are significant. Byte 1 contains the DEC of the directory record of which this is an extension. An extended directory record may point to yet another directory record, so a file may contain an "unlimited" number of extents (limited only by the total number of directory records available).

Granule Allocation Table (GAT)

The Granule Allocation Table (GAT) contains information on the free and assigned space on the disk. The GAT also contains data about the formatting used on the disk.

A disk is divided into cylinders (tracks) and sectors. Each cylinder has a specified number of sectors. A group of sectors is allocated whenever additional space is needed. This group is called a granule. The number of sectors per granule depends on the total number of sectors available on a logical drive. The GAT provides for a maximum of eight granules per cylinder.

In the GAT bytes, each bit set to "1" indicates a corresponding granule in use (or locked out). Each bit reset to "Ø" indicates a granule free to be used. In a GAT byte, bit Ø corresponds to the first relative granule, bit 1 to the second relative granule, bit 2 the third, and so on. A 5¼" single density diskette is formatted at 1Ø sectors per cylinder, 5 sectors per granule, 2 granules per cylinder. Thus, that configuration uses only bits Ø and 1 of the GAT byte. The remainder of the GAT byte contains all 1's, denoting unavailable granules. Other formatting conventions are as follows:

	Sectors per Cylinder	Sectors per Granule	Granules per Cylinder	Maximum No. of Cylinders
5" SDEN	1Ø	5	2	8Ø
5" DDEN	18	6	3	8Ø
8" SDEN	16	8	2	77
8" DDEN	3Ø	1Ø	3	77
5-MEG HARD*	32	16	8	153

*Hard drive format depends on the drive size and type, as well as the user's division of the drive into logical drives. These values assume that one physical hard disk is treated as one logical drive.

The above table is valid for single-sided disks. TRSDOS supports double-sided operation if the hardware interfacing the physical drives to the CPU allows it. A two-headed drive functions as a single logical drive, with the second side as a cylinder-for-cylinder extension of the first side. A bit in the Drive Code Table (DCT + 4, Bit 5) indicates one-sided or two-sided drive configuration.

A Winchester-type hard disk can be divided by heads into multiple logical drives. Details are supplied with Radio Shack drives.

The Granule Allocation Table is the first relative sector of the directory cylinder. The following information describes the layout and contents of the GAT.

GAT + X'ØØ' through GAT + X'5F'

Contains the free/assigned table information. GAT + Ø corresponds to cylinder Ø, GAT + 1 corresponds to cylinder 1, GAT + 2 corresponds to cylinder 2, and so on. As noted above, bit Ø of each byte corresponds to the first granule on the cylinder, bit 1 to the second granule, and so on. A value of "1" indicates the granule is not available for use.

GAT + X'6Ø' through GAT + X'BF'

Contains the available/locked out table information. It corresponds cylinder for cylinder in the same way as the free/assigned table. It is used during mirror-image backups to determine if the destination diskette has the proper capacity to effect a backup of the source diskette. This table does not exist for hard disks; for this reason, mirror-image backups cannot be performed on hard disk.

GAT + X'CØ' through GAT + X'CA'

Used in hard drive configurations; extends the free/assigned table from X'ØØ' through X'CA'. Hard drive capacity up to 2Ø3 (Ø-2Ø2) logical or 4Ø6 physical cylinders is supported.

GAT + X'CB'

Contains the operating system version that was used in formatting the disk. For example, disks formatted under TRSDOS 6.1 have a value of X'61' contained in this byte. It is used to determine whether or not the disk contains all of the parameters needed for TRSDOS operation.

GAT + X'CC'

Contains the number of cylinders in excess of 35. It is used to minimize the time required to compute the highest numbered cylinder formatted on the disk. It is excess 35 to provide compatibility with alien systems not maintaining this byte. If you have a disk that was formatted on an alien system for other than 35 cylinders, this byte can be automatically configured by using the REPAIR utility. (See the section on the REPAIR utility in the *Disk System Owner's Manual*.)

GAT + X'CD'

Contains data about the formatting of the disk.

Bit 7 — If set to "1," the disk is a data disk. If "0," the disk is a system disk.

Bit 6 — If set to "1," indicates double-density formatting. If "0," indicates single-density formatting.

Bit 5 — If set to "1," indicates 2-sided disk. If "0," indicates 1-sided disk.

Bits 3-4 — Reserved.

Bits 0-2 — Contain the number of granules per cylinder minus 1.

GAT + X'CE' and GAT + X'CF'

Contain the 16-bit hash code of the disk master password. The code is stored in standard low-order, high-order format.

GAT + X'D0' through GAT + X'D7'

Contain the disk name. This is the name displayed during a FREE or DIR operation. The disk name is assigned during formatting or during an ATTRIB disk renaming operation. The name is left justified and padded with blanks.

GAT + X'D8' through GAT + X'DF'

Contain the date that the diskette was formatted or the date that it was used as the destination in a mirror image backup operation in the format mm/dd/yy.

GAT + X'E0' through GAT + X'FF'

Reserved for system use.

Hash Index Table (HIT)

The Hash Index Table is the key to addressing any file in the directory. It pinpoints the location of a file's directory with a minimum of disk accesses, keeping overhead low and providing rapid file access.

The system's procedure is to construct an 11-byte filename/extension field. The filename is left-justified and padded with blanks. The file extension is then inserted and padded with blanks; it occupies the three least significant bytes of the 11-byte field. This field is processed through a hashing algorithm which produces a single byte value in the range X'01' through X'FF'. (A hash value of X'00' indicates a spare HIT position.)

The system then stores the hash code in the Hash Index Table (HIT) at a position corresponding to the directory record that contains the file's directory. Since more than one 11-byte string can hash to identical codes, the opportunity for "collisions" exists. For this reason, the search algorithm scans the HIT for a matching code entry, reads the directory record corresponding to the matching HIT position, and compares the filename/extension stored in the directory with that provided in the file specification. If both match, the directory has been found. If the two fields do not match, the HIT entry was a collision and the algorithm continues its search from the next HIT entry.

The position of the HIT entry in the hash table is called the Directory Entry Code (DEC) of the file. All files have at least one DEC. Files that are extended beyond

four extents have a DEC for each extended directory entry and use more than one filename slot. To maximize the number of file slots available, you should keep your files below five extents where possible.

Each HIT entry is mapped to the directory sectors by the DEC's position in the HIT. Think of the HIT as eight rows of 32-byte fields. Each row is mapped to one of the directory records in a directory sector: The first HIT row is mapped to the first directory record, the second HIT row to the second directory record, and so on. Each column of the HIT field (Ø-31) is mapped to a directory sector. The first column is mapped to the first directory sector in the directory cylinder (not including the GAT and HIT). Therefore, the first column corresponds to sector 2, the second column to sector 3, and so on. The maximum number of HIT columns used depends on the disk formatting according to the formula: N = number of sectors per cylinder minus two, up to 32.

The following chart shows the correlation of the Hash Index Table to the directory records. Each byte value shown represents the position in the HIT. This position value is the DEC. The actual contents of each byte is either a X(ØØ) indicating a spare slot, or the 1-byte hash code of the file that occupies the corresponding directory record.

	Columns															
Row 1	ØØ	Ø1	Ø2	Ø3	Ø4	Ø5	Ø6	Ø7	Ø8	Ø9	ØA	ØB	ØC	ØD	ØE	ØF
	1Ø	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Row 2	2Ø	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
	3Ø	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
Row 3	4Ø	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
	5Ø	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
Row 4	6Ø	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
	7Ø	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
Row 5	8Ø	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
	9Ø	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
Row 6	AØ	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
	BØ	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
Row 7	CØ	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
	DØ	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
Row 8	EØ	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
	FØ	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

A 5¼″ single density disk has 1Ø sectors per cylinder, two of which are reserved for the GAT and HIT. Since only eight directory sectors are possible, only the first eight positions of each HIT row are used. Other formats use more columns of the HIT, depending on the number of sectors per cylinder in the formatting scheme.

The eight directory records for sector 2 of the directory cylinder correspond to assignments in HIT positions ØØ, 2Ø, 4Ø, 6Ø, 8Ø, AØ, CØ, and EØ. On system disks, the following positions are reserved for system overlays. On data disks, these positions (except for ØØ and Ø1) are available to the user.

ØØ — BOOT/SYS	2Ø — SYS6/SYS
Ø1 — DIR/SYS	21 — SYS7/SYS
Ø2 — SYSØ/SYS	22 — SYS8/SYS
Ø3 — SYS1/SYS	23 — SYS9/SYS
Ø4 — SYS2/SYS	24 — SYS1Ø/SYS
Ø5 — SYS3/SYS	25 — SYS11/SYS
Ø6 — SYS4/SYS	26 — SYS12/SYS
Ø7 — SYS5/SYS	27 — SYS13/SYS

These entry positions correspond to the first two rows of each directory sector for the first eight directory sectors. Since the operating system accesses these overlays by position in the HIT rather than by filename, these positions are reserved on system disks.

The design of the Hash Index Table limits the number of files on any one drive to a maximum of 256.

Locating a Directory Record

Because of the coding scheme used on the entries in the HIT table, you can locate a directory record with only a few instructions. The instructions are:

```
AND   1FH
ADD   A,2
```

(calculates the sector)

and

```
AND   0E0H
```

(calculates the offset in that sector)

For example, if you have a Directory Entry Code (DEC) of X'84', the following occurs when these instructions are performed:

	Value of accumulator
	A = X'84'
AND 1FH	
	A = X'04'
ADD A,2	
	A = X'06' The record is in the seventh sector of the directory cylinder (0-6)

Using the Directory Entry Code (DEC) again, you can find the offset into the sector that was found using the above instructions by executing one instruction:

	Value of accumulator
	A = X'84'
AND 0E0H	
	A = X'80' The directory record is X'80' (128) bytes from the beginning of the sector

If the record containing the sector is loaded on a 256-byte boundary (LSB of the address is X'00') and HL points to the starting address of the sector, then you can use the above value to calculate the actual address of the directory record by executing the instruction:

```
LD    L,A
```

When executed after the calculation of the offset, this causes HL to point to the record. For example:

A = X'80'

```
LD    HL,4200H    ;Where sector is loaded
LD    L,A         ;Replace LSB with offset
```

HL now contains 4280H, which is the address of the directory record you wanted.

If you cannot place the sector on a 256-byte boundary, then you can use the following instructions:

A = X'80'

```
LD    HL,4256H    ;Where sector is loaded
LD    E,A         ;Put offset in E (LSB)

LD    D,0         ;Put a zero in D (MSB)
ADD   HL,DE       ;Add two values together
```

HL now contains 42D6H, which is the address of the directory record.

Note that the first DEC found with a matching hash code may be the file's extended directory entry (FXDE). Therefore, if you are going to write system code to deal with this directory scheme, you must properly deal with the FPDE/ FXDE entries. See Directory Records for more information.

6/File Control

File Control Block (FCB)

The File Control Block (FCB) is a 32-byte memory area. Before the file is opened, this space holds the file's filespec. After an @OPEN or @INIT supervisor call is performed, the system uses this area to interface with the file, and replaces the filespec with other information. When the file is closed, the filespec (without any specified password) is returned to the FCB.

While a file is open, the contents of the FCB are dynamic. As records are written to or read from the disk file, specific fields in the FCB are modified. Avoid changing the contents of the FCB during the time a file is open, unless you are sure that the change will not affect the integrity of the file.

During most system access of the FCB, the IX index register is used to reference each field of data. Register pair DE is used mainly for the initial reference to the FCB address. The information contained in each field of the FCB is as follows:

FCB+Ø

Contains the TYPE code of the control block.

Bit 7 — If set to "1," indicates that the file is in an open condition; if "Ø," the file is assumed closed. This bit can be tested to determine the "open" or "closed" status of an FCB.

Bit 6 — Is set to "1" if the file was opened with UPDATE access or higher.

Bit 5 — Indicates a Partition Data Set (PDS) type file.

Bits 4-3 — Reserved for future use.

Bit 2 — Is set to "1" if the system performed any WRITE operation on this file. It is used to update the MOD flag in the directory record when the file is closed.

Bits 1-Ø — Reserved for future use.

FCB+1

Contains status flag bits used in read/write operations by the system.

Bit 7 — If set to "1," indicates that I/O operations will be either full sector operations or byte operations of logical record length (LRL) less than 256. If "Ø," only sector operations will be performed. If you are going to use only full-sector I/O, you can reduce system overhead by specifying the LRL at open time as Ø (indicating 256). An LRL of other than 256 sets bit 7 to "1" on open.

Bit 6 — If set to "1," indicates that the end of file (EOF) is to be set to ending record number (ERN) only if next record number (NRN) exceeds the current value of EOF. This is the case if random access is to be used. During random access, the EOF is not disturbed unless you extend the file beyond the last record slot. Any time the position routine (@POSN) is called, bit 6 is automatically set. If bit 6 is "Ø," then EOF will be updated on every WRITE operation.

Bit 5 — If "Ø," then the disk I/O buffer contains the current sector denoted by NRN. If set to "1," then the buffer does not contain the current sector. During byte I/O, bit 5 is set when the last byte of the sector is read. A sector read resets the bit, showing the buffer to be current.

Bit 4 — If set to "1," indicates that the buffer contents have been changed since the buffer was read from the file. It is used by the system to determine whether the buffer must be written back to the file before reading another record. If "Ø," then the buffer contents were not changed.

Bit 3 — Used to specify that the directory record is to be updated each time the NRN exceeds the EOF. (The normal operation is to update the directory only when an FCB is closed.) Some unattended operations may use this extra measure of file protection. It is specified by adding an exclamation mark ("!") to the end of a filespec when the filespec is requested at open time.

Bits 2-Ø — Contain the user (access) protection level as retrieved from the directory of the file. The 3-bit binary value is one of the following:

Ø = FULL	2 = RENAME	4 = UPDATE	6 = EXECUTE
1 = REMOVE	3 = WRITE	5 = READ	7 = NO ACCESS

FCB + 2

Used by Partition Data Set (PDS) files.

FCB + 3 and FCB + 4

Contain the buffer address in low-order, high-order format. This is the buffer address specified in register pair HL when the @INIT or @OPEN SVC is performed.

FCB + 5

Contains the relative byte offset within the current buffer for the next I/O operation. If this byte has a zero value, then FCB + 1, Bit 5 must be examined to see if the first byte in the current buffer is the target position or if it is the first byte of the next record. If you are performing sector I/O of byte data (that is, maintaining your own buffering), then it is important to maintain this byte when you close the file if the true end of file is not at a sector boundary.

FCB + 6

Bits 3-7 — Reserved for system use.

Bits Ø-2 — Contain the logical drive number in binary of the drive containing the file. Do not modify this byte; altering this value may damage other files. This byte and FCB + 7 are the only links to the file's directory information.

FCB + 7

Contains the directory entry code (DEC) for the file. This code is the offset in the Hash Index Table where the hash code for the file appears. Do not modify this byte; altering this value may damage other files. This byte and FCB + 6 are the only links to the directory information for the file.

FCB + 8

Contains the end-of-file byte offset. This byte is similar to FCB + 5 except that it pertains to the end of file rather than to the next record number.

FCB + 9

Contains the logical record length that was in effect when the file was opened. This may not be the same LRL that exists in the directory. The directory LRL is generated at the file creation and never changes unless the file is overwritten.

FCB + 10 and FCB + 11

Contain the next record number (NRN), which is a pointer for the next I/O operation. When a file is opened, NRN is zero, indicating a pointer to the beginning. Each sequential sector I/O advances NRN by one.

FCB + 12 and FCB + 13

Contain the ending record number (ERN) of the file. This is a pointer to the sector that contains the end-of-file indicator. In a null file (one with no records), ERN equals 0. If one sector has been written, ERN equals 1.

FCB + 14 and FCB + 15

Contain the same information as the first extent of the directory. This represents the starting cylinder of the file (FCB + 14) and the starting relative granule within the starting cylinder (FCB + 15). FCB + 15 also contains the number of contiguous granules allocated in the extent. These bytes are used as a pointer to the beginning of the file referenced by the FCB.

FCB + 16 through FCB + 19

This 4-byte entry contains granule allocation information for an extent of the file. Relative bytes 0 and 1 contain the total number of granules allocated to the file up to but not including the extent referenced by this field. Relative byte 2 contains the starting cylinder of this extent. Relative byte 3 contains the starting relative granule for the extent and the number of contiguous granules.

FCB + 20 through FCB + 23

Contain information similar to the above but for a second extent of the file.

FCB + 24 through FCB + 27

Contain information similar to the above but for a third extent of the file.

FCB + 28 through FCB + 31

Contain information similar to the above but for a fourth extent of the file.

The file control block contains information on only four extents at one time. If the file has more than four extents, additional directory accessing is done to shift the 4-byte entries in order to make space for the new extent information.

Although the system can handle a file of any number of extents, you should keep the number of extents small. The most efficient file is one with a single extent. The number of extents can be reduced by copying the file to a disk that contains a large amount of free space.

7/TRSDOS Version 6 Programming Guidelines

Converting to TRSDOS Version 6

This section provides suggestions on writing programs effectively with TRSDOS Version 6, and on converting programs created with TRSDOS 1.3 and LDOS 5.1 operating systems for use with TRSDOS Version 6. This information is by no means complete, but presents some important concepts to keep in mind when using TRSDOS Version 6.

When programming in assembly language, you can use TRSDOS Version 6 routines for commonly used operations. These are accessed through the supervisor calls (SVCs) instead of absolute call addresses. Nothing in the system can be accessed via any absolute address reference (except Z-80 RST and NMI jump vectors).

IMPORTANT NOTE: TRSDOS provides all functions and storage through supervisor calls. No address or entry point below 3000H is documented or supported by Radio Shack.

The keyboard is not accessible via "peeking," and the video RAM cannot be "poked." The keyboard and video are accessible only through the appropriate SVCs.

Another distinction is that TRSDOS Version 6 handling of logical byte I/O devices (keyboard, video, printer, communications line) completely supports error status feedback. A FLAG convention is uniform throughout these device drivers as well as physical byte I/O associated with files. The device handling in TRSDOS Version 6 is completely independent. That means that byte I/O, both logical and physical, can be routed, filtered, and linked. Therefore, it is important to test status return codes in all applications using byte I/O regardless of the device that the application expects to be used, since re-direction to some other device is possible at the TRSDOS level. Appropriate action must be taken when errors are detected.

Modules loaded into memory and protected by lowering HIGH$ must include the standard header, as described earlier under "Memory Header." The @GTMOD supervisor call requires that this header be present in every resident module for proper operation.

The file password protection terms of UPDATE and ACCESS have been changed in TRSDOS Version 6 to OWNER and USER, respectively. The additional file protection level of UPDATE has been added. A file with UPDATE protection level can be read or written to, but its end of file cannot be extended. This protection can be useful in a random access fixed-size file or in a file where shared access is to take place.

Files opened with UPDATE or greater access are indicated as open in their directory. Attempting to open the file again forces a change to READ access protection and a "File already open" error code. It is therefore important for applications to CLOSE files that are opened.

For the convenience of applications that access files only for reading, you can inhibit the "file open bit." If you set bit 0 of the system flag SFLAG$ (see the @FLAGS supervisor call), the file open bit is not set in the file's directory. Once set, the next @OPEN or @INIT SVC automatically resets bit 0 of SFLAG$. Note that you cannot use this procedure for files being written to, since it inhibits the CLOSE process.

Some application programs need access to certain system parameters and variables. A number of flags, variables, and port images can be accessed relative to a flag pointer obtained via the @FLAGS supervisor call. These parameters are only accessible relative to this pointer, as the pointer's location may change. (See the explanation of the @FLAGS SVC.)

All applications must honor the contents of HIGH$. This pointer contains the highest RAM address usable by any program. You can retrieve and change HIGH$ by using the @HIGH$ SVC.

TRSDOS Version 6 library commands and utilities supply a return code (RC) at completion. The RC is returned in register pair HL. The value returned is either zero (indicating no error), a number from one through 62 (indicating an error as noted in Appendix A, TRSDOS Error Messages), or X'FFFF' (indicating an extended error which is currently not assigned an error number). TRSDOS Version 6 Job Control Language (JCL) aborts on any program terminating with a non-zero RC value. Applications should therefore properly set the return code register pair HL before exiting.

TRSDOS Version 6 library commands are also invokable via the @CMNDR SVC which executes the command. Library commands properly maintain the Stack Pointer (SP) and exit via a RET instruction. In this manner, control is returned to the invoking program with the RC present for testing. For commands invoked with the @CMNDI SVC or prompted for via the @EXIT SVC, the SP is restored to the system stack. The top of the stack will contain an address suitable for simulating an @EXIT SVC; thus, if your application program properly maintains the integrity of the stack pointer, it can exit after setting the RC via a RET instruction instead of an @EXIT SVC.

TRSDOS Version 6 diskette and file structure is identical to that used in LDOS 5.1. This includes formatting, directory structure, and data address mark conventions. TRSDOS Version 6 system diskettes, however, use the entire BOOT track (track Ø). This compatibility means that data files may be used interchangeably between LDOS 5.1 equipped machines and TRSDOS Version 6 equipped machines; the diskettes themselves are readable and writable across both operating systems.

The methods of internal handling of device linking and filtering have been changed from LDOS 5.1. (It is beyond the scope of this manual to explain the internal functioning of TRSDOS Version 6.) Device filters must adhere to a strict protocol of linkage in order to function properly. See the section on "Device Driver and Filter Templates" for information on device driver and filter protocol.

Stack Handling Restrictions*

Interrupt tasks and filters that deal with the keyboard or video must not place the stack pointer above X'F3FF'. This is because any operation that requires the keyboard or video RAM switches in the 3K bank at X'F4ØØ' and suppresses the stack until it is switched out again. If the system accesses the stack at any time during this period, the integrity of the stack is destroyed.

*In TRSDOS 6.0.0, the stack cannot be placed above X'F3FF' for any reason.

Programming With Restart Vectors

The Restart instruction (RST) provides the assembly language programmer with the ability to call a subroutine with a one-byte call. If a routine is called many times by a program, the amount of space that is saved by using the RST instruction (instead of a three-byte CALL) can be significant.

In TRSDOS a RST instruction is also used to interface to the operating system. The system uses RST 28H for supervisor calls. RSTS 00H, 30H, and 38H are for the system's internal use.

RSTs 08H, 10H, 18H, and 20H are available for your use. Caution: Some programs, such as BASIC, may use some of these RSTs.

Each RST instruction calls the address given in the operand field of the instruction. For example, RST 18H causes the system to push the current program counter address onto the stack and then set the program counter to address 0018H. RST 20H causes a jump to location 0020H, and so on.

Each RST has three bytes reserved for the subroutine to use. If the subroutine will not fit in three bytes, then you should code a jump instruction (JP) to where the subroutine is located. At the end of the subroutine, code a return instruction (RET). Control is then transferred to the instruction that follows the RST.

For example, suppose you want to use RST 18H to call a subroutine named "ROUTINE." The following routine loads the restart vector with a jump instruction and saves the old contents of the restart vector for later use.

```
SETRST:  LD      IX,0018H   ;Restart area address
         LD      IY,RDATA   ;Data area address
         LD      B,3        ;Number of bytes to move
LOOP:    LD      A,(IX)     ;Read a byte from
                            ;restart area
         LD      C,(IY)     ;Read a byte from data
                            ;area
         LD      (IX),C     ;Store this byte in
                            ;restart area
         LD      (IY),A     ;Store this byte in data
                            ;area
         INC     IX         ;Increment restart area
                            ;pointer
         INC     IY         ;Increment data area
                            ;pointer
         DJNZ    LOOP       ;Loop till 3 bytes moved
         RET                ;Return when done
RDATA:   DEFB    0C3H       ;Jump instruction (JP)
         DEFW    ROUTINE    ;Operand (name of
                            ;subroutine)
```

Before exiting the program, calling the above routine again puts the original contents of the restart vector back in place.

KFLAG$ (BREAK), (PAUSE), and (ENTER) Interfacing

KFLAG$ contains three bits associated with the keyboard functions of BREAK, PAUSE ((SHIFT) (@)), and ENTER. A task processor interrupt routine (called the KFLAG$ scanner) examines the physical keyboard and sets the appropriate KFLAG$ bit if any of the conditions are observed. Similarly, the RS-232C driver routine also sets the KFLAG$ bits if it detects the matching conditions being received.

Many applications need to detect a PAUSE or BREAK while they are running. BASIC checks for these conditions after each logical statement is executed (that is, at the end of a line or at a ":"). That is how, in BASIC, you can stop a program with the (BREAK) key or pause a listing.

One method of detecting the condition in previous TRSDOS operating systems was to issue the @KBD supervisor call to check for BREAK or PAUSE ((SHIFT)(@)), ignoring all other keys. Unfortunately, this caused keyboard type-ahead to be ineffective; the @KBD SVC flushed out the type-ahead buffer if any other keystrokes were stacked up.

Another method was to scan the keyboard, physically examining the keyboard matrix. An undesirable side effect of this method was that type-ahead stored up the keyboard depression for some future unexpected input request. Examining the keyboard directly also inhibits remote terminals from passing the BREAK or PAUSE condition.

In TRSDOS Version 6, the KFLAG$ scanner examines the keyboard for the BREAK, PAUSE, and ENTER functions. If any of these conditions are detected, appropriate bits in the KFLAG$ are set (bits 0, 1, and 2 respectively).

Note that the KFLAG$ scanner only sets the bits. It does not reset them because the "events" would occur too fast for your program to detect. Think of the KFLAG$ bits as a latch. Once a condition is detected (latched), it remains latched until something examines the latch and resets it—a function to be performed by your KFLAG$ detection routine.

For illustration, the following example routine uses the BREAK and PAUSE conditions:

```
KFLAG$   EQU   10
@FLAGS   EQU   101
@KBD     EQU   8
@KEY     EQU   1
@PAUSE   EQU   16
CKPAWS   LD    A,@FLAGS        ;Get Flags pointer
         RST   28H             ;into register IY
         LD    A,(IY+KFLAG$)   ;Get the KFLAG$
         RRCA                  ;Bit 0 to carry
         JP    C,GOTBRK        ;Go on BREAK
         RRCA                  ;Bit 1 to carry
         RET   NC              ;Return if no pause
         CALL  RESKFL          ;Reset the flag
         PUSH  DE
FLUSH    LD    A,@KBD          ;Flush type-ahead
         RST   28H             ;buffer while
         JR    Z,FLUSH         ;ignoring errors
         POP   DE
PROMPT   PUSH  DE
         LD    A,@KEY          ;Wait on key entry
         RST   28H
         POP   DE
         CP    80H             ;Abort on (BREAK)
         JP    Z,GOTBRK
         CP    60H             ;Ignore PAUSE;
         JR    Z,PROMPT        ;else . . .
RESKFL   PUSH  HL              ;reset KFLAG$
         PUSH  AF
         LD    A,@FLAGS        ;Get flags pointer
         RST   28H             ;into register IY
RESKFL1  LD    A,(IY+KFLAG$)   ;Get the flag
         AND   0F8H            ;Strip ENTER,
         LD    (IY+KFLAG$),A   ;PAUSE, BREAK
         PUSH  BC
         LD    B,16
```

```
        LD      A,@PAUSE        ;Pause a while
        RST     28H
        POP     BC
        LD      A,(IY+KFLAG$)   ;Check if finger is
        AND     3               ;still on key
        JR      NZ,RESKFL1      ;Reset it again
        POP     AF              ;Restore registers
        POP     HL              ;and exit
        RET
```

The best way to explain this KFLAG$ detection routine is to take it apart and discuss each subroutine. The first piece reads the KFLAG$ contents:

```
KFLAG$  EQU     10
CKPAWS  LD      A,@FLAGS        ;Get Flags pointer
        RST     28H             ;into register IY
        LD      A,(IY+KFLAG$)   ;Get the KFLAG$
        RRCA                    ;Bit 0 to carry
        JP      C,GOTBRK        ;Go on BREAK
        RRCA                    ;Bit 1 to carry
        RET     NC              ;Return if no pause
```

The @FLAGS SVC obtains the flags pointer from TRSDOS. Note that if your application uses the IY index register, you should save and restore it within the CKPAWS routine. (Alternatively, you could use @FLAGS to calculate the location of KFLAG$, use register HL instead of IY, and place the address into the LD instructions of CKPAWS at the beginning of your application.)

The first rotate instruction places the BREAK bit into the carry flag. Thus, if a BREAK condition is in effect, the subroutine branches to "GOTBRK," which is your BREAK handling routine.

If there is no BREAK condition, the second rotate places what was originally in the PAUSE bit into the carry flag. If no PAUSE condition is in effect, the routine returns to the caller.

This sequence of code gives a higher priority to BREAK (that is, if both BREAK and PAUSE conditions are pending, the BREAK condition has precedence). Note that the GOTBRK routine needs to clear the KFLAG$ bits after it services the BREAK condition. This is easily done via a call to RESKFL.

The next part of the routine is executed on a PAUSE condition:

```
        CALL    RESKFL          ;Reset the flag
        PUSH    DE
FLUSH   LD      A,@KBD          ;Flush type-ahead
        RST     28H             ;buffer while
        JR      Z,FLUSH         ;ignoring errors
        POP     DE
```

First the KFLAG$ bits are reset via the call to RESKFL. Next, the routine takes care of the possibility that type-ahead is active. If it is, the PAUSE key was probably detected by the type-ahead routine and so is stacked in the type-ahead buffer also. To flush out (remove all stored characters from) the type-ahead buffer, @KBD is called until no characters remain (an NZ is returned).

Now that a PAUSEd state exists and the type-ahead buffer is cleared, the routine waits for a key input:

```
PROMPT  PUSH    DE
        LD      A,@KEY          ;Wait on key entry
        RST     28H
        POP     DE
        CP      80H             ;Abort on (BREAK)
        JP      Z,GOTBRK
        CP      60H             ;Ignore PAUSE;
        JR      Z,PROMPT        ;else . . .
```

The PROMPT routine accepts a BREAK and branches to your BREAK handling routine. It ignores repeated PAUSE (the 60H). Any other character causes it to fall through to the following routine which clears the KFLAG$:

```
RESKFL   PUSH  HL              ;reset KFLAG$
         PUSH  AF
         LD    A,@FLAGS        ;Get flags pointer
         RST   28H             ;into register IY
RESKFL1  LD    A,(IY+KFLAG$)   ;Get the flag
         AND   0F8H            ;Strip ENTER,
         LD    (IY+KFLAG$),A   ;PAUSE, BREAK
         PUSH  BC
         LD    B,16
         LD    A,@PAUSE        ;Pause a while
         RST   28H
         POP   BC
         LD    A,(IY+KFLAG$)   ;Check if finger is
         AND   3               ;still on key
         JR    NZ,RESKFL1      ;Reset it again
         POP   AF              ;Restore registers
         POP   HL              ;and exit
         RET
```

The RESKFL subroutine should be called when you first enter your application. This is necessary to clear the flag bits that were probably in a "set" condition. This "primes" the detection. The routine should also be called once a BREAK, PAUSE, or ENTER condition is detected and handled. (You need to deal with the flag bits for only the conditions you are using.)

Interfacing to @ICNFG

With the TRSDOS library command SYSGEN, many users may wish to SYSGEN the RS-232C driver. Before doing that, the RS-232C hardware (UART, Baud Rate Generator, etc.) must be initialized. Simply using the SYSGEN command with the RS-232C driver resident is not enough; some initialization routine is necessary. The @ICNFG (Initialization CoNFiGuration) vector is included in TRSDOS to provide a way to invoke a routine to initialize the RS-232C driver when the system is booted. It also provides a way to initialize the hard disk controller at power-up (required by the Radio Shack hard disk system).

The final stages of the booting process loads the configuration file CONFIG/SYS if it exists. After the configuration file is loaded, an initialization subroutine CALLs the @ICNFG vector. Thus, any initialization routine that is part of a memory configuration can be invoked by chaining into @ICNFG.

If you need to configure your own routine that requires initialization at power-up, you can chain into @ICNFG. The following procedure illustrates this link. The first thing to do is to move the contents of the @ICNFG vector into your initialization routine:

```
         LD    A,@FLAGS        ;Get flags pointer
         RST   28H             ;into register IY
         LD    A,(IY+28)       ;Get opcode
         LD    (LINK),A
         LD    L,(IY+29)       ;Get address LOW
         LD    H,(IY+30)       ;Get address HIGH
         LD    (LINK+1),HL
```

This subroutine does this by transferring the 3-byte vector to your routine. You then need to relocate your routine to its execution memory address. Once this is done, transfer the relocated initialization entry point to the @ICNFG vector as a jump instruction:

```
        LD      HL,INIT         ;Get (relocated)
        LD      (IY+29),L       ;init address
        LD      (IY+30),H
        LD      A,0C3H          ;Set JP instruction
        LD      (IY+28),A
```

If you need to invoke the initialization routine at this point, then you can use:

```
        CALL    ROUTINE         ;Invoke your routine
```

Your initialization routine would be unique to the function it was to perform, but an overall design would look like this:

```
INIT    CALL    ROUTINE         ;Start of init
LINK    DEFS    3               ;Continue on
ROUTINE ·
        your initialization routine

RET
```

After linking in your routine, perform the SYSGEN. If you have followed these procedures, your routine will be invoked every time you start up TRSDOS.

Interfacing to @KITSK

Background tasks can be invoked in one of two ways. For tasks that do not require disk I/O, you can use the RTC (Real Time Clock) interrupt and one of the 12 task slots (or other external interrupt). For tasks that require disk I/O, you can use the keyboard task process.

At the beginning of the TRSDOS keyboard driver is a call to @KITSK. This means that any time that @KBD is called, the @KITSK vector is also called. (The type-ahead task, however, bypasses this entry so that @KITSK is not called from the type-ahead routine.) Therefore, if you want to interface a background routine that does disk I/O, you must chain into @KITSK.

The interfacing procedure to @KITSK is identical to that shown in the section "Interfacing to @ICNFG," except that IY+31 through IY+33 is used to reference the @KITSK vector. You may want to start your background routine with:

```
START   CALL    ROUTINE         ;Invoke task
LINK    DEFS    3               ;For @KITSK hook
ROUTINE EQU     $               ;Start of the task
```

Be aware of one major pitfall. The @KBD routine is invoked from @CMNDI and @CMNDR (which is in SYS1/SYS). This invocation is from the @KEYIN call, which fetches the next command line after issuing the "TRSDOS Ready" message. If your background task executes and opens or closes a file (or does anything to cause the execution of a system overlay other than SYS1), then SYS1 is overwritten by SYS2 or SYS3. When your routine finishes, the @KEYIN handler tries to return to what called it—SYS1, which is no longer resident. Therefore, any task chained to @KITSK which causes a resident SYS1 to be overwritten must reload SYS1 before returning.

You can use the following code to reload SYS1 if SYS1 was resident prior to your task's execution:

```
ROUTINE LD      A,@FLAGS        ;Get flags pointer
        RST     28H             ;into register IY
        LD      A,(IY-1)        ;Get resident over-
        AND     8FH             ;lay and remove
        LD      (OLDSYS+1),A    ;the entry code

        rest of your task
```

```
EXIT     EQU   $
OLDSYS   LD    A,0          ;Get old overlay #
         CP    83H          ;Was it SYS1?
         RET   NZ           ;Return if not; else
         RST   28H          ;Get SYS1 per reg. A
                            ;(no RET needed)
```

Interfacing to the Task Processor

This section explains how to integrate interrupt tasks into your applications.

One of the hardware interrupts in the TRS-80 is the real time clock (RTC). The RTC is synchronized to the AC line frequency and pulses at 60 pulses per second, or once every 16.67 milliseconds. (Computers operating with 50 Hz AC use a 50 pulses per second RTC interrupt. In this case, all time relationships discussed in this section should be adjusted to the 50 Hz base.)

A software task processor manages the RTC interrupt in performing background tasks necessary to specific functions of TRSDOS (such as the time clock, blinking cursor, and so on). The task processor allows up to 12 individual tasks to be performed on a "time-sharing" basis.

These tasks are assigned to "task slots" numbered from 0 to 11. Slots 0-7 are considered "low priority" tasks (executing every 266.67 milliseconds). Slots 8-10 are medium priority tasks (executing every 33.33 milliseconds). Slot 11 is a high priority task (executing every 16.66 milliseconds SYSTEM (FAST) or 33.33 milliseconds SYSTEM (SLOW)). Task slots 3, 7, 9, and 10 are reserved by the system for the ALIVE, TRACE, SPOOL, and TYPE-AHEAD functions, respectively.

TRSDOS maintains a Task Control Block Vector Table (TCBVT) which contains 12 vectors, one for each of the 12 task slots. TRSDOS contains five supervisor calls that manage the task vectors. The five SVCs and their functions are:

@CKTSK	Checks to see whether a task slot is unused or active
@ADTSK	Adds a task to the TCBVT
@RMTSK	Removes a task from the TCBVT
@KLTSK	Removes the currently executing task
@RPTSK	Replaces the TCB address for the current task

The TRSDOS Task Control Block Vector Table contains vector pointers. Each TCBVT vector points to an address in memory, which in turn contains the address of the task. Thus, the tasks themselves are indirectly addressed.

When you are programming a task to be called by the task processor, the entry point of the routine needs to be stored in memory. If you make this storage location the beginning of a Task Control Block (TCB), the reason for indirect vectoring of interrupt tasks will become more clear. Consider an example TCB:

```
MYTCB    DEFW  MYTASK
COUNTER  DEFB  15
TEMPY    DEFS  1
MYTASK   RET
```

This is a useless task, since the only thing it does is return from the interrupt. However, note that a TCB location has been defined as "MYTCB" and that this location contains the address of the task. A few more data bytes immediately following the task address storage have also been defined.

Upon entry to a service routine, index register IX contains the address of the TCB. You can therefore address any TCB data using index instructions. For example, you could use the instruction "DEC (IX+2)" to decrement the value contained in COUNTER in the above routine.

Here is the routine expanded slightly:

```
MYTCB    DEFW  MYTASK
COUNTER  DEFB  15
TEMPY    DEFB  0
MYTASK   DEC   (IX+2)
         RET   NZ
         LD    (IX+2),15
         RET
```

This version makes use of the counter. Each time the task executes, the counter is decremented. When the count reaches zero, the counter is restored to its original value.

In order to be executed, all tasks must be added to the TCBVT. The @ADTSK supervisor call does this. For the above routine, assume the task slot chosen is low-priority slot 2. You can ascertain that slot 2 is available for use by using the @CKTSK SVC as follows:

```
         LD    C,2            ;Reference slot 2
         LD    A,28           ;Set for @CKTSK SVC
         RST   28H            ;An "NZ" indication
         JP    NZ,INUSE       ;says that the slot is
                              ;being used.
```

Once you determine that the slot is available (that is, not being used by some other task), you can add your task routine. The following code adds this task to the TCBVT:

```
         LD    DE,MYTCB       ;Point to the TCB
         LD    C,2            ;Reference slot 2
         LD    A,29           ;Set for @ADTSK SVC
         RST   28H            ;Issue the SVC
```

The above program lines point register DE to the TCB, load the task slot number into register C, and then issue the @ADTSK supervisor call. If you want this task to run regardless of what is in memory, you can place it in high memory (of bank 0) and protect it by moving HIGH$ below it via the @HIGH$ supervisor call.

Once a task has been activated, it is sometimes necessary to deactivate it. You can do this in two ways. The most common way is to use the @RMTSK supervisor call:

```
         LD    C,2            ;Designate the task
                              ;slot
         LD    A,30           ;Set for @RMTSK SVC
         RST   28H            ;Issue the SVC
```

You identify the task slot to remove by placing a value in register C, and then you issue the supervisor call.

You can use another method if you want to remove the task while it is being executed. Examine the routine modified as follows:

```
MYTCB    DEFW  MYTASK
COUNTER  DEFB  10
TEMPY    DEFB  0
MYTASK   DEC   (IX+2)
         RET   NZ
         LD    A,32           ;Set for @KLTSK SVC
         RST   28H            ;Issue the SVC
```

The @KLTSK supervisor call removes the currently executing task from the TCBVT. The system does not return to your routine, but continues as if you had executed a RET instruction. For this reason, the @KLTSK SVC should be the last instruction you want executed. In this example, MYTASK decrements the counter by one on each entry to the task. When the counter reaches zero, the task is removed from slot 2.

The last task processor supervisor call is @RPTSK. The @RPTSK function updates the TCB storage vector (the vector address in your Task Control Block) to be the address immediately following the @RPTSK SVC instruction. As with @KLTSK, the system does not return to your service routine after the SVC is made, but continues on with the task processor. The following example illustrates how @RPTSK can be used in a program:

```
          ORG    9000H
@ADTSK    EQU    29
@RPTSK    EQU    31
@RMTSK    EQU    30
@EXIT     EQU    22
@VDCTL    EQU    15
BEGIN     LD     DE,TCB        ;Point to TCB
          LD     C,0           ;and add the task
          LD     A,@ADTSK      ;to slot 0
          RST    28H
          LD     A,@EXIT       ;Exit to TRSDOS
          RST    28H
TCB       DEFW   TASK
COUNTER   DEFB   15
TASKA     LD     A,@RPTSK      ;Replace current
          RST    28H           ;task with TASKA
TASK      LD     BC,027CH      ;Put a character
          LD     HL,004FH      ;at Row 0, Col. 79
          LD     A,@VDCTL
          RST    28H
          DEC    (IX+2)        ;Decrement the counter
          RET    NZ            ;and return if not
          LD     (IX+2),15     ;expired; else reset
          LD     A,@RPTSK      ;Replace the previous
          RST    28H           ;task with TASKB
TASKB     LD     BC,022DH      ;Put a character
          LD     HL,004FH      ;at Row 0, Col. 79
          LD     A,@VDCTL
          RST    28H
          DEC    (IX+2)
          RET    NZ
          LD     (IX+2),15
          JR     TASKA
          END    BEGIN
```

This task routine contains no method of relocating it to protected RAM. The statements starting at the label BEGIN add the task to TCBVT slot 0 and return to TRSDOS Ready. The task contains a four-second down counter and a routine to put a character in video RAM (80th character of Row 0). At four-second intervals, the character toggles between '|' and '–'. This is done by using the @RPTSK SVC to toggle the execution of two separate routines which perform the character display.

TRSDOS uses bank-switched memory. In order to properly control and manage this additional memory, certain restrictions are placed on tasks. All tasks must be placed either in low memory (addresses X'0000' through X'7FFF') or in bank zero of high memory (addresses X'8000' through X'FFFF'). The task processor always enables bank zero when performing background tasks. The assembly language programmer must ensure that tasks are placed in the correct memory area.

Interfacing RAM Banks 1 and 2

The proper use of the RAM bank transfer techniques described here requires a high degree of skill in assembly language programming. This section on bank switching is intended for the professional.

The TRS-80 Model 4 can optionally support a second set of 64K RAM, bringing the total RAM to 128K. TRSDOS designates this extra 64K RAM as two banks of 32K RAM each, which are banks 1 and 2 of bank-switched RAM. The upper 32K of standard RAM is designated bank 0. At any one time, only one of the banks is resident. The resident bank is always addressed at X'8000' through X'FFFF.' When a bank transfer is performed, the specified bank becomes addressable and the previous bank is no longer available. Since memory refresh is performed on all banks at all times, nothing in the previously resident bank is altered during whatever time it is not addressable (that is, not resident).

You can access this additional RAM by means of the @BANK supervisor call (SVC 102). When you power up your computer or press reset, TRSDOS looks to see which banks of RAM are installed in your machine. TRSDOS maintains a bit map in one byte of storage, with each bit representing one of the banks of RAM. This byte is called "Bank Available RAM" (BAR), and its information is set when you boot TRSDOS. Bit 0 corresponds to bank 0, bit 1 corresponds to bank 1, and so on up to bit 7. From a hardware standpoint, the Model 4 has a maximum of three banks. You have either bank 0 only (a 64K machine), or banks 0-2 (a 128K machine).

Another bit map is used to indicate whether a bank is reserved or available for use. This byte is called the "Bank Used RAM" (BUR). Again, bit 0 corresponds to bank 0, bit 1 to bank 1, and so on. TRSDOS design supports the use of banks 1 and 2 primarily for data storage (for example, a spool buffer, Memdisk, etc.). The management of any memory space within a particular bank of RAM (excluding bank 0) is the responsibility of the application program "reserving" a particular bank.

TRSDOS requires that any device driver or filter that is relocated to high memory (X'8000' through X'FFFF') reside in bank 0. The TRSDOS device handler always invokes bank 0 upon execution of any byte I/O service request (@PUT, @GET, @CTL, as well as other byte I/O SVCs that use @PUT/@GET/@CTL). This ensures that any filter or driver attached to the device in question will be available. If a RAM bank other than 0 was resident, it is restored upon return from the device handler. This ensures that device I/O is never impacted by bank switching.

TRSDOS also requires that all interrupt tasks reside in bank 0 or low memory (X'0000' through X'7FFF'). The interrupt task processor always enables bank 0 and restores whatever bank was previously resident. An interrupt task may perform a bank transfer from 0 to another bank provided the necessary linkage and stack area is used. This is discussed in more detail later.

All bank transfer requests must be performed using the @BANK SVC. This SVC provides four functions, three of which are interrogatory and one of which performs the actual bank switching.

As mentioned previously, the contents of banks other than 0 are managed by the application, not by TRSDOS. Therefore, the application needs a way of finding out if any given bank is available. For example, if an application wants to reserve use of bank 1, it must first check to see if bank 1 is free to use. This is done by using function 2 as follows:

```
LD      C,1          ;Specify bank 1
LD      B,2          ;Check BUR if bank in use
LD      A,@BANK      ;Set @BANK SVC (102)
RST     28H
JR      NZ,INUSE     ;NZ if bank already in use
```

Note that the return condition (NZ or Z) shows whether or not you can use the specified bank (it may not even be installed).

If the specified bank is available, you then need to reserve it. Do this by using function 3 as follows:

```
LD      C,1          ;Specify bank 1
LD      B,3          ;Set BUR to show "in use"
```

```
LD      A,@BANK       ;Set @BANK SVC (102)
RST     28H
JR      NZ,ERROR
```

You must check for an error by examining the Z flag. In general (discounting a system error), an NZ condition returned means that the specified bank is already in use. If you had performed a function 2 (testing to see if the bank was available) and got a not-in-use indication, but got an NZ condition on function 3, then the @BANK SVC routine has been altered and is probably unusable.

When an application no longer requires a memory bank, it can return the bank to a "free" state by using function 1 as follows:

```
LD      C,1           ;Specify bank 1
LD      B,1           ;Set BUR to show free
LD      A,@BANK       ;Set @BANK SVC (102)
RST     28H
```

No error condition is checked, as none is returned by TRSDOS. If you should mistakenly use function 1 with a bank that is nonexistent, an error is returned if you try to invoke the nonexistent bank.

To find out which bank is resident at any time, use function 4 as follows:

```
LD      B,4           ;Which bank is resident?
LD      A,@BANK       ;Set @BANK SVC (102)
RST     28H
```

The current bank number is returned in register A.

To exchange the current bank with the specified bank, use function 0. Since a memory transfer takes place in the address range X'8000' through X'FFFF,' the transfer cannot proceed correctly if the stack pointer (SP) contains a value that places the stack in that range. @BANK inhibits function 0 and returns an SVC error if the stack pointer violates this condition.

A bank can be used purely as a data storage buffer. The application's routines for invoking and indexing the bank switching probably reside in the user range X'3000' through X'7FFF.' As an example, the following code invokes a previously tested and reserved bank (via functions 2 and 3), accesses the buffer, and then restores the previous bank:

```
LD      C,1           ;Specify bank 1
LD      B,0           ;Bring up bank
LD      A,@BANK       ;Set @BANK SVC (102)
RST     28H
JR      NZ,ERROR      ;Error trap
PUSH    BC            ;Save old bank data
;
your code to access the buffer region
;
POP     BC            ;Recover old bank data
LD      A,@BANK       ;Set @BANK SVC (102)
RST     28H
JR      NZ,ERROR      ;Error trap
```

Note that the @BANK function 0 conveniently returns a zero in register B to effect a function 0 later, as well as provides the old bank number in register C. This means that you only have to save register pair BC, pop it when you want to restore the previous bank, and then issue the @BANK SVC.

Suppose you want to transfer to another bank from a routine that is executing in high memory. (Recall that the only limitation is that the stack must not be in high memory.) The @BANK SVC function 0 provides a technique for automatically transferring to an address in the new bank. This technique is called the transfer function. It relies on the assumption that since you are managing the entire 32K bank 1 or 2, your application should know exactly where it needs to transfer (that is, where the application originally placed the code to execute).

The code to perform a bank transfer is similar to the above example. Register pair HL is loaded with the transfer address. Register C, which contains the number of the bank to invoke, must have its high order bit (bit 7) set. After the specified bank is enabled, control is passed to the transfer address that is in HL. Upon entry to your routine in the new bank (referred to here as "PROGB"), register HL will contain the old return address so that PROGB will know where to return transfer. Register C will also contain the old bank number with bit 7 set and register B will contain a zero. This register set-up provides for an easy return to the routine in the old bank that invoked the bank transfer. An illustration of the transfer code follows:

```
            LD      C,1             ;Specify bank 1
            LD      B,0             ;Bring up bank 0
            LD      HL,(TRAADR)     ;Set the transfer
                                    ;address
            SET     7,C             ;and denote a
                                    ;transfer
            LD      A,@BANK         ;Set @BANK SVC (102)
            RST     28H
RETADR      JR      NZ,ERROR
```

Control is returned to "RETADR" under either of two conditions. If there was an error in executing the bank transfer (for example, if an invalid bank number was specified or the stack pointer is in high memory), the returned condition is NZ. If the transfer took place and PROGB transferred back, the returned condition is Z. Thus, the Z flag shows whether or not there was a problem with the transfer.

If PROGB needs to provide a return code, it must be done by using register pair DE, IX, or IY, as registers AF, BC, and HL are used to perform the transfer. (Or, some other technique can be used, such as altering the return transfer address to a known error trapping routine.)

PROGB should contain code that is similar to that shown earlier. For example, PROGB could be:

```
PROGB       PUSH    BC              ;Save old bank data
            PUSH    HL              ;Save the RET
                                    ;address
            ;
            your PROGB routines
            ;
            POP     HL              ;Recover transfer
                                    ;address
            POP     BC              ;Get bank transfer
                                    ;data
            LD      A,102           ;Set @BANK SVC
            RST     28H
            JR      NZ,ERROR        ;Error trap
```

PROGB saves the bank data (register BC). Don't forget that a transfer was effected and register C has bit 7 already set when PROGB is entered. PROGB also saves the address it needs to transfer back (which is in HL). It then performs whatever routines it has been coded for, recovers the transfer data, and issues the bank transfer request. As explained earlier, an NZ return condition from the @BANK SVC indicates that the bank transfer was not performed. You should verify that your application has not violated the integrity of the stack where the transfer data was stored.

Never place disk drivers, device drivers, device filters, or interrupt service routines in banks other than bank 0. It is possible to segment one of the above modules and place segments in bank 1 or 2, provided the segment containing the primary entry is placed in bank 0. You can transfer between segments by using the bank transfer techniques discussed above.

Device Driver and Filter Templates

Device independence has its roots in "byte I/O." Byte I/O is any I/O passed through a device channel one byte at a time.

Three primitive routines are available at the assembly language level for byte I/O. These byte I/O primitives can be used to build larger routines. The three primitives are the TRSDOS supervisor calls @GET, @PUT, and @CTL. @GET is used to input a byte from a device or file. @PUT is used to output a byte to a device or file. @CTL is used to communicate with the driver routine servicing the device or file.

Other supervisor calls perform byte I/O, such as @KBD (scan the keyboard and return the key code if a key is down), @DSP (display a character on the video screen), and @PRT (output a character to the line printer). These functions operate by first loading register pair DE with a pointer to a specific Device Control Block (DCB) assigned for use by the device, then issuing a @GET or @PUT SVC for input or output requests.

When TRSDOS passes control over to the device driver routine, the Z-80 flag conditions are unique for each different primitive. This enables the driver to establish which primitive was used to access the routine, so it can turn over the I/O request to the proper driver or filter subroutine according to the type of request — input, output, or control.

The following table shows the FLAG register conditions upon entry to a driver or filter:

C,NZ = @GET primitive
Z,NC = @PUT primitive
NZ,NC = @CTL primitive

Register B contains the I/O direction code: 1 = @GET, 2 = @PUT, 4 = @CTL. Register C contains the character code that was passed in the @PUT or @CTL supervisor call. Register IX points to the TYPE byte (DCB + 0) of the Device Control Block. Registers BC, DE, HL, and IX have been saved on the stack and are available for use. Register AF is not saved; if you want it preserved, your program must do so.

Your driver must start with a standard front-end header (see "Memory Header"):

```
BEGIN    JR      START          ;Go to actual code
                                ;beginning
         DEFW    MODEND-1       ;Last byte used by
                                ;module
         DEFB    7              ;Length of name
         DEFM    'MODNAME'      ;Name
MODDCB   DEFW    $-$            ;DCB ptr. for this
                                ;module
         DEFW    0              ;Reserved by TRSDOS
```

At the start of the actual module code, test the condition of the F register flags for @GET, @PUT, and @CTL:

```
START    EQU     $
;        Actual module code start
         JR      C,WASGET       ;Go if @GET request
         JR      Z,WASPUT       ;Go if @PUT request
         .                      ;Was @CTL request
```

At the label START, a test is made on the carry flag. If the carry was set, then the disk primitive must have been an input request (@GET). An input request could be directed to a part of the driver which only handles input from the device.

If the request was not from the @GET primitive, the carry will not be set. The next test checks to see if the zero flag is set. The zero condition is preset when a @PUT primitive was the initial request. The jump to WASPUT can go to a part of the driver that deals specifically with output to the device.

If neither the zero nor carry flags are set, the routine falls through to the next instruction (not shown), which would begin the part of the driver that handles @CTL calls. For example, you may want to have an RS-232C driver handle a BREAK by issuing a @CTL call so that the RS-232C driver emits a true modem break, but a CONTROL C would @PUT a X'03.'

Some drivers are written to assume that @CTL requests are to be handled exactly like @PUT requests. This is entirely up to the author and the function of the driver.

Note that when a device is routed to a disk file, TRSDOS ignores @CTL requests. That is, the @CTL codes are not written to the disk file.

On @GET requests, the character input should be placed in the accumulator. On output requests (either @PUT or @CTL), the character is obtained from register C. It is important for drivers and filters to observe return codes. Specifically, if the request is @GET and no byte is available, the driver returns an NZ condition and the accumulator contains a zero (that is, OR 1 : LD A,0 : RET). If a byte is available, the byte is placed in the accumulator and the Z flag is set (that is, LD A,CHAR : CP A : RET). If there is an input error, the error code is returned in the accumulator and the Z flag is reset (that is, LD A,ERRNUM : OR A : RET). On output requests, the accumulator will contain the byte output with the Z flag set if no error occurred. In the case of an output error, the accumulator must be loaded with the error code and the Z flag reset as shown above.

A filter module is inserted between the DCB and driver routine (or between the DCB and the current filter when it is applied to a DCB already filtered). The insertion is performed by the TRSDOS FILTER command once the filter module is resident and attached to a phantom DCB. The usual linkage for a filter is to access the chained module by calling the @CHNIO supervisor call with specific linkage data in registers IX and BC. Register IX is loaded with the filter's DCB pointer obtained from the memory header MODDCB pointer. Register B must contain the I/O direction code (1 = @GET, 2 = @PUT, 4 = @CTL). This code is already in register B when the filter is entered. You can either keep register B undisturbed or load it with the proper direction code. Also, output requests expect the output byte to be in register C.

The DCB pointer obtained from MODDCB is passed in register DE by the SET command and is loaded into MODDCB by your filter initialization routine. The initialization routine needs to relocate the filter to high memory and attach itself to the DCB assigned by the SET command. If the initialization front end had transferred the DCB pointer from DE to IX, then the following code could be used to establish the TYPE byte and vector for the filter:

```
LD      (IX),47H        ;Init DCB type to
LD      (IX+1),E        ;FILTER, G/P/C I/O,
LD      (IX+2),D        ;& stuff vector
```

A filter module can operate on input, output, control, or any combination based on the author's design. The memory header provides a region for user data storage conveniently indexed by the module.

An illustration of a filter follows. The purpose of this filter is to add a linefeed on output whenever a carriage return is to be sent. Although the filter requires no data storage, the technique for accessing data storage is shown.

```
BEGIN     JR      START          ;Branch to start
          DEFW    FLTEND-1       ;Last byte used
          DEFB    6              ;Name length
          DEFM    'SAMPLE'       ;Name
MODDCB    DEFW    0              ;Link to DCB
          DEFW    0              ;Reserved
;         Data storage area for your filter
CR        EQU     0DH
LF        EQU     0AH
DATA$     EQU     $
DATA1     EQU     $-DATA$
          DEFB    0              ;Data storage
DATA2     EQU     $-DATA$
          DEFB    0              ;Data storage
;         Start of filter
START     JR      Z,GOTPUT       ;Go if @PUT
;         @GET and @CTL requests are chained to
;         the next module attached to the device.
;         This is accomplished by falling through
;         to the @CHNIO call. Note that the sample
;         filter does not affect the B register,
;         so the filter does not have to load it
;         with the direction code.
FLTPUT    PUSH    IX             ;Save your data
                                 ;pointer
          LD      IX,(MODDCB)
RX01      EQU     $-2            ;Grab the DCB vector
          LD      A,@CHNIO       ;and chain to it
          RST     28H
          POP     IX
          RET
;         Filter code
GOTPUT    LD      IX,PFDATA$     ;Base register is
RX02      EQU     $-2            ;used to index data
          LD      A,C            ;Get character to
                                 ;test
          CP      CR             ;If not CR, put it
          JR      NZ,FLTPUT
          CALL    FLTPUT         ;else put it
RX03      EQU     $-2
          RET     NZ             ;Back on error
          LD      C,LF           ;Add linefeed
          JR      FLTPUT
FLTEND    EQU     $
;         Relocation table
RELTAB    DEFW    RX01,RX02,RX03
TABLEN    EQU     $-RELTAB/2
```

The relocation table, RELTAB, would be used by the filter initialization relocation routine.

@CTL Interfacing to Device Drivers

This section discusses the @CTL functions supported by the system device drivers. To invoke a @CTL function, point register pair DE to the Device Control Block (DCB), load the function code into register C, and issue the @CTL supervisor call. You can locate the DCB address by either 1) using the @GTDCB SVC, or 2) using the @OPEN SVC to open a File Control Block containing the device specification and using the FCB address. See the @CTL supervisor call for a list of the function codes and their meanings.

The @CTL functions are listed below for each driver.

Keyboard Driver (resident driver assigned to *KI)

A function value of X'Ø3' clears the type-ahead buffer. This serves the same purpose as repeated calls to @KBD until no character is available.

A function value of X'FF' is reserved for system use.

All other function values are treated as @GET requests.

The module name assigned to this driver is "$KI".

Video Driver (resident driver assigned to *DO)

All @CTL requests are treated as if they were @PUT requests.

The module name assigned to this driver is "$DO".

Printer Driver (resident driver assigned to *PR)

The printer driver is transparent to all code values when requested by the @PUT SVC. That means that all values from X'ØØ' through X'FF' (Ø-255) can be sent to the printer. If the FORMS filter is attached to the *PR device, then various codes are trapped and used by the filter according to parameters specified with the FORMS library command, as follows:

X'ØD' — Generates a carriage return and optionally a linefeed (ADDLF). Generates form feeds as required.
X'ØA' — Treated the same way as X'ØD'.
X'ØC' — Generates form feeds (via repeated line feeds if soft form feed). (FFHARD = OFF)
X'Ø9' — Advances to next tab column.
X'Ø6' — Sets top-of-form by resetting the internal line counter to zero.

Other character codes may be altered if the user translation option of the FORMS command (XLATE) is set.

The printer driver accepts a function value of X'ØØ' via the @CTL request to return the printer status. If the printer is available, the Z flag will be set and register A will contain X'3Ø'. If the Z flag is reset, register A will contain the four high-order bits of the parallel printer port (bits 4-7).

The module name assigned to the printer driver is "$PR". The module name of the FORMS filter is "$FF".

COM Driver (non-resident driver for the RS-232C)

This driver handles the interfacing between the RS-232C hardware and byte I/O (usually the *CL device).

A @CTL function value of X'ØØ' returns an image of the RS-232 status register in the accumulator. The Z flag will be set if the RS-232 is available for "sending" (that is, if the transmit holding register is empty and the flag conditions match as specified by SETCOM).

A function value of X'Ø1' transmits a "modem break" until the next character is @PUT to the driver.

A function value of X'Ø2' re-initializes the UART to the values last established by SETCOM.

A function value of X'Ø4' enables or disables the WAKEUP feature.

All other function values are ignored and the driver returns with register A containing a zero value and the Z flag set.

The WAKEUP feature is useful for application software specializing in communications. The RS-232 hardware can generate a machine interrupt under any of three conditions: when the transmit holding register is empty, when a received character is available, or when an error condition has been detected (framing error, parity error, and so on). The COM driver makes use of the

"received character available" interrupt to take control when a fully formed character is in the holding register. The COM driver services the interrupt by reading the character and storing it in a one-character buffer. COM then normally returns from the interrupt.

An application can request that, instead of returning, control be passed to the application for immediate attention. Note that this action would occur during interrupt handling, and any processing by the application must be kept to a minimum before control is returned to COM via a RET instruction.

If you use a @CTL function value of X'04' then register IY must contain the address of the handling routine in your application. Upon return from the @CTL request, register IY contains the address of the previous WAKEUP vector. This should be restored when your application is finished with the WAKEUP feature.

When control is passed to your WAKEUP vector upon detection of a "receive character available" interrupt, certain information is immediately available. Register A contains an image of the UART status register. The Z flag is set if a valid character is actually available. The character, if any, is in the C register.

Since system overhead takes a small amount of time in the @GET supervisor call, you may need to @GET the character via standard device interfacing. This ensures that any filtering or linking in the *CL device chain will be honored. If, on the other hand, your application is attempting to transfer data at a very high rate (9600 baud or higher), you may need to bypass the @GET SVC and use the character immediately available in the C register. Note that this procedure bypasses the normal device chain (device routing and linking).

The module name of the COM driver is "$CL".

8/Using the Supervisor Calls

Supervisor Calls (SVCs) are operating system routines that are available to assembly language programs. These routines alter certain system functions and conditions, provide file access, and perform various computations. They also perform I/O to the keyboard, video display, and printer.

Each SVC has a number which you specify to invoke it. These numbers range from 0 to 104.

Calling Procedure

To call a TRSDOS SVC:

1. Load the SVC number for the desired SVC into register A. Also load any other registers which are needed by the SVC, as detailed under Supervisor Calls.
2. Execute a RST 28H instruction.

Note: If the SVC number supplied in register A is invalid, the system prints the message "System Error *xx*", where *xx* is usually 2B. It then returns you to TRSDOS Ready (*not* to the program that made the invalid SVC call).

The alternate register set (AF', BC', DE', HL') is not used by the operating system.

Program Entry and Return Conditions

When a program executed from the @CMNDI SVC is entered, the system return address is placed on the top of the stack. Register HL will point to the first non-blank character following the command name. Register BC will point to the first byte of the command line buffer.

Three methods of return from a program back to the system are available: the @ABORT SVC, the @EXIT SVC, and the RET instruction. For application programs and utilities, the normal return method is the @EXIT SVC. If no error condition is to be passed back, the HL register pair must contain a zero value. Any non-zero value in HL causes an active JCL to abort.

The @ABORT SVC can be used as an error return back to the system; it automatically aborts any active JCL processing. This is done by loading the value X'FFFF' into the HL register pair and internally executing an @EXIT SVC.

If stack integrity is maintained, a RET instruction can be used since the system return address is put on the stack by @CMNDI. This allows a return if the program was called with @CMNDR.

Most of the SVCs in TRSDOS Version 6 set the Z flag when the operation specified was successful. When an operation fails or encounters an error, the Z flag is reset (also known as NZ flag set) and a TRSDOS error code is placed in the A register. The remaining SVCs use the Z/NZ flag in differing ways, so you should refer to the description of the SVCs you are using to determine the exit conditions.

Supervisor Calls

The TRSDOS Supervisor Calls are:

Keyboard SVCs

@KBD
@KEY
@KEYIN

Byte I/O SVCs

@CTL
@GET
@PUT

Printer and Video SVCs

@DSP
@DSPLY
@LOGER
@LOGOT
@MSG
@PRT
@PRINT
@VDCTL

File Control SVCs

@CLOSE
@FEXT
@FNAME
@FSPEC
@INIT
@REMOV
@OPEN
@RENAM

Disk SVCs

@DCINIT
@DCRES
@DCSTAT
@RDSEC
@RDSSC
@RSLCT
@RSTOR
@SEEK
@SLCT
@STEPI
@VRSEC
@WRSEC
@WRSSC
@WRTRK

Disk File Handler SVCs

@BKSP
@CKEOF
@LOC
@LOF
@PEOF
@POSN
@READ
@REW
@RREAD
@RWRIT
@SEEKSC
@SKIP
@VER
@WEOF
@WRITE

System Control SVCs

@ABORT
@BREAK
@CMNDI
@CMNDR
@EXIT
@FLAGS
@HIGH$
@IPL
@LOAD
@RUN

TRSDOS Task Control SVCs

@ADTSK
@CKTSK
@KLTSK
@RMTSK
@RPTSK

Special Purpose Disk SVCs

@DIRRD
@DIRWR
@GTDCT
@HDFMT
@RDHDR
@RDTRK

Special Overlay SVCs

@CKDRV
@DEBUG
@DODIR
@ERROR
@PARAM
@RAMDIR

Miscellaneous SVCs

@BANK
@DATE
@DECHEX
@DIV8
@DIV16
@HEXDEC
@HEX8
@HEX16
@MUL8
@MUL16
@PAUSE
@SOUND
@TIME
@WHERE

Special Purpose SVCs

@CHNIO
@GTDCB
@GTMOD

See the pages that follow for a detailed description of each supervisor call.

Abort Program

Loads HL with an X'FFFF' error code and exits through the @EXIT supervisor call. Any active JCL processing is aborted.

Entry Conditions:

A = 21 (X'15')

General:

This SVC does not return.

Example:

See the example for @EXIT in Sample Program B, lines 206-207.

Add an Interrupt Level Task

Adds an interrupt level task to the real time clock task table. The task slot number can be 0-11; however, some slots are already assigned to certain functions in TRSDOS. Slot assignments 0-7 are low priority tasks executing every 266.67 milliseconds. Slots 8-10 are medium priority tasks executing every 33.33 milliseconds. Slot 11 is a high priority task, executing every 16.66 milliseconds High Speed or 33.33 milliseconds Low Speed. The system uses task slots 3, 7, 9, and 10 for the ALIVE, TRACE, SPOOL, and TYPE-AHEAD functions, respectively.

It is a good practice to remove an existing task (using the @RMTSK or @KLTSK SVC) before installing a new task in the same task slot.

Entry Conditions:

A = 29 (X'1D')
DE = *pointer to Task Control Block (TCB)*
C = *task slot assignment* (0-11)

Exit Conditions:

Success always.
HL and AF are altered by this SVC.

The Task Control Block, or TCB, is a 2-byte block of RAM which contains the address of the task driver entry point. If your task is prefixed with the memory header described earlier under "Device Access," then the TCB can be stored in the memory header data storage area. If the task is not a driver or filter, the TCB can be stored in the memory header location MODDCB. Upon entry to your task routine, the IX register contains the TCB address.

Example:

See Sample Program F, lines 109-120.

Memory Bank Use

Controls 32K memory bank operation. The top half of the main 64K block is bank 0, and the alternate 64K block is divided into banks 1 and 2. The system maintains two locations to perform bank management. These areas are known as "bank available RAM" (BAR) and "bank in use RAM" (BUR).

If the Stack Pointer is not X'7FFE' or lower, the SVC aborts with an Error 43 only if B = 0.

Entry Conditions:

A = 102 (X'66')
B selects one of the following functions:
If B = 0, the specified bank is selected and is made addressable. The 32K bank starts at X'8000' and ends at X'FFFF'.
C = *bank number to be selected* (0-2)
If bit 7 is set, then execution will resume in the newly loaded bank at the address specified.
HL = *address to start execution in the new bank*
If B = 1, reset BUR and show the bank not in use.
C = *bank number to be selected* (0-2)
If B = 2, test BUR if bank is in use.
C = *bank number to be selected* (0-2)
If B = 3, set BUR to show bank in use.
C = *bank number to be selected* (0-2)
If B = 4, return number of bank currently selected.

Exit Conditions:

If B = 0:
Success, Z flag set.
C = *the bank number that was replaced.* If bit 7 was set in register C on entry, it is also set on exit.
HL = *SVC return address.* By keeping the contents of C and HL, you can later return to the instruction following the first @BANK SVC. See "Interfacing RAM Banks 1 and 2" for more information.
Failure, NZ flag set. Bank not present or parameter error.
A = *error number*

If B = 1:
Success, Z flag set. Bank available for use.
Failure, NZ flag set. Bank not present.

If B = 2:
Success always.
If Z flag is set, then the bank is available for use.
If NZ flag is set, then test register A:
If A ≠ X'2B', then the bank is either in use or it does not exist on your machine. Banks 1 and 2 produce this error on a 64K machine.
If A = X'2B', then an entry parameter is out of range.

If B = 3:
Success, Z flag set. Bank is now reserved for your use.
Failure, NZ flag set. Test register A:
If A ≠ X'2B', then the bank is already in use or does not exist. Banks 1 and 2 produce this error on a 64K machine.
If A = X'2B', then an entry parameter is out of range.

If B = 4:

Success always.

A = *number of the bank which is currently resident*

General:

AF is altered for all functions.

BC is altered if the SVC is successful.

Example:

See the section "Interfacing RAM Banks 1 and 2."

Backspace One Logical Record

Performs a backspace of one logical record.

Entry Conditions:

A =61 (X'3D')
DE=*pointer to FCB of the file to backspace*

Exit Conditions:

If the Z flag is set or if A=X'1C' or X'1D' then the operation was successful. The LOC pointer to the file was backspaced one record. Otherwise, A=*error number*.
If A=X'1C' is returned, the file pointer is positioned at the end of the file. Any Appending operations would be performed here.
If A=X'1D' is returned, the file pointer is positioned beyond the end of the file.

General:

Only AF is altered by this SVC.
If the LOC pointer was at record 0 when the call was executed, the results are indeterminate.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Set Break Vector

Sets a user or system break vector. The BREAK vector is an abort mechanism; there is no return.

The BREAK vector executes whenever the following conditions occur at the same time: 1) the Program Counter is greater than X'2400' 2) the BREAK key is pressed, and 3) a real time clock interrupt which executes 30 times per second occurs.

After executing this SVC, you must reset bit 4 of SFLAG$. The BREAK flag in KFLAG$ (bit 0) requires the setting of SFLAG$ bit 4 and a delay of 0.1 to 0.5 second to clear any other interrupts that may be pending. Then you can enter your BREAK key handler (in which the BREAK key bit in SFLAG$ is reset). See KFLAG$ and SFLAG$ in the section about the @FLAGS SVC for more information.

Entry Conditions:

A = 103 (X'67')
HL = *user break vector*
HL = 0 (sets system break vector)

Exit Conditions:

Success always.
HL = *existing break vector* (if user break vector was set)

Note: @EXIT and @CMNDI automatically restore BREAK to the system handler. @CMNDR does not do this.

Pass Control to Next Module in Device Chain

Passes control to the next module in the device chain.

Entry Conditions:

A = 20 (X'14')
IX = *contents of DCB in the header block*
B = *GET/PUT/CTL direction code (1/2/4)*
C = *character* (if output request)

General:

IX is not checked for validity.

Example:

See the section "Device Driver and Filter Templates."

Check Drive

Checks a drive reference to ensure that the drive is in the system and a TRSDOS Version 6 or LDOS 5.1.3 (Model III Hard Disk Operating System) formatted disk is in place.

Entry Conditions:

A=33 (X'21')
C=*logical drive number* (0-7)

Exit Conditions:

Success always.
If Z flag is set, the drive is ready.
If CF is set, the disk is write protected.
If NZ flag is set, the drive is not ready. The user may examine DCT+0 to see if the drive is disabled.

Example:

See Sample Program D, lines 35-55.

Check for End-Of-File

Checks for the end of file at the current logical record number.

Entry Conditions:

A = 62 (X'3E')
DE = *pointer to the FCB of the file to check*

Exit Conditions:

Success always.
If Z flag is set, LOC does not point at the end of file (LOC < LOF).
If NZ flag is set, test A for error number:
If A = X'1C', LOC points at the end of the file (LOC = LOF).
If A = X'1D', LOC points beyond the end of the file (LOC > LOF).
If A ≠ X'1C' or X'1D', then A = *error number.*

General:

Only AF is altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 300-311 and 352-353. (A substantial part of this code could be replaced with the @CKEOF SVC.)

Check if Task Slot in Use

Checks to see if the specified task slot is in use.

Entry Conditions:
A = 28 (X'1C')
C = *task slot to check* (0-11)

Exit Conditions:
Success always.
If Z flag is set, the task slot is available for use.
If NZ flag is set, the task slot is already in use.

General:
AF and HL are altered by this SVC.

Example:
See Sample Program F, lines 70-73.

Close a File or Device

Terminates output to a file or device. Any unsaved data in the buffer area is saved to disk and the directory is updated. All files that have been written to must be closed, as well as all files opened with UPDATE or higher access.

Entry Conditions:

A = 60 (X'3C')
DE = *pointer to FCB or DCB to close*

Exit Conditions:

Success, Z flag set. The file or device was closed. The filespec (excluding the password) or the devspec is returned to the FCB or DCB.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program C, lines 355-363.

Execute Command with Return to System

Passes a command string to TRSDOS for execution. After execution is complete, control returns to TRSDOS Ready. If the command gets an error, it still returns to TRSDOS Ready.

Entry Conditions:

A = 24 (X'18')
HL = *pointer to buffer containing command string terminated with X'0D'* (up to 80 bytes, including the X'0D')

General:

This SVC does not return.

Example:

See Sample Program E, lines 43-58.

Execute Command

Executes a command or program and returns to the calling program. The executed program should maintain the Stack Pointer and exit via a RET instruction. All TRSDOS library commands comply with this requirement.

If bit 4 of CFLAG$ is set (see the @FLAGS SVC), then @CMNDR executes only system library commands.

Entry Conditions:

A = 25 (X'19')
HL = *pointer to buffer containing command string terminated with X'0D'* (up to 80 bytes, including the X'0D')

Exit Conditions:

Success always.
HL = *return code* (See the section "Converting to TRSDOS Version 6" for information on return codes.)
Registers AF, BC, DE, IX, and IY are altered by the command or program executed by this SVC.
If the command invokes a user program which uses the alternate registers, they are modified also.

Example:

See Sample Program E, lines 18-29.

Output a Control Byte

Outputs a control byte to a logical device. The DCB TYPE byte (DCB+Ø, Bit 2) must permit CTL operation. See the section "@CTL Interfacing to Device Drivers" for information on which of the functions listed below are supported by the system device drivers.

Entry Conditions:

A =5 (X'Ø5')
DE=*pointer to DCB to control output*
C selects one of the following functions:
If C=Ø, the status of the specified device will be returned.
If C=1, the driver is requested to send a BREAK or force an interrupt.
If C=2, the initialization code of the driver is to be executed.
If C=3, all buffers in the driver are to be reset. This causes all pending I/O to be cleared.
If C=4, the wakeup vector for an interrupt-driven driver is specified by the caller.
IY=address to vector when leaving driver. If IY=Ø, then the wakeup vector function is disabled.
If C=8, the next character to be read will be returned. This allows data to be "previewed" before the actual @GET returns the character.

Exit Conditions:

If C=Ø,
Z flag set, device is ready
NZ flag set, device is busy
A=status image, if applicable
Note: This is a hardware dependent image.
If C=1,
Success, Z flag set. BREAK or interrupt generated.
Failure, NZ flag set
A=*error number*
If C=2,
Success, Z flag set. Driver initialized.
Failure, NZ flag set
A=*error number*
If C=3,
Success, Z flag set. Buffers cleared.
Failure, NZ flag set.
A=*error number*
If C=4,
Success, Z flag set. Wakeup vector set or disabled.
IY=*previous vector address*
Failure, NZ flag set
A=*error number*
If C=8,
Success, Z flag set. Next character returned.
A=*next character in buffer*
Failure, NZ flag set. Test register A:
If A=Ø, no pending character is in buffer
If A≠Ø, A contains *error number*. (TRSDOS driver returns Error 43.)

General:

BC, DE, HL, and IX are saved.

Function codes 5 to 7, 9 to 31, and 255 are reserved for the system. Function codes 32 to 254 are available for user definition.

Entry and exit conditions for user-defined functions are up to the design of the user-supplied driver.

Example:

See the section "Device Driver and Filter Templates."

Get Date

Returns today's date in display format (MM/DD/YY).

Entry Conditions:

A = 18 (X'12')
HL = *pointer to 8-byte buffer to receive date string*

Exit Conditions:

Success always.
HL = *pointer to the end of the buffer supplied + 1*
DE = *pointer to start of DATE$ storage area in TRSDOS*
BC is altered by this SVC.

Example:

See Sample Program F, lines 252-253.

Initialize the FDC

Issues a disk controller initialization command. The floppy disk driver treats this the same as @RSTOR (SVC 44).

Entry Conditions:
A = 42 (X'2A')
C = *logical drive number* (Ø-7)

Exit Conditions:
Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:
See the example for @CKDRV in Sample Program D, lines 38-39.

Reset the FDC

Issues a disk controller reset command. The floppy disk driver treats this the same as @RSTOR (SVC 44).

Entry Conditions:
A = 43 (X'2B')
C = *logical drive number* (Ø-7)

Exit Conditions:
Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:
See the example for @CKDRV in Sample Program D, lines 38-39.

Test if Drive Assigned in DCT

Tests to determine whether a drive is defined in the Drive Code Table (DCT).

Entry Conditions:
A= 40 (X'28')
C=*logical drive number* (0-7)

Exit Conditions:
Success always.
If Z is set, the specified drive is already defined in the DCT.
If NZ is set, the specified drive is not defined in the DCT.

General:
Only AF is altered by this SVC.

Example:
See Sample Program D, lines 27-33.

@DEBUG SVC Number 27

Enter DEBUG

Forces the system to enter the DEBUG utility. Pressing Ⓖ (ENTER) from the DEBUG monitor causes program execution to continue with the next instruction. If you want to use the functions in the extended debugger when DEBUG is entered in this fashion, you must issue the DEBUG (E) command (optionally with the @CMNDR SVC) before this SVC is executed.

Entry Conditions:
A=27 (X'1B')

General:
This SVC does not return unless Ⓖ is entered in DEBUG.

Example:
See Sample Program A, lines 54-6Ø.

Convert Decimal ASCII to Binary

Converts a decimal ASCII string to a 16-bit binary number. Overflow is not trapped. Conversion stops on the first out-of-range character.

Entry Conditions:

A = 96 (X'60')
HL = *pointer to decimal string*

Exit Conditions:

Success always.
BC = *binary conversion of ASCII string*
HL = *pointer to the terminating byte*
AF is altered by this SVC.

Example:

See Sample Program B, lines 88-95.

@DIRRD SVC Number 87

Directory Record Read

Reads a directory sector that contains the directory entry for a specified Directory Entry Code (DEC). The sector is placed in the system buffer and the register pair HL points to the first byte of the directory entry specified by the DEC.

Entry Conditions:

A = 87 (X'57')
B = *Directory Entry Code of the file*
C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.
HL = *pointer to directory entry specified by register B*
Failure, NZ flag set.
A = *error number*
HL is altered.

General:

AF is always altered.
If the drive does not contain a disk, this SVC may hang indefinitely waiting for formatted media to be placed in the drive. The programmer should perform a @CKDRV SVC before executing this call.
If the Directory Entry Code is invalid, the SVC may not return or it may return with the Z flag set and HL pointing to a random address. Care should be taken to avoid using the wrong value for the DEC in this call.

Example:

See Sample Program C, lines 152-174.

Directory Record Write

Writes the system buffer back to the disk directory sector that contains the directory entry of the specified DEC.

Entry Conditions:

A = 88 (X'58')
B = *Directory Entry Code of the file*
C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.
HL = *pointer to directory entry specified by register B*
Failure, NZ flag set.
A = *error number*
HL is altered.

General:

AF is always altered.
If the drive does not contain a disk, this SVC may hang indefinitely waiting for formatted media to be placed in the drive. The programmer should perform a @CKDRV SVC before executing this call.
If the Directory Entry Code is invalid, the SVC may not return or it may return with the Z flag set and HL pointing to a random address. Care should be taken to avoid using the wrong value for the DEC in this call.

Example:

See the example for @DIRRD in Sample Program C, lines 152-174.

8-Bit Divide

Performs an 8-bit unsigned integer divide.

Entry Conditions:

A = 93 (X'5D')
E = *dividend*
C = *divisor*

Exit Conditions:

Success always.
A = *quotient*
E = *remainder*
No other registers are altered.

Example:

See Sample Program B, lines 61-64.

16-Bit by 8-Bit Divide

Performs a division of a 16-bit unsigned integer by an 8-bit unsigned integer.

Entry Conditions:

A = 94 (X'5E')
HL = *dividend*
C = *divisor*

Exit Conditions:

Success always.
HL = *quotient*
A = *remainder*
No other registers are altered.

Example:

See Sample Program B, lines 105-109.

Do Directory Display / Buffer

Reads files from a disk directory or finds the free space on a disk. The directory information is either displayed on the screen (in five-across format) or sent to a buffer. The directory information buffer consists of 18 bytes per active, visible file: the first 16 bytes of the directory record, plus the ERN (ending record number). An X'FF' marks the buffer end.

Entry Conditions:

A = 34 (X'22')
C = *logical drive number* (0-7)
B selects one of the following functions:

If B = 0, the directory of the visible, non-system files on the disk in the specified drive is displayed on the screen. The filenames are displayed in columns, 5 filenames per line.

If B = 1, the directory is written to memory.
HL = *pointer to buffer to receive information*

If B = 2, a directory of the files on the specified drive is displayed for files that are visible, non-system, and match the extension partspec pointed to by HL.
HL = *partspec for the filename's extension*
This field must contain a valid 3-character extension, padded with dollar signs ($). For example, to display all visible, non-system files that have the letter 'C' as the first character of the extension, HL should point to the string "C$$".

If B = 3, a directory of the files on the specified drive is written to the buffer that is specified by HL for files that match the extension partspec pointed to by HL.
HL = *pointer to the 3-byte partspec and to the buffer to receive the directory records* (see general notes)
Keep in mind that the area pointed to by HL is shared. If you are using this buffer more than once, you have to re-create the partspec in the buffer before each call because the previous call will have erased the partspec by writing the directory records.

If B = 4, the disk name, original free space, and current free space on the disk is read.
HL = *pointer to a 20-byte buffer to receive information*

Exit Conditions:

Success, Z flag set.

If B = 1 or 3, the directory records have been stored.
HL = *pointer to the beginning of the buffer*

If B = 0 or 2, the filenames or matching filenames are displayed with 5 filenames per line.

If B = 4, the disk name and free space information are stored in the format:

Bytes 0-7 = Disk name. Disk name is padded on the right with blanks (X'20').
Bytes 8-15 = Creation date (the date the disk was formatted or was the target disk in a mirror image backup). The date is in the format MM/DD/YY.
Bytes 16-17 = Total K originally available in binary LSB-MSB format.
Bytes 18-19 = Free K available now in binary LSB-MSB format.

HL = *pointer to the beginning of the data area*

Failure, NZ flag set.
A = *error number*

General:

AF is the only register altered by this SVC.

The size of the buffer to receive directory records must be large enough to hold directory entries for the maximum number of files allowed on the drive and disk you specify. For example, if the drive is a hard disk, you must be able to store 256 directory entries, and each entry requires 18 bytes of storage. For more information on calculating the amount of space needed for this buffer, see the tables under "Directory Records." They give the maximum number of entries allowed on a given type of disk. You must add 2 records to this value when B = 1 to store the directory entry for DIR/SYS and BOOT/SYS.

Example:

See Sample Program E, lines 32-40.

Display Character

Outputs a byte to the video display. The byte is displayed at the current cursor position.

Entry Conditions:
A = 2 (X'02')
C = *byte to display*

Exit Conditions:
Success, Z flag set.
A = *byte displayed*
Failure, NZ flag set.
A = *error number*

General:
DE is altered by this SVC.

Example:
See Sample Program C, lines 219-221.

Display Message Line

Displays a message line, starting at the current cursor position. The line must be terminated with either a carriage return (X'0D') or an ETX (X'03'). If an ETX terminates the line, the cursor is positioned immediately after the last character displayed.

Entry Conditions:

A = 10 (X'0A')
HL = *pointer to first byte of message*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

AF and DE are altered by this SVC.

Example:

See Sample Program C, lines 35-37.

Entry to Post an Error Message

Provides an entry to post an error message. If bit 7 of register C is set, the error message is displayed and return is made to the calling program. If bit 6 is not set, the extended error message is displayed.

```
*** Errcod=xx, Error message string ***
      <filespec or devspec>
Referenced at X'dddd'
```

If bit 6 is set, then only the "Error message string" is displayed. This bit is ignored if bit 6 of SFLAG$ (the extended error message bit) is set. If bit 6 of CFLAG$ is set, then no error message is displayed. If bit 7 of CFLAG$ is set, then the "Error message string" is placed in a user buffer pointed to by register pair DE. See @FLAGS (SVC 101) for more information on SFLAG$ and CFLAG$.

Entry Conditions:

A = 26 (X'1A')
C = *error number with bits 6 and 7 optionally set*

Exit Conditions:

Success always.

General:

To avoid a looping condition that could result from the display device generating an error, do not check for errors after returning from @ERROR.

If you do not set bit 6 of register C, then you should execute this SVC only after an error has actually occurred.

Example:

See Sample Program C, lines 374-384.

Exit to TRSDOS

This is the normal program exit and return to TRSDOS. An error exit can be done by placing a non-zero value in HL. Values 1 to 62 indicate a primary error as described in TRSDOS Error Codes (Appendix A). (A non-zero value in HL causes an active JCL to abort.)

Entry Conditions:

A = 22 (X'16')
HL = *Return Code*
If HL = 0, then no error on exit.
If HL ≠ 0, then the @ABORT SVC returns X'FFFF' in HL automatically.

General:

This SVC does not return.

Example:

See Sample Program B, lines 206-207.

@FEXT SVC Number 79

Set Up Default File Extension

Inserts a default file extension into the File Control Block if the file specification entered contains no extension. @FEXT must be done before the file is opened.

Entry Conditions:

A = 79 (X'4F')
DE = *pointer to FCB*
HL = *pointer to default extension* (3 characters; alphabetic characters must be upper case and first character must be a letter)

Exit Conditions:

Success always.
AF and BC are altered by this SVC.
If the default extension is used, HL is also altered.

Example:

See Sample Program C, lines 111-132.

Point IY to System Flag Table

Points the IY register to the base of the system flag table. The status flags listed below can be referenced off IY. You can alter those bits marked with an asterisk (*). Bits without an asterisk are indicators of current conditions, or are unused or reserved.

Note: You may wish to save KFLAG$ and SFLAG$ if you intend to modify them in your program, and restore them on exit.

Entry Conditions:
A = 101 (X'65')

Exit Conditions:
Success always.
IY = *pointer to the following system information:*

IY − 1 Contains the overlay request number of the last system module resident in the system overlay region.

IY + 2 = CFLAG$
- * bit 7 — If set, then @ERROR will transfer the "Error message string" to your buffer instead of displaying it. The message is terminated with X'0D'.
- * bit 6 — If set, do not display system error messages 0-62. See @ERROR (SVC 26) for more information.
- * bit 4 — If set, then @CMNDR will execute only system library commands.
- bit 3 — If set, @RUN is requested from either the SET or SYSTEM (DRIVER =) commands.
- bit 2 — If set, @KEYIN is executing due to a request from SYS1.
- bit 1 — If set, @CMNDR is executing. This bit is reset by @EXIT and @CMNDI.
- * bit 0 — If set, HIGH$ cannot be changed using @HIGH$ (SVC 100). This bit is reset by @EXIT and @CMNDI.

IY + 3 = DFLAG$ (device flag)
- * bit 7 — "1" if GRAPHIC printer capability desired on screen print (CONTROL :) causes screen print. See the SYSTEM (GRAPHIC) command under "Technical Information on TRSDOS Commands and Utilities.")
- bit 6 — "1" if KSM module is resident
- bit 5 — Currently unused
- bit 4 — "1" if MemDisk active
- bit 3 — Reserved
- bit 2 — "1" if Disk Verify is enabled
- * bit 1 — "1" if TYPE-AHEAD is active
- bit 0 — "1" if SPOOL is active

IY + 5 = FEMSK$ (mask for port 0FEH)

IY + 8 = IFLAG$ (international flag)
- * bit 7 — If "1," 7-bit printer filter is active
  If "0," normal 8-bit filters are present
- * bit 6 — If "1," international character translation will be performed by printer driver
  If "0," characters received by printer driver will be sent to the printer unchanged
- bit 5 — Reserved for future languages
- bit 4 — Reserved for future languages
- bit 3 — Reserved for future languages
- bit 2 — Reserved for future languages
- bit 1 — If "1," German version of TRSDOS is present

bit 0 — If "1," French version of TRSDOS is present
If bits 5-0 are all zero, then USA version of TRSDOS is present.

IY + 10 = KFLAG$ (keyboard flag)
- bit 7 — "1" if a character is present in the type-ahead buffer
- bit 6 — Currently unused
- * bit 5 — "1" if CAPS lock is set
- bit 4 — Currently unused
- bit 3 — Currently unused
- * bit 2 — "1" if (ENTER) has been pressed
- * bit 1 — "1" if (SHIFT) (@) has been pressed (PAUSE)
- * bit 0 — "1" if (BREAK) has been pressed

Note: To use bits 0-2, you must first reset them and then test to see if they become set.

IY + 12 = MODOUT (image of port 0ECH)

IY + 14 = OPREG$ (memory management & video control image)

IY + 18 = SFLAG$ (system flag)
- bit 7 — "1" if DEBUG is to be turned on
- * bit 6 — "1" if extended error messages desired (see @ERROR for message format); overrides the setting of bit 6 of register C on @ERROR (SVC 26) and should be used only when testing
- bit 5 — "1" if DO commands are being executed
- * bit 4 — "1" if BREAK disabled
- bit 3 — "1" if the hardware is running at 4 mhz (SYSTEM (FAST)). If "0," the hardware is running at 2 mhz (SYSTEM (SLOW)).
- * bit 2 — "1" if LOAD called from RUN
- * bit 1 — "1" if running an EXECute only file
- * bit 0 — "1" specifies no check for matching LRL on file open and do not set file open bit in directory. This bit should be set just before executing an @OPEN (SVC 59) if you want to force the opened file to be READ only during current I/O operations. As soon as either call is executed, SFLAG$ bit 0 is reset. If you want to disable LRL checking on another file, you must set SFLAG$ bit 0 again.

IY + 21 = VFLAG$
- bit 7 — Reserved for system use
- * bit 6 — "1" selects solid cursor, "0" selects blinking cursor
- bit 5 — Reserved for system use
- * bit 4 — "1" if real time clock is displayed on the screen
- bits 0-3 — Reserved for system use

IY + 22 = WRINTMASK$ (mask for WRINTMASK port)

IY + 26 = SVCTABPTR$ (pointer to the high order byte of the SVC table address; low order byte = 00)

IY + 27 = Version ID byte (60H = TRSDOS version 6.0.x.x, 61H = TRSDOS version 6.1.x.x, etc.)

IY − 47 = Operating system release number. Provides a third and fourth character (12H = TRSDOS version x.x.1.2)

IY + 28
to
IY + 30 = @ICNFG vector

IY + 31
to
IY + 33 = @KITSK vector

@FNAME

SVC Number 80

Get Filename

Gets the filename and extension from the directory using the specified Directory Entry Code (DEC) for the file.

Entry Conditions:

A = 80 (X'50')
DE = *pointer to 15-byte buffer to receive filename/extension:drive, followed by a X'0D' as a terminator*
B = *DEC of desired file*
C = *logical drive number of drive containing file* (0-7)

Exit Conditions:

Success, Z flag set.
HL = *pointer to directory entry specified by register B*
Failure, NZ flag set.
A = *error number*
HL is altered.

General:

AF and BC are always altered.
If the drive does not contain a disk, this SVC may hang indefinitely waiting for formatted media to be placed in the drive. The programmer should perform a @CKDRV SVC before executing this call.
If the Directory Entry Code is invalid, the SVC may not return or it may return with the Z flag set and HL pointing to a random address. Care should be taken to avoid using the wrong value for the DEC in this call.

Example:

See Sample Program C, lines 274-286.

Assign File or Device Specification

Moves a file or device specification from an input buffer into a File Control Block (FCB). Conversion of lower case to upper case is made automatically.

Entry Conditions:

A =78 (X'4E')
HL=*pointer to buffer containing filespec or devspec*
DE=*pointer to 32-byte FCB or DCB*

Exit Conditions:

Success always.
If the Z flag is set, the file specification is valid.
HL=*pointer to terminating character*
DE=*pointer to start of FCB*
If the NZ flag is set, a syntax error was found in the filespec.
HL=*pointer to invalid character*
DE=*pointer to start of FCB*
A =*invalid character*

General:

AF and BC are altered.

Example:

See Sample Program C, lines 53-65.

Get One Byte From Device or File

Gets a byte from a logical device or a file. The DCB TYPE byte (DCB + Ø, Bit Ø) must permit a GET operation for this call to be successful.

Entry Conditions:

A = 3 (X'Ø3')
DE = *pointer to DCB or FCB*

Exit Conditions:

Success, Z flag set.
A = *character read from the device or file*
Failure, NZ flag set. Test register A:
If A = Ø, no character was available.
If A ≠ Ø, A contains *error number.*

Example:

See the section "Device Driver and Filter Templates."

@GTDCB

SVC Number 82

Get Device Control Block Address

Finds the location of a Device Control Block (DCB). If DE = Ø (no device name specified), HL returns the address of the first unused DCB found.

Entry Conditions:

A = 82 (X'52')
DE = *2-character device name* (E = first character, D = second character)

Exit Conditions:

Success, Z flag set. DCB was found.
HL = *pointer to start of DCB*
Failure, NZ flag set. No DCB was available.
A = *Error 8 (Device not available)*
HL is altered.

General:

AF is always altered by this SVC.

Example:

See the section "Device Driver and Filter Templates."

Get Drive Code Table Address

Gets the address of the Drive Code Table for the requested drive.

Entry Conditions:

A = 81 (X'51')
C = *logical drive number* (0-7)

Exit Conditions:

Success always.
IY = *pointer to the DCT entry for the specified drive*
AF is always altered by this SVC.

General:

If the drive number is out of range, the IY pointer will be invalid. This call does not return Z/NZ to indicate if the drive number specified is valid (0-7) or enabled.

Example:

See the example for @DCSTAT in Sample Program D, lines 27-33.

Get Memory Module Address

Locates a memory module, if the standard memory header is at the start of the module. The scanning starts with the system drivers in low memory, then moves to any high memory modules. If any routine is encountered that does not start with a proper header, scanning stops.

Entry Conditions:

A = 83 (X'53')

DE = *pointer to memory module name in upper case, terminated with any character in the range 00-31*

Exit Conditions:

Success always.

If the Z flag is set, the module was found.

HL = *pointer to first byte of memory header*

DE = *pointer to first byte after module name*

If the NZ flag is set, the module was not found.

HL is altered.

General:

AF is always altered by this SVC.

Example:

See Sample Program F, lines 144-154.

@HDFMT

SVC Number 52

Hard Disk Format

Passes a format drive command to a hard disk driver. If the hard disk controller accepts it as a valid command, then it formats the entire disk drive. If the hard disk controller does not accept it, then an error is returned. Radio Shack hardware does not currently support @HDFMT.

Entry Conditions:

A = 52 (X'34')
C = *logical drive number* (Ø-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

@HEXDEC SVC Number 97

Convert Binary to Decimal ASCII

Converts a binary number in HL to decimal ASCII.

Entry Conditions:

A = 97 (X'61')
HL = *number to convert*
DE = *pointer to 5-character buffer to hold converted number*

Exit Conditions:

Success always.
DE = *pointer to end of buffer + 1*
AF, BC, and HL are altered by this SVC.

Example:

See Sample Program B, lines 73-76.

Convert 1 Byte to Hex ASCII

Converts a 1-byte number to hexadecimal ASCII.

Entry Conditions:

A = 98 (X'62')
C = *number to convert*
HL = *pointer to a 2-character buffer to hold the converted number*

Exit Conditions:

Success always.
HL = *pointer to the end of buffer + 1*
Only AF is altered by this SVC.

Example:

See Sample Program B, lines 236-246.

Convert 2 Bytes to Hex ASCII

Converts a 2-byte number to hexadecimal ASCII.

Entry Conditions:

A = 99 (X'63')
DE = *number to convert*
HL = *pointer to 4-character buffer to hold converted number*

Exit Conditions:

Success always.
HL = *pointer to end of buffer + 1*
Only AF is altered by this SVC.

Example:

See Sample Program B, lines 248-258.

Get or Alter HIGH$ or LOW$

Provides the means to read or alter the HIGH$ and LOW$ values.

Note: HIGH$ must be greater than LOW$. LOW$ is reset to X'2FFF' by @EXIT, @ABORT, and @CMNDI.

Entry Conditions:

A = 100 (X'64')
B selects HIGH$ or LOW$
If B = 0, SVC deals with HIGH$
If B ≠ 0, SVC deals with LOW$
HL selects one of the following functions:
If HL = 0, the current HIGH$ or LOW$ is returned
If HL ≠ 0, then HIGH$ or LOW$ is set to the value in HL

Exit Conditions:

Success, Z flag set.
HL = *current HIGH$ or LOW$*. If HL ≠ 0 on entry, then HIGH$ or LOW$ is now set to that value.
Failure, NZ flag set.
A = *error number*

General:

If bit 0 of CFLAG$ is set (see @FLAGS), then HIGH$ cannot be changed with this call. The call returns error 43, "SVC parameter error."

Example:

See Sample Program F, lines 75-86.

Open or Initialize File

Opens a file. If the file is not found, this SVC creates it according to the file specification.

Entry Conditions:

A =58 (X'3A')
HL=*pointer to 256-byte disk I/O buffer*
DE=*pointer to FCB containing the file specification*
B =*Logical Record Length to be used while file is open*

Exit Conditions:

Success, Z flag set. File was opened or created.
The CF flag is set if a new file was created.
Failure, NZ flag set.
A=*error number*

General:

Only AF is altered by this SVC.
The file open bit is set in the directory if the access level is UPDATE or greater.

Example:

See Sample Program C, lines 260-272.

Reboot the System

Does a software reset. Floppy drive 0 must contain a system disk. @IPL uses the standard boot sequence, the same as for a hard reset (pressing the reset button). Memory locations X'41E5'-X'4225' and X'4300'-X'43FF' are altered during the boot of the machine.

Entry Conditions:
A = 0 (X'00')

General:
This SVC does not return.

Scan Keyboard and Return

Scans the keyboard and returns a character if a key is pressed. If no key is pressed, a zero value is returned.

Entry Conditions:
A = 8 (X'Ø8')

Exit Conditions:
Success, Z flag set.
A = *character pressed*
Failure, NZ set.
If A = Ø, no character was available.
If A ≠ Ø, then A contains *error number.*

General:
DE is altered by this SVC.

Example:
See Sample Program C, lines 198-2ØØ.

Scan *KI Device, Wait for Character

Scans the *KI device and returns with a character. It does not return until a character is input to the device.

Note: The system suspends execution of the program that issued the SVC until a character can be obtained. Background tasks will continue to run normally.

Entry Conditions:
A = 1 (X'01')

Exit Conditions:
Success, Z flag set.
A = *character entered*
Failure, NZ flag set.
A = *error number*

General:
DE is altered by this SVC.

Example:
See Sample Program B, lines 202-203.

@KEYIN

SVC Number 9

Accept a Line of Input

Accepts a line of input until terminated by either an (ENTER) or a (BREAK). Entries are displayed on the screen, starting at the current cursor position. Backspace, tab, and line delete are supported. If JCL is active, the line is fetched from the active JCL file.

Entry Conditions:

A = 9 (X'Ø9')
HL = *pointer to user line buffer of length B + 1*
B = *maximum number of characters to input*
C = Ø

Exit Conditions:

Success, Z flag set.
HL = *pointer to start of buffer*
B = *actual number of characters input*
CF is set if (BREAK) terminated the input.
Failure, NZ flag set.
A = *error number*

General:

DE and C are altered by this SVC.

Example:

See Sample Program C, lines 39-47.

Remove Currently Executing Task

When called by an executing task driver, removes the task assignment from the task table and returns to the foreground application that was interrupted.

Entry Conditions:
A = 32 (X'20')

General:
This SVC does not return.

Example:
See the example for @RMTSK in Sample Program F, lines 134-142.

@LOAD

SVC Number 76

Load Program File

Loads a program file. The file must be in load module format.

Entry Conditions:

A =76 (X'4C')
DE = *pointer to FCB containing filespec of the file to load*

Exit Conditions:

Success, Z flag set.
HL = *transfer address retrieved from file*
Failure, NZ flag set.
A = *error number*

Example:

See Sample Program A, lines 50-56.

Calculate Current Logical Record Number

Returns the current logical record number.

Entry Conditions:

A = 63 (X'3F')
DE = pointer to the file's FCB

Exit Conditions:

Success, Z flag set.
BC = *logical record number*
Failure, NZ flag set.
A = *error number*

General:

AF is altered by this SVC.

Example:

See Sample Program C, lines 305-311.

@LOF SVC Number 64

Calculate the EOF Logical Record Number

Returns the EOF (End of File) logical record number.

Entry Conditions:

A = 64 (X'40')
DE = *pointer to FCB for the file to check*

Exit Conditions:

Success, Z flag set.
BC = *the EOF logical record number*
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Issue Log Message

Issues a log message to the Job Log. The message can be any character string terminating with a carriage return (X'0D').

Entry Conditions:

A = 11 (X'0B')
HL = *pointer to first character in message line*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

```
        LD      HL,TEXT     ;Point at message to output
        LD      A,@LOGER    ;and output it to the Job
                            ;Log
        RST     28H         ;Call the @LOGER SVC
        ...
TEXT:   DEFM    'This is a message for the Job Log'
        DEFB    0DH         ;Message must be terminated
                            ;with an <ENTER>.
```

Display and Log Message

Displays and logs a message. Performs the same function as @DSPLY followed by @LOGER.

Entry Conditions:

A = 12 (X'0C')
HL = *pointer to first character in message line*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.
To avoid a looping condition that could result from the display device generating an error, no error checking should be done after returning from @LOGOT.

Example:

```
        LD      HL,TEXT     ;Point at message to output
        LD      A,@LOGOT    ;and output it to the Job
                            ;Log AND the display
        RST     28H         ;Call the @LOGOT SVC
        ...
TEXT:   DEFM    'This message will be displayed both in'
        DEFM    'the Job Log and on the display.'
        DEFB    0DH         ;Must terminate text with an
                            ;<ENTER>.
```

Send Message to Device

Sends a message line to any device or file.

Entry Conditions:

A = 13 (X'0D')
DE = *pointer to DCB or FCB of device or file to receive output*
HL = *pointer to message line terminated with X'0D' or X'03'*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

```
        LD      HL,TEXT     ;Point at message to output
        LD      DE,DCBP     ;Point at the device control
                            ;block for our device
        LD      A,@MSG      ;and write this text to it
        RST     28H         ;Call the @MSG SVC
        ...
TEXT:   DEFM    'D555-555<LOGIN USER>' ;Text to write to
                            ;this device. In this case,
                            ;it is a dialing modem.
        DEFB    03H         ;Terminate the message
```

@MUL8 SVC Number 90

8-Bit Multiplication

Performs an 8-bit by 8-bit unsigned integer multiplication. The resultant product must fit into an 8-bit field.

Entry Conditions:

A = 90 (X'5A')
C = *multiplicand*
E = *multiplier*

Exit Conditions:

Success always.
A = *product*
DE is altered by this SVC.

Example:

See Sample Program B, lines 150-153.

@MUL16 SVC Number 91

16-Bit by 8-Bit Multiplication

Performs an unsigned integer multiplication of a 16-bit multiplicand by an 8-bit multiplier. The resultant product is stored in a 3-byte register field.

Entry Conditions:

A =91 (X'5B')
HL=*multiplicand*
C =*multiplier*

Exit Conditions:

Success always.
HL=*two high-order bytes of product*
A =*low-order byte of product*
DE is altered by this SVC.

Example:

See Sample Program B, lines 183-187.

@OPEN SVC Number 59

Open Existing File or Device

Opens an existing file or device.

Entry Conditions:

A =59 (X'3B')
HL=*pointer to 256-byte disk I/O buffer*
DE=*pointer to FCB or DCB containing filespec or devspec*
B =*logical record length for open file*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A=*error number*

General:

AF is altered by this SVC.
The file open bit is set in the directory if the access level is UPDATE or greater.

Example:

See Sample Program C, lines 134-150.

Parse Parameter String

Parses an optional parameter string. Its primary function is to parse command parameters contained in a command line starting with a parenthesis. The acceptable parameter format is:

PARM = X'nnnn'....hexadecimal entry
PARM = nnnnn ....decimal entry
PARM = "string" ...alphanumeric entry
PARM = flag ....ON, OFF, Y, N, YES, or NO

Note: Entering a parameter with no equal sign or value is the same as using PARM = ON. Entering PARM = with no value is the same as using PARM = OFF.

Entry Conditions:

A = 17 (X'11')
DE = *pointer to beginning of your parameter table*
HL = *pointer to command line to parse* (the parameter string is enclosed within parentheses)

Exit Conditions:

Success always.
If Z is set, either valid parameters or no parameters were found.
If NZ is set, a bad parameter was found.

General:

NZ is not returned if parameter types other than those specified are entered. The application must check the validity of the response byte.

The valid parameters are contained in a user table which must be in one of the following formats. (Parameter names must consist of alphanumeric characters, the first of which is a letter.)

For use with TRSDOS Version 6, use this format:

The parameter table starts with a single byte X'80'. Each parameter is stored in a variable length field as described below.

1) Type Byte (Type and length byte)
   Bit 7 — If set, accept numeric value
   Bit 6 — If set, accept flag parameter
   Bit 5 — If set, accept "string" value
   Bit 4 — If set, accept first character of name as abbreviation
   Bits 3-0 — Length of parameter name

2) Actual Parameter Name

3) Response byte (Type and length found)
   Bit 7 — Numeric value found
   Bit 6 — Flag parameter found
   Bit 5 — String parameter found
   Bits 4-0 — Length of parameter entered. If length is 0 and the 2-byte vector points to a quotation mark (X'22'), then the parameter was a null string. Otherwise, a length of 0 indicates that the parameter was longer than 31 characters.

4) 2-byte address vector to receive the parsed parameter values.

The 2-byte memory area pointed to by the address field of your table receives the value of PARM if PARM is non-string. If a string is entered, the 2-byte memory area receives the address of the first byte of "string." The entries ON, YES, and Y return a value of X'FFFF'; OFF, NO, and N return X'0000'. If a parameter name is specified on the command line and is fol-

lowed by an equal sign and no value, then X'0000' or NO is returned. If a parameter name is used on the command line without the equal sign, then a value of X'FFFF' or ON is assumed. For any allowed parameter that is completely omitted on the command line, the 2-byte area remains unchanged and the response byte is 0.

The parameter table is terminated with a single byte X'00'.

For compatibility with LDOS 5.1.3, use this format:

A 6-character "word" left justified and padded with blanks followed by a 2-byte address to receive the parsed values. Repeat word and address for as many parameters as are necessary. You must place a byte of X'00' at the end of the table.

Example:

```
          LD    HL,COMAND  ;Point at command buffer
          LD    DE,PARM    ;Point at parameter list
          LD    A,@PARAM   ;Parse the items on the
                           ;command line
          RST   28H        ;Call the @PARAM SVC
          JR    NZ,ERROR   ;An error occurred (not
                           ;included here)
          LD    A,(RESP)   ;Get response code
          AND   040H       ;Test response flags
          JR    Z,BAD      ;User specified something
                           ;like UPDATE=X'1234' or
                           ;UPDATE="HELLO"
          LD    A,(VAL)    ;Get 1st byte of VAL word
          OR    A          ;Test the value
          JR    Z,OFF      ;UPDATE=OFF or UPDATE=NO was
                           ;specified
          JR    ON         ;UPDATE=ON or UPDATE=YES was
                           ;specified
          ...
COMAND:   DEFS  80         ;Area where command is
                           ;stored
PARM:     DEFB  80H        ;Table header code
          DEFB  40H+6      ;40 says we want a flag
                           ;(YES/NO), 6 is length of
                           ;the parameter name
          DEFM  'UPDATE'   ;Parameter name
RESP:     DEFB  0          ;Response area
          DEFW  VAL        ;Vector to VAL
          DEFB  0          ;End of Table code
VAL:      DEFS  2          ;Area to receive a parameter
                           ;value
```

@PAUSE

SVC Number 16

Suspend Program Execution

Suspends program execution for a specified period of time and goes into a "holding" state. The delay is at least 14.3 microseconds per count.

Entry Conditions:

A = 16 (X'10')
BC = *delay count*

Exit Conditions:

Success always.

Example:

```
LD    BC,36A2H   ;Wait for about 200 milli-
                 ;seconds, 14.3 usecs *
                 ;13986 is approx. 200
                 ;msecs
LD    A,@PAUSE   ;Suspend execution
RST   28H        ;Call the @PAUSE SVC
```

@PEOF

SVC Number 65

Position to End Of File

Positions an open file to the End Record Number (ERN). An end-of-file-encountered error (X'1C') is returned if the operation is successful. Your program may ignore this error.

Entry Conditions:

A = 65 (X'41')
DE = *pointer to FCB of the file to position*

Exit Conditions:

NZ flag always set.
If A = X'1C', then success.
If A ≠ X'1C', then failure.
A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

@POSN

SVC Number 66

Position File

Positions a file to a logical record. This is useful for positioning to records of a random access file.

When the @POSN routine is used, Bit 6 of FCB + 1 is automatically set. This ensures that the EOF (End Of File) is updated when the file is closed only if the NRN (Next Record Number) exceeds the current ERN (End Record Number).

Note that @POSN must be used for *each* write, even if two records are side by side.

Entry Conditions:

A = 66 (X'42')
DE = *pointer to FCB for the file to position*
BC = *the logical record number*

Exit Conditions:

If Z flag is set or A = X'1C' or X'1D', then success.
The file was positioned.
Otherwise, failure.
A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

@PRINT SVC Number 14

Prints Message Line

Outputs a message line to the printer. The line must be terminated with either a carriage return (X'0D') or an ETX (X'03').

Entry Conditions:

A = 14 (X'0E')
HL = *pointer to message to be output*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

AF and DE are altered by this SVC.

Example:

```
        LD      HL,TEXT     ;Text to be output to the
                            ;printer
        LD      A,@PRINT    ;Write this message to the
                            ;printer device
        RST     28H         ;Call the @PRINT SVC
        ...
TEXT:   DEFB    0CH         ;Do a Top of Form
        DEFM    'Report continued                    Page '
        DEFB    3           ;Terminate with a <ETX> or
                            ;an <ENTER>
```

Send Character to Printer

Outputs a byte to the line printer.

Entry Conditions:
A = 6 (X'06')
C = *character to print*

Exit Conditions:
Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:
AF and DE are altered by this SVC.
If the line printer is attached but becomes unavailable (out of paper, out of ribbon, turned off, off-line, buffer full, etc.), the printer driver waits approximately ten seconds. If the printer is still not ready, a "Device not available" error is returned.

Example:

```
        LD      A,(PAGE)   ;Get the page number
        ADD     A,'0'      ;Make it ASCII
        LD      C,A        ;Put the value here
        LD      A,@PRT     ;Write this character to the
                           ;printer
        RST     28H        ;Call the @PRT SVC
        ...
PAGE:   DEFB    2          ;Start with page 2
```

Write One Byte to Device or File

Outputs a byte to a logical device or file. The DCB TYPE byte (DCB+Ø, Bit 1) must permit PUT operation.

Entry Conditions:

A =4 (X'Ø4')
DE=*pointer to DCB or FCB of the output device*
C =*byte to output*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A=*error number*

General:

AF is always altered by this SVC.

Example:

See the section "Device Driver and Filter Templates."

@RAMDIR SVC Number 35

Get Directory Record or Free Space

Reads the directory information of visible files from a disk directory, or gets the amount of free space on a disk.

Entry Conditions:

A = 35 (X'23')
HL = *pointer to RAM buffer to receive information*
B = *logical drive number* (0-7)
C selects one of the following functions:
If C = 0, get directory records of all visible files.
If C = 255, get free space information.
If C = 1-254, get a single directory record (see below).

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Each directory record requires 22 bytes of space in the buffer. If C = 0, one additional byte is needed to mark the end of the buffer.

For single directory records, the number in the C register should be one less than the desired directory record. For example, if C = 1, directory record 2 is fetched and put in the buffer. If a single record request is for an inactive record or an invisible file, the A register returns an error code 25 (File access denied).

The directory information is placed in the buffer as follows:

Byte	Contents
00-14	filename/ext:d (left justified, padded with spaces)
15	protection level, 0 to 6
16	EOF offset byte
17	logical record length, 0 to 255
18-19	ERN of file
20-21	file size in K (1024-byte blocks)
22	LAST RECORD ONLY. Contains "+" to mark buffer end.

If C = 255, HL should point to a 4-byte buffer. Upon return, the buffer contains:

Bytes 00-01	Space in use in K, stored LSB, MSB
Bytes 02-03	Space available in K, stored LSB, MSB

Example:

See the example for @DODIR in Sample Program E, lines 32-40.

@RDHDR SVC Number 48

Read a Sector Header

Reads the next ID header when supported by the controller driver. The floppy disk driver supplied treats this as a @RDSEC (SVC 49).

Entry Conditions:

A = 48 (X'30')
HL = *pointer to buffer to receive the data*
D = *cylinder to read*
C = *logical drive number*
E = *sector to read*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:

See the example for @RDSEC in Sample Program D, lines 63-66.

@RDSEC

SVC Number 49

Read Sector

Transfers a sector of data from the disk to your buffer.

Entry Conditions:

A = 49 (X'31')
HL = *pointer to the buffer to receive the sector*
D = *cylinder to read*
E = *sector to read*
C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC

Example:

See Sample Program D, lines 63-66.

Read System Sector

Reads the specified system (directory) sector. If the cylinder number in register D is not the directory cylinder, the value in D is changed to reflect the real directory cylinder and the sector is then read.

Entry Conditions:

A = 85 (X'55')
HL = *pointer to the buffer to receive the sector*
D = *cylinder to read*
E = *sector to read*
C = *logical drive number* (Ø-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 78-92.

@RDTRK

SVC Number 51

Read a Track

Reads an entire track when supported by the controller driver. The floppy disk driver supplied treats this as a @RDSEC (SVC 49) and does not do a track read.

Entry Conditions:

A = 51 (X'33')
HL = *pointer to buffer to receive the sector*
D = *track to read*
C = *logical drive number*
E = *sector to read*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

AF is altered by the supplied floppy disk driver.

Example:

See the example for @RDSEC in Sample Program D, lines 63-66.

Read a Record

Reads a logical record from a file. If the LRL defined at open time was 256 (specified by Ø), then the NRN sector is transferred to the buffer established at open time. For LRL between 1 and 255, the next logical record is placed into a user record buffer, UREC. The 3-byte NRN is updated after the read operation.

Entry Conditions:

A = 67 (X'43')
DE = *pointer to FCB for the file to read*
HL = *pointer to user record buffer UREC* (needed if LRL = 1-255; unused if LRL = 256)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:

See Sample Program C, lines 3ØØ-3Ø4.

@REMOV SVC Number 57

Remove File or Device

Removes a file or device.

If a file is to be removed, the File Control Block must be in an open condition. When this SVC is performed, the file's directory is updated and the space occupied by the file is deallocated.

If a device was specified, the device is closed. To remove a device, use the REMOVE library command.

Entry Conditions:

A = 57 (X'39')
DE = *pointer to FCB or DCB to remove*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:

See Sample Program C, lines 223-231.

@RENAM SVC Number 56

Rename File or Device

Changes a file's filename and/or extension.

Entry Conditions:

A =56 (X'38')

DE=*pointer to an FCB containing the file's current name*
This FCB must be in a closed state.

HL=*pointer to new filename string terminated with a X'0D' or X'03'.* This filespec must be in upper case and must be a valid filespec. You can convert the filespec to upper case and check its validity by using the @FSPEC SVC before using @RENAM.

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A=*error number*

General:

After the call is completed, the FCB pointed to by DE is altered.
Only AF is altered by this SVC.

Example:

```
      LD    DE,FCB          ;Point at a closed FCB
                            ;containing the old
                            ;filespec
      LD    HL,NEW          ;Point to the new filespec
                            ;to use
      LD    A,@RENAM        ;Change the name of the
                            ;file
      RST   28H             ;Call the @RENAM SVC
      ...
FCB:  DEFS  32              ;A File Control Block used
                            ;by the @RENAM SVC. In
                            ;this example, it is
                            ;assumed that an @FSPEC
                            ;SVC has loaded a filespec
                            ;into the FCB before the
                            ;@RENAM SVC is performed.
NEW:  DEFM  'NEWNAME/TXT'   ;The new filespec for the
                            ;file
      DEFB  0DH             ;Terminate the filespec
```

Rewind File to Beginning

Rewinds a file to its beginning and resets the 3-byte NRN to Ø. The next record to be read or written sequentially is the first record of the file.

Entry Conditions:

A = 68 (X'44')
DE = *pointer to FCB for the file to rewind*

Exit Conditions:

Success, Z flag set. File positioned to record number Ø.
Failure, NZ flag set.
A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 3Ø5-311.

@RMTSK SVC Number 30

Remove Interrupt Level Task

Removes an interrupt level task from the Task Control Block table.

Entry Conditions:
A = 30 (X'1E')
C = *task slot assignment to remove* (0-11)

Exit Conditions:
Success always.
HL and DE are altered by this SVC.

Example:
See Sample Program F, lines 134-142.

Replace Task Vector

Exits the task process executing and replaces the currently executing task's vector address in the Task Control Block table with the address following the SVC instruction. Return is made to the foreground application that was interrupted.

Entry Conditions:
A=31 (X'1F')

General:
This SVC does not return.

Example:

```
        LD    A,RPTSK  ;Replace this task with the
                       ;one located at the
                       ;following address:
        RST   28H      ;Call the @RPTSK SVC
NEWADD: DEFW  0        ;Address of the new task is
                       ;loaded here. This word
                       ;must be immediately after
                       ;the @RPTSK SVC. The label
                       ;NEWADD is present only to
                       ;allow the address to be
                       ;stored.
```

@RREAD

SVC Number 69

Reread Sector

Forces a reread of the current sector to occur before the next I/O request is performed. Its most probable use is in applications that reuse the disk I/O buffer for multiple files, to make sure that the buffer contains the proper file sector. This routine is valid only for byte I/O or blocked files. Do not use it when positioned at the start of a file.

Entry Conditions:

A = 69 (X'45')
DE = *pointer to FCB for the file to reread*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

AF is always altered by this SVC.

Example:

```
LD      DE,FCB      ;Point to File Control Block
                    ;of the file that requires
                    ;the re-read
LD      A,@RREAD    ;Before next I/O, reload
                    ;the current sector into
                    ;the system buffer for
                    ;this file
RST     28H         ;Call the @RREAD SVC
```

Test for Drive Busy

Performs a test of the last selected drive to see if it is in a busy state. If busy, it is re-selected until it is no longer busy.

Entry Conditions:

A = 47 (X'2F')
C = *logical drive number* (0-7)

Exit Conditions:

Success always.
Only AF is altered by this SVC.

Example:

```
LD    C,1        ;Test Drive 1 to see if it
                 ;is busy
LD    A,@RSLCT   ;If it is, continue
                 ;selecting it
RST   28H        ;Call the @RSLCT SVC
```

Issue FDC RESTORE Command

Issues a disk controller RESTORE command.

Entry Conditions:
A = 44 (X'2C')
C = *logical drive number* (0-7)

Exit Conditions:
Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:
See the example for @CKDRV in Sample Program D, lines 38-39.

Run Program

Loads and executes a program file. If an error occurs during the load, the system prints the appropriate message and returns.

Entry Conditions:

A = 77 (X'4D')
DE = *pointer to FCB containing the filespec of the file to RUN*
Note: The FCB must be located where the program being loaded will not overwrite it.

Exit Conditions:

Success, the new program is loaded and executed.
Failure, the error is displayed and return is made to your program.
HL = *return code* (See the section "Converting to TRSDOS Version 6" for information on return codes.)

General:

HL is returned unchanged if no error occurred and can be used as a pointer to a command line.

Example:

See Sample Program A, lines 62-74.

Rewrite Sector

Rewrites the current sector, following a write operation. The @WRITE function advances the NRN after the sector is written. @RWRIT decrements the NRN and writes the disk buffer again. Do not use @RWRIT when positioned to the start of a file.

Entry Conditions:

A = 70 (X'46')
DE = *pointer to FCB for the file to rewrite*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:

```
LD      DE,FCB      ;Point to the File Control
                    ;Block
LD      A,@RWRIT    ;Perform a re-write of the
                    ;current sector
RST     28H         ;Call the @RWRIT SVC
```

Seek a Cylinder

Seeks a specified cylinder and sector. @SEEK does not return an error if you specified a non-existent drive or an invalid cylinder. @SEEK performs no action if the specified drive is a hard disk.

Note: Seek of a sector is not supported by TRS-80 hardware. An implied seek is included in sector reads and writes.

Entry Conditions:

A = 46 (X'2E')
C = *logical drive number*
D = *cylinder to seek*
E = *sector to seek*

Exit Conditions:

Success always.
Only AF is altered by this SVC.

@SEEKSC

SVC Number 71

Seek Cylinder and Sector

Seeks the cylinder and sector corresponding to the next record of the specified file. (This is done by examining the NRN field of the FCB.) No error is returned on physical seek errors.

Entry Conditions:

A = 71 (X'47')
DE = *pointer to the file's FCB*

Exit Conditions:

Success always.

Example:

```
        LD      DE,FCB      ;Point to the File Control
                            ;Block
        LD      A,@SEEKSC   ;Cause the next sector to be
                            ;SEEKed before it is
                            ;actually needed
        RST     28H         ;Call the @SEEKSC SVC
```

Skip a Record

Causes a skip past the next logical record. Only the record number contained in the FCB is changed; no physical I/O takes place.

Entry Conditions:

A =72 (X'48')
DE=*pointer to FCB for the file to skip*

Exit Conditions:

If the Z flag is set or if A=X'1C' or X'1D', then the operation was successful. Otherwise, A=*error number*. If A=X'1C' is returned, the file pointer is positioned at the end of the file. Any Appending operations would be performed here. If A=X'1D' is returned, the file pointer is positioned beyond the end of the file.

General:

AF is altered by this SVC.
BC contains the current record number. This is the same value as that returned by the @LOC SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Select a New Drive

Selects a drive. The time delay specified in your configuration (SYSTEM (DELAY = Y/N)) is made if the drive selection requires it.

Entry Conditions:

A = 41 (X'29')
C = *logical drive number* (Ø-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

@SOUND

SVC Number 104

Sound Generation

Generates sound using specified tone and duration codes. Interrupts are disabled during execution.

Entry Conditions:

A = 104 (X'68')
B = *function code*
bits 0-2: tone selection (0-7 with 0 = highest and 7 = lowest)
bits 3-7: tone duration (0-31 with 0 = shortest and 31 = longest)

Exit Conditions:

Success always.
Only AF is altered by this SVC.

Example:

See Sample Program B, lines 43-45.

Issue FDC STEP IN Command

Issues a disk controller STEP IN command. This moves the drive head to the next higher-numbered cylinder. @STEPI is intended for sequential read/write operations, such as disk formatting.

Entry Conditions:

A = 45 (X'2D')
C = *logical drive number*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Get Time

Gets the system time in display format (HH:MM:SS).

Entry Conditions:

A = 19 (X'13')
HL = *pointer to buffer to receive the time string*

Exit Conditions:

Success always.
HL = *pointer to the end of buffer + 1*
DE = *pointer to start of TIME$ storage area in TRSDOS*
AF and BC are altered by this SVC.

Example:

See the example for @DATE in Sample Program F, lines 252-253.

Video Functions

Performs various functions related to the video display. The B register is used to pass the function number.

Entry Conditions:

A = 15 (X'0F')

B selects one of the following functions:

If B = 1, return the character at the screen position specified by HL.
H = *row on the screen* (0-23), where 0 is the top row
L = *column on the screen* (0-79), where 0 is the leftmost column

If B = 2, display the specified character at the position specified by HL.
C = *character to be displayed*
H = *row on the screen* (0-23), where 0 is the top row
L = *column on the screen* (0-79), where 0 is the leftmost column

If B = 3, move the cursor to the position specified by HL. This is done even if the cursor is not currently displayed.
H = *row on the screen* (0-23), where 0 is the top row
L = *column on the screen* (0-79), where 0 is the leftmost column

If B = 4, return the current position of the cursor.

If B = 5, move a 2048-byte (2K) block of data to video memory. Only 1920 bytes of this are displayed, but a 2K area should be maintained if you plan to read information back from the video using the same area of memory.
HL = *pointer to 2K area to be moved to video memory*
HL must be in the range X'23FF' < HL < X'EC01'.

If B = 6, move a 2048-byte (2K) block of data from video memory to a buffer you supply. The first 1920 bytes contain the data that is on the screen in 80x24 display mode. The remaining bytes are the remainder of the video RAM area.

If B = 7, scroll protect the specified number of lines from the top of the screen.
C = *number of lines to scroll protect* (0-7). Once set, scroll protect can be removed only by executing @VDCTL with B = 7 and C = 0, or by resetting the system. Clearing the screen with (SHIFT)(CLEAR) erases the data in the scroll protect area, but the scroll protect still exists.

If B = 8, change cursor character to specified character. If the cursor is currently not displayed, the character is accepted anyway and is used as the cursor character when it is turned back on.
C = *character to use as the cursor character*

Exit Conditions:

If B = 1:
Success, Z flag set.
A = *character found at the location specified by HL*
DE is altered.
Failure, NZ flag set.
A = *error number*

If B = 2:
Success, Z flag set.
DE is altered.
Failure, NZ flag set.
A = *error number*

If B = 3:
Success, Z flag set.
DE and HL are altered.
Failure, NZ flag set.
A = *error number*

If B = 4:
Success always.
HL = *row and column position of the cursor.* H = row on the screen (Ø-23), where Ø is the top row; L = column on the screen (Ø-79), where Ø is the leftmost column.

If B = 5:
Success always.
HL = *pointer to the last byte moved to the video + 1*
BC and DE are altered.

If B = 6:
Success always.
BC, DE, and HL are altered.

If B = 7:
Success always.
BC and DE are altered.

If B = 8:
Success always.
A = *previous cursor character*
DE is altered.

General:

Functions 5, 6, and 7 do not do range checking on the entry parameters.
If HL is not in the valid range in functions 5 and 6, the results may be unpredictable.
Only function 3 (B = 3) moves the cursor.
If C is greater than 7 in function 7, it is treated as modulo 8.
AF and B are altered by this SVC.

Example:

See Sample Program F, lines 3Ø4-327.

Write and Verify a Record

Performs a @WRITE operation followed by a test read of the sector (if the write required physical I/O) to verify that it is readable.

If the logical record length is less than 256, then the logical record in the user buffer UREC is transferred to the file. If the LRL is equal to 256, a full sector write is made using the disk I/O buffer identified at file open time.

Entry Conditions:

A = 73 (X'49')
DE = *pointer to FCB for the file to verify*

Exit Conditions:

Success, Z flag set.
HL = *pointer to user buffer containing the logical record*
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program C, lines 338-346.

Verify Sector

Verifies a sector without transferring any data from disk.

Entry Conditions:

A = 50 (X'32')
D = *cylinder to verify*
E = *sector to verify*
C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set
A = *error number*

General:

AF is always altered by this SVC.
If the sector is a system sector, the sector is readable if an error 6 is returned; any other error number signifies an error has occurred.

Example:

See the example for @WRSEC in Sample Program D, lines 68-76.

Write End Of File

Forces the system to update the directory entry with the current end-of-file information.

Entry Conditions:

A = 74 (X'4A')
DE = *pointer to the FCB for the file to WEOF*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

AF is always altered by this SVC.

Example:

```
        LD      DE,FCB      ;Point at the File Control
                            ;Block
        LD      A,@WEOF     ;Force the directory entry
                            ;to be updated now,
                            ;instead of when the file
                            ;is closed
        RST     28H         ;Call the @WEOF SVC
```

@WHERE

SVC Number 7

Locate Origin of SVC

Used to resolve the relocation address of the calling routine.

Entry Conditions:
A = 7 (X'07')

Exit Conditions:
Success always.
HL = *pointer to address following RST 28H instruction*
AF is always altered by this SVC.

Example:
See Sample Program F, lines 36-60.

Write a Record

Causes a write to the next record identified in the File Control Block.

If the logical record length is less than 256, then the logical record in the user buffer UREC is transferred to the file. If the LRL is equal to 256, a full sector write is made using the disk I/O buffer identified at file open time.

Entry Conditions:

A =75 (X'4B')
HL=*pointer to user record buffer UREC* (unused if LRL=256)
DE=*pointer to FCB for the file to write*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A=*error number*

General:

AF is always altered by this SVC.

Example:

See the example for @VER in Sample Program C, lines 338-346.

Write a Sector

Writes a sector to the disk.

Entry Conditions:

A = 53 (X'35')
HL = *pointer to the buffer containing the sector of data*
D = *cylinder to write*
E = *sector to write*
C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 68-76.

Write a System Sector

Writes a system sector (used in directory cylinder).

Entry Conditions:

A = 54 (X'36')
HL = *pointer to the buffer containing the sector of data*
D = *cylinder to write*
E = *sector to write*
C = *logical drive number*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 94-104.

@WRTRK

SVC Number 55

Write a Track

Writes an entire track of properly formatted data. The data format must conform to that described in the disk controller's reference manual. @WRTRK must always be preceded by @SLCT.

Entry Conditions:

A = 55 (X'37')
HL = *pointer to format data*
D = *track to write*
C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Numerical List of SVCs

Following is a numerical list of the SVCs:

Dec	Hex	Label	Function
0	00	@IPL	Reboot the system
1	01	@KEY	Scan *KI device, wait for character
2	02	@DSP	Display character at cursor, advance cursor
3	03	@GET	Get one byte from a logical device
4	04	@PUT	Write one byte to a logical device
5	05	@CTL	Make a control request to a logical device
6	06	@PRT	Send character to the line printer
7	07	@WHERE	Locate origin of CALL
8	08	@KBD	Scan keyboard and return
9	09	@KEYIN	Accept a line of input
10	0A	@DSPLY	Display a message line
11	0B	@LOGER	Issue a log message
12	0C	@LOGOT	Display and log a message
13	0D	@MSG	Message line handler
14	0E	@PRINT	Print a message line
15	0F	@VDCTL	Position/locate cursor, get/put character at cursor
16	10	@PAUSE	Suspend program execution
17	11	@PARAM	Parse an optional parameter string
18	12	@DATE	Get system date in the format MM/DD/YY
19	13	@TIME	Get system time in the format HH:MM:SS
20	14	@CHNIO	Pass control to the next module in a device chain
21	15	@ABORT	Load HL with X'FFFF' error and goto @ EXIT
22	16	@EXIT	Exit program and return to TRSDOS
23			Reserved for future use
24	18	@CMNDI	Entry to command interpreter with return to the system
25	19	@CMNDR	Entry to command interpreter with return to the user
26	1A	@ERROR	Entry to post an error message
27	1B	@DEBUG	Enter DEBUG
28	1C	@CKTSK	Check if task slot in use
29	1D	@ADTSK	Add an interrupt level task
30	1E	@RMTSK	Remove an interrupt level task
31	1F	@RPTSK	Replace the currently executing task vector
32	20	@KLTSK	Remove the currently executing task
33	21	@CKDRV	Check for drive availability
34	22	@DODIR	Do a directory display/buffer
35	23	@RAMDIR	Get directory record(s) or free space into RAM
36-39			Reserved for future use
40	28	@DCSTAT	Test if drive is assigned in DCT
41	29	@SLCT	Select a new drive
42	2A	@DCINIT	Initialize the FDC
43	2B	@DCRES	Reset the FDC
44	2C	@RSTOR	Issue FDC RESTORE command
45	2D	@STEPI	Issue FDC STEP IN command

Dec	Hex	Label	Function
46	2E	@SEEK	Seek a cylinder
47	2F	@RSLCT	Test if requested drive is busy
48	30	@RDHDR	Read a sector header
49	31	@RDSEC	Read a sector
50	32	@VRSEC	Verify a sector
51	33	@RDTRK	Read a track
52	34	@HDFMT	Hard disk format
53	35	@WRSEC	Write a sector
54	36	@WRSSC	Write a system sector
55	37	@WRTRK	Write a track
56	38	@RENAM	Rename a file
57	39	@REMOV	Remove a file or device
58	3A	@INIT	Open or initialize a file or device
59	3B	@OPEN	Open an existing file or device
60	3C	@CLOSE	Close a file or device
61	3D	@BKSP	Backspace one logical record
62	3E	@CKEOF	Check for end of file
63	3F	@LOC	Calculate the current logical record number
64	40	@LOF	Calculate the EOF logical record number
65	41	@PEOF	Position to the end of file
66	42	@POSN	Position a file to a logical record
67	43	@READ	Read a record from a file
68	44	@REW	Rewind a file to its beginning
69	45	@RREAD	Reread the current sector
70	46	@RWRIT	Rewrite the current sector
71	47	@SEEKSC	Seek a specified cylinder and sector
72	48	@SKIP	Skip the next record
73	49	@VER	Write a record to a file and verify
74	4A	@WEOF	Write end of file
75	4B	@WRITE	Write a record to a file
76	4C	@LOAD	Load a program file
77	4D	@RUN	Load and execute a program file
78	4E	@FSPEC	Fetch a file or device specification
79	4F	@FEXT	Set up a default file extension
80	50	@FNAME	Fetch filename/extension from directory
81	51	@GTDCT	Get Drive Code Table address
82	52	@GTDCB	Find specified or first free DCB
83	53	@GTMOD	Find specified memory module address
84			Reserved for future use
85	55	@RDSSC	Read a system sector
86			Reserved for future use
87	57	@DIRRD	Read directory record
88	58	@DIRWR	Write directory record
89			Reserved for future use
90	5A	@MUL8	Multiply 8-bit unsigned integers
91	5B	@MUL16	Multiply 16-bit by 8-bit unsigned integers
92			Reserved for future use
93	5D	@DIV8	Divide 8-bit unsigned integers
94	5E	@DIV16	Divide 16-bit by 8-bit unsigned integers
95			Reserved for future use
96	60	@DECHEX	Convert decimal ASCII to 16-bit binary value
97	61	@HEXDEC	Convert a number in HL to decimal ASCII

Dec	Hex	Label	Function
98	62	@HEX8	Convert a 1-byte number to hex ASCII
99	63	@HEX16	Convert a 2-byte number to hex ASCII
100	64	@HIGH$	Obtain or set the highest and lowest unused RAM addresses
101	65	@FLAGS	Point IY to the system flag table
102	66	@BANK	Check, set, or reset a 32K bank of memory
103	67	@BREAK	Set user or system break vector
104	68	@SOUND	Generate sound (tone and duration)
105-127			Reserved for future use.

Alphabetical List of SVCs

Following is an alphabetical list of the SVC labels and numbers:

Label	Dec	Hex
@ABORT	21	15
@ADTSK	29	1D
@BANK	102	66
@BKSP	61	3D
@BREAK	103	67
@CHNIO	20	14
@CKDRV	33	21
@CKEOF	62	3E
@CKTSK	28	1C
@CLOSE	60	3C
@CMNDI	24	18
@CMNDR	25	19
@CTL	5	5
@DATE	18	12
@DCINIT	42	2A
@DCRES	43	2B
@DCSTAT	40	28
@DEBUG	27	1B
@DECHEX	96	60
@DIRRD	87	57
@DIRWR	88	58
@DIV8	93	5D
@DIV16	94	5E
@DODIR	34	22
@DSP	2	2
@DSPLY	10	0A
@ERROR	26	1A
@EXIT	22	16
@FEXT	79	4F
@FLAGS	101	65
@FNAME	80	50
@FSPEC	78	4E
@GET	3	3
@GTDCB	82	52
@GTDCT	81	51
@GTMOD	83	53
@HDFMT	52	34
@HEXDEC	97	61
@HEX8	98	62
@HEX16	99	63
@HIGH$	100	64
@INIT	58	3A
@IPL	0	0
@KBD	8	8
@KEY	1	1
@KEYIN	9	9
@KLTSK	32	20
@LOAD	76	4C
@LOC	63	3F
@LOF	64	40
@LOGER	11	0B
@LOGOT	12	0C
@MSG	13	0D

Label	Dec	Hex
@MUL8	90	5A
@MUL16	91	5B
@OPEN	59	3B
@PARAM	17	11
@PAUSE	16	10
@PEOF	65	41
@POSN	66	42
@PRINT	14	0E
@PRT	6	6
@PUT	4	4
@RAMDIR	35	23
@RDHDR	48	30
@RDSEC	49	31
@RDSSC	85	55
@RDTRK	51	33
@READ	67	43
@REMOV	57	39
@RENAM	56	38
@REW	68	44
@RMTSK	30	1E
@RPTSK	31	1F
@RREAD	69	45
@RSLCT	47	2F
@RSTOR	44	2C
@RUN	77	4D
@RWRIT	70	46
@SEEK	46	2E
@SEEKSC	71	47
@SKIP	72	48
@SLCT	41	29
@SOUND	104	68
@STEPI	45	2D
@TIME	19	13
@VDCTL	15	0F
@VER	73	49
@VRSEC	50	32
@WEOF	74	4A
@WHERE	7	7
@WRITE	75	4B
@WRSEC	53	35
@WRSSC	54	36
@WRTRK	55	37

Sample Programs

The following sample programs use many of the supervisor calls described in this manual. These programs are not meant to be examples of the most efficient programming, but are designed to illustrate as many supervisor calls as possible.

Sample Program A

```
Ln #                Source Line

ØØØØ1   ;           This program asks the user whether to run a program
ØØØØ2   ;           or debug it and executes the SVCs required to perform
ØØØØ3   ;           the chosen action.
ØØØØ4
ØØØØ5               PSECT   5ØØØH           ;The program begins at x'5ØØØ'
ØØØØ7
ØØØØ8   ;           Define the equates for the SVCs that will be used.
ØØØØ9
ØØØ1Ø   @DEBUG:     EQU     27              ;Enter the debugger  (DEBUG)
ØØØ11   @DSPLY:     EQU     1Ø              ;Display a message
ØØØ12   @FSPEC:     EQU     78              ;Verify a filespec or devspec and
ØØØ13                                       ;load it into a File Control Block
ØØØ14   @KEY:       EQU     1               ;Get a character from the keyboard
ØØØ15   @LOAD:      EQU     76              ;Load a program into memory
ØØØ16   @RUN:       EQU     77              ;Execute a program
ØØØ17
ØØØ18   MESS1:      DEFM    'Do you wish to RUN this Program or DEBUG it ?'
ØØØ19               DEFB    ØAH             ;This moves the cursor to the next line
ØØØ2Ø               DEFM    'Press <ENTER> to RUN or <BREAK> to DEBUG'
ØØØ21               DEFB    ØDH             ;Terminate the message string
ØØØ22
ØØØ23   PROGRM:     DEFM    'DIREX/CMD'     ;Sample program to debug or execute
ØØØ24               DEFB    ØDH             ;Terminate the filespec
ØØØ25
ØØØ26   FCB1:       DEFS    32              ;File Control Block for the program
ØØØ27
ØØØ28   ;           Get the File Control Block for the program 'DIREX/CMD'.
ØØØ29
ØØØ3Ø   START:      LD      HL,PROGRM       ;Point at the filespec we want to
ØØØ31                                       ;execute or load into memory
ØØØ32               LD      DE,FCB1         ;Point at the File Control Block
ØØØ33               LD      A,@FSPEC        ;Perform a validity check on the filespec
ØØØ34                                       ;and copy the filespec into the FCB.
ØØØ35               RST     28H             ;Call the @FSPEC svc
ØØØ36
ØØØ37               LD      HL,MESS1        ;Point at our prompting message
ØØØ38               LD      A,@DSPLY        ;and print it on the display
ØØØ39               RST     28H             ;Call the @DSPLY svc
ØØØ4Ø
ØØØ41               LD      A,@KEY          ;Get the reply from the keyboard
ØØØ42               RST     28H             ;Call the @KEY svc
ØØØ43
ØØØ44               CP      ØDH             ;Was the character an <ENTER>?
ØØØ45               JR      Z,RUNIT         ;If Z was set, then run the program
ØØØ46
ØØØ47   ;           If it wasn't an <ENTER>, then we assume it was a <BREAK> and
ØØØ48   ;           load the program and enter the debugger.
ØØØ49
ØØØ5Ø               LD      DE,FCB1         ;Point at the File Control Block
ØØØ51               LD      A,@LOAD         ;and have this program loaded into memory
ØØØ52               RST     28H             ;Call the @LOAD svc
ØØØ53
ØØØ54   ;           Note that this program must not be overwritten by the program
ØØØ55   ;           we are loading.  In this example, it is known that the program
ØØØ56   ;           we are loading starts at x'3ØØØ' and ends below x'5ØØØ'.
ØØØ57
ØØØ58               LD      A,@DEBUG        ;Now invoke the system debugger, DEBUG
ØØØ59               RST     28H             ;Call the @DEBUG svc
ØØØ6Ø                                       ;Note that @DEBUG does not return
ØØØ61
ØØØ62   ;           Execute the program
ØØØ63
ØØØ64   RUNIT:      LD      DE,FCB1         ;Point at the File Control Block
ØØØ65               LD      A,@RUN          ;Tell TRSDOS to load and execute the
ØØØ66                                       ;program
ØØØ67               RST     28H             ;Call the @RUN svc
```

```
ØØØ68                                   ;Note that @RUN returns only if it can't
ØØØ69                                   ;find the program
ØØØ7Ø
ØØØ71    ;       Note that the program that is loaded by the @RUN svc must not
ØØØ72    ;       overwrite the File Control Block in this program.  In this case,
ØØØ73    ;       it is known that the program we are executing starts at x'3ØØØ'
ØØØ74    ;       and ends below the starting point of this program, x'5ØØØ'.
ØØØ75
ØØØ76            END     START
```

Sample Program B

```
ØØØØ1   ;This program accepts numbers from the keyboard
ØØØØ2   ;and uses them to demonstrate the
ØØØØ3   ;arithmetic and numeric conversion SVCs.
ØØØØ4
ØØØØ5   ;It also uses the sound function to produce a tone at the
ØØØØ6   ;beginning of the program.
ØØØØ7
ØØØØ8           PSECT   3ØØØH
ØØØ1Ø
ØØØ11   ;       These are the SVCs used in this program.
ØØØ12
ØØØ13   @DECHEX:EQU    96              ;Convert decimal ASCII to binary
ØØØ14   @DIV8:  EQU     93              ;Perform 8-bit division
ØØØ15   @DIV16: EQU     94              ;Perform 16-bit division
ØØØ16   @DSP:   EQU     2               ;Display a character
ØØØ17   @DSPLY: EQU     1Ø              ;Display a message
ØØØ18   @EXIT:  EQU     22              ;Return to TRSDOS Ready or the caller
ØØØ19   @HEX8:  EQU     98              ;Convert an 8-bit value to hex ASCII
ØØØ2Ø   @HEX16: EQU     99              ;Convert a 16-bit value to hex ASCII
ØØØ21   @HEXDEC:EQU     97              ;Convert a binary value to Decimal ASCII
ØØØ22   @KEY:   EQU     1               ;Read a character from *KI
ØØØ23   @KEYIN: EQU     9               ;Accept an input line from *KI
ØØØ24   @MUL8:  EQU     9Ø              ;Perform 8-bit multiplication
ØØØ25   @MUL16: EQU     91              ;Perform 16-bit multiplication
ØØØ26   @SOUND: EQU     1Ø4             ;Produce a tone
ØØØ27
ØØØ28   ;       Other equates.
ØØØ29
ØØØ3Ø   NUM5:   EQU     5
ØØØ31   NUM4:   EQU     4
ØØØ32   NUM3:   EQU     3
ØØØ33   NUM2:   EQU     2
ØØØ34   NUM1:   EQU     1
ØØØ35   BRK:    EQU     8ØH             ;Character code for <BREAK> key
ØØØ36   CCC:    EQU     ØDH             ;Next line position
ØØØ37
ØØØ38
ØØØ39   ;Perform a subroutine 2 times to display prompting messages, key in
ØØØ4Ø   ;and display divisor and dividend, convert those numbers to
ØØØ41   ;binary for the divide, and position the cursor.
ØØØ42
ØØØ43   START:  LD      B,5AH           ;Make the longest, highest tone
ØØØ44           LD      A,@SOUND        ;Make the noise
ØØØ45           RST     28H
ØØØ46           CALL    KEYIN           ;Perform keyin subroutine for dividend
ØØØ47           LD      A,C
ØØØ48           LD      (DIVD1),A       ;Store the dividend in memory
ØØØ49           LD      HL,MESS9        ;Address of hex message
ØØØ5Ø           CALL    DSPLAY          ;Display hex message
ØØØ51           LD      A,(DIVD1)       ;Get the divisor into C for conversion
ØØØ52           LD      C,A             ;from binary to hex
ØØØ53           CALL    HEX8            ;Convert the number to hex
ØØØ54           CALL    KEYIN           ;Perform subroutine for divisor
ØØØ55           LD      A,C
ØØØ56           LD      (DIVR1),A       ;Store the divisor in memory
ØØØ57
ØØØ58   ;Now we are ready to perform the divide on the numbers entered.
ØØØ59
ØØØ6Ø           LD      C,A             ;Put the divisor back for the @DIV8 SVC
ØØØ61           LD      A,(DIVD1)       ;Get the dividend into E
ØØØ62           LD      E,A             ;for the @DIV8 SVC
ØØØ63           LD      A,@DIV8         ;Call the @DIV8 SVC
ØØØ64           RST     28H
ØØØ65
ØØØ66   ;Now display the answer and the remainder in decimal.
ØØØ67
ØØØ68           LD      (ANS1),A        ;Store the answer in memory
```

```
ØØØ69          LD      A,E             ;Get the remainder
ØØØ7Ø          LD      (REM1),A        ;Store the remainder in memory
ØØØ71          LD      HL,MESS3        ;Load address of answer message
ØØØ72          CALL    DSPLAY          ;Display the message
ØØØ73          LD      A,(ANS1)        ;Get the answer into L for conversion
ØØØ74          LD      L,A             ;Number to convert
ØØØ75          LD      H,Ø             ;Put a Ø in the MSB
ØØØ76          CALL    HEXDEC          ;Perform subroutine to display decimal value
ØØØ77          LD      HL,MESS4        ;Address of remainder message
ØØØ78          CALL    DSPLAY          ;Display remainder message
ØØØ79          LD      A,(REM1)        ;Put remainder in A for hex conversion
ØØØ8Ø          LD      L,A             ;Number to convert
ØØØ81          LD      H,Ø             ;Put Ø in the MSB
ØØØ82          CALL    HEXDEC          ;Display decimal value
ØØØ83
ØØØ84   ;Now divide with a 16-bit dividend.
ØØØ85
ØØØ86          LD      HL,MESS6        ;Address of 2nd dividend message
ØØØ87          CALL    DSPLAY          ;Display next message
ØØØ88          LD      A,@KEYIN        ;Key in up to 5 digits
ØØØ89          LD      HL,BUF6         ;Store the number
ØØØ9Ø          LD      B,NUM5          ;Maximum length of number
ØØØ91          LD      C,Ø
ØØØ92          RST     28H
ØØØ93          LD      A,@DECHEX       ;Convert the number to binary
ØØØ94          RST     28H
ØØØ95          LD      (DIVD2),BC      ;Store the dividend
ØØØ96          LD      HL,MESS9        ;Address of hex message
ØØØ97          CALL    DSPLAY          ;Display hex message
ØØØ98          LD      DE,(DIVD2)      ;Put dividend into DE for conversion
ØØØ99          CALL    HEX16           ;Convert the number from binary to hex
ØØ1ØØ          CALL    KEYIN           ;Key in divisor
ØØ1Ø1          LD      A,C             ;Put the divisor into A
ØØ1Ø2          LD      (DIVR1),A       ;Store the divisor in memory
ØØ1Ø3          LD      HL,MESS3        ;Address of answer message
ØØ1Ø4          CALL    DSPLAY          ;Display the message
ØØ1Ø5          LD      HL,(DIVD2)      ;Put dividend into HL
ØØ1Ø6          LD      A,(DIVR1)       ;Get divisor into C
ØØ1Ø7          LD      C,A
ØØ1Ø8          LD      A,@DIV16
ØØ1Ø9          RST     28H
ØØ11Ø          LD      (REM1),A        ;Store the remainder
ØØ111          LD      (ANS2),HL       ;Put the answer into HL
ØØ112          CALL    HEXDEC          ;Display answer in decimal
ØØ113          LD      HL,MESS4        ;Address of remainder message
ØØ114          CALL    DSPLAY          ;Display remainder message
ØØ115          LD      A,(REM1)        ;Get the remainder
ØØ116          LD      L,A             ;into L
ØØ117          LD      H,Ø             ;Put a Ø in MSB
ØØ118          CALL    HEXDEC          ;Convert the remainder to decimal
ØØ119
ØØ12Ø   ;Now try some multiplication of 8 bits.
ØØ121
ØØ122          LD      HL,MESS8        ;Address of MUL8 message
ØØ123          CALL    DSPLAY          ;Display first multiplicand message
ØØ124          LD      A,@KEYIN        ;Key in a 2-digit number
ØØ125          LD      HL,BUF2         ;Put it here
ØØ126          LD      B,NUM2          ;Maximum number of characters
ØØ127          LD      C,Ø
ØØ128          RST     28H
ØØ129          LD      A,@DECHEX       ;Convert the number to binary for math
ØØ13Ø          RST     28H
ØØ131          LD      (MCAND1),BC     ;Store the multiplicand
ØØ132          LD      HL,MESS1Ø       ;Address of MUL8 multiplier message
ØØ133          CALL    DSPLAY          ;Display first multiplier message
ØØ134          LD      A,@KEYIN        ;Key in the multiplier
ØØ135          LD      HL,BUF2         ;Put it here
```

```
ØØ136          LD      B,NUM1          ;Maximum number of characters
ØØ137          LD      C,Ø
ØØ138          RST     28H
ØØ139          LD      A,@DECHEX       ;Convert the multiplier to binary for math
ØØ14Ø          RST     28H
ØØ141          LD      (MIER1),BC      ;Store multiplier in memory
ØØ142          LD      HL,MESS13       ;Address of multiplier message
ØØ143          LD      A,@DSPLY        ;Display multiplier message
ØØ144          RST     28H
ØØ145
ØØ146   ;Now multiply the two numbers just entered.
ØØ147
ØØ148          LD      A,(MCAND1)      ;Get the multiplicand into C
ØØ149          LD      C,A
ØØ15Ø          LD      A,(MIER1)       ;Get the multiplier into E
ØØ151          LD      E,A
ØØ152          LD      A,@MUL8
ØØ153          RST     28H
ØØ154          LD      L,A             ;Put the product into L
ØØ155          LD      H,Ø             ;Put Ø in the MSB
ØØ156          CALL    HEXDEC          ;Convert the product to decimal
ØØ157
ØØ158   ;Now multiply a 16-bit by an 8-bit.
ØØ159
ØØ16Ø          LD      HL,MESS11       ;Address of multiplicand message
ØØ161          CALL    DSPLAY          ;Display 2nd multiplicand message
ØØ162          LD      A,@KEYIN        ;Enter larger multiplicand
ØØ163          LD      HL,BUF5         ;Put it here
ØØ164          LD      B,NUM4          ;Maximum number of characters
ØØ165          LD      C,Ø
ØØ166          RST     28H
ØØ167          LD      A,@DECHEX       ;Convert the number to binary for math
ØØ168          RST     28H
ØØ169          LD      (MCAND2),BC     ;Store the multiplicand in memory
ØØ17Ø          LD      HL,MESS12       ;Address of multiplier message
ØØ171          CALL    DSPLAY          ;Display message
ØØ172          LD      A,@KEYIN        ;Enter larger multiplier
ØØ173          LD      HL,BUF3         ;Put it here
ØØ174          LD      B,NUM2          ;Maximum number of characters
ØØ175          LD      C,Ø
ØØ176          RST     28H
ØØ177          LD      A,@DECHEX       ;Convert the number to binary for math
ØØ178          RST     28H
ØØ179          LD      (MIER1),BC      ;Store the multiplier in memory
ØØ18Ø          LD      HL,MESS13       ;Address of product message
ØØ181          LD      A,@DSPLY        ;Display the message
ØØ182          RST     28H
ØØ183          LD      HL,(MCAND2)     ;Put multiplicand into HL
ØØ184          LD      A,(MIER1)       ;Get the multiplier into C
ØØ185          LD      C,A
ØØ186          LD      A,@MUL16        ;Multiply the two numbers
ØØ187          RST     28H
ØØ188          LD      H,L             ;Get the 2nd byte of the product into
ØØ189                                  ;H for conversion
ØØ19Ø          LD      L,A             ;Get the LSB into L for conversion
ØØ191          LD      DE,BUF5         ;Convert the high-order byte to decimal
ØØ192          LD      A,@HEXDEC       ;for the  display
ØØ193          RST     28H
ØØ194          LD      A,CCC           ;Tell the display when to stop
ØØ195          LD      (DE),A
ØØ196          LD      HL,BUF5
ØØ197          LD      A,@DSPLY        ;Display the product
ØØ198          RST     28H
ØØ199          LD      HL,MESS14       ;Address of end message
ØØ2ØØ          LD      A,@DSPLY        ;Display end message
ØØ2Ø1          RST     28H
ØØ2Ø2          LD      A,@KEY          ;Allow the user to enter any character
ØØ2Ø3          RST     28H             ;or hit <BREAK>
```

```
00204          CP      BRK             ;Is it <BREAK>?
00205          JP      NZ,START        ;Yes, go back to beginning
00206          LD      A,@EXIT         ;No, exit the program
00207          RST     28H
00208
00209  ;These are the subroutines used by the calls to
00210  ;display a message, key in a 3-digit number, and convert it
00211  ;from decimal to binary.
00212
00213  KEYIN:  LD      HL,MESS1
00214          CALL    DSPLAY          ;Display message
00215          LD      HL,BUF4         ;Put the number here
00216          LD      B,NUM3          ;Maximum number of characters
00217          LD      C,0
00218          LD      A,@KEYIN        ;Key in a number
00219          RST     28H
00220          LD      A,@DECHEX       ;Convert the number to binary
00221          RST     28H
00222          RET                     ;Return to next sequential instruction
00223
00224  ;Display what was loaded into HL before the call.
00225
00226  DSPLAY: LD      A,@DSPLY    ;@DISPLAY SVC
00227          RST     28H
00228          DEC     HL              ;Set HL back to blank byte
00229          LD      B,(HL)          ;Load B with the number of bytes
00230  DSPLYLP:LD      C,' '           ;Put a blank into C
00231          LD      A,@DSP          ;Display the blank
00232          RST     28H             ;until the correct number
00233          DJNZ    DSPLYLP         ;of blanks have been displayed
00234          RET                     ;Return to next instruction
00235
00236  ;Convert 1 byte to hexadecimal.
00237
00238  HEX8:   LD      A,@HEX8         ;Convert 1 byte to hex ASCII
00239          LD      HL,BUF3         ;Put the converted value here
00240          RST     28H
00241          LD      A,CCC           ;Tell display when to stop
00242          LD      (HL),A          ;Put CCC at end of buffer
00243          LD      A,@DSPLY        ;Display the hex value
00244          LD      HL,BUF3
00245          RST     28H
00246          RET                     ;Return to next instruction
00247
00248  ;Convert 2 bytes to hexadecimal.
00249
00250  HEX16:  LD      A,@HEX16        ;Convert a 2-byte number to hex ASCII
00251          LD      HL,BUF6         ;Put the converted value here
00252          RST     28H
00253          LD      A,CCC           ;CCC at end of buffer so display
00254          LD      (HL),A          ;knows when to stop
00255          LD      A,@DSPLY        ;Display the converted value
00256          LD      HL,BUF6         ;Address of converted value
00257          RST     28H
00258          RET                     ;Return to next instruction
00259
00260  ;Convert from binary to decimal and display decimal value.
00261
00262  HEXDEC: LD      A,@HEXDEC       ;Convert from binary to decimal
00263          LD      DE,BUF5         ;Put converted value here
00264          RST     28H
00265          LD      A,CCC           ;CCC at end of buffer so display
00266          LD      (DE),A          ;knows when to stop
00267          LD      A,@DSPLY        ;Display the hex value
00268          LD      HL,BUF5         ;It's here
00269          RST     28H
00270          RET                     ;Return to next instruction
00271
```

```
ØØ272   ;These are the storage declarations.
ØØ273
ØØ274   BUF6:   DEFS    6
ØØ275   BUF5    DEFS    5
ØØ276   BUF4:   DEFS    4
ØØ277   BUF3:   DEFS    3
ØØ278   BUF2:   DEFS    2
ØØ279   DIVR1:  DEFB    Ø
ØØ28Ø   DIVD1:  DEFB    Ø
ØØ281   ANS1:   DEFB    Ø
ØØ282   REM1:   DEFB    Ø
ØØ283   MCAND1: DEFB    Ø
ØØ284   MIER1:  DEFB    Ø
ØØ285   MCAND2: DEFW    Ø
ØØ286   DIVD2:  DEFW    Ø
ØØ287   ANS2:   DEFW    Ø
ØØ288
ØØ289   ;Below are messages and prompting text used in the program.
ØØ29Ø
ØØ291           DEFB    13              ;Number of blanks to print after message 1
ØØ292   MESS1:  DEFM    'Enter a number (1-255).'
ØØ293           DEFB    3               ;Message-terminating character
ØØ294           DEFB    21              ;Number of blanks to print after message 3
ØØ295   MESS3:  DEFM    'The answer is'
ØØ296           DEFB    3               ;Terminating character
ØØ297           DEFB    18              ;Blanks after message
ØØ298   MESS4:  DEFM    'The remainder is'
ØØ299           DEFB    3               ;Terminating character
ØØ3ØØ           DEFB    6               ;Blanks after message
ØØ3Ø1   MESS6:  DEFM    'Enter a number (4369-65535).'
ØØ3Ø2           DEFB    3               ;Terminating character
ØØ3Ø3           DEFB    15              ;Blanks after message
ØØ3Ø4   MESS8:  DEFM    'Enter a number (1-28).'
ØØ3Ø5           DEFB    3               ;Terminating character
ØØ3Ø6           DEFB    16              ;Blanks after message
ØØ3Ø7   MESS9:  DEFM    'In hex ASCII, that is'
ØØ3Ø8           DEFB    3               ;Terminating character
ØØ3Ø9           DEFB    17              ;Blanks after message
ØØ31Ø   MESS1Ø: DEFM    'Enter a number (1-9).
ØØ311           DEFB    3               ;Terminating character
ØØ312           DEFB    11              ;Blanks after message
ØØ313   MESS11: DEFM    'Enter a number (1-41ØØ).'
ØØ314           DEFB    3               ;Terminating character
ØØ315           DEFB    15              ;Blanks after message
ØØ316   MESS12: DEFM    'Enter a number (1-15).'
ØØ317           DEFB    3               ;Terminating character
ØØ318   MESS13: DEFM    'The product of those 2 numbers is '
ØØ319           DEFB    3               ;Terminating character
ØØ32Ø   MESS14: DEFM    'Press <BREAK> to end or any other key to continue.'
ØØ321           DEFB    ØDH             ;Terminating character
ØØ322
ØØ323           END     START
```

Sample Program C

```
Ln #            Source Line

ØØØØ1   ;       This program prompts for two filenames, opens the first
ØØØØ2   ;       file, and creates the second.  Then the data in the first
ØØØØ3   ;       file is copied to the second file.  While the Copy progresses,
ØØØØ4   ;       the current record number is displayed in parentheses.
ØØØØ5
ØØØØ6           PSECT   3ØØØH           ;This program starts at x'3ØØØ'
ØØØØ8
ØØØØ9   ;       First, declare the equates for the SVCs we intend to use.
ØØØ1Ø   ;       This is not mandatory, but it makes the program easier to follow.
ØØØ11
ØØØ12   @CLOSE: EQU     6Ø              ;Close a file or device
ØØØ13   @DIRRD: EQU     87              ;Read a directory record
ØØØ14   @DSP:   EQU     2               ;Display character at cursor
ØØØ15   @DSPLY: EQU     1Ø              ;Display a message
ØØØ16   @ERROR: EQU     26              ;Display an error message
ØØØ17   @EXIT:  EQU     22              ;Exit and return to TRSDOS or the caller
ØØØ18   @FEXT:  EQU     79              ;Add a default file extension
ØØØ19   @FNAME: EQU     8Ø              ;Fetch a filespec from the directory
ØØØ2Ø   @FSPEC: EQU     78              ;Verify and load a filespec into the FCB
ØØØ21   @HEXDEC:EQU     97              ;Convert a binary value to decimal ASCII
ØØØ22   @INIT:  EQU     58              ;Open an existing file or create a new file
ØØØ23   @KBD:   EQU     8               ;Scan the keyboard for a character
ØØØ24   @KEYIN: EQU     9               ;Accept a line of text from the *KI device
ØØØ25   @LOC:   EQU     63              ;Return the current logical record number
ØØØ26   @OPEN:  EQU     59              ;Open an existing file
ØØØ27   @READ:  EQU     67              ;Read a record from an open file
ØØØ28   @REMOV: EQU     57              ;Delete a file from disk
ØØØ29   @VER:   EQU     73              ;Write a record to disk.  Does the same thing
ØØØ3Ø                                   ;as @WRITE (Svc 75), but it also makes sure
ØØØ31                                   ;the written data is readable.
ØØØ32
ØØØ33   ;       First, prompt for the source filespec using the @DSPLY svc.
ØØØ34
ØØØ35   BEGIN:  LD      HL,MESG1        ;Get the first message
ØØØ36           LD      A,@DSPLY        ;Display a line on the screen
ØØØ37           RST     28H             ;Call the @DSPLY svc
ØØØ38
ØØØ39   ;       Now, read the filename from the keyboard using the @KEYIN svc.
ØØØ4Ø
ØØØ41           LD      HL,FILE1        ;Put the name of the 1st file here
ØØØ42           LD      B,24            ;Allow up to 24 characters
ØØØ43           LD      C,Ø             ;A zero is required by the svc
ØØØ44           LD      A,@KEYIN        ;Get a filename from the user
ØØØ45           RST     28H             ;Call the @KEYIN svc
ØØØ46   JP      C,QUIT                  ;The user pressed <Break>
ØØØ47           JP      NZ,ERR          ;An Error occurred
ØØØ48
ØØØ49           LD      A,B             ;Get the number of characters
ØØØ5Ø           OR      A               ;See if that value was zero
ØØØ51           JR      Z,BEGIN         ;Nothing was entered, ask again
ØØØ52
ØØØ53   ;       The user has typed something, so it must be checked for validity
ØØØ54   ;       using the @FSPEC svc.
ØØØ55
ØØØ56           LD      HL,FILE1        ;Point at the text the user entered
ØØØ57           LD      DE,FCB1         ;Point at the File Control Block
ØØØ58                                   ;that is to be used for the source file.
ØØØ59           LD      A,@FSPEC        ;The @FSPEC svc will make sure the filename
ØØØ6Ø                                   ;that is in buffer named "file1" is valid.
ØØØ61                                   ;If it is, it is copied into the File
ØØØ62                                   ;Control Block (FCB) to be used by the @OPEN
ØØØ63                                   ;or @INIT svc later on.
ØØØ64           RST     28H             ;Call the @FSPEC svc
ØØØ65           JR      Z,ASK2          ;The name for file 1 is ok, so skip this
ØØØ66
ØØØ67   ;       At this point the filename specified for file 1 has been found
```

```
ØØØ68   ;       to be in an invalid format.  The following code will print the
ØØØ69   ;       error message.
ØØØ7Ø
ØØØ71           LD      HL,BADFIL       ;Point at the bad filename message
ØØØ72           LD      A,@DSPLY        ;Display it
ØØØ73           RST     28H             ;Call the @DSPLY svc
ØØØ74           JR      BEGIN           ;Start over
ØØØ75
ØØØ76   ;       At this point, the source filename appears to be valid.
ØØØ77   ;       The code below asks for the second filename and checks it for
ØØØ78   ;       validity also.
ØØØ79
ØØØ8Ø   ASK2:   LD      HL,MESG2        ;Prompt for the target filename
ØØØ81           LD      A,@DSPLY        ;Print that on the screen
ØØØ82           RST     28H             ;Call the @DSPLY svc
ØØØ83           LD      HL,FILE2        ;Put the name of the 2nd file here
ØØØ84           LD      B,24            ;Allow up to 24 characters
ØØØ85           LD      C,Ø             ;A zero is required by the svc
ØØØ86           LD      A,@KEYIN        ;Get a filename from the user
ØØØ87           RST     28H             ;Call the @KEYIN svc
ØØØ88           JP      C,QUIT          ;The user pressed <Break>
ØØØ89           JP      NZ,ERR          ;An Error occurred
ØØØ9Ø
ØØØ91           LD      A,B             ;Get the number of characters
ØØØ92           OR      A               ;See if that value was zero.
ØØØ93           JR      Z,ASK2          ;Nothing was entered, ask again
ØØØ94
ØØØ95   ;       The user has typed something, so it must be checked for validity
ØØØ96   ;       using the @FSPEC svc.
ØØØ97
ØØØ98           LD      HL,FILE2        ;Point at the text the user entered
ØØØ99           LD      DE,FCB2         ;Point at the File Control Block
ØØ1ØØ           LD      A,@FSPEC        ;Check the name for validity
ØØ1Ø1           RST     28H             ;Call the @FSPEC svc
ØØ1Ø2           JR      Z,F2OK          ;The name for file 2 is ok, so skip this
ØØ1Ø3
ØØ1Ø4   ;       The name for file 2 is invalid so print an error message
ØØ1Ø5
ØØ1Ø6           LD      HL,BADFIL       ;Point at the bad filename message
ØØ1Ø7           LD      A,@DSPLY        ;Display it
ØØ1Ø8           RST     28H             ;Call the @DSPLY svc
ØØ1Ø9           JR      BEGIN           ;Start over
ØØ11Ø
ØØ111   ;       Now we will attempt to add an extension to the target file
ØØ112   ;       if the user did not specify one.  We use the extension that
ØØ113   ;       was specified on the source file.  If it does
ØØ114   ;       not have one, then we will not try to add one to the target file.
ØØ115
ØØ116   F2OK:   LD      HL,FCB1+1       ;Point at the source filename
ØØ117                                   ;We start with the second character since
ØØ118                                   ;the filename must be at least one character
ØØ119   FDIV:   LD      A,(HL)          ;Get a character from the filespec
ØØ12Ø           CP      '/'             ;Is the character the extension prefix?
ØØ121           JR      Z,EXTN          ;Yes, this will be our default extension
ØØ122           CP      ØDH             ;Have we reached the end of the filespec?
ØØ123           JR      Z,NOEXT         ;Yes, there is no extension so don't add one
ØØ124           CP      Ø3H             ;Test both terminators
ØØ125           JR      Z,NOEXT
ØØ126           INC     HL              ;Advance the pointer to the next character
ØØ127           JR      FDIV            ;Keep looking
ØØ128
ØØ129   EXTN:   INC     HL              ;Advance pointer to first byte of extension
ØØ13Ø           LD      DE,FCB2         ;Point at FCB for the target file (file 2)
ØØ131           LD      A,@FEXT         ;Add an extension if one is not present
ØØ132           RST     28H             ;Call the @FEXT svc
ØØ133
ØØ134   ;       Now we have two filenames.  First we will open the source file
ØØ135   ;       to make sure it exists.
```

```
ØØ136
ØØ137   NOEXT:  LD      DE,FCB1         ;Point at the File Control Block for file1
ØØ138           LD      HL,BUF1         ;Point at the system buffer.  This buffer
ØØ139                                   ;is used by the system to block data that
ØØ14Ø                                   ;is written to disk and de-block data that
ØØ141                                   ;is read from disk when the Logical Record
ØØ142                                   ;Length of the file is not 256.  If it is
ØØ143                                   ;256, then this buffer is not used.
ØØ144           LD      B,Ø             ;Use LRL 256 for now since we don't know
ØØ145                                   ;what to use yet.
ØØ146           LD      A,@OPEN         ;Open the file
ØØ147           RST     28H             ;Call the @OPEN svc
ØØ148           JR      Z,SIZ           ;The file opened and is LRL 256.
ØØ149           CP      42              ;Was the error a LRL Open Fault?
ØØ15Ø           JP      NZ,ERR          ;No, perhaps the file does not exist.
ØØ151
ØØ152   ;       At this point, the file is open and we can now examine the
ØØ153   ;       directory to find out what LRL it was created with so we can
ØØ154   ;       use that value to make the copy.
ØØ155
ØØ156   SIZ:    LD      A,(FCB1+6)      ;Get the byte in the FCB which contains
ØØ157                                   ;the drive number the file is on
ØØ158           AND     7               ;Erase all other information in that byte
ØØ159           LD      C,A             ;Save that value here
ØØ16Ø           LD      A,(FCB1+7)      ;This reads the Directory Entry Code (DEC)
ØØ161                                   ;out of the FCB so we can use it
ØØ162           LD      B,A             ;Store the DEC here
ØØ163           PUSH    BC              ;Save that value for now
ØØ164           LD      A,@CLOSE        ;We can close the source file for now
ØØ165           RST     28H             ;Call the @CLOSE svc
ØØ166
ØØ167           POP     BC              ;Get the DEC value back off the stack
ØØ168           LD      A,@DIRRD        ;Read the directory record for that file
ØØ169           RST     28H             ;Call the @DIRRD svc
ØØ17Ø
ØØ171           LD      IX,HL           ;Put the pointer to the directory record
ØØ172           LD      A,(IX+4)        ;here and read the DIR+4 entry which
ØØ173                                   ;contains the LRL of the source file.
ØØ174           LD      (LRL),A         ;Save that value
ØØ175
ØØ176   ;       Before we go any further, we should check to see if the target file
ØØ177   ;       already exists.
ØØ178
ØØ179           LD      DE,COPY         ;First, make a copy of the FCB
ØØ18Ø           LD      HL,FCB2         ;in case we have to delete a file
ØØ181           LD      BC,32           ;Move the entire block
ØØ182           LDIR
ØØ183
ØØ184           LD      DE,FCB2         ;Point at the target File Control Block
ØØ185           LD      HL,BUF2         ;Use this as the buffer for now
ØØ186           LD      B,Ø             ;Use LRL 256 for now
ØØ187           LD      A,@OPEN         ;Open it and see if it is there
ØØ188           RST     28H             ;Call the @OPEN svc
ØØ189           JR      Z,EXISTS        ;The file already exists, better ask
ØØ19Ø           CP      42              ;Was the error a LRL mismatch?
ØØ191           JR      NZ,NOFILE       ;No, so the file does not exist.
ØØ192
ØØ193   EXISTS: LD      HL,FEXST        ;Point at a prompt asking if it is ok
ØØ194                                   ;to erase the file that already exists
ØØ195           LD      A,@DSPLY        ;Print that message
ØØ196           RST     28H             ;Call the @DSPLY svc
ØØ197
ØØ198   WAIT:   LD      A,@KBD          ;Wait for the user to type Y or N
ØØ199           RST     28H             ;Call the @KBD svc
ØØ2ØØ           JR      NZ,WAIT         ;Loop until something is typed
ØØ2Ø1
ØØ2Ø2           CP      'Y'             ;Was a 'Y' typed?
ØØ2Ø3           JR      Z,KILLIT        ;Then kill the file
```

```
00204           CP      'y'             ;Check for lowercase too
00205           JR      Z,KILLIT
00206           CP      'N'             ;Do they want to leave the file alone?
00207           JR      Z,SHUT          ;No, just close the file and quit
00208           CP      'n'             ;Was it a lowercase 'N'?
00209           JR      NZ,WAIT         ;No, loop until we see something we like
00210
00211   SHUT:   LD      DE,FCB2         ;Close the target file
00212           LD      A,@CLOSE
00213           RST     28H             ;Call the @CLOSE svc
00214           JP      QUIT            ;Exit to TRSDOS
00215
00216   ;       At this point, we have been given the OK to delete the file
00217   ;       that has the same name as the target file.
00218
00219   KILLIT: LD      C,0DH           ;First move display to a new line
00220           LD      A,@DSP          ;Display an <Enter>
00221           RST     28H             ;Call the @DSP svc
00222
00223           LD      DE,FCB2         ;Point at the target file's FCB
00224           LD      A,@REMOV        ;Delete the file from disk
00225           RST     28H             ;Call the @REMOV svc.  (This is the same
00226                                   ;as the @KILL call on other TRSDOS systems.)
00227           JP      NZ,ERR          ;An error occurred, print it and quit
00228                                   ;Note that after a @REMOV succeeds,
00229                                   ;the filespec is removed from the FCB.
00230                                   ;So we have to keep a copy around
00231                                   ;in case we need it.
00232           LD      HL,COPY         ;Get the copy
00233           LD      DE,FCB2         ;Put it here
00234           LD      BC,32           ;Move up to 32 bytes
00235           LDIR                    ;Copy the FCB so we can continue
00236
00237   ;       Now we know what Logical Record Length (LRL) to use in the
00238   ;       copy, so we can open the source file and create the target file
00239   ;       with the correct record lengths.
00240
00241   NOFILE: LD      HL,FCB1         ;Point at the filename in the FCB
00242           LD      A,@DSPLY        ;Print that name
00243           RST     28H             ;Call the @DSPLY svc
00244           LD      HL,SPACES       ;Point at some spaces
00245           LD      A,@DSPLY        ;Space over a few places on the screen
00246           RST     28H             ;Call the @DSPLY svc
00247
00248           LD      DE,FCB1         ;Point at File Control Block for source file
00249           LD      HL,BUF1         ;Put data in this
00250           LD      A,(LRL)         ;Read the Logical Record Length
00251           LD      B,A             ;Load the Logical Record Length
00252           LD      A,@OPEN         ;Open the source file
00253           RST     28H             ;Call the @OPEN svc
00254           JP      NZ,ERR          ;Open failed
00255
00256           LD      HL,ARROW        ;Point at the arrow text
00257           LD      A,@DSPLY        ;Print that to show the direction of copy
00258           RST     28H             ;Call the @DSPLY svc
00259
00260           LD      DE,FCB2         ;Point at File Control Block for target file
00261           LD      A,(LRL)         ;Get the Logical Record Length
00262           CP      0               ;Is the LRL 256?
00263           JR      Z,LRL256        ;Then we do something special
00264           LD      HL,BUF2         ;Use a different buffer for target file
00265           JR      LRLCOM          ;Jump to common code
00266   LRL256: LD      HL,BUF1         ;We use the same buffer when the LRL is 256
00267                                   ;since there is no need to block and de-block
00268                                   ;the data.
00269   LRLCOM: LD      B,A             ;Load the Logical Record Length
00270           LD      A,@INIT         ;Open the target file
```

```
ØØ271            RST     28H             ;Call the @INIT svc
ØØ272            JR      NZ,ERR          ;Init failed
ØØ273
ØØ274            LD      DE,FILE2        ;We are going to get the filename for
ØØ275                                    ;the target file from the system
ØØ276                                    ;instead of using the one we have.  The
ØØ277                                    ;reason for this is that the system will
ØØ278                                    ;append the drive number to the filename
ØØ279                                    ;if one was not specified.
ØØ28Ø            LD      A,(FCB2+7)      ;Get the Directory Entry Code for the file
ØØ281            LD      B,A             ;Put the DEC here
ØØ282            LD      A,(FCB2+6)      ;Get the Drive Number from the FCB
ØØ283            AND     7               ;Lose all data except the drive number
ØØ284            LD      C,A             ;Store drive number here
ØØ285            LD      A,@FNAME        ;Have the system produce a filespec
ØØ286            RST     28H             ;Call the @FNAME svc
ØØ287            LD      HL,FILE2        ;Now point at the filespec produced
ØØ288            LD      A,@DSPLY        ;and print it out
ØØ289            RST     28H             ;Call the @DSPLY svc
ØØ29Ø
ØØ291            LD      HL,SPACES       ;Space over a few more places
ØØ292            LD      A,@DSPLY        ;so the display will look neat
ØØ293            RST     28H             ;Call the @DSPLY svc
ØØ294
ØØ295    ;       At this point, both files are open and ready to be used.
ØØ296    ;       The following code reads a record from the source file
ØØ297    ;       and writes it to the target file.  This is done until an
ØØ298    ;       end of file is encountered.
ØØ299
ØØ3ØØ    LOOP:   LD      DE,FCB1         ;Point at file 1 (source file)
ØØ3Ø1            LD      HL,BUFFER       ;Put data here
ØØ3Ø2            LD      A,@READ         ;Read a record from the source file
ØØ3Ø3            RST     28H             ;Call the @READ svc
ØØ3Ø4            JR      NZ,EOF          ;Jump if the eof has been reached
ØØ3Ø5            LD      DE,FCB2         ;Point at file 2 (target file)
ØØ3Ø6
ØØ3Ø7    ;       Before writing the record, display the record number, which
ØØ3Ø8    ;       is obtained from the @LOC svc.
ØØ3Ø9
ØØ31Ø            LD      A,@LOC          ;Get the current record number
ØØ311            RST     28H             ;Call the @LOC svc
ØØ312
ØØ313            PUSH    BC              ;Get the current record number
ØØ314            POP     HL              ;and put it in register HL
ØØ315            LD      DE,LOCMSG+1     ;Store the result here.
ØØ316            LD      A,@HEXDEC       ;Convert binary to ASCII in decimal format
ØØ317            RST     28H             ;Call the @HEXDEC svc
ØØ318
ØØ319            LD      A,' '           ;Get a blank
ØØ32Ø            LD      HL,LOCMSG       ;Look at the front of the buffer
ØØ321    EDIT:   CP      (HL)            ;Is the character a blank?
ØØ322            JR      NZ,NUMBR        ;A number has been found
ØØ323            INC     HL              ;Advance the pointer
ØØ324            JR      EDIT            ;Loop until we find a number
ØØ325
ØØ326    NUMBR:  DEC     HL              ;Back up one position
ØØ327            LD      A,'('           ;Get the character we want to insert
ØØ328            LD      (HL),A          ;Store that character.
ØØ329                                    ;The buffer now contains
ØØ33Ø                                    ;<none or more spaces>(record number)
ØØ331                                    ;<7 left-cursor characters><etx>
ØØ332            LD      HL,LOCMSG       ;Point at this text
ØØ333            LD      A,@DSPLY        ;and display it on the screen
ØØ334            RST     28H             ;Call the @DSPLY svc
ØØ335
ØØ336    ;       Now write the record to the target file.
ØØ337
ØØ338            LD      DE,FCB2         ;Point at the FCB for the target file
```

```
ØØ339           LD      HL,BUFFER       ;Point at the data read from file 1
ØØ34Ø           LD      A,@VER          ;Write a record to the target file
ØØ341                                   ;The @VER does the same thing as the
ØØ342                                   ;@WRITE svc, only it also checks the
ØØ343                                   ;data to make sure it is readable.
ØØ344           RST     28H             ;Call the @VER svc
ØØ345           JR      NZ,ERR          ;An error occurred on write; possibly
ØØ346                                   ;the disk is full.
ØØ347           JR      LOOP            ;Loop until an error occurs.
ØØ348
ØØ349   ;       This code checks the error to make sure it was an end of file
ØØ35Ø   ;       condition and, if so, closes the source & target files.
ØØ351
ØØ352   EOF:    CP      28              ;Was it an end of file encountered?
ØØ353           JR      NZ,ERR          ;It was some other type of error - Abort
ØØ354
ØØ355           LD      DE,FCB1         ;Point at file 1 (source file)
ØØ356           LD      A,@CLOSE        ;Close the file
ØØ357           RST     28H             ;Call the @CLOSE svc
ØØ358           JR      NZ,ERR          ;An error occurred, abort
ØØ359
ØØ36Ø           LD      DE,FCB2         ;Point at file 2 (target file)
ØØ361           LD      A,@CLOSE        ;Close it also
ØØ362           RST     28H             ;Call the @CLOSE svc
ØØ363           JR      NZ,ERR          ;An error occurred, abort
ØØ364
ØØ365           LD      HL,OK           ;Print a message saying the copy is done
ØØ366           LD      A,@DSPLY
ØØ367           RST     28H             ;Call the @DSPLY svc
ØØ368
ØØ369   QUIT:   LD      A,@EXIT         ;Exit to TRSDOS or the calling program
ØØ37Ø           RST     28H             ;Call the @EXIT svc
ØØ371
ØØ372   ;       The @EXIT svc does not return.
ØØ373
ØØ374   ERR:    OR      Ø4ØH            ;Turn on bit 6, which
ØØ375                                   ;will cause the @ERROR svc to print
ØØ376                                   ;the short error message. Bit 7
ØØ377                                   ;is not set, which instructs the @ERROR
ØØ378                                   ;to abort this program and return to
ØØ379                                   ;TRSDOS Ready.
ØØ38Ø           LD      C,A             ;Put error code & flags in register C
ØØ381           LD      A,@ERROR        ;Call the system error displayer
ØØ382           RST     28H             ;Call the @ERROR svc
ØØ383
ØØ384   ;       Because bit 7 is not set, the @ERROR svc will not return.
ØØ385
ØØ386   ;       Storage Declaration
ØØ387
ØØ388   SPACES: DEFM    '  '            ;ASCII Space characters for display formatting
ØØ389           DEFB    3
ØØ39Ø   ARROW:  DEFM    '=>  '          ;Arrow for display to indicate data direction
ØØ391           DEFB    3
ØØ392   OK:     DEFB    1Ø%25           ;Advance cursor 1Ø spaces without erasing
ØØ393           DEFM    '[Ok]'          ;Used to indicate the Copy is complete
ØØ394           DEFB    ØDH             ;Terminated with an <Enter>
ØØ395   MESG1:  DEFM    'Copy Filespec >'
ØØ396           DEFB    3
ØØ397   MESG2:  DEFM    'To Filespec >'
ØØ398           DEFB    3
ØØ399   FEXST:  DEFM    'Destination File Already Exists - Ok to Delete it (Y/N) ?'
ØØ4ØØ           DEFB    3
ØØ4Ø1   BADFIL: DEFM    'Invalid Filename - Try Again'
ØØ4Ø2           DEFB    ØDH
ØØ4Ø3   LOCMSG: DEFM    ' 12345)'       ;This will be used in building the LOC
ØØ4Ø4                                   ;display which will appear as (d) to (ddddd).
ØØ4Ø5           DEFB    7%24            ;Backspace without erasing
```

```
ØØ4Ø6            DEFB    3              ;Etx, used to get the @DSPLY svc to stop
ØØ4Ø7
ØØ4Ø8   FILE1:   DEFS    32             ;User Text Originally placed here
ØØ4Ø9   FILE2:   DEFS    32             ;Target Filename goes here
ØØ41Ø   FCB1:    DEFS    32             ;32 bytes for the File Control Block
ØØ411   FCB2:    DEFS    32             ;32 bytes for the File Control Block
ØØ412   COPY:    DEFS    32             ;An extra copy of the target FCB goes here
ØØ413   LRL:     DEFB    Ø              ;The Logical Record Length of the source
ØØ414                                   ;file will be stored here
ØØ415   BUF1:    DEFS    256            ;System buffer for File 1
ØØ416   BUF2:    DEFS    256            ;System buffer for File 2
ØØ417   BUFFER:  DEFS    256            ;Data buffer for both files
ØØ418
ØØ419            END     BEGIN          ;"begin" is the starting address
```

Sample Program D

```
Ln #            Source Line

ØØØØ1   ;       This program will read a sector from the disk in Drive Ø
ØØØØ2   ;       and will write it to a disk in Drive 1.  The disk in Drive 1
ØØØØ3   ;       must be formatted, but should not have anything important on
ØØØØ4   ;       it.  This program makes an assumption that the directory is
ØØØØ5   ;       located on cylinder 2Ø (x'14').
ØØØØ6
ØØØØ7           PSECT   3ØØØH           ;This program begins at x'3ØØØ'.
ØØØØ9
ØØØ1Ø   ;       Define the equates for the SVCs that will be used.
ØØØ11
ØØØ12   @ABORT: EQU     21              ;Abort and return to TRSDOS
ØØØ13   @CKDRV: EQU     33              ;Test to see if a drive is ready
ØØØ14   @DCSTAT:EQU     4Ø              ;Verify that a drive is defined in the DCT
ØØØ15   @ERROR: EQU     26              ;Display an error message
ØØØ16   @EXIT:  EQU     22              ;Return to TRSDOS or the calling program
ØØØ17   @RDSEC: EQU     49              ;Read a sector
ØØØ18   @RDSSC: EQU     85              ;Read a system sector
ØØØ19   @WRSEC: EQU     53              ;Write a sector
ØØØ2Ø   @WRSSC: EQU     54              ;Write a system sector
ØØØ21
ØØØ22   ;       Other Equates
ØØØ23
ØØØ24   SYSSEC: EQU     14ØØH           ;The system sector is Cylinder 2Ø, Sector Ø
ØØØ25   USRSEC: EQU     ØØØØH           ;The regular sector is Cylinder Ø, Sector Ø
ØØØ26
ØØØ27   ;       First, test the target drive and make sure it is defined.
ØØØ28
ØØØ29   START:  LD      C,1             ;Select Drive 1
ØØØ3Ø           LD      A,@DCSTAT       ;Ask if the drive is listed in the DCT
ØØØ31           RST     28H             ;Call the @DCSTAT svc
ØØØ32           JR      NZ,ERROR        ;If NZ, then the drive is not defined
ØØØ33                                   ;and we will abort execution.
ØØØ34
ØØØ35   ;       Now, test and make sure the target drive contains a formatted
ØØØ36   ;       disk and is write-enabled.
ØØØ37
ØØØ38           LD      C,1             ;Select Drive 1
ØØØ39           LD      A,@CKDRV        ;Test to see if the disk is formatted
ØØØ4Ø                                   ;and is write-enabled.  Note that the
ØØØ41                                   ;disk must be formatted by TRSDOS 6.x
ØØØ42                                   ;or by LDOS 5.1.x to be considered
ØØØ43                                   ;"formatted" by this svc.
ØØØ44           RST     28H             ;Call the @CKDRV svc
ØØØ45           LD      A,8             ;This will become the error number if the
ØØØ46                                   ;drive was not ready.  This is done
ØØØ47                                   ;because the @CKDRV svc does not return error
ØØØ48                                   ;codes.
ØØØ49           JR      NZ,ERROR        ;The drive is not ready
ØØØ5Ø           LD      A,15            ;This will become the error number if the
ØØØ51                                   ;drive is ready and is write-protected.
ØØØ52                                   ;As above, this is done because @CKDRV does
ØØØ53                                   ;not return error messages.
ØØØ54           JR      C,ERROR         ;The disk is formatted, but it is
ØØØ55                                   ;write-protected.  In either case, abort.
ØØØ56
ØØØ57   ;       Now that we know the target drive is ready, read a sector
ØØØ58   ;       from the source drive and write it to the target drive (Drive 1).
ØØØ59
ØØØ6Ø           LD      C,Ø             ;Select Drive Ø
ØØØ61           LD      DE,USRSEC       ;Read the first sector on the disk,
ØØØ62                                   ;Cylinder Ø, Sector Ø.
ØØØ63           LD      HL,BUFF         ;Point to a buffer which will hold the sector
ØØØ64           LD      A,@RDSEC        ;Read a non-system sector
ØØØ65           RST     28H             ;Call the @RDSEC svc
ØØØ66           JR      NZ,ERROR        ;If NZ, an error occurred, so abort
ØØØ67
```

```
ØØØ68   ;       Now, write the sector to the target drive.
ØØØ69
ØØØ7Ø           LD      C,1             ;Select Drive 1
ØØØ71           LD      DE,USRSEC       ;Write the sector to Cylinder Ø, Sector Ø
ØØØ72                                   ;on Drive 1
ØØØ73           LD      HL,BUFF         ;Point to the buffer containing the sector
ØØØ74           LD      A,@WRSEC        ;Write the sector to disk
ØØØ75           RST     28H             ;Call the @WRSEC svc
ØØØ76           JR      NZ,ERROR        ;If NZ, an error occurred, so abort
ØØØ77
ØØØ78   ;       Now we will read a system sector from Drive Ø and write it on
ØØØ79   ;       drive 1.  The difference between a system sector and a non-system
ØØØ8Ø   ;       sector is that the Data Address Marks (DAM) are different.  These
ØØØ81   ;       were written to the disk when it was formatted.  TRSDOS 6.x uses
ØØØ82   ;       these as an extra check to make sure that a write of user data
ØØØ83   ;       does not accidentally get placed over a sector containing system
ØØØ84   ;       data.  All of the sectors in the directory cylinder are marked
ØØØ85   ;       as system sectors.
ØØØ86
ØØØ87           LD      C,Ø             ;Select Drive Ø
ØØØ88           LD      DE,SYSSEC       ;Read Cylinder 2Ø, Sector Ø
ØØØ89           LD      HL,BUFF         ;Store the sector at this address
ØØØ9Ø           LD      A,@RDSSC        ;Read a system sector
ØØØ91           RST     28H             ;Call the @RDSSC svc
ØØØ92           JR      NZ,ERROR        ;An error occurred, so abort
ØØØ93
ØØØ94   ;       Now write the sector to the target drive as a system sector.
ØØØ95   ;       There is no requirement that a sector must be placed at the
ØØØ96   ;       same cylinder and sector location as it was read from, but
ØØØ97   ;       for simplicity, we are doing that.
ØØØ98
ØØØ99           LD      C,1             ;Select Drive 1
ØØ1ØØ           LD      DE,SYSSEC       ;Write Cylinder 2Ø, Sector Ø
ØØ1Ø1           LD      HL,BUFF         ;Point to the data to be written
ØØ1Ø2           LD      A,@WRSSC        ;Write a system sector
ØØ1Ø3           RST     28H             ;Call the @WRSSC svc
ØØ1Ø4           JR      NZ,ERROR        ;An error occurred, so abort
ØØ1Ø5
ØØ1Ø6           LD      A,@EXIT         ;Return to TRSDOS or the calling program
ØØ1Ø7           RST     28H             ;Call the @EXIT svc
ØØ1Ø8
ØØ1Ø9   ;       This routine displays an error message if anything goes wrong.
ØØ11Ø   ;       Note that @CKDRV does not return an error message, so @ERROR
ØØ111   ;       cannot be used for it without some manipulation.
ØØ112
ØØ113   ERROR:  OR      ØCØH            ;Set bit 7
ØØ114           LD      C,A             ;Load error number into register C
ØØ115           LD      A,@ERROR        ;This will display the error message
ØØ116                                   ;and return to the calling program
ØØ117           RST     28H             ;Call the @ERROR svc
ØØ118
ØØ119           LD      A,@ABORT        ;Now, force an abort. This will return
ØØ12Ø                                   ;to TRSDOS Ready and will abort any
ØØ121                                   ;JCL file that is currently executing
ØØ122           RST     28H             ;Call the @ABORT svc
ØØ123
ØØ124   BUFF:   DEFS    256             ;256-byte buffer to store the sector that
ØØ125                                   ;is read and then written
ØØ126
ØØ127           END     START
```

Sample Program E

```
Ln #            Source Line

ØØØØ1   ;       This program displays the filenames of the disk in
ØØØØ2   ;       Drive Ø three different ways.
ØØØØ3
ØØØØ4           PSECT   3ØØØH           ;Program begins at x'3ØØØ'
ØØØØ6
ØØØØ7   ;       First, declare the equates for the SVCs we intend to use.
ØØØØ8   ;       This is not mandatory, but it makes the program easier to follow.
ØØØØ9
ØØØ1Ø   @CMNDI: EQU     24              ;Execute a TRSDOS command and return
ØØØ11                                   ;to TRSDOS Ready
ØØØ12   @CMNDR: EQU     25              ;Execute a TRSDOS command and return
ØØØ13                                   ;to the calling program
ØØØ14   @DODIR: EQU     34              ;Display visible filenames on the
ØØØ15                                   ;specified disk drive
ØØØ16
ØØØ17
ØØØ18   ;       First, pass a "DIR :Ø" command to the system.  TRSDOS will
ØØØ19   ;       execute this command and then return to this program.
ØØØ2Ø
ØØØ21  START:   LD      HL,DIRØ         ;Point at command we want to execute
ØØØ22           LD      A,@CMNDR        ;Execute the specified command and return
ØØØ23           RST     28H             ;Call the @CMNDR svc
ØØØ24
ØØØ25   ;       You may have noticed that the DIR displayed the files, but that
ØØØ26   ;       they were not sorted alphabetically.  This is because the DIR
ØØØ27   ;       command will not use memory above x'3ØØØ' when it is invoked with
ØØØ28   ;       a @CMNDR svc.  This prevents the DIR command from performing a
ØØØ29   ;       sort of the filenames.
ØØØ3Ø
ØØØ31
ØØØ32   ;       Now do a directory command using the @DODIR svc.
ØØØ33
ØØØ34           LD      B,Ø             ;Use Function Ø which displays all
ØØØ35                                   ;visible files in the directory.
ØØØ36           LD      C,Ø             ;Put source drive number in register C
ØØØ37           LD      A,@DODIR        ;The filenames will be read from the
ØØØ38                                   ;directory and displayed in the
ØØØ39                                   ;order they appear in the directory.
ØØØ4Ø           RST     28H             ;Call the @DODIR svc
ØØØ41
ØØØ42
ØØØ43   ;       Now pass a "DIR :Ø" command to the system.  This time
ØØØ44   ;       the command will be executed and then TRSDOS will not return
ØØØ45   ;       to this program, but will return to TRSDOS Ready.
ØØØ46
ØØØ47           LD      HL,DIRØ         ;Point at the command we want performed
ØØØ48           LD      A,@CMNDI        ;and execute it, but don't return to
ØØØ49                                   ;this program.
ØØØ5Ø           RST     28H             ;Call the @CMNDI svc
ØØØ51                                   ;This svc returns to TRSDOS Ready.
ØØØ52
ØØØ53   ;       Note that when the library command DIR is performed this time,
ØØØ54   ;       the display of files is sorted.  This is because DIR determines
ØØØ55   ;       that it was invoked with a @CMNDI svc, and it will not return
ØØØ56   ;       to the calling program.  Therefore, DIR is free to use the
ØØØ57   ;       memory above x'3ØØØ' to perform the sort of the filenames in
ØØØ58   ;       the directory.
ØØØ59
ØØØ6Ø
ØØØ61   ;       Constants
ØØØ62
ØØØ63   DIRØ:   DEFM    'DIR :Ø'        ;This command is passed to TRSDOS
ØØØ64                                   ;via the @CMNDR and @CMNDI SVCs.
ØØØ65           DEFB    ØDH             ;It must be terminated with an <ENTER>.
ØØØ66
ØØØ67           END     START
```

Sample Program F

```
Ln #            Source Line

ØØØØ1   ;       This program adds to the system task scheduler a task
ØØØØ2   ;       which displays the date and a running count of the number
ØØØØ3   ;       of times the task has been executed.
ØØØØ4   ;       For simplicity, the program tries to use task slot Ø.
ØØØØ5   ;       If it is already in use, it assumes that the task using that
ØØØØ6   ;       slot is this program, and it kills the task. It then tries to
ØØØØ7   ;       recover the memory used by the task in high memory.
ØØØØ8   ;       If the task slot is not in use, the task is placed in high memory,
ØØØØ9   ;       and the address of the task is passed to the task scheduler.
ØØØ1Ø   ;       The first time you run this program it adds the task, and the
ØØØ11   ;       next time you run this program, it removes the task.
ØØØ12
ØØØ13           PSECT   3ØØØH           ;This program starts at x'3ØØØ'
ØØØ15
ØØØ16   ;       First, declare the equates for the SVCs we intend to use.
ØØØ17   ;       This is not mandatory, but it makes the program easier to follow.
ØØØ18
ØØØ19   @ADTSK: EQU     29              ;Add a task entry to the scheduler
ØØØ2Ø   @CKTSK: EQU     28              ;Check to see if a task slot is in use
ØØØ21   @DATE:  EQU     18              ;Return the date in ASCII format
ØØØ22   @DSPLY: EQU     1Ø              ;Display a message
ØØØ23   @EXIT:  EQU     22              ;Return to TRSDOS Ready or the caller
ØØØ24   @GTMOD: EQU     83              ;Locate a memory module
ØØØ25   @HEXDEC:EQU     97              ;Convert a binary value to decimal ASCII
ØØØ26   @HIGH$: EQU     1ØØ             ;Read or modify HIGH$ or LOW$
ØØØ27   @RMTSK: EQU     3Ø              ;Remove a task entry from the scheduler
ØØØ28   @VDCTL: EQU     15              ;Perform video operations
ØØØ29   @WHERE: EQU     7               ;Find out where the program counter is
ØØØ3Ø                                   ;when this SVC is executed.  This is
ØØØ31                                   ;useful in relocatable code that must
ØØØ32                                   ;make absolute address references to
ØØØ33                                   ;call subroutines or modify data.
ØØØ34
ØØØ35
ØØØ36   ;       Below we will define a macro to simulate a call relative
ØØØ37   ;       instruction.  Since the task must be able to run no matter
ØØØ38   ;       where it is placed, it must use relative jumps and calls.
ØØØ39   ;       The Z8Ø instruction set has a jump relative (JR), but does
ØØØ4Ø   ;       not have a call relative instruction.  This can be simulated
ØØØ41   ;       using the @WHERE SVC, which returns the address of the caller
ØØØ42   ;       in a register.  This address can be adjusted and placed on
ØØØ43   ;       the stack as a return address. Then a jump relative can be used
ØØØ44   ;       to reach the subroutine.
ØØØ45
ØØØ46   CALLR:  MACRO   #1              ;#1 will be the address you want to call
ØØØ47           PUSH    HL              ;Save the registers we damage
ØØØ48           PUSH    BC              ;Save it
ØØØ49           PUSH    AF              ;Save it
ØØØ5Ø           LD      A,@WHERE        ;Get our current address
ØØØ51           RST     28H             ;Call the @WHERE svc
ØØØ52           LD      BC,3+1+1+1+1+2  ;Get the lengths of the instructions after
ØØØ53                                   ;the SVC.  This will allow the subroutine
ØØØ54                                   ;to return to the correct address.
ØØØ55           ADD     HL,BC           ;Add that offset to where we are
ØØØ56           POP     AF              ;Put stack back
ØØØ57           POP     BC              ;Restore registers
ØØØ58           EX      (SP),HL         ;Put return address on stack and restore HL
ØØØ59           JR      #1              ;Jump to the subroutine
ØØØ6Ø           ENDM                    ;End of the macro
ØØØ61
ØØØ62
ØØØ63   ;       This is the main program.  It loads at x'3ØØØ'.  It decides
ØØØ64   ;       if it needs to add or remove the task in the scheduler tables.
ØØØ65   ;       If it adds the task, it moves a copy to the top of memory and
ØØØ66   ;       protects it, and adds a task entry to the scheduler.
ØØØ67   ;       If it is removing a task, it kills the entry in the scheduler
```

```
00068   ;       tables, and then attempts to recover the memory used by the task.
00069
00070   BEGIN:  LD      C,0             ;First, we will test slot 0
00071           LD      A,@CKTSK        ;to see if anyone is using it
00072           RST     28H             ;Call the @CKTSK svc
00073           JR      NZ,KILLIT       ;There is a task using slot 0, kill it
00074
00075   ;       At this point, we want to add a task to high memory.
00076   ;       First we find the value for HIGH$ and put a copy of the
00077   ;       task there.  Then we protect the task by moving HIGH$ below
00078   ;       the new task.
00079
00080           LD      HL,0            ;First, get the value of HIGH$
00081           LD      B,H             ;Read HIGH$
00082           LD      A,@HIGH$
00083           RST     28H             ;Call the @HIGH$ svc
00084           LD      (ENDADD),HL     ;Save this value as the last address
00085                                   ;that the task will be stored in once it
00086                                   ;is moved to high memory
00087
00088           LD      DE,HL           ;Put that value here
00089           LD      HL,MODEND-1     ;Point at the end of the module
00090           LD      BC,MODEND-MODULE;Move the module from where it is
00091                                   ;right now to a position below HIGH$
00092           LDDR                    ;Do the copy
00093
00094           LD      HL,DE           ;Now protect the module using HIGH$
00095           LD      B,0             ;Update HIGH$
00096           LD      A,@HIGH$
00097           RST     28H             ;Call the @HIGH$ svc
00098
00099   ;       Now we need to load the TCB entry in the module with the address
00100   ;       of the first instruction to be executed.
00101
00102           LD      IX,HL           ;IX now points at memory header
00103           LD      BC,ENTRY-MODULE+1       ;Get the offset into the module
00104                                           ;of the first instruction
00105           ADD     HL,BC           ;HL now contains the actual starting address
00106           LD      (IX+(1+MODTCB-MODULE)),L        ;Store LSB of the address
00107           LD      (IX+1+(1+MODTCB-MODULE)),H      ;Store MSB of the address
00108
00109   ;       Now the task is ready to run.  We now add the entry to the task
00110   ;       scheduler table.
00111
00112           LD      BC,MODTCB-MODULE+1      ;Get offset into the
00113                                           ;module of the TCB word
00114           PUSH    IX              ;Get a copy of the base address
00115           POP     HL              ;Put base address here
00116           ADD     HL,BC           ;Now HL points at TCB address
00117           LD      DE,HL           ;Put that value in DE
00118           LD      C,0             ;Add this entry to task slot 0
00119           LD      A,@ADTSK        ;Add this task, to be run every 266.67 msec
00120           RST     28H             ;Call the @ADTSK svc
00121
00122   ;       The main program has now done its work and can exit.
00123
00124           LD      HL,ADDED        ;Point at a message saying what was done
00125           LD      A,@DSPLY        ;and print it
00126           RST     28H             ;Call the @DSPLY svc
00127
00128           LD      A,@EXIT         ;Now exit
00129           RST     28H             ;Call the @EXIT svc
00130
00131   ;       This SVC does not return.
00132
00133
00134   ;       This part of the code removes the task from the scheduler
00135   ;       tables and then attempts to recover the memory that was used
```

```
ØØ136   ;       by the task in high memory.  If another high memory module
ØØ137   ;       was added AFTER this task was added, then the memory that
ØØ138   ;       was used by this task cannot be recovered.
ØØ139
ØØ14Ø   KILLIT: LD      C,Ø             ;We want to remove the task in slot Ø
ØØ141           LD      A,@RMTSK
ØØ142           RST     28H             ;Call the @RMTSK svc
ØØ143
ØØ144   ;       At this point, the task is no longer called by the operating
ØØ145   ;       system.  Now we want to determine if we can
ØØ146   ;       reclaim the memory it was using.
ØØ147
ØØ148           LD      DE,MODNAM       ;Point at the name of the module
ØØ149           LD      A,@GTMOD        ;Look for a module with that name
ØØ15Ø           RST     28H             ;Call the @GTMOD svc
ØØ151           JR      NZ,CANT         ;If NZ is set, then we killed some other
ØØ152                                   ;task that was using slot Ø.  Oops.
ØØ153                                   ;In that case, just stop and don't do any
ØØ154                                   ;more damage.
ØØ155           LD      IX,HL           ;Set IX to point to the module.
ØØ156           LD      B,Ø             ;Read the current value of HIGH$
ØØ157           LD      HL,Ø            ;to see if this is the first program in
ØØ158                                   ;high memory
ØØ159           LD      A,@HIGH$        ;If it is, then we can recover the space
ØØ16Ø           RST     28H             ;Call the @HIGH$ svc
ØØ161           INC     HL              ;Move HIGH$ up by one byte
ØØ162           PUSH    IX              ;Take the address of our module
ØØ163           POP     DE              ;and store it here
ØØ164           XOR     A               ;Compare these
ØØ165           SBC     HL,DE           ;Are they the same?
ØØ166           JR      NZ,CANT         ;No, the high memory module can't be removed
ØØ167
ØØ168   ;       At this point, we know it is ok to reclaim the memory used by the
ØØ169   ;       high memory task.
ØØ17Ø
ØØ171           LD      HL,(IX+2)       ;Read the end of module value out of the
ØØ172                                   ;header information
ØØ173           LD      B,Ø             ;Update the HIGH$ value
ØØ174           LD      A,@HIGH$
ØØ175           RST     28H             ;Call the @HIGH$ svc
ØØ176
ØØ177           LD      HL,OK           ;Point to a message saying all is well
ØØ178           LD      A,@DSPLY        ;and print it
ØØ179           RST     28H             ;Call the @DSPLY svc
ØØ18Ø
ØØ181           LD      A,@EXIT         ;Exit the main program
ØØ182           RST     28H             ;Call the @EXIT svc
ØØ183
ØØ184
ØØ185   ;       Here we will display a message saying we removed the task from
ØØ186   ;       the scheduler table, but we cannot reclaim the memory that was
ØØ187   ;       used.
ØØ188
ØØ189   CANT:   LD      HL,RECLM        ;Point to the message
ØØ19Ø           LD      A,@DSPLY        ;and display it
ØØ191           RST     28H             ;Call the @DSPLY svc
ØØ192
ØØ193           LD      A,@EXIT         ;Now exit
ØØ194           RST     28H             ;Call the @EXIT svc
ØØ195
ØØ196
ØØ197   ;       Messages
ØØ198
ØØ199   ADDED:  DEFM    'Task placed in high memory and scheduled.'
ØØ2ØØ           DEFB    ØDH
ØØ2Ø1   OK:     DEFM    'Task removed from scheduler table and memory reclaimed.'
ØØ2Ø2           DEFB    ØDH
ØØ2Ø3   RECLM:  DEFM    'Task removed from scheduler table, but memory could not '
```

```
ØØ2Ø4            DEFM    'be recovered.'
ØØ2Ø5            DEFB    ØDH
ØØ2Ø6
ØØ2Ø7   ;        The Task begins at this point.  This part of the program loads
ØØ2Ø8   ;        in low memory but is relocated to a point just below HIGH$.
ØØ2Ø9
ØØ21Ø   ;        This is the Memory Header Block.  This block of data allows
ØØ211   ;        the system to locate this module in memory by name,
ØØ212   ;        using the @GTMOD svc.
ØØ213
ØØ214   MODULE: JR      ENTRY            ;Jump (relative) to the starting address
ØØ215   ENDADD: DEFW    Ø                ;The highest address in the program.
ØØ216                                    ;This value is patched in before the program
ØØ217                                    ;is relocated.  This will be used
ØØ218                                    ;later in recovering the memory used by
ØØ219                                    ;this task.
ØØ22Ø            DEFB    MODTCB-MODNAM    ;Number of bytes in the name field below.
ØØ221   MODNAM: DEFM    'UPTIME'         ;This is the name of the module and is
ØØ222                                    ;used to identify the module.
ØØ223   MODTCB: DEFW    Ø                ;Actual address to start execution.  This
ØØ224                                    ;value is patched in after the program is
ØØ225                                    ;relocated.
ØØ226            DEFW    Ø                ;Spare system pointer - RESERVED
ØØ227
ØØ228   ;        This area contains data used by the task.  It is addressed using
ØØ229   ;        the IX register which points to the task when it is executed.
ØØ23Ø
ØØ231   COUNTER:DEFW    Ø                ;Count of how many times we have run
ØØ232   DATBUF: DEFS    9                ;The date is stored here
ØØ233
ØØ234   ;        This is the actual task.
ØØ235   ;        On entry to the task, IX points at the Task Control Block (TCB),
ØØ236   ;        which in this program is the label 'MODTCB'.  All data is
ØØ237   ;        referenced by indexing from that address.
ØØ238
ØØ239
ØØ24Ø   ENTRY:   PUSH    IY               ;Save this register.  It is not saved by
ØØ241                                    ;the Task Scheduler, and we use it.
ØØ242                                    ;Registers AF, BC, DE, and HL are saved
ØØ243
ØØ244   ;        Now we will read the current date.
ØØ245
ØØ246            LD      HL,IX            ;Get a copy of the index pointer
ØØ247            LD      BC,DATBUF-MODTCB;Get the offset needed to access the date
ØØ248            ADD     HL,BC            ;Now we have a pointer to the date
ØØ249
ØØ25Ø            PUSH    IX               ;Save the pointer to the start of the task
ØØ251            PUSH    HL               ;Save a copy of that pointer
ØØ252            LD      A,@DATE          ;Ask the system what the date is
ØØ253            RST     28H              ;Call the @DATE svc
ØØ254
ØØ255            LD      (HL),Ø           ;Terminate the date string
ØØ256
ØØ257            POP     DE               ;Put pointer to the date here
ØØ258            PUSH    DE               ;We will use this pointer later on
ØØ259            LD      HL,ØØ28H         ;Put the cursor on the top line,
ØØ26Ø                                     ;specified in register HL
ØØ261                                     ;at the 41st position on the screen
ØØ262            CALLR   WRITE            ;Write the message at the position
+                PUSH    HL               ;Save the registers we damage
+                PUSH    BC               ;Save it
+                PUSH    AF               ;Save it
+                LD      A,@WHERE         ;Get our current address
+                RST     28H              ;Call the @WHERE svc
+                LD      BC,3+1+1+1+1+2   ;Get the lengths of the instructions after
+                                         ;the SVC.  This will allow the subroutine
+                                         ;to return to the correct address.
```

```
+                ADD     HL,BC              ;Add that offset to where we are
+                POP     AF                 ;Put stack back
+                POP     BC                 ;Restore registers
+                EX      (SP),HL            ;Put return address on stack and restore HL
+                JR      WRITE              ;Jump to the subroutine
ØØ263                                       ;Note that the above was actually a macro
ØØ264                                       ;which performs a relative call.
ØØ265
ØØ266    ;       This part of the task displays a count of the number of times
ØØ267    ;       the task has been executed.
ØØ268
ØØ269           POP     DE                 ;Get the pointer to DATBUF back
ØØ27Ø           POP     IX                 ;Get the pointer to the beginning of
ØØ271                                       ;this task
ØØ272           PUSH    DE                 ;Save the pointer to DATBUF again
ØØ273           LD      BC,COUNTER-MODTCB        ;Get the offset to our data
ØØ274                                            ;area
ØØ275           LD      HL,IX              ;Put a copy of the base address in HL
ØØ276           ADD     HL,BC              ;Add offset.  Now HL points to COUNTER:
ØØ277           LD      IY,HL              ;Put the pointer to COUNTER in IY
ØØ278           LD      L,(IY)             ;Get LSB of the counter
ØØ279           LD      H,(IY+1)           ;Get MSB of the counter
ØØ28Ø           INC     HL                 ;Increment the number of times we have run
ØØ281           LD      (IY),L             ;Store the LSB of the counter
ØØ282           LD      (IY+1),H           ;Store the MSB of the counter
ØØ283
ØØ284           LD      A,@HEXDEC          ;Convert the count to decimal
ØØ285           RST     28H                ;Call the @HEXDEC svc
ØØ286
ØØ287           XOR     A                  ;Get a zero
ØØ288           LD      (DE),A             ;Terminate the count string
ØØ289
ØØ29Ø           POP     DE                 ;Put pointer to date here
ØØ291           LD      HL,ØØ36H           ;Put the cursor on the top line,
ØØ292                                       ;specified in register HL
ØØ293                                       ;at the 55th position on the screen
ØØ294           CALLR   WRITE              ;Write the message at the position
+                PUSH    HL                 ;Save the registers we damage
+                PUSH    BC                 ;Save it
+                PUSH    AF                 ;Save it
+                LD      A,@WHERE           ;Get our current address
+                RST     28H                ;Call the @WHERE svc
+                LD      BC,3+1+1+1+1+2     ;Get the lengths of the instructions after
+                                           ;the SVC.  This will allow the subroutine
+                                           ;to return to the correct address.
+                ADD     HL,BC              ;Add that offset to where we are
+                POP     AF                 ;Put stack back
+                POP     BC                 ;Restore registers
+                EX      (SP),HL            ;Put return address on stack and restore HL
+                JR      WRITE              ;Jump to the subroutine
ØØ295                                       ;Note that the above was actually a macro
ØØ296                                       ;which performs a relative call.
ØØ297
ØØ298    ;       Now we restore the IY register and return to the task scheduler.
ØØ299
ØØ3ØØ           POP     IY                 ;Restore IY value
ØØ3Ø1           RET                        ;Return to the task scheduler
ØØ3Ø2
ØØ3Ø3
ØØ3Ø4    ;       This routine places characters on the display using the @VDCTL
ØØ3Ø5    ;       svc instead of @DSP or @DSPLY.  This allows the cursor to
ØØ3Ø6    ;       remain at its current position when we write to the screen.
ØØ3Ø7    ;       This routine must be called using the relocatable call macro
ØØ3Ø8    ;       CALLR.
ØØ3Ø9
ØØ31Ø   WRITE:   LD      B,2                ;Put character on the display
ØØ311
ØØ312   TSKLP:   LD      A,(DE)             ;Get a character to display
```

Sample Program F, continued

```
ØØ313            OR      A              ;Is it time to stop putting this on
ØØ314                                   ;the display?
ØØ315            RET     Z              ;Yes, return to the caller
ØØ316            PUSH    HL             ;Save the registers, as the SVC will
ØØ317            PUSH    DE             ;alter the contents
ØØ318            PUSH    BC
ØØ319            LD      C,A            ;Put the character here
ØØ32Ø            LD      A,@VDCTL       ;Put character on screen at specified position
ØØ321            RST     28H            ;Call the @VDCTL svc
ØØ322            POP     BC             ;Restore registers
ØØ323            POP     DE
ØØ324            POP     HL
ØØ325            INC     L              ;Advance display position
ØØ326            INC     DE             ;Point to next character to display
ØØ327            JR      TSKLP          ;Loop till date is completely displayed
ØØ328
ØØ329   MODEND:  END     BEGIN          ;End of task and main program
```

9/Technical Information on TRSDOS Commands and Utilities

TRSDOS commands and utilities are covered extensively in the *Disk System Owner's Manual*. This section presents additional information of a technical nature on several of the commands and utilities.

Changing the Step Rate

The step rate is the rate at which the drive head moves from cylinder to cylinder. You can change the step rate for any drive by using one of the commands described below.

To set the step rate for a particular drive, use the following command:

SYSTEM (DRIVE = *drive*, STEP = *number*)

drive is any drive enabled in the system. *number* can be 0, 1, 2, or 3 and represents one of the following step rates in milliseconds:

0 = 6 milliseconds
1 = 12 milliseconds
2 = 20 milliseconds
3 = 30 milliseconds

Unless it is SYSGENed, the step value you select remains in effect for the specified drive only until the system is re-booted or turned off. If you use the SYSGEN command while the step value is in effect, then this step rate is written to the configuration file (CONFIG/SYS) on the disk in the drive specified by the SYSGEN command.

On a new TRSDOS disk, the step rate is set to 12 milliseconds.

To set the default bootstrap step rate used with the FORMAT utility, use the following command:

SYSTEM (BSTEP = *number*)

number is 0, 1, 2, or 3, which correspond to 6, 12, 20, and 30 milliseconds, respectively.

The value you select for *number* is stored in the system information sector on the disk in Drive 0. (On a new TRSDOS disk, the bootstrap step rate is set to 12 milliseconds.)

If you switch Drive 0 disks or change the logical Drive 0 with the SYSTEM (SYSTEM) command, the default value is taken off the new Drive 0 disk if you format a disk.

You can change the bootstrap step rate for a particular FORMAT operation if you do not want to use the default. Specify the new value for STEP on the FORMAT command line as follows:

FORMAT *:drive* (STEP = *number*)

drive is the drive to be used for the FORMAT. *number* is 0, 1, 2, or 3, which correspond to 6, 12, 20, and 30 milliseconds, respectively.

The step rate is important only if you will be using the disk in Drive 0 to start up the system. Keep in mind that too low a step rate may keep the disk from booting.

Changing the WAIT Value

The WAIT parameter compensates for hardware incompatibility between certain disk drives. The only time you should use it is when *all* tracks above a certain point during a FORMAT operation are shown as locked out when the FORMAT is verified.

The value assigned to WAIT signifies the amount of time between the arrival of the drive head at the location for a read or write, and the actual start of the read or write.

If you want to change the WAIT value, specify the new value on the FORMAT command line as follows:

FORMAT *:drive* (WAIT = *number*)

number is a value between 5000 and 50000. The exact value depends on the particular disk drive you are using. We recommend that you use a value around 25000 at first. Adjust this value higher if tracks are still locked out, or lower until the bottom limit is determined.

Logging in a Diskette

LOG is a utility program that logs in the directory track, number of sides, and density of a diskette. The syntax is:

LOG *:drive*

drive is any drive currently enabled in the system.

The LOG utility provides a way to log in diskette information and update the drive's Drive Code Table (DCT). It performs the same log-in function as the DEVICE library command, except for a single drive rather than all drives. It also provides a way to swap the Drive 0 diskette for a double-sided diskette.

The LOG :0 command prompts you to switch the Drive 0 diskette. You must use this command when switching between double- and single-sided diskettes in Drive 0. Otherwise, it is not needed.

Example

If you want to switch disks in Drive 0, type:

```
LOG :0 (ENTER)
```

The system prompts you with the message:

```
Exchange disks and hit <ENTER>
```

Remove the current disk from Drive 0 and insert the new system disk. When you press (ENTER), information about the new disk is entered to the system.

Printing Graphics Characters

If your printer is capable of directly reproducing the TRS-80 graphics characters, you can use the SYSTEM (GRAPHIC) command. Once you have issued this command, any graphics characters on the screen will be sent to the line printer during a screen print. (Pressing (CTRL)(:) causes the contents of the video display to be printed on the printer.)

Do not use this command unless your printer is capable of directly reproducing the TRS-80 graphics characters.

Changing the Clock Rate

The system normally runs at the fast clock rate of 4 megahertz.

A slow mode of 2 megahertz is available, and may be necessary for real time-dependent programs. (This slow rate is the same as the Model III clock rate.)

To switch to the slow rate, enter the following command:

```
SYSTEM (SLOW)
```

To switch back to the fast rate, enter:

```
SYSTEM (FAST)
```

Appendix A/TRSDOS Error Messages

If the computer displays one of the messages listed in this appendix, an operating system error occurred. Any other error message may refer to an application program error, and you should check your application program manual for an explanation.

When an error message is displayed:

- Try the operation several times.
- Look up operating system errors below and take any recommended actions. (See your application program manual for explanations of application program errors.)
- Try using other diskettes.
- Reset the computer and try the operation again.
- Check all the power connections.
- Check all interconnections.
- Remove all diskettes from drives, turn off the computer, wait 15 seconds, and turn it on again.
- If you try all these remedies and still get an error message, contact a Radio Shack Service Center.

Note: If there is more than one thing wrong, the computer might wait until you correct the first error before displaying the second error message.

This list of error messages is alphabetical, with the binary and hexadecimal error numbers in parentheses. Following it is a quick reference list of the messages arranged in numerical order.

Attempted to read locked/deleted data record (Error 7, X'07')

In a system that supports a "deleted record" data address mark, an attempt was made to read a deleted sector. TRSDOS currently does not use the deleted sector data address mark. Check for an error in your application program.

Attempted to read system data record (Error 6, X'06')

An attempt was made to read a directory cylinder sector without using the directory read routines. Directory cylinder sectors are written with a data address mark that differs from the data sector's data address mark. Check for an error in your application program.

Data record not found during read (Error 5, X'05')

The sector number for the read operation is not on the cylinder being referenced. Either the disk is flawed, you requested an incorrect number, or the cylinder is improperly formatted. Try the operation again. If it fails, use another disk. Reformatting the old disk should lock out the flaw.

Data record not found during write (Error 13, X'0D')

The sector number requested for the write operation cannot be found on the cylinder being referenced. Either the disk is flawed, you requested an incorrect number, or the cylinder is improperly formatted. Try the operation again. If it fails, use another disk.

Device in use (Error 39, X'27')

A request was made to REMOVE a device (delete it from the Device Control Block tables) while it was in use. RESET the device in use before removing it.

Device not available (Error 8, X'08')

A reference was made for a logical device that cannot be found in the Device Control Block. Probably, your device specification was wrong or the device peripheral was not ready. Use the DEVICE command to display all devices available to the system.

Directory full — can't extend file (Error 30, X'1E')

A file has all extent fields of its last directory record in use and must find a spare directory slot but none is available. (See the "Directory Records" section.) Copy the disk's files to a newly formatted diskette to reduce file fragmentation. You may use backup by class or backup reconstruct to reduce fragmentation.

Directory read error (Error 17, X'11')

A disk error occurred during a directory read. The problem may be media, hardware, or program failure. Move the disk to another drive and try the operation again.

Directory write error (Error 18, X'12')

A disk error occurred during a directory write to disk. The directory may no longer be reliable. If the problem recurs, use a different diskette.

Disk space full (Error 27, X'1B')

While a file was being written, all available disk space was used. The disk contains only a partial copy of the file. Write the file to a diskette that has more available space. Then, REMOVE the partial copy to recover disk space.

End of file encountered (Error 28, X'1C')

You tried to read past the end of file pointer. Use the DIR command to check the size of the file. This error also occurs when you use the @PEOF supervisor call to successfully position to the end of a file. Check for an error in your application program.

Extended error (Error 63)

An error has occurred and the extended error code is in the HL register pair.

File access denied (Error 25, X'19')

You specified a password for a file that is not password protected or you specified the wrong password for a file that is password protected.

File already open (Error 41, X'29')

You tried to open a file for UPDATE level or higher, and the file already is open with this access level or higher. This forces a change to READ access protection. Use the RESET library command to close the file.

File not in directory (Error 24, X'18')

The specified filespec cannot be found in the directory. Check the spelling of the filespec.

File not open (Error 38, X'26')

You requested an I/O operation on an unopened file. Open the file before access.

GAT read error (Error 20, X'14')

A disk error occurred during the reading of the Granule Allocation Table. The problem may be media, hardware, or program failure. Move the diskette to another drive and try the operation again.

GAT write error (Error 21, X'15')

A disk error occurred during the writing of the Granule Allocation Table. The GAT may no longer be reliable. If the problem recurs, use a different drive or different diskette.

HIT read error (Error 22, X'16')

A disk error occurred during the reading of the Hash Index Table. The problem may be media, hardware, or program failure. Move the diskette to another drive and try the operation again.

HIT write error (Error 23, X'17')

A disk error occurred during the writing of the Hash Index Table. The HIT may no longer be reliable. If the problem recurs, use a different drive or different diskette.

Illegal access attempted to protected file (Error 37, X'25')

The USER password was given for access to a file, but the requested access required the OWNER password. (See the ATTRIB library command in your *Disk System Owner's Manual*.)

Illegal drive number (Error 32, X'20')

The specified disk drive is not included in your system or is not ready for access (no diskette, non-TRSDOS diskette, drive door open, and so on). See the DEVICE command in your *Disk System Owner's Manual*.)

Illegal file name (Error 19, X'13')

The specified filespec does not meet TRSDOS filespec requirements. See your *Disk System Owner's Manual* for proper filespec syntax.

Illegal logical file number (Error 16, X'10')

A bad Directory Entry Code (DEC) was found in the File Control Block (FCB). This usually indicates that your program has altered the FCB improperly. Check for an error in your application program.

Load file format error (Error 34, X'22')

An attempt was made to load a file that cannot be loaded by the system loader. The file was probably a data file or a BASIC program file.

Lost data during read (Error 3, X'03')

During a sector read, the CPU did not accept a byte from the Floppy Disk Controller (FDC) data register in the time allotted. The byte was lost. This may indicate a hardware problem with the drive. Move the diskette to another drive and try again. If the error recurs, try another diskette.

Lost data during write (Error 11, X'0B')

During a sector write, the CPU did not transfer a byte to the Floppy Disk Controller (FDC) in the time allotted. The byte was lost; it was not transferred to the disk. This may indicate a hardware problem with the drive. Move the diskette to another drive and try again. If the error recurs, try another diskette.

LRL open fault (Error 42, X'2A')

The logical record length specified when the file was opened is different than the LRL used when the file was created. COPY the file to another file that has the specified LRL.

No device space available (Error 33, X'21')

You tried to SET a driver or filter and all of the Device Control Blocks were in use. Use the DEVICE command to see if any non-system devices can be removed to provide more space. This error also occurs on a "global" request to initialize a new file (that is, no drive was specified), if no file can be created.

No directory space available (Error 26, X'1A')

You tried to open a new file and no space was left in the directory. Use a different disk or REMOVE some files that you no longer need.

No error (Error 0)

The @ERROR supervisor call was called without any error condition being detected. A return code of zero indicates no error. Check for an error in your application program.

Parity error during header read (Error 1, X'01')

During a sector I/O request, the system could not read the sector header successfully. If this error occurs repeatedly, the problem is probably media or hardware failure. Try the operation again, using a different drive or diskette.

Parity error during header write (Error 9, X'09')

During a sector write, the system could not write the sector header satisfactorily. If this error occurs repeatedly, the problem is probably media or hardware failure. Try the operation again, using a different drive or diskette.

Parity error during read (Error 4, X'04')

An error occurred during a sector read. Its probable cause is media failure or a dirty or faulty disk drive. Try the operation again, using a different drive or diskette.

Parity error during write (Error 12, X'0C')

An error occurred during a sector write operation. Its probable cause is media failure or a dirty or faulty disk drive. Try the operation again, using a different drive or diskette.

Program not found (Error 31, X'1F')

The file cannot be loaded because it is not in the directory. Either the filespec was misspelled or the disk that contains the file was not loaded.

Protected system device (Error 40, X'28')

You cannot REMOVE any of the following devices: *KI, *DO, *PR, *JL, *SI, *SO. If you try, you get this error message.

Record number out of range (Error 29, X'1D')

A request to read a record within a random access file (see the @POSN supervisor call) provided a record number that was beyond the end of the file. Correct the record number or try again using another copy of the file.

Seek error during read (Error 2, X'02')

During a read sector disk I/O request, the cylinder that should contain the sector was not found within the time allotted. (The time is set by the step rate specified in the Drive Code Table.) Either the cylinder is not formatted or it is no longer readable, or the step rate is too low for the hardware to respond. You can set an appropriate step rate using the SYSTEM library command. The problem may also be caused by media or hardware failure. In this case, try the operation again, using a different drive or diskette.

Seek error during write (Error 10, X'0A')

During a sector write, the cylinder that should contain the sector was not found within the time allotted. (The time is set by the step rate specified in the Drive Code Table.) Either the cylinder is not formatted or it is no longer readable, or the step rate is too low for the hardware to respond. You can set an appropriate step rate using the SYSTEM library command. The problem may also be caused by media or hardware failure. In this case, try the operation again, using a different drive or diskette.

— Unknown error code

The @ERROR supervisor call was called with an error number that is not defined. Check for an error in your application program.

Write fault on disk drive (Error 14, X'0E')

An error occurred during a write operation. This probably indicates a hardware problem. Try a different diskette or drive. If the problem continues, contact a Radio Shack Service Center.

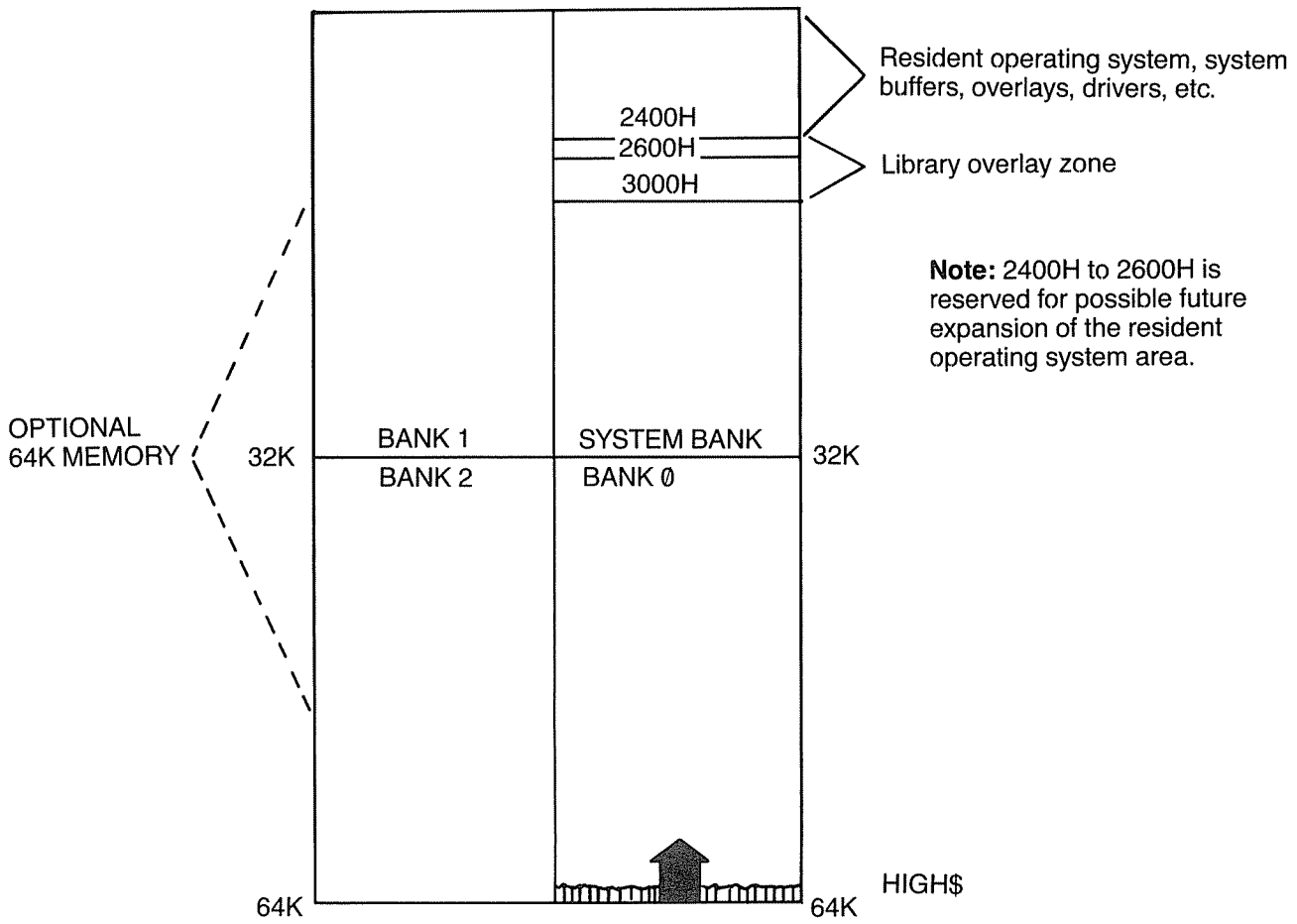
Write protected disk (Error 15, X'0F')

You tried to write to a drive that has a write-protected diskette or is software write-protected. Remove the write-protect tab, if the diskette has one. If it does not, use the DEVICE command to see if the drive is set as write protected. If it is, you can use the SYSTEM library command with the (WP=OFF) parameter to write enable the drive. If the problem recurs, use a different drive or different diskette.

Numerical List of Error Messages

Decimal	Hex	Message
0	X'00'	No Error
1	X'01'	Parity error during header read
2	X'02'	Seek error during read
3	X'03'	Lost data during read
4	X'04'	Parity error during read
5	X'05'	Data record not found during read
6	X'06'	Attempted to read system data record
7	X'07'	Attempted to read locked/deleted data record
8	X'08'	Device not available
9	X'09'	Parity error during header write
10	X'0A'	Seek error during write
11	X'0B'	Lost data during write
12	X'0C'	Parity error during write
13	X'0D'	Data record not found during write
14	X'0E'	Write fault on disk drive
15	X'0F'	Write protected disk
16	X'10'	Illegal logical file number
17	X'11'	Directory read error
18	X'12'	Directory write error
19	X'13'	Illegal file name
20	X'14'	GAT read error
21	X'15'	GAT write error
22	X'16'	HIT read error
23	X'17'	HIT write error
24	X'18'	File not in directory
25	X'19'	File access denied
26	X'1A'	Full or write protected disk
27	X'1B'	Disk space full
28	X'1C'	End of file encountered
29	X'1D'	Record number out of range
30	X'1E'	Directory Full—can't extend file
31	X'1F'	Program not found
32	X'20'	Illegal drive number
33	X'21'	No device space available
34	X'22'	Load file format error
37	X'25'	Illegal access attempted to protected file
38	X'26'	File not open
39	X'27'	Device in use
40	X'28'	Protected system device
41	X'29'	File already open
42	X'2A'	LRL open fault
43	X'2B'	SVC parameter error
63	X'3F'	Extended error
—		Unknown error code

Appendix B/Memory Map



All software must observe HIGH$.

User software which does not allow TRSDOS library commands to be executed during run time may use memory from 2600H to HIGH$.

User software which allows for library commands during execution must reside in and use memory only between 3000H and HIGH$.

TRSDOS provides all functions and storage through supervisor calls. No address or entry point below 3000H is documented by Radio Shack.

Appendix C/Character Codes

Text, control functions, and graphics are represented in the computer by codes. The character codes range from zero through 255.

Codes one through 31 normally represent certain control functions. For example, code 13 represents a carriage return or "end of line." These same codes also represent special characters. To display the special character that corresponds to a particular code (1-31), precede the code with a code zero.

Codes 32 through 127 represent the text characters — all those letters, numbers, and other characters that are commonly used to represent textual information.

Codes 128 through 191, when output to the video display, represent 64 graphics characters.

Codes 192 through 255, when output to the video display, represent either space compression codes or special characters, as determined by software.

ASCII Character Set

Code Dec.	Code Hex.	ASCII Abbrev.	Keyboard	Video Display
0	00	NUL	CTRL @	Treat next character as displayable; if in the range 1-31, a special character is displayed (see list of special characters later in this Appendix).
1	01	SOH	CTRL A	
2	02	STX	CTRL B	
3	03	ETX	CTRL C	
4	04	EOT	CTRL D	
5	05	ENQ	CTRL E	
6	06	ACK	CTRL F	
7	07	BEL	CTRL G	
8	08	BS	← CTRL H	Backspace and erase
9	09	HT	→ CTRL I	
10	0A	LF	↓ CTRL J	Move cursor to start of next line
11	0B	VT	↑ CTRL K	
12	0C	FF	CTRL L	
13	0D	CR	ENTER CTRL M	Move cursor to start of next line
14	0E	SO	CTRL N	Turn cursor on
15	0F	SI	CTRL O	Turn cursor off
16	10	DLE	CTRL P	Enable reverse video and set high bit routine on*
17	11	DC1	CTRL Q	Set reverse video high bit routine off*
18	12	DC2	CTRL R	
19	13	DC3	CTRL S	
20	14	DC4	CTRL T	
21	15	NAK	CTRL U	Swap space compression/special characters
22	16	SYN	CTRL V	Swap special/alternate characters
23	17	ETB	CTRL W	Set to 40 characters per line
24	18	CAN	SHIFT ← CTRL X	Backspace without erasing
25	19	EM	SHIFT → CTRL Y	Advance cursor
26	1A	SUB	SHIFT ↓ CTRL Z	Move cursor down
27	1B	ESC	SHIFT ↑ CTRL ,	Move cursor up
28	1C	FS	CTRL /	Move cursor to upper left corner. Disable reverse video and set high bit routine off.* Set to 80 characters per line.
29	1D	GS	CTRL ENTER CTRL .	Erase line and start over
30	1E	RS	CTRL ;	Erase to end of line

*When the high bit routine is on, characters 128 through 191 are displayed as standard ASCII characters in reverse video.

Code Dec.	Code Hex.	ASCII Abbrev.	Keyboard	Video Display
31	1F	VS	SHIFT CLEAR	Erase to end of display
32	20	SPA	SPACEBAR	(blank)
33	21		!	!
34	22		"	"
35	23		#	#
36	24		$	$
37	25		%	%
38	26		&	&
39	27		'	'
40	28		(	(
41	29		)	)
42	2A		*	*
43	2B		+	+
44	2C		,	,
45	2D		-	–
46	2E		.	•
47	2F		/	/
48	30		0	0
49	31		1	1
50	32		2	2
51	33		3	3
52	34		4	4
53	35		5	5
54	36		6	6
55	37		7	7
56	38		8	8
57	39		9	9
58	3A		:	:
59	3B		;	;
60	3C		<	<
61	3D		=	=
62	3E		>	>
63	3F		?	?
64	40		@	@
65	41		SHIFT A	A
66	42		SHIFT B	B
67	43		SHIFT C	C
68	44		SHIFT D	D
69	45		SHIFT E	E
70	46		SHIFT F	F
71	47		SHIFT G	G
72	48		SHIFT H	H
73	49		SHIFT I	I
74	4A		SHIFT J	J
75	4B		SHIFT K	K
76	4C		SHIFT L	L
77	4D		SHIFT M	M
78	4E		SHIFT N	N
79	4F		SHIFT O	O
80	50		SHIFT P	P
81	51		SHIFT Q	Q
82	52		SHIFT R	R
83	53		SHIFT S	S
84	54		SHIFT T	T
85	55		SHIFT U	U
86	56		SHIFT V	V
87	57		SHIFT W	W
88	58		SHIFT X	X
89	59		SHIFT Y	Y

Code Dec.	Code Hex.	ASCII Abbrev.	Keyboard	Video Display
90	5A		SHIFT Z	Z
91	5B		CLEAR ,	[
92	5C		CLEAR /	\
93	5D		CLEAR .	]
94	5E		CLEAR ;	^
95	5F		CLEAR ENTER	—
96	60		SHIFT @	`
97	61		A	a
98	62		B	b
99	63		C	c
100	64		D	d
101	65		E	e
102	66		F	f
103	67		G	g
104	68		H	h
105	69		I	i
106	6A		J	j
107	6B		K	k
108	6C		L	l
109	6D		M	m
110	6E		N	n
111	6F		O	o
112	70		P	p
113	71		Q	q
114	72		R	r
115	73		S	s
116	74		T	t
117	75		U	u
118	76		V	v
119	77		W	w
120	78		X	x
121	79		Y	y
122	7A		Z	z
123	7B		CLEAR SHIFT ,	{
124	7C		CLEAR SHIFT /	|
125	7D		CLEAR SHIFT .	}
126	7E		CLEAR SHIFT ;	~
127	7F	DEL	CLEAR SHIFT ENTER	±

Extended (non-ASCII) Character Set

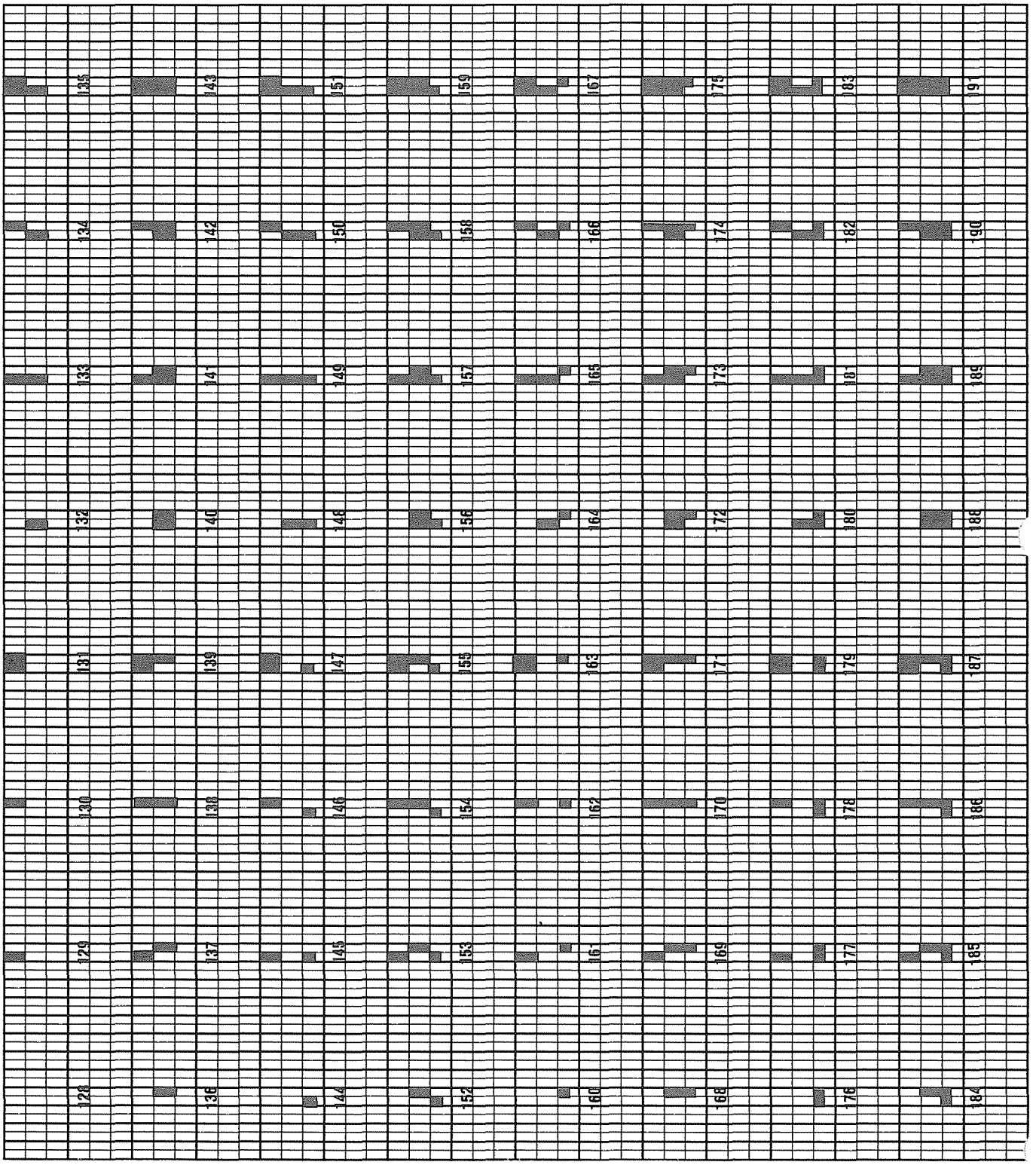
Code Dec.	Code Hex.	Keyboard	Video Display
128	80	BREAK	See graphics character table in this Appendix.
129	81	F1 CLEAR CTRL A	
130	82	F2 CLEAR CTRL B	
131	83	F3 CLEAR CTRL C	
132	84	CLEAR CTRL D	
133	85	CLEAR CTRL E	
134	86	CLEAR CTRL F	
135	87	CLEAR CTRL G	
136	88	CLEAR CTRL H	
137	89	CLEAR CTRL I	
138	8A	CLEAR CTRL J	
139	8B	CLEAR CTRL K	
140	8C	CLEAR CTRL L	
141	8D	CLEAR CTRL M	
142	8E	CLEAR CTRL N	
143	8F	CLEAR CTRL O	
144	90	CLEAR CTRL P	
145	91	SHIFT F1 CLEAR CTRL Q	
146	92	SHIFT F2 CLEAR CTRL R	
147	93	SHIFT F3 CLEAR CTRL S	
148	94	CLEAR CTRL T	
149	95	CLEAR CTRL U	
150	96	CLEAR CTRL V	
151	97	CLEAR CTRL W	
152	98	CLEAR CTRL X	
153	99	CLEAR CTRL Y	
154	9A	CLEAR CTRL Z	
155	9B	CLEAR SHIFT ↑	
156	9C		
157	9D		
158	9E		
159	9F		
160	A0	CLEAR SPACE	
161	A1	CLEAR SHIFT 1	
162	A2	CLEAR SHIFT 2	
163	A3	CLEAR SHIFT 3	
164	A4	CLEAR SHIFT 4	
165	A5	CLEAR SHIFT 5	
166	A6	CLEAR SHIFT 6	
167	A7	CLEAR SHIFT 7	
168	A8	CLEAR SHIFT 8	
169	A9	CLEAR SHIFT 9	
170	AA	CLEAR SHIFT :	
171	AB		
172	AC		
173	AD	CLEAR -	
174	AE		
175	AF		
176	B0	CLEAR 0	
177	B1	CLEAR 1	
178	B2	CLEAR 2	

Code Dec.	Code Hex.	Keyboard	Video Display
179	B3	(CLEAR)(3)	See graphics character table in this Appendix.
180	B4	(CLEAR)(4)	
181	B5	(CLEAR)(5)	
182	B6	(CLEAR)(6)	
183	B7	(CLEAR)(7)	
184	B8	(CLEAR)(8)	
185	B9	(CLEAR)(9)	
186	BA	(CLEAR)(:)	
187	BB		
188	BC		
189	BD	(CLEAR)(SHIFT)(-)	
190	BE		
191	BF		
192	C0	(CLEAR)(@)*	See list of special characters in this Appendix.
193	C1	(CLEAR)(A)**	
194	C2	(CLEAR)(B)**	
195	C3	(CLEAR)(C)**	
196	C4	(CLEAR)(D)**	
197	C5	(CLEAR)(E)**	
198	C6	(CLEAR)(F)**	
199	C7	(CLEAR)(G)**	
200	C8	(CLEAR)(H)**	
201	C9	(CLEAR)(I)**	
202	CA	(CLEAR)(J)**	
203	CB	(CLEAR)(K)**	
204	CC	(CLEAR)(L)**	
205	CD	(CLEAR)(M)**	
206	CE	(CLEAR)(N)**	
207	CF	(CLEAR)(O)**	
208	D0	(CLEAR)(P)**	
209	D1	(CLEAR)(Q)**	
210	D2	(CLEAR)(R)**	
211	D3	(CLEAR)(S)**	
212	D4	(CLEAR)(T)**	
213	D5	(CLEAR)(U)**	
214	D6	(CLEAR)(V)**	
215	D7	(CLEAR)(W)**	
216	D8	(CLEAR)(X)**	
217	D9	(CLEAR)(Y)**	
218	DA	(CLEAR)(Z)**	
219	DB		
220	DC		
221	DD		
222	DE		
223	DF		
224	E0	(CLEAR)(SHIFT)(@)	
225	E1	(CLEAR)(SHIFT)(A)	
226	E2	(CLEAR)(SHIFT)(B)	
227	E3	(CLEAR)(SHIFT)(C)	
228	E4	(CLEAR)(SHIFT)(D)	
229	E5	(CLEAR)(SHIFT)(E)	
230	E6	(CLEAR)(SHIFT)(F)	
231	E7	(CLEAR)(SHIFT)(G)	
232	E8	(CLEAR)(SHIFT)(H)	
233	E9	(CLEAR)(SHIFT)(I)	
234	EA	(CLEAR)(SHIFT)(J)	

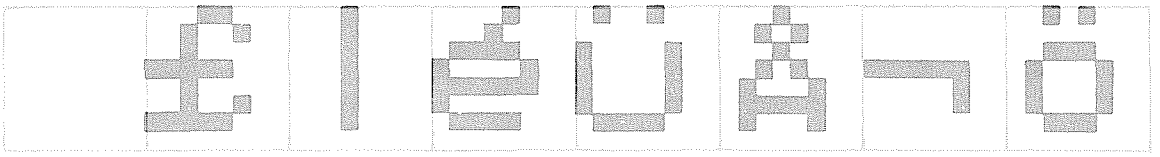
*Empties the type-ahead buffer.
**Used by Keystroke Multiply, if KSM is active.

Code Dec.	Code Hex.	Keyboard	Video Display
235	EB	CLEAR SHIFT K	See list of special characters in this Appendix.
236	EC	CLEAR SHIFT L	
237	ED	CLEAR SHIFT M	
238	EE	CLEAR SHIFT N	
239	EF	CLEAR SHIFT O	
240	F0	CLEAR SHIFT P	
241	F1	CLEAR SHIFT Q	
242	F2	CLEAR SHIFT R	
243	F3	CLEAR SHIFT S	
244	F4	CLEAR SHIFT T	
245	F5	CLEAR SHIFT U	
246	F6	CLEAR SHIFT V	
247	F7	CLEAR SHIFT W	
248	F8	CLEAR SHIFT X	
249	F9	CLEAR SHIFT Y	
250	FA	CLEAR SHIFT Z	
253	FD		
254	FE		
255	FF		

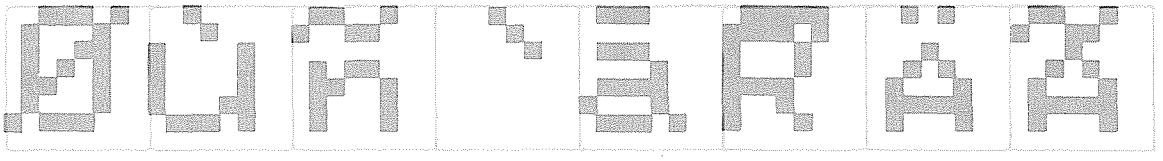
Graphics Characters (Codes 128-191)



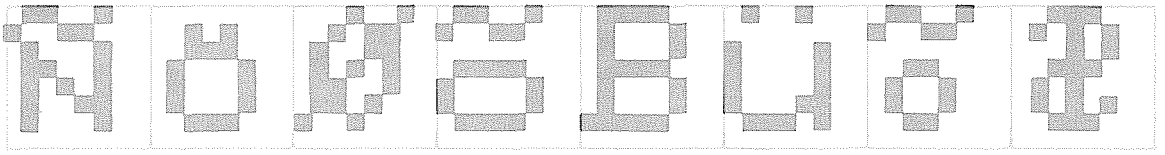
Special Characters (0-31, 192-255)



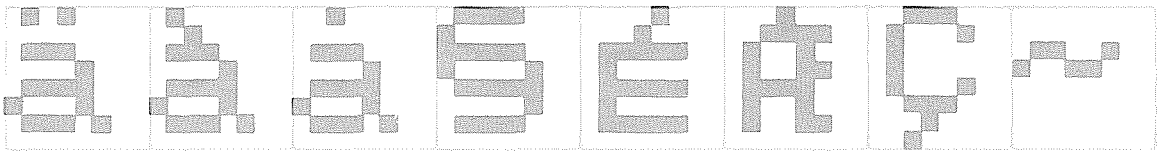
0 1 2 3 4 5 6 7



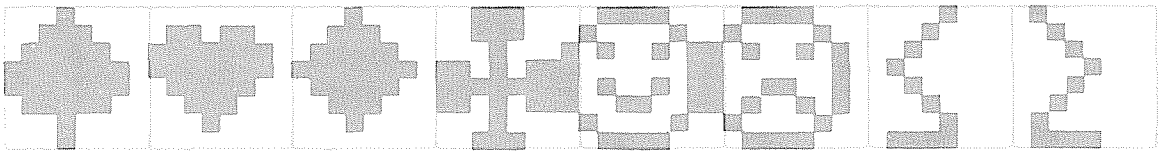
8 9 10 11 12 13 14 15



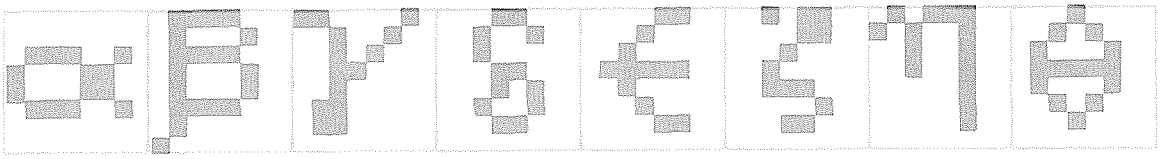
16 17 18 19 20 21 22 23



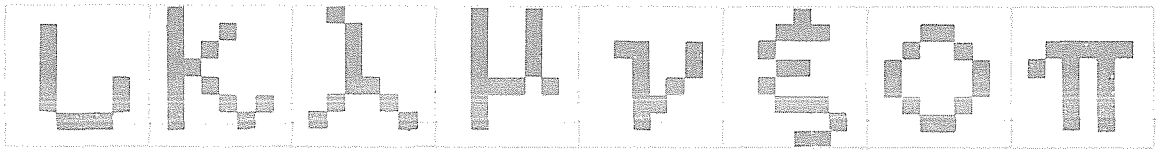
24 25 26 27 28 29 30 31



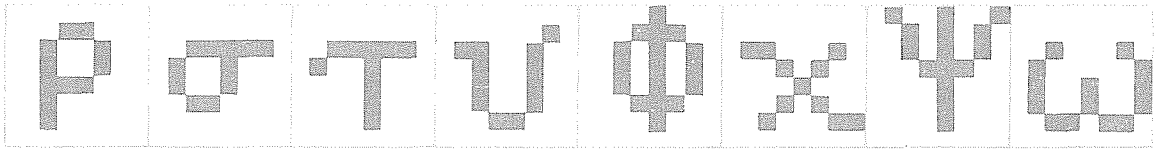
192 193 194 195 196 197 198 199



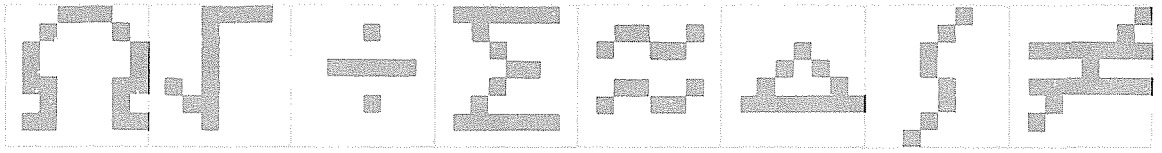
200 201 202 203 204 205 206 207



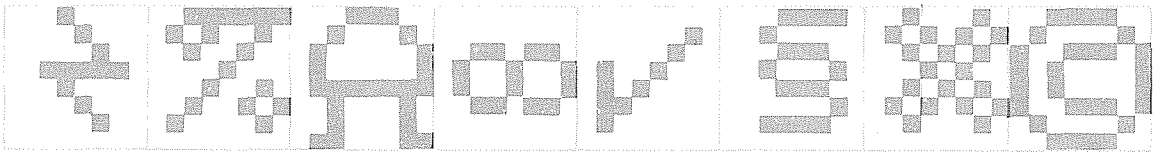
208 209 210 211 212 213 214 215



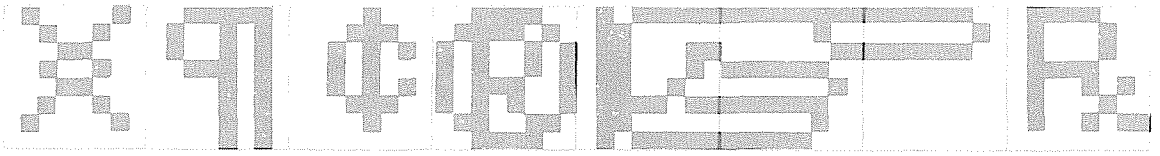
216 217 218 219 220 221 222 223



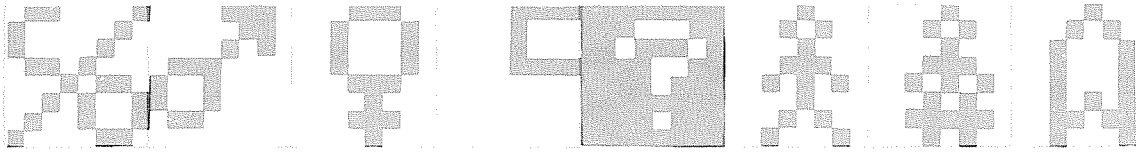
224 225 226 227 228 229 230 231



232 233 234 235 236 237 238 239



240 241 242 243 244 245 246 247



248 249 250 251 252 253 254 255

Appendix D/Keyboard Code Map

The keyboard code map shows the code that TRSDOS returns for each key, in each of the modes: control, shift, unshift, clear and control, clear and shift, clear and unshift.

For example, pressing (CLEAR), (SHIFT), and (1) at the same time returns the code X'A1.'

A program executing under TRSDOS — for example, BASIC — may translate some of these codes into other values. Consult the program's documentation for details.

(BREAK) Key Handling

The (BREAK) key (X'80') is handled in different ways, depending on the settings of three system functions. The table below shows what happens for each combination of settings.

Break Enabled	Break Vector Set	Type-Ahead Enabled	
Y	N	Y	If characters are in the type-ahead buffer, then the buffer is emptied.* If the type-ahead buffer is empty, then a BREAK character (X'80') is placed in the buffer.*
Y	N	N	A BREAK character (X'80') is placed in the buffer.
Y	Y	Y	The type-ahead buffer is emptied of its contents (if any), and control is transferred to the address in the BREAK vector (see @BREAK SVC).*
Y	Y	N	Control is transferred to the address in the BREAK vector (see @BREAK SVC).
N	X	X	No action is taken and characters in the type-ahead buffer are not affected.

*Because the (BREAK) key is checked for more frequently than other keys on the keyboard, it is possible for (BREAK) to be pressed after another key on the keyboard and yet be detected first.

Y means that the function is on or enabled
N means that the function is off or disabled
X means that the state of the function has no effect

Break is enabled with the SYSTEM (BREAK = ON) command (this is the default condition).
The break vector is set using the @BREAK SVC (normally off).
Type-ahead is enabled using the SYSTEM (TYPE = ON) command (this is the default condition).

B1 31	B2 32	B3 33	B4 34	B5 35	B6 36	B7 37	B8 38	B9 39	BØ 3Ø	BA ††	AD 2D	80 80		
!	"	#	$	%	&	'	(	)	Ø †	*	=	BRK †††		
A1 1 21	A2 2 22	A3 3 23	A4 4 24	A5 5 25	A6 6 26	A7 7 27	A8 8 28	A9 9 29	AØ	AA : 2A	BD — 3D	80		
B1 31	B2 32	B3 33	B4 34	B5 35	B6 36	B7 37	B8 38	B9 39	BØ 3Ø	BA 3A	AD 2D	80 80		
8B ØB	91 11	97 17	85 Ø5	92 12	94 14	99 19	95 15	89 Ø9	8F ØF	9Ø 1Ø	Ø Ø	88 Ø8	89 Ø9	
↑	Q	W	E	R	T	Y	U	I	O	P	@	←	→	
9B 1B	F1 51	F7 57	E5 45	F2 52	F4 54	F9 59	F5 55	E9 49	EF 4F	FØ 5Ø	EØ 6Ø	98 18	99 19	
8B ØB	D1 71	D7 77	C5 65	D2 72	D4 74	D9 79	D5 75	C9 69	CF 6F	DØ 7Ø	CØ 4Ø	88 Ø8	89 Ø9	
8A ØA	81 Ø1	93 13	84 Ø4	86 Ø6	87 Ø7	88 Ø8	8A ØA	8B ØB	8C ØC	1E 1E	8D ØD			
↓	A	S	D	F	G	H	J	K	L	+ ;	ENTER		CLEAR 1F	
9A 1A	E1 41	F3 53	E4 44	E6 46	E7 47	E8 48	EA 4A	EB 4B	EC 4C	7E 2B	7F 1D			
8A ØA	C1 61	D3 73	C4 64	C6 66	C7 67	C8 68	CA 6A	CB 6B	CC 6C	5E 3B	5F ØD			
SHIFT	9A 1A	98 18	83 Ø3	96 16	82 Ø2	8E ØE	8D ØD	1B 1B	1D 1D	1C 1C	SHIFT			
	Z	X	C	V	B	N	M	< ,	> .	? /				
	FA 5A	F8 58	E3 43	F6 56	E2 42	EE 4E	ED 4D	7B 3C	7D 3E	7C 3F				
	DA 7A	D8 78	C3 63	D6 76	C2 62	CE 6E	CD 6D	5B 2C	5D 2E	5C 2F				
CTRL	A Ø									Ø Ø	CAPS			
	A Ø									2 Ø				
	A Ø									2 Ø				

384

LEGEND:

Clear and Control — Control

Clear and Left Shift — Shift

Clear and Unshift — Unshift

Note: Pressing CONTROL, SHIFT, and @ at the same time generates an EOF (end of file) — — X'1C' with NZ return flag.

Whenever pressing CLEAR, SHIFT, and another key at the same time, be sure to use the left SHIFT key — not the right SHIFT key.

† Pressing SHIFT and Ø at the same time (or CAPS alone) turns the CAPS mode on or off.

†† Pressing CONTROL and : at the same time causes a screen print.

††† Pressing SHIFT and BREAK at the same time reselects the last drive.

Codes for these keys are the same as for the main keyboard.

81 81 / 91 F1 91 / 81 81	82 82 / 92 F2 92 / 82 82	83 83 / 93 F3 93 / 83 83
7	8	9
4	5	6
1	2	3
Ø	•	ENT

Index

Subject	Page
@ABORT	230
Access	
device	191-192
drive	193-203
file	186
Address decoding	15
Adjustment, drive motor	93
Adjustments, FDC	61
@ADTSK	231
Alien disk controller	194
Alignment, disk drive	93
Allocation	
dynamic	185
information	194, 207
methods of	185
pre-	185
unit of	184
ASCII codes	374-376
Background tasks, invoking	215-216
@BANK	219-221, 232-233
Bank switching	218-221
Baud	15, 21
Baud rate generator	169
@BKSP	234
BOOT/SYS	187
BREAK	
detection	211-214, 235
key handling	383
@BREAK	235
Buffering	15, 59, 69
Byte I/O	222-224
Carriage movement	93
CASIN*	29
CASOUT*	28
Cassette circuitry	21
Cat eyes adjustment	94
Characters	
ASCII	374-376
codes	373-382
graphics	377-378, 380
special	378-379, 381-382
@CHNIO	236
@CKDRV	237
@CKEOF	238
@CKTSK	239
Cleaning the magnetic head	93
Clock generation	60, 70
Clock rate, changing	363
@CLOSE	240
@CMNDI	241
@CMNDR	242
Codes	
ASCII	374-376
character	373-382
error	369
graphics	377-378, 380
keyboard	383-384
return	210
special character	378-379, 381-382
Compensated write data	88
Compliance check	96
Control chain	132
Controller, CRT	19
Controller, floppy disk	9
Converting to TRSDOS Version 6	209-210
CPU board	9, 10, 11, 15
CREATEd files	197
Crowbar	113, 124
CRT	10, 11
@CTL	222-224, 243-244
interfacing to device drivers	224-226
Current limit circuit	130
Cylinder	
highest numbered	194
number of	200
position, current	194
starting	207
@DATE	245
@DCINIT	246
@DCRES	247
@DCSTAT	248
DEBUG	188
@DEBUG	249
@DECHEX	250
Decoding, address	15
Density, double and single	183, 193, 200
Device	
access	191-192
handling	209
NIL	191
Device Control Block (DCB)	191
Device driver	189, 190, 195
address	191
COM	225-226
@CTL interfacing to	224-226
keyboard	225
printer	225
templates	222-224
video	225
Devspec	191

Index

Subject	Page
Directory	
location on disk	184, 194
primary and extended entries	196, 198, 202
record, locating a	202
records (DIREC)	195-198
sectors, number of	196
Directory Entry Code (DEC)	200-201, 202, 206
@DIRRD	251
DIR/SYS	187
@DIRWR	252
Disk drive	9, 10, 11, 81
Disk, diskette	81
controller	194
double-sided	193-194, 199, 200
files	185-186
floppy	183
formatting	199, 200
hard	184
I/O table	195
minimum configuration	189-190
name	200
organization	183-184
single-sided	193-194, 199, 200
space, available	184
@DIV8	253
@DIV16	254
@DODIR	255-256
Drive	
access	193-204
address	194
floppy	183, 193
hard	184, 193
size	193
Drive Code Table DCT	193-195
Drive motor adjustment	93
Drive select	59, 70, 88
Driver — see Device driver	
DRVSEL*	29
@DSP	257
@DSPLY	258
Duty cycle	127
End of File (EOF)	197
Ending Record Number (ERN)	198, 207
ENTER detection	211-214
Environmental specs, power supply	124
Erase gaps	85
Error	
codes and messages	365-369
dictionary	188
@ERROR	259
@EXIT	260
External disk drive	81
FDC controller	9, 10, 11, 59, 61, 69, 72
Feedback control, power supply	109
@FEXT	261
File	
access	186
descriptions, TRSDOS	187-190
modification	197
File Control Block (FCB)	205
Files	
CREATEd	197
device driver	189
filter	189
system (/SYS)	187-188, 189-190, 201
utility	189
Filter templates	222-224
Filters	189, 190, 222-224
example of	224
@FLAGS	210, 262-263
Floppy disk data separator	72
Flyback converter	121
@FNAME	264
@FSPEC	265
Fusing, power supply	109, 112
@GET	222-224, 266
Gran, granule	
allocation information	207
definition	184, 199
per track	183-184, 194
Granule Allocation Table (GAT)	
location on disk	184
contents of	198-200
Graphics	
characters, printing	362
codes	377-378, 380
@GTDCB	267
@GTDCT	268
@GTMOD	269
Guidelines, programming	209-226
Hash code	197, 200
Hash Index Table (HIT)	
location on disk	184
explanation of	200-201
@HDFMT	270
Head amplitude	96
Head, disk drive	93

Index

Subject	Page
Head positioning	84
@HEXDEC	271
@HEX8	272
@HEX16	273
@HIGH$	274
Hold-Up time, power supply	124
Horizontal linearity	146
@ICNFG, interfacing to	214-215
I/O bus	26
Index pulse	84, 90
Index sector timing	95
Index sensor	81, 84
@INIT	275
Initialization configuration vector	214-215
Input line terminator	91
Interrupt tasks	216-218
Interrupts	59, 69, 170
@IPL	276
Job Control Language (JCL)	188, 210
Jumper options	5
@KBD	277
@KEY	278
Keyboard	19
Keyboard codes	383-384
@KEYIN	279
KFLAG$	211
Kick start latch	125
@KITSK, interfacing to	215-216
@KLTSK	280
Library commands	210
technical information on	361-363
@LOAD	281
Load board values, power supply	114
@LOC	282
@LOF	283
LOG utility	362
@LOGER	284
Logic board, disk drive	91
Logical Record Length (LRL)	197, 206
@LOGOT	285
Low voltage outputs	113, 130
Memory address decoding	18
Memory banks — see RAM banks	
Memory header	192, 209
Memory map	18, 371
Minimum configuration disk	189
Modification date	197
MODOUT	28
Motor adjustment, disk drive	93
@MSG	286
@MUL8	287
@MUL16	288
Next Record Number (NRN)	206
NIL device	191
NMI logic	59, 69
@OPEN	289
Oscillator	15
Overlays, system	187-188, 201
Over-Current protection	124
Over-Voltage protection	109, 124, 131
PAL circuits	15
@PARAM	290-291
Password	
for TRSDOS files	190
protection levels	196, 206
@PAUSE	292
PAUSE detection	211-214
@PEOF	293
Port address decoding	15
Port bit map	18, 28, 171
@POSN	294
Power supplies	9, 10, 11, 109, 112, 121
Precompensation, write	60
Preventive maintenance	93
@PRINT	295
Printer status	21
Printing Graphics Characters	362
Programming Guidelines	209-226
Protection Levels	196, 206, 209
@PRT	296
@PUT	222-224, 297
Radial Alignment, Head	94
RAM	19, 20
RAM Banks	
switching	218-221
use of	232-233
@RAMDIR	2998
@RDHDR	299
RDINSTATUS*	28
RDNMISTATUS*	28
@RDSEC	300
@RDSSC	301
@RDTRK	302
@READ	303
Read Data Pulse	86, 90
Real Time Clock	21
Record	
length	185-186, 197, 206

Index

Subject	Page
logical and physical	185-186
numbers	186
processing	186
spanning	185-186
Rectifier	113
@REMOV	304
@RENAM	305
Resistor Termination	83
Restart Vectors (RSTs)	211
Return Code (RC)	210
@REW	306
RFI Shield	9
Ripple Specifications	114, 124
@RMTSK	307
ROM	19
@RPTSK	308
@RREAD	309
RS-232	
initializing	214
COM driver for	225-226
RS-232 Board	9, 11, 169
@RSLCT	310
@RSTOR	311
@RUN	312
@RWRIT	313
Sample Programs	336-359
A	337-338
B	339-343
C	344-350
D	351-352
E	353
F	354-359
Sectors	
per cylinder	196, 201
per granule	183-184, 194
@SEEK	314
@SEEKSC	315
@SKIP	316
@SLCT	317
Snubber Circuit	129
@SOUND	318
Sound Option	22
Special Character Codes	378-379, 381-382
Spindle Drive	84
Stack handling	210
Step rate	193
changing	361
@STEPI	319
Stepper motor	81
Supervisor calls (SVCs)	
calling procedure	227
lists of	228-229, 331-333, 334-335
program entry and return conditions	227
sample programs using	336-359
using	227-359
Surge limiter	124
SYS files	187-188, 189-190, 201
System	
files	187-188, 189-190, 201
overlays	187-188, 201
Task	
interrupt level, adding	231
slots	216, 217, 231
Task Control Block (TCB)	216, 217, 231
Vector Table (TCBVT)	216, 217
Task processor, interfacing to	216-218
@TIME	320
Timing, CPU	15
Track 00 Alignment	95
Track 00 Switch	84, 90
Trim erase	86
TRSDOS	
converting to Version 6	209-210
error messages and codes	365-369
file descriptions	187-190
technical information on commands and utilities	361-363
TYPE code	205
Under-Voltage Lockout	130
@VDCTL	321-322
@VER	323
Version, operating system	199
Video Controller	19
Video Monitor	10, 145
Visibility	196
Voltage Controlled Oscillator	60
Voltage Regulation	124
@VRSEC	324
Wait State	60, 69
WAIT value, changing	362
@WEOF	325
@WHERE	326
WRINTMASKREG*	29
@WRITE	327
Write Enable	86
Write Gate	88
Write Precompensation	60, 69, 70
Write Protect	84, 90, 95, 193
WRNMIMASKREG*	28

Index

Subject	Page
@WRSEC	328
@WRSSC	329
@WRTRK	330

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

Printed in U.S.A.