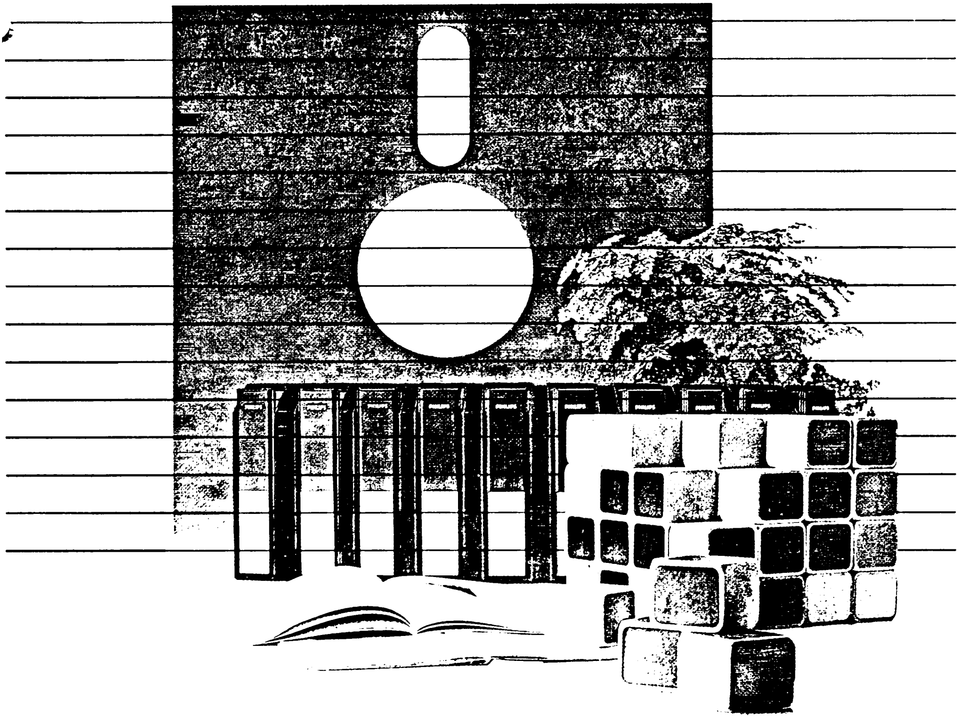
P3500/P3800
TurboDOS 1.4
Z80 implementor's guide (8 bits)



PHILIPS

July 1984

Order Number 5122 993 81837

Manual Number F75H

TurboDOS 1.4

Z80 Implementor's Guide

June 1984

Software 2000, Inc.
1127 Hetrick Avenue
Arroyo Grande, CA 93420
U.S.A.

TurboDOS 1.4 Documentation Page Counts

Copyright Notice

Copyright 1984 by Software 2000, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Software 2000, Inc., 1127 Hetrick Avenue, Arroyo Grande, California 93420, U.S.A.

Trademark Notice

TurboDOS is a registered trademark of Software 2000, Inc., and has been registered in the United States and in most major countries of the free world.

CP/M, CP/M Plus, and MP/M are trademarks of Digital Research.

Disclaimer

Software 2000, Inc., makes no representations or warranties with respect to the contents of this publication, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Software 2000, Inc., shall under no circumstances be liable for consequential damages or related expenses, even if it has been notified of the possibility of such damages.

Software 2000, Inc., reserves the right to revise this publication from time to time without obligation to notify any person of such revision.

First Edition: June 1984

ABOUT THIS GUIDE

Purpose

We've designed this Z80 Implementor's Guide to provide the information you need to know in order to generate various TurboDOS configurations for Z80-based microcomputers, and to write the driver modules for various peripheral devices. This document describes the modular architecture and internal programming conventions of TurboDOS, and explains the procedures for system generation, serialization, and distribution. It also provides detailed interface specifications for hardware-dependent driver modules, and includes assembler source listings of sample drivers.

Assumptions

In writing this guide, we've assumed that you are an OEM, dealer, or sophisticated TurboDOS user, knowledgable in Z80-based microcomputer hardware and assembly-language programming. We've also assumed you have read both the User's Guide and the Z80 Programmer's Guide, and are therefore familiar with the commands, external features, and internal functions of Z80 TurboDOS.

Organization

This guide starts with a section that describes the architecture of TurboDOS. It explains the function of each internal module of the operating system, and how these modules may be combined to create the various configurations of TurboDOS.

The next section explains the system generation procedure in detail, and describes each TurboDOS parameter which can be modified during system generation.

The third section of this guide explains the TurboDOS distribution procedure, including licensing, serialization, and support.

Organization (Continued)

The fourth section is devoted to an in-depth discussion of internal programming conventions, aimed at the programmer writing drivers or resident processes for TurboDOS.

The fifth section presents formal interface specifications for implementing hardware-dependent driver modules.

This guide concludes with a large appendix containing assembler source listings of actual driver modules. The sample drivers cover a wide range of peripheral devices, and provide an excellent starting point for programmers involved in driver development.

Related Documents

In addition to this guide, you might be interested in four other related documents:

- _TurboDOS 1.4 User's Guide_
- _TurboDOS 1.4 Z80 Programmer's Guide_
- _TurboDOS 1.4 8086 Programmer's Guide_
- _TurboDOS 1.4 8086 Implementor's Guide_

You should read the first two volumes before start into this document. The _User's Guide_ introduces the external features and facilities of TurboDOS, and describes each TurboDOS command. The _Z80 Programmer's Guide_ explains the internal workings of Z80 TurboDOS, and describes each operating system function in detail.

You'll need the 8086 guides if you are programming or configuring a TurboDOS system that uses 8086-family microprocessors.

<u>TurboDOS 1.4 Documentation Page Counts</u>

TurboDOS 1.4 Z80
Implementor's Guide

TABLE OF CONTENTS

ARCHITECTURE

SYSTEM GENERATION

DISTRIBUTION

CODING CONVENTIONS

DRIVER INTERFACE

APPENDIX

ARCHITECTURE

This section introduces you to the internal architecture of the TurboDOS operating system. TurboDOS is highly modular, consisting of more than forty separate functional modules distributed in relocatable form. These modules are "building blocks" that you can combine in various ways to produce a family of compatible operating systems. This section describes the modules in detail, and describes how to combine them in various configurations.

Possible TurboDOS configurations include:

- single-user without spooling
- single-user with spooling
- network master
- simple network slave (no local disks)
- complex network slave (with local disks)

Numerous subtle variations are possible in each of these categories.

Module Hierarchy

The diagram on page 1-3 illustrates how the functional modules of TurboDOS interact. As the diagram shows, the architecture of TurboDOS can be viewed as a three-level hierarchy.

Process Level

The highest level of the hierarchy is the process level. TurboDOS can support many concurrent processes at this level. There is one active process that supports the local user who is executing commands and programs in the local TPA. There are also processes to support users running on other computers and making requests of the local computer over the network. There are processes to handle background printing (de-spooling) on local printers. Finally, there is a process that periodically causes disk buffers to be written out to disk.

Kernel Level

The intermediate level of the hierarchy is the <u>kernel level</u>. The kernel supports the various C-functions and T-functions, and controls the sharing of computer resources such as processor time, memory, peripheral devices, and disk files. Processes make requests of the kernel through the entrypoint module OSNTRY, which decodes each C-function and T-function by number and invokes the appropriate kernel module.

Driver Level

The lowest level of the hierarchy is the <u>driver level</u>, and contains all the device-dependent drivers necessary to interface TurboDOS to the particular hardware being used. Drivers must be provided for all peripherals, including console, printers, disks, communications channels, and network interface. Drivers are also required for the real-time clock (or other periodic interrupt source), and for bank-switched memory (if applicable).

TurboDOS is designed to interface with almost any kind of peripheral hardware. It operates most efficiently with interrupt-driven, DMA-type interfaces, but can also work fine using polled and programmed-I/O devices.

TurboDOS Loader

The TurboDOS loader OSLOAD.COM is a program containing an abbreviated version of the kernel and drivers. Its purpose is to load the full TurboDOS operating system from a disk file (OSMASTER.SYS) into memory at each system cold-start.

```
 _______________________________________________________________
|___________________TurboDOS Module Hierarchy___________________|
|                                                               |
|                       Despool Lcl Usr Net Svc Buffers         |
|                       DSPOOL  LCLUSR  NETSVC  FLUSHR          |
|                          |    LCLMSG  NETTBL     |            |
| Process Level            |    LCLTBL  NETFWD     |            |
|                          |    CMDINT     |       |            |
|                          |    AUTLOD     |       |            |
|               Loader     |    SGLUSR     |       |            |
|               OSLOAD     |    AUTLOG     |       |            |
|               LDRMSG     |    SUBMIT     |       |            |
|                  |_______|_______|_______|_______|            |
|----------------------------------|----------------------------|
|                                  |                            |
|                               Decode                          |
| Kernel Level                  OSNTRY                          |
|    ______________________________|________________________    |
|    |             |               |       |       |       |    |
|  Bank          Other           File   Net Req  Clock  Support |
| BNKMGR        NONFIL          FILMGR  NETMGR  RTCMGR  DSPCHR  |
| BNKREQ        CPMSUP          FILSUP  NETREQ     |    DSPSGL  |
|    |          MPMSUP          FILCOM  MSGFMT     |    MEMMGR  |
|    |          QUEMGR          FILLOK  NETTBL     |    COMSUB  |
|    |             |            FFOMGR  NETLOD     |            |
|    |             |            DRVLOK     |       |            |
|    |             |            FASLOD     |       |            |
|    |     ________|_______     NORLOD     |       |            |
|    |     |       |       |       |       |       |            |
|    |  Comm Ch Printer Console Record     |       |    Initial |
|    |  COMMGR  LSTMGR  CONMGR  BUFMGR     |       |    SYSNIT  |
|    |     |    LSTTBL  CONTBL  DSKMGR     |       |       |    |
|    |     |    SPOOLR  DOMGR   DSKTBL     |       |       |    |
|    |     |    SPLMSG  INPLN      |       |       |       |    |
|    |     |       |       |       |       |       |       |    |
|----|-----|-------|-------|-------|-------|-------|-------|----|
| Driver Level     |       |       |       |       |       |    |
|    |     |       |       |       |       |       |       |    |
|    |  Comm Ch Printer Console  Disk   Network  Clock  Initial |
|    |  COMDRV  LSTDRA  CONDRA  DSKDRA  CKTDRA  RTCDRV  HDWNIT  |
|  Bank         LSTDRB   or     DSKDRB  CKTDRB   or             |
| SELBNK         etc.   CONREM   etc.    etc.   RTCNUL          |
|_______________________________________________________________|
```

Process Modules

Module	Function
LCLUSR	Responsible for supporting local user's TPA activities.
LCLMSG	Contains all O/S error messages.
LCLTBL	Local user option table.
CMDINT	Command interpreter, processes commands from local user.
AUTLOD	Autoload routine which processes COLDSTRT.AUT and WARMSTRT.AUT.
SGLUSR	Routine to flush/free disk buffers at each console input. Use for single-user configurations instead of FLUSHR.
AUTLOG	Automatic log-on routine. Used when full log-on security is not desired. See AUTUSR patch point.
SUBMIT	Routine to emulate CP/M processing of $$$.SUB files. (Use is not recommended.)
NETSVC	Services network requests from other processors on the network.
NETTBL	Tables to define local network topology, used by NETSVC+NETREQ.
NETFWD	Manages network message forwarding. Requires NETREQ+NETSVC.
DSPOOL	Processes background printing.
FLUSHR	Periodically flushes disk buffers. Use for network master configuration instead of SGLUSR.

Kernel Modules

Module	Function
OSNTRY	Kernel entrypoint module which decodes each C-function and T-function by number and invokes the appropriate kernel module.
FILMGR	File manager responsible for requests involving local files.
FILSUP	Support routines for FILMGR.
FILCOM	Processes common file requests always processed locally.
FILLOK	File- and record-level interlock routines called by FILMGR.
FFOMGR	FIFO support, called by FILLOK.
DRVLOK	Drive interlock routines.
FASLOD	Program loader incorporating an optimizer for fastest loading.
NORLOD	Unoptimized program loader, an alternative to FASLOD.
BUFMGR	Buffer manager called by FILMGR. Maintains pool of disk buffers used to speed local file access.
DSKMGR	Disk manager responsible for physical access to local disks, called by BUFMGR and FASLOD.
DSKTBL	Table defining drives A-P as local or remote disk drives.
NONFIL	Processes non-file functions.
COMMGR	Processes comm-channel funct's.

Kernel Modules (Continued)

Module	Function
CPMSUP	Processes C-functions 7, 8, 24, 28, 29, 31, 37, 107 (optional).
MPMSUP	Processes C-functions 141-143, 153, 160, 161 (optional).
QUEMGR	Emulates MP/M queues, supports C-functions 134-140 (optional). Requires MPMSUP.
CONMGR	Responsible for console I/O.
CONTBL	Links CONMGR to console driver.
DOMGR	Responsible for do-files.
INPLN	Console input line editor used by CMDINT and C-function 10.
LSTMGR	Responsible for printer output.
LSTTBL	Table defining printers A-P and queues A-P as local or remote.
SPOOLR	Print spooler which diverts print output to a spool file when spooling is activated. Also handles direct printing to remote printers.
NETREQ	Responsible for issuing network request messages for all functions not processed locally.
MSGFMT	Network message format table used by NETREQ.
NETMGR	Network message routing routine used by NETSVC and NETREQ.

Kernel Modules (Continued)

Module	Function
RTCMGR	Real-time clock manager keeps system date and time.
NETLOD	Loads programs over the network.
BNKMGR	Responsible for bank-switching in banked-memory systems.
BNKREQ	Alternative to NETLOD for use in banked-memory systems.
DSPCHR	Multi-task dispatcher which controls sharing of the local processor among multiple processes.
DSPSGL	Null dispatcher used as alternative to DSPCHR when only one process is required (OSLOAD.COM and single-user w/o spooling).
MEMMGR	Memory manager responsible for dynamic allocation of memory.
COMSUB	Common subroutines used in all configurations.
SYSNIT	System initialization routine executed at system cold-start.
RTCNUL	Null real-time clock driver, used in configurations where there is no periodic interrupt source.
CONREM	Remote console driver for network master to support MASTER command.
PATCH	128 bytes of zeroes, may be included to provide patch area.

Driver Modules

Module	Function
CONDR@	Console I/O driver.
LSTDR@	Printer output driver(s).
DSKDR@	Disk driver(s).
CKTDR@	Network circuit driver(s).
COMDRV	Communications channel driver.
RTCDRV	Real-time clock driver.
SELBNK	Bank-select driver for banked-memory systems.
HDWNIT	Cold-start initialization for all hardware-dependent drivers.

Standard Packages

To simplify the system generation process, the most commonly-used combinations of TurboDOS modules are pre-packaged into the following standard configurations:

Package	Description
STDLOADR	cold-start loader
STDSINGL	single-user without spooling
STDSPOOL	single-user with spooling
STDMASTR	network master
STDSLAVE	simple slave w/o local disks
STDSLAVX	complex slave with local disks

The contents of each standard package is detailed in the matrix on the facing page. Most TurboDOS requirements can be satisfied by linking the appropriate standard package together with a few additional optional modules plus the requisite driver modules.

Module	K	LOADR	SINGL	SPOOL	MASTR	SLAVE	SLAVX
AUTLOD	.2	-	AUTLOD	AUTLOD	AUTLOD	AUTLOD	AUTLOD
AUTLOG	.0	-	AUTLOG	AUTLOG	AUTLOG	AUTLOG	AUTLOG
BNKMGR	2.0	-	+	+	+	+	+
BNKREQ	.3	-	-	-	+	+	+
BUFMGR	1.1	BUFMGR	BUFMGR	BUFMGR	BUFMGR	-	BUFMGR
CMDINT	1.3	-	CMDINT	CMDINT	CMDINT	CMDINT	CMDINT
COMMGR	.1	-	COMMGR	COMMGR	COMMGR	COMMGR	COMMGR
COMSUB	.3	COMSUB	COMSUB	COMSUB	COMSUB	COMSUB	COMSUB
CONMGR	.3	CONMGR	CONMGR	CONMGR	CONMGR	CONMGR	CONMGR
CONREM	.4	-	-	-	+	-	-
CONTBL	.0	CONTBL	CONTBL	CONTBL	CONTBL	CONTBL	CONTBL
CPMSUP	.2	-	+	+	+	+	+
DOMGR	.4	-	DOMGR	DOMGR	DOMGR	DOMGR	DOMGR
DRVLOK	.2	-	-	-	DRVLOK	-	-
DSKMGR	.6	DSKMGR	DSKMGR	DSKMGR	DSKMGR	-	DSKMGR
DSKTBL	.0	DSKTBL	DSKTBL	DSKTBL	DSKTBL	DSKTBL	DSKTBL
DSPCHR	.7	-	-	DSPCHR	DSPCHR	DSPCHR	DSPCHR
DSPOOL	.9	-	-	DSPOOL	DSPOOL	-	DSPOOL
DSPSGL	.2	DSPSGL	DSPSGL	-	-	-	-
FASLOD	.4	-	+	+	+	+	+
FFOMGR	.9	-	-	-	FFOMGR	-	-
FILCOM	.4	FILCOM	FILCOM	FILCOM	FILCOM	FILCOM	FILCOM
FILLOK	1.7	-	-	-	FILLOK	-	-
FILMGR	2.1	FILMGR	FILMGR	FILMGR	FILMGR	-	FILMGR
FILSUP	2.4	FILSUP	FILSUP	FILSUP	FILSUP	-	FILSUP
FLUSHR	.2	-	-	-	FLUSHR	-	-
INPLN	.1	-	INPLN	INPLN	INPLN	INPLN	INPLN
LCLMSG	.4	-	LCLMSG	LCLMSG	LCLMSG	LCLMSG	LCLMSG
LCLTBL	.0	-	LCLTBL	LCLTBL	LCLTBL	LCLTBL	LCLTBL
LCLUSR	1.2	-	LCLUSR	LCLUSR	LCLUSR	LCLUSR	LCLUSR
LDRMSG	.2	LDRMSG	-	-	-	-	-
LSTMGR	.2	-	LSTMGR	LSTMGR	LSTMGR	LSTMGR	LSTMGR
LSTTBL	.1	-	LSTTBL	LSTTBL	LSTTBL	LSTTBL	LSTTBL
MEMMGR	.3	-	MEMMGR	MEMMGR	MEMMGR	MEMMGR	MEMMGR
MPMSUP	.1	-	+	+	+	+	+
MSGFMT	.1	-	-	-	+	MSGFMT	MSGFMT
NETFWD	.3	-	-	-	+	+	+
NETLOD	.4	-	-	-	+	+	+
NETMGR	.9	-	-	-	NETMGR	NETMGR	NETMGR
NETREQ	1.5	-	-	-	+	NETREQ	NETREQ
NETSVC	1.7	-	-	-	NETSVC	+	+
NETTBL	.0	-	-	-	NETTBL	NETTBL	NETTBL

Module	K	LOADR	SINGL	SPOOL	MASTR	SLAVE	SLAVX
NONFIL	.2	NONFIL	NONFIL	NONFIL	NONFIL	NONFIL	NONFIL
NORLOD	.1	-	+	+	+	+	+
OSLOAD	1.3	OSLOAD	-	-	-	-	-
OSNTRY	.5	OSNTRY	OSNTRY	OSNTRY	OSNTRY	OSNTRY	OSNTRY
PATCH	.1	+	+	+	+	+	+
QUEMGR	1.1	-	-	-	+	+	+
RTCMGR	.1	-	RTCMGR	RTCMGR	RTCMGR	-	RTCMGR
RTCNUL	.1	+	+	+	+	+	+
SGLUSR	.1	-	SGLUSR	SGLUSR	-	-	SGLUSR
SPLMSG	.1	-	-	SPLMSG	SPLMSG	SPLMSG	SPLMSG
SPOOLR	.5	-	-	SPOOLR	SPOOLR	SPOOLR	SPOOLR
SUBMIT	.1	-	+	+	+	+	+
SYSNIT	.0	-	SYSNIT	SYSNIT	SYSNIT	SYSNIT	SYSNIT

Optional Modules

To supplement the standard packages, certain optional modules (marked by "+" in the matrix above) may have to be added. The following table explains where these optional modules are required:

Module	Where Required
BNKMGR	All systems with banked memory.
BNKREQ	Banked systems that load programs over the network.
CONREM	Network masters with no console (instead of CONDR@).
CPMSUP	To support C-fcns 7, 8, 24, 28, 29, 31, 37 and 107.
FASLOD	Non-banked systems that load pgms from local disks.
MPMSUP	To support C-fcns 134-143, 153, 160 and 161.
MSGFMT	Network masters that make requests over the network.
NETFWD	To support forwarding of network messages.
NETLOD	Non-banked systems that load pgms over the network.
NETREQ	Network masters that make requests over the network.
NORLOD	Smaller, unoptimized alternative to FASLOD (above).
PATCH	Wherever a supplementary patch area is required.
QUEMGR	To support MP/M queue emulation (C-fcns 134-140.)
RTCNUL	Wherever no RTC driver is available.
SUBMIT	To emulate CP/M processing of $$$.SUB.

Memory Required

To estimate the memory required by a particular TurboDOS configuration, you need to take into account the combined size of all functional modules, driver modules, disk buffers, and other dynamic storage.

Drivers typically require 1K to 4K, and can be even larger if the hardware is especially complex. Disk buffer space should be as large as possible for optimum performance, especially in a network master. About 4K of disk buffer space is reasonable for a single-user system, although less can be used in a pinch. Other dynamic storage doesn't usually exceed 1K in single-user systems, 2K in network masters.

The following table gives typical memory requirements for standard TurboDOS configurations on non-banked hardware:

	LOADR	SINGL	SPOOL	MASTR	SLAVE	SLAVX
O/S	10K	13K	15K	20K	10K	18K
Drivers	2K	2K	2K	3K	1K	2K
Buffers	4K	4K	4K	16K	-	4K
Dynamic	1K	1K	1K	3K	2K	2K
Total	17K	20K	22K	42K	13K	26K
TPA	-	44K	42K	22K	51K	38K

In banked-memory systems, a full 63K TPA is always available.

Other Languages

To facilitate translation into languages other than English, TurboDOS has been implemented with all textual messages segregated into separate modules. All such message modules are available in source form to TurboDOS OEM licensees upon request.

The following modules contain all TurboDOS operating system messages:

Module	Contains
LCLMSG	Most operating system messages.
SPLMSG	Spooler error messages.
LDRMSG	Loader messages for OSLOAD.COM.

In addition, a separate message module is available for each TurboDOS command.

TurboDOS 1.4 Documentation Page Counts

SYSTEM GENERATION

This section explains the TurboDOS system generation procedure in detail. It describes how to use the GEN command to link a desired set of TurboDOS modules together, and details the numerous system patch points which may be modified during system generation. Step-by-step procedures and examples are provided.

Introduction

The functional modules of TurboDOS are distributed in relocatable form (.REL files). Hardware-dependent driver modules are furnished in the same fashion. The TurboDOS GEN command is a specialized linker used to bind the desired combination of modules together into an executable version of TurboDOS. The GEN command also includes a symbolic patch facility used to modify a variety of operating system parameters.

To generate a complete TurboDOS system, you typically must use the GEN command several times. At minimum, you have to generate both a loader OSLOAD.COM and a master operating system OSMASTER.SYS. For a networking system you also have to generate a slave operating system OSSLAVE.SYS. Complex networks may require generation of several different slave or master configurations. Finally, you may have to use GEN to generate a cold-start bootstrap routine for the start-up PROM or boot track.

At cold-start, the bootstrap routine loads the loader program OSLOAD.COM into the TPA of the master computer and executes it. OSLOAD loads the master operating system from the file OSMASTER.SYS into the upper portion of memory. The master operating system then down-loads the slave operating system from the file OSSLAVE.SYS over the network into each slave computer.

GEN Command

The GEN command is a specialized linker for software modules in Microsoft relocatable format, and is designed primarily for use in TurboDOS system generation.

Syntax

```
GEN srcefile {destfile} {;options}
```

Explanation

The GEN command links a specified collection of relocatable modules together into a single executable program. The "srcefile" argument specifies the names of two input files: a configuration file "srcefile.GEN" and a parameter file "srcefile.PAR". The "destfile" argument specifies the name of the executable output file to be created (normally type .COM or .SYS). If "destfile" is omitted, then the "srcefile" argument is also used as the name of the executable output file, and should include an explicit file type (.COM or .SYS).

If the configuration file "srcefile.GEN" is found, it must contain the list of relocatable modules (.REL files) to be linked together. If the configuration file is not found, then the GEN command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
relfile {,relfile}... {;comment}
```

A null directive terminates the prompting sequence and causes processing to proceed.

After obtaining the list of modules from the file or console, GEN links all of the modules together, a two-pass process that displays the name of each module as it is encountered.

Explanation (Continued)

When the linking phase is complete, GEN looks for a parameter file "srcefile.PAR" and processes it if found. The parameter file (if present) must be a text file containing symbolic patches. The syntax of each .PAR file entry is:

```
location = value {,value}... {;comment}
```

where the "value" arguments are to be stored in consecutive memory locations starting with the address specified by "location".

The "location" argument may be the name of a public symbol, a hexadecimal number, or an expression composed of names and hex numbers connected by + or - operators. Hex numbers must begin with a digit (for example, 0FFFF) to distinguish them from names. The "location" expression must be followed by an equal-sign = character.

The "value" arguments may be expressions (as defined above) or quoted ASCII strings, and must be separated by commas. A "value" expression is stored as a 16-bit word if its value exceeds 255 or if it is enclosed in parentheses; otherwise, it is stored as an 8-bit byte. A quoted ASCII string may be enclosed by either quotes "..." or apostrophes '...', and is stored as a sequence of 8-bit bytes. Within a quoted string, ASCII control characters may be specified by using circumflex (example: "^X" denotes CTRL-X).

After the .PAR file (if any) is processed and the necessary patches made, GEN writes the executable file out to disk.

Explanation (Continued)

Each relocatable TurboDOS module is magnetically serialized with a unique serial number. The serial number consists of two components: an "origin number" which identifies the issuing TurboDOS licensee, and a "unit number" which uniquely identifies each copy of TurboDOS issued by that licensee. The GEN command verifies that all modules to be linked are serialized consistently, and serializes the executable file accordingly.

Options

Option	Explanation
;Kxxxx	Indicates that a system for a banked-memory environment is to be generated, and defines the hexadecimal base address "xxxx" of the common (non-switched) memory segment.
;Lxxxx	Defines the hexadecimal address "xxxx" as the lower boundary of the executable program. Default for .COM files is ;L0100.
;M	Prints a load map.
;S	Prints a sorted symbol table.
;Uxxxx	Defines the hexadecimal address "xxxx" as the upper boundary of the executable program. Default for .SYS files is ;UFFFF.
;X	Diagnoses any references to undefined symbols. Default is not to diagnose such references, since they are quite normal in TurboDOS system generation.

Example

In the following example, GEN is used to link a single-user TurboDOS system for a banked-memory system, using the modules listed in OSMASTER.GEN and the patches in OSMASTER.PAR, creating the executable file OSMASTER.SYS.

```
0A}GEN OSMASTER.SYS ;MKC000
Copyright 1984, Software 2000, Inc.
* ; Single-user without spooling
* ; Banked-memory, common starts at C000
* STDSINGL  ;standard single-user system
* BNKMGR    ;banked-memory system
* CPMSUP    ;seldom-used CP/M functions
* BNKDRV    ;bank-select driver
* CONDRV    ;console driver
* LSTDRV    ;printer driver
* SPDDRV    ;serial/parallel driver
* RTCDRV    ;real-time clock driver
* DSKDRV    ;floppy disk driver
* DST58F    ;disk spec table 5/8 floppy
* NITDRV    ;driver initialization
* INTDRV    ;driver interrupt handler

Pass 1
LCLUSR LCLTBL CMDINT AUTLOD SGLUSR etc.

Pass 2
LCLUSR LCLTBL CMDINT AUTLOD SGLUSR etc.

Processing parameter file:
; Patches for single-user w/o spooling
AUTUSR = 80 ;logon to user 0 privileged
NMBUFS = 8  ;number of disk buffers
PTRAST = 2  ;printer on channel 2
EOPCHR = "^Z" ;end-of-print character
SRHDRV = 1  ;search drive A
PRTMOD = 0  ;direct printing mode

Writing output file A:OSMASTER.SYS
0A}
```

Error Messages

```
File name missing from command
Invalid input file name
Non-privileged user
Serial number violation
Not enough memory
Vacuous input file(s)
Unexpected EOF in input file
Disk is full
Can't make output file
No input files
Can't open input file
Load address out-of-bounds
Multiple defined starting address
Duplicate symbol: <name>
Undefined symbol: <name>
```

Patch Points

The following table describes various public symbols in TurboDOS which you may wish to modify using the symbolic patch facility of the GEN command. (Other patch points may exist in hardware-dependent drivers, but they are beyond the scope of this document.)

Symbol	Default Value	Module
ABTCHR	= "^C"	CONTBL
Abort character (after attention).		
ATNBEL	= "^G"	CONTBL
Attention-received warning character.		
ATNCHR	= "^S"	CONTBL
Attention character. May be patched to another character if the default value of CTRL-S is needed by application programs. A common choice is zero (NUL), which allows the console BREAK key to be used as an attention key.		
AUTUSR	= OFF	AUTLOG
Automatic log-on user number. Default value of OFF requires that user log-on via LOGON command. If automatic log-on desired at cold-start, patch AUTUSR to the desired user number (00-1F), and set the sign-bit if a privileged log-on is desired. Generally patched to 80 in single-user systems to cause automatic privileged log-on to user zero.		

Patch Points (Continued)

Symbol	Default Value	Module

BFLDLY = (012C) FLUSHR

Buffer flush delay determines how often disk buffers are written to disk, stated in system "ticks". Default value (300 decimal) causes buffers to be flushed about every five seconds (assuming 60 ticks per second).

BUFSIZ = 3 BUFMGR

Default disk buffer size (0=128, 1=256, 2=512, 3=1K,..., 7=16K). Default value specifies 1K disk buffers.

CKTAST = (0000),CKTDRA, NETTBL
(0100),CKTDRB,
(0200),CKTDRC,
(0300),CKTDRD

Circuit assignment table defines network topology. Contains NMBCKT two-word entries, one for each network circuit to which this processor is attached. The first word of each entry specifies the network address by which this processor is known on a particular circuit, and the second word specifies the entrypoint address of the circuit driver responsible for that circuit. (Possibly several circuits may be handled by the same driver.)

CLBLEN = 9D CMDINT

Command line buffer length defines longest permissible command line. The default value permits two 80-char lines.

Patch Points
(Continued)

Symbol	Default Value	Module
CLPCHR	= "}"	CMDINT
Command line prompt character.		
CLSCHR	= "\"	CMDINT
Command line separator character.		
COLDFN	= 0,"COLDSTRT","AUT"	AUTLOD
File name and drive for cold-start auto-load processing (in FCB format).		
COMPAT	= 0	FILCOM
Default compatibility flags which define rules to be used for file-sharing. Patch to 0F8 to relax most MP/M restrictions.		
CONAST	= 0,CONDRA	CONTBL
Console assignment table defines how console I/O is handled. First byte passed to console driver, and commonly defines the channel number (e.g., serial port) to be used for the console. Following word specifies the entrypoint address of the console driver to be used.		
CPMVER	= 31	NONFIL
CP/M BDOS version number returned by C-function 12 in L-register.		

Patch Points (Continued)

<table>
<tr><th>Symbol</th><th>Default Value</th><th>Module</th></tr>
<tr><td>CURBNK</td><td>= 1</td><td>BNKMGR</td></tr>
<tr><td colspan="3">Initial memory bank selected for TPA at cold-start. Applicable to banked-memory systems only. Patch to 0 to select non-banked mode at cold-start.</td></tr>
<tr><td>DEFDID</td><td>= (0000)</td><td>NETTBL</td></tr>
<tr><td colspan="3">Default network destination ID, used for routing all network requests that are not related to a particular disk drive, queue or printer. In a slave, DEFDID should be set to the network address of the master.</td></tr>
<tr><td>DSKAST</td><td>= 00,DSKDRA,01,DSKDRB,
OFF,(0000),OFF,(0000),...</td><td>DSKTBL</td></tr>
<tr><td colspan="3">Disk assignment table, an array of 16 three-byte entries (one for each drive letter A-P) that defines which drives are local, remote, and invalid.

For a local drive, the first byte must not have the sign-bit set. That byte is passed to the disk driver, and is commonly used to differentiate between multiple drives connected to a single controller. The following word specifies the entry-point address of the disk driver to be used.

For a remote drive, the first byte must have the sign-bit set. The low-order bits of that byte specify the drive letter to be accessed on the remote processor. The following word specifies the network address of the remote processor.</td></tr>
</table>

Patch Points
(Continued)

Symbol	Default Value	Module
DSKAST	(Continued) For an invalid drive, the first byte must be 0FF, and the following word should be (0000). NOTE: In slave configurations STDSLAVE and STDSLAVX, the default values are: DSKAST = 80,(0000),81,(0000), 82,(0000),83,(0000), ...,8E,(0000),8F,(0000)	DSKTBL
DSPPAT	= 01,01,01,...,01 De-spool printer assignment table, an array of 16 bytes (one for each printer letter A-P) that defines the initial queue to which each printer is assigned. Hex values 01 through 10 correspond to queues A-P, and 0 means that the printer is off-line. The default value assigns all printers to queue A.	LSTTBL
ECOCHR	= "^P" Echo-print character (after attention).	CONTBL
EOPCHR	= 0 End-of-print character. May be patched to any non-null character, in which case the presence of that character in the print output stream will automatically signal an end-of-print-job condition. The value zero disables this feature.	LSTTBL

Patch Points
(Continued)

Symbol	Default Value	Module
FWDTBL	= (0FFFF),(0FFFF),(0FFFF),(0FFFF),0FF	NETTBL
Network forwarding table, an array of two-byte entries that define any explicit message forwarding routes to be used by this processor. The first byte of each entry specifies a "foreign" circuit number N, and the second byte a "domestic" circuit number C. Any messages destined for circuit N will be routed via circuit C. This table is variable-length, terminated by 0FF, and defaults to empty.		
LDCOLD	= 0FF	AUTLOD
Cold-start autoload enable flag. Patch to zero if you want to disable the cold-start autoload feature (COLDSTRT.AUT).		
LDWARM	= 0FF	AUTLOD
Warm-start autoload enable flag. Patch to zero if you want to disable the warm-start autoload feature (WARMSTRT.AUT).		
LOADFN	= 0,"OSMASTER","SYS"	OSLOAD
Default file name and drive (in FCB format) loaded by OSLOAD.COM. Drive field (FCB byte 0) may be patched to an explicit drive value to inhibit scanning.		

**Patch Points
(Continued)**

Symbol	Default Value	Module
LOGUSR	= 1F User number for logged-off state. Default value is 31 decimal.	FILCOM
MAXMBS	= 0 Maximum number of message buffers that will ever be allocated. Default value of 0 means number of message buffers is limited only to size of available memory.	NETMGR
MAXRPS	= 0 Maximum number of reply packets that will ever be allocated. Default value of 0 means number of reply packets is limited only to the size of available memory.	NETMGR
MEMBLL	= (1100) Memory base lower limit, prevents allocation of dynamic memory space below this address when bank 0 is selected. Default value guarantees minimum of 4K TPA in bank 0 (enough to run BANK or BUFFERS).	MEMMGR

Patch Points (Continued)

Symbol	Default Value	Module
MEMRES	= (0100)	LCLUSR
Memory reserve, used when loading a program into TPA to provide a safety margin between the base of dynamic memory space and the top of bank 0 TPA. This allows dynamic space to grow by MEMRES bytes before the program in bank 0 TPA has to be aborted by TurboDOS. The MEMRES value may have to be increased above the 256-byte default value for reliable operation especially in non-banked network masters.		
MEMTOP	= (0FFFF)	OSLOAD
Top of memory address for purposes of the RAM diagnostic test performed by OSLOAD. Patch to (0000) to omit test altogether.		
NMBCKT	= 1	NETTBL
Number of network circuits to which this processor is connected.		
NMBMBS	= 0	NETMGR
Number of message buffers pre-allocated at cold-start. Message buffers are allocated dynamically as needed, but this may cause fragmentation which prevents you from obtaining more TPA by reducing the size of the disk buffer pool. If this is important, patching NMBMBS to a suitable positive value will eliminate the problem (twice the number of network nodes is a good starting value to try).		

Patch Points (Continued)

Symbol	Default Value	Module
NMBRPS	= 0	NETMGR
Number of reply packets pre-allocated at cold-start. Reply packets are allocated dynamically as needed, but this may cause fragmentation which prevents you from obtaining more TPA by reducing the size of the disk buffer pool. If this is important, patching NMBRPS to a suitable positive value will eliminate the problem. (The number of network nodes is a good starting value to try.)		
NMBSVC	= 2	NETSVC
Number of network server processes to be activated. (The number of network nodes is a good starting value to try.)		
NMBUFS	= 4	BUFMGR
Default number of disk buffers allocated at cold-start. Must be at least 2. For optimum performance, allocate as many buffers as possible (consistent with TPA and other memory requirements).		
PRTCHR	= "^L"	CONTBL
End-print character (after attention). This is a console attention-response, not to be confused with EOPCHR.		

Patch Points
(Continued)

Symbol	Default Value	Module
PRTMOD	= 1	LCLTBL
	Initial print mode for local user. The default value of 1 specifies spooling. Patch to 0 for direct, or 2 for console.	
PTRAST	= 00,LSTDRA,0FF,(0000), 0FF,(0000),0FF,(0000),...	LSTTBL
	Printer assignment table, an array of 16 three-byte entries (one for each printer letter A-P) that defines which printers are local, remote, and invalid. For a local printer, the first byte must not have the sign-bit set. That byte is passed to the disk printerr, and is commonly defines the channel number (e.g., serial port) to be used for the printer. The following word specifies the entry-point address of the printer driver to be used. For a remote printer, the first byte must have the sign-bit set. The low-order bits of that byte specify the printer letter to be accessed on the remote processor. The following word specifies the network address of the remote processor. For an invalid printer, the entry should be 0FF,(0000). NOTE: In slave configurations STDSLAVE and STDSLAVX, the default values are: PTRAST = 80,(0000),81,(0000), 82,(0000),83,(0000), ...,8E,(0000),8F,(0000)	

Patch Points (Continued)

Symbol	Default Value	Module
QUEAST	= 00,(0000),0FF,(0000), 0FF,(0000),0FF,(0000),... Queue assignment table, an array of 16 three-byte entries (one for each queue letter A-P) that defines which queues are local, remote, and invalid. For a local queue, all three bytes must be set to zero. For a remote queue, the first byte must have the sign-bit set. The low-order bits of that byte specify the queue letter to be accessed on the remote processor. The following word specifies the network address of the remote processor. For an invalid queue, the entry should be 0FF,(0000). NOTE: In slave configurations STDSLAVE and STDSLAVX, the default values are: QUEAST = 80,(0000),81,(0000), 82,(0000),83,(0000), ...,8E,(0000),8F,(0000)	LSTTBL
QUEDLY	= (0000) Polling delay used in unconditional Read Queue (when queue is empty) and Write Queue (when queue is full), stated in system "ticks". If RTC driver is available, patch to largest delay that yields reasonable queue performance.	QUEMGR

Patch Points
(Continued)

Symbol	Default Value	Module
QUEDRV	= OFF	QUEMGR
	Drive used for FIFOs that emulate MP/M queues. Default value OFF means use the system disk (disk from which TurboDOS was loaded at cold-start). Patch to 00 - 0F to specify a particular drive A-P.	
QUEPTR	= 1	LCLTBL
	Initial queue or printer assignment. If PRTMOD = 1 (spooling), QUEPTR specifies a queue assignment. If PRTMOD = 0 (direct) QUEPTR specifies a printer assignment. In both cases, hex values 01 through 10 correspond to letters A-P, and zero means do not queue or print off-line.	
RCNMSK	= OFF	MPMSUP
	Mask used in deriving a console number from a network node in C-function 153.	
RCNOFF	= 0	MPMSUP
	Offset used in deriving a console number from a network node in C-function 153.	
RESCHR	= "^Q"	CONTBL
	Resume character (after attention).	

Patch Points (Continued)

Symbol	Default Value	Module
SCANDN	= 0	OSLOAD
	Scan direction flag for OSLOAD. Patch to 0FFH to scan P-to-A (instead of A-to-P).	
SLVFN	= "OSSLAVE ","SYS"	NETSVC
	Name and type of file (in FCB format) to be down-loaded into slave processors.	
SPLDRV	= 0FF	LCLTBL
	Initial spool drive. Default value 0FF spools to system disk (from which TurboDOS was loaded at cold-start). Patch to 00 - 0F to specify a drive A-P.	
SRHDRV	= 0	CMDINT
	Search drive for command files. Patch to 01 - 10 hex to search drive A-P if command is not found on current drive, or patch t0 0FF to search system disk (from which TurboDOS was loaded at cold-start). Default value 0 disables this feature.	
SUBFN	= 0,"$$$ ","SUB"	SUBMIT
	Submit file name searched for by optional CP/M submit-file emulator.	
WARMFN	= 0,"WARMSTRT","AUT"	AUTLOD
	File name and drive for warm-start auto-load processing (in FCB format).	

Network Operation

TurboDOS accomodates a wide variety of network topologies, ranging from the simplest point-to-point master/slave networks to the most complex star, ring, and hierarchical structures.

Network Model

A TurboDOS network is defined to consist of up to 255 circuits, with up to 255 nodes (processors) on each circuit. Each node has a unique 16-bit network address consisting of an 8-bit circuit number plus an 8-bit node number (on that circuit).

Any processor may be connected to several circuits, if desired. A processor connected to multiple circuits has multiple network addresses, one for each circuit. Such a processor even may be set up to perform message forwarding from one circuit to another, permitting dialogue between network nodes that do not share a common circuit between them (more on this later).

Network Tables

The actual network topology is defined by a series of tables in each processor. The tables are set up during system generation, and define the network as "seen" from the viewpoint of each processor. The tables are:

Symbol	Description
NMBCKT	A byte value that defines the number of network circuits to which this processor is connected.

Network Tables
(Continued)

Symbol	Description
CKTAST	The circuit assignment table containing NMBCKT entries defining the network address by which this processor is known on each circuit, and specifying the network circuit driver responsible for each handling each circuit.
DSKAST	The disk assignment table that specifies for all drive letters A-P which are local, remote, and invalid. This table specifies a network address for each remote drive, and a disk driver for each local drive.
PTRAST	The printer assignment table that specifies for all printer letters A-P which are local, remote, and invalid. This table specifies a network address for each remote printer, and a printer driver for each local printer.
QUEAST	The queue assignment table that specifies for all queue letters A-P which are local, remote, and invalid. This table specifies a network address for each remote queue.
DEFDID	The default network destination ID, used for routing all network requests that are not related to a specific disk drive, printer, or queue.

Network Tables
(Continued)

Symbol	Description
FWDTBL	The message forwarding table that specifies any additional circuits (not directly connected to this processor) which may be accessed via explicit message forwarding, and how messages destined for such circuits are to be routed.

These tables are pre-defined with default values to make set-up of simple master/slave networks very easy. For complex multi-circuit networks, the set-up is somewhat more complicated (as might be expected).

Refer to the preceding Patch Points sub-section for details of the organization and defaults for these network tables.

Message Forwarding

The forwarding module of TurboDOS (NETFWD) supports both "implicit" and "explicit" forwarding of network messages. To understand the distinction, consider the case of a network with three processors (P1, P2, and P3) connected by two circuits (C1 and C2) as follows:

```
 ____               ____              ____
|    |             |    |            |    |
| P1 |------C1------| P2 |------C2-----| P3 |
|____|             |____|            |____|
```

A program running in P1 makes an access to drive D. Suppose the disk assignment tables in the three processors are set up in the following fashion:

- P1's DSKAST defines its drive D as a remote reference to P2's drive B.
- P2's DSKAST defines its drive B as a remote reference to P3's drive A.
- P3's DSKAST defines its drive A as a local device attached directly to P3.

In this case, P1's access to its drive D actually winds up implicitly accessing P3's drive A. This is <u>implicit</u> forwarding.

Alternatively, suppose P1's DSKAST defines its drive D as a remote reference to P3's drive A, and that P1's FWDTBL provides that messages destined for circuit C2 may be routed via C1. In this case, P1 sends a request to P3 on circuit C1. P2 receives the request, recognizes that it should be forwarded, and retransmits the request to P3 via circuit C2. Thus, P1 accesses P3's drive A with the assistance of P2, but this time P1 is not aware of P2's role in the transaction. This is <u>explicit</u> forwarding.

A Complex Example

Let's take a reasonably complex network situation and see how to construct the required .GEN and .PAR files.

Our hardware is an S-100 microcomputer system consisting of a Z80 CPU board, a 64K memory board, hard disk and floppy disk controller boards (all these make up the master processor), and several single-board slave processors on the same bus. The master processor is interfaced to two printers, one daisywheel and the other matrix, via RS232 serial ports. The daisywheel printer is on serial port 0 and uses XON/XOFF protocol, while the matrix printer is on port 1 and uses clear-to-send handshaking. In addition, the master has a high-speed RS422 interface connecting it to another S-100 system of similar configuration some distance away.

We want to configure a TurboDOS system for this hardware that permits all of the users of each S-100 system to access the hard disk, floppy disks, and printers attached to both the local and remote S-100 system. We might create the following OSMASTER.GEN file:

```
; OSMASTER.GEN for complex example
STDMASTR ; standard master package
FASLOD   ; non-banked program load
NETREQ   ; to make requests of other sys
MSGFMT   ; needed by NETREQ
CONREM   ; no console on the master
LSTXON   ; XON/XOFF for daisy    (LSTDRA)
LSTCTS   ; CTS for matrix        (LSTDRB)
DSKHDC   ; hard disk controller  (DSKDRA)
DSKFDC   ; floppy disk control.  (DSKDRB)
CKTSLV   ; circuit driver for slaves (C0)
CKT422   ; circuit driver for RS422  (C1)
RTCDRV   ; real-time clock driver
NITDRV   ; hardware initialization driver
```

A Complex Example (Continued)

Our system generation task is completed by creating the companion OSMASTER.PAR file:

```
; OSMASTER.PAR for complex example
NMBCKT = 2               ; 2 net circuits
CKTAST = (0000),CKTDRA ; ckt 0 for slaves
         (0100),CKTDRB ; ckt 1 via RS422
DSKAST = 00,DSKDRA ; drv A is local HD
         00,DSKDRB ; drv B is local FD0
         01,DSKDRB ; drv C is local FD1
         80,(0101) ; drv D is remote HD
         81,(0101) ; drv E is remote FD0
         82,(0101) ; drv F is remote FD1
PTRAST = 00,LSTDRA ; ptr A is lcl daisy
         01,LSTDRB ; ptr B is lcl matrix
         80,(0101) ; ptr C is rmt daisy
         81,(0101) ; ptr D is rmt matrix
QUEAST = 00,(0000) ; queue A is local
         00,(0000) ; queue B is local
         80,(0101) ; queue C is remote A
         81,(0101) ; queue D is remote B
DEFDID = (0101)    ; default other master
DSPPAT = 1,2,3,4   ; assgn ptrs to queues
MEMRES = (0400)    ; 1K safety margin
NMBMBS = 0A        ; 10 message buffers
NMBRPS = 5         ; 5 reply packets
NMBSVC = 5         ; 5 server processes
NMBUFS = 14        ; 20 1K disk buffers
```

The generation of the second master operating system could be identical, except that all occurrences of network addresses (0100) and (0101) in the OSMASTER.PAR file would be reversed. Generation of the slave operating system would be very straightforward, and identical for both systems.

If you study this example thoroughly until you understand the reason for every .GEN and .PAR file entry, you should have little trouble setting up your own "sysgens".

Sysgen Procedure

To conclude this section, here is a suggested step-by-step procedure for generating a new version of TurboDOS:

1. Bring up a previous version of TurboDOS. If this is your first attempt to generate a TurboDOS system, you may bring up CP/M instead. However, if you are using CP/M, all disks will have to be in a format compatible with both CP/M and TurboDOS (e.g., eight-inch one-sided single-density with 128-byte sectors).

2. Make a working copy of your TurboDOS distribution disk. Do not use the original disk (in case something goes wrong). Insert the working diskette in a convenient disk drive.

3. Using your favorite text editor, create or revise the file OSMASTER.GEN containing the names of the relocatable modules to be linked together. Generally, this will consist of the appropriate STDxxxxx standard package plus selected additional modules and all required device drivers.

4. Using your editor once again, create or revise the file OSMASTER.PAR containing any required patches. This may be omitted if no patches are desired.

5. Using the command GEN OSMASTER.SYS, generate an executable system in accordance with the .GEN and .PAR files just constructed. If your hardware has less than 64K installed, don't forget to use the ;Uxxxx option on the GEN command. If your hardware has banked memory, don't forget to use the ;Kxxxx option.

Sysgen Procedure (Continued)

6. In a similar fashion, construct a new loader by creating or revising the files OSLOAD.GEN and OSLOAD.PAR, then using the command GEN OSLOAD.COM to generate the executable loader.

7. For a master/slave network system, construct a slave operating system in the same manner. Create or revise the files OSSLAVE.GEN and OSSLAVE.PAR, then use the command GEN OSSLAVE.SYS to generate the down-loadable slave operating system.

8. To test the newly-generated system, eject all disks other than your working disk (again, in case something goes wrong). Enter the command OSLOAD. The new system should cold-start. If it fails to come up or to function properly, you will have to start over at step 1 and check your work carefully -- there is most likely an error in one of your .GEN or .PAR files, or a "bug" in one of your drivers.

DISTRIBUTION

This section explains the TurboDOS distribution procedure in detail. It covers TurboDOS licensing requirements, and the obligations of licensed distributors, dealers, and end-users. It describes how to make up and serialize TurboDOS distribution disks.

Although this section is of concern primarily to licensed TurboDOS distributors, we've included it here so that dealers and end-users can gain a better perspective on the overall distribution process.

TurboDOS Licensing

TurboDOS is a proprietary software product of Software 2000, Inc. As such, it is protected by law against unauthorized use and reproduction. Authorization to use and/or reproduce TurboDOS is granted only by written license agreement.

Legal Protection

TurboDOS programs and documentation are copyrighted, which means it is against the law to make copies without express written authorization from Software 2000 to do so.

The word "TurboDOS" is a trademark owned by Software 2000 and registered in Class 9 (computer software) and Class 16 (documentation) with the trademark offices of the United States and most of the developed countries of the free world. This means it is against the law to make use of the TurboDOS trademark without express written authorization from Software 2000.

Software 2000 has licensed certain companies to distribute TurboDOS. Such distributors are authorized to use the TurboDOS trademark, and to reproduce, distribute, and sub-license TurboDOS programs and documentation to dealers and end-users.

User Obligations

TurboDOS may be used only after the user has paid the required license fee, signed a copy of the TurboDOS end-user license agreement, and returned the signed agreement to the issuing TurboDOS distributor. Then, TurboDOS may be used only in strict conformance with the terms of the license.

Each end-user license allows TurboDOS to be used on one specific computer system identified by make, model, and serial number. The end-user license may not be transferred from one computer system to another, and expressly forbids copying programs and documentation except as required for backup purposes only.

A separate license fee must be paid and a separate license signed for each computer system on which TurboDOS is used. Network slave computers that cannot operate stand-alone do not have to be licensed separately from the network master. (This would be the case, for example, if the slave computers have no local disk storage, or if TurboDOS is furnished in a form that cannot be run stand-alone on the slave computers.) However, networked computers that are also capable of stand-alone operation under TurboDOS must each be licensed separately.

Dealer Obligations

A dealer must sign a TurboDOS dealer agreement and return the signed agreement to the issuing distributor. Then, the dealer is permitted to purchase pre-serialized copies of TurboDOS programs and documentation from the distributor, and to resell them to end-users. Dealers may not reproduce TurboDOS programs or documentation for any purpose. Before delivering each copy of TurboDOS, the dealer must see to it that the end-user signs the TurboDOS end-user license agreement and returns it to the issuing distributor.

Distributor Obligations

Each licensed TurboDOS distributor is provided a master copy of TurboDOS relocatable modules and command programs on diskette. A distributor is allowed to reproduce and distribute copies of TurboDOS to dealers and end-users, but only in connection with certain specifically authorized hardware (usually manufactured or sold by the distributor). The distributor is required to serialize each copy of TurboDOS with a unique sequential magnetic serial number, and to register each serial number promptly with Software 2000. (Serialization is described in more detail below.)

Each distributor is also provided with a master copy of TurboDOS documentation, either in camera-ready hardcopy or in ASCII files on disk. The distributor is responsible for reproducing the documentation and furnishing it with each copy of TurboDOS it issues.

A distributor must require each dealer to sign and return a TurboDOS dealer agreement before issuing copies of TurboDOS to the dealer for resale. A distributor must require each end-user to sign and return a TurboDOS end-user license agreement before issuing a copy of TurboDOS directly to the end-user.

Serialization

Each copy of TurboDOS is magnetically serialized with a unique serial number. Such serialization helps ensure that reproduction and distribution of TurboDOS is done in strict accordance with the required licensing and registration procedures, and facilitates tracing of unlicensed copies of the software.

Each relocatable module of TurboDOS distributed to a dealer or end-user has a magnetic serial number composed of two parts:

- an origin number that identifies the issuing distributor, and
- a sequential unit number that uniquely identifies each copy of TurboDOS issued by that distributor.

During system generation, the GEN command verifies that all modules making up a TurboDOS configuration are serialized consistently, and magnetically serializes the resulting executable version of TurboDOS accordingly.

The relocatable modules on the master disk furnished to each licensed TurboDOS distributor are partially serialized with an origin number only. Each distributor is provided a serialization program (SERIAL.COM) that must be used to add a unique sequential unit number to each copy of TurboDOS issued by the distributor. The GEN command will not accept partially-serialized modules that have not been serialized with a unit number. Conversely, the SERIAL command will not re-serialize modules that have already been fully serialized.

Technical Support

Software 2000 maintains telephone and telex "hot-lines" to provide TurboDOS technical assistance to its distributors. These are unlisted numbers providing direct access to the authors of the TurboDOS operating system, and are furnished only to licensed TurboDOS distributors. We encourage distributors to take advantage of this service whenever technical questions or problems arise in using or configuring TurboDOS.

It is the responsibility of each licensed distributor to provide technical support to its dealers and end-user customers. Software 2000 cannot assist dealers or end-users directly. Where exceptional circumstances seem to require direct contact between Software 2000 technical personnel and a dealer or end-user, this must be handled strictly by prior arrangement between Software 2000 and the distributor.

SERIAL Command

The SERIAL command enables TurboDOS distributors to magnetically serialize relocatable modules of TurboDOS for distribution.

Syntax

```
SERIAL srcefile destfile ;Unnn {options}
SERIAL ;Unnn {options}
```

Explanation

The SERIAL command works exactly like the COPY command, and accepts exactly the same arguments and options. However, SERIAL has the additional function of magnetically serializing relocatable modules as they are copied. SERIAL serializes files of type .REL (Z80 modules) and type .O (8086 modules). Other files are copied without any change.

The unit number must be specified on the command line as ;Unnn, where "nnn" represents a decimal unit number in the range 0-65535. Unit numbers must be assigned sequentially, starting with 1. Unit number 0 is reserved by convention for in-house use by the distributor.

SERIAL produces fully-serialized modules that are encoded with the distributor's origin number and the specified unit number. GEN does not accept TurboDOS modules unless they have been fully serialized in this fashion.

Options

Option	Explanation
SERIAL accepts all COPY options, plus:	
;Unnn	Relocatable modules (type .REL or .O) are magnetically serialized with unit number nnn, which must be a decimal integer in the range 0 to 65535. This "option" is mandatory for SERIAL.

Example

```
0A}SERIAL *.REL B: ;U289N
0A:AUTLOD  .REL copied to 0B:AUTLOD.  REL
0A:AUTLOG  .REL copied to 0B:AUTLOG.  REL
                     :
0A:SYSNIT.  REL copied to 0B:SYSNIT.  REL
0A}
```

Error Messages

SERIAL incorporates all COPY error messages, plus:

Unit number not specified
Origin number violation
File is already serialized
Unexpected EOF in .0 or .REL file

PACKAGE Command

The PACKAGE command lets you combine any collection of relocatable modules into a single concatenated .REL file.

Syntax

```
PACKAGE srcefile {destfile}
```

Explanation

PACKAGE may be used to construct custom packages of TurboDOS modules, make additions or changes to the supplied STDxxxxx packages, pre-package collections of driver modules, and so forth.

The "srcefile" argument specifies the name of an input file "srcefile.PKG" that lists the modules to be packaged. The "destfile" argument specifies the name of the concatenated .REL file to be created. If "destfile" is omitted, then the "srcefile" argument is also used as the name of the output .REL file.

If the .PKG file is found, it must contain the list of relocatable modules (.REL files) to be linked together. If the configuration file is not found, then the PACKAGE command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
relfile {,relfile}... {;comment}
```

A null directive terminates the prompting sequence and causes processing to proceed.

After obtaining the list of modules from the file or console, PACKAGE concatenates all of the modules together (displaying the name of each module as it is encountered) and writes the result to the output file.

Example

```
0A}PACKAGE STDLOADR
* ; STDLOADR.PKG standard loader package
* OSLOAD,LDRMSG,OSNTRY,FILMGR,FILSUP
* FILCOM,BUFMGR,DSKMGR,DSKTBL,NONFIL
* CONMGR,CONTBL,DSPSGL,COMSUB
OSLOAD LDRMSG OSNTRY FILMGR FILSUP etc.
0A}
```

Error Messages

```
File name missing from command
Invalid input file name
Non-privileged user
Unexpected EOF in input file
Disk is full
Can't make output file
Can't open input file
No input files
```

Distribution Procedure

Here is the procedure to be followed by distributors when creating each copy of TurboDOS to be issued to a dealer or end-user:

1. Assign a unique sequential unit number for this copy of TurboDOS, and register it immediately by filling out a serial number registration card (or agreed-to substitute) and mailing to Software 2000, Inc.

2. Format a new disk, and label it with the following information clearly legible:

   . trademark TurboDOS^{R}

   . version number (1.4x)

   . origin and unit numbers (oo/uuuu)

   . statutory copyright notice:
   Copyright 198x by Software 2000, Inc.
   All rights reserved.

3. Use the SERIAL command to copy and serialize the appropriate files from your distribution master disk to the new disk. Use the tables on the following page to guide you in determining what files to put on the new disk.

   IMPORTANT NOTE: Be absolutely certain that the new disk does <u>not</u> contain any unserialized modules or SERIAL.COM!

4. Using the new serialized disk, use the GEN command to generate an executable loader and operating system. Follow the system generation procedure described in the previous section.

5. In addition to the serialized disk, you should issue copies of TurboDOS documentation and a start-up PROM (if applicable).

Distribution Procedure (Continued)

The following table may be used for guidance in preparing TurboDOS disks for distribution. In addition to the files shown, you need to include hardware-dependent driver modules and utility programs as appropriate.

single-user w/o spooler	single-user with spooler	multi-user networking
STDLOADR.REL	STDLOADR.REL	STDLOADR.REL
STDSINGL.REL	STDSINGL.REL	STDSINGL.REL
-	STDSPOOL.REL	STDSPOOL.REL
-	-	STDMASTR.REL
-	-	STDSLAVE.REL
-	-	STDSLAVX.REL
FASLOD .REL	FASLOD .REL	FASLOD .REL
BNKMGR .REL	BNKMGR .REL	BNKMGR .REL
CPMSUP .REL	CPMSUP .REL	CPMSUP .REL
MPMSUP .REL	MPMSUP .REL	MPMSUP .REL
RTCNUL .REL	RTCNUL .REL	RTCNUL .REL
PATCH .REL	PATCH .REL	PATCH .REL
SUBMIT .REL	SUBMIT .REL	SUBMIT .REL
OSBOOT .REL	OSBOOT .REL	OSBOOT .REL
-	-	NETLOD .REL
-	-	NETREQ .REL
-	-	NETFWD .REL
-	-	BNKREQ .REL
-	-	MSGFMT .REL
-	-	NETSVC .REL
-	-	QUEMGR .REL
-	-	CONREM .REL
AUTOLOAD.COM	AUTOLOAD.COM	AUTOLOAD.COM
BACKUP .COM	BACKUP .COM	BACKUP .COM
BANK .COM	BANK .COM	BANK .COM
-	-	BATCH .COM
BOOT .COM	BOOT .COM	BOOT .COM
BUFFERS .COM	BUFFERS .COM	BUFFERS .COM
-	-	CHANGE .COM
COPY .COM	COPY .COM	COPY .COM
DATE .COM	DATE .COM	DATE .COM

Distribution Procedure (Continued)

single-user w/o spooler	single-user with spooler	multi-user networking
DELETE .COM	DELETE .COM	DELETE .COM
DIR .COM	DIR .COM	DIR .COM
DO .COM	DO .COM	DO .COM
DRIVE .COM	DRIVE .COM	DRIVE .COM
DUMP .COM	DUMP .COM	DUMP .COM
ERASEDIR.COM	ERASEDIR.COM	ERASEDIR.COM
-	-	FIFO .COM
FIXDIR .COM	FIXDIR .COM	FIXDIR .COM
FIXMAP .COM	FIXMAP .COM	FIXMAP .COM
FORMAT .COM	FORMAT .COM	FORMAT .COM
GEN .COM	GEN .COM	GEN .COM
LABEL .COM	LABEL .COM	LABEL .COM
-	-	LOGOFF .COM
-	-	LOGON .COM
-	-	MASTER .COM
PRINT .COM	PRINT .COM	PRINT .COM
-	PRINTER .COM	PRINTER .COM
-	QUEUE .COM	QUEUE .COM
-	-	RECEIVE .COM
RELCVT .COM	RELCVT .COM	RELCVT .COM
RENAME .COM	RENAME .COM	RENAME .COM
-	-	SEND .COM
SET .COM	SET .COM	SET .COM
SHOW .COM	SHOW .COM	SHOW .COM
TYPE .COM	TYPE .COM	TYPE .COM
VERIFY .COM	VERIFY .COM	VERIFY .COM

CODING CONVENTIONS

This section is devoted to in-depth discussion of TurboDOS internal coding conventions, aimed at the systems programmer writing hardware-dependent drivers or resident processes.

Assembler Notes

Drivers and resident processes for Z80 TurboDOS must be written using a Z80 assembler capable of producing relocatable modules with symbolic linkage information in the industry-standard Microsoft relocatable module format. Both Microsoft's M80 and Digital Research's RMAC assemblers produce object code in this format, and are fine choices for use with TurboDOS.

Another excellent relocatable Z80 assembler is PASM from Phoenix Software Associates. However, PASM produces object modules in a non-standard format.

To make it possible for PASM to be used with TurboDOS, a conversion utility (RELVCT.COM) for converting PASM object modules to standard Microsoft format is furnished with TurboDOS. The command:

```
RELCVT filename
```

converts the specified PASM-format .REL file into Microsoft .REL format. During conversion, the character . is converted to ?, and the character % is converted to @ wherever these characters appear in symbol names.

Assembler Notes (Continued)

Programming examples and driver listings in this document are coded for PASM. If you are used to another assembler, please take note of certain syntax features of PASM which may be different in other assemblers.

Names followed by £ are external references to public names defined in other modules. Labels followed :: are public names available for reference in other modules. Some assemblers require such names to be declared using an EXTERN or PUBLIC directive.

Program, data, and common segments are introduced with a .LOC directive. Other assemblers use different directives such as CSEG, DSEG, COMMON, etc. to accomplish the same thing.

Finally, the symbol . represents the current location counter value. Some assemblers use $ or * instead.

Undefined External References

To allow various TurboDOS modules to be included or omitted at will, the GEN command automatically resolves all undefined external references to the default symbol public ?UND? (.UND. using PASM). The common subroutine module COMSUB contains the following subroutine:

```
.UND.:: NOP             ;two bytes of zero
        NOP             ; "    "    "   "
        XRA   A         ;clear A to zero
        RET             ;done
```

Thus, it is always safe to load or call an external name, whether or not it is present at GEN time. It is bad form to store into an undefined external name, however!

Memory Allocation

The TurboDOS resident occupies the topmost portion of memory in a Z80 system. A common memory management module MEMMGR provides dynamic allocation and deallocation of memory space required for disk and message buffers, print queues, file and record locks, do-file nesting, and so forth. Memory segments are allocated downward from the base of the TurboDOS resident, reducing the space available for TPA. Deallocated segments are concatenated with any neighbors and threaded on a free-memory list. A best-fit algorithm is used to reduce memory fragmentation.

Allocation and deallocation requests are coded in this manner:

```
;code to allocate a memory segment
        LXI   H,36     ;HL=segment size
        CALL  ALLOC£   ;allocate segment
        ORA   A        ;alloc successful?
        JNZ   ERROR    ;NZ -> not enuf mem
        PUSH  H        ;HL=segment address
          :
;code to deallocate a memory segment
        POP   H        ;HL=segment address
        CALL  DEALOC£  ;deallocate segment
```

ALLOC£ prefixes each allocated segment with a word containing the segment length, so that DEALOC£ can tell how much memory is to be deallocated. ALLOC£ does not zero the newly-allocated segment.

List Processing

TurboDOS maintains its dynamic structures as threaded lists with bidirectional linkages. This technique permits a node to be added or deleted anywhere in a list without searching. The list head and each list node have a two-word linkage (forward and backward pointers).

List manipulation is coded in this manner:

```
        .LOC  .DATA.£  ;data segment
;list head (linkage initialized empty)
LSTHED: .WORD LSTHED   ;forward pointer
        .WORD LSTHED   ;backward pointer

;list node (linkage not initialized)
LSTNOD: .WORD 0        ;forward pointer
        .WORD 0        ;backward pointer
        .BYTE [128]0   ;contents of node

        .LOC  .PROG.£  ;program segment
;code to add node to end of list
        LXI   H,LSTHED ;HL=head address
        LXI   D,LSTNOD ;DE=node address
        CALL  LNKEND£  ;link to list end

;code to unlink node from list
        LXI   H,LSTNOD ;HL=node address
        CALL  UNLINK£  ;unlink node

;code to add node to beginning of list
        LXI   H,LSTHED ;HL=head address
        LXI   D,LSTNOD ;DE=node address
        CALL  LNKBEG£  ;link to list beg.
```

Task Dispatching

TurboDOS incorporates a flexible, efficient mechanism for dispatching the Z80 processor among various competing processes. In coding drivers for TurboDOS, you must take extreme care to use the dispatcher correctly in order to attain maximum system performance.

The dispatcher allows one process to wait for some event (for example, data-available or seek-complete) while allowing other processes to use the processor. For each such event, you must define a three-word structure called a "semaphore".

A semaphore consists of a count-word followed by a two-word list head. The count-word is used by the dispatcher to keep track of the status of the event. (At present, only the LSB of the count word is used, supporting counts in the range -128 to +127.) The list head anchors a threaded list of processes waiting for the event to occur.

Two primitive operations operate on a semaphore: waiting for the event to occur (WAIT£), and signalling that the event has occurred (SIGNAL£). They are coded in this following manner:

```
;this semaphore represents some event
EVENT:  .WORD 0         ;semaphore count
        .WORD EVENT+2   ;semaphore f-ptr
        .WORD EVENT+2   ;semaphore b-ptr

;wait for the event to occur
        LXI   H,EVENT   ;HL=semaphore addr
        CALL  WAIT£     ;wait for event

;signal that event has occurred
        LXI   H,EVENT   ;HL=semaphore addr
        CALL  SIGNAL£   ;signal event
```

Task Dispatching (Continued)

Whenever a process waits on a semaphore, WAIT£ decrements the semaphore's count-word. Thus, a negative count -N signifies that there are N processes waiting for the event to occur. Whenever an event is signalled, SIGNAL£ increments the semaphore count-word and awakens the process that has been waiting longest.

If an event is signalled but no process is waiting for it, then SIGNAL£ increments the count-word to a positive value. Thus, a positive count N signifies that there have been N occurrences of the event for which no process was waiting. In this case, the next N calls to WAIT£ on that semaphore will return immediately without waiting.

Sometimes it is necessary for a process to wait for a specific time interval (for example, a motor-start delay or carriage-return delay) rather than for a specific event. TurboDOS provides a delay facility (DELAY£) that permits other processes to use the Z80 while one process is waiting for such a timed delay. Delay intervals are specified as some number of "ticks". A tick is an implementation-defined interval, usually 1/50 or 1/60 of a second. Delays are coded thus:

```
;delay for one-tenth of a second
        LXI     H,6       ;HL=delay in ticks
        CALL    DELAY£    ;delay process
```

Accuracy of delays is usually plus-or-minus one tick. A delay of zero ticks may be specified to relinquish the processor to other processes on a "courtesy" basis.

All driver delays should be accomplished via WAIT£ or DELAY£, <u>never</u> by spinning in a loop.

Interrupt Service

Dispatching is especially efficient when used with interrupt-driven devices. Usually, the interrupt service routine just calls SIGNAL£ to signal the interrupt-associated event.

Most interrupt service routines should exit via the usual EI/RETI sequence. However, some periodic interrupt (usually a 50 or 60 hertz clock interrupt) should have an interrupt service routine that exits by jumping to the dispatcher entrypoint ISRXIT£ (without enabling interrupts) to provide periodic time-slicing of processes. To avoid excessive dispatcher overhead, don't use ISRXIT£ more than about 60 times per second.

It is good programming practice for interrupt service routines to set up an auxilliary stack, in order to avoid the possibility of overflowing the stack area of some transient program. TurboDOS provides a standard interrupt stack area INTSTK£ and stack pointer save location INTSP£. A simple interrupt service routine might be coded like this:

```
DEVISR: SSPD   INTSP£     ;save user SP
        LXI    SP,INTSTK£  ;SP=aux stack
        PUSH   PSW        ;save registers
        PUSH   B          ;  "        "
        PUSH   D          ;  "        "
        PUSH   H          ;  "        "
        IN     PORT       ;reset interrupt
        LXI    H,EVENT    ;HL=semaphore addr
        CALL   SIGNAL£    ;signal event
        POP    H          ;restore registers
        POP    D          ;   "         "
        POP    B          ;   "         "
        POP    PSW        ;   "         "
        LSPD   INTSP£     ;restore user SP
        EI                ;enable interrupts
        RETI              ;return from int.
```

Poll Routines

Devices incapable of interrupting the Z80 have to be polled by the driver. The dispatcher maintains a threaded list of poll routines, and executes them every dispatch. The function of each poll routine is to check the status of its device, and to signal the occurrence of some event (for example, data-available) when it occurs. The routine LNKPOL£ links a poll routine onto the poll list, and UNLINK£ removes it.

A poll routine must be coded so that it will not signal the occurrence of a particular event more than once. The best way to assure this is for the poll routine to unlink itself from the poll list as soon as it has signalled the event. An example:

```
EVENT:  WORD  0         ;semaphore
        WORD  EVENT+2
        WORD  EVENT+2

;driver waits for event
        LXI   D,POLNOD  ;DE=poll node addr
        CALL  LNKPOL£   ;activate poll rtn
        CALL  POLRTN    ;optional pretest
        LXI   H,EVENT   ;HL=semaphore addr
        CALL  WAIT£     ;wait for event
          :

;poll routine signals event when detected
POLNOD: .WORD 0         ;poll rtn linkage
        .WORD 0         ;  "   "     "
POLRTN: IN    PORT      ;get device status
        ANI   MASK      ;did event occur?
        RZ              ;if not, exit
        LXI   H,EVENT   ;HL=semaphore addr
        CALL  SIGNAL£   ;signal event
        LXI   H,POLNOD  ;HL=linkage addr
        CALL  UNLINK£   ;unlink poll rtn
        RET             ;all done
```

Mutual Exclusion

TurboDOS is fully re-entrant at the process and kernel levels. However, most driver modules are not coded re-entrantly (since most peripheral devices can only do one thing at a time). Consequently, most drivers must make use of a mutual-exclusion interlock to prevent TurboDOS from invoking them re-entrantly.

This is very easy to accomplish using the basic semaphore mechanism of the dispatcher. It is only necessary to define a semaphore with its count-word initialized to 1 (instead of 0). Mutual exclusion may then be accomplished by calling WAIT£ upon entry and SIGNAL£ upon exit. An example:

```
;mutual-exclusion semaphore
MXSPH:  .WORD 1           ;count-word=1!
        .WORD MXSPH+2
        .WORD MXSPH+2

DRIVER: LXI   H,MXSPH     ;HL=semaphore addr
        CALL  WAIT£       ;wait if in-use
          :


          :
        LXI   H,MXSPH     ;HL=semaphore addr
        CALL  SIGNAL£     ;unlock mut-excl
        RET               ;done
```

Interrupt Status

To permit reliable testing of the interrupt status (enabled or disabled) of the Z80 CPU, TurboDOS provides the subroutine TSTIFF£. It is called with no arguments, and returns with the carry-flag set if and only if interrupts are disabled.

Sample Driver Using Interrupts

Here is a simple device driver for an interrupt-driven serial input device. It illustrates coding techniques discussed so far:

```
MXSPH:  .WORD 1        ;MX semaphore
        .WORD MXSPH+2
        .WORD MXSPH+2
RDASPH: .WORD 0        ;RDA semaphore
        .WORD RDASPH+2
        .WORD RDASPH+2
CHRSAV: .BYTE 0        ;saved input char
;device driver main code
INPDRV::LXI   H,MXSPH  ;HL=MX semaph addr
        CALL  WAIT£    ;lock MX
        EI             ;need ints enabled
        LXI   H,RDASPH ;HL=semaphore addr
        CALL  WAIT£    ;wait data avail
        LDA   CHRSAV   ;get input char
        PUSH  PSW      ;save on stack
        LXI   H,MXSPH  ;HL=MX semaph addr
        CALL  SIGNAL£  ;unlock MX
        POP   PSW      ;return char in A
        RET            ;done
;interrupt service routine
INPISR::SSPD  INTSP£   ;save user's SP
        LXI   SP,INTSTK£ ;SP=aux stack
        PUSH  PSW      ;save registers
        PUSH  B        ;  "     "
        PUSH  D        ;  "     "
        PUSH  H        ;  "     "
        IN    PORT     ;get input char
        STA   CHRSAV   ;save for driver
        LXI   H,RDASPH ;HL=semaphore addr
        CALL  SIGNAL£  ;signal data avail
        POP   H        ;restore registers
        POP   D        ;   "        "
        POP   B        ;   "        "
        POP   PSW      ;   "        "
        LSPD  INTSP£   ;restore user SP
        EI             ;enable interrupts
        RETI           ;return from int.
```

Sample Driver Using Polling

Here is a simple device driver for non-interrupting serial input device. It illustrates how polling is used:

```
MXSPH:  .WORD 1           ;MX semaphore
        .WORD MXSPH+2
        .WORD MXSPH+2
RDASPH: .WORD 0           ;RDA semaphore
        .WORD RDASPH+2
        .WORD RDASPH+2
CHRSAV: .BYTE 0           ;saved input char
;device driver main code
INPDRV::LXI   H,MXSPH   ;HL=MX semaph addr
        CALL  WAIT£     ;lock MX
        LXI   D,POLNOD  ;DE=poll rtn node
        CALL  LNKPOL£   ;activate poll rtn
        CALL  POLRTN    ;optional pretest
        LXI   H,RDASPH  ;HL=semaphore addr
        CALL  WAIT£     ;wait data avail
        LDA   CHRSAV    ;get input char
        PUSH  PSW       ;save on stack
        LXI   H,MXSPH   ;HL=MX semaph addr
        CALL  SIGNAL£   ;unlock MX
        POP   PSW       ;return char in A
        RET             ;done
;device poll routine with linkage
POLNOD: .WORD 0           ;poll rtn linkage
        .WORD 0
POLRTN: IN    STATUS    ;get device status
        ANI   MASK      ;data available?
        RZ              ;if not, exit
        IN    DATA      ;get input char
        STA   CHRSAV    ;save for driver
        LXI   H,RDASPH  ;HL=semaphore addr
        CALL  SIGNAL£   ;signal data avail
        LXI   H,POLNOD  ;HL=linkage addr
        CALL  UNLINK£   ;unlink poll rtn
        RET             ;done
```

Special Segments

In addition to the usual code and data segments, GEN command supports three special location counters (common blocks):

M80/RMAC	PASM	Description
?INIT?	.INIT.£	Initialization code
?PAGE?	.PAGE.£	Page-boundary aligned
?BANK?	.BANK.£	Banked-memory common

?INIT? Segment

In coding driver modules, you will often find a considerable amount of initialization code that is executed only once at cold-start and never needed again. By assembling such code under ?INIT? (.INIT.£ using PASM), it will be loaded and executed in lower memory (TPA), and will not occupy space in the resident operating system.

?PAGE? Segment

Sometimes you may need to force a segment of code or data to begin on a 256-byte page boundary. Examples are the simulated CP/M BIOS branch table, and interrupt vectors for Z80 interrupt mode 2. By assembling under ?PAGE? (.PAGE.£ using PASM), the segment is guaranteed to be page-aligned.

?BANK? Segment

In banked-memory implementations, you need to be able to place certain code and data in the topmost part of memory which is common to both banks (not switched). Anything assembled under ?BANK? (.BANK.£ using PASM) will be assigned to this common region (as specified by the ;Kxxxx option on the GEN command).

Inter-Process Messages

To pass messages from one process to another, a five-word structure called a "message node" is used. A message node consists of a three-word semaphore followed by a two-word message list head. Routines are provided for sending messages to a message node (SNDMSG£), and receiving messages from a message node (RCVMSG£). Typically, the sending process allocates a memory segment in which to build the message, and the receiving process deallocates the segment after reading the message. The first two words of each message must be reserved for a list-processing linkage. Coding is done in this manner:

```
;message node
MSGNOD: .WORD 0          ;semaphore part
        .WORD MSGNOD+2  ;    "       "
        .WORD MSGNOD+2  ;    "       "
        .WORD MSGNOD+6  ;message list head
        .WORD MSGNOD+6  ;   "      "    "

;one process allocates/builds/sends msg
        LXI   H,12+4    ;HL=message size+4
        CALL  ALLOC£    ;allocate segment
        PUSH  H         ;save segment addr
          :             ;build msg in seg
        POP   D         ;DE=message addr
        LXI   H,MSGNOD  ;HL=msg node addr
        CALL  SNDMSG£   ;send message

;other process reads/deallocates message
        LXI   H,MSGNOD  ;HL=msg node addr
        CALL  RCVMSG£   ;receive message
        PUSH  H         ;save message addr
          :             ;process message
        POP   H         ;HL=segment addr
        CALL  DEALOC£   ;deallocate seg
```

Console Routines

TurboDOS includes several handy console I/O subroutines which may be called from within driver modules as illustrated:

```
;raw console I/O routines
        CALL  CONST£   ;get status in A
        ORA   A        ;input char avail?
        RZ             ;if not, exit
        CALL  CONIN£   ;get input in A
        CALL  UPRCASE£ ;make upper-case
        MOV   C,A      ;C=character
        CALL  CONOUT£  ;output chr from C

;message output routines
;last char of message has sign-bit set
        CALL  DMS£     ;output following
        .ASCIS "This is a message"
        LXI   H,MSGADR ;HL=message addr
        CALL  DMSHL£   ;output msg @ HL

;binary-to-decimal output routine
        LXI   H,31416  ;HL=word value
        CALL  DECOUT£  ;displays decimal
```

Sign-On Message

You may add your own custom sign-on message to TurboDOS. Your message will be displayed at cold-start immediately following the normal TurboDOS sign-on and copyright notice.

Your sign-on message must be coded as an ASCII character string terminated with a $ delimiter, and labelled with the public entry symbol USRSOM. An example:

```
USRSOM::.ASCII [0DH] [0AH]
        .ASCII "Implementation by "
        .ASCII "Trigon Computer Corp."
        .ASCII "$"
```

Resident Process

You can code a resident process that runs in the background concurrent with other system activities, and link it into TurboDOS. The create-process subroutine CRPROC£ may be called to create such a process at cold-start as shown:

```
        .LOC  .INIT.£  ;init code
HDWNIT::LXI   H,64     ;HL=workspace size
        CALL  ALLOC£   ;alloc workspace
                       ;HL=workspace addr
        LXI   D,MYPROC ;DE=entrypoint add
        CALL  CRPROC£  ;create process
          :

        .LOC  .PROG.£  ;code segment
MYPROC: INR   COUNT(Y) ;increment counter
        LXI   D,60*60  ;1 minute in ticks
        MVI   C,2      ;T-function 2
        CALL  OTNTRY£  ;delay 1 minute
        JMP   MYPROC   ;loop forever
```

CRPROC£ automatically allocates a TurboDOS process area (address appears in register X) and a stack area (address appears in SP). If the process requires a re-entrant workspace, it should be allocated with ALLOC£ and passed to CRPROC£ in HL (as shown above), and will appear to the new process in register Y.

The resident process must make all operating system requests by calling OCNTRY£ or OTNTRY£ with a C-function or T-function number register C. It <u>must</u> <u>not</u> call location 0005H or 0050H in the base page, nor make direct calls on kernel routines such as WAIT£, SIGNAL£, DELAY£, SNDMSG£, RCVMSG£, ALLOC£, and DEALOC£.

Resident Process (Continued)

A resident process is not attached to a console, so any console I/O requests will be ignored.

You can do file processing within a resident process, using the normal C-functions open, close, read, write, and so forth, called via OCNTRY£. First, however, you must remember to warm-start with C-function 0 (OCNTRY£), and then log-on with T-function 14 (OTNTRY£).

A resident process must always be coded to preserve the contents of index register X, which Turbodos relies upon as a pointer to its process area. The process may use all other registers as desired.

User-Defined Function

The User-Defined Function (T-function 41) provides a means of adding your own special functions to the normal TurboDOS repertoire of C-functions and T-functions. To do this, you simply create a function processor subroutine with the public entrypoint symbol USRFCN.

Whenever a program invokes T-function 41, TurboDOS transfers control to your USRFCN routine. On entry, register BC contains the address of the 128-byte record area passed from the caller's current DMA address, and registers DE and HL contain whatever values the caller loaded into them. Your USRFCN routine may return data to the caller in the 128-byte record area (address in BC at entry) and in any of the registers A-B-C-D-E-H-L.

Architecturally, your USRFCN routine is inside the TurboDOS kernel. Consequently, it may call kernel subroutines directly. Any calls to C-functions and T-functions must therefore be made by means of two special recursive entrypoints: XCNTRY£ and XTNTRY£.

DRIVER INTERFACE

This section explains how to code hardware-dependent device driver modules, and presents formal interface specifications for each category of driver required by TurboDOS.

Following this section is a large appendix that contains assembler source listings of actual driver modules. The sample drivers cover a wide range of peripheral devices, and provide an excellent starting point for your driver development work.

General Notes

Drivers modules are coded with standard public entrypoint names, and linked to TurboDOS using the GEN command. You may package your drivers into as many or few separate modules as you like. In general, it is easier to reconfigure TurboDOS for a variety of devices if the driver for each device is packaged as a separate module.

TurboDOS is designed to accomodate multiple disk, console, printer, and network drivers. For disk drivers, for instance, the DSKAST is normally set up to refer to disk driver entrypoints DSKDRA£, DSKDRB£, DSKDRC£, and so forth. Each disk driver should be coded with the public entrypoint DSKDR@ (DSKDR% using PASM). The GEN command automatically maps successive definitions of such names by replacing the trailing @ by A, B, C, etc. The same technique may be used for console, printer, and network driver entrypoints.

You must code driver routines to preserve the stack and index registers X and Y, but you may use other registers as desired.

Initialization

Hardware initialization and interrupt vector set-up should be performed in an initialization routine labelled with the public entry symbol HDWNIT::. TurboDOS calls this routine during cold-start with interrupts disabled.

Your HDWNIT:: routine _must_ _not_ enable interrupts or make calls to WAIT£ or DELAY£. In most cases, HDWNIT:: will contain a series of calls to individual driver initialization subroutines contained in other modules.

One-time initialization code that is not needed again should be assembled under the special location counter ?INIT?, so that it doesn't take up space in the resident operating system.

Console Driver

A console driver should be labelled with the public entry symbol CONDR@ (CONDR%:: using PASM). A console number (from CONAST) is passed in register B. The driver must perform a console I/O operation according to the operation code passed in register E:

E-reg	Function
0	Return status in A, char in C
1	Return input character in A
2	Output character passed in C
8	Enter error-message mode
9	Exit error-message mode
10	Conditional output char in C

If E=0, the driver determines if a console input character is available. If no character is available, the driver returns A=0. If an input character is available, the driver returns A=-1 and the input character in C, <u>but must not "consume" the character</u>. TurboDOS depends upon this look-ahead capability to detect attention requests. The driver must not dispatch (via WAIT£ or DELAY£) when processing an E=0 call.

If E=1, the driver obtains an input character (waiting if necessary) and returns it in A.

If E=2, the driver displays the output character passed in C (waiting if necessary).

If E=8, the driver prepares to display a TurboDOS error message; if E=9, it reverts to normal. TurboDOS always precedes each error message with an E=8 call and follows it with an E=9 call. This gives the driver an opportunity to take special action (25th line, reverse video, etc.) for error messages. For simple consoles, the driver should output a CR-LF in response to E=8 and E=9 calls.

Console Driver (Continued)

If E=10, the driver determines whether or not it can accept a console output character without dispatching (via WAIT£ or DELAY£). If so, it outputs the character passed in C, and returns A=-1 to indicate that the character was accepted. However, if the driver cannot accept a console output character without dispatching, it returns A=0 to indicate that the character was not accepted; TurboDOS will then make an E=2 call to output the same character. This special conditional output call is used by TurboDOS to optimize console output speed by avoiding certain dispatch-related overhead whenever possible.

You should make a special effort to code the console driver to execute the minimum number of instructions possible, especially functions 0, 2, and 10. Excessive use of subroutine calls, stack operations, and other time-consuming coding techniques can make the difference between running the console device at full rated speed or something less. Study the sample driver listings in the appendix with this in mind.

Printer Driver

A printer driver should be labelled with the public entry symbol LSTDR@ (LSTDR%:: using PASM). A printer number (from PTRAST) is passed in register B. The driver must perform a printer output operation according to the operation code passed in register E:

E-reg	Function
2	Print character passed in C
7	Perform end-of-print-job action

If E=2, the driver prints the output character passed in C (waiting if necessary).

If E=7, the driver takes any appropriate end-of-print-job action. This is quite hardware-dependent, and may include slewing to top-of-form, homing the print head, dropping the ribbon, and so forth.

Disk Driver

A disk driver should be labelled with the public entry symbol DSKDR@ (DSKDR%:: using PASM). The driver performs the physical disk operation specified by the Physical Disk Request (PDR) packet whose address is passed by TurboDOS in index register X. The structure of the PDR packet is:

```
 Offset |             Contents

 ;physical disk request (PDR) packet
  0(X)    .BYTE OPCODE   ;operation code
  1(X)    .BYTE DRIVE    ;drive (base 0)
  2(X)    .WORD TRACK    ;track (base 0)
  4(X)    .WORD SECTOR   ;sector (base 0)
  6(X)    .WORD SECCNT   ;£sectors to rd/wr
  8(X)    .WORD BYTCNT   ;£bytes to rd/wr
 10(X)    .WORD DMAADR   ;DMA addr to rd/wr
 12(X)    .WORD DSTADR   ;DST address
 ;copy of disk specification table (DST)
 14(X)    .BYTE BLKSIZ   ;block size (3-7)
 15(X)    .WORD NMBLKS   ;£blocks on disk
 17(X)    .BYTE NMBDIR   ;£directory blocks
 18(X)    .BYTE SECSIZ   ;sector size (0-7)
 19(X)    .WORD SECTRK   ;sectors per track
 21(X)    .WORD TRKDSK   ;tracks on disk
 23(X)    .WORD RESTRK   ;reserved tracks
```

The operation to be performed by the driver is specified in the first byte of the PDR packet (OPCODE) as follows:

OPCODE	Function
0	Read sectors from disk
1	Write sectors to disk
2	Determine disk type, return DST
3	Determine if drive is ready
4	Format track on disk

Disk Driver (Continued)

If OPCODE=0, the driver reads SECCNT physical sectors (or equivalently, BYTCNT bytes) into DMAADR, starting at TRACK and SECTOR on DRIVE. The driver returns A=0 if the operation is successful, or A=-1 if an unrecoverable error occurs. TurboDOS may request multiple consecutive sectors to be read, but will never request an operation that extends past the end of the track.

If OPCODE=1, the driver writes SECCNT physical sectors (or BYTCNT bytes) from DMAADR, starting at TRACK and SECTOR on DRIVE. The driver returns A=0 if the operation is successful, or A=-1 if an unrecoverable error occurs. TurboDOS may request multiple consecutive sectors to be written, but will never request an operation that extends past the end of the track.

If OPCODE=2, the driver must determine the type of disk mounted in DRIVE, and must return, in the DSTADR field of the PDR packet, the address of an 11-byte disk specification table (DST) structured as follows:

Offset	Description
0	block size (3=1K,4=2K,...,7=16K)
1-2	total number of blocks on disk
3	number of directory blocks
4	sector size (0=128,...,7=16K)
5-6	number of sectors per track
7-8	number of tracks on the disk
9-10	number of reserved (boot) tracks

The first byte of the DST (BLKSIZ) specifies the allocation block size in bits 2-0. In addition, bit 7 is set if the disk is fixed (non-removable), and bit 6 is set if file extents are limited to 16K (EXM=0).

Disk Driver (Continued)

The driver returns A=-1 if the operation is successful, or A=0 if the drive is not ready or the disk type is unrecognizable. On successful return, TurboDOS moves a copy of the DST into 14(X) through 24(X), where it is available for subsequent operations.

If OPCODE=3, the driver determines whether DRIVE is ready, and returns A=-1 if it is ready or A=0 if not.

If OPCODE=4, the driver formats (initializes) TRACK on DRIVE, using hardware-dependent formatting information at DMAADR (put there by the FORMAT command). The driver returns A=0 if successful, or A=-1 if an unrecoverable error occurs.

Bank-Select Driver

Banked-memory systems must include a bank-select driver labelled with the public entry symbol SELBNK::. The function of this routine is simply to select the memory bank (0 or 1) passed in register A. The routine should be coded under the special location counter ?BANK? to ensure it is situated in unswitched common memory. In addition, the SELBNK:: routine must preserve all registers other than A.

All interrupt-driven drivers in a banked-memory system must be designed to service interrupts properly regardless of which bank is active when an interrupt occurs. Drivers for DMA disk controllers must ensure that DMA operations transfer into or out of bank 0 only. Study the sample drivers in the appendix for suggested techniques.

Network Driver

A network circuit driver should be labelled with the public entry symbol CKTDR@ (CKTDR%:: using PASM). A message buffer address is passed in register DE. The driver must either send or receive a network message, according to the operation code passed in register C:

C-reg	Function
0	Receive message into buffer at DE
1	Send message from buffer at DE

If C=0, the driver receives a network message into the message buffer whose address is passed in DE (waiting if necessary). If a message is received successfully, the driver returns A=0. If an unrecoverable malfunction of any remote processor is detected, the driver returns A=-1 with the network address of the crashed processor in DE.

If C=1, the driver sends a network message from the message buffer whose address is passed in DE. If the message is sent successfully, the driver returns A=0. If the message could not be sent because of an unrecoverable malfunction of the destination processor, the driver returns A=-1 with the network address of the crashed processor in DE.

The structure of a network message buffer is shown on the next page. The first two words of the buffer are reserved for a linkage used by TurboDOS, and should be ignored by the driver. The 11-byte message header and variable-length message body should be sent or received over the circuit. The driver needs to look at only the first two header fields (MSGLEN and MSGDID) and possibly the last field (MSGFCD).

Network Driver (Continued)

```
; message buffer format
        .WORD ?          ;linkage (ignored)
        .WORD ?          ;  "         "
; 11-byte message header
        .BYTE MSGLEN     ;msg length
        .WORD MSGDID     ;destination addr
        .BYTE MSGPID     ;process id
        .WORD MSGSID     ;source addr
        .WORD MSGOID     ;originator addr
        .BYTE MSGOPR     ;orig'r process id
        .BYTE MSGLVL     ;forwarding level
        .BYTE MSGFCD     ;msg format code
; variable-length body
        .BLKB 7          ;registers ACBEDLH
        .BLKB 1          ;user £ and flags
        .BLKB 37         ;optional FCB data
        .BLKB 128        ;optional record
```

The message format code field MSGFCD contains bit-encoded flags that define the format and context of each network message. This field may be ignored by most simple drivers, but its contents may be useful in complex network environments. Encoding of MSGFCD is:

Bit	Meaning
0	first message of session
1	last message of session
2	continuation message follows
3	request includes FCB data
4	request includes record data
5	reply includes FCB data
6	reply includes record data
7	this is a reply message

Network Driver (Continued)

The length field MSGLEN represents the number of bytes in the message, including the header and body (but excluding the linkage). On a receive request (C=0), TurboDOS presets MSGLEN to the maximum allowable message length, and expects MSGLEN to contain the actual message length on return. On a send request (C=1), TurboDOS presets MSGLEN to the actual length of the message to be sent.

In a master/slave network, it is often desirable for the circuit driver in the master to periodically "poll" the slave processors on the circuit to detect any slave malfunctions quickly and to effect recovery. If the driver reports that a slave has crashed (by returning A=-1 and DE=network-address), then the circuit driver must not accept any further messages from that slave until TurboDOS has completed its recovery process.

TurboDOS signals the driver that such recovery is complete by sending a dummy message destined for the slave in question with a length of zero. The driver should not actually send such a message to the slave, but could initiate whatever action is appropriate to reset the slave and download a new copy of the slave operating system.

A slave must request an operating system download by sending a special download request message to the master (usually done by a bootstrap routine). The download request message consists of a standard 11-byte header (with MSGPID, MSGOID and MSGFCD zeroed) followed by a 1-byte body containing a "download suffix" character. The master processor addressed by MSGDID will return a reply message whose 128-byte body is the first record of the download file OSSLAVEx.SYS (where "x" is the specified download suffix).

Network Driver (Continued)

The slave continues to send download request messages and to receive successive download records until it receives a short reply message (1-byte body) signifying end-of-file. The first word of the downloaded file specifies the base address to which the downloaded system should be moved, and the second word specifies the total byte-length of the system. The single byte passed as the body of the final short message identifies the system disk, and should be passed to the system in register A.

The entire failure detection, failure recovery, and slave downloading procedure is very hardware-dependent. Study the driver listing in the appendix for guidance.

Comm Driver

The comm driver supports the TurboDOS communications extensions (T-functions 34-40), and may be omitted if these functions are not used. The driver should be labelled with the public entry symbol COMDRV::. A comm channel number is passed in register B. The driver must perform an I/O operation according to the operation code passed in register E:

E-reg	Function
0	Return input status in A
1	Return input character in A
2	Output character passed in C
3	Set channel baud rate from C
4	Return channel baud rate in A
5	Set modem controls from C
6	Return modem status in A

If E=0, the driver determines if an input character is available. If one is available, the driver returns A=-1, otherwise A=0.

If E=1, the driver obtains an input character (waiting if necessary) and returns it in A.

If E=2, the driver outputs the character passed in C.

If E=3, the driver sets the channel baud rate according to the baud-rate code passed in C. If E=4, the driver returns the channel baud-rate code in A. See T-functions 37 and 38 in the _Z80 Programmer's Guide_ for baud-rate code definitions.

If E=5, the driver sets the modem controls according to the bit-vector passed in C. If E=6, the driver returns the modem status vector in A. See T-functions 39 and 40 in the _Z80 Programmer's Guide_ for bit-vector definitions.

Clock Driver

The real-time clock driver does not take the form of a subroutine called by TurboDOS, as do the other drivers described in this section. Rather, the clock driver generally consists of an interrupt service routine which responds to interrupts from a periodic interrupt source (preferably 50 to 60 times a second). The interrupt service routine should call DLYTIC£ once per system tick (to synchronize DELAY£ requests). It should also call RTCSEC£ once per second (that is, every 50 to 60 ticks) to update the system time and date. Finally, it should exit by jumping to ISRXIT£ to provide a periodic dispatcher time-slice. Excluding initialization code, a typical clock driver might be coded thus:

```
RTCCNT: .BYTE 60        ;divide-by-60 cntr
RTCISR: SSPD  INTSP£    ;save user's SP
        LXI   SP,INTSTK£ ;SP=aux stack
        PUSH  PSW       ;save registers
        PUSH  B         ; "     "
        PUSH  D         ; "     "
        PUSH  H         ; "     "
        IN    PORT      ;reset interrupt
        CALL  DLYTIC£   ;signal one tick
        LXI   H,RTCCNT  ;get div-by-60 cnt
        DCR   M         ;decrement counter
        JRNZ  ..X       ;not 60 ticks yet
        MVI   M,60      ;reset counter
        CALL  RTCSEC£   ;signal one second
..X:    POP   H         ;restore registers
        POP   D         ; "        "
        POP   B         ; "        "
        POP   PSW       ; "        "
        LSPD  INTSP£    ;restore user's SP
        JMP   ISRXIT£   ;go to dispatcher
```

Clock Driver (Continued)

If the hardware is capable of determining the date and time-of-day at cold-start (by means of a battery-powered clock, for example), the clock driver may initialize the following public symbols in the RTCMGR module:

```
SECS::  .BYTE 0        ;seconds 0-59
MINS::  .BYTE 0        ;minutes 0-59
HOURS:: .BYTE 0        ;hours   0-24
JDATE:: .WORD 8001H    ;Julian date
                       ;base 31-Dec-47
```

Bootstrap

The bootstrap is usually contained in a ROM or on a boot track. Its function is to search all disk drives for the TurboDOS loader program OSLOAD.COM, and to load and execute it if found. To generate a bootstrap, use the GEN command to combine the standard bootstrap module OSBOOT with your own hardware-dependent driver. Your driver must define the following public entry symbols: INIT, SELECT, READ, XFER, and RAM.

INIT:: is called once to perform any required hardware initialization. It returns with the load base address (where OSLOAD.COM will be loaded) in HL. This address should normally be 0100H, but may have to be higher for a bootstrap ROM in low-memory.

SELECT:: is called to select the disk drive passed in A (0-15). If the selected drive is not ready or non-existent, it returns A=0. Otherwise, it returns A=-1 and the address of an 11-byte disk specification table (DST) in HL. The DST format is described on page 5-7.

READ:: is called to read one physical sector from the last-selected drive. The track is passed in BC, the sector in DE, and the DMA address in HL. It must return A=0 if successful, or A=-1 if an unrecoverable error occurred.

XFER:: is transferred to at the end of the bootstrap process. In most cases, it needs only to set location 0080H to zero (to simulate a null command tail) and jump to 0100H. However, if INIT returned a loader base other than 0100H, then XFER must move the loader down to 0100H before executing it.

RAM:: defines a 64-byte area that OSBOOT can use for working storage. It should not be located where OSLOAD.COM will be loaded!

Sample Driver Source Listings

The remainder of this document consists of assembler source listings of actual drivers. The listings comprise the drivers for a networking TurboDOS system. The master processor is an S-100 single board computer, which incorporates 128K of banked memory, a floppy disk controller (supporting both 5" and 8" drivers), and a pair of RS232 serial ports on-board. The master also has a hard disk controller board connected to a pair of winchester drives. The slave processors are S-100 single-board computers with 128K of banked memory and a pair of RS232 serial ports.

The listings appear in the following order:

Module	Description
EQUATE	common symbolic equates
MPBMAS	master bootstrap driver
NITMAS	master driver initialization
INTMAS	master interrupt handler
BNKMAS	master bank-select driver
CON192	serial console driver, 19.2KB
LSTCTS	serial printer driver, CTS
LSTETX	serial printer driver, ETX/ACK
LSTXON	serial printer driver, XON/XOFF
SPDMAS	master serial/parallel driver
RTCMAS	master clock driver
DSKFDC	master floppy disk driver
DSTFDC	DSTs for 5" and 8" floppy disks
DSKHDC	Winchester hard disk driver
MCDMAS	master circuit driver
NITSLV	slave driver initialization
BNKSLV	slave bank-select driver
SCDSLV	slave circuit driver
RTCSLV	slave clock driver
SPDSLV	slave serial/parallel driver
SLVRES	general slave-reset subroutine

EQUATE - COMMON SYMBOLIC CONSTANTS
FOR SAMPLE Z80 TURBODOS DRIVERS

```
                        ;
                        .IDENT  EQUATE
                        ;
                        ; ASCII EQUIVALENCES
                        ;
0000                    ANUL     == 00H  ;NULL
0001                    ASOH     == 01H  ;SOH
0002                    ASTX     == 02H  ;STX
0003                    AETX     == 03H  ;ETX
0004                    AEOT     == 04H  ;EOT
0005                    AENQ     == 05H  ;ENQ
0006                    AACK     == 06H  ;ACK
0007                    ABEL     == 07H  ;BELL
0008                    ABS      == 08H  ;BS
0009                    AHT      == 09H  ;HT
000A                    ALF      == 0AH  ;LF
000B                    AVT      == 0BH  ;VT
000C                    AFF      == 0CH  ;FF
000D                    ACR      == 0DH  ;CR
000E                    ASO      == 0EH  ;SO
000F                    ASI      == 0FH  ;SI
0010                    ADLE     == 10H  ;DLE
0011                    ADC1     == 11H  ;DC1
0012                    ADC2     == 12H  ;DC2
0013                    ADC3     == 13H  ;DC3
0014                    ADC4     == 14H  ;DC4
0015                    ANAK     == 15H  ;NAK
0016                    ASYN     == 16H  ;SYN
0017                    AETB     == 17H  ;ETB
0018                    ACAN     == 18H  ;CAN
0019                    AEM      == 19H  ;EM
001A                    ASUB     == 1AH  ;SUB
001B                    AESC     == 1BH  ;ESC
001C                    AFS      == 1CH  ;FS
001D                    AGS      == 1DH  ;GS
001E                    ARS      == 1EH  ;RS
001F                    AUS      == 1FH  ;US
0020                    ASP      == 20H  ;SPACE
007F                    ARUB     == 7FH  ;RUBOUT (DEL)
                        ;
0000                    WBOOT    == 0000H        ;WARM START ENTRYPOINT
0003                    IOBYTE   == 0003H        ;I/O CONFIGURATION BYTE
0004                    CURDRV   == 0004H        ;CURRENT DEFAULT DRIVE
0005                    OPSYSC   == 0005H        ;OPERATING SYSTEM ENTRYPOINT (CP/M)
0050                    OPSYST   == 0050H        ;OPERATING SYSTEM ENTRYPOINT (TDOS)
005C                    TFCB     == 005CH        ;DEFAULT FILE CONTROL BLOCK
0080                    TBUF     == 0080H        ;DEFAULT DISK BUFFER ADDRESS
0100                    TPA      == 0100H        ;TRANSIENT PROGRAM AREA BASE
                        ;
0000                             .LOC    0       ;WORKING STORAGE RELATIVE TO 0
                        ;
0000                    PDRDP:                   ;PD REQUEST DESCRIPTOR PACKET
0000                    PDRFCN: .BLKB    1       ;PD REQUEST FUNCTION NUMBER
0001                    PDRDRV: .BLKB    1       ;PD REQUEST DRIVE NUMBER
```

```
   0002                   PDRTRK: .BLKW   1       ;PD REQUEST TRACK NUMBER
   0004                   PDRSEC: .BLKW   1       ;PD REQUEST SECTOR NUMBER
   0006                   PDRSC:  .BLKW   1       ;PD REQUEST SECTOR COUNT
   0008                   PDRTC:  .BLKW   1       ;PD REQUEST TRANSFER COUNT
   000A                   PDRDMA: .BLKW   1       ;PD REQUEST DMA ADDRESS
   000C                   PDRDST: .BLKW   1       ;PD REQUEST DRIVE SPEC TABLE ADDR
   000E                   PDRLEN  == .-PDRDP      ;PD REQUEST DESCRIPTOR PACKET LENGTH

   000E                   DSKNFO:                 ;DISK TYPE INFORMATION
   000E                   BLKSIZ: .BLKB   1       ;BLOCK SIZE
   000F                   NMBLKS: .BLKW   1       ;NUMBER OF BLOCKS
   0011                   NMBDIR: .BLKB   1       ;NUMBER OF DIRECTORY BLOCKS
   0012                   SECSIZ: .BLKB   1       ;PHYSICAL SECTOR SIZE (2^N*128)
   0013                   SECTRK: .BLKW   1       ;PHYSICAL SECTORS PER TRACK
   0015                   TRKDSK: .BLKW   1       ;PHYSICAL TRACKS PER DISK
   0017                   RESTRK: .BLKW   1       ;NUMBER OF RESERVED TRACKS
   000B                   DNFOL   == .-DSKNFO     ;DISK INFO LENGTH
                          ;
   0000'                          .RELOC
                          .PRGEND
```

```
                        ;
                        ; VERSION: 01/05/84
                        ;
                        .IDENT  MPBMAS          ;MODULE ID
                        ;
                        .INSERT EQUATE          ;SYMBOLIC EQUIVALENCES
                        ;
  0040                  RAM     =: 40H          ;WORKING STORAGE ADDRESS
  0040                  RAMLEN  = 64            ;WORKING STORAGE LENGTH
                        ;
  0000                  SIOADR  = 00H           ;SIO PORT A DATA REGISTER
  0001                  SIOACR  = 01H           ;SIO PORT A CONTROL REGISTER
  0002                  SIOBDR  = 02H           ;SIO PORT B DATA REGISTER
  0003                  SIOBCR  = 03H           ;SIO PORT B CONTROL REGISTER
                        ;
  0004                  PIOADR  = 04H           ;PIO PORT A DATA REGISTER
  0005                  PIOBDR  = 05H           ;PIO PORT B DATA REGISTER
  0006                  PIOACR  = 06H           ;PIO PORT A CONTROL REGISTER
  0007                  PIOBCR  = 07H           ;PIO PORT B CONTROL REGISTER
                        ;
  0008                  CTCCH0  = 08H           ;CTC CHANNEL 0 REGISTER
  0009                  CTCCH1  = 09H           ;CTC CHANNEL 1 REGISTER
  000A                  CTCCH2  = 0AH           ;CTC CHANNEL 2 REGISTER
  000B                  CTCCH3  = 0BH           ;CTC CHANNEL 3 REGISTER
                        ;
  000C                  FDCCSR  = 0CH           ;FDC COMMAND/STATUS REGISTER
  000D                  FDCTRK  = 0DH           ;FDC TRACK REGISTER
  000E                  FDCSEC  = 0EH           ;FDC SECTOR REGISTER
  000F                  FDCDAT  = 0FH           ;FDC DATA REGISTER
                        ;
  0010                  DMACTL  = 10H           ;DMA CONTROL REGISTER
                        ;
  0014                  FDCDSR  = 14H           ;FDC DRIVE SELECT REGISTER
  0015                  TSSOJR  = 15H           ;TWO-SIDED STATUS/OPTION JUMPER R
  0015                  EXTADR  = 15H           ;EXTENDED ADDRESS REGISTER
                        ;
  0016                  MEMCR1  = 16H           ;MEMORY CONTROL REGISTER #1
  0017                  MEMCR2  = 17H           ;MEMORY CONTROL REGISTER #2
                        ;
  0018                  SIOBRR  = 18H           ;SIO BAUD RATE GENERATOR REGISTER
                        ;
  0000                          .LOC    0       ;LOCATE IN BASE PAGE
                        ;
  0000                  INTPAG:                 ;IM2 INTERRUPT PAGE
  0000                          .BLKB   8       ;(RESERVED FOR TURBODOS)
  0008                  CTCVEC::.BLKW   4       ;CTC INTERRUPT VECTOR
  0010                  SIOVEC::.BLKW   8       ;SIO INTERRUPT VECTOR
  0020                  PIOVEC::.BLKW   2       ;PIO INTERRUPT VECTOR
                        ;
  0080                          .LOC    RAM+RAMLEN  ;LOCATE IN WORKING STORAGE AR
                        ;
  0080                  OPTION: .BLKB   1       ;OPTION JUMPER REGISTER SAVE AREA
                        ;
  0000'                         .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
```

```
                                ;
 0000'   F3             START:  DI                ;DISABLE INTERRUPTS
 0001'   3E4F                   MVI     A,4FH     ;DISABLE POWER ON JUMP
 0003'   D316                   OUT     MEMCR1
 0005'   AF                     XRA     A         ;DISABLE SECOND MEMORY BANK
 0006'   D317                   OUT     MEMCR2
 0008'   C3 0000:04             JMP     OSBOOT# ;CONTINUE
                                ;
 000B'   3E18           INIT::  MVI     A,18H
 000D'   D301                   OUT     SIOACR  ;RESET SIO PORT A
 000F'   D303                   OUT     SIOBCR  ;RESET SIO PORT B
 0011'   3E03                   MVI     A,03H
 0013'   D308                   OUT     CTCCH0  ;RESET CTC CHANNEL 0
 0015'   D309                   OUT     CTCCH1  ;RESET CTC CHANNEL 1
 0017'   D30A                   OUT     CTCCH2  ;RESET CTC CHANNEL 2
 0019'   D30B                   OUT     CTCCH3  ;RESET CTC CHANNEL 3
 001B'   3E07                   MVI     A,07H
 001D'   D306                   OUT     PIOACR  ;RESET PIO PORT A INTERRUPTS
 001F'   D307                   OUT     PIOBCR  ;RESET PIO PORT B INTERRUPTS
 0021'   3E00                   MVI     A,INTPAG>8  ;GET INTERRUPT PAGE
 0023'   ED47                   STAI              ;SET INTERRUPT PAGE REGISTER
 0025'   ED5E                   IM2               ;SET INTERRUPT MODE 2
 0027'   3E08                   MVI     A,CTCVEC&0FFH  ;GET CTC INTERRUPT VECTOR
 0029'   D308                   OUT     CTCCH0  ;INITIALIZE CTC INTERRUPT VECTOR
 002B'   DB15                   IN      TSSOJR  ;GET OPTION JUMPER REGISTER
 002D'   32 0080                STA     OPTION  ;SAVE OPTION JUMPER REGISTER
 0030'   CB77                   BIT     6,A     ;OPTION JUMPER BIT 6 CLOSED?
 0032'   C4 0000:05             CNZ     NITFDC# ;IF SO, INITIALIZE FLOPPY DISK
 0035'   3A 0080                LDA     OPTION  ;GET OPTION JUMPER REGISTER
 0038'   CB6F                   BIT     5,A     ;OPTION JUMPER BIT 5 CLOSED?
 003A'   C4 0000:06             CNZ     NITHDC# ;IF SO, INITIALIZE HARD DISK
 003D'   21 0100                LXI     H,TPA   ;GET LOAD ADDRESS
 0040'   FB                     EI              ;ENABLE INTERRUPTS
 0041'   C9                     RET             ;DONE
                                ;
 0042'   21 0080        SELECT::LXI     H,OPTION  ;GET OPTION JUMPER REGISTER
 0045'   FE04                   CPI     4       ;REQUESTED DRIVE 0-3?
 0047'   3806                   JRC     ..FDC   ;IF SO, CONTINUE
 0049'   FE08                   CPI     8       ;REQUESTED DRIVE 4-7?
 004B'   380B                   JRC     ..HDC   ;IF SO, CONTINUE
 004D'   AF             ..NRDY: XRA     A       ;ELSE, SET REURN CODE=0
 004E'   C9                     RET             ;DONE
 004F'   CB76           ..FDC:  BIT     6,M     ;OPTION JUMPER BIT 6 CLOSED?
 0051'   28FA                   JRZ     ..NRDY  ;IF NOT, CONTINUE
 0053'   CBBE                   RES     7,M     ;RESET HARD DISK SELECTED FLAG
 0055'   C3 0000:07             JMP     SELFDC# ;CONTINUE
 0058'   CB6E           ..HDC:  BIT     5,M     ;OPTION JUMPER BIT 5 CLOSED?
 005A'   28F1                   JRZ     ..NRDY  ;IF NOT, CONTINUE
 005C'   CBFE                   SET     7,M     ;SET HARD DISK SELECTED FLAG
 005E'   D604                   SUI     4       ;REMOVE FLOPPY DISK DRIVE BIAS
 0060'   C3 0000:08             JMP     SELHDC# ;CONTINUE
                                ;
 0063'   3A 0080        READ::  LDA     OPTION  ;GET OPTION JUMPER REGISTER
 0066'   CB7F                   BIT     7,A     ;HARD DISK SELECTED FLAG SET?
```

```
0068'   CA 0000:09              JZ      RDFDC#  ;IF NOT, CONTINUE
006B'   C3 0000:0A              JMP     RDHDC#  ;ELSE, CONTINUE
                        ;
006E'   F3              XFER::  DI              ;DISABLE INTERRUPTS
006F'   AF                      XRA     A
0070'   32 0080                 STA     TBUF    ;MAKE DEFAULT BUFFER EMPTY
0073'   21 0081'                LXI     H,XFRCOD  ;GET TRANSFER CODE
0076'   11 0000                 LXI     D,0     ;GET TRANSFER CODE DESTINATION
0079'   01 0007                 LXI     B,XFRCL ;GET TRANSFER CODE LENGTH
007C'   EDB0                    LDIR            ;MOVE TRANSFER CODE
007E'   C3 0000                 JMP     0       ;EXECUTE TRANSFER CODE
                        ;
0081'   3E6F            XFRCOD: MVI     A,6FH   ;DISABLE PROM/POWER ON RESET
0083'   D316                    OUT     MEMCR1
0085'   C3 0100                 JMP     TPA     ;TRANSFER TO O/S LOADER
                        ;
0007                    XFRCL   = .-XFRCOD      ;TRANSFER CODE LENGTH
                        ;
0000'                   .PRGEND START
```

```
                                ;
                                ; VERSION: 01/05/84
                                ;
                                .IDENT  MPBFDC          ;MODULE ID
                                ;
                                .INSERT EQUATE          ;SYMBOLIC EQUIVALENCES
                                ;
    0000                        SIOADR  = 00H           ;SIO PORT A DATA REGISTER
    0001                        SIOACR  = 01H           ;SIO PORT A CONTROL REGISTER
    0002                        SIOBDR  = 02H           ;SIO PORT B DATA REGISTER
    0003                        SIOBCR  = 03H           ;SIO PORT B CONTROL REGISTER
                                ;
    0004                        PIOADR  = 04H           ;PIO PORT A DATA REGISTER
    0005                        PIOBDR  = 05H           ;PIO PORT B DATA REGISTER
    0006                        PIOACR  = 06H           ;PIO PORT A CONTROL REGISTER
    0007                        PIOBCR  = 07H           ;PIO PORT B CONTROL REGISTER
                                ;
    0008                        CTCCH0  = 08H           ;CTC CHANNEL 0 REGISTER
    0009                        CTCCH1  = 09H           ;CTC CHANNEL 1 REGISTER
    000A                        CTCCH2  = 0AH           ;CTC CHANNEL 2 REGISTER
    000B                        CTCCH3  = 0BH           ;CTC CHANNEL 3 REGISTER
                                ;
    000C                        FDCCSR  = 0CH           ;FDC COMMAND/STATUS REGISTER
    000D                        FDCTRK  = 0DH           ;FDC TRACK REGISTER
    000E                        FDCSEC  = 0EH           ;FDC SECTOR REGISTER
    000F                        FDCDAT  = 0FH           ;FDC DATA REGISTER
                                ;
    0010                        DMACTL  = 10H           ;DMA CONTROL REGISTER
                                ;
    0014                        FDCDSR  = 14H           ;FDC DRIVE SELECT REGISTER
    0015                        TSSOJR  = 15H           ;TWO-SIDED STATUS/OPTION JUMPER REG
    0015                        EXTADR  = 15H           ;EXTENDED ADDRESS REGISTER
                                ;
    0016                        MEMCR1  = 16H           ;MEMORY CONTROL REGISTER #1
    0017                        MEMCR2  = 17H           ;MEMORY CONTROL REGISTER #2
                                ;
    0018                        SIOBRR  = 18H           ;SIO BAUD RATE GENERATOR REGISTER
                                ;
    0008                        FDCCAL  = 08H           ;FDC RE-CALIBRATE COMMAND
    0010                        FDCSKN  = 10H           ;FDC SEEK COMMAND WITHOUT HEAD LOAD
    0018                        FDCSKH  = 18H           ;FDC SEEK COMMAND WITH HEAD LOAD
    0082                        FDCRDC  = 82H           ;FDC READ SECTOR COMMAND
    00A2                        FDCWRC  = 0A2H          ;FDC WRITE SECTOR COMMAND
    00C0                        FDCRID  = 0C0H          ;FDC READ ID COMMAND
    00D0                        FDCINT  = 0D0H          ;FDC INTERRUPT COMMAND
    00F0                        FDCFMT  = 0F0H          ;FDC FORMAT TRACK COMMAND
                                ;
    0002                        HSDBIT  = 2             ;HEAD SETTLE DELAY BIT
                                ;
    0001                        DMARDC  = 01H           ;DMA READ COMMAND
    0005                        DMAWRC  = 05H           ;DMA WRITE COMMAND
                                ;
    0002                        TSD     = 2             ;TWO SIDED DISK BIT
    0003                        DDD     = 3             ;DOUBLE DENSITY DISK BIT
```

```
0004                    MINI    = 4             ;MINI-FLOPPY DISK BIT
0005                    TPI96   = 5             ;96-TPI DISK BIT
                        ;
000A                    MAXTRY  = 10            ;MAX DISK TRY COUNT
                        ;
0081                            .LOC    81H     ;LOCATE IN WORKING STORAGE AREA
                        ;
0081                    DRIVE:  .BLKB   1       ;DRIVE NUMBER
0082                    TRACK:  .BLKB   1       ;TRACK NUMBER
0083                    SECTOR: .BLKB   1       ;SECTOR NUMBER
0084                    TRYCNT: .BLKB   1       ;TRY COUNTER
0085                    DLYBIT: .BLKB   1       ;HEAD SETTLE DELAY BIT
0086                    INTCST: .BLKB   1       ;INTERRUPT COMPLETION STATUS
0087                    DSRSAV: .BLKB   1       ;DRIVE SELECT REGISTER SAVE
0088                    NDXCNT: .BLKB   1       ;INDEX PULSE SEQUENCE COUNT
0089                    NDXTIC: .BLKB   1       ;INDEX PULSE TICK COUNT
008A                    TICCNT: .BLKB   1       ;TICK COUNT
008B                    DWTFLG: .BLKB   1       ;DISK WAIT FLAG
008C                    RETSP:  .BLKW   1       ;ERROR RETURN STACK POINTER
008E                    RIDBUF: .BLKB   6       ;READ ID BUFFER
                        ;
0094                    DSKNFO:                 ;DISK TYPE INFORMATION
0094                    BLKSIZ: .BLKB   1       ;BLOCK SIZE
0095                    NMBLKS: .BLKW   1       ;NUMBER OF BLOCKS
0097                    NMBDIR: .BLKB   1       ;NUMBER OF DIRECTORY BLOCKS
0098                    SECSIZ: .BLKB   1       ;PHYSICAL SECTOR SIZE (2^N*128)
0099                    SECTRK: .BLKW   1       ;PHYSICAL SECTORS PER TRACK
009B                    TRKDSK: .BLKW   1       ;PHYSICAL TRACKS PER DISK
009D                    RESTRK: .BLKW   1       ;NUMBER OF RESERVED TRACKS
009F                    XLTBL:  .BLKW   1       ;TRANSLATION TABLE ADDRESS
00A1                    TYPCOD: .BLKB   1       ;DISK TYPE CODE
000E                    DNFOL   = .-DSKNFO      ;DISK INFO LENGTH
                        ;
00A2                    DMAPGM:                 ;DMA CONTROLLER PROGRAM LIST
00A2                            .BLKB   1       ;WRITE REGISTER 6
00A3                            .BLKB   1       ;WRITE REGISTER 6
00A4                            .BLKB   1       ;WRITE REGISTER 0
00A5                    DMAADR: .BLKW   1       ;DMA ADDRESS
00A7                    DMALEN: .BLKW   1       ;DMA LENGTH
00A9                            .BLKB   1       ;WRITE REGISTER 1
00AA                            .BLKB   1       ;WRITE REGISTER 2
00AB                            .BLKB   1       ;WRITE REGISTER 4
00AC                            .BLKB   1       ;FDC DATA PORT ADDRESS
00AD                            .BLKB   1       ;WRITE REGISTER 5
00AE                            .BLKB   1       ;WRITE REGISTER 6
00AF                    DMARWC: .BLKB   1       ;DMA READ/WRITE COMMAND
00B0                            .BLKB   1       ;WRITE REGISTER 6
00B1                            .BLKB   1       ;WRITE REGISTER 6
                        ;
0010                    DMAPLL  = .-DMAPGM      ;DMA CONTROLLER PROGRAM LIST LENGT
                        ;
0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
0000'   00001030        DRVTBL::.BYTE   0,0,1<MINI,1<MINI!1<TPI96  ;DRIVE TABLE
```

```
                                ;
 0004'    21 0295'        NITFDC::LXI      H,RTCISR  ;GET INTERRUPT SERVICE ADDRESS
 0007'    22 0002:04              SHLD     CTCVEC#+2  ;SET INTERRUPT SERVICE VECTOR
 000A'    3E47                    MVI      A,47H     ;GET CTC CHANNEL 0 CONTROL WORD
 000C'    D308                    OUT      CTCCH0    ;INITIALIZE CTC CHANNEL 0
 000E'    3EFA                    MVI      A,250     ;GET TIME CONSTANT VALUE
 0010'    D308                    OUT      CTCCH0    ;SET CTC CHANNEL 0 TIME CONSTANT
 0012'    3EC7                    MVI      A,0C7H    ;GET CTC CHANNEL 1 CONTROL WORD
 0014'    D309                    OUT      CTCCH1    ;INITIALIZE CTC CHANNEL 1
 0016'    3E64                    MVI      A,100     ;GET TIME CONSTANT VALUE
 0018'    D309                    OUT      CTCCH1    ;SET CTC CHANNEL 1 TIME CONSTANT
 001A'    AF                      XRA      A
 001B'    32 0088                 STA      NDXCNT    ;SET INDEX PULSE SEQUENCE COUNT=0
 001E'    CD 022E'                CALL     CLRFDC    ;CLEAR FDC
 0021'    21 023D'                LXI      H,DSKISR  ;GET INTERRUPT SERVICE ROUTINE
 0024'    22 0006:04              SHLD     CTCVEC#+6  ;SET INTERRUPT VECTOR ADDRESS
 0027'    21 02A1'                LXI      H,DCPLST  ;GET DMA CONTROLLER PROGRAM LIST
 002A'    11 00A2                 LXI      D,DMAPGM  ;GET DMA CONTROLLER PROGRAM AREA
 002D'    01 0010                 LXI      B,DMAPLL  ;GET DMA PROGRAM LIST LENGTH
 0030'    EDB0                    LDIR               ;MOVE DMA PROGRAM LIST
 0032'    C9                      RET                ;DONE
                                ;
 0033'    FE04            SELFDC::CPI      4         ;VALID DRIVE NUMBER?
 0035'    D2 00D6'                JNC      ..NRDY    ;IF NOT, CONTINUE
 0038'    32 0081                 STA      DRIVE     ;ELSE, SET DRIVE NUMBER
 003B'    CD 01B3'                CALL     SELDSK    ;SELECT DISK
 003E'    C2 00D6'                JNZ      ..NRDY    ;IF DRIVE NOT READY, CONTINUE
 0041'    3A 0087                 LDA      DSRSAV    ;GET SAVED DRIVE SELECT REGISTER
 0044'    CB67                    BIT      4,A       ;MINI-FLOPPY DISK BIT SET?
 0046'    2828                    JRZ      ..NMFD    ;IF NOT, CONTINUE
 0048'    3E02                    MVI      A,2       ;GET INDEX PULSE SEQUENCE COUNT
 004A'    32 0088                 STA      NDXCNT    ;SET INDEX PULSE SEQUENCE COUNT=2
 004D'    CD 027B'                CALL     ENACTC    ;ENABLE CTC INTERRUPT CONTROLLER
 0050'    3ED4                    MVI      A,FDCINT!1<2  ;GET FDC INTERRUPT COMMAND
 0052'    D30C                    OUT      FDCCSR    ;OUTPUT FDC INTERRUPT COMMAND
 0054'    3A 008A                 LDA      TICCNT    ;GET TICK COUNT
 0057'    4F                      MOV      C,A       ;TICK COUNT TO C-REG
 0058'    3A 008A         ..DLYL: LDA      TICCNT    ;GET TICK COUNT
 005B'    91                      SUB      C         ;CALC ELAPSED TICK COUNT
 005C'    FE3C                    CPI      60        ;ONE SECOND DELAY COMPLETE?
 005E'    38F8                    JRC      ..DLYL    ;IF NOT, CONTINUE
 0060'    3E03                    MVI      A,03H     ;ELSE, GET CTC RESET COMMAND
 0062'    D30B                    OUT      CTCCH3    ;RESET CTC CHANNEL 3
 0064'    CD 022E'                CALL     CLRFDC    ;CLEAR FDC
 0067'    21 0088                 LXI      H,NDXCNT  ;SET INDEX PULSE SEQUENCE COUNT
 006A'    7E                      MOV      A,M       ;GET INDEX PULSE SEQUENCE COUNT
 006B'    3600                    MVI      M,0       ;SET INDEX PULSE SEQUENCE COUNT=0
 006D'    B7                      ORA      A         ;INDEX PULSE SEQUENCE COUNT=0?
 006E'    2066                    JRNZ     ..NRDY    ;IF NOT, CONTINUE
 0070'    AF              ..NMFD: XRA      A
 0071'    D30D                    OUT      FDCTRK    ;SET FDC TRACK REGISTER
 0073'    D30F                    OUT      FDCDAT    ;SET FDC DATA REGISTER
 0075'    3E10                    MVI      A,FDCSKN  ;GET FDC SEEK COMMAND
 0077'    CD 0218'                CALL     FDCCMD    ;OUTPUT FDC SEEK COMMAND
```

```
007A'   CD 019F'              CALL    RECAL   ;RE-CALIBRATE DRIVE
007D'   CD 01CD'              CALL    READID  ;READ SECTOR ID
0080'   2054                  JRNZ    ..NRDY  ;IF READ UNSUCCESSFUL, CONTINUE
0082'   3A 0091               LDA     RIDBUF+3  ;ELSE, GET SECTOR SIZE
0085'   4F                    MOV     C,A     ;SECTOR SIZE TO C-REG
0086'   3A 0087               LDA     DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
0089'   CB5F                  BIT     3,A     ;DOUBLE DENSITY BIT SET?
008B'   2802                  JRZ     ..NDDD  ;IF NOT, CONTINUE
008D'   CBD9                  SET     DDD,C   ;ELSE, SET DOUBLE DENSITY DISK BIT
008F'   CBD7          ..NDDD: SET     2,A     ;SET SIDE ONE SELECT BIT
0091'   D314                  OUT     FDCDSR  ;SELECT REQUESTED DRIVE
0093'   32 0087               STA     DSRSAV  ;SAVE DRIVE SELECT REGISTER VALUE
0096'   CD 022E'              CALL    CLRFDC  ;CLEAR FDC
0099'   E680                  ANI     80H     ;DRIVE READY?
009B'   201E                  JRNZ    ..NTSD  ;IF NOT, CONTINUE
009D'   CD 028A'              CALL    GETDTA  ;ELSE, GET DRIVE TABLE ADDRESS
00A0'   CB66                  BIT     MINI,M  ;MINI-FLOPPY DISK?
00A2'   2006                  JRNZ    ..MFD   ;IF SO, CONTINUE
00A4'   DB15                  IN      TSSOJR  ;ELSE, GET TWO SIDED DISK STATUS
00A6'   E680                  ANI     80H     ;TWO SIDED DISK?
00A8'   2811                  JRZ     ..NTSD  ;IF NOT, CONTINUE
00AA'   C5            ..MFD:  PUSH    B       ;ELSE, SAVE DISK TYPE CODE
00AB'   CD 01CD'              CALL    READID  ;READ SECTOR ID
00AE'   C1                    POP     B       ;RESTORE DISK TYPE CODE
00AF'   200A                  JRNZ    ..NTSD  ;IF READ UNSUCCESSFUL, CONTINUE
00B1'   3A 0087               LDA     DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
00B4'   A9                    XRA     C       ;COMPARE SIDE ONE/TWO DENSITIES
00B5'   E608                  ANI     1<DDD
00B7'   2002                  JRNZ    ..NTSD  ;IF DENSITIES DIFFERENT, CONTINUE
00B9'   CBD1                  SET     TSD,C   ;ELSE, SET TWO SIDED DISK BIT
00BB'   CD 028A'      ..NTSD: CALL    GETDTA  ;GET DRIVE TABLE ADDRESS
00BE'   7E                    MOV     A,M     ;GET DRIVE TABLE VALUE
00BF'   E630                  ANI     1<MINI!1<TPI96  ;EXTRACT RELEVANT BITS
00C1'   B1                    ORA     C       ;COMBINE WITH DISK TYPE CODE
00C2'   4F                    MOV     C,A     ;DISK TYPE CODE TO C-REG
00C3'   11 0000:05            LXI     D,DSTBLS#  ;GET DST TABLE BASE
00C6'   21 0000:06    ..DSTL: LXI     H,DTCO# ;GET OFFSET TO DISK TYPE CODE
00C9'   19                    DAD     D       ;CALC DISK TYPE CODE ADDRESS
00CA'   79                    MOV     A,C     ;GET DISK TYPE CODE
00CB'   BE                    CMP     M       ;DST TYPE CODE MATCH?
00CC'   EB                    XCHG            ;DST ADDRESS TO HL-REG
00CD'   2809                  JRZ     ..DSTF  ;IF DST FOUND, CONTINUE
00CF'   5E                    MOV     E,M     ;ELSE, GET NEXT DST ADDRESS
00D0'   23                    INX     H
00D1'   56                    MOV     D,M
00D2'   7A                    MOV     A,D
00D3'   B3                    ORA     E       ;END OF DST CHAIN?
00D4'   20F0                  JRNZ    ..DSTL  ;IF NOT, CONTINUE
00D6'   AF            ..NRDY: XRA     A       ;ELSE, SET RETURN CODE=0
00D7'   C9                    RET             ;DONE
00D8'   23            ..DSTF: INX     H       ;ADVANCE PAST LINK POINTER
00D9'   23                    INX     H
00DA'   E5                    PUSH    H       ;SAVE DST ADDRESS
00DB'   11 0094               LXI     D,DSKNFO  ;GET DISK INFO WORK AREA
```

```
 00DE'   01 000E              LXI     B,DNFOL ;GET DISK INFO LENGTH
 00E1'   EDB0                 LDIR            ;COPY DST INTO WORK AREA
 00E3'   E1                   POP     H       ;RESTORE DST ADDRESS
 00E4'   3EFF                 MVI     A,0FFH  ;SET RETURN CODE=0FFH
 00E6'   C9                   RET             ;DONE
                      ;
 00E7'   ED73 008C    RDFDC:: SSPD    RETSP   ;SAVE ERROR RETURN STACK POINTER
 00EB'   79                   MOV     A,C     ;GET REQUESTED TRACK NUMBER
 00EC'   32 0082              STA     TRACK   ;SET TRACK NUMBER
 00EF'   7B                   MOV     A,E     ;GET REQUESTED SECTOR NUMBER
 00F0'   32 0083              STA     SECTOR  ;SET SECTOR NUMBER
 00F3'   22 00A5              SHLD    DMAADR  ;SET DMA ADDRESS
 00F6'   3E0A                 MVI     A,MAXTRY  ;GET MAX TRY COUNT
 00F8'   32 0084              STA     TRYCNT  ;SET TRY COUNTER
 00FB'   CD 0114'     ..RD:   CALL    SETUP   ;DO COMMON SETUP
 00FE'   200F                 JRNZ    ..ERR   ;IF SEEK ERROR, CONTINUE
 0100'   CD 012D'             CALL    RWCOM1  ;ELSE, DO READ/WRITE COMMON #1
 0103'   11 829D              LXI     D,FDCRDC<8!9DH  ;GET FDC READ COMMAND/MASK
 0106'   CD 0162'             CALL    RWCOM2  ;DO READ/WRITE COMMON #2
 0109'   3E01                 MVI     A,DMARDC  ;GET DMA READ COMMAND
 010B'   CD 01F9'             CALL    DMACOM  ;DO DMA COMMON
 010E'   C8                   RZ              ;IF NO ERRORS, DONE
 010F'   CD 016B'     ..ERR:  CALL    RETRY   ;ELSE, RE-CALIBRATE DRIVE
 0112'   18E7                 JMPR    ..RD    ;TRY AGAIN
                      ;
 0114'   3A 0098      SETUP:  LDA     SECSIZ  ;GET SECTOR SIZE
 0117'   47                   MOV     B,A     ;SECTOR SIZE TO B-REG
 0118'   04                   INR     B       ;INCREMENT SECTOR SIZE
 0119'   21 0040              LXI     H,128/2 ;GET SECTOR SIZE=0 (/2)
 011C'   29           ..SL:   DAD     H       ;SHIFT SECTOR SIZE LEFT
 011D'   10FD                 DJNZ    ..SL    ;SECTOR SIZE TIMES
 011F'   2B                   DCX     H       ;DECREMENT SECTOR SIZE
 0120'   22 00A7              SHLD    DMALEN  ;SET DMA LENGTH
 0123'   CD 01B3'             CALL    SELDSK  ;SELECT DISK
 0126'   C2 0178'             JNZ     FATAL   ;IF DRIVE NOT READY, CONTINUE
 0129'   CD 017F'             CALL    SEEK    ;ELSE, SEEK TO REQUESTED TRACK
 012C'   C9                   RET             ;DONE
                      ;
 012D'   3A 00A1      RWCOM1: LDA     TYPCOD  ;GET DISK TYPE CODE
 0130'   21 0087              LXI     H,DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
 0133'   CB5F                 BIT     DDD,A   ;DOUBLE DENSITY DISK?
 0135'   2002                 JRNZ    ..DDD   ;IF SO, CONTINUE
 0137'   CB9E                 RES     3,M     ;ELSE, RESET DOUBLE DENSITY BIT
 0139'   CB57         ..DDD:  BIT     TSD,A   ;TWO SIDED DISK?
 013B'   3A 0083              LDA     SECTOR  ;GET SECTOR NUMBER
 013E'   280E                 JRZ     ..NTSD  ;IF NOT TWO SIDED DISK, CONTINUE
 0140'   E5                   PUSH    H       ;ELSE, SAVE DRIVE SELECT REGISTER
 0141'   21 0099              LXI     H,SECTRK  ;GET SECTORS PER TRACK ADDRESS
 0144'   4E                   MOV     C,M     ;GET NUMBER OF SECTORS/TRACK
 0145'   E1                   POP     H       ;RESTORE DRIVE SELECT REGISTER
 0146'   CB39                 SRLR    C       ;CALC NUMBER OF SECTORS/SIDE
 0148'   B9                   CMP     C       ;REQUESTED SECTOR ON SIDE ONE?
 0149'   3803                 JRC     ..NTSD  ;IF NOT, CONTINUE
 014B'   91                   SUB     C       ;ELSE, ADJUST SECTOR NUMBER
 014C'   CBD6                   SET     2,M      ;SET SIDE ONE SELECT BIT
 014E'   4F            ..NTSD:  MOV     C,A      ;SECTOR NUMBER TO C-REG
 014F'   CD 0284'               CALL    GETXLT   ;GET TRANSLATION TABLE ADDRESS
 0152'   2804                   JRZ     ..NSTR   ;IF NO SECTOR TRANSLATION, CONTINUE
 0154'   0600                   MVI     B,0      ;MAKE SECTOR NUMBER DOUBLE LENGTH
 0156'   09                     DAD     B        ;INDEX INTO TRANSLATION TABLE
 0157'   4E                     MOV     C,M      ;GET TRANSLATED SECTOR NUMBER
 0158'   79            ..NSTR:  MOV     A,C      ;GET SECTOR NUMBER
 0159'   3C                     INR     A        ;INCREMENT SECTOR NUMBER TO BASE 1
 015A'   D30E                   OUT     FDCSEC   ;SET FDC SECTOR REGISTER
 015C'   3A 0087                LDA     DSRSAV   ;GET SAVED DRIVE SELECT REGISTER
 015F'   D314                   OUT     FDCDSR   ;SELECT DRIVE/SIDE/DENSITY
 0161'   C9                     RET              ;DONE
                       ;
 0162'   3A 0087       RWCOM2:  LDA     DSRSAV   ;GET SAVED DRIVE SELECT REGISTER
 0165'   CB57                   BIT     2,A      ;SIDE ONE SELECTED?
 0167'   C8                     RZ               ;IF NOT, DONE
 0168'   CBDA                   SET     3,D      ;ELSE, SET SIDE ONE VERIFY BIT
 016A'   C9                     RET              ;DONE
                       ;
 016B'   3A 0084       RETRY:   LDA     TRYCNT   ;GET TRY COUNTER
 016E'   E601                   ANI     01H      ;EVEN TRY?
 0170'   C4 019F'               CNZ     RECAL    ;IF NOT, RE-CALIBRATE DRIVE
 0173'   21 0084                LXI     H,TRYCNT  ;ELSE, GET TRY COUNTER
 0176'   35                     DCR     M        ;DECREMENT TRY COUNTER
 0177'   C0                     RNZ              ;IF COUNT NOT EXHAUSTED, DONE
                       ;
 0178'   ED7B 008C     FATAL:   LSPD    RETSP    ;RESTORE STACK POINTER
 017C'   3EFF                   MVI     A,0FFH   ;RETURN ERROR CODE
 017E'   C9                     RET              ;DONE
                       ;
 017F'   21 0082       SEEK:    LXI     H,TRACK  ;GET REQUESTED TRACK NUMBER
 0182'   DB0D                   IN      FDCTRK   ;GET FDC TRACK REGISTER
 0184'   BE                     CMP     M        ;FDC TRACK=REQUESTED TRACK?
 0185'   C8                     RZ               ;IF SO, DONE
 0186'   3E04                   MVI     A,1<HSDBIT  ;GET HEAD SETTLE DELAY BIT
 0188'   32 0085                STA     DLYBIT   ;SET HEAD SETTLE DELAY BIT
 018B'   CD 022E'               CALL    CLRFDC   ;CLEAR FDC
 018E'   7E                     MOV     A,M      ;GET REQUESTED TRACK NUMBER
 018F'   D30F                   OUT     FDCDAT   ;OUTPUT REQUESTED TRACK NUMBER
 0191'   CD 028A'               CALL    GETDTA   ;GET DRIVE TABLE ADDRESS
 0194'   7E                     MOV     A,M      ;GET DRIVE TABLE VALUE
 0195'   E603                   ANI     3        ;EXTRACT STEP RATE
 0197'   F618                   ORI     FDCSKH   ;COMBINE WITH FDC SEEK COMMAND
 0199'   CD 0218'               CALL    FDCCMD   ;OUTPUT FDC SEEK COMMAND
 019C'   E691                   ANI     91H      ;EXTRACT RELEVANT STATUS BITS
 019E'   C9                     RET              ;DONE
                       ;
 019F'   3E04          RECAL:   MVI     A,1<HSDBIT  ;GET HEAD SETTLE DELAY BIT
 01A1'   32 0085                STA     DLYBIT   ;SET HEAD SETTLE DELAY BIT
 01A4'   CD 022E'               CALL    CLRFDC   ;CLEAR FDC
 01A7'   CD 028A'               CALL    GETDTA   ;GET DRIVE TABLE ADDRESS
 01AA'   7E                     MOV     A,M      ;GET DRIVE TABLE VALUE
 01AB'   E603                   ANI     3        ;EXTRACT STEP RATE
 01AD'   F608                 ORI    FDCCAL  ;COMBINE WITH RE-CALIBRATE COMMAND
 01AF'   CD 0218'             CALL   FDCCMD  ;OUTPUT FDC RE-CALIBRATE COMMAND
 01B2'   C9                   RET            ;DONE
                      ;
 01B3'   CD 028A'     SELDSK: CALL   GETDTA  ;GET DRIVE TABLE ADDRESS
 01B6'   CBDF                 SET    3,A     ;SET DOUBLE DENSITY BIT
 01B8'   CB66                 BIT    MINI,M  ;MINI-FLOPPY DISK?
 01BA'   2802                 JRZ    ..NMFD  ;IF NOT, CONTINUE
 01BC'   CBE7                 SET    4,A     ;ELSE, SET MINI-FLOPPY DISK BIT
 01BE'   D314         ..NMFD: OUT    FDCDSR  ;SELECT REQUESTED DRIVE
 01C0'   32 0087              STA    DSRSAV  ;SAVE DRIVE SELECT REGISTER VALUE
 01C3'   CD 022E'             CALL   CLRFDC  ;CLEAR FDC
 01C6'   E680                 ANI    80H     ;DRIVE READY?
 01C8'   C0                   RNZ            ;IF NOT, DONE
 01C9'   32 0085              STA    DLYBIT  ;ELSE, SET HEAD SETTLE DELAY BIT=0
 01CC'   C9                   RET            ;DONE
                      ;
 01CD'   21 008E      READID: LXI    H,RIDBUF  ;GET READ ID BUFFER
 01D0'   22 00A5              SHLD   DMAADR  ;SET DMA ADDRESS
 01D3'   21 0005              LXI    H,6-1   ;GET SECTOR ID LENGTH (-1)
 01D6'   22 00A7              SHLD   DMALEN  ;SET DMA LENGTH
 01D9'   21 0087              LXI    H,DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
 01DC'   CBDE                 SET    3,M     ;SET DOUBLE DENSITY BIT
 01DE'   7E                   MOV    A,M     ;GET SAVED DRIVE SELECT REGISTER
 01DF'   D314                 OUT    FDCDSR  ;SELECT DRIVE/SIDE/DENSITY
 01E1'   11 C09D      ..RIDL: LXI    D,FDCRID<8!9DH  ;GET READ ID COMMAND/MASK
 01E4'   3E01                 MVI    A,DMARDC  ;GET DMA READ COMMAND
 01E6'   CD 01F9'             CALL   DMACOM  ;READ ID
 01E9'   C8                   RZ             ;IF READ OK, DONE
 01EA'   3A 0087              LDA    DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
 01ED'   EE08                 XRI    1<3     ;TOGGLE SINGLE/DOUBLE DENSITY BIT
 01EF'   D314                 OUT    FDCDSR  ;OUTPUT DRIVE SELECT REGISTER VALUE
 01F1'   32 0087              STA    DSRSAV  ;SAVE DRIVE SELECT REGISTER VALUE
 01F4'   E608                 ANI    1<3     ;DOUBLE DENSITY SELECTED?
 01F6'   28E9                 JRZ    ..RIDL  ;IF NOT, CONTINUE
 01F8'   C9                   RET            ;ELSE, DONE
                      ;
 01F9'   32 00AF      DMACOM: STA    DMARWC  ;SET DMA READ/WRITE COMMAND
 01FC'   CD 022E'             CALL   CLRFDC  ;CLEAR FDC
 01FF'   21 00A2              LXI    H,DMAPGM  ;GET DMA PROGRAM LIST
 0202'   01 1010              LXI    B,DMAPLL<8!DMACTL  ;B=PROGRAM LENGTH/C=PORT
 0205'   EDB3                 OUTIR          ;PROGRAM DMA CONTROLLER
 0207'   3A 0085              LDA    DLYBIT  ;GET HEAD SETTLE DELAY BIT
 020A'   B2                   ORA    D       ;COMBINE WITH FDC COMMAND
 020B'   D5                   PUSH   D       ;SAVE ERROR MASK
 020C'   CD 0218'             CALL   FDCCMD  ;OUTPUT FDC COMMAND
 020F'   D1                   POP    D       ;RESTORE ERROR MASK
 0210'   A3                   ANA    E       ;EXTRACT RELEVANT STATUS BITS
 0211'   F5                   PUSH   PSW     ;SAVE ERROR STATUS
 0212'   3EC3                 MVI    A,0C3H  ;GET DMA RESET COMMAND
 0214'   D310                 OUT    DMACTL  ;DISABLE DMA CONTROLLER
 0216'   F1                   POP    PSW     ;RESTORE ERROR STATUS
 0217'   C9                   RET            ;DONE
                      ;
 0218'  F5              FDCCMD: PUSH    PSW     ;SAVE FDC COMMAND
 0219'  AF                      XRA     A
 021A'  32 008B                 STA     DWTFLG  ;SET DISK WAIT FLAG=0
 021D'  CD 027B'                CALL    ENACTC  ;ENABLE CTC INTERRUPT CONTROLLER
 0220'  F1                      POP     PSW     ;RESTORE FDC COMMAND
 0221'  D30C                    OUT     FDCCSR  ;OUTPUT FDC COMMAND
 0223'  76              ..WTL:  HLT             ;WAIT FOR INTERRUPT
 0224'  3A 008B                 LDA     DWTFLG  ;GET DISK WAIT FLAG
 0227'  B7                      ORA     A       ;DISK WAIT FLAG=0?
 0228'  28F9                    JRZ     ..WTL   ;IF SO, CONTINUE
 022A'  3A 0086                 LDA     INTCST  ;GET INTERRUPT COMPLETION STATUS
 022D'  C9                      RET             ;DONE
                        ;
 022E'  3ED0            CLRFDC: MVI     A,FDCINT  ;GET FDC INTERRUPT COMMAND
 0230'  D30C                    OUT     FDCCSR  ;OUTPUT FDC INTERRUPT COMMAND
 0232'  E3                      XTHL            ;DELAY
 0233'  E3                      XTHL
 0234'  E3                      XTHL
 0235'  E3                      XTHL
 0236'  E3                      XTHL
 0237'  E3                      XTHL
 0238'  DB0F                    IN      FDCDAT  ;CLEAR DRQ
 023A'  DB0C                    IN      FDCCSR  ;CLEAR INTRQ
 023C'  C9                      RET             ;DONE
                        ;
 023D'  F5              DSKISR: PUSH    PSW     ;SAVE REGISTERS
 023E'  C5                      PUSH    B
 023F'  DB0C                    IN      FDCCSR  ;GET FDC COMPLETION STATUS
 0241'  32 0086                 STA     INTCST  ;SAVE INTERRUPT COMPLETION STATUS
 0244'  3A 0088                 LDA     NDXCNT  ;GET INDEX PULSE SEQUENCE COUNT
 0247'  3D                      DCR     A       ;INDEX PULSE SEQUENCE COUNT=0?
 0248'  FA 026D'                JM      ..ISC0  ;IF SO, CONTINUE
 024B'  2015                    JRNZ    ..FIP   ;IF FIRST INDEX PULSE, CONTINUE
 024D'  3A 0089                 LDA     NDXTIC  ;ELSE, GET INDEX PULSE TICK COUNT
 0250'  4F                      MOV     C,A     ;INDEX PULSE TICK COUNT TO C-REG
 0251'  3A 008A                 LDA     TICCNT  ;GET TICK COUNT
 0254'  32 0089                 STA     NDXTIC  ;UPDATE INDEX PULSE TICK COUNT
 0257'  91                      SUB     C       ;CALC ELAPSED TICK COUNTS
 0258'  FE0E                    CPI     14      ;INDEX PULSE TIMING WITHIN LIMITS?
 025A'  301A                    JRNC    ..ISRX  ;IF NOT, CONTINUE
 025C'  AF                      XRA     A
 025D'  32 0088                 STA     NDXCNT  ;SET INDEX PULSE SEQUENCE COUNT=0
 0260'  1810                    JMPR    ..ISCX  ;CONTINUE
 0262'  32 0088         ..FIP:  STA     NDXCNT  ;SET INDEX PULSE SEQUENCE COUNT=1
 0265'  3A 008A                 LDA     TICCNT  ;GET TICK COUNT
 0268'  32 0089                 STA     NDXTIC  ;SAVE INDEX PULSE TICK COUNT
 026B'  1809                    JMPR    ..ISRX  ;CONTINUE
 026D'  3EFF            ..ISC0: MVI     A,0FFH
 026F'  32 008B                 STA     DWTFLG  ;SET DISK WAIT FLAG=0FFH
 0272'  3E03            ..ISCX: MVI     A,03H   ;GET CTC RESET COMMAND
 0274'  D30B                    OUT     CTCCH3  ;RESET CTC CHANNEL 3
 0276'  C1              ..ISRX: POP     B       ;RESTORE REGISTERS
 0277'  F1                      POP     PSW
 0278'  FB                      EI              ;ENABLE INTERRUPTS
```

```
0279'   ED4D                 RETI              ;DONE
                     ;
027B'   3ED7         ENACTC: MVI     A,0D7H    ;GET CTC CHANNEL 3 CONTROL WORD
027D'   D30B                 OUT     CTCCH3    ;INITIALIZE CTC CHANNEL 3
027F'   3E01                 MVI     A,1       ;GET CTC CHANNEL 3 TIME CONSTANT
0281'   D30B                 OUT     CTCCH3    ;SET CTC CHANNEL 3 TIME CONSTANT
0283'   C9                   RET               ;DONE
                     ;
0284'   2A 009F      GETXLT: LHLD    XLTBL     ;GET TRANSLATION TABLE ADDRESS
0287'   7C                   MOV     A,H
0288'   B5                   ORA     L         ;TRANSLATION TABLE REQUIRED?
0289'   C9                   RET               ;DONE
                     ;
028A'   3A 0081      GETDTA: LDA     DRIVE     ;GET DRIVE NUMBER
028D'   5F                   MOV     E,A       ;DRIVE NUMBER TO DE-REG
028E'   1600                 MVI     D,0       ;DOUBLE LENGTH
0290'   21 0000'             LXI     H,DRVTBL  ;GET DRIVE TABLE
0293'   19                   DAD     D         ;INDEX INTO DRIVE TABLE
0294'   C9                   RET               ;DONE
                     ;
0295'   F5           RTCISR: PUSH    PSW       ;SAVE REGISTERS
0296'   3A 008A              LDA     TICCNT    ;GET TICK COUNT
0299'   3C                   INR     A         ;INCREMENT TICK COUNT
029A'   32 008A              STA     TICCNT    ;UPDATE TICK COUNT
029D'   F1                   POP     PSW
029E'   FB                   EI                ;ENABLE INTERRUPTS
029F'   ED4D                 RETI              ;DONE
                     ;
02A1'                DCPLST:                   ;DMA CONTROLLER PROGRAM LIST
02A1'   C3                   .BYTE   0C3H      ;WRITE REGISTER 6
02A2'   8B                   .BYTE   08BH      ;WRITE REGISTER 6
02A3'   79                   .BYTE   79H       ;WRITE REGISTER 0
02A4'   0000                 .WORD   0         ;DMA ADDRESS
02A6'   0000                 .WORD   0         ;DMA LENGTH
02A8'   14                   .BYTE   14H       ;WRITE REGISTER 1
02A9'   28                   .BYTE   28H       ;WRITE REGISTER 2
02AA'   85                   .BYTE   85H       ;WRITE REGISTER 4
02AB'   0F                   .BYTE   FDCDAT    ;FDC DATA PORT ADDRESS
02AC'   8A                   .BYTE   8AH       ;WRITE REGISTER 5
02AD'   CF                   .BYTE   0CFH      ;WRITE REGISTER 6
02AE'   05                   .BYTE   05H       ;DMA READ/WRITE COMMAND
02AF'   CF                   .BYTE   0CFH      ;WRITE REGISTER 6
02B0'   87                   .BYTE   87H       ;WRITE REGISTER 6
                     ;
                     .PRGEND
```

```
                      ;
                      ; VERSION: 01/05/84
                      ;
                      .IDENT  MPBHDC          ;MODULE ID
                      ;
                      .INSERT EQUATE          ;SYMBOLIC EQUIVALENCES
                      ;
 0000                 SIOADR  = 00H           ;SIO PORT A DATA REGISTER
 0001                 SIOACR  = 01H           ;SIO PORT A CONTROL REGISTER
 0002                 SIOBDR  = 02H           ;SIO PORT B DATA REGISTER
 0003                 SIOBCR  = 03H           ;SIO PORT B CONTROL REGISTER
                      ;
 0004                 PIOADR  = 04H           ;PIO PORT A DATA REGISTER
 0005                 PIOBDR  = 05H           ;PIO PORT B DATA REGISTER
 0006                 PIOACR  = 06H           ;PIO PORT A CONTROL REGISTER
 0007                 PIOBCR  = 07H           ;PIO PORT B CONTROL REGISTER
                      ;
 0008                 CTCCH0  = 08H           ;CTC CHANNEL 0 REGISTER
 0009                 CTCCH1  = 09H           ;CTC CHANNEL 1 REGISTER
 000A                 CTCCH2  = 0AH           ;CTC CHANNEL 2 REGISTER
 000B                 CTCCH3  = 0BH           ;CTC CHANNEL 3 REGISTER
                      ;
 000C                 FDCCSR  = 0CH           ;FDC COMMAND/STATUS REGISTER
 000D                 FDCTRK  = 0DH           ;FDC TRACK REGISTER
 000E                 FDCSEC  = 0EH           ;FDC SECTOR REGISTER
 000F                 FDCDAT  = 0FH           ;FDC DATA REGISTER
                      ;
 0010                 DMACTL  = 10H           ;DMA CONTROL REGISTER
                      ;
 0014                 FDCDSR  = 14H           ;FDC DRIVE SELECT REGISTER
 0015                 TSSOJR  = 15H           ;TWO-SIDED STATUS/OPTION JUMPER REG
 0015                 EXTADR  = 15H           ;EXTENDED ADDRESS REGISTER
                      ;
 0016                 MEMCR1  = 16H           ;MEMORY CONTROL REGISTER #1
 0017                 MEMCR2  = 17H           ;MEMORY CONTROL REGISTER #2
                      ;
 0018                 SIOBRR  = 18H           ;SIO BAUD RATE GENERATOR REGISTER
                      ;
 0090                 IOBASE  = 90H           ;I/O BASE ADDRESS
                      ;
 0090                 HDCDAT  = IOBASE+0      ;HDC DATA REGISTER
 0091                 HDCEWP  = IOBASE+1      ;HDC ERROR/WRITE PRECOMP REGISTER
 0092                 HDCSCT  = IOBASE+2      ;HDC SECTOR COUNT REGISTER
 0093                 HDCSEC  = IOBASE+3      ;HDC SECTOR NUMBER REGISTER
 0094                 HDCCYL  = IOBASE+4      ;HDC CYLINDER REGISTER (LOW)
 0095                 HDCCYH  = IOBASE+5      ;HDC CYLINDER REGISTER (HIGH)
 0096                 HDCSDH  = IOBASE+6      ;HDC SIZE/DRIVE/HEAD REGISTER
 0097                 HDCCSR  = IOBASE+7      ;HDC COMMAND/STATUS REGISTER
                      ;
 0010                 HDCCAL  = 10H           ;HDC CALIBRATE DRIVE COMMAND
 0020                 HDCRDS  = 20H           ;HDC READ SECTOR COMMAND
 0030                 HDCWRS  = 30H           ;HDC WRITE SECTOR COMMAND
 0050                 HDCFMT  = 50H           ;HDC FORMAT TRACK COMMAND
 0070                 HDCSEK  = 70H           ;HDC SEEK COMMAND
```

```
                            ;
 0001                       ERRMSK  = 01H           ;COMPLETION ERROR MASK
 0040                       RDYMSK  = 40H           ;DRIVE READY MASK
                            ;
 0006                       STEPRT  = 6             ;STEP RATE (3-MS)
                            ;
 00B2                               .LOC    0B2H    ;LOCATE IN WORKING STORAGE AREA
                            ;
 00B2                       DRIVE:  .BLKB   1       ;DRIVE NUMBER
 00B3                       TRACK:  .BLKW   1       ;TRACK NUMBER
 00B5                       SECTOR: .BLKB   1       ;SECTOR NUMBER
 00B6                       DMAADR: .BLKW   1       ;DMA ADDRESS
 00B8                       INTCST: .BLKB   1       ;INTERRUPT COMPLETION STATUS
 00B9                       DWTFLG: .BLKB   1       ;DISK WAIT FLAG
 00BA                       RETSP:  .BLKW   1       ;ERROR RETURN STACK POINTER
 00BC                       INTMSK: .BLKB   1       ;INTERRUPT MASK
 00BD                       ISRTBL: .BLKW   8       ;INTERRUPT SERVICE ROUTINE TABLE
                            ;
 0000'                              .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                            ;
 0000'   CD 00CE'           NITHDC::CALL    ICINIT  ;INITIALIZE INTERRUPT CONTROLLER
 0003'   21 00C3'                   LXI     H,DSKISR  ;GET INTERRUPT SERVICE ADDRESS
 0006'   3E02                       MVI     A,2     ;GET VECTORED INTERRUPT NUMBER
 0008'   CD 00EE'                   CALL    INTNIT  ;INITIALIZE INTERRUPT VECTOR
 000B'   C9                         RET             ;DONE
                            ;
 000C'   CD 008B'           SELHDC::CALL    RETRDY  ;RETURN READY STATUS
 000F'   B7                         ORA     A       ;DRIVE READY?
 0010'   C8                         RZ              ;IF NOT, DONE
 0011'   CD 0071'                   CALL    RECAL   ;ELSE, RE-CALIBRATE DRIVE
 0014'   2F                         CMA             ;COMPLIMENT RETURN CODE
 0015'   C0                         RNZ             ;IF ERRORS, DONE
 0016'   21 0143'                   LXI     H,HDCDST  ;ELSE, GET DST ADDRESS
 0019'   C9                         RET             ;DONE
                            ;
 001A'   ED73 00BA          RDHDC:: SSPD    RETSP   ;SAVE ERROR RETURN STACK POINTER
 001E'   ED43 00B3                  SBCD    TRACK   ;SET TRACK NUMBER
 0022'   7B                         MOV     A,E     ;GET REQUESTED SECTOR NUMBER
 0023'   32 00B5                    STA     SECTOR  ;SET SECTOR NUMBER
 0026'   22 00B6                    SHLD    DMAADR  ;SET DMA ADDRESS
 0029'   CD 0040'                   CALL    SETUP   ;DO COMMON SETUP
 002C'   3E20                       MVI     A,HDCRDS  ;GET READ SECTOR COMMAND
 002E'   CD 00AE'                   CALL    WTINT   ;WAIT FOR INTERRUPT
 0031'   01 0090                    LXI     B,HDCDAT  ;GET DATA PORT ADDRESS
 0034'   2A 00B6                    LHLD    DMAADR  ;GET DMA ADDRESS
 0037'   EDB2                       INIR            ;INPUT 256 BYTES OF DATA
 0039'   EDB2                       INIR            ;INPUT 256 BYTES OF DATA
 003B'   B7                         ORA     A       ;ANY ERRORS?
 003C'   C8                         RZ              ;IF NOT, DONE
 003D'   3EFF                       MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
 003F'   C9                         RET             ;DONE
                            ;
 0040'   CD 009A'           SETUP:  CALL    SELDSK  ;SELECT REQUESTED DRIVE
 0043'   CA 0084'                   JZ      FATAL   ;IF DRIVE NOT READY, CONTINUE
```

```
0046'    3A 00B5                 LDA    SECTOR  ;ELSE, GET SECTOR NUMBER
0049'    D393                    OUT    HDCSEC  ;OUTPUT SECTOR NUMBER
004B'    2A 00B3                 LHLD   TRACK   ;GET REQUESTED TRACK NUMBER
004E'    7D                      MOV    A,L     ;GET LSB OF TRACK NUMBER
004F'    F5                      PUSH   PSW     ;SAVE LSB OF TRACK NUMBER
0050'    CB3C                    SRLR   H       ;ELIMINATE HEAD NUMBER
0052'    CB1D                    RARR   L
0054'    CB3C                    SRLR   H
0056'    CB1D                    RARR   L
0058'    7D                      MOV    A,L     ;GET LSB OF TRACK NUMBER
0059'    D394                    OUT    HDCCYL  ;OUTPUT LSB OF TRACK NUMBER
005B'    7C                      MOV    A,H     ;GET MSB OF TRACK NUMBER
005C'    D395                    OUT    HDCCYH  ;OUTPUT MSB OF TRACK NUMBER
005E'    F1                      POP    PSW     ;RESTORE LSB OF TRACK NUMBER
005F'    E603                    ANI    3       ;EXTRACT HEAD NUMBER
0061'    21 00B2                 LXI    H,DRIVE ;GET DRIVE NUMBER
0064'    CB46                    BIT    0,M     ;SECOND LOGICAL VOLUME?
0066'    2802                    JRZ    ..NSLV  ;IF NOT, CONTINUE
0068'    CBD7                    SET    2,A     ;ELSE, SET BIT 2 OF HEAD NUMBER
006A'    4F              ..NSLV: MOV    C,A     ;HEAD NUMBER TO C-REG
006B'    DB96                    IN     HDCSDH  ;GET SIZE/DRIVE/HEAD REGISTER
006D'    B1                      ORA    C       ;SET HEAD NUMBER FIELD
006E'    D396                    OUT    HDCSDH  ;OUTPUT SIZE/DRIVE/HEAD
0070'    C9                      RET            ;DONE
                         ;
0071'    AF              RECAL:  XRA    A       ;GET TRACK 0
0072'    D391                    OUT    HDCEWP  ;SET WRITE PRECOMP TRACK REGISTER
0074'    3E16                    MVI    A,HDCCAL!STEPRT  ;GET CALIBRATE COMMAND
0076'    CD 00AE'                CALL   WTINT   ;WAIT FOR INTERRUPT
0079'    2006                    JRNZ   ..ERR   ;IF ERRORS, CONTINUE
007B'    3E10                    MVI    A,HDCCAL  ;GET CALIBRATE DRIVE COMMAND
007D'    CD 00AE'                CALL   WTINT   ;WAIT FOR INTERRUPT
0080'    C8                      RZ             ;IF NO ERRORS, DONE
0081'    3EFF            ..ERR:  MVI    A,0FFH  ;ELSE, SET RETURN CODE=0FFH
0083'    C9                      RET            ;DONE
                         ;
0084'    ED7B 00BA       FATAL:  LSPD   RETSP   ;RESTORE STACK POINTER
0088'    3EFF                    MVI    A,0FFH  ;RETURN ERROR CODE
008A'    C9                      RET            ;DONE
                         ;
008B'    32 00B2         RETRDY: STA    DRIVE   ;SAVE DRIVE NUMBER
008E'    FE04                    CPI    4       ;TEST FOR VALID DRIVE NUMBER
0090'    3E00                    MVI    A,0     ;PRESET RETURN CODE=0
0092'    D0                      RNC            ;IF INVALID DRIVE, RETURN NOT READY
0093'    CD 009A'                CALL   SELDSK  ;ELSE, SELECT REQUESTED DRIVE
0096'    C8                      RZ             ;IF DRIVE NOT READY, DONE
0097'    3EFF                    MVI    A,0FFH  ;ELSE, SET RETURN CODE=0FFH
0099'    C9                      RET            ;DONE
                         ;
009A'    DB97            SELDSK: IN     HDCCSR  ;GET STATUS REGISTER
009C'    3C                      INR    A       ;CONTROLLER PRESENT?
009D'    C8                      RZ             ;IF NOT, DONE
009E'    3A 00B2                 LDA    DRIVE   ;ELSE, GET REQUESTED DRIVE
00A1'    E602                    ANI    2       ;EXTRACT PHYSICAL DRIVE NUMBER
```

```
00A3'   87                      ADD     A       ;SHIFT DRIVE NUMBER LEFT
00A4'   87                      ADD     A
00A5'   F6A0                    ORI     0A0H    ;SET ERROR CORRECTION/SECTOR SIZE
00A7'   D396                    OUT     HDCSDH  ;OUTPUT SIZE/DRIVE/HEAD
00A9'   DB97                    IN      HDCCSR  ;GET STATUS REGISTER
00AB'   E640                    ANI     RDYMSK  ;DRIVE READY?
00AD'   C9                      RET             ;DONE
                        ;
00AE'   F5              WTINT:  PUSH    PSW     ;SAVE COMMAND
00AF'   AF                      XRA     A
00B0'   32 00B9                 STA     DWTFLG  ;SET DISK WAIT FLAG=0
00B3'   F1                      POP     PSW     ;RESTORE COMMAND
00B4'   D397                    OUT     HDCCSR  ;OUTPUT COMMAND
00B6'   76              ..WTL:  HLT             ;WAIT FOR INTERRUPT
00B7'   3A 00B9                 LDA     DWTFLG  ;GET DISK WAIT FLAG
00BA'   B7                      ORA     A       ;DISK WAIT FLAG=0?
00BB'   28F9                    JRZ     ..WTL   ;IF SO, CONTINUE
00BD'   3A 00B8                 LDA     INTCST  ;GET INTERRUPT COMPLETION STATUS
00C0'   E601                    ANI     ERRMSK  ;ANY ERRORS?
00C2'   C9                      RET             ;DONE
                        ;
00C3'   DB97            DSKISR: IN      HDCCSR  ;GET INTERRUPT COMPLETION STATUS
00C5'   32 00B8                 STA     INTCST  ;SAVE INTERRUPT COMPLETION STATUS
00C8'   3EFF                    MVI     A,0FFH
00CA'   32 00B9                 STA     DWTFLG  ;SET DISK WAIT FLAG=0FFH
00CD'   C9                      RET             ;DONE
                        ;
00CE'   21 0002:04      ICINIT: LXI     H,PIOVEC#+2  ;GET INTERRUPT VECTOR ADDRESS
00D1'   7D                      MOV     A,L     ;GET LSB OF INTERRUPT VECTOR
00D2'   D307                    OUT     PIOBCR  ;SET PIO INTERRUPT VECTOR ADDRESS
00D4'   21 0111'                LXI     H,PIOISR  ;GET INTERRUPT SERVICE ADDRESS
00D7'   22 0002:04              SHLD    PIOVEC#+2  ;SET INTERRUPT SERVICE VECTOR
00DA'   3ECF                    MVI     A,0CFH  ;GET MODE 3 CONTROL WORD
00DC'   D307                    OUT     PIOBCR  ;SET PIO PORT B TO MODE 3
00DE'   3EFF                    MVI     A,0FFH  ;GET I/O DIRECTION CONTROL WORD
00E0'   D307                    OUT     PIOBCR  ;SET PIO PORT B DIRECTION TO INPUT
00E2'   3E97                    MVI     A,97H   ;GET PIO INTERRUPT CONTROL WORD
00E4'   D307                    OUT     PIOBCR  ;ENABLE PIO INTERRUPTS
00E6'   3EFF                    MVI     A,0FFH  ;GET INTERRUPT MASK
00E8'   D307                    OUT     PIOBCR  ;MASK ALL INTERRUPTS
00EA'   32 00BC                 STA     INTMSK  ;SAVE INTERRUPT MASK
00ED'   C9                      RET             ;DONE
                        ;
00EE'   F5              INTNIT: PUSH    PSW     ;SAVE VECTORED INTERRUPT NUMBER
00EF'   87                      ADD     A       ;CALC VECTORED INTERRUPT NUMBER * 2
00F0'   4F                      MOV     C,A     ;INTERRUPT NUMBER TO BC-REG
00F1'   0600                    MVI     B,0     ;DOUBLE LENGTH
00F3'   EB                      XCHG            ;INTERRUPT SERVICE ADDR TO DE-REG
00F4'   21 00BD                 LXI     H,ISRTBL  ;GET ISR ADDRESS TABLE
00F7'   09                      DAD     B       ;INDEX INTO ISR TABLE
00F8'   73                      MOV     M,E     ;STORE ISR ADDRESS IN ISR TABLE
00F9'   23                      INX     H
00FA'   72                      MOV     M,D
00FB'   F1                      POP     PSW     ;RESTORE VECTORED INTERRUPT NUMBER
```

```
  00FC'   3C                  INR     A         ;INCREMENT INTERRUPT NUMBER
  00FD'   47                  MOV     B,A       ;INTERRUPT NUMBER TO B-REG
  00FE'   AF                  XRA     A         ;INITIALIZE RESULT VECTOR
  00FF'   37                  STC               ;SET CARRY FLAG
  0100'   8F          ..SL:   ADC     A         ;SHIFT CARRY FLAG LEFT
  0101'   10FD                DJNZ    ..SL      ;INTERRUPT NUMBER + 1 TIMES
  0103'   2F                  CMA               ;COMPLIMENT RESULT VECTOR
  0104'   21 00BC             LXI     H,INTMSK  ;GET INTERRUPT MASK
  0107'   A6                  ANA     M         ;RESET INTERRUPT MASK BIT
  0108'   77                  MOV     M,A       ;UPDATE INTERRUPT MASK
  0109'   3E97                MVI     A,97H     ;GET PIO INTERRUPT CONTROL WORD
  010B'   D307                OUT     PIOBCR    ;ENABLE PIO INTERRUPTS
  010D'   7E                  MOV     A,M       ;GET INTERRUPT MASK
  010E'   D307                OUT     PIOBCR    ;SET PIO INTERRUPT MASK REGISTER
  0110'   C9                  RET               ;DONE
                      ;
  0111'   F5          PIOISR: PUSH    PSW       ;SAVE REGISTERS
  0112'   C5                  PUSH    B
  0113'   D5                  PUSH    D
  0114'   E5                  PUSH    H
  0115'   DB05        ..ISRL: IN      PIOBDR    ;GET VECTORED INTERRUPT STATUS
  0117'   FEFF                CPI     0FFH      ;ANY VECTORED INTERRUPTS PENDING?
  0119'   2821                JRZ     ..ISRX    ;IF NOT, CONTINUE
  011B'   0608                MVI     B,8       ;ELSE, GET MAX NUMBER OF INTERRUPTS
  011D'   1F          ..SVCL: RAR               ;VECTORED INTERRUPT PENDING?
  011E'   3818                JRC     ..SVCC    ;IF NOT, CONTINUE
  0120'   C5                  PUSH    B         ;ELSE, SAVE INTERRUPT COUNTER
  0121'   F5                  PUSH    PSW       ;SAVE INTERRUPT STATUS
  0122'   3E08                MVI     A,8       ;GET MAX NUMBER OF INTERRUPTS
  0124'   90                  SUB     B         ;CALC CURRENT INTERRUPT NUMBER
  0125'   87                  ADD     A         ;CALC CURRENT INTERRUPT NUMBER * 2
  0126'   4F                  MOV     C,A       ;INTERRUPT NUMBER TO BC-REG
  0127'   0600                MVI     B,0       ;DOUBLE LENGTH
  0129'   21 00BD             LXI     H,ISRTBL  ;GET ISR ADDRESS TABLE
  012C'   09                  DAD     B         ;INDEX INTO ISR TABLE
  012D'   5E                  MOV     E,M       ;GET INTERRUPT SERVICE ADDRESS
  012E'   23                  INX     H
  012F'   56                  MOV     D,M
  0130'   21 0136'            LXI     H,..RET   ;GET RETURN ADDRESS
  0133'   E5                  PUSH    H         ;PUSH RETURN ADDRESS ONTO STACK
  0134'   EB                  XCHG              ;INTERRUPT SERVICE ADDR TO HL-REG
  0135'   E9                  PCHL              ;TRANSFER TO INT SERVICE ROUTINE
  0136'   F1          ..RET:  POP     PSW       ;RESTORE INTERRUPT STATUS
  0137'   C1                  POP     B         ;RESTORE INTERRUPT COUNTER
  0138'   10E3        ..SVCC: DJNZ    ..SVCL    ;CONTINUE
  013A'   18D9                JMPR    ..ISRL    ;CONTINUE
  013C'   E1          ..ISRX: POP     H         ;RESTORE REGISTERS
  013D'   D1                  POP     D
  013E'   C1                  POP     B
  013F'   F1                  POP     PSW
  0140'   FB                  EI                ;ENABLE INTERRUPTS
  0141'   ED4D                RETI              ;DONE
                      ;
  0143'   84          HDCDST: .BYTE   84H       ;ALLOCATION BLOCK SIZE
```

```
   0144'   1540                     .WORD   5440     ;NUMBER OF ALLOCATION BLOCKS
   0146'   28                       .BYTE   40       ;NUMBER OF DIRECTORY BLOCKS
   0147'   02                       .BYTE   2        ;PHYSICAL SECTOR SIZE (2^N*128)
   0148'   0011                     .WORD   17       ;PHYSICAL SECTORS PER TRACK
   014A'   0500                     .WORD   1280     ;PHYSICAL TRACKS PER DISK
   014C'   0000                     .WORD   0        ;NUMBER OF RESERVED TRACKS
                            ;
                            .PRGEND
```

```
NITMAS - SAMPLE DRIVER INITIALIZATION MODULE
FOR TURBODOS ON S-100 MASTER Z80 PROCESSOR


                        ;
                        ; VERSION: 01/05/84
                        ;
                        .IDENT  NITMAS          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
 0000                   SIOADR  = 00H           ;SIO PORT A DATA REGISTER
 0001                   SIOACR  = 01H           ;SIO PORT A CONTROL REGISTER
 0002                   SIOBDR  = 02H           ;SIO PORT B DATA REGISTER
 0003                   SIOBCR  = 03H           ;SIO PORT B CONTROL REGISTER
                        ;
 0004                   PIOADR  = 04H           ;PIO PORT A DATA REGISTER
 0005                   PIOBDR  = 05H           ;PIO PORT B DATA REGISTER
 0006                   PIOACR  = 06H           ;PIO PORT A CONTROL REGISTER
 0007                   PIOBCR  = 07H           ;PIO PORT B CONTROL REGISTER
                        ;
 0008                   CTCCH0  = 08H           ;CTC CHANNEL 0 REGISTER
 0009                   CTCCH1  = 09H           ;CTC CHANNEL 1 REGISTER
 000A                   CTCCH2  = 0AH           ;CTC CHANNEL 2 REGISTER
 000B                   CTCCH3  = 0BH           ;CTC CHANNEL 3 REGISTER
                        ;
 000C                   FDCCSR  = 0CH           ;FDC COMMAND/STATUS REGISTER
 000D                   FDCTRK  = 0DH           ;FDC TRACK REGISTER
 000E                   FDCSEC  = 0EH           ;FDC SECTOR REGISTER
 000F                   FDCDAT  = 0FH           ;FDC DATA REGISTER
                        ;
 0010                   DMACTL  = 10H           ;DMA CONTROL REGISTER
                        ;
 0014                   FDCDSR  = 14H           ;FDC DRIVE SELECT REGISTER
 0015                   TSSOJR  = 15H           ;TWO-SIDED STATUS/OPTION JUMPER R
 0015                   EXTADR  = 15H           ;EXTENDED ADDRESS REGISTER
                        ;
 0016                   MEMCR1  = 16H           ;MEMORY CONTROL REGISTER #1
 0017                   MEMCR2  = 17H           ;MEMORY CONTROL REGISTER #2
                        ;
 0018                   SIOBRR  = 18H           ;SIO BAUD RATE GENERATOR REGISTER
                        ;
 0000                           .LOC    0       ;LOCATE IN BASE PAGE
                        ;
 0000                   INTPAG:                 ;IM2 INTERRUPT PAGE
 0000                           .BLKB   8       ;(RESERVED FOR TURBODOS)
 0008                   CTCVEC::.BLKW   4       ;CTC INTERRUPT VECTOR
 0010                   SIOVEC::.BLKW   8       ;SIO INTERRUPT VECTOR
 0020                   PIOVEC::.BLKW   2       ;PIO INTERRUPT VECTOR
                        ;
 0000:04                        .LOC    .INIT.# ;LOCATE IN INIT-CODE AREA
                        ;
 0000:04 3EEF           HDWNIT::MVI     A,0EFH  ;DISABLE PROM/POWER ON JUMP
 0002:04 D316                   OUT     MEMCR1
 0004:04 AF                     XRA     A       ;DISABLE SECOND MEMORY BANK
 0005:04 D317                   OUT     MEMCR2
 0007:04 32 0003                STA     IOBYTE  ;SET I/O BYTE=0
 000A:04 21 0000                LXI     H,0     ;INITIALIZE MEMORY PARITY
 000D:04 11 0000             LXI   D,0
 0010:04 01 0000             LXI   B,0
 0013:04 EDB0                LDIR
 0015:04 3E6F                MVI   A,6FH   ;RESET PARITY ERROR LATCH
 0017:04 D316                OUT   MEMCR1
 0019:04 3EEF                MVI   A,0EFH  ;ENABLE PARITY ERROR DETECTION
 001B:04 D316                OUT   MEMCP1
 001D:04 3E18                MVI   A,18H
 001F:04 D301                OUT   SIOACR  ;RESET SIO PORT A
 0021:04 D303                OUT   SIOBCR  ;RESET SIO PORT B
 0023:04 3E03                MVI   A,03H
 0025:04 D308                OUT   CTCCH0  ;RESET CTC CHANNEL 0
 0027:04 D309                OUT   CTCCH1  ;RESET CTC CHANNEL 1
 0029:04 D30A                OUT   CTCCH2  ;RESET CTC CHANNEL 2
 002B:04 D30B                OUT   CTCCH3  ;RESET CTC CHANNEL 3
 002D:04 3E07                MVI   A,07H
 002F:04 D306                OUT   PIOACR  ;RESET PIO PORT A INTERRUPTS
 0031:04 D307                OUT   PIOBCR  ;RESET PIO PORT B INTERRUPTS
 0033:04 3E00                MVI   A,INTPAG>8  ;GET INTERRUPT PAGE
 0035:04 ED47                STAI          ;SET INTERRUPT PAGE REGISTER
 0037:04 ED5E                IM2           ;SET INTERRUPT MODE 2
 0039:04 3E08                MVI   A,CTCVEC&0FFH  ;GET CTC INTERRUPT VECTOR
 003B:04 D308                OUT   CTCCH0  ;INITIALIZE CTC INTERRUPT VECTOR
 003D:04 CD 0000:05          CALL  ICINIT# ;INITIALIZE INTERRUPT HANDLER
 0040:04 CD 0000:06          CALL  BNKNIT# ;INITIALIZE BANKED TPA DRIVER
 0043:04 CD 0000:07          CALL  SPINIT# ;INITIALIZE SERIAL/PARALLEL I/O
 0046:04 CD 0000:08          CALL  RTCNIT# ;INITIALIZE REAL TIME CLOCK
 0049:04 CD 0000:09          CALL  DSKINA# ;INITIALIZE DISK DRIVER A
 004C:04 CD 0000:0A          CALL  DSKINB# ;INITIALIZE DISK DRIVER B
 004F:04 CD 0000:0B          CALL  DSKINC# ;INITIALIZE DISK DRIVER C
 0052:04 CD 0000:0C          CALL  DSKIND# ;INITIALIZE DISK DRIVER D
 0055:04 CD 0000:0D          CALL  CKTINA# ;INITIALIZE CIRCUIT DRIVER A
 0058:04 CD 0000:0E          CALL  CKTINB# ;INITIALIZE CIRCUIT DRIVER B
 005B:04 CD 0000:0F          CALL  CKTINC# ;INITIALIZE CIRCUIT DRIVER C
 005E:04 C3 0000:10          JMP   CKTIND# ;INITIALIZE CIRCUIT DRIVER D
                      ;
                      .PRGEND
```

```
                              ;
                              ; VERSION: 01/05/84
                              ;
                              .IDENT  INTMAS          ;MODULE ID
                              ;
                              .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                              ;
0005                          PIOBDR  = 05H           ;PIO PORT B DATA REGISTER
0007                          PIOBCR  = 07H           ;PIO PORT B CONTROL REGISTER
                              ;
0000"                                 .LOC    .DATA.# ;LOCATE IN DATA AREA
                              ;
0000"                         INTMSK: .BLKB   1       ;INTERRUPT MASK
0001"                         ISRTBL: .BLKW   8       ;INTERRUPT SERVICE ROUTINE TABLE
0011"                         INTSP:  .BLKW   1       ;INTERRUPT STACK POINTER SAVE AREA
0013"                                 .BLKW   16      ;INTERRUPT STACK AREA
0033"                         INTSTK  = .             ;TOP OF INTERRUPT STACK AREA
                              ;
0000:04                               .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                              ;
0000:04 21 0002:05            ICINIT::LXI     H,PIOVEC#+2  ;GET INTERRUPT VECTOR ADDRESS
0003:04 7D                            MOV     A,L     ;GET LSB OF INTERRUPT VECTOR
0004:04 D307                          OUT     PIOBCR  ;SET PIO INTERRUPT VECTOR ADDRESS
0006:04 21 0000'                      LXI     H,PIOISR  ;GET INTERRUPT SERVICE ADDRESS
0009:04 22 0002:05                    SHLD    PIOVEC#+2  ;SET INTERRUPT SERVICE VECTOR
000C:04 3ECF                          MVI     A,0CFH  ;GET MODE 3 CONTROL WORD
000E:04 D307                          OUT     PIOBCR  ;SET PIO PORT B TO MODE 3
0010:04 3EFF                          MVI     A,0FFH  ;GET I/O DIRECTION CONTROL WORD
0012:04 D307                          OUT     PIOBCR  ;SET PIO PORT B DIRECTION TO INPUT
0014:04 3E97                          MVI     A,97H   ;GET PIO INTERRUPT CONTROL WORD
0016:04 D307                          OUT     PIOBCR  ;ENABLE PIO INTERRUPTS
0018:04 3EFF                          MVI     A,0FFH  ;GET INTERRUPT MASK
001A:04 D307                          OUT     PIOBCR  ;MASK ALL INTERRUPTS
001C:04 32 0000"                      STA     INTMSK  ;SAVE INTERRUPT MASK
001F:04 C9                            RET             ;DONE
                              ;
0020:04 F5                    INTNIT::PUSH    PSW     ;SAVE VECTORED INTERRUPT NUMBER
0021:04 87                            ADD     A       ;CALC VECTORED INTERRUPT NUMBER * 2
0022:04 4F                            MOV     C,A     ;INTERRUPT NUMBER TO BC-REG
0023:04 0600                          MVI     B,0     ;DOUBLE LENGTH
0025:04 EB                            XCHG            ;INTERRUPT SERVICE ADDR TO DE-REG
0026:04 21 0001"                      LXI     H,ISRTBL  ;GET ISR ADDRESS TABLE
0029:04 09                            DAD     B       ;INDEX INTO ISR TABLE
002A:04 73                            MOV     M,E     ;STORE ISR ADDRESS IN ISR TABLE
002B:04 23                            INX     H
002C:04 72                            MOV     M,D
002D:04 F1                            POP     PSW     ;RESTORE VECTORED INTERRUPT NUMBER
002E:04 3C                            INR     A       ;INCREMENT INTERRUPT NUMBER
002F:04 47                            MOV     B,A     ;INTERRUPT NUMBER TO B-REG
0030:04 AF                            XRA     A       ;INITIALIZE RESULT VECTOR
0031:04 37                            STC             ;SET CARRY FLAG
0032:04 8F                    ..SL:   ADC     A       ;SHIFT CARRY FLAG LEFT
0033:04 10FD                          DJNZ    ..SL    ;INTERRUPT NUMBER + 1 TIMES
0035:04 2F                            CMA             ;COMPLIMENT RESULT VECTOR
```

```
0036:04 21 0000"            LXI     H,INTMSK  ;GET INTERRUPT MASK
0039:04 A6                  ANA     M       ;RESET INTERRUPT MASK BIT
003A:04 77                  MOV     M,A     ;UPDATE INTERRUPT MASK
003B:04 3E97                MVI     A,97H   ;GET PIO INTERRUPT CONTROL WORD
003D:04 D307                OUT     PIOBCR  ;ENABLE PIO INTERRUPTS
003F:04 7E                  MOV     A,M     ;GET INTERRUPT MASK
0040:04 D307                OUT     PIOBCR  ;SET PIO INTERRUPT MASK REGISTER
0042:04 C9                  RET             ;DONE
                    ;
0000'                       .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                    ;
0000'   ED73 0011"  PIOISR::SSPD    INTSP   ;SAVE STACK POINTER
0004'   31 0033"            LXI     SP,INTSTK  ;SET UP AUX STACK POINTER
0007'   F5                  PUSH    PSW     ;SAVE REGISTERS
0008'   C5                  PUSH    B
0009'   D5                  PUSH    D
000A'   E5                  PUSH    H
000B'   DB05        ..ISRL: IN      PIOBDR  ;GET VECTORED INTERRUPT STATUS
000D'   FEFF                CPI     0FFH    ;ANY VECTORED INTERRUPTS PENDING?
000F'   2822                JRZ     ..ISRX  ;IF NOT, CONTINUE
0011'   0608                MVI     B,8     ;ELSE, GET MAX NUMBER OF INTERRUPTS
0013'   1F          ..SVCL: RAR             ;VECTORED INTERRUPT PENDING?
0014'   3819                JRC     ..SVCC  ;IF NOT, CONTINUE
0016'   C5                  PUSH    B       ;ELSE, SAVE INTERRUPT COUNTER
0017'   F5                  PUSH    PSW     ;SAVE INTERRUPT STATUS
0018'   3E08                MVI     A,8     ;GET MAX NUMBER OF INTERRUPTS
001A'   90                  SUB     B       ;CALC CURRENT INTERRUPT NUMBER
001B'   87                  ADD     A       ;CALC CURRENT INTERRUPT NUMBER * 2
001C'   4F                  MOV     C,A     ;INTERRUPT NUMBER TO BC-REG
001D'   0600                MVI     B,0     ;DOUBLE LENGTH
001F'   21 0001"            LXI     H,ISRTBL  ;GET ISR ADDRESS TABLE
0022'   09                  DAD     B       ;INDEX INTO ISR TABLE
0023'   5E                  MOV     E,M     ;GET INTERRUPT SERVICE ADDRESS
0024'   23                  INX     H
0025'   56                  MOV     D,M
0026'   21 002C'            LXI     H,..RET ;GET RETURN ADDRESS
0029'   E5                  PUSH    H       ;PUSH RETURN ADDRESS ONTO STACK
002A'   EB                  XCHG            ;INTERRUPT SERVICE ADDR TO HL-REG
002B'   E9                  PCHL            ;TRANSFER TO INT SERVICE ROUTINE
002C'   F3          ..RET:  DI              ;DISABLE INTERRUPTS
002D'   F1                  POP     PSW     ;RESTORE INTERRUPT STATUS
002E'   C1                  POP     B       ;RESTORE INTERRUPT COUNTER
002F'   10E2        ..SVCC: DJNZ    ..SVCL  ;CONTINUE
0031'   18D8                JMPR    ..ISRL  ;CONTINUE
0033'   E1          ..ISRX: POP     H       ;RESTORE REGISTERS
0034'   D1                  POP     D
0035'   C1                  POP     B
0036'   F1                  POP     PSW
0037'   ED7B 0011"          LSPD    INTSP   ;RESTORE STACK POINTER
003B'   FB                  EI              ;ENABLE INTERRUPTS
003C'   ED4D                RETI            ;DONE
                    ;
                    .PRGEND
```

```
                                ;
                                ; VERSION: 01/26/84
                                ;
                                .IDENT  BNKMAS          ;MODULE ID
                                ;
                                .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                                ;
 0016                           MEMCR1  = 16H           ;MEMORY CONTROL REGISTER #1
 0017                           MEMCR2  = 17H           ;MEMORY CONTROL REGISTER #2
                                ;
 0000:04                                .LOC    .BANK.# ;LOCATE IN COMMON AREA
                                ;
 0000:04 ED73 00CF:04           BNKNIT::SSPD    SPSAVE  ;SAVE STACK POINTER
 0004:04 31 00F1:04                     LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
 0007:04 3E01                           MVI     A,1     ;GET BANK 1
 0009:04 CD 00B6:04                     CALL    SELMEM  ;SELECT BANK 1
 000C:04 21 0000                        LXI     H,0     ;INITIALIZE MEMORY PARITY
 000F:04 11 0000                        LXI     D,0
 0012:04 01 0000                        LXI     B,0
 0015:04 EDB0                           LDIR
 0017:04 3E68                           MVI     A,68H   ;RESET PARITY ERROR LATCH
 0019:04 D316                           OUT     MEMCR1
 001B:04 3EE8                           MVI     A,0E8H  ;ENABLE PARITY ERROR DETECTION
 001D:04 D316                           OUT     MEMCR1
 001F:04 21 0040:04                     LXI     H,RTCINT  ;GET INTERRUPT SERVICE ADDRESS
 0022:04 22 0002:05                     SHLD    CTCVEC#+2  ;SET INTERRUPT VECTOR
 0025:04 21 005C:04                     LXI     H,DSKINT  ;GET INTERRUPT SERVICE ADDRESS
 0028:04 22 0006:05                     SHLD    CTCVEC#+6  ;SET INTERRUPT VECTOR
 002B:04 21 0078:04                     LXI     H,SIOINT  ;GET INTERRUPT SERVICE ADDRESS
 002E:04 22 0000:06                     SHLD    SIOVEC# ;SET INTERRUPT VECTOR
 0031:04 21 0094:04                     LXI     H,PIOINT  ;GET INTERRUPT SERVICE ADDRESS
 0034:04 22 0002:07                     SHLD    PIOVEC#+2  ;SET INTERRUPT VECTOR
 0037:04 AF                             XRA     A       ;GET BANK 0
 0038:04 CD 00B6:04                     CALL    SELMEM  ;SELECT BANK 0
 003B:04 ED7B 00CF:04                   LSPD    SPSAVE  ;RESTORE STACK POINTER
 003F:04 C9                             RET             ;DONE
                                ;
 0040:04 ED73 00CF:04           RTCINT: SSPD    SPSAVE  ;SAVE STACK POINTER
 0044:04 31 00F1:04                     LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
 0047:04 F5                             PUSH    PSW     ;SAVE AF-REG
 0048:04 AF                             XRA     A       ;GET BANK 0
 0049:04 CD 00B6:04                     CALL    SELMEM  ;SELECT BANK 0
 004C:04 CD 0000:08                     CALL    RTCISR# ;PROCESS REAL TIME CLOCK INTERRUPT
 004F:04 F3                             DI              ;DISABLE INTERRUPTS
 0050:04 3E01                           MVI     A,1     ;GET BANK 1
 0052:04 CD 00B6:04                     CALL    SELMEM  ;SELECT BANK 1
 0055:04 F1                             POP     PSW     ;RESTORE AF-REG
 0056:04 ED7B 00CF:04                   LSPD    SPSAVE  ;RESTORE STACK POINTER
 005A:04 FB                             EI              ;ENABLE INTERRUPTS
 005B:04 C9                             RET             ;DONE
                                ;
 005C:04 ED73 00CF:04           DSKINT: SSPD    SPSAVE  ;SAVE STACK POINTER
 0060:04 31 00F1:04                     LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
 0063:04 F5                             PUSH    PSW     ;SAVE AF-REG
```

```
0064:04 AF                       XRA     A         ;GET BANK 0
0065:04 CD 00B6:04               CALL    SELMEM    ;SELECT BANK 0
0068:04 CD 0000:09               CALL    DSKISR#   ;PROCESS DISK INTERRUPT
006B:04 F3                       DI                ;DISABLE INTERRUPTS
006C:04 3E01                     MVI     A,1       ;GET BANK 1
006E:04 CD 00B6:04               CALL    SELMEM    ;SELECT BANK 1
0071:04 F1                       POP     PSW       ;RESTORE AF-REG
0072:04 ED7B 00CF:04             LSPD    SPSAVE    ;RESTORE STACK POINTER
0076:04 FB                       EI                ;ENABLE INTERRUPTS
0077:04 C9                       RET               ;DONE
                         ;
0078:04 ED73 00CF:04     SIOINT: SSPD    SPSAVE    ;SAVE STACK POINTER
007C:04 31 00F1:04               LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
007F:04 F5                       PUSH    PSW       ;SAVE AF-REG
0080:04 AF                       XRA     A         ;GET BANK 0
0081:04 CD 00B6:04               CALL    SELMEM    ;SELECT BANK 0
0084:04 CD 0000:0A               CALL    SIOISR#   ;PROCESS SERIAL I/O INTERRUPT
0087:04 F3                       DI                ;DISABLE INTERRUPTS
0088:04 3E01                     MVI     A,1       ;GET BANK 1
008A:04 CD 00B6:04               CALL    SELMEM    ;SELECT BANK 1
008D:04 F1                       POP     PSW       ;RESTORE AF-REG
008E:04 ED7B 00CF:04             LSPD    SPSAVE    ;RESTORE STACK POINTER
0092:04 FB                       EI                ;ENABLE INTERRUPTS
0093:04 C9                       RET               ;DONE
                         ;
0094:04 ED73 00CF:04     PIOINT: SSPD    SPSAVE    ;SAVE STACK POINTER
0098:04 31 00F1:04               LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
009B:04 F5                       PUSH    PSW       ;SAVE AF-REG
009C:04 AF                       XRA     A         ;GET BANK 0
009D:04 CD 00B6:04               CALL    SELMEM    ;SELECT BANK 0
00A0:04 CD 0000:0B               CALL    PIOISR#   ;PROCESS PARALLEL I/O INTERRUPT
00A3:04 F3                       DI                ;DISABLE INTERRUPTS
00A4:04 3E01                     MVI     A,1       ;GET BANK 1
00A6:04 CD 00B6:04               CALL    SELMEM    ;SELECT BANK 1
00A9:04 F1                       POP     PSW       ;RESTORE AF-REG
00AA:04 ED7B 00CF:04             LSPD    SPSAVE    ;RESTORE STACK POINTER
00AE:04 FB                       EI                ;ENABLE INTERRUPTS
00AF:04 C9                       RET               ;DONE
                         ;
00B0:04 F3               SELBNK::DI                ;DISABLE INTERRUPTS
00B1:04 CD 00B6:04               CALL    SELMEM    ;SELECT MEMORY BANK
00B4:04 FB                       EI                ;ENABLE INTERRUPTS
00B5:04 C9                       RET               ;DONE
                         ;
00B6:04 B7               SELMEM: ORA     A         ;BANK 0 REQUESTED?
00B7:04 200A                     JRNZ    ..BNK1    ;IF NOT, CONTINUE
00B9:04 AF                       XRA     A         ;ELSE, GET BANK 1 COMMAND
00BA:04 D317                     OUT     MEMCR2    ;DE-SELECT LOWER 48K OF MEMORY
00BC:04 3EEF                     MVI     A,0EFH    ;GET BANK 0 COMMAND
00BE:04 D316                     OUT     MEMCR1    ;ENABLE LOWER 48K OF MEMORY
00C0:04 C3 0016'                 JMP     FREBNK    ;FREE BANK 1 MUTUAL EXCLUSION
00C3:04 CD 0000'         ..BNK1: CALL    LOKBNK    ;GAIN BANK 1 MUTUAL EXCLUSION
00C6:04 3EE8                     MVI     A,0E8H    ;GET BANK 0 COMMAND
00C8:04 D316                     OUT     MEMCR1    ;DE-SELECT LOWER 48K OF MEMORY
```

```
00CA:04 3E07                      MVI     A,07H   ;GET BANK 1 COMMAND
00CC:04 D317                      OUT     MEMCR2  ;ENABLE LOWER 48K OF MEMORY
00CE:04 C9                        RET             ;DONE
                         ;
00CF:04 0000             SPSAVE: .WORD    0       ;STACK POINTER SAVE AREA
00D1:04 000000000000             .BYTE    [16*2]0 ;AUXILLIARY STACK AREA
00F1:04                  AUXSTK  = .              ;TOP OF AUXILLIARY STACK AREA
                         ;
0000"                            .LOC     .DATA.# ;LOCATE IN DATA AREA
                         ;
0000"   0001             BK1SPH: .WORD    1       ;MEMORY BANK 1 EXCLUSION SEMAPHOR
0002"   0002"            ..MXHD: .WORD    ..MXHD
0004"   0002"                    .WORD    ..MXHD
                         ;
0000'                            .LOC     .PROG.# ;LOCATE IN PROGRAM AREA
                         ;
0000'   C5               LOKBNK::PUSH     B       ;SAVE REGISTERS
0001'   D5                       PUSH     D
0002'   E5                       PUSH     H
0003'   CD 0000:0C               CALL     TSTIFF# ;GET INTERRUPT STATUS
0006'   F5                       PUSH     PSW     ;SAVE INTERRUPT STATUS
0007'   21 0000"                 LXI      H,BK1SPH  ;GET MUTUAL EXCLUSION SEMAPHORE
000A'   CD 0000:0D               CALL     WAIT#   ;WAIT ON MUTUAL EXCLUSION
000D'   F3                       DI               ;DISABLE INTERRUPTS
000E'   F1                       POP      PSW     ;RESTORE INTERRUPT STATUS
000F'   3801                     JRC      ..X     ;IF INTERRUPTS DISABLED, CONTINU
0011'   FB                       EI               ;ELSE, ENABLE INTERRUPTS
0012'   E1               ..X:    POP      H       ;RESTORE REGISTERS
0013'   D1                       POP      D
0014'   C1                       POP      B
0015'   C9                       RET              ;DONE
                         ;
0016'   C5               FREBNK::PUSH     B       ;SAVE REGISTERS
0017'   D5                       PUSH     D
0018'   E5                       PUSH     H
0019'   21 0000"                 LXI      H,BK1SPH  ;GET MUTUAL EXCLUSION SEMAPHOR
001C'   CD 0000:0E               CALL     SIGNAL# ;RELEASE MUTUAL EXCLUSION
001F'   E1                       POP      H       ;RESTORE REGISTERS
0020'   D1                       POP      D
0021'   C1                       POP      B
0022'   C9                       RET              ;DONE
                         ;
                         .PRGEND
```

```
                              ;
                              ; VERSION: 01/05/84
                              ;
                              .IDENT  CON192          ;MODULE ID
                              ;
                              .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                              ;
0000"                                 .LOC    .DATA.# ;LOCATE IN DATA AREA
                              ;
0000"   8F                    CONBR:: .BYTE   8FH     ;CONSOLE BAUD RATE CODE (19200 BAUD)

0001"   0C                    FFCHR:: .BYTE   AFF     ;FORM FEED CHARACTER
0002"   00                    INITC:  .BYTE   0       ;INITIALIZATION COMPLETE FLAG
                              ;
0000'                                 .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                              ;
0000'   21 0002"              CONDR$::LXI     H,INITC ;GET INITIALIZATION COMPLETE FLAG
0003'   7E                            MOV     A,M
0004'   B7                            ORA     A       ;INITIALIZATION COMPLETE FLAG SET?
0005'   CC 0013'                      CZ      ..INIT  ;IF NOT, INITIALIZE CONSOLE BAUD RAT
                              E
0008'   7B                    ..CDRV: MOV     A,E     ;GET FUNCTION NUMBER
0009'   D608                          SUI     8       ;FUNCTION NUMBER=8?
000B'   2823                          JRZ     CONSO   ;IF SO, ERROR SHIFT OUT
000D'   3D                            DCR     A       ;FUNCTION NUMBER=9?
000E'   2820                          JRZ     CONSI   ;IF SO, ERROR SHIFT IN
0010'   C3 0000:04                    JMP     SERIAL# ;ELSE, CONTINUE
0013'   35                    ..INIT: DCR     M       ;SET INITIALIZATION COMPLETE FLAG
0014'   D5                            PUSH    D       ;SAVE FUNCTION NUMBER
0015'   C5                            PUSH    B       ;SAVE CHANNEL NUMBER/CHARACTER
0016'   3A 0000"                      LDA     CONBR   ;GET CONSOLE BAUD RATE CODE
0019'   4F                            MOV     C,A     ;TELEVIDEO BAUD RATE CODE TO C-REG
001A'   1E03                          MVI     E,3     ;SET FUNCTION NUMBER=3
001C'   CD 0000:04                    CALL    SERIAL# ;SET CHANNEL BUAD RATE
001F'   3A 0001"                      LDA     FFCHR   ;GET FORM FEED CHARACTER
0022'   B7                            ORA     A       ;FORM FEED CHARACTER=0?
0023'   2808                          JRZ     ..NITX  ;IF SO, CONTINUE
0025'   C1                            POP     B       ;ELSE, RESTORE CHANNEL NUMBER
0026'   C5                            PUSH    B       ;SAVE CHANNEL NUMBER
0027'   4F                            MOV     C,A     ;FORM FEED CHARACTER TO C-REG
0028'   1E02                          MVI     E,2     ;SET FUNCTION NUMBER=2
002A'   CD 0000:04                    CALL    SERIAL# ;OUTPUT FORM FEED
002D'   C1                    ..NITX: POP     B       ;RESTORE CHANNEL NUMBER/CHARACTER
002E'   D1                            POP     D       ;RESTORE FUNCTION NUMBER
002F'   C9                            RET             ;DONE
                              ;
0030'                         CONSO:
0030'   CD 0000:05            CONSI:  CALL    DMS#    ;POSITION TO NEXT LINE
0033'   0D8A                          .ASCIS  [ACR] [ALF]
0035'   C9                            RET             ;DONE
                              ;
                              .PRGEND
```

```
                          ;
                          ; VERSION: 01/05/84
                          ;
                          .IDENT  LSTCTS          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
0000"                             .LOC    .DATA.# ;LOCATE IN DATA AREA
                          ;
0000"   6E                CTSBR:: .BYTE   6EH     ;BAUD RATE CODE (9600 BAUD)
0001"   0C                CTSFF:: .BYTE   AFF     ;FORM FEED CHARACTER
0002"   000000000000      INITC:  .BYTE   [16]0   ;INITIALIZATION COMPLETE FLAGS
                          ;
0000'                             .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                          ;
0000'   21 0002"          LSTDR%::LXI     H,INITC ;GET INITIALIZATION COMPLETE FLAGS
0003'   D5                        PUSH    D       ;SAVE FUNCTION NUMBER
0004'   58                        MOV     E,B     ;CHANNEL NUMBER TO DE-REG
0005'   1600                      MVI     D,0     ;DOUBLE LENGTH
0007'   19                        DAD     D       ;INDEX INTO FLAGS TABLE
0008'   D1                        POP     D       ;RESTORE FUNCTION NUMBER
0009'   7E                        MOV     A,M     ;GET INITIALIZATION COMPLETE FLAG
000A'   B7                        ORA     A       ;INITIALIZATION COMPLETE FLAG SET?
000B'   CC 0018'                  CZ      ..INIT  ;IF NOT, INITIALIZE LIST CHANNEL
000E'   7B                        MOV     A,E     ;GET FUNCTION NUMBER
000F'   FE02                      CPI     2       ;FUNCTION NUMBER=2?
0011'   281A                      JRZ     LSTOUT  ;IF SO, CONTINUE
0013'   FE07                      CPI     7       ;FUNCTION NUMBER=7?
0015'   2810                      JRZ     LSTWSR  ;IF SO, CONTINUE
0017'   C9                        RET             ;ELSE, DONE
0018'   35                ..INIT: DCR     M       ;SET INITIALIZATION COMPLETE FLAG
0019'   D5                        PUSH    D       ;SAVE FUNCTION NUMBER
001A'   C5                        PUSH    B       ;SAVE CHANNEL NUMBER/CHARACTER
001B'   3A 0000"                  LDA     CTSBR   ;GET BAUD RATE CODE
001E'   4F                        MOV     C,A     ;BAUD RATE CODE TO C-REG
001F'   1E03                      MVI     E,3     ;SET FUNCTION NUMBER=3
0021'   CD 0000:04                CALL    SERIAL# ;SET CHANNEL BUAD RATE
0024'   C1                        POP     B       ;RESTORE CHANNEL NUMBER/CHARACTER
0025'   D1                        POP     D       ;RESTORE FUNCTION NUMBER
0026'   C9                        RET             ;DONE
                          ;
0027'   3A 0001"          LSTWSR: LDA     CTSFF   ;GET FORM FEED CHARACTER
002A'   4F                        MOV     C,A     ;FORM FEED CHARACTER TO C-REG
002B'   1E02                      MVI     E,2     ;SET FUNCTION NUMBER=2
                          ;
002D'   C3 0000:04        LSTOUT: JMP     SERIAL# ;CONTINUE
                          ;
                          .PRGEND
```

```
                            ;
                            ; VERSION: 01/05/84
                            ;
                            .IDENT  LSTETX          ;MODULE ID
                            ;
                            .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                            ;
 0000"                              .LOC    .DATA.# ;LOCATE IN DATA AREA
                            ;
 0000"   07                 ETXBR:: .BYTE   7       ;BAUD RATE CODE (1200 BAUD)
 0001"   8C                 ETXLEN::.BYTE   140     ;CHARACTER COUNT BETWEEN ETX'S
 0002"   03                 ETXSEQ::.BYTE   3       ;MAX ESCAPE SEQUENCE LENGTH
 0003"   0C                 ETXFF:: .BYTE   AFF     ;FORM FEED CHARACTER
 0004"   000000000000       CHRCNT: .BYTE   [16]0   ;CHARACTER COUNT
 0014"   000000000000       SEQCNT: .BYTE   [16]0   ;SEQUENCE COUNT
 0024"   000000000000       INITC:  .BYTE   [16]0   ;INITIALIZATION COMPLETE FLAGS
                            ;
 0000'                              .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                            ;
 0000'   21 0024"           LSTDR$::LXI     H,INITC ;GET INITIALIZATION COMPLETE FLAGS
 0003'   CD 0084'                   CALL    INDEX   ;INDEX INTO FLAGS TABLE
 0006'   7E                         MOV     A,M     ;GET INITIALIZATION COMPLETE FLAG
 0007'   B7                         ORA     A       ;INITIALIZATION COMPLETE FLAG SET?
 0008'   CC 0015'                   CZ      ..INIT  ;IF NOT, INITIALIZE LIST CHANNEL
 000B'   7B                         MOV     A,E     ;GET FUNCTION NUMBER
 000C'   FE02                       CPI     2       ;FUNCTION NUMBER=2?
 000E'   281A                       JRZ     LSTOUT  ;IF SO, CONTINUE
 0010'   FE07                       CPI     7       ;FUNCTION NUMBER=7?
 0012'   2810                       JRZ     LSTWSR  ;IF SO, CONTINUE
 0014'   C9                         RET             ;ELSE, DONE
 0015'   35                 ..INIT: DCR     M       ;SET INITIALIZATION COMPLETE FLAG
 0016'   D5                         PUSH    D       ;SAVE FUNCTION NUMBER
 0017'   C5                         PUSH    B       ;SAVE CHANNEL NUMBER/CHARACTER
 0018'   3A 0000"                   LDA     ETXBR   ;GET BAUD RATE CODE
 001B'   4F                         MOV     C,A     ;BAUD RATE CODE TO C-REG
 001C'   1E03                       MVI     E,3     ;SET FUNCTION NUMBER=3
 001E'   CD 0000:04                 CALL    SERIAL# ;SET CHANNEL BUAD RATE
 0021'   C1                         POP     B       ;RESTORE CHANNEL NUMBER/CHARACTER
 0022'   D1                         POP     D       ;RESTORE FUNCTION NUMBER
 0023'   C9                         RET             ;DONE
                            ;
 0024'   3A 0003"           LSTWSR: LDA     ETXFF   ;GET FORM FEED CHARACTER
 0027'   4F                         MOV     C,A     ;FORM FEED CHARACTER TO C-REG
 0028'   1E02                       MVI     E,2     ;SET FUNCTION NUMBER=2
                            ;
 002A'   CD 007C'           LSTOUT: CALL    ..GCCA  ;GET CHARACTER COUNT ADDRESS
 002D'   7E                         MOV     A,M     ;GET CHARACTER COUNT
 002E'   21 0001"                   LXI     H,ETXLEN  ;GET CHARACTER COUNT BETWEEN ETX'S

 0031'   BE                         CMP     M       ;MAX CHARACTER COUNT OUTSTANDING?
 0032'   381B                       JRC     ..OUT   ;IF NOT, CONTINUE
 0034'   CD 0081'                   CALL    ..GSCA  ;ELSE, GET SEQUENCE COUNT ADDRESS
 0037'   7E                         MOV     A,M     ;GET SEQUENCE COUNT
 0038'   B7                         ORA     A       ;IN ESCAPE SEQUENCE?
```

```
0039'   2014                  JRNZ    ..OUT    ;IF SO, CONTINUE
003B'   C5                    PUSH    B        ;ELSE, SAVE OUTPUT CHARACTER
003C'   0E03                  MVI     C,AETX   ;GET ETX CHARACTER
003E'   CD 0074'              CALL    ..SOUT   ;OUTPUT ETX CHARACTER
0041'   CD 006B'      ..WAIT: CALL    ..SIN    ;ELSE, GET SERIAL INPUT
0044'   E67F                  ANI     7FH      ;STRIP SIGN BIT
0046'   D606                  SUI     AACK     ;CHARACTER=ACK?
0048'   20F7                  JRNZ    ..WAIT   ;IF NOT, WAIT
004A'   CD 007C'              CALL    ..GCCA   ;ELSE, GET CHARACTER COUNT ADDRESS
004D'   77                    MOV     M,A      ;RESET CHARACTER COUNT
004E'   C1                    POP     B        ;RESTORE OUTPUT CHARACTER
004F'   79            ..OUT:  MOV     A,C      ;GET OUTPUT CHARACTER
0050'   E67F                  ANI     7FH      ;STRIP SIGN BIT
0052'   FE1B                  CPI     AESC     ;CHARACTER=ESCAPE?
0054'   2007                  JRNZ    ..NESC   ;IF NOT, CONTINUE
0056'   CD 0081'              CALL    ..GSCA   ;ELSE, GET SEQUENCE COUNT ADDRESS
0059'   3A 0002"              LDA     ETXSEQ   ;GET MAX ESCAPE SEQUENCE LENGTH
005C'   77                    MOV     M,A      ;SET SEQUENCE COUNT
005D'   CD 0074'      ..NESC: CALL    ..SOUT   ;OUTPUT CHARACTER
0060'   CD 007C'              CALL    ..GCCA   ;GET CHARACTER COUNT ADDRESS
0063'   34                    INR     M        ;INCREMENT CHARACTER COUNT
0064'   CD 0081'              CALL    ..GSCA   ;GET SEQUENCE COUNT ADDRESS
0067'   35                    DCR     M        ;DECREMENT SEQUENCE COUNT
0068'   F0                    RP               ;IF POSITIVE, DONE
0069'   34                    INR     M        ;ELSE, RESTORE COUNT TO 0
006A'   C9                    RET              ;DONE
006B'   C5            ..SIN:  PUSH    B        ;SAVE CHANNEL NUMBER/CHARACTER
006C'   D5                    PUSH    D        ;SAVE FUNCTION NUMBER
006D'   1E01                  MVI     E,1      ;SET FUNCTION NUMBER=1
006F'   CD 0000:04            CALL    SERIAL#  ;GET SERIAL INPUT
0072'   1805                  JMPR    ..SIOC   ;CONTINUE
0074'   C5            ..SOUT: PUSH    B        ;SAVE CHANNEL NUMBER/CHARACTER
0075'   D5                    PUSH    D        ;SAVE FUNCTION NUMBER
0076'   CD 0000:04            CALL    SERIAL#  ;OUTPUT CHARACTER
0079'   D1            ..SIOC: POP     D        ;RESTORE FUNCTION NUMBER
007A'   C1                    POP     B        ;RESTORE CHANNEL NUMBER/CHARACTER
007B'   C9                    RET              ;DONE
007C'   21 0004"      ..GCCA: LXI     H,CHRCNT  ;GET CHARACTER COUNT TABLE
007F'   1803                  JMPR    INDEX    ;CONTINUE
0081'   21 0014"      ..GSCA: LXI     H,SEQCNT  ;GET SEQUENCE COUNT TABLE
                      ;
0084'   D5            INDEX:  PUSH    D        ;SAVE FUNCTION NUMBER
0085'   58                    MOV     E,B      ;CHANNEL NUMBER TO DE-REG
0086'   1600                  MVI     D,0      ;DOUBLE LENGTH
0088'   19                    DAD     D        ;INDEX INTO TABLE
0089'   D1                    POP     D        ;RESTORE FUNCTION NUMBER
008A'   C9                    RET              ;DONE
                      ;
                      .PRGEND
```

```
                            ;
                            ; VERSION: 01/05/84
                            ;
                            .IDENT  LSTXON          ;MODULE ID
                            ;
                            .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                            ;
0000"                               .LOC    .DATA.# ;LOCATE IN DATA AREA
                            ;
0000"   07                  XONBR:: .BYTE   7       ;BAUD RATE CODE (1200 BAUD)
0001"   0C                  XONFF:: .BYTE   AFF     ;FORM FEED CHARACTER
0002"   000000000000        INITC:  .BYTE   [16]0   ;INITIALIZATION COMPLETE FLAGS
                            ;
0000'                               .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                            ;
0000'   21 0002"            LSTDR$::LXI     H,INITC ;GET INITIALIZATION COMPLETE FLAGS
0003'   D5                          PUSH    D       ;SAVE FUNCTION NUMBER
0004'   58                          MOV     E,B     ;CHANNEL NUMBER TO DE-REG
0005'   1600                        MVI     D,0     ;DOUBLE LENGTH
0007'   19                          DAD     D       ;INDEX INTO FLAGS TABLE
0008'   D1                          POP     D       ;RESTORE FUNCTION NUMBER
0009'   7E                          MOV     A,M     ;GET INITIALIZATION COMPLETE FLAG
000A'   B7                          ORA     A       ;INITIALIZATION COMPLETE FLAG SET?
000B'   CC 0018'                    CZ      ..INIT  ;IF NOT, INITIALIZE LIST CHANNEL
000E'   7B                          MOV     A,E     ;GET FUNCTION NUMBER
000F'   FE02                        CPI     2       ;FUNCTION NUMBER=2?
0011'   281A                        JRZ     LSTOUT  ;IF SO, CONTINUE
0013'   FE07                        CPI     7       ;FUNCTION NUMBER=7?
0015'   2810                        JRZ     LSTWSR  ;IF SO, CONTINUE
0017'   C9                          RET             ;ELSE, DONE
0018'   35                  ..INIT: DCR     M       ;SET INITIALIZATION COMPLETE FLAG
0019'   D5                          PUSH    D       ;SAVE FUNCTION NUMBER
001A'   C5                          PUSH    B       ;SAVE CHANNEL NUMBER/CHARACTER
001B'   3A 0000"                    LDA     XONBR   ;GET BAUD RATE CODE
001E'   4F                          MOV     C,A     ;BAUD RATE CODE TO C-REG
001F'   1E03                        MVI     E,3     ;SET FUNCTION NUMBER=3
0021'   CD 0000:04                  CALL    SERIAL# ;SET CHANNEL BUAD RATE
0024'   C1                          POP     B       ;RESTORE CHANNEL NUMBER/CHARACTER
0025'   D1                          POP     D       ;RESTORE FUNCTION NUMBER
0026'   C9                          RET             ;DONE
                            ;
0027'   3A 0001"            LSTWSR: LDA     XONFF   ;GET FORM FEED CHARACTER
002A'   4F                          MOV     C,A     ;FORM FEED CHARACTER TO C-REG
002B'   1E02                        MVI     E,2     ;SET FUNCTION NUMBER=2
                            ;
002D'   CD 0048'            LSTOUT: CALL    ..SST   ;GET SERIAL STATUS
0030'   B7                          ORA     A       ;CHARACTER AVAILABLE?
0031'   2812                        JRZ     ..OUT   ;IF NOT, CONTINUE
0033'   CD 0051'                    CALL    ..SIN   ;ELSE, GET SERIAL INPUT
0036'   E67F                        ANI     7FH     ;STRIP SIGN BIT
0038'   FE13                        CPI     ADC3    ;CHARACTER=DC3 (XOFF)?
003A'   20F1                        JRNZ    LSTOUT  ;IF NOT, WAIT
003C'   CD 0051'            ..WAIT: CALL    ..SIN   ;GET SERIAL INPUT
003F'   E67F                        ANI     7FH     ;STRIP SIGN BIT
```

```
0041'  FE11                      CPI     ADC1     ;CHARACTER=DC1 (XON)?
0043'  20F7                      JRNZ    ..WAIT   ;IF NOT, WAIT
0045'  C3 0000:04       ..OUT:   JMP     SERIAL#  ;OUTPUT CHARACTER
0048'  C5               ..SST:   PUSH    B        ;SAVE CHANNEL NUMBER/CHARACTER
0049'  D5                        PUSH    D        ;SAVE FUNCTION NUMBER
004A'  1E00                      MVI     E,0      ;SET FUNCTION NUMBER=0
004C'  CD 0000:04                CALL    SERIAL#  ;GET SERIAL STATUS
004F'  1807                      JMPR    ..SSIC   ;CONTINUE
0051'  C5               ..SIN:   PUSH    B        ;SAVE CHANNEL NUMBER/CHARACTER
0052'  D5                        PUSH    D        ;SAVE FUNCTION NUMBER
0053'  1E01                      MVI     E,1      ;SET FUNCTION NUMBER=1
0055'  CD 0000:04                CALL    SERIAL#  ;GET SERIAL INPUT
0058'  D1               ..SSIC:  POP     D        ;RESTORE FUNCTION NUMBER
0059'  C1                        POP     B        ;RESTORE CHANNEL NUMBER/CHARACTER
005A'  C9                        RET              ;DONE
                        ;
                        .PRGEND
                        ;
                        ; VERSION: 01/22/84
                        ;
                        .IDENT  SPDMAS          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
0000                    SIOADR  = 00H           ;SIO PORT A DATA REGISTER
0001                    SIOACR  = 01H           ;SIO PORT A CONTROL REGISTER
0002                    SIOBDR  = 02H           ;SIO PORT B DATA REGISTER
0003                    SIOBCR  = 03H           ;SIO PORT B CONTROL REGISTER
                        ;
0004                    PIOADR  = 04H           ;PIO PORT A DATA REGISTER
0005                    PIOBDR  = 05H           ;PIO PORT B DATA REGISTER
0006                    PIOACR  = 06H           ;PIO PORT A CONTROL REGISTER
0007                    PIOBCR  = 07H           ;PIO PORT B CONTROL REGISTER
                        ;
0018                    SIOBRR  = 18H           ;SIO BAUD RATE GENERATOR REGISTER
                        ;
0000                    RDA     = 0             ;RECEIVED DATA AVAILABLE BIT
0002                    TBE     = 2             ;TRANSMIT BUFFER EMPTY BIT
0003                    DCD     = 3             ;DATA CARRIER DETECT BIT
0005                    CTS     = 5             ;CLEAR TO SEND BIT
                        ;
0000                    PAUSE   = 0             ;PAUSE FLAG (S00FLG/S10FLG)
                        ;
0000"                           .LOC    .DATA.# ;LOCATE IN DATA AREA
                        ;
0000"   08              CTSMSK::.BYTE   1<DCD   ;CTS HANDSHAKE MASK
0001"   00              BAUDRT: .BYTE   0       ;BAUD RATE REGISTER VALUE
                        ;
0002"   0040            S0IBSZ::.WORD   64      ;SERIAL 0 INPUT BUFFER SIZE
0004"   0000            S0IBUF: .WORD   0       ;SERIAL 0 INPUT BUFFER ADDRESS
0006"   0000            S0IPTR: .WORD   0       ;SERIAL 0 INPUT POINTER
0008"   0000            S0OPTR: .WORD   0       ;SERIAL 0 OUTPUT POINTER
000A"   0000            S0ICNT: .WORD   0       ;SERIAL 0 INPUT COUNT
000C"   00              S0IWCT: .BYTE   0       ;SERIAL 0 INPUT WAIT COUNT
000D"   00              S0OCHR: .BYTE   0       ;SERIAL 0 OUTPUT CHARACTER
000E"   00              S0OFLG: .BYTE   0       ;SERIAL 0 OUTPUT XON/XOFF FLAG
000F"   00              S0BR:   .BYTE   0       ;SERIAL 0 BAUD RATE CODE
                        ;
0010"                   S0ISPH:                 ;SERIAL 0 INPUT SEMAPHORE
0010"   0000                    .WORD   0       ;SEMAPHORE COUNT
0012"   0012"           ..S0IH: .WORD   ..S0IH  ;SEMAPHORE P/D HEAD
0014"   0012"                   .WORD   ..S0IH
                        ;
                                                ;SERIAL 0 OUTPUT SEMAPHORE
0016"   0000            S0OSPH: .WORD   0       ;SEMAPHORE COUNT
0018"   0018"           ..S0OH: .WORD   ..S0OH  ;SEMAPHORE P/D HEAD
001A"   0018"                   .WORD   ..S0OH
                        ;
001C"   0040            S1IBSZ::.WORD   64      ;SERIAL 1 INPUT BUFFER SIZE
001E"   0000            S1IBUF: .WORD   0       ;SERIAL 1 INPUT BUFFER ADDRESS
0020"   0000            S1IPTR: .WORD   0       ;SERIAL 1 INPUT POINTER
```

```
0022"    0000             S1OPTR: .WORD    0         ;SERIAL 1 OUTPUT POINTER
0024"    0000             S1ICNT: .WORD    0         ;SERIAL 1 INPUT COUNT
0026"    00               S1IWCT: .BYTE    0         ;SERIAL 1 INPUT WAIT COUNT
0027"    00               S1OCHR: .BYTE    0         ;SERIAL 1 OUTPUT CHARACTER
0028"    00               S1OFLG: .BYTE    0         ;SERIAL 1 OUTPUT XON/XOFF FLAG
0029"    00               S1BR:   .BYTE    0         ;SERIAL 1 BAUD RATE CODE
                          ;
                                                     ;SERIAL 1 INPUT SEMAPHORE
002A"    0000             S1ISPH: .WORD    0         ;SEMAPHORE COUNT
002C"    002C"            ..S1IH: .WORD    ..S1IH    ;SEMAPHORE P/D HEAD
002E"    002C"                    .WORD    ..S1IH
                          ;
                                                     ;SERIAL 1 OUTPUT SEMAPHORE
0030"    0000             S1OSPH: .WORD    0         ;SEMAPHORE COUNT
0032"    0032"            ..S1OH: .WORD    ..S1OH    ;SEMAPHORE P/D HEAD
0034"    0032"                    .WORD    ..S1OH
                          ;
0000:04                           .LOC     .INIT.# ;LOCATE IN INITIALIZATION AREA
                          ;
0000:04 21 0000:05        SPINIT::LXI      H,SIOVEC#  ;GET INTERRUPT VECTOR ADDRESS
0003:04 7D                        MOV      A,L       ;GET LSB OF INTERRUPT VECTOR
0004:04 32 0046:04                STA      WR2CWD    ;SET WRITE REGISTER 2 CONTROL WORD
0007:04 21 01BE'                  LXI      H,SIOISR  ;GET INTERRUPT SERVICE ADDRESS
000A:04 22 0000:05                SHLD     SIOVEC# ;SET INTERRUPT VECTOR ADDRESS
000D:04 21 003C:04                LXI      H,SIOPGM  ;GET SIO PROGRAM LIST
0010:04 01 0901                   LXI      B,SIOAPL<8!SIOACR  ;B=LENGTH/C=CONTROL REG
0013:04 EDB3                      OUTIR              ;PROGRAM SIO PORT A
0015:04 21 003C:04                LXI      H,SIOPGM  ;GET SIO PROGRAM LIST
0018:04 01 0B03                   LXI      B,SIOBPL<8!SIOBCR  ;B=LENGTH/C=CONTROL REG
001B:04 EDB3                      OUTIR              ;PROGRAM SIO PORT B
001D:04 2A 0002"                  LHLD     S0IBSZ    ;GET SERIAL 0 INPUT BUFFER SIZE
0020:04 CD 0000:06                CALL     ALLOC#    ;ALLOCATE PACKET FOR SERIAL BUFFER
0023:04 22 0004"                  SHLD     S0IBUF    ;SAVE SERIAL 0 INPUT BUFFER ADDRESS
0026:04 22 0006"                  SHLD     S0IPTR    ;SET SERIAL 0 INPUT POINTER
0029:04 22 0008"                  SHLD     S0OPTR    ;SET SERIAL 0 OUTPUT POINTER
002C:04 2A 001C"                  LHLD     S1IBSZ    ;GET SERIAL 1 INPUT BUFFER SIZE
002F:04 CD 0000:06                CALL     ALLOC#    ;ALLOCATE PACKET FOR SERIAL BUFFER
0032:04 22 001E"                  SHLD     S1IBUF    ;SAVE SERIAL 1 INPUT BUFFER ADDRESS
0035:04 22 0020"                  SHLD     S1IPTR    ;SET SERIAL 1 INPUT POINTER
0038:04 22 0022"                  SHLD     S1OPTR    ;SET SERIAL 1 OUTPUT POINTER
003B:04 C9                        RET                ;DONE
                          ;
003C:04 18                SIOPGM: .BYTE    18H       ;RESET CHANNEL
003D:04 04                        .BYTE    4         ;SELECT WR4
003E:04 44                        .BYTE    44H       ;WRITE REGISTER 4 CONTROL WORD
003F:04 05                        .BYTE    5         ;SELECT WR5
0040:04 EA                        .BYTE    0EAH      ;WRITE REGISTER 5 CONTROL WORD
0041:04 03                        .BYTE    3         ;SELECT WR3
0042:04 C1                        .BYTE    0C1H      ;WRITE REGISTER 3 CONTROL WORD
0043:04 01                        .BYTE    1         ;SELECT WR1
0044:04 10                        .BYTE    10H       ;WRITE REGISTER 1 CONTROL WORD
                          ;
0009                      SIOAPL  = .-SIOPGM         ;SIO PORT A PROGRAM LENGTH
                          ;
```

```
 0045:04 02                      .BYTE   2         ;SELECT WR2
 0046:04 00             WR2CWD: .BYTE   0         ;WRITE REGISTER 2 CONTROL WORD
                        ;
 000B                   SIOBPL  = .-SIOPGM        ;SIO PORT B PROGRAM LENGTH
                        ;
 0000'                          .LOC    .PROG.#  ;LOCATE IN PROGRAM AREA
                        ;
 0000'                  SERIAL::
 0000'   7B             COMDRV::MOV     A,E       ;GET FUNCTION NUMBER
 0001'   B7                     ORA     A         ;FUNCTION NUMBER=0?
 0002'   281D                   JRZ     SERST     ;IF SO, CONTINUE
 0004'   FE0A                   CPI     10        ;FUNCTION NUMBER=10?
 0006'   CA 00A4'               JZ      SEROPT    ;IF SO, CONTINUE
 0009'   3D                     DCR     A         ;FUNCTION NUMBER=1?
 000A'   282E                   JRZ     SERIN     ;IF SO, CONTINUE
 000C'   3D                     DCR     A         ;FUNCTION NUMBER=2?
 000D'   CA 00F0'               JZ      SEROUT    ;IF SO, CONTINUE
 0010'   3D                     DCR     A         ;FUNCTION NUMBER=3?
 0011'   CA 02DA'               JZ      SERSBR    ;IF SO, CONTINUE
 0014'   3D                     DCR     A         ;FUNCTION NUMBER=4?
 0015'   CA 0304'               JZ      SERRBR    ;IF SO, CONTINUE
 0018'   3D                     DCR     A         ;FUNCTION NUMBER=5?
 0019'   CA 0310'               JZ      SERSMC    ;IF SO, CONTINUE
 001C'   3D                     DCR     A         ;FUNCTION NUMBER=6?
 001D'   CA 0332'               JZ      SERRMC    ;IF SO, CONTINUE
 0020'   C9                     RET               ;ELSE, DONE
                        ;
 0021'   78             SERST:  MOV     A,B       ;GET CHANNEL NUMBER
 0022'   ED4B 000A"             LBCD    SOICNT    ;GET SERIAL 0 INPUT BUFFER COUNT
 0026'   2A 0008"               LHLD    SOOPTR    ;GET SERIAL 0 OUTPUT POINTER
 0029'   B7                     ORA     A         ;CHANNEL NUMBER=0
 002A'   2807                   JRZ     ..COM     ;IF SO, CONTINUE
 002C'   ED4B 0024"             LBCD    S1ICNT    ;GET SERIAL 1 INPUT BUFFER COUNT
 0030'   2A 0022"               LHLD    S1OPTR    ;GET SERIAL 1 OUTPUT POINTER
 0033'   78             ..COM:  MOV     A,B
 0034'   B1                     ORA     C         ;SERIAL INPUT BUFFER COUNT=0?
 0035'   C8                     RZ                ;IF SO, DONE
 0036'   4E                     MOV     C,M       ;ELSE, GET SERIAL INPUT CHARACTER
 0037'   3EFF                   MVI     A,0FFH    ;SET RETURN CODE=0FFH
 0039'   C9                     RET               ;DONE
                        ;
 003A'   78             SERIN:  MOV     A,B       ;GET CHANNEL NUMBER
 003B'   B7                     ORA     A         ;CHANNEL NUMBER=0?
 003C'   2033                   JRNZ    ..S1I     ;IF NOT, CONTINUE
 003E'   F3             ..S0I:  DI                ;ELSE, DISABLE INTERRUPTS
 003F'   2A 000A"               LHLD    SOICNT    ;GET SERIAL 0 INPUT COUNT
 0042'   7C                     MOV     A,H
 0043'   B5                     ORA     L         ;SERIAL 0 INPUT COUNT=0?
 0044'   281F                   JRZ     ..WT0     ;IF SO, CONTINUE
 0046'   2B                     DCX     H         ;DECREMENT SERIAL 0 INPUT COUNT
 0047'   22 000A"               SHLD    SOICNT    ;UPDATE SERIAL 0 INPUT COUNT
 004A'   2A 0008"               LHLD    SOOPTR    ;GET SERIAL 0 OUTPUT POINTER
 004D'   7E                     MOV     A,M       ;GET CHARACTER FROM BUFFER
 004E'   23                     INX     H         ;INCREMENT SERIAL 0 OUTPUT POINTER
```

```
004F'   EB                      XCHG            ;SERIAL 0 OUTPUT POINTER TO DE-REG
0050'   2A 0002"                LHLD    S0IBSZ  ;GET SERIAL 0 INPUT BUFFER SIZE
0053'   2B                      DCX     H       ;DECREMENT INPUT BUFFER SIZE
0054'   ED4B 0004"              LBCD    S0IBUF  ;GET SERIAL 0 INPUT BUFFER ADDRESS
0058'   09                      DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
0059'   ED52                    DSBC    D       ;BUFFER WRAP-AROUND?
005B'   3002                    JRNC    ..NWA0  ;IF NOT, CONTINUE
005D'   59                      MOV     E,C     ;GET SERIAL 0 INPUT BUFFER ADDRESS
005E'   50                      MOV     D,B
005F'   ED53 0008"      ..NWA0: SDED    S0OPTR  ;UPDATE SERIAL 0 OUTPUT POINTER
0063'   FB                      EI              ;ENABLE INTERRUPTS
0064'   C9                      RET             ;DONE
0065'   21 000C"        ..WT0:  LXI     H,S0IWCT  ;GET SERIAL 0 INPUT WAIT COUNT
0068'   34                      INR     M       ;INCREMENT INPUT WAIT COUNT
0069'   21 0010"                LXI     H,S0ISPH  ;GET SERIAL 0 INPUT SEMAPHORE
006C'   CD 0000:07              CALL    WAIT#   ;WAIT FOR CONSOLE INPUT
006F'   18CD                    JMPR    ..S0I   ;CONTINUE
0071'   F3              ..S1I:  DI              ;DISABLE INTERRUPTS
0072'   2A 0024"                LHLD    S1ICNT  ;GET SERIAL 1 INPUT COUNT
0075'   7C                      MOV     A,H
0076'   B5                      ORA     L       ;SERIAL 1 INPUT COUNT=0?
0077'   281F                    JRZ     ..WT1   ;IF SO, CONTINUE
0079'   2B                      DCX     H       ;DECREMENT SERIAL 1 INPUT COUNT
007A'   22 0024"                SHLD    S1ICNT  ;UPDATE SERIAL 1 INPUT COUNT
007D'   2A 0022"                LHLD    S1OPTR  ;GET SERIAL 1 OUTPUT POINTER
0080'   7E                      MOV     A,M     ;GET CHARACTER FROM BUFFER
0081'   23                      INX     H       ;INCREMENT SERIAL 1 OUTPUT POINTER
0082'   EB                      XCHG            ;SERIAL 1 OUTPUT POINTER TO DE-REG
0083'   2A 001C"                LHLD    S1IBSZ  ;GET SERIAL 1 INPUT BUFFER SIZE
0086'   2B                      DCX     H       ;DECREMENT INPUT BUFFER SIZE
0087'   ED4B 001E"              LBCD    S1IBUF  ;GET SERIAL 1 INPUT BUFFER ADDRESS
008B'   09                      DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
008C'   ED52                    DSBC    D       ;BUFFER WRAP-AROUND?
008E'   3002                    JRNC    ..NWA1  ;IF NOT, CONTINUE
0090'   59                      MOV     E,C     ;GET SERIAL 1 INPUT BUFFER ADDRESS
0091'   50                      MOV     D,B
0092'   ED53 0022"      ..NWA1: SDED    S1OPTR  ;UPDATE SERIAL 1 OUTPUT POINTER
0096'   FB                      EI              ;ENABLE INTERRUPTS
0097'   C9                      RET             ;DONE
0098'   21 0026"        ..WT1:  LXI     H,S1IWCT  ;GET SERIAL 1 INPUT WAIT COUNT
009B'   34                      INR     M       ;INCREMENT INPUT WAIT COUNT
009C'   21 002A"                LXI     H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
009F'   CD 0000:07              CALL    WAIT#   ;WAIT FOR CONSOLE INPUT
00A2'   18CD                    JMPR    ..S1I   ;CONTINUE
                        ;
00A4'   78              SEROPT: MOV     A,B     ;GET CHANNEL NUMBER
00A5'   B7                      ORA     A       ;CHANNEL NUMBER=1?
00A6'   3E10                    MVI     A,10H   ;GET RESET EXTERNAL STATUS COMMAND
00A8'   2023                    JRNZ    ..S10   ;IF CHANNEL NUMBER=1, CONTINUE
00AA'   D301                    OUT     SIOACR  ;ELSE, RESET EXTERNAL STATUS
00AC'   DB01                    IN      SIOACR  ;GET SIO PORT A STATUS
00AE'   E604                    ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
00B0'   C8                      RZ              ;IF NOT, DONE
00B1'   3A 000E"                LDA     S0OFLG  ;GET FLAG
```

```
 00B4'   CB47                BIT     PAUSE,A ;IN A PAUSE?
 00B6'   3E00                MVI     A,0     ;PRESET RETURN CODE
 00B8'   C0                  RNZ             ;IF SO, DONE
 00B9'   3A 000F"            LDA     S0BR    ;ELSE, GET SERIAL 0 BAUD RATE CODE
 00BC'   E640                ANI     1<6     ;CTS HANDSHAKING REQUESTED?
 00BE'   2807                JRZ     ..NHR0  ;IF NOT, CONTINUE
 00C0'   DB01                IN      SIOACR  ;ELSE, GET SIO PORT A STATUS
 00C2'   21 0000"            LXI     H,CTSMSK  ;GET CTS MASK
 00C5'   A6                  ANA     M       ;CHECK IF CLEAR TO SEND
 00C6'   C8                  RZ              ;IF CLEAR TO SEND FALSE, DONE
 00C7'   79          ..NHR0: MOV     A,C     ;GET SERIAL 0 OUTPUT CHARACTER
 00C8'   D300                OUT     SIOADR  ;OUTPUT CHARACTER
 00CA'   3EFF                MVI     A,0FFH  ;SET RETURN CODE=0FFH
 00CC'   C9                  RET             ;DONE
 00CD'   D303        ..S10:  OUT     SIOBCR  ;RESET EXTERNAL STATUS
 00CF'   DB03                IN      SIOBCR  ;GET SIO PORT B STATUS
 00D1'   E604                ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
 00D3'   C8                  RZ              ;IF NOT, DONE
 00D4'   3A 0028"            LDA     S1OFLG  ;GET FLAG
 00D7'   CB47                BIT     PAUSE,A ;IN A PAUSE?
 00D9'   3E00                MVI     A,0     ;PRESET RETURN CODE
 00DB'   C0                  RNZ             ;IF SO, DONE
 00DC'   3A 0029"            LDA     S1BR    ;ELSE, GET SERIAL 1 BAUD RATE CODE
 00DF'   E640                ANI     1<6     ;CTS HANDSHAKING REQUESTED?
 00E1'   2807                JRZ     ..NHR1  ;IF NOT, CONTINUE
 00E3'   DB03                IN      SIOBCR  ;ELSE, GET SIO PORT B STATUS
 00E5'   21 0000"            LXI     H,CTSMSK  ;GET CTS MASK
 00E8'   A6                  ANA     M       ;CHECK IF CLEAR TO SEND
 00E9'   C8                  RZ              ;IF CLEAR TO SEND FALSE, DONE
 00EA'   79          ..NHR1: MOV     A,C     ;GET SERIAL 1 OUTPUT CHARACTER
 00EB'   D302                OUT     SIOBDR  ;OUTPUT CHARACTER
 00ED'   3EFF                MVI     A,0FFH  ;SET RETURN CODE=0FFH
 00EF'   C9                  RET             ;DONE
                     ;
 00F0'   78          SEROUT: MOV     A,B     ;GET CHANNEL NUMBER
 00F1'   B7                  ORA     A       ;CHANNEL NUMBER=1?
 00F2'   3E10                MVI     A,10H   ;GET RESET EXTERNAL STATUS COMMAND
 00F4'   2032                JRNZ    ..S10   ;IF CHANNEL NUMBER=1, CONTINUE
 00F6'   D301                OUT     SIOACR  ;ELSE, RESET EXTERNAL STATUS
 00F8'   DB01                IN      SIOACR  ;GET SIO PORT A STATUS
 00FA'   E604                ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
 00FC'   281A                JRZ     ..S0NR  ;IF NOT, CONTINUE
 00FE'   3A 000E"            LDA     S0OFLG  ;GET FLAG
 0101'   CB47                BIT     PAUSE,A ;IN A PAUSE?
 0103'   2013                JRNZ    ..S0NR  ;IF SO, CONTINUE
 0105'   3A 000F"            LDA     S0BR    ;ELSE, GET SERIAL 0 BAUD RATE CODE
 0108'   E640                ANI     1<6     ;CTS HANDSHAKING REQUESTED?
 010A'   2808                JRZ     ..NHR0  ;IF NOT, CONTINUE
 010C'   DB01                IN      SIOACR  ;ELSE, GET SIO PORT A STATUS
 010E'   21 0000"            LXI     H,CTSMSK  ;GET CTS MASK
 0111'   A6                  ANA     M       ;CHECK IF CLEAR TO SEND
 0112'   2804                JRZ     ..S0NR  ;IF CLEAR TO SEND FALSE, CONTINUE
 0114'   79          ..NHR0: MOV     A,C     ;GET SERIAL 0 OUTPUT CHARACTER
 0115'   D300                OUT     SIOADR  ;OUTPUT CHARACTER
```

```
0117'   C9                      RET              ;DONE
0118'   79              ..SONR: MOV     A,C      ;GET SERIAL 0 OUTPUT CHARACTER
0119'   32 000D"                STA     SOOCHR   ;SAVE OUTPUT CHARACTER
011C'   11 015A'                LXI     D,SOOPOL  ;GET SERIAL 0 OUT POLL ROUTINE
011F'   CD 0000:08              CALL    LNKPOL# ;CREATE POLL ROUTINE
0122'   21 0016"                LXI     H,SOOSPH  ;GET SERIAL 0 OUT SEMAPHORE
0125'   C3 0000:07              JMP     WAIT#    ;DISPATCH IF NECESSARY
0128'   D303            ..S1O:  OUT     SIOBCR   ;RESET EXTERNAL STATUS
012A'   DB03                    IN      SIOBCR   ;GET SIO PORT B STATUS
012C'   E604                    ANI     1<TBE    ;TRANSMIT BUFFER EMPTY?
012E'   281A                    JRZ     ..S1NR   ;IF NOT, CONTINUE
0130'   3A 0028"                LDA     S1OFLG   ;GET FLAG
0133'   CB47                    BIT     PAUSE,A ;IN A PAUSE?
0135'   2013                    JRNZ    ..S1NR   ;IF SO, CONTINUE
0137'   3A 0029"                LDA     S1BR     ;ELSE, GET SERIAL 1 BAUD RATE CODE
013A'   E640                    ANI     1<6      ;CTS HANDSHAKING REQUESTED?
013C'   2808                    JRZ     ..NHR1   ;IF NOT, CONTINUE
013E'   DB03                    IN      SIOBCR   ;ELSE, GET SIO PORT B STATUS
0140'   21 0000"                LXI     H,CTSMSK  ;GET CTS MASK
0143'   A6                      ANA     M        ;CHECK IF CLEAR TO SEND
0144'   2804                    JRZ     ..S1NR   ;IF CLEAR TO SEND FALSE, CONTINUE
0146'   79              ..NHR1: MOV     A,C      ;GET SERIAL 1 OUTPUT CHARACTER
0147'   D302                    OUT     SIOBDR   ;OUTPUT CHARACTER
0149'   C9                      RET              ;DONE
014A'   79              ..S1NR: MOV     A,C      ;GET SERIAL 1 OUTPUT CHARACTER
014B'   32 0027"                STA     S1OCHR   ;SAVE OUTPUT CHARACTER
014E'   11 018C'                LXI     D,S1OPOL  ;GET SERIAL 1 OUT POLL ROUTINE
0151'   CD 0000:08              CALL    LNKPOL# ;CREATE POLL ROUTINE
0154'   21 0030"                LXI     H,S1OSPH  ;GET SERIAL 1 OUT SEMAPHORE
0157'   C3 0000:07              JMP     WAIT#    ;DISPATCH IF NECESSARY
                        ;
015A'                   SOOPOL:                  ;SERIAL 0 OUTPUT POLL ROUTINE
015A'   0000                    .WORD   0        ;SUCCESSOR LINK POINTER
015C'   0000                    .WORD   0        ;PREDECESSOR LINK POINTER
                        ;
015E'   3E10                    MVI     A,10H    ;GET RESET EXTERNAL STATUS COMMAND
0160'   D301                    OUT     SIOACR   ;RESET EXTERNAL STATUS
0162'   DB01                    IN      SIOACR   ;GET SIO PORT A STATUS
0164'   E604                    ANI     1<TBE    ;TRANSMIT BUFFER EMPTY?
0166'   C8                      RZ               ;IF NOT, DONE
0167'   3A 000E"                LDA     SOOFLG   ;GET FLAG
016A'   CB47                    BIT     PAUSE,A ;IN A PAUSE?
016C'   C0                      RNZ              ;IF SO, DONE
016D'   3A 000F"                LDA     SOBR     ;ELSE, GET SERIAL 0 BAUD RATE CODE
0170'   E640                    ANI     1<6      ;CTS HANDSHAKING REQUESTED?
0172'   2807                    JRZ     ..NHR    ;IF NOT, CONTINUE
0174'   DB01                    IN      SIOACR   ;ELSE, GET SIO PORT A STATUS
0176'   21 0000"                LXI     H,CTSMSK  ;GET CTS MASK
0179'   A6                      ANA     M        ;CHECK IF CLEAR TO SEND
017A'   C8                      RZ               ;IF CLEAR TO SEND FALSE, DONE
017B'   3A 000D"        ..NHR:  LDA     SOOCHR   ;GET SERIAL 0 OUTPUT CHARACTER
017E'   D300                    OUT     SIOADR   ;OUTPUT CHARACTER
0180'   21 015A'                LXI     H,SOOPOL  ;GET SERIAL 0 OUT POLL ROUTINE
0183'   CD 0000:09              CALL    UNLINK# ;UNLINK POLL ROUTINE
```

```
0186'   21 0016"                LXI     H,S0OSPH  ;GET SERIAL 0 OUT SEMAPHORE
0189'   C3 0000:0A              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
018C'                   S1OPOL:                 ;SERIAL 1 OUTPUT POLL ROUTINE
018C'   0000                    .WORD   0       ;SUCCESSOR LINK POINTER
018E'   0000                    .WORD   0       ;PREDECESSOR LINK POINTER
                        ;
0190'   3E10                    MVI     A,10H   ;GET RESET EXTERNAL STATUS COMMAND
0192'   D303                    OUT     SIOBCR  ;RESET EXTERNAL STATUS
0194'   DB03                    IN      SIOBCR  ;GET SIO PORT B STATUS
0196'   E604                    ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
0198'   C8                      RZ              ;IF NOT, DONE
0199'   3A 0028"                LDA     S1OFLG  ;GET FLAG
019C'   CB47                    BIT     PAUSE,A ;IN A PAUSE?
019E'   C0                      RNZ             ;IF SO, DONE
019F'   3A 0029"                LDA     S1BR    ;ELSE, GET SERIAL 1 BAUD RATE CODE
01A2'   E640                    ANI     1<6     ;CTS HANDSHAKING REQUESTED?
01A4'   2807                    JRZ     ..NHR   ;IF NOT, CONTINUE
01A6'   DB03                    IN      SIOBCR  ;ELSE, GET SIO PORT B STATUS
01A8'   21 0000"                LXI     H,CTSMSK  ;GET CTS MASK
01AB'   A6                      ANA     M       ;CHECK IF CLEAR TO SEND
01AC'   C8                      RZ              ;IF CLEAR TO SEND FALSE, DONE
01AD'   3A 0027"        ..NHR:  LDA     S1OCHR  ;GET SERIAL 1 OUTPUT CHARACTER
01B0'   D302                    OUT     SIOBDR  ;OUTPUT CHARACTER
01B2'   21 018C'                LXI     H,S1OPOL  ;GET SERIAL 1 OUT POLL ROUTINE
01B5'   CD 0000:09              CALL    UNLINK# ;UNLINK POLL ROUTINE
01B8'   21 0030"                LXI     H,S1OSPH  ;GET SERIAL 1 OUT SEMAPHORE
01BB'   C3 0000:0A              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
01BE'   ED73 0000:0B    SIOISR::SSPD    INTSP#  ;SAVE STACK POINTER
01C2'   31 0000:0C              LXI     SP,INTSTK#  ;SET UP AUX STACK POINTER
01C5'   F5                      PUSH    PSW     ;SAVE REGISTERS
01C6'   C5                      PUSH    B
01C7'   D5                      PUSH    D
01C8'   E5                      PUSH    H
01C9'   CD 01DA'                CALL    ..S0I   ;CHECK FOR SERIAL 0 INPUT
01CC'   CD 0256'                CALL    ..S1I   ;CHECK FOR SERIAL 1 INPUT
01CF'   E1                      POP     H       ;RESTORE REGISTERS
01D0'   D1                      POP     D
01D1'   C1                      POP     B
01D2'   F1                      POP     PSW
01D3'   ED7B 0000:0B            LSPD    INTSP#  ;RESTORE STACK POINTER
01D7'   FB                      EI              ;ENABLE INTERRUPTS
01D8'   ED4D                    RETI            ;DONE
01DA'   DB01            ..S0I:  IN      SIOACR  ;GET SIO PORT A STATUS
01DC'   CB47                    BIT     RDA,A   ;CHARACTER AVAILABLE
01DE'   C8                      RZ              ;IF NOT, DONE
01DF'   DB00                    IN      SIOADR  ;GET SIO PORT A DATA CHARACTER
01E1'   21 000F"                LXI     H,S0BR  ;GET SERIAL 0 BAUD RATE CODE
01E4'   CB6E                    BIT     5,M     ;INHIBIT INPUT FLAG SET?
01E6'   C0                      RNZ             ;IF SO, DONE
01E7'   4F                      MOV     C,A     ;SERIAL 0 DATA CHARACTER TO C-REG
01E8'   CB66                    BIT     4,M     ;XON/XOFF HANDSHAKING?
01EA'   281B                    JRZ     ..NXO   ;NO, CONTINUE
```

```
 01EC'   21 000E"              LXI    H,SOOFLG  ;GET FLAG
 01EF'   E67F                  ANI    7FH     ;STRIP SIGN BIT
 01F1'   FE13                  CPI    ADC3    ;XOFF?
 01F3'   2007                  JRNZ   ..NXOO  ;NO, CONTINUE
 01F5'   CB46                  BIT    PAUSE,M ;IN A PAUSE?
 01F7'   200E                  JRNZ   ..NXO   ;YES, ACCEPT XOFF AS DATA
 01F9'   CBC6                  SET    PAUSE,M ;ELSE, SET PAUSE FLAG
 01FB'   C9                    RET            ;AND INHIBIT INPUT OF XOFF
 01FC'   FE11          ..NXOO: CPI    ADC1    ;XON?
 01FE'   2007                  JRNZ   ..NXO   ;NO, CONTINUE
 0200'   CB46                  BIT    PAUSE,M ;IN A PAUSE?
 0202'   2803                  JRZ    ..NXO   ;NO, ACCEPT XON AS DATA
 0204'   CB86                  RES    PAUSE,M ;ELSE, SET PAUSE FLAG
 0206'   C9                    RET            ;AND INHIBIT INPUT OF XOFF
 0207'   21 000F"      ..NXO:  LXI    H,SOBR  ;GET SERIAL 0 BAUD RATE CODE
 020A'   CB7E                  BIT    7,M     ;SIGN BIT ON BAUD RATE CODE?
 020C'   2814                  JRZ    ..NADO  ;IF NOT, CONTINUE
 020E'   CBB9                  RES    7,C     ;ELSE, STRIP SIGN BIT ON CHARAC
 0210'   3A 0000:0D            LDA    ATNCHR# ;GET ATTENTION CHARACTER
 0213'   B9                    CMP    C       ;CHARACTER=ATTENTION CHARACTER?
 0214'   200C                  JRNZ   ..NADO  ;IF NOT, CONTINUE
 0216'   2A 0006"              LHLD   SOIPTR  ;ELSE, GET SERIAL 0 INPUT POINT
 0219'   22 0008"              SHLD   SOOPTR  ;RESET SERIAL 0 OUTPUT POINTER
 021C'   21 0000               LXI    H,0
 021F'   22 000A"              SHLD   SOICNT  ;SET SERIAL 0 INPUT COUNT=0
 0222'   2A 0002"      ..NADO: LHLD   SOIBSZ  ;GET SERIAL 0 INPUT BUFFER SIZE
 0225'   ED5B 000A"            LDED   SOICNT  ;GET SERIAL 0 INPUT COUNT
 0229'   13                    INX    D       ;INCREMENT SERIAL 0 INPUT COUNT
 022A'   B7                    ORA    A       ;CLEAR CARRY FLAG
 022B'   ED52                  DSBC   D       ;SERIAL 0 INPUT BUFFER FULL?
 022D'   D8                    RC             ;IF SO, DONE
 022E'   ED53 000A"            SDED   SOICNT  ;ELSE, UPDATE SERIAL 0 INPUT CC
 0232'   2A 0006"              LHLD   SOIPTR  ;GET SERIAL 0 INPUT POINTER
 0235'   71                    MOV    M,C     ;STORE INPUT CHARACTER IN BUFFE
 0236'   23                    INX    H       ;INCREMENT INPUT POINTER
 0237'   EB                    XCHG           ;DE=INPUT POINTER/HL=BUFFER SIZ
 0238'   2A 0002"              LHLD   SOIBSZ  ;GET SERIAL 0 INPUT BUFFER SIZE
 023B'   2B                    DCX    H       ;DECREMENT INPUT BUFFER SIZE
 023C'   ED4B 0004"            LBCD   SOIBUF  ;GET SERIAL 0 INPUT BUFFER ADDF
 0240'   09                    DAD    B       ;CALC LAST INPUT BUFFER ADDRESS
 0241'   ED52                  DSBC   D       ;BUFFER WRAP-AROUND?
 0243'   3002                  JRNC   ..NWAO  ;IF NOT, CONTINUE
 0245'   59                    MOV    E,C     ;GET SERIAL 0 INPUT BUFFER ADDF
 0246'   50                    MOV    D,B
 0247'   ED53 0006"    ..NWAO: SDED   SOIPTR  ;UPDATE SERIAL 0 INPUT POINTER
 024B'   11 000C"              LXI    D,SOIWCT  ;GET SERIAL 0 INPUT WAIT COUN
 024E'   21 0010"              LXI    H,SOISPH  ;GET SERIAL 0 INPUT SEMAPHORE
 0251'   CD 02D2'              CALL   ..SIGC  ;SIGNAL IF NECESSARY
 0254'   1884                  JMPR   ..SOI   ;CONTINUE
 0256'   DB03          ..S1I:  IN     SIOBCR  ;GET SIO PORT B STATUS
 0258'   CB47                  BIT    RDA,A   ;CHARACTER AVAILABLE
 025A'   C8                    RZ             ;IF NOT, DONE
 025B'   DB02                  IN     SIOBDR  ;GET SIO PORT B DATA CHARACTER
 025D'   21 0029"              LXI    H,S1BR  ;GET SERIAL 1 BAUD RATE CODE
```

```
0260'   CB6E                 BIT    5,M       ;INHIBIT INPUT FLAG SET?
0262'   C0                   RNZ              ;IF SO, DONE
0263'   4F                   MOV    C,A       ;SERIAL 1 DATA CHARACTER TO C-REG
0264'   CB66                 BIT    4,M       ;XON/XOFF HANDSHAKING?
0266'   281B                 JRZ    ..NX1     ;NO, CONTINUE
0268'   21 0028"             LXI    H,S1OFLG  ;GET FLAG
026B'   E67F                 ANI    7FH       ;STRIP SIGN BIT
026D'   FE13                 CPI    ADC3      ;XOFF?
026F'   2007                 JRNZ   ..NXO1    ;NO, CONTINUE
0271'   CB46                 BIT    PAUSE,M   ;IN A PAUSE?
0273'   200E                 JRNZ   ..NX1     ;YES, ACCEPT XOFF AS DATA
0275'   CBC6                 SET    PAUSE,M   ;ELSE, SET PAUSE FLAG
0277'   C9                   RET              ;AND INHIBIT INPUT OF XOFF
0278'   FE11         ..NXO1: CPI    ADC1      ;XON?
027A'   2007                 JRNZ   ..NX1     ;NO, CONTINUE
027C'   CB46                 BIT    PAUSE,M   ;IN A PAUSE?
027E'   2803                 JRZ    ..NX1     ;NO, ACCEPT XON AS DATA
0280'   CB86                 RES    PAUSE,M   ;ELSE, SET PAUSE FLAG
0282'   C9                   RET              ;AND INHIBIT INPUT OF XOFF
0283'   21 0029"     ..NX1:  LXI    H,S1BR    ;GET SERIAL 1 BAUD RATE CODE
0286'   CB7E                 BIT    7,M       ;ATTENTION DETECTION FLAG SET?
0288'   2814                 JRZ    ..NAD1    ;IF NOT, CONTINUE
028A'   CBB9                 RES    7,C       ;ELSE, STRIP SIGN BIT ON CHARACTER
028C'   3A 0000:0D           LDA    ATNCHR#   ;GET ATTENTION CHARACTER
028F'   B9                   CMP    C         ;CHARACTER=ATTENTION CHARACTER?
0290'   200C                 JRNZ   ..NAD1    ;IF NOT, CONTINUE
0292'   2A 0020"             LHLD   S1IPTR    ;ELSE, GET SERIAL 1 INPUT POINTER
0295'   22 0022"             SHLD   S1OPTR    ;RESET SERIAL 1 OUTPUT POINTER
0298'   21 0000              LXI    H,0
029B'   22 0024"             SHLD   S1ICNT    ;SET SERIAL 1 INPUT COUNT=1
029E'   2A 001C"     ..NAD1: LHLD   S1IBSZ    ;GET SERIAL 1 INPUT BUFFER SIZE
02A1'   ED5B 0024"           LDED   S1ICNT    ;GET SERIAL 1 INPUT COUNT
02A5'   13                   INX    D         ;INCREMENT SERIAL 1 INPUT COUNT
02A6'   B7                   ORA    A         ;CLEAR CARRY FLAG
02A7'   ED52                 DSBC   D         ;SERIAL 1 INPUT BUFFER FULL?
02A9'   D8                   RC               ;IF SO, DONE
02AA'   ED53 0024"           SDED   S1ICNT    ;ELSE, UPDATE SERIAL 1 INPUT COUNT
02AE'   2A 0020"             LHLD   S1IPTR    ;GET SERIAL 1 INPUT POINTER
02B1'   71                   MOV    M,C       ;STORE INPUT CHARACTER IN BUFFER
02B2'   23                   INX    H         ;INCREMENT INPUT POINTER
02B3'   EB                   XCHG             ;DE=INPUT POINTER/HL=BUFFER SIZE
02B4'   2A 001C"             LHLD   S1IBSZ    ;GET SERIAL 1 INPUT BUFFER SIZE
02B7'   2B                   DCX    H         ;DECREMENT INPUT BUFFER SIZE
02B8'   ED4B 001E"           LBCD   S1IBUF    ;GET SERIAL 1 INPUT BUFFER ADDRESS
02BC'   09                   DAD    B         ;CALC LAST INPUT BUFFER ADDRESS
02BD'   ED52                 DSBC   D         ;BUFFER WRAP-AROUND?
02BF'   3002                 JRNC   ..NWA1    ;IF NOT, CONTINUE
02C1'   59                   MOV    E,C       ;GET SERIAL 1 INPUT BUFFER ADDRESS
02C2'   50                   MOV    D,B
02C3'   ED53 0020"   ..NWA1: SDED   S1IPTR    ;UPDATE SERIAL 1 INPUT POINTER
02C7'   11 0026"             LXI    D,S1IWCT   ;GET SERIAL 1 INPUT WAIT COUNT
02CA'   21 002A"             LXI    H,S1ISPH   ;GET SERIAL 1 INPUT SEMAPHORE
02CD'   CD 02D2'             CALL   ..SIGC    ;SIGNAL IF NECESSARY
02D0'   1884                 JMPR   ..S1I     ;CONTINUE
```

SPDMAS - SAMPLE SERIAL/PARALLEL DRIVER MODULE
FOR TURBODOS ON S-100 MASTER Z80 PROCESSOR

```
02D2'   1A              ..SIGC: LDAX    D       ;GET SERIAL INPUT WAIT COUNT
02D3'   B7                      ORA     A       ;SERIAL INPUT WAIT COUNT=0?
02D4'   C8                      RZ              ;IF SO, DONE
02D5'   3D                      DCR     A       ;DECREMENT SERIAL INPUT WAIT COUI
02D6'   12                      STAX    D       ;UPDATE SERIAL INPUT WAIT COUNT
02D7'   C3 0000:0A              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
02DA'   78              SERSBR: MOV     A,B     ;GET CHANNEL NUMBER
02DB'   21 000F"                LXI     H,SOBR  ;GET SERIAL 0 BAUD RATE CODE
02DE'   B7                      ORA     A       ;CHANNEL NUMBER=0?
02DF'   2803                    JRZ     ..COM1  ;IF SO, CONTINUE
02E1'   21 0029"                LXI     H,S1BR  ;ELSE, GET SERIAL 1 BAUD RATE CO
02E4'   71              ..COM1: MOV     M,C     ;SAVE BAUD RATE CODE
02E5'   79                      MOV     A,C     ;GET REQUESTED BAUD RATE CODE
02E6'   E60F                    ANI     0FH     ;EXTRACT RELEVANT BITS
02E8'   4F                      MOV     C,A     ;UPDATE REQUESTED BAUD RATE CODE
02E9'   78                      MOV     A,B     ;GET CHANNEL NUMBER
02EA'   B7                      ORA     A       ;CHANNEL NUMBER=0?
02EB'   3A 0001"                LDA     BAUDRT  ;GET BAUD RATE REGISTER VALUE
02EE'   280B                    JRZ     ..CHO   ;IF CHANNEL NUMBER=0, CONTINUE
02F0'   E60F                    ANI     0FH     ;ELSE, STRIP UPPER FOUR BITS
02F2'   47                      MOV     B,A     ;BAUD RATE REGISTER VALUE TO B-R
02F3'   79                      MOV     A,C     ;GET REQUESTED BAUD RATE CODE
02F4'   87                      ADD     A       ;SHIFT BAUD RATE CODE TO MSN
02F5'   87                      ADD     A
02F6'   87                      ADD     A
02F7'   87                      ADD     A
02F8'   B0                      ORA     B       ;COMBINE WITH BAUD RATE REGISTEF
02F9'   1803                    JMPR    ..COM2  ;CONTINUE
02FB'   E6F0            ..CHO:  ANI     0F0H    ;STRIP LOWER FOUR BITS
02FD'   B1                      ORA     C       ;COMBINE WITH BAUD RATE REGISTEI
02FE'   D318            ..COM2: OUT     SIOBRR  ;SET BAUD RATE GENERATOR REGISTI
0300'   32 0001"                STA     BAUDRT  ;UPDATE BAUD RATE REGISTER VALUI
0303'   C9                      RET             ;DONE
                        ;
0304'   21 000F"        SERRBR: LXI     H,SOBR  ;GET SERIAL 0 BAUD RATE
0307'   78                      MOV     A,B     ;GET CHANNEL NUMBER
0308'   B7                      ORA     A       ;CHANNEL NUMBER=0?
0309'   2803                    JRZ     ..COM   ;IF SO, CONTINUE
030B'   21 0029"                LXI     H,S1BR  ;ELSE, GET SERIAL 1 BAUD RATE
030E'   7E              ..COM:  MOV     A,M     ;GET CURRENT BAUD RATE CODE
030F'   C9                      RET             ;DONE
                        ;
0310'   3EEA            SERSMC: MVI     A,0EAH  ;GET WRITE REGISTER 5 CONTROL W
0312'   E67D                    ANI     #82H    ;STRIP RTS/CTS CONTROL BITS
0314'   CB79                    BIT     7,C     ;RTS REQUESTED?
0316'   2802                    JRZ     ..NRTS
0318'   CBCF                    SET     1,A     ;IF SO, SET RTS BIT
031A'   CB71            ..NRTS: BIT     6,C     ;DTR REQUESTED?
031C'   2802                    JRZ     ..NDTR
031E'   CBFF                    SET     7,A     ;IF SO, SET DTR BIT
0320'   57              ..NDTR: MOV     D,A     ;REQUESTED MODEM CONTROLS TO D-
0321'   0E01                    MVI     C,SIOACR  ;GET SIO PORT A CONTROL REGIS
0323'   78                      MOV     A,B     ;GET CHANNEL NUMBER
```

```
  0324'   B7                   ORA     A       ;CHANNEL NUMBER=0?
  0325'   2802                 JRZ     ..COM   ;IF SO, CONTINUE
  0327'   0E03                 MVI     C,SIOBCR  ;GET SIO PORT B CONTROL REGISTER
  0329'   3E05         ..COM:  MVI     A,5     ;GET WRITE REGISTER 5
  032B'   F3                   DI              ;DISABLE INTERRUPTS
  032C'   ED79                 OUTP    A       ;SELECT WRITE REGISTER 5
  032E'   ED51                 OUTP    D       ;OUTPUT CONTROL WORD
  0330'   FB                   EI              ;ENABLE INTERRUPTS
  0331'   C9                   RET             ;DONE
                       ;
  0332'   0E01         SERRMC: MVI     C,SIOACR  ;GET SIO PORT A CONTROL REGISTER
  0334'   78                   MOV     A,B     ;GET CHANNEL NUMBER
  0335'   B7                   ORA     A       ;CHANNEL NUMBER=0?
  0336'   2802                 JRZ     ..COM   ;IF SO, CONTINUE
  0338'   0E03                 MVI     C,SIOBCR  ;GET SIO PORT B CONTROL REGISTER
  033A'   3E10         ..COM:  MVI     A,10H   ;GET RESET EXTERNAL STATUS COMMAND
  033C'   ED79                 OUTP    A       ;RESET EXTERNAL STATUS
  033E'   ED50                 INP     D       ;GET SIO MODEM STATUS
  0340'   AF                   XRA     A       ;CLEAR RETURN VECTOR
  0341'   CB6A                 BIT     CTS,D   ;CTS SET?
  0343'   2802                 JRZ     ..NCTS  ;IF NOT, CONTINUE
  0345'   CBFF                 SET     7,A     ;ELSE, SET CTS BIT
  0347'   CB5A         ..NCTS: BIT     DCD,D   ;DCD SET?
  0349'   C8                   RZ              ;IF NOT, DONE
  034A'   CBEF                 SET     5,A     ;ELSE, SET DCD BIT
  034C'   C9                   RET             ;DONE
                       ;
                       .PRGEND
```

```
RTCMAS - SAMPLE REAL-TIME CLOCK DRIVER MODULE
FOR TURBODOS ON S-100 MASTER Z80 PROCESSOR


                        ;
                        ; VERSION: 01/05/84
                        ;
                        .IDENT  RTCMAS          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
0008                    CTCCH0  = 08H           ;CTC CHANNEL 0 REGISTER
0009                    CTCCH1  = 09H           ;CTC CHANNEL 1 REGISTER
                        ;
0000"                           .LOC    .DATA.# ;LOCATE IN DATA AREA
                        ;
0000"   00              TICCTR: .BYTE   0       ;TICK COUNTER
                        ;
0000:04                         .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                        ;
0000:04 21 0000'        RTCNIT::LXI     H,RTCISR  ;GET INTERRUPT SERVICE ADDRESS
0003:04 22 0002:05              SHLD    CTCVEC#+2  ;SET INTERRUPT SERVICE VECTOR
0006:04 3E47                    MVI     A,47H   ;GET CTC CHANNEL 0 CONTROL WORD
0008:04 D308                    OUT     CTCCH0  ;INITIALIZE CTC CHANNEL 0
000A:04 3EFA                    MVI     A,250   ;GET TIME CONSTANT VALUE
000C:04 D308                    OUT     CTCCH0  ;SET CTC CHANNEL 0 TIME CONSTANT
000E:04 3EC7                    MVI     A,0C7H  ;GET CTC CHANNEL 1 CONTROL WORD
0010:04 D309                    OUT     CTCCH1  ;INITIALIZE CTC CHANNEL 1
0012:04 3E64                    MVI     A,100   ;GET TIME CONSTANT VALUE
0014:04 D309                    OUT     CTCCH1  ;SET CTC CHANNEL 1 TIME CONSTANT
0016:04 C9                      RET             ;DONE
                        ;
0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
0000'   ED73 0000:06    RTCISR::SSPD    INTSP#  ;SAVE STACK POINTER
0004'   31 0000:07              LXI     SP,INTSTK#  ;SET UP AUX STACK POINTER
0007'   F5                      PUSH    PSW     ;SAVE REGISTERS
0008'   C5                      PUSH    B
0009'   D5                      PUSH    D
000A'   E5                      PUSH    H
000B'   21 0000"                LXI     H,TICCTR  ;GET TICK COUNTER
000E'   34                      INR     M       ;INCREMENT TICK COUNTER
000F'   7E                      MOV     A,M     ;GET TICK COUNT
0010'   FE3C                    CPI     60      ;SECONDS COUNT REACHED?
0012'   3805                    JRC     ..NSEC  ;IF NOT, CONTINUE
0014'   3600                    MVI     M,0     ;ELSE, RESET TICK COUNTER
0016'   CD 0000:08              CALL    RTCSEC# ;SERVICE REAL TIME CLOCK MANAGER
0019'   CD 0000:09      ..NSEC: CALL    DLYTIC# ;SERVICE DISPATCHER DELAY MANAGER
001C'   E1                      POP     H       ;RESTORE REGISTERS
001D'   D1                      POP     D
001E'   C1                      POP     B
001F'   F1                      POP     PSW
0020'   ED7B 0000:06            LSPD    INTSP#  ;RESTORE STACK POINTER
0024'   C3 0000:0A              JMP     ISRXIT# ;CONTINUE
                        ;
                        .PRGEND
```

```
                          ;
                          ; VERSION: 01/05/84
                          ;
                          .IDENT  DSKFDC          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
 000B                     CTCCH3  = 0BH           ;CTC CHANNEL 3 REGISTER
                          ;
 000C                     FDCCSR  = 0CH           ;FDC COMMAND/STATUS REGISTER
 000D                     FDCTRK  = 0DH           ;FDC TRACK REGISTER
 000E                     FDCSEC  = 0EH           ;FDC SECTOR REGISTER
 000F                     FDCDAT  = 0FH           ;FDC DATA REGISTER
                          ;
 0010                     DMACTL  = 10H           ;DMA CONTROL REGISTER
                          ;
 0014                     FDCDSR  = 14H           ;FDC DRIVE SELECT REGISTER
 0015                     TSSOJR  = 15H           ;TWO-SIDED STATUS/OPTION JUMPER REG
                          ;
 0008                     FDCCAL  = 08H           ;FDC RE-CALIBRATE COMMAND
 0010                     FDCSKN  = 10H           ;FDC SEEK COMMAND WITHOUT HEAD LOAD
 0018                     FDCSKH  = 18H           ;FDC SEEK COMMAND WITH HEAD LOAD
 0082                     FDCRDC  = 82H           ;FDC READ SECTOR COMMAND
 00A2                     FDCWRC  = 0A2H          ;FDC WRITE SECTOR COMMAND
 00C0                     FDCRID  = 0C0H          ;FDC READ ID COMMAND
 00D0                     FDCINT  = 0D0H          ;FDC INTERRUPT COMMAND
 00F0                     FDCFMT  = 0F0H          ;FDC FORMAT TRACK COMMAND
                          ;
 0002                     HSDBIT  = 2             ;HEAD SETTLE DELAY BIT
                          ;
 0001                     DMARDC  = 01H           ;DMA READ COMMAND
 0005                     DMAWRC  = 05H           ;DMA WRITE COMMAND
                          ;
 0002                     TSD     = 2             ;TWO SIDED DISK BIT
 0003                     DDD     = 3             ;DOUBLE DENSITY DISK BIT
 0004                     MINI    = 4             ;MINI-FLOPPY DISK BIT
 0005                     TPI96   = 5             ;96-TPI DISK BIT
                          ;
 000A                     MAXTRY  = 10            ;MAX DISK TRY COUNT
                          ;
 0000"                            .LOC    .DATA.# ;LOCATE IN DATA AREA
                          ;
 0000"    00001030        DRVTBL::.BYTE   0,0,1<MINI,1<MINI!1<TPI96  ;DRIVE TABLE
                          ;
 0004"    FF              DRIVE:  .BYTE   0FFH    ;DRIVE NUMBER
 0005"    00              TRACK:  .BYTE   0       ;TRACK NUMBER
 0006"    00              SECTOR: .BYTE   0       ;SECTOR NUMBER
 0007"    00              SECCNT: .BYTE   0       ;SECTOR COUNT
 0008"    00              TRYCNT: .BYTE   0       ;TRY COUNTER
 0009"    00              DLYBIT: .BYTE   0       ;HEAD SETTLE DELAY BIT
 000A"    00              INTCST: .BYTE   0       ;INTERRUPT COMPLETION STATUS
 000B"    00              RWERRS: .BYTE   0       ;READ/WRITE ERROR STATUS
 000C"    00              DSRSAV: .BYTE   0       ;DRIVE SELECT REGISTER SAVE
 000D"    00              NDXCNT: .BYTE   0       ;INDEX PULSE SEQUENCE COUNT
```

```
000E"   00             NDXTIC: .BYTE   0        ;INDEX PULSE TICK COUNT
000F"   0000           RETSP:  .WORD   0        ;ERROR RETURN STACK POINTER
0011"   FFFFFFFF       TRKTBL: .BYTE   [4]0FFH  ;TRACK SAVE TABLE
0015"   000000000000   RIDBUF: .BYTE   [6]0     ;READ ID BUFFER
                       ;
001B"                  DMXSPH:                  ;MUTUAL EXCLUSION SEMAPHORE
001B"   0001                   .WORD   1        ;SEMAPHORE COUNT
001D"   001D"          ..DMXH: .WORD   ..DMXH   ;SEMAPHORE P/D HEAD
001F"   001D"                  .WORD   ..DMXH
                       ;
0021"                  DWTSPH:                  ;DISK WAIT SEMAPHORE
0021"   0000 .                 .WORD   0        ;SEMAPHORE COUNT
0023"   0023"          ..DWTH: .WORD   ..DWTH   ;SEMAPHORE P/D HEAD
0025"   0023"                  .WORD   ..DWTH
                       ;
0027"                  DMAPGM:                  ;DMA CONTROLLER PROGRAM LIST
0027"   C3                     .BYTE   0C3H     ;WRITE REGISTER 6
0028"   8B                     .BYTE   08BH     ;WRITE REGISTER 6
0029"   79                     .BYTE   79H      ;WRITE REGISTER 0
002A"   0000           DMAADR: .WORD   0        ;DMA ADDRESS
002C"   0000           DMALEN: .WORD   0        ;DMA LENGTH
002E"   14                     .BYTE   14H      ;WRITE REGISTER 1
002F"   28                     .BYTE   28H      ;WRITE REGISTER 2
0030"   85                     .BYTE   85H      ;WRITE REGISTER 4
0031"   0F                     .BYTE   FDCDAT   ;FDC DATA PORT ADDRESS
0032"   8A                     .BYTE   8AH      ;WRITE REGISTER 5
0033"   CF                     .BYTE   0CFH     ;WRITE REGISTER 6
0034"   05             DMARWC: .BYTE   05H      ;DMA READ/WRITE COMMAND
0035"   CF                     .BYTE   0CFH     ;WRITE REGISTER 6
0036"   87                     .BYTE   87H      ;WRITE REGISTER 6
                       ;
0010                   DMAPLL  = .-DMAPGM       ;DMA CONTROLLER PROGRAM LIST LENGTH
                       ;
0000:04                        .LOC    .INIT.#  ;LOCATE IN INITIALIZATION AREA
                       ;
0000:04 CD 038F'     . DSKIN%::CALL    CLRFDC  ;CLEAR FDC
0003:04 21 039E'               LXI     H,DSKISR  ;GET INTERRUPT SERVICE ROUTINE
0006:04 22 0006:05             SHLD    CTCVEC#+6  ;SET INTERRUPT VECTOR ADDRESS
0009:04 C9                     RET              ;DONE
                       ;
0000'                          .LOC    .PROG.#  ;LOCATE IN PROGRAM AREA
                       ;
0000'   21 001B"       DSKDR%::LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
0003'   CD 0000:06             CALL    WAIT#    ;DISPATCH IF NECESSARY
0006'   CD 0012'               CALL    ..DD     ;CALL DISK DRIVER
0009'   F5                     PUSH    PSW      ;SAVE RETURN CODE
000A'   21 001B"               LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
000D'   CD 0000:07             CALL    SIGNAL#  ;SIGNAL PROCESS AS READY
0010'   F1                     POP     PSW      ;RESTORE RETURN CODE
0011'   C9                     RET              ;DONE
                       ;
0012'   ED73 000F"     ..DD:   SSPD    RETSP    ;SAVE ERROR RETURN STACK POINTER
0016'   DD7E00                 MOV     A,PDRFCN(X)  ;GET FUNCTION NUMBER
0019'   B7                     ORA     A        ;FUNCTION NUMBER=0?
```

```
001A'   280F                  JRZ     RDDSK   ;IF SO, CONTINUE
001C'   3D                    DCR     A       ;FUNCTION NUMBER=1?
001D'   2833                  JRZ     WRDSK   ;IF SO, CONTINUE
001F'   3D                    DCR     A       ;FUNCTION NUMBER=2?
0020'   CA 01EB'              JZ      RETDST  ;IF SO, CONTINUE
0023'   3D                    DCR     A       ;FUNCTION NUMBER=3?
0024'   CA 02A6'              JZ      RETRDY  ;IF SO, CONTINUE
0027'   3D                    DCR     A       ;FUNCTION NUMBER=4?
0028'   286D                  JRZ     FMTDSK  ;IF SO, CONTINUE
002A'   C9                    RET             ;ELSE, DONE
                      ;
002B'   3E0A          RDDSK:  MVI     A,MAXTRY  ;GET MAX TRY COUNT
002D'   32 0008"              STA     TRYCNT  ;SET TRY COUNTER
0030'   CD 00EA'      ..RD:   CALL    SETUP   ;DO COMMON SETUP
0033'   2018                  JRNZ    ..ERR   ;IF SEEK ERROR, CONTINUE
0035'   CD 0153'      ..RDL:  CALL    RWCOM1  ;ELSE, DO READ/WRITE COMMON #1
0038'   11 829D               LXI     D,FDCRDC<8!9DH  ;GET FDC READ COMMAND/MASK
003B'   CD 0190'              CALL    RWCOM2  ;DO READ/WRITE COMMON #2
003E'   3E01                  MVI     A,DMARDC  ;GET DMA READ COMMAND
0040'   CD 0355'              CALL    DMACOM  ;DO DMA COMMON
0043'   CD 0199'              CALL    RWCOM3  ;DO READ/WRITE COMMON #3
0046'   20ED                  JRNZ    ..RDL   ;IF NOT LAST SECTOR, CONTINUE
0048'   3A 000B"              LDA     RWERRS  ;ELSE, GET READ/WRITE ERROR STATUS
004B'   B7                    ORA     A       ;READ/WRITE ERROR STATUS=0?
004C'   C8                    RZ              ;IF SO, DONE
004D'   CD 01B3'      ..ERR:  CALL    RETRY   ;ELSE, RE-CALIBRATE DRIVE
0050'   18DE                  JMPR    ..RD    ;TRY AGAIN
                      ;
0052'   3E0A          WRDSK:  MVI     A,MAXTRY  ;GET MAX TRY COUNT
0054'   32 0008"              STA     TRYCNT  ;SET TRY COUNTER
0057'   CD 00EA'      ..WR:   CALL    SETUP   ;DO COMMON SETUP
005A'   2036                  JRNZ    ..ERR   ;IF SEEK ERROR, CONTINUE
005C'   CD 0153'      ..WRL:  CALL    RWCOM1  ;ELSE, DO READ/WRITE COMMON #1
005F'   11 A2FD               LXI     D,FDCWRC<8!0FDH  ;GET FDC READ COMMAND/MASK
0062'   CD 0190'              CALL    RWCOM2  ;DO READ/WRITE COMMON #2
0065'   3E05                  MVI     A,DMAWRC  ;GET DMA WRITE COMMAND
0067'   CD 0355'              CALL    DMACOM  ;DO DMA COMMON
006A'   CD 0199'              CALL    RWCOM3  ;DO READ/WRITE COMMON #3
006D'   20ED                  JRNZ    ..WRL   ;IF NOT LAST SECTOR, CONTINUE
006F'   3A 000B"              LDA     RWERRS  ;ELSE, GET READ/WRITE ERROR STATUS
0072'   B7                    ORA     A       ;READ/WRITE ERROR STATUS=0?
0073'   201D                  JRNZ    ..ERR   ;IF NOT, CONTINUE
0075'   CD 00EA'              CALL    SETUP   ;ELSE, DO COMMON SETUP
0078'   2018                  JRNZ    ..ERR   ;IF SEEK ERROR, CONTINUE
007A'   CD 0153'      ..VFL:  CALL    RWCOM1  ;ELSE, DO READ/WRITE COMMON #1
007D'   1682                  MVI     D,FDCRDC  ;GET FDC READ COMMAND
007F'   CD 0190'              CALL    RWCOM2  ;DO READ/WRITE COMMON #2
0082'   7A                    MOV     A,D     ;GET FDC READ COMMAND
0083'   CD 037E'              CALL    FDCCMD  ;OUTPUT FDC READ COMMAND
0086'   E699                  ANI     99H     ;EXTRACT RELEVANT STATUS BITS
0088'   CD 0199'              CALL    RWCOM3  ;DO READ/WRITE COMMON #3
008B'   20ED                  JRNZ    ..VFL   ;IF NOT LAST SECTOR, CONTINUE
008D'   3A 000B"              LDA     RWERRS  ;ELSE, GET READ/WRITE ERROR STATUS
0090'   B7                    ORA     A       ;READ/WRITE ERROR STATUS=0?
```

```
0091'   C8                      RZ              ;IF SO, DONE
0092'   CD 01B3'        ..ERR:  CALL    RETRY   ;ELSE, RE-CALIBRATE DRIVE
0095'   18C0                    JMPR    ..WR    ;TRY AGAIN
                        ;
0097'   3E0A            FMTDSK: MVI     A,MAXTRY  ;GET MAX TRY COUNT
0099'   32 0008"                STA     TRYCNT  ;SET TRY COUNTER
009C'   CD 02DF'        ..FMT:  CALL    SELDSK  ;SELECT DISK
009F'   C2 01E4'                JNZ     FATAL   ;IF DRIVE NOT READY, CONTINUE
00A2'   3A 000C"                LDA     DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
00A5'   DDCB047E                BIT     7,PDRSEC(X)  ;DOUBLE DENSITY REQUESTED?
00A9'   2002                    JRNZ    ..DDR   ;IF SO, CONTINUE
00AB'   CB9F                    RES     3,A     ;ELSE, RESET DOUBLE DENSITY BIT
00AD'   DDCB057E        ..DDR:  BIT     7,PDRSEC+1(X)  ;SIDE ONE REQUESTED?
00B1'   2802                    JRZ     ..S1NR  ;IF NOT, CONTINUE
00B3'   CBD7                    SET     2,A     ;ELSE, SET SIDE ONE SELECT BIT
00B5'   D314            ..S1NR: OUT     FDCDSR  ;SELECT DRIVE/SIDE/DENSITY
00B7'   32 000C"                STA     DSRSAV  ;SAVE DRIVE SELECT REGISTER VALUE
00BA'   DD7E02                  MOV     A,PDRTRK(X)  ;GET REQUESTED TRACK NUMBER
00BD'   32 0005"                STA     TRACK   ;SET TRACK NUMBER
00C0'   B7                      ORA     A       ;REQUESTED TRACK NUMBER=0?
00C1'   CC 0292'                CZ      RECAL   ;IF SO, RE-CALIBRATE DRIVE
00C4'   CD 026A'                CALL    SEEK    ;SEEK TO REQUESTED TRACK
00C7'   201C                    JRNZ    ..ERR   ;IF SEEK ERROR, CONTINUE
00C9'   DD6E0A                  MOV     L,PDRDMA(X)  ;GET REQUESTED DMA ADDRESS
00CC'   DD660B                  MOV     H,PDRDMA+1(X)
00CF'   22 002A"                SHLD    DMAADR  ;SET DMA ADDRESS
00D2'   DD6E08                  MOV     L,PDRTC(X)  ;GET REQUESTED TRANSFER COUNT
00D5'   DD6609                  MOV     H,PDRTC+1(X)
00D8'   2B                      DCX     H       ;DECREMENT REQUESTED TRANSFER COUNT
00D9'   22 002C"                SHLD    DMALEN  ;SET DMA LENGTH
00DC'   11 F0E1                 LXI     D,FDCFMT<8!0E1H  ;GET FORMAT COMMAND/MASK
00DF'   3E05                    MVI     A,DMAWRC  ;GET DMA WRITE COMMAND
00E1'   CD 0355'                CALL    DMACOM  ;DO DMA COMMON
00E4'   C8                      RZ              ;IF NO ERRORS, DONE
00E5'   CD 01B3'        ..ERR:  CALL    RETRY   ;ELSE, RE-CALIBRATE DRIVE
00E8'   18B2                    JMPR    ..FMT   ;TRY AGAIN
                        ;
00EA'   DD7E02          SETUP:  MOV     A,PDRTRK(X)  ;GET REQUESTED TRACK NUMBER
00ED'   32 0005"                STA     TRACK   ;SET TRACK NUMBER
00F0'   DD7E04                  MOV     A,PDRSEC(X)  ;GET REQUESTED SECTOR NUMBER
00F3'   32 0006"                STA     SECTOR  ;SET SECTOR NUMBER
00F6'   CD 0403'                CALL    GETTCA  ;GET DISK TYPE CODE ADDRESS
00F9'   7E                      MOV     A,M     ;GET DISK TYPE CODE
00FA'   CB97                    RES     TSD,A   ;RESET TWO-SIDED DISK BIT
00FC'   FE1A                    CPI     1<MINI!1<DDD!2  ;IBM PC MINI-FLOPPY FORMAT?
00FE'   2028                    JRNZ    ..NIPC  ;IF NOT, CONTINUE
0100'   DD7E02                  MOV     A,PDRTRK(X)  ;ELSE, GET LSB OF TRACK NUMBER
0103'   E60F                    ANI     0FH     ;EXTRACT LOWER FOUR BITS
0105'   CB56                    BIT     TSD,M   ;TWO-SIDED DISK?
0107'   2002                    JRNZ    ..TSD   ;IF SO, CONTINUE
0109'   E607                    ANI     07H     ;ELSE, EXTRACT LOWER THREE BITS
010B'   32 0006"        ..TSD:  STA     SECTOR  ;SET SECTOR NUMBER
010E'   DD5E02                  MOV     E,PDRTRK(X)  ;GET REQUESTED TRACK NUMBER
0111'   DD5603                  MOV     D,PDRTRK+1(X)
```

```
0114'   0603                  MVI    B,3      ;GET SHIFT COUNT
0116'   CB3A         ..SL1:   SRLR   D        ;SHIFT REQUESTED TRACK NUMBER RIGHT
0118'   CB1B                  RARR   E
011A'   10FA                  DJNZ   ..SL1    ;THREE TIMES
011C'   CB56                  BIT    TSD,M    ;TWO-SIDED DISK?
011E'   2804                  JRZ    ..NTSD   ;IF NOT, CONTINUE
0120'   CB3A                  SRLR   D        ;SHIFT REQUESTED TRACK NUMBER RIGHT
0122'   CB1B                  RARR   E
0124'   7B           ..NTSD:  MOV    A,E      ;GET REQUESTED TRACK NUMBER
0125'   32 0005"              STA    TRACK    ;SET TRACK NUMBER
0128'   DD7E06       ..NIPC:  MOV    A,PDRSC(X)   ;GET REQUESTED SECTOR COUNT
012B'   32 0007"              STA    SECCNT   ;SET SECTOR COUNT
012E'   DD6E0A                MOV    L,PDRDMA(X)  ;GET REQUESTED DMA ADDRESS
0131'   DD660B                MOV    H,PDRDMA+1(X)
0134'   22 002A"              SHLD   DMAADR   ;SET DMA ADDRESS
0137'   DD4612                MOV    B,SECSIZ(X)  ;GET SECTOR SIZE
013A'   04                    INR    B        ;INCREMENT SECTOR SIZE
013B'   21 0040               LXI    H,128/2  ;GET SECTOR SIZE=0 (/2)
013E'   29           ..SL2:   DAD    H        ;SHIFT SECTOR SIZE LEFT
013F'   10FD                  DJNZ   ..SL2    ;SECTOR SIZE TIMES
0141'   2B                    DCX    H        ;DECREMENT SECTOR SIZE
0142'   22 002C"              SHLD   DMALEN   ;SET DMA LENGTH
0145'   AF                    XRA    A
0146'   32 000B"              STA    RWERRS   ;SET READ/WRITE ERROR STATUS=0
0149'   CD 02DF'              CALL   SELDSK   ;SELECT DISK
014C'   C2 01E4'              JNZ    FATAL    ;IF DRIVE NOT READY, CONTINUE
014F'   CD 026A'              CALL   SEEK     ;ELSE, SEEK TO REQUESTED TRACK
0152'   C9                    RET             ;DONE
                     ;
0153'   CD 0403'     RWCOM1:  CALL   GETTCA   ;GET DISK TYPE CODE ADDRESS
0156'   46                    MOV    B,M      ;GET DISK TYPE CODE
0157'   21 000C"              LXI    H,DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
015A'   CB58                  BIT    DDD,B    ;DOUBLE DENSITY DISK?
015C'   2002                  JRNZ   ..DDD    ;IF SO, CONTINUE
015E'   CB9E                  RES    3,M      ;ELSE, RESET DOUBLE DENSITY BIT
0160'   3A 0006"     ..DDD:   LDA    SECTOR   ;GET SECTOR NUMBER
0163'   CB50                  BIT    TSD,B    ;TWO SIDED DISK?
0165'   2815                  JRZ    ..NTSD   ;IF NOT, CONTINUE
0167'   78                    MOV    A,B      ;ELSE, GET DISK TYPE CODE
0168'   FE1E                  CPI    1<MINI!1<DDD!1<TSD!2  ;IBM PC FORMAT?
016A'   3A 0006"              LDA    SECTOR   ;GET SECTOR NUMBER
016D'   0E10                  MVI    C,16     ;PRESET NUMBER OF SECTORS/TRACK=16
016F'   2803                  JRZ    ..IPCF   ;IF IBM PC FORMAT, CONTINUE
0171'   DD4E13                MOV    C,SECTRK(X)  ;GET NUMBER OF SECTORS/TRACK
0174'   CB39         ..IPCF:  SRLR   C        ;CALC NUMBER OF SECTORS/SIDE
0176'   B9                    CMP    C        ;REQUESTED SECTOR ON SIDE ONE?
0177'   3803                  JRC    ..NTSD   ;IF NOT, CONTINUE
0179'   91                    SUB    C        ;ELSE, ADJUST SECTOR NUMBER
017A'   CBD6                  SET    2,M      ;SET SIDE ONE SELECT BIT
017C'   4F           ..NTSD:  MOV    C,A      ;SECTOR NUMBER TO C-REG
017D'   CD 03F5'              CALL   GETXLT   ;GET TRANSLATION TABLE ADDRESS
0180'   2804                  JRZ    ..NSTR   ;IF NO SECTOR TRANSLATION, CONTINUE
0182'   0600                  MVI    B,0      ;MAKE SECTOR NUMBER DOUBLE LENGTH
0184'   09                    DAD    B        ;INDEX INTO TRANSLATION TABLE
```

```
0185'   4E                      MOV     C,M       ;GET TRANSLATED SECTOR NUMBER
0186'   79              ..NSTR: MOV     A,C       ;GET SECTOR NUMBER
0187'   3C                      INR     A         ;INCREMENT SECTOR NUMBER TO BASE 1
0188'   D30E                    OUT     FDCSEC    ;SET FDC SECTOR REGISTER
018A'   3A 000C"                LDA     DSRSAV    ;GET SAVED DRIVE SELECT REGISTER
018D'   D314                    OUT     FDCDSR    ;SELECT DRIVE/SIDE/DENSITY
018F'   C9                      RET               ;DONE
                        ;
0190'   3A 000C"        RWCOM2: LDA     DSRSAV    ;GET SAVED DRIVE SELECT REGISTER
0193'   CB57                    BIT     2,A       ;SIDE ONE SELECTED?
0195'   C8                      RZ                ;IF NOT, DONE
0196'   CBDA                    SET     3,D       ;ELSE, SET SIDE ONE VERIFY BIT
0198'   C9                      RET               ;DONE
                        ;
0199'   21 000B"        RWCOM3: LXI     H,RWERRS  ;GET READ/WRITE ERROR STATUS
019C'   B6                      ORA     M         ;COMBINE WITH COMPLETION STATUS
019D'   77                      MOV     M,A       ;UPDATE READ/WRITE ERROR STATUS
019E'   2A 002A"                LHLD    DMAADR    ;GET DMA ADDRESS
01A1'   ED5B 002C"              LDED    DMALEN    ;GET DMA LENGTH
01A5'   13                      INX     D         ;INCREMENT DMA LENGTH
01A6'   19                      DAD     D         ;CALC NEXT DMA ADDRESS
01A7'   22 002A"                SHLD    DMAADR    ;UPDATE DMA ADDRESS
01AA'   21 0006"                LXI     H,SECTOR  ;GET SECTOR NUMBER
01AD'   34                      INR     M         ;INCREMENT SECTOR NUMBER
01AE'   21 0007"                LXI     H,SECCNT  ;GET SECTOR COUNT
01B1'   35                      DCR     M         ;DECREMENT SECTOR COUNT
01B2'   C9                      RET               ;DONE
                        ;
01B3'   0E07            RETRY:  MVI     C,ABEL    ;GET BELL CHARACTER
01B5'   CD 0000:08              CALL    CONOUT#   ;OUTPUT BELL CHARACTER TO CONSOLE
01B8'   3A 0008"                LDA     TRYCNT    ;GET TRY COUNTER
01BB'   E601                    ANI     01H       ;EVEN TRY?
01BD'   C4 0292'                CNZ     RECAL     ;IF NOT, RE-CALIBRATE DRIVE
01C0'   3A 0008"                LDA     TRYCNT    ;GET TRY COUNTER
01C3'   F5                      PUSH    PSW       ;SAVE TRY COUNTER
01C4'   2A 000F"                LHLD    RETSP     ;GET ERROR RETURN STACK POINTER
01C7'   E5                      PUSH    H         ;SAVE ERROR RETURN STACK POINTER
01C8'   21 001B"                LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
01CB'   CD 0000:07              CALL    SIGNAL#   ;SIGNAL PROCESS AS READY
01CE'   21 0000                 LXI     H,0       ;SET DELAY COUNT=0
01D1'   CD 0000:09              CALL    DELAY#    ;DISPATCH
01D4'   21 001B"                LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
01D7'   CD 0000:06              CALL    WAIT#     ;DISPATCH IF NECESSARY
01DA'   E1                      POP     H         ;RESTORE ERROR RETURN STACK POINTER
01DB'   22 000F"                SHLD    RETSP     ;SET ERROR RETURN STACK POINTER
01DE'   F1                      POP     PSW       ;RESTORE TRY COUNTER
01DF'   3D                      DCR     A         ;DECREMENT TRY COUNTER
01E0'   32 0008"                STA     TRYCNT    ;UPDATE TRY COUNTER
01E3'   C0                      RNZ               ;IF COUNT NOT EXHAUSTED, DONE
                        ;
01E4'   ED7B 000F"      FATAL:  LSPD    RETSP     ;RESTORE STACK POINTER
01E8'   3EFF                    MVI     A,0FFH    ;RETURN ERROR CODE
01EA'   C9                      RET               ;DONE
                        ;
```

```
01EB'   CD 02A6'      RETDST: CALL    RETRDY  ;RETURN READY STATUS
01EE'   B7                    ORA     A       ;DRIVE READY?
01EF'   C8                    RZ              ;IF NOT, DONE
01F0'   DB0D                  IN      FDCTRK  ;ELSE, GET FDC TRACK REGISTER
01F2'   3C                    INR     A       ;FDC TRACK REGISTER=0FFH?
01F3'   CC 0292'              CZ      RECAL   ;IF SO, RE-CALIBRATE DRIVE
01F6'   CD 0329'              CALL    READID  ;READ SECTOR ID
01F9'   2062                  JRNZ    ..NRDY  ;IF READ UNSUCCESSFUL, CONTINUE
01FB'   3A 0018"              LDA     RIDBUF+3  ;ELSE, GET SECTOR SIZE
01FE'   4F                    MOV     C,A     ;SECTOR SIZE TO C-REG
01FF'   3A 000C"              LDA     DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
0202'   CB5F                  BIT     3,A     ;DOUBLE DENSITY BIT SET?
0204'   2802                  JRZ     ..NDDD  ;IF NOT, CONTINUE
0206'   CBD9                  SET     DDD,C   ;ELSE, SET DOUBLE DENSITY DISK BIT
0208'   CBD7          ..NDDD: SET     2,A     ;SET SIDE ONE SELECT BIT
020A'   D314                  OUT     FDCDSR  ;SELECT REQUESTED DRIVE
020C'   32 000C"              STA     DSRSAV  ;SAVE DRIVE SELECT REGISTER VALUE
020F'   CD 038F'              CALL    CLRFDC  ;CLEAR FDC
0212'   E680                  ANI     80H     ;DRIVE READY?
0214'   202C                  JRNZ    ..NTSD  ;IF NOT, CONTINUE
0216'   CD 0412'              CALL    GETDTA  ;ELSE, GET DRIVE TABLE ADDRESS
0219'   CB66                  BIT     MINI,M  ;MINI-FLOPPY DISK?
021B'   2006                  JRNZ    ..MFD   ;IF SO, CONTINUE
021D'   DB15                  IN      TSSOJR  ;ELSE, GET TWO-SIDED DISK STATUS
021F'   E680                  ANI     80H     ;TWO SIDED DISK?
0221'   281F                  JRZ     ..NTSD  ;IF NOT, CONTINUE
0223'   C5            ..MFD:  PUSH    B       ;ELSE, SAVE DISK TYPE CODE
0224'   CD 0329'              CALL    READID  ;READ SECTOR ID
0227'   C1                    POP     B       ;RESTORE DISK TYPE CODE
0228'   2018                  JRNZ    ..NTSD  ;IF READ UNSUCCESSFUL, CONTINUE
022A'   3A 000C"              LDA     DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
022D'   A9                    XRA     C       ;COMPARE SIDE ONE/TWO DENSITIES
022E'   E608                  ANI     1<DDD
0230'   2010                  JRNZ    ..NTSD  ;IF DENSITIES DIFFERENT, CONTINUE
0232'   3A 0018"              LDA     RIDBUF+3  ;ELSE, GET SECTOR SIZE
0235'   A9                    XRA     C       ;COMPARE SIDE ONE/TWO SECTOR SIZES
0236'   E603                  ANI     3
0238'   2008                  JRNZ    ..NTSD  ;IF SIZES DIFFERENT, CONTINUE
023A'   3A 0016"              LDA     RIDBUF+1  ;ELSE, GET HEAD NUMBER
023D'   3D                    DCR     A       ;HEAD NUMBER=1?
023E'   2002                  JRNZ    ..NTSD  ;IF NOT, CONTINUE
0240'   CBD1                  SET     TSD,C   ;ELSE, SET TWO SIDED DISK BIT
0242'   CD 0412'      ..NTSD: CALL    GETDTA  ;GET DRIVE TABLE ADDRESS
0245'   7E                    MOV     A,M     ;GET DRIVE TABLE VALUE
0246'   E630                  ANI     1<MINI!1<TPI96  ;EXTRACT RELEVANT BITS
0248'   B1                    ORA     C       ;COMBINE WITH DISK TYPE CODE
0249'   4F                    MOV     C,A     ;DISK TYPE CODE TO C-REG
024A'   11 0000:0A            LXI     D,DSTBLS#  ;GET DST TABLE BASE
024D'   21 0000:0B    ..DSTL: LXI     H,DTCO# ;GET OFFSET TO DISK TYPE CODE
0250'   19                    DAD     D       ;CALC DISK TYPE CODE ADDRESS
0251'   79                    MOV     A,C     ;GET DISK TYPE CODE
0252'   BE                    CMP     M       ;DST TYPE CODE MATCH?
0253'   EB                    XCHG            ;DST ADDRESS TO HL-REG
0254'   2809                  JRZ     ..DSTF  ;IF DST FOUND, CONTINUE
```

```
0256'   5E                  MOV     E,M     ;ELSE, GET NEXT DST ADDRESS
0257'   23                  INX     H
0258'   56                  MOV     D,M
0259'   7A                  MOV     A,D
025A'   B3                  ORA     E       ;END OF DST CHAIN?
025B'   20F0                JRNZ    ..DSTL  ;IF NOT, CONTINUE
025D'   AF          ..NRDY: XRA     A       ;ELSE, SET RETURN CODE=0
025E'   C9                  RET             ;DONE
025F'   23          ..DSTF: INX     H       ;ADVANCE PAST LINK POINTER
0260'   23                  INX     H
0261'   DD750C              MOV     PDRDST(X),L  ;SET DST ADDRESS
0264'   DD740D              MOV     PDRDST+1(X),H
0267'   3EFF                MVI     A,0FFH  ;SET RETURN CODE=0FFH
0269'   C9                  RET             ;DONE
                    ;
026A'   DB0D        SEEK:   IN      FDCTRK  ;GET FDC TRACK REGISTER
026C'   3C                  INR     A       ;FDC TRACK REGISTER=0FFH?
026D'   CC 0292'            CZ      RECAL   ;IF SO, RE-CALIBRATE DRIVE
0270'   21 0005"            LXI     H,TRACK ;GET REQUESTED TRACK NUMBER
0273'   DB0D                IN      FDCTRK  ;GET FDC TRACK REGISTER
0275'   BE                  CMP     M       ;FDC TRACK=REQUESTED TRACK?
0276'   C8                  RZ              ;IF SO, DONE
0277'   3E04                MVI     A,1<HSDBIT  ;GET HEAD SETTLE DELAY BIT
0279'   32 0009"            STA     DLYBIT  ;SET HEAD SETTLE DELAY BIT
027C'   CD 038F'            CALL    CLRFDC  ;CLEAR FDC
027F'   3A 0005"            LDA     TRACK   ;GET REQUESTED TRACK NUMBER
0282'   D30F                OUT     FDCDAT  ;OUTPUT REQUESTED TRACK NUMBER
0284'   CD 0412'            CALL    GETDTA  ;GET DRIVE TABLE ADDRESS
0287'   7E                  MOV     A,M     ;GET DRIVE TABLE VALUE
0288'   E603                ANI     3       ;EXTRACT STEP RATE
028A'   F618                ORI     FDCSKH  ;COMBINE WITH FDC SEEK COMMAND
028C'   CD 037E'            CALL    FDCCMD  ;OUTPUT FDC SEEK COMMAND
028F'   E691                ANI     91H     ;EXTRACT RELEVANT STATUS BITS
0291'   C9                  RET             ;DONE
                    ;
0292'   3E04        RECAL:  MVI     A,1<HSDBIT  ;GET HEAD SETTLE DELAY BIT
0294'   32 0009"            STA     DLYBIT  ;SET HEAD SETTLE DELAY BIT
0297'   CD 038F'            CALL    CLRFDC  ;CLEAR FDC
029A'   CD 0412'            CALL    GETDTA  ;GET DRIVE TABLE ADDRESS
029D'   7E                  MOV     A,M     ;GET DRIVE TABLE VALUE
029E'   E603                ANI     3       ;EXTRACT STEP RATE
02A0'   F608                ORI     FDCCAL  ;COMBINE WITH RE-CALIBRATE COMMAND
02A2'   CD 037E'            CALL    FDCCMD  ;OUTPUT FDC RE-CALIBRATE COMMAND
02A5'   C9                  RET             ;DONE
                    ;
02A6'   DD7E01      RETRDY: MOV     A,PDRDRV(X)  ;GET REQUESTED DRIVE
02A9'   FE04                CPI     4       ;TEST FOR VALID DRIVE NUMBER
02AB'   3E00                MVI     A,0     ;PRESET RETURN CODE=0
02AD'   D0                  RNC             ;IF INVALID DRIVE, RETURN NOT READY
02AE'   CD 02DF'            CALL    SELDSK  ;ELSE, SELECT REQUESTED DRIVE
02B1'   3E00                MVI     A,0     ;PRESET RETURN CODE=0
02B3'   C0                  RNZ             ;IF DRIVE NOT READY, DONE
02B4'   2F                  CMA             ;ELSE, SET RETURN CODE=0FFH
02B5'   21 000C"            LXI     H,DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
```

```
02B8'   CB66                    BIT     4,M       ;MINI-FLOPPY DISK BIT SET?
02BA'   C8                      RZ                ;IF NOT, DONE
02BB'   3E02                    MVI     A,2       ;GET INDEX PULSE SEQUENCE COUNT
02BD'   32 000D"                STA     NDXCNT    ;SET INDEX PULSE SEQUENCE COUNT=2
02C0'   CD 03EC'                CALL    ENACTC    ;ENABLE CTC INTERRUPT CONTROLLER
02C3'   3E04                    MVI     A,FDCINT!1<2  ;GET FDC INTERRUPT COMMAND
02C5'   D30C                    OUT     FDCCSR    ;OUTPUT FDC INTERRUPT COMMAND
02C7'   21 003C                 LXI     H,60      ;GET ONE SECOND DELAY COUNT
02CA'   CD 0000:09              CALL    DELAY#    ;DELAY FOR ONE SECOND
02CD'   3E03                    MVI     A,03H     ;GET CTC RESET COMMAND
02CF'   D30B                    OUT     CTCCH3    ;RESET CTC CHANNEL 3
02D1'   CD 038F'                CALL    CLRFDC    ;CLEAR FDC
02D4'   21 000D"                LXI     H,NDXCNT   ;SET INDEX PULSE SEQUENCE COUNT
02D7'   7E                      MOV     A,M       ;GET INDEX PULSE SEQUENCE COUNT
02D8'   3600                    MVI     M,0       ;SET INDEX PULSE SEQUENCE COUNT=0
02DA'   B7                      ORA     A         ;INDEX PULSE SEQUENCE COUNT=0?
02DB'   2F                      CMA               ;PRESET RETURN CODE=0FFH
02DC'   C8                      RZ                ;IF SEQUENCE COUNT=0, DONE
02DD'   AF                      XRA     A         ;ELSE, SET RETURN CODE=0
02DE'   C9                      RET               ;DONE
                        ;
02DF'   DD7E01          SELDSK: MOV     A,PDRDRV(X)  ;GET REQUESTED DRIVE
02E2'   CBDF                    SET     3,A       ;SET DOUBLE DENSITY BIT
02E4'   CD 0412'                CALL    GETDTA    ;GET DRIVE TABLE ADDRESS
02E7'   CB66                    BIT     MINI,M    ;MINI-FLOPPY DISK?
02E9'   2802                    JRZ     ..NMFD    ;IF NOT, CONTINUE
02EB'   CBE7                    SET     4,A       ;ELSE, SET MINI-FLOPPY DISK BIT
02ED'   D314            ..NMFD: OUT     FDCDSR    ;SELECT REQUESTED DRIVE
02EF'   32 000C"                STA     DSRSAV    ;SAVE DRIVE SELECT REGISTER VALUE
02F2'   CD 038F'                CALL    CLRFDC    ;CLEAR FDC
02F5'   E680                    ANI     80H       ;DRIVE READY?
02F7'   C0                      RNZ               ;IF NOT, DONE
02F8'   32 0009"                STA     DLYBIT    ;ELSE, SET HEAD SETTLE DELAY BIT=0
02FB'   3A 0004"                LDA     DRIVE     ;GET DRIVE NUMBER
02FE'   DDBE01                  CMP     PDRDRV(X)  ;DRIVE NUMBER=REQUESTED DRIVE?
0301'   C8                      RZ                ;IF SO, DONE
0302'   FEFF                    CPI     0FFH      ;DRIVE NUMBER INVALID?
0304'   2806                    JRZ     ..DNI     ;IF SO, CONTINUE
0306'   CD 0321'                CALL    ..GTTA    ;ELSE, GET TRACK SAVE TABLE ADDRESS
0309'   DB0D                    IN      FDCTRK    ;GET FDC TRACK REGISTER
030B'   77                      MOV     M,A       ;SAVE FDC TRACK REGISTER
030C'   DD7E01          ..DNI:  MOV     A,PDRDRV(X)  ;GET REQUESTED DRIVE
030F'   32 0004"                STA     DRIVE     ;SET DRIVE NUMBER
0312'   CD 0321'                CALL    ..GTTA    ;GET TRACK SAVE TABLE ADDRESS
0315'   7E                      MOV     A,M       ;GET FDC TRACK REGISTER
0316'   D30D                    OUT     FDCTRK    ;SET FDC TRACK REGISTER
0318'   D30F                    OUT     FDCDAT    ;SET FDC DATA REGISTER
031A'   3E10                    MVI     A,FDCSKN   ;GET FDC SEEK COMMAND
031C'   CD 037E'                CALL    FDCCMD    ;OUTPUT FDC SEEK COMMAND
031F'   AF                      XRA     A         ;SET RETURN CODE=0
0320'   C9                      RET               ;DONE
0321'   5F              ..GTTA: MOV     E,A       ;DRIVE NUMBER TO DE-REG
0322'   1600                    MVI     D,0       ;DOUBLE LENGTH
0324'   21 0011"                LXI     H,TRKTBL   ;GET TRACK SAVE TABLE
```

```
0327'   19                      DAD     D           ;INDEX INTO TRACK SAVE TABLE
0328'   C9                      RET                 ;DONE
                        ;
0329'   21 0015"        READID: LXI     H,RIDBUF  ;GET READ ID BUFFER
032C'   22 002A"                SHLD    DMAADR  ;SET DMA ADDRESS
032F'   21 0005                 LXI     H,6-1   ;GET SECTOR ID LENGTH (-1)
0332'   22 002C"                SHLD    DMALEN  ;SET DMA LENGTH
0335'   21 000C"                LXI     H,DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
0338'   CBDE                    SET     3,M     ;SET DOUBLE DENSITY BIT
033A'   7E                      MOV     A,M     ;GET SAVED DRIVE SELECT REGISTER
033B'   D314                    OUT     FDCDSR  ;SELECT DRIVE/SIDE/DENSITY
033D'   11 C09D         ..RIDL: LXI     D,FDCRID<8!9DH  ;GET READ ID COMMAND/MASK
0340'   3E01                    MVI     A,DMARDC  ;GET DMA READ COMMAND
0342'   CD 0355'                CALL    DMACOM  ;READ ID
0345'   C8                      RZ              ;IF READ OK, DONE
0346'   3A 000C"                LDA     DSRSAV  ;GET SAVED DRIVE SELECT REGISTER
0349'   EE08                    XRI     1<3     ;TOGGLE SINGLE/DOUBLE DENSITY BIT
034B'   D314                    OUT     FDCDSR  ;OUTPUT DRIVE SELECT REGISTER VALUE
034D'   32 000C"                STA     DSRSAV  ;SAVE DRIVE SELECT REGISTER VALUE
0350'   E608                    ANI     1<3     ;DOUBLE DENSITY SELECTED?
0352'   28E9                    JRZ     ..RIDL  ;IF NOT, CONTINUE
0354'   C9                      RET             ;ELSE, DONE
                        ;
0355'   32 0034"        DMACOM: STA     DMARWC  ;SET DMA READ/WRITE COMMAND
0358'   CD 0000:0C              CALL    LOKBNK# ;GAIN MUTUAL EXCLUSION ON BANK 1
035B'   CD 038F'                CALL    CLRFDC  ;CLEAR FDC
035E'   21 0027"                LXI     H,DMAPGM  ;GET DMA PROGRAM LIST
0361'   01 1010                 LXI     B,DMAPLL<8!DMACTL  ;B=PROGRAM LENGTH/C=PORT
0364'   EDB3                    OUTIR           ;PROGRAM DMA CONTROLLER
0366'   3A 0009"                LDA     DLYBIT  ;GET HEAD SETTLE DELAY BIT
0369'   B2                      ORA     D       ;COMBINE WITH FDC COMMAND
036A'   D5                      PUSH    D       ;SAVE ERROR MASK
036B'   CD 037E'                CALL    FDCCMD  ;OUTPUT FDC COMMAND
036E'   D1                      POP     D       ;RESTORE ERROR MASK
036F'   A3                      ANA     E       ;EXTRACT RELEVANT STATUS BITS
0370'   F5                      PUSH    PSW     ;SAVE ERROR STATUS
0371'   3EC3                    MVI     A,0C3H  ;GET DMA RESET COMMAND
0373'   D310                    OUT     DMACTL  ;DISABLE DMA CONTROLLER
0375'   AF                      XRA     A
0376'   32 0009"                STA     DLYBIT  ;SET HEAD SETTLE DELAY BIT=0
0379'   CD 0000:0D              CALL    FREBNK# ;FREE MUTUAL EXCLUSION ON BANK 1
037C'   F1                      POP     PSW     ;RESTORE ERROR STATUS
037D'   C9                      RET             ;DONE
                        ;
037E'   F5              FDCCMD: PUSH    PSW     ;SAVE FDC COMMAND
037F'   CD 03EC'                CALL    ENACTC  ;ENABLE CTC INTERRUPT CONTROLLER
0382'   F1                      POP     PSW     ;RESTORE FDC COMMAND
0383'   D30C                    OUT     FDCCSR  ;OUTPUT FDC COMMAND
0385'   21 0021"                LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
0388'   CD 0000:06              CALL    WAIT#   ;WAIT FOR OPERATION TO COMPLETE
038B'   3A 000A"                LDA     INTCST  ;GET INTERRUPT COMPLETION STATUS
038E'   C9                      RET             ;DONE
                        ;
038F'   3ED0            CLRFDC: MVI     A,FDCINT  ;GET FDC INTERRUPT COMMAND
```

```
0391'   D30C                    OUT     FDCCSR  ;OUTPUT FDC INTERRUPT COMMAND
0393'   E3                      XTHL            ;DELAY
0394'   E3                      XTHL
0395'   E3                      XTHL
0396'   E3                      XTHL
0397'   E3                      XTHL
0398'   E3                      XTHL
0399'   DB0F                    IN      FDCDAT  ;CLEAR DRQ
039B'   DB0C                    IN      FDCCSR  ;CLEAR INTRQ
039D'   C9                      RET             ;DONE
                        ;
039E'   ED73 0000:0E    DSKISR::SSPD    INTSP#  ;SAVE INTERRUPT STACK POINTER
03A2'   31 0000:0F              LXI     SP,INTSTK#  ;SET UP AUX STACK
03A5'   F5                      PUSH    PSW     ;SAVE REGISTERS
03A6'   C5                      PUSH    B
03A7'   D5                      PUSH    D
03A8'   E5                      PUSH    H
03A9'   DB0C                    IN      FDCCSR  ;GET FDC COMPLETION STATUS
03AB'   32 000A"                STA     INTCST  ;SAVE INTERRUPT COMPLETION STATUS
03AE'   3A 000D"                LDA     NDXCNT  ;GET INDEX PULSE SEQUENCE COUNT
03B1'   3D                      DCR     A       ;INDEX PULSE SEQUENCE COUNT=0?
03B2'   FA 03D7'                JM      ..ISCO  ;IF SO, CONTINUE
03B5'   2015                    JRNZ    ..FIP   ;IF FIRST INDEX PULSE, CONTINUE
03B7'   3A 000E"                LDA     NDXTIC  ;ELSE, GET INDEX PULSE TICK COUNT
03BA'   4F                      MOV     C,A     ;INDEX PULSE TICK COUNT TO C-REG
03BB'   3A 0000:10              LDA     TICCNT# ;GET CURRENT TICK COUNT
03BE'   32 000E"                STA     NDXTIC  ;UPDATE INDEX PULSE TICK COUNT
03C1'   91                      SUB     C       ;CALC ELAPSED TICK COUNTS
03C2'   FE0E                    CPI     14      ;INDEX PULSE TIMING WITHIN LIMITS?
03C4'   301B                    JRNC    ..ISRX  ;IF NOT, CONTINUE
03C6'   AF                      XRA     A
03C7'   32 000D"                STA     NDXCNT  ;SET INDEX PULSE SEQUENCE COUNT=0
03CA'   1811                    JMPR    ..ISCX  ;CONTINUE
03CC'   32 000D"        ..FIP:  STA     NDXCNT  ;SET INDEX PULSE SEQUENCE COUNT=1
03CF'   3A 0000:10              LDA     TICCNT# ;GET TICK COUNT
03D2'   32 000E"                STA     NDXTIC  ;SAVE INDEX PULSE TICK COUNT
03D5'   180A                    JMPR    ..ISRX  ;CONTINUE
03D7'   21 0021"        ..ISCO: LXI     H,DWTSPH   ;GET DISK WAIT SEMAPHORE
03DA'   CD 0000:07              CALL    SIGNAL# ;SIGNAL PROCESS AS READY
03DD'   3E03            ..ISCX: MVI     A,03H   ;GET CTC RESET COMMAND
03DF'   D30B                    OUT     CTCCH3  ;RESET CTC CHANNEL 3
03E1'   E1              ..ISRX: POP     H       ;RESTORE REGISTERS
03E2'   D1                      POP     D
03E3'   C1                      POP     B
03E4'   F1                      POP     PSW
03E5'   ED7B 0000:0E            LSPD    INTSP#  ;RESTORE STACK POINTER
03E9'   FB                      EI              ;ENABLE INTERRUPTS
03EA'   ED4D                    RETI            ;DONE
                        ;
03EC'   3ED7            ENACTC: MVI     A,0D7H  ;GET CTC CHANNEL 3 CONTROL WORD
03EE'   D30B                    OUT     CTCCH3  ;INITIALIZE CTC CHANNEL 3
03F0'   3E01                    MVI     A,1     ;GET CTC CHANNEL 3 TIME CONSTANT
03F2'   D30B                    OUT     CTCCH3  ;SET CTC CHANNEL 3 TIME CONSTANT
03F4'   C9                      RET             ;DONE
```

```
                                ;
03F5'   CD 040B'        GETXLT: CALL    GETDST  ;GET DST ADDRESS
03F8'   11 0000:11              LXI     D,XLTBL#  ;GET OFFSET TO TRANSLATION TABLE
03FB'   19                      DAD     D       ;CALC TRANSLATION TABLE ADDRESS
03FC'   5E                      MOV     E,M     ;GET TRANSLATION TABLE ADDRESS
03FD'   23                      INX     H
03FE'   56                      MOV     D,M
03FF'   EB                      XCHG            ;TRANSLATION TABLE TO HL-REG
0400'   7C                      MOV     A,H
0401'   B5                      ORA     L       ;TRANSLATION TABLE REQUIRED?
0402'   C9                      RET             ;DONE
                                ;
0403'   CD 040B'        GETTCA: CALL    GETDST  ;GET DST ADDRESS
0406'   11 0000:12              LXI     D,TYPCOD#  ;GET OFFSET TO DISK TYPE CODE
0409'   19                      DAD     D       ;CALC DISK TYPE CODE ADDRESS
040A'   C9                      RET             ;DONE
                                ;
040B'   DD6E0C          GETDST: MOV     L,PDRDST(X)  ;GET DST ADDRESS
040E'   DD660D                  MOV     H,PDRDST+1(X)
0411'   C9                      RET             ;DONE
                                ;
0412'   DD5E01          GETDTA: MOV     E,PDRDRV(X)  ;GET REQUESTED DRIVE
0415'   1600                    MVI     D,0     ;DOUBLE LENGTH
0417'   21 0000"                LXI     H,DRVTBL  ;GET DRIVE TABLE
041A'   19                      DAD     D       ;INDEX INTO DRIVE TABLE
041B'   C9                      RET             ;DONE
                                ;
                                .PRGEND
```

```
                         ;
                         ; VERSION: 01/05/84
                         ;
                         .IDENT  DSTFDC          ;MODULE ID
                         ;
                         .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                         ;
0002                     TSD     = 2             ;TWO-SIDED DISK BIT (TYPE CODE)
0003                     DDD     = 3             ;DOUBLE DENSITY DISK BIT (TYPE CODE

0004                     MINI    = 4             ;MINI-FLOPPY DISK BIT (TYPE CODE)
0005                     TPI96   = 5             ;96-TPI DISK BIT (TYPE CODE)
                         ;
0000'                            .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                         ;
                         ;       1024 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED
                         ;
0000'   0011'            DSTBLS::.WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
0002'   04                       .BYTE   4       ;BLOCK SIZE
0003'   0268                     .WORD   (77*(16*(1<3)))/(1<4)  ;NUMBER OF BLOCKS
0005'   04                       .BYTE   4       ;NUMBER OF DIRECTORY BLOCKS
0006'   03                       .BYTE   3       ;PHYSICAL SECTOR SIZE (2^N*128)
0007'   0010                     .WORD   16      ;PHYSICAL SECTORS PER TRACK
0009'   004D                     .WORD   77      ;PHYSICAL TRACKS PER DISK
000B'   0000                     .WORD   0       ;NUMBER OF RESERVED TRACKS
000D'   0000                     .WORD   0       ;TRANSLATION TABLE ADDRESS
000F'   0F                       .BYTE   1<DDD!1<TSD!3  ;DISK TYPE CODE
0010'   35                       .BYTE   35H     ;GAP LENGTH
                         ;
                         ;       1024 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED, 96-TPI
                         (MINI)
                         ;
0011'   0022'                    .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
0013'   04                       .BYTE   4       ;BLOCK SIZE
0014'   0190                     .WORD   (80*(10*(1<3)))/(1<4)  ;NUMBER OF BLOCKS
0016'   04                       .BYTE   4       ;NUMBER OF DIRECTORY BLOCKS
0017'   03                       .BYTE   3       ;PHYSICAL SECTOR SIZE (2^N*128)
0018'   000A                     .WORD   10      ;PHYSICAL SECTORS PER TRACK
001A'   0050                     .WORD   80      ;PHYSICAL TRACKS PER DISK
001C'   0000                     .WORD   0       ;NUMBER OF RESERVED TRACKS
001E'   0000                     .WORD   0       ;TRANSLATION TABLE ADDRESS
0020'   3F                       .BYTE   1<TPI96!1<MINI!1<DDD!1<TSD!3  ;DISK TYPE CO
                         E
0021'   35                       .BYTE   35H     ;GAP LENGTH
                         ;
                         ;       1024 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED (MINI)
                         ;
0022'   0033'                    .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
0024'   04                       .BYTE   4       ;BLOCK SIZE
0025'   00C8                     .WORD   (40*(10*(1<3)))/(1<4)  ;NUMBER OF BLOCKS
0027'   02                       .BYTE   2       ;NUMBER OF DIRECTORY BLOCKS
0028'   03                       .BYTE   3       ;PHYSICAL SECTOR SIZE (2^N*128)
0029'   000A                     .WORD   10      ;PHYSICAL SECTORS PER TRACK
002B'   0028                     .WORD   40      ;PHYSICAL TRACKS PER DISK
```

```
002D'    0000                     .WORD   0        ;NUMBER OF RESERVED TRACKS
002F'    0000                     .WORD   0        ;TRANSLATION TABLE ADDRESS
0031'    1F                       .BYTE   1<MINI!1<DDD!1<TSD!3  ;DISK TYPE CODE
0032'    35                       .BYTE   35H      ;GAP LENGTH
                         ;
                         ;        1024 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED
                         ;
0033'    0044'                    .WORD   .+DSTL   ;DISK SPEC TABLE LINK POINTER
0035'    04                       .BYTE   4        ;BLOCK SIZE
0036'    0134                     .WORD   (77*(8*(1<3)))/(1<4)  ;NUMBER OF BLOCKS
0038'    03                       .BYTE   3        ;NUMBER OF DIRECTORY BLOCKS
0039'    03                       .BYTE   3        ;PHYSICAL SECTOR SIZE (2^N*128)
003A'    0008                     .WORD   8        ;PHYSICAL SECTORS PER TRACK
003C'    004D                     .WORD   77       ;PHYSICAL TRACKS PER DISK
003E'    0000                     .WORD   0        ;RESERVED TRACKS
0040'    0000                     .WORD   0        ;TRANSLATION TABLE ADDRESS
0042'    0B                       .BYTE   1<DDD!3  ;DISK TYPE CODE
0043'    35                       .BYTE   35H      ;GAP LENGTH
                         ;
                         ;        1024 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED, 96-TPI
                        (MINI)
                         ;
0044'    0055'                    .WORD   .+DSTL   ;DISK SPEC TABLE LINK POINTER
0046'    04                       .BYTE   4        ;BLOCK SIZE
0047'    00C8                     .WORD   (80*(5*(1<3)))/(1<4)  ;NUMBER OF BLOCKS
0049'    02                       .BYTE   2        ;NUMBER OF DIRECTORY BLOCKS
004A'    03                       .BYTE   3        ;PHYSICAL SECTOR SIZE (2^N*128)
004B'    0005                     .WORD   5        ;PHYSICAL SECTORS PER TRACK
004D'    0050                     .WORD   80       ;PHYSICAL TRACKS PER DISK
004F'    0000                     .WORD   0        ;RESERVED TRACKS
0051'    0000                     .WORD   0        ;TRANSLATION TABLE ADDRESS
0053'    3B                       .BYTE   1<TPI96!1<MINI!1<DDD!3  ;DISK TYPE CODE
0054'    35                       .BYTE   35H      ;GAP LENGTH
                         ;
                         ;        1024 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED (MINI)
                         ;
0055'    0066'                    .WORD   .+DSTL   ;DISK SPEC TABLE LINK POINTER
0057'    03                       .BYTE   3        ;BLOCK SIZE
0058'    00C8                     .WORD   (40*(5*(1<3)))/(1<3)  ;NUMBER OF BLOCKS
005A'    02                       .BYTE   2        ;NUMBER OF DIRECTORY BLOCKS
005B'    03                       .BYTE   3        ;PHYSICAL SECTOR SIZE (2^N*128)
005C'    0005                     .WORD   5        ;PHYSICAL SECTORS PER TRACK
005E'    0028                     .WORD   40       ;PHYSICAL TRACKS PER DISK
0060'    0000                     .WORD   0        ;RESERVED TRACKS
0062'    0000                     .WORD   0        ;TRANSLATION TABLE ADDRESS
0064'    1B                       .BYTE   1<MINI!1<DDD!3  ;DISK TYPE CODE
0065'    35                       .BYTE   35H      ;GAP LENGTH
                         ;
                         ;        512 BYTE SECTOR, SINGLE-DENSITY, TWO-SIDED
                         ;
0066'    0077'                    .WORD   .+DSTL   ;DISK SPEC TABLE LINK POINTER
0068'    04                       .BYTE   4        ;BLOCK SIZE
0069'    0134                     .WORD   (77*(16*(1<2)))/(1<4)  ;NUMBER OF BLOCKS
006B'    03                       .BYTE   3        ;NUMBER OF DIRECTORY BLOCKS
```

```
006C'   02                      .BYTE   2       ;PHYSICAL SECTOR SIZE (2^N*128)
006D'   0010                    .WORD   16      ;PHYSICAL SECTORS PER TRACK
006F'   004D                    .WORD   77      ;PHYSICAL TRACKS PER DISK
0071'   0000                    .WORD   0       ;RESERVED TRACKS
0073'   0000                    .WORD   0       ;TRANSLATION TABLE ADDRESS
0075'   06                      .BYTE   1<TSD!2 ;DISK TYPE CODE
0076'   1B                      .BYTE   1BH     ;GAP LENGTH
                        ;
                        ;       512 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED
                        ;
0077'   0088'                   .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
0079'   04                      .BYTE   4       ;BLOCK SIZE
007A'   009A                    .WORD   (77*(8*(1<2)))/(1<4)  ;NUMBER OF BLOCKS
007C'   02                      .BYTE   2       ;NUMBER OF DIRECTORY BLOCKS
007D'   02                      .BYTE   2       ;PHYSICAL SECTOR SIZE (2^N*128)
007E'   0008                    .WORD   8       ;PHYSICAL SECTORS PER TRACK
0080'   004D                    .WORD   77      ;PHYSICAL TRACKS PER DISK
0082'   0000                    .WORD   0       ;RESERVED TRACKS
0084'   0000                    .WORD   0       ;TRANSLATION TABLE ADDRESS
0086'   02                      .BYTE   2       ;DISK TYPE CODE
0087'   1B                      .BYTE   1BH     ;GAP LENGTH
                        ;
                        ;       512 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED (MINI)
                        ;
0088'   0099'                   .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
008A'   04                      .BYTE   4       ;BLOCK SIZE
008B'   00A0                    .WORD   (40*(16*(1<2)))/(1<4)  ;NUMBER OF BLOCKS
008D'   02                      .BYTE   2       ;NUMBER OF DIRECTORY BLOCKS
008E'   02                      .BYTE   2       ;PHYSICAL SECTOR SIZE (2^N*128)
008F'   0001                    .WORD   1       ;PHYSICAL SECTORS PER TRACK
0091'   0280                    .WORD   40*8*2  ;PHYSICAL TRACKS PER DISK
0093'   0001                    .WORD   1       ;RESERVED TRACKS
0095'   0000                    .WORD   0       ;TRANSLATION TABLE ADDRESS
0097'   1E                      .BYTE   1<MINI!1<DDD!1<TSD!2  ;DISK TYPE CODE
0098'   1B                      .BYTE   1BH     ;GAP LENGTH
                        ;
                        ;       512 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED (MINI)
                        ;
0099'   00AA'                   .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
009B'   03                      .BYTE   3       ;BLOCK SIZE
009C'   00A0                    .WORD   (40*(8*(1<2)))/(1<3)  ;NUMBER OF BLOCKS
009E'   02                      .BYTE   2       ;NUMBER OF DIRECTORY BLOCKS
009F'   02                      .BYTE   2       ;PHYSICAL SECTOR SIZE (2^N*128)
00A0'   0001                    .WORD   1       ;PHYSICAL SECTORS PER TRACK
00A2'   0140                    .WORD   40*8    ;PHYSICAL TRACKS PER DISK
00A4'   0001                    .WORD   1       ;RESERVED TRACKS
00A6'   0000                    .WORD   0       ;TRANSLATION TABLE ADDRESS
00A8'   1A                      .BYTE   1<MINI!1<DDD!2  ;DISK TYPE CODE
00A9'   1B                      .BYTE   1BH     ;GAP LENGTH
                        ;
                        ;       128 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED
                        ;
00AA'   0000            DSTA:   .WORD   0       ;DISK SPEC TABLE LINK POINTER
00AC'   03              DSTB:   .BYTE   3       ;BLOCK SIZE
```

```
 00AD'   00F3                   .WORD   (75*(26*(1<0)))/(1<3)  ;NUMBER OF BLOCKS
 00AF'   02                     .BYTE   2         ;NUMBER OF DIRECTORY BLOCKS
 00B0'   00                     .BYTE   0         ;PHYSICAL SECTOR SIZE (2^N*128)
 00B1'   001A                   .WORD   26        ;PHYSICAL SECTORS PER TRACK
 00B3'   004D                   .WORD   77        ;PHYSICAL TRACKS PER DISK
 00B5'   0002                   .WORD   2         ;RESERVED TRACKS
                        ;
 000B                   XLTBL   =: .-DSTB         ;TRANSLATION TABLE ADDRESS OFFSET
                        ;
 00B7'   00BB'                  .WORD   TRTBL     ;TRANSLATION TABLE ADDRESS
                        ;
 000F                   DTCO    =: .-DSTA         ;DISK TYPE CODE OFFSET
 000D                   TYPCOD  =: .-DSTB         ;DISK TYPE CODE OFFSET
                        ;
 00B9'   00                     .BYTE   0         ;DISK TYPE CODE
                        ;
 000E                   GAPLEN  =: .-DSTB         ;GAP LENGTH OFFSET
                        ;
 00BA'   07                     .BYTE   7         ;GAP LENGTH
                        ;
 0011                   DSTL    = .-DSTA          ;DISK SPEC TABLE LENGTH
                        ;
                        ; SINGLE-DENSITY/SINGLE-SIDED SECTOR TRANSLATION TABLE
                        ;
 00BB'   00060C121804   TRTBL:  .BYTE   0,6,12,18,24,4,10,16,22
 00C4'   02080E140107           .BYTE   2,8,14,20,1,7,13,19,25
 00CD'   050B11170309           .BYTE   5,11,17,23,3,9,15,21
                        ;
                        .PRGEND
```

```
                      ;
                      ; VERSION: 01/05/84
                      ;
                      .IDENT  DSKHDC          ;MODULE ID
                      ;
                      .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                      ;
0090                  IOBASE  = 90H           ;I/O BASE ADDRESS
                      ;
0090                  HDCDAT  = IOBASE+0      ;HDC DATA REGISTER
0091                  HDCEWP  = IOBASE+1      ;HDC ERROR/WRITE PRECOMP REGISTER
0092                  HDCSCT  = IOBASE+2      ;HDC SECTOR COUNT REGISTER
0093                  HDCSEC  = IOBASE+3      ;HDC SECTOR NUMBER REGISTER
0094                  HDCCYL  = IOBASE+4      ;HDC CYLINDER REGISTER (LOW)
0095                  HDCCYH  = IOBASE+5      ;HDC CYLINDER REGISTER (HIGH)
0096                  HDCSDH  = IOBASE+6      ;HDC SIZE/DRIVE/HEAD REGISTER
0097                  HDCCSR  = IOBASE+7      ;HDC COMMAND/STATUS REGISTER
                      ;
0010                  HDCCAL  = 10H           ;HDC CALIBRATE DRIVE COMMAND
0020                  HDCRDS  = 20H           ;HDC READ SECTOR COMMAND
0030                  HDCWRS  = 30H           ;HDC WRITE SECTOR COMMAND
0050                  HDCFMT  = 50H           ;HDC FORMAT TRACK COMMAND
0070                  HDCSEK  = 70H           ;HDC SEEK COMMAND
                      ;
0001                  ERRMSK  = 01H           ;COMPLETION ERROR MASK
0040                  RDYMSK  = 40H           ;DRIVE READY MASK
                      ;
0006                  STEPRT  = 6             ;STEP RATE (3-MS)
                      ;
0000"                         .LOC    .DATA.# ;LOCATE IN DATA AREA
                      ;
0000"   84            HDCDST: .BYTE   84H     ;ALLOCATION BLOCK SIZE
0001"   1540                  .WORD   5440    ;NUMBER OF ALLOCATION BLOCKS
0003"   28                    .BYTE   40      ;NUMBER OF DIRECTORY BLOCKS
0004"   02                    .BYTE   2       ;PHYSICAL SECTOR SIZE (2^N*128)
0005"   0011                  .WORD   17      ;PHYSICAL SECTORS PER TRACK
0007"   0500                  .WORD   1280    ;PHYSICAL TRACKS PER DISK
0009"   0000                  .WORD   0       ;NUMBER OF RESERVED TRACKS
                      ;
000B"   0000          DMAADR: .WORD   0       ;DMA ADDRESS
000D"   00            SECCNT: .BYTE   0       ;SECTOR COUNT
000E"   00            INTCST: .BYTE   0       ;INTERRUPT COMPLETION STATUS
000F"   00            ERRORS: .BYTE   0       ;ERROR ACCUMULATOR
0010"   0000          CALTBL: .BYTE   0,0     ;DRIVE CALIBRATED TABLE
0012"   0000          RETSP:  .WORD   0       ;ERROR RETURN STACK POINTER
                      ;
0014"                 DMXSPH:                 ;MUTUAL EXCLUSION SEMAPHORE
0014"   0001                  .WORD   1       ;SEMAPHORE COUNT
0016"   0016"         ..DMXH: .WORD   ..DMXH  ;SEMAPHORE P/D HEAD
0018"   0016"                 .WORD   ..DMXH
                      ;
001A"                 DWTSPH:                 ;DISK WAIT SEMAPHORE
001A"   0000                  .WORD   0       ;SEMAPHORE COUNT
001C"   001C"         ..DWTH: .WORD   ..DWTH  ;SEMAPHORE P/D HEAD
```

```
001E"    001C"                  .WORD   ..DWTH
                        ;
0000:04                         .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                        ;
0000:04 21 016E'        DSKIN%::LXI     H,DSKISR  ;GET INTERRUPT SERVICE ADDRESS
0003:04 3E02                    MVI     A,2      ;GET VECTORED INTERRUPT NUMBER
0005:04 CD 0000:05              CALL    INTNIT# ;INITIALIZE INTERRUPT VECTOR
0008:04 C9                      RET             ;DONE
                        ;
0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
0000'    21 0014"       DSKDR%::LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
0003'    CD 0000:06             CALL    WAIT#   ;DISPATCH IF NECESSARY
0006'    CD 0012'               CALL    ..DD    ;CALL DISK DRIVER
0009'    F5                     PUSH    PSW     ;SAVE RETURN CODE
000A'    21 0014"               LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
000D'    CD 0000:07             CALL    SIGNAL# ;SIGNAL PROCESS AS READY
0010'    F1                     POP     PSW     ;RESTORE RETURN CODE
0011'    C9                     RET             ;DONE
                        ;
0012'    ED73 0012"     ..DD:   SSPD    RETSP   ;SAVE ERROR RETURN STACK POINTER
0016'    DD7E00                 MOV     A,PDRFCN(X)  ;GET FUNCTION NUMBER
0019'    B7                     ORA     A       ;FUNCTION NUMBER=0?
001A'    280F                   JRZ     RDDSK   ;IF SO, CONTINUE
001C'    3D                     DCR     A       ;FUNCTION NUMBER=1?
001D'    282A                   JRZ     WRDSK   ;IF SO, CONTINUE
001F'    3D                     DCR     A       ;FUNCTION NUMBER=2?
0020'    CA 0130'               JZ      RETDST  ;IF SO, CONTINUE
0023'    3D                     DCR     A       ;FUNCTION NUMBER=3?
0024'    CA 013F'               JZ      RETRDY  ;IF SO, CONTINUE
0027'    3D                     DCR     A       ;FUNCTION NUMBER=4?
0028'    2850                   JRZ     FMTDSK  ;IF SO, CONTINUE
002A'    C9                     RET             ;ELSE, DONE
                        ;
002B'    CD 009A'       RDDSK:  CALL    SETUP   ;DO COMMON SETUP
002E'    3E20           ..RDL:  MVI     A,HDCRDS  ;GET READ SECTOR COMMAND
0030'    D397                   OUT     HDCCSR  ;OUTPUT READ SECTOR COMMAND
0032'    CD 0162'               CALL    WTINT   ;WAIT FOR INTERRUPT
0035'    01 0090                LXI     B,HDCDAT  ;GET DATA PORT ADDRESS
0038'    2A 000B"               LHLD    DMAADR  ;GET DMA ADDRESS
003B'    EDB2                   INIR            ;INPUT 256 BYTES OF DATA
003D'    EDB2                   INIR            ;INPUT 256 BYTES OF DATA
003F'    22 000B"               SHLD    DMAADR  ;UPDATE DMA ADDRESS
0042'    CD 0106'               CALL    RWSCOM  ;DO COMMON WRAP UP
0045'    20E7                   JRNZ    ..RDL   ;IF NOT LAST SECTOR, CONTINUE
0047'    182C                   JMPR    RWCXIT  ;ELSE, CONTINUE
                        ;
0049'    CD 009A'       WRDSK:  CALL    SETUP   ;DO COMMON SETUP
004C'    3E30           ..WRL:  MVI     A,HDCWRS  ;GET WRITE SECTOR COMMAND
004E'    CD 00F5'               CALL    WRFCOM  ;DO COMMON SETUP
0051'    CD 0106'               CALL    RWSCOM  ;DO COMMON WRAP UP
0054'    20F6                   JRNZ    ..WRL   ;IF NOT LAST SECTOR, CONTINUE
0056'    CD 0075'               CALL    RWCXIT  ;ANY ERRORS?
0059'    C0                     RNZ             ;IF SO, DONE
```

```
005A'   CD 009A'            CALL    SETUP   ;ELSE, DO COMMON SETUP
005D'   3E20        ..VFL:  MVI     A,HDCRDS  ;GET READ SECTOR COMMAND
005F'   D397                OUT     HDCCSR  ;OUTPUT READ SECTOR COMMAND
0061'   CD 0162'            CALL    WTINT   ;WAIT FOR INTERRUPT
0064'   F5                  PUSH    PSW     ;SAVE ERROR STATUS
0065'   01 0002             LXI     B,2     ;GET LOOP COUNTS
0068'   DB90        ..RDL:  IN      HDCDAT  ;INPUT DATA CHARACTER
006A'   10FC                DJNZ    ..RDL   ;CONTINUE
006C'   0D                  DCR     C       ;DECREMENT LOOP COUNT
006D'   20F9                JRNZ    ..RDL   ;CONTINUE
006F'   F1                  POP     PSW     ;RESTORE ERROR STATUS
0070'   CD 0106'            CALL    RWSCOM  ;DO COMMON WRAP UP
0073'   20E8                JRNZ    ..VFL   ;IF NOT LAST SECTOR, CONTINUE
                    ;
0075'   3A 000F"    RWCXIT: LDA     ERRORS  ;GET ERROR ACCUMULATOR
0078'   181B                JMPR    RWFXIT  ;CONTINUE
                    ;
007A'   CD 014E'    FMTDSK: CALL    SELDSK  ;SELECT REQUESTED DRIVE
007D'   CA 0129'            JZ      FATAL   ;IF DRIVE NOT READY, CONTINUE
0080'   DD7E02              MOV     A,PDRTRK(X)  ;GET REQUESTED TRACK NUMBER
0083'   DDB603              ORA     PDRTRK+1(X)  ;REQUESTED TRACK NUMBER=0?
0086'   CC 0115'            CZ      RECAL   ;IF SO, RE-CALIBRATE DRIVE
0089'   3E11                MVI     A,17    ;GET NUMBER OF SECTORS PER TRACK
008B'   D392                OUT     HDCSCT  ;OUTPUT NUMBER OF SECTORS PER TRAC
008D'   CD 00C4'            CALL    RWFCOM  ;DO COMMON SETUP
0090'   3E50                MVI     A,HDCFMT  ;GET FORMAT TRACK COMMAND
0092'   CD 00F5'            CALL    WRFCOM  ;DO COMMON SETUP
                    ;
0095'   B7          RWFXIT: ORA     A       ;ANY ERRORS?
0096'   C8                  RZ              ;IF NOT, DONE
0097'   3EFF                MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
0099'   C9                  RET             ;DONE
                    ;
009A'   CD 014E'    SETUP:  CALL    SELDSK  ;SELECT REQUESTED DRIVE
009D'   CA 0129'            JZ      FATAL   ;IF DRIVE NOT READY, CONTINUE
00A0'   21 0010"            LXI     H,CALTBL  ;ELSE, GET DRIVE CALIBRATED TABL
00A3'   DDCB014E            BIT     1,PDRDRV(X)  ;SECOND PHYSICAL VOLUME?
00A7'   2801                JRZ     ..NSPV  ;IF NOT, CONTINUE
00A9'   23                  INX     H       ;ELSE, INCREMENT TABLE ADDRESS
00AA'   7E          ..NSPV: MOV     A,M     ;GET DRIVE CALIBRATED STATUS
00AB'   36FF                MVI     M,0FFH  ;SET DRIVE CALIBRATED STATUS=0FFH
00AD'   B7                  ORA     A       ;DRIVE CALIBRATED?
00AE'   CC 0115'            CZ      RECAL   ;IF NOT, RE-CALIBRATE DRIVE
00B1'   DD7E04              MOV     A,PDRSEC(X)  ;GET SECTOR NUMBER
00B4'   D393                OUT     HDCSEC  ;OUTPUT SECTOR NUMBER
00B6'   DD7E06              MOV     A,PDRSC(X)  ;GET SECTOR COUNT
00B9'   32 000D"            STA     SECCNT  ;SET SECTOR COUNT
00BC'   CD 00C4'            CALL    RWFCOM  ;DO COMMON SETUP
00BF'   AF                  XRA     A
00C0'   32 000F"            STA     ERRORS  ;CLEAR ERROR ACCUMULATOR
00C3'   C9                  RET             ;DONE
                    ;
00C4'   DD6E0A      RWFCOM: MOV     L,PDRDMA(X)  ;GET DMA ADDRESS
00C7'   DD660B              MOV     H,PDRDMA+1(X)
```

```
00CA'   22 000B"            SHLD    DMAADR  ;SET DMA ADDRESS
00CD'   DD6E02              MOV     L,PDRTRK(X)  ;GET REQUESTED TRACK NUMBER
00D0'   DD6603              MOV     H,PDRTRK+1(X)
00D3'   7D                  MOV     A,L     ;GET LSB OF TRACK NUMBER
00D4'   F5                  PUSH    PSW     ;SAVE LSB OF TRACK NUMBER
00D5'   CB3C                SRLR    H       ;ELIMINATE HEAD NUMBER
00D7'   CB1D                RARR    L
00D9'   CB3C                SRLR    H
00DB'   CB1D                RARR    L
00DD'   7D                  MOV     A,L     ;GET LSB OF TRACK NUMBER
00DE'   D394                OUT     HDCCYL  ;OUTPUT LSB OF TRACK NUMBER
00E0'   7C                  MOV     A,H     ;GET MSB OF TRACK NUMBER
00E1'   D395                OUT     HDCCYH  ;OUTPUT MSB OF TRACK NUMBER
00E3'   F1                  POP     PSW     ;RESTORE LSB OF TRACK NUMBER
00E4'   E603                ANI     3       ;EXTRACT HEAD NUMBER
00E6'   DDCB0146            BIT     0,PDRDRV(X)  ;SECOND LOGICAL VOLUME?
00EA'   2802                JRZ     ..NSLV  ;IF NOT, CONTINUE
00EC'   CBD7                SET     2,A     ;ELSE, SET BIT 2 OF HEAD NUMBER
00EE'   4F          ..NSLV: MOV     C,A     ;HEAD NUMBER TO C-REG
00EF'   DB96                IN      HDCSDH  ;GET SIZE/DRIVE/HEAD REGISTER
00F1'   B1                  ORA     C       ;SET HEAD NUMBER FIELD
00F2'   D396                OUT     HDCSDH  ;OUTPUT SIZE/DRIVE/HEAD
00F4'   C9                  RET             ;DONE
                    ;
00F5'   D397        WRFCOM: OUT     HDCCSR  ;OUTPUT COMMAND
00F7'   01 0090             LXI     B,HDCDAT  ;GET DATA PORT ADDRESS
00FA'   2A 000B"            LHLD    DMAADR  ;GET DMA ADDRESS
00FD'   EDB3                OUTIR           ;OUTPUT 256 BYTES OF DATA
00FF'   EDB3                OUTIR           ;OUTPUT 256 BYTES OF DATA
0101'   22 000B"            SHLD    DMAADR  ;UPDATE DMA ADDRESS
0104'   185C                JMPR    WTINT   ;WAIT FOR INTERRUPT
                    ;
0106'   21 000F"    RWSCOM: LXI     H,ERRORS  ;GET ERROR ACCUMULATOR
0109'   B6                  ORA     M       ;COMBINE STATUS WITH ACCUMULATOR
010A'   77                  MOV     M,A     ;UPDATE ERROR ACCUMULATOR
010B'   DB93                IN      HDCSEC  ;GET SECTOR NUMBER
010D'   3C                  INR     A       ;INCREMENT SECTOR NUMBER
010E'   D393                OUT     HDCSEC  ;UPDATE SECTOR NUMBER
0110'   21 000D"            LXI     H,SECCNT  ;GET SECTOR COUNT
0113'   35                  DCR     M       ;DECREMENT SECTOR COUNT
0114'   C9                  RET             ;DONE
                    ;
0115'   AF          RECAL:  XRA     A       ;GET TRACK 0
0116'   D391                OUT     HDCEWP  ;SET WRITE PRECOMP TRACK REGISTER
0118'   3E16                MVI     A,HDCCAL!STEPRT  ;GET CALIBRATE COMMAND
011A'   D397                OUT     HDCCSR  ;OUTPUT CALIBRATE DRIVE COMMAND
011C'   CD 0162'            CALL    WTINT   ;WAIT FOR INTERRUPT
011F'   2008                JRNZ    FATAL   ;IF ERRORS, CONTINUE
0121'   3E10                MVI     A,HDCCAL  ;GET CALIBRATE DRIVE COMMAND
0123'   D397                OUT     HDCCSR  ;OUTPUT CALIBRATE DRIVE COMMAND
0125'   CD 0162'            CALL    WTINT   ;WAIT FOR INTERRUPT
0128'   C8                  RZ              ;IF NO ERRORS, DONE
                    ;
0129'   ED7B 0012"  FATAL:  LSPD    RETSP   ;RESTORE STACK POINTER
```

```
012D'   3EFF                    MVI     A,0FFH  ;RETURN ERROR CODE
012F'   C9                      RET             ;DONE
                        ;
0130'   CD 013F'        RETDST: CALL    RETRDY  ;RETURN READY STATUS
0133'   B7                      ORA     A       ;DRIVE READY?
0134'   C8                      RZ              ;IF NOT, DONE
0135'   21 0000"                LXI     H,HDCDST  ;ELSE, GET DST ADDRESS
0138'   DD750C                  MOV     PDRDST(X),L  ;SET DST ADDRESS
013B'   DD740D                  MOV     PDRDST+1(X),H
013E'   C9                      RET             ;DONE
                        ;
013F'   DD7E01          RETRDY: MOV     A,PDRDRV(X)  ;GET DISK NUMBER
0142'   FE04                    CPI     4       ;TEST FOR VALID DRIVE NUMBER
0144'   3E00                    MVI     A,0     ;PRESET RETURN CODE=0
0146'   D0                      RNC             ;IF INVALID DRIVE, RETURN NOT READ
0147'   CD 014E'                CALL    SELDSK  ;ELSE, SELECT REQUESTED DRIVE
014A'   C8                      RZ              ;IF DRIVE NOT READY, DONE
014B'   3EFF                    MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
014D'   C9                      RET             ;DONE
                        ;
014E'   DB97            SELDSK: IN      HDCCSR  ;GET STATUS REGISTER
0150'   3C                      INR     A       ;CONTROLLER PRESENT?
0151'   C8                      RZ              ;IF NOT, DONE
0152'   DD7E01                  MOV     A,PDRDRV(X)  ;ELSE, GET REQUESTED DRIVE
0155'   E602                    ANI     2       ;EXTRACT PHYSICAL DRIVE NUMBER
0157'   87                      ADD     A       ;SHIFT DRIVE NUMBER LEFT
0158'   87                      ADD     A
0159'   F6A0                    ORI     0A0H    ;SET ERROR CORRECTION/SECTOR SIZE
015B'   D396                    OUT     HDCSDH  ;OUTPUT SIZE/DRIVE/HEAD
015D'   DB97                    IN      HDCCSR  ;GET STATUS REGISTER
015F'   E640                    ANI     RDYMSK  ;DRIVE READY?
0161'   C9                      RET             ;DONE
                        ;
0162'   21 001A"        WTINT:  LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
0165'   CD 0000:06              CALL    WAIT#   ;WAIT FOR OPERATION TO COMPLETE
0168'   3A 000E"                LDA     INTCST  ;GET INTERRUPT COMPLETION STATUS
016B'   E601                    ANI     ERRMSK  ;ANY ERRORS?
016D'   C9                      RET             ;DONE
                        ;
016E'   ED73 0000:08    DSKISR: SSPD    INTSP#  ;SAVE STACK POINTER
0172'   31 0000:09              LXI     SP,INTSTK#  ;SET UP AUXILLIARY STACK
0175'   F5                      PUSH    PSW     ;SAVE REGISTERS
0176'   C5                      PUSH    B
0177'   D5                      PUSH    D
0178'   E5                      PUSH    H
0179'   DB97                    IN      HDCCSR  ;GET INTERRUPT COMPLETION STATUS
017B'   32 000E"                STA     INTCST  ;SAVE INTERRUPT COMPLETION STATUS
017E'   21 001A"                LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
0181'   CD 0000:07              CALL    SIGNAL# ;SIGNAL PROCESS AS READY
0184'   E1                      POP     H       ;RESTORE REGISTERS
0185'   D1                      POP     D
0186'   C1                      POP     B
0187'   F1                      POP     PSW
0188'   ED7B 0000:08            LSPD    INTSP#  ;RESTORE STACK POINTER
```

```
   018C'   FB                     EI              ;ENABLE INTERRUPTS
   018D'   C9                     RET             ;DONE
                          ;
                          .PRGEND
```

```
                        ;
                        ; VERSION: 05/01/84
                        ;
                        .IDENT  MCDMAS          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
0000                    RESET   = 0             ;RESET SLAVE PROCESSOR
0001                    INT     = 1             ;INTERRUPT SLAVE PROCESSOR
0003                    RESREQ  = 3             ;RESET SLAVE PROCESSOR REQUEST
                        ;
0000                    SPWOUT  = 0             ;SLAVE PROCESSOR WAITING ON OUTPUT
0001                    SPWIN   = 1             ;SLAVE PROCESSOR WAITING ON INPUT
0002                    OVRRUN  = 2             ;SLAVE PROCESSOR OVERRUN
0003                    REQEST  = 3             ;SLAVE PROCESSOR REQUEST
                        ;
0000                    RB      = 0             ;RECEIVE BLINDED (FLGTBL)
0001                    RR      = 1             ;RECEIVE REQUEST (FLGTBL)
0002                    RA      = 2             ;RECEIVE ACTIVATED (FLGTBL)
0003                    SW      = 3             ;SEND WAITING (FLGTBL)
0004                    TO      = 4             ;TIMEOUT FLAG (FLGTBL)
0005                    DN      = 5             ;SLAVE IS DOWN (FLGTBL)
0006                    DL      = 6             ;DOWNLOAD IN PROGRESS (FLGTBL)
0007                    ED      = 7             ;END OF DOWNLOAD (FLGTBL)
                        ;
0000"                           .LOC    .DATA.# ;LOCATE IN DATA AREA
                        ;
0000"   02              NMBSLP::.BYTE   2       ;NUMBER OF SLAVES
                        ;
0001"   00              CKTSLP::.BYTE   00H     ;SLAVE CIRCUIT NUMBER
                        ;
0002"                   PATSLP::                ;SLAVE PORT ADDRESS TABLE
0002"   20                      .BYTE   20H
0003"   22                      .BYTE   22H
0004"   24                      .BYTE   24H
0005"   26                      .BYTE   26H
0006"   28                      .BYTE   28H
0007"   2A                      .BYTE   2AH
0008"   2C                      .BYTE   2CH
0009"   2E                      .BYTE   2EH
000A"   30                      .BYTE   30H
000B"   32                      .BYTE   32H
000C"   34                      .BYTE   34H
000D"   36                      .BYTE   36H
000E"   38                      .BYTE   38H
000F"   3A                      .BYTE   3AH
0010"   3C                      .BYTE   3CH
0011"   3E                      .BYTE   3EH
                        ;
0012"                   SSTSLP::                ;SLAVE SUFFIX LETTER TABLE
0012"   20                      .BYTE   ASP
0013"   20                      .BYTE   ASP
0014"   20                      .BYTE   ASP
0015"   20                      .BYTE   ASP
```

```
0016"   20                        .BYTE   ASP
0017"   20                        .BYTE   ASP
0018"   20                        .BYTE   ASP
0019"   20                        .BYTE   ASP
001A"   20                        .BYTE   ASP
001B"   20                        .BYTE   ASP
001C"   20                        .BYTE   ASP
001D"   20                        .BYTE   ASP
001E"   20                        .BYTE   ASP
001F"   20                        .BYTE   ASP
0020"   20                        .BYTE   ASP
0021"   20                        .BYTE   ASP
                         ;
0022"   00               RCVSLV: .BYTE   0       ;RECEIVE SLAVE PROCESSOR NUMBER
0023"   00               SNDSLV: .BYTE   0       ;SEND SLAVE PROCESSOR NUMBER
0024"   00               MAXLEN: .BYTE   0       ;MAXIMUM MESSAGE LENGTH
0025"   00               NITCNT: .BYTE   0       ;INITIALIZATION COUNT
0026"   000000000000     FLGTBL: .BYTE   [16]0   ;FLAG TABLE
0036"   000000000000     TICTBL: .BYTE   [16]0   ;TICK COUNT TABLE
0046"   000000000000     ERRTBL: .BYTE   [16]0   ;ERROR COUNT TABLE
                         ;
0056"   0000             RCVSPH: .WORD   0       ;RECEIVE MESSAGE SEMAPHORE
0058"   0058"            ..RMHD: .WORD   ..RMHD
005A"   0058"                    .WORD   ..RMHD
                         ;
005C"   0001             SMXSPH: .WORD   1       ;SEND MUTUAL EXCLUSION SEMAPHORE
005E"   005E"            ..MXHD: .WORD   ..MXHD
0060"   005E"                    .WORD   ..MXHD
                         ;
0062"   0000             SNDSPH: .WORD   0       ;SEND MESSAGE SEMAPHORE
0064"   0064"            ..SMHD: .WORD   ..SMHD
0066"   0064"                    .WORD   ..SMHD
                         ;
0068"   000000000000     REQBLK: .BYTE   [10]0   ;REQUEST BLOCK
                         ;
0000:04                          .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                         ;
0000:04 3A 0000"         CKTIN%::LDA     NMBSLP  ;GET NUMBER OF SLAVE PROCESSORS
0003:04 32 0025"                 STA     NITCNT  ;SET INITIALIZATION COUNT
0006:04 C9                       RET             ;DONE
                         ;
0000'                            .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                         ;
0000'   79               CKTDR%::MOV     A,C     ;GET FUNCTION NUMBER
0001'   B7                       ORA     A       ;FUNCTION NUMBER=0?
0002'   2805                     JRZ     RCVMSG  ;IF SO, CONTINUE
0004'   3D                       DCR     A       ;FUNCTION NUMBER=1?
0005'   CA 0094'                 JZ      SNDMSG  ;IF SO, CONTINUE
0008'   C9                       RET             ;ELSE, DONE
                         ;
0009'   3A 0025"         RCVMSG: LDA     NITCNT  ;GET INITIALIZATION COUNT
000C'   3D                       DCR     A       ;DECREMENT INITIALIZATION COUNT
000D'   F2 006D'                 JP      ..ERR1  ;IF POSITIVE, CONTINUE
0010'   13                       INX     D       ;ELSE, ADVANCE PAST LINK POINTERS
```

```
0011'   13                    INX     D
0012'   13                    INX     D
0013'   13                    INX     D
0014'   1A                    LDAX    D       ;GET MAXIMUM MESSAGE LENGTH
0015'   32 0024"              STA     MAXLEN  ;SAVE MAXIMUM MESSAGE LENGTH
0018'   D5                    PUSH    D       ;SAVE MESSAGE BUFFER ADDRESS
0019'   11 01EC'    ..RCVL:   LXI     D,RCVPOL  ;GET RECEIVE POLL ROUTINE
001C'   CD 0000:05            CALL    LNKPOL# ;LINK POLL ROUTINE ON POLL LIST
001F'   21 0056"              LXI     H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
0022'   CD 0000:06            CALL    WAIT#   ;WAIT FOR REQUEST
0025'   CD 02B7'              CALL    RMCOM   ;DO COMMON SETUP
0028'   CB46                  BIT     RB,M    ;RECEIVE BLINDED FLAG SET?
002A'   E1                    POP     H       ;RESTORE MESSAGE BUFFER ADDRESS
002B'   2015                  JRNZ    ..RCV   ;IF RECEIVE BLINDED SET, CONTINUE
002D'   CD 0248'              CALL    INTSLP  ;ELSE, INTERRUPT SLAVE PROCESSOR
0030'   3841                  JRC     ..ERR2  ;IF ERROR, CONTINUE
0032'   F5                    PUSH    PSW     ;ELSE, SAVE REQUEST BIT STATUS
0033'   0D                    DCR     C       ;C=SLAVE PROCESSOR DATA PORT
0034'   3E15                  MVI     A,ANAK  ;GET ASCII NACK
0036'   ED79                  OUTP    A       ;OUTPUT NACK TO SLAVE PROCESSOR
0038'   0C                    INR     C       ;C=SLAVE PROCESSOR STATUS PORT
0039'   F1                    POP     PSW     ;RESTORE REQUEST BIT STATUS
003A'   2006                  JRNZ    ..RCV   ;IF REQUEST BIT SET, CONTINUE
003C'   E5                    PUSH    H       ;ELSE, SAVE MESSAGE BUFFER ADDRESS
003D'   CD 007C'              CALL    ..SSTC  ;SIGNAL SEND/SET TICK COUNT
0040'   18D7                  JMPR    ..RCVL  ;CONTINUE
0042'   3E08        ..RCV:    MVI     A,1<RESREQ  ;GET RESET REQUEST BIT
0044'   ED79                  OUTP    A       ;RESET SLAVE PROCESSOR REQUEST BIT
0046'   0601                  MVI     B,1     ;GET LENGTH OF MESSAGE LENGTH
0048'   CD 0279'              CALL    RCVSLP  ;RECEIVE MESSAGE LENGTH
004B'   2026                  JRNZ    ..ERR2  ;IF ERROR, CONTINUE
004D'   2B                    DCX     H       ;ELSE, BACK UP TO MESSAGE LENGTH
004E'   7E                    MOV     A,M     ;GET MESSAGE LENGTH
004F'   B7                    ORA     A       ;MESSAGE LENGTH=0?
0050'   2821                  JRZ     ..ERR2  ;IF SO, CONTINUE
0052'   3A 0024"              LDA     MAXLEN  ;ELSE, GET MAXIMUM MESSAGE LENGTH
0055'   BE                    CMP     M       ;MAXIMUM MESSAGE LENGTH EXCEEDED?
0056'   381B                  JRC     ..ERR2  ;IF SO, CONTINUE
0058'   46                    MOV     B,M     ;ELSE, GET MESSAGE LENGTH
0059'   23                    INX     H       ;RESTORE MESSAGE BUFFER ADDRESS
005A'   05                    DCR     B       ;DECREMENT MESSAGE LENGTH
005B'   2816                  JRZ     ..ERR2  ;IF MESSAGE LENGTH=0, CONTINUE
005D'   CD 0279'              CALL    RCVSLP  ;ELSE, RECEIVE REMAINDER OF MESSAGE
0060'   2011                  JRNZ    ..ERR2  ;IF ERROR, CONTINUE
0062'   CD 007C'              CALL    ..SSTC  ;ELSE, SIGNAL SEND/SET TICK COUNT
0065'   3A 0022"              LDA     RCVSLV  ;GET RECEIVE SLAVE PROCESSOR NUMBER
0068'   CD 02CD'              CALL    RERCNT  ;RESET ERROR COUNT
006B'   AF                    XRA     A       ;SET RETURN CODE=0
006C'   C9                    RET             ;DONE
006D'   32 0025"    ..ERR1:   STA     NITCNT  ;UPDATE INITIALIZATION COUNT
0070'   32 0022"              STA     RCVSLV  ;SET RECIEVE SLAVE PROCESSOR NUMBER
0073'   CD 007C'    ..ERR2:   CALL    ..SSTC  ;SIGNAL SEND/SET TICK COUNT
0076'   3A 0022"              LDA     RCVSLV  ;GET RECEIVE SLAVE PROCESSOR NUMBER
0079'   C3 0161'              JMP     ERRCOM  ;CONTINUE
```

```
007C'   3A 0022"      ..SSTC: LDA     RCVSLV  ;GET RECEIVE SLAVE PROCESSOR NUMBER
007F'   CD 029E'              CALL    SETTC   ;SET TICK COUNT
0082'   CD 02C8'              CALL    GETFTA  ;GET FLAG TABLE ADDRESS
0085'   CB96                  RES     RA,M    ;RESET RECEIVE ACTIVATED FLAG
0087'   CB86                  RES     RB,M    ;RESET RECEIVE BLINED FLAG
0089'   CB5E                  BIT     SW,M    ;SEND WAITING FLAG SET?
008B'   C8                    RZ              ;IF NOT, DONE
008C'   CB9E                  RES     SW,M    ;ELSE, RESET SEND WAITING FLAG
008E'   21 0062"              LXI     H,SNDSPH  ;GET SEND MESSAGE SEMAPHORE
0091'   C3 0000:07            JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                      ;
0094'   EB            SNDMSG: XCHG            ;MESSAGE BUFFER ADDRESS TO HL-REG
0095'   23                    INX     H       ;ADVANCE PAST LINK POINTERS
0096'   23                    INX     H
0097'   23                    INX     H
0098'   23                    INX     H
0099'   23                    INX     H       ;ADVANCE TO MESSAGE DESTINATION ID
009A'   7E                    MOV     A,M     ;GET MESSAGE DESTINATION ID
009B'   3D                    DCR     A       ;DECREMENT MESSAGE DESTINATION ID
009C'   2B                    DCX     H       ;BACK UP TO MESSAGE LENGTH
009D'   34                    INR     M
009E'   35                    DCR     M       ;MESSAGE LENGTH=0?
009F'   CA 012F'              JZ      ..MLO   ;IF SO, CONTINUE
00A2'   E5                    PUSH    H       ;ELSE, SAVE MESSAGE BUFFER ADDRESS
00A3'   21 005C"              LXI     H,SMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
00A6'   F5            ..SWTL: PUSH    PSW     ;SAVE SEND SLAVE PROCESSOR NUMBER
00A7'   CD 0000:06            CALL    WAIT#   ;WAIT ON EXCLUSION/RECEIVED MESSAGE
00AA'   F1                    POP     PSW     ;RESTORE SLAVE PROCESSOR NUMBER
00AB'   32 0023"              STA     SNDSLV  ;SET SEND SLAVE PROCESSOR NUMBER
00AE'   CD 02BC'              CALL    SMCOM   ;DO COMMON SETUP
00B1'   CB56                  BIT     RA,M    ;RECEIVE ACTIVATE FLAG SET?
00B3'   C2 0140'              JNZ     ..SWT   ;IF SO, CONTINUE
00B6'   CB6E                  BIT     DN,M    ;SLAVE IS DOWN FLAG SET?
00B8'   2813                  JRZ     ..SND   ;IF NOT, CONTINUE
00BA'   CB76                  BIT     DL,M    ;DOWNLOAD IN PROGRESS FLAG SET?
00BC'   CA 0151'              JZ      ..ERR1  ;IF NOT, CONTINUE
00BF'   E3                    XTHL            ;RESTORE MESSAGE BUFFER ADDRESS
00C0'   7E                    MOV     A,M     ;GET MESSAGE LENGTH
00C1'   E3                    XTHL            ;RESTORE FLAGS ADDRESS
00C2'   FE8B                  CPI     11+128  ;DOWNLOAD RECORD MESSAGE?
00C4'   2807                  JRZ     ..SND   ;IF SO, CONTINUE
00C6'   FE0C                  CPI     11+1    ;END OF DOWNLOAD MESSAGE?
00C8'   C2 0151'              JNZ     ..ERR1  ;IF NOT, CONTINUE
00CB'   CBFE                  SET     ED,M    ;ELSE, SET END OF DOWNLOAD FLAG
00CD'   CD 0248'      ..SND:  CALL    INTSLP  ;INTERRUPT SLAVE PROCESSOR
00D0'   DA 015A'              JC      ..ERR2  ;IF ERROR, CONTINUE
00D3'   2064                  JRNZ    ..RCV   ;IF REQUEST BIT SET, CONTINUE
00D5'   0D                    DCR     C       ;C=SLAVE PROCESSOR DATA PORT
00D6'   3E06                  MVI     A,AACK  ;GET ASCII ACK
00D8'   ED79                  OUTP    A       ;OUTPUT ACK TO SLAVE PROCESSOR
00DA'   CBC6                  SET     RB,M    ;SET RECEIVE BLINDED FLAG
00DC'   CBA6                  RES     TO,M    ;RESET TIMEOUT FLAG
00DE'   11 0229'              LXI     D,SNDPOL  ;GET SEND POLL ROUTINE
00E1'   CD 0000:05            CALL    LNKPOL# ;LINK POLL ROUTINE ON POLL LIST
```

```
00E4'   3A 0023"                LDA     SNDSLV  ;GET SEND SLAVE PROCESSOR NUMBER
00E7'   F5                      PUSH    PSW     ;SAVE SEND SLAVE PROCESSOR NUMBER
00E8'   CD 029E'                CALL    SETTC   ;SET TICK COUNT
00EB'   21 0062"                LXI     H,SNDSPH  ;GET SEND MESSAGE SEMAPHORE
00EE'   CD 0000:06              CALL    WAIT#   ;WAIT FOR SLAVE TO RECEIVE
00F1'   F1                      POP     PSW     ;RESTORE SLAVE PROCESSOR NUMBER
00F2'   32 0023"                STA     SNDSLV  ;SET SEND SLAVE PROCESSOR NUMBER
00F5'   CD 02BC'                CALL    SMCOM   ;DO COMMON SETUP
00F8'   CB86                    RES     RB,M    ;RESET RECEIVE BLINDED FLAG
00FA'   CB66                    BIT     TO,M    ;TIMEOUT FLAG SET?
00FC'   205C                    JRNZ    ..ERR2  ;IF SO, CONTINUE
00FE'   3E08                    MVI     A,1<RESREQ  ;ELSE, GET RESET REQUEST BIT
0100'   ED79                    OUTP    A       ;RESET SLAVE PROCESSOR REQUEST BIT
0102'   E1                      POP     H       ;RESTORE MESSAGE BUFFER ADDRESS
0103'   46                      MOV     B,M     ;GET MESSAGE LENGTH
0104'   05                      DCR     B       ;DECREMENT MESSAGE LENGTH
0105'   C5                      PUSH    B       ;SAVE MESSAGE LENGTH
0106'   0601                    MVI     B,1     ;GET LENGTH OF MESSAGE LENGTH
0108'   CD 0268'                CALL    SNDSLP  ;SEND MESSAGE LENGTH TO SLAVE
010B'   C1                      POP     B       ;RESTORE MESSAGE LENGTH
010C'   CC 0268'                CZ      SNDSLP  ;IF NO ERROR, SEND MESSAGE
010F'   204A                    JRNZ    ..ERR3  ;IF ERROR, CONTINUE
0111'   CD 014B'                CALL    ..RMXC  ;ELSE, RELEASE MUTUAL EXCLUSION
0114'   3A 0023"                LDA     SNDSLV  ;GET SEND SLAVE PROCESSOR NUMBER
0117'   CD 029E'                CALL    SETTC   ;SET TICK COUNT
011A'   CD 02C8'                CALL    GETFTA  ;GET FLAG TABLE ADDRESS
011D'   CB7E                    BIT     ED,M    ;END OF DOWNLOAD FLAG SET?
011F'   2806                    JRZ     ..NED   ;IF NOT, CONTINUE
0121'   CBBE                    RES     ED,M    ;ELSE, RESET END OF DOWNLOAD FLAG
0123'   CBB6                    RES     DL,M    ;RESET DOWNLOAD IN PROGRESS FLAG
0125'   CBAE                    RES     DN,M    ;RESET SLAVE IS DOWN FLAG
0127'   3A 0023"        ..NED:  LDA     SNDSLV  ;GET SEND SLAVE PROCESSOR NUMBER
012A'   CD 02CD'                CALL    RERCNT  ;RESET ERROR COUNT
012D'   AF                      XRA     A       ;SET RETURN CODE=0
012E'   C9                      RET             ;DONE
012F'   4F              ..MLO:  MOV     C,A     ;SLAVE PROCESSOR NUMBER TO C-REG
0130'   C5                      PUSH    B       ;SAVE SLAVE PROCESSOR NUMBER
0131'   CD 017A'                CALL    RESSLV  ;RESET SLAVE PROCESSOR
0134'   C1                      POP     B       ;RESTORE SLAVE PROCESSOR NUMBER
0135'   C8                      RZ              ;IF NO ERROR, DONE
0136'   79                      MOV     A,C     ;ELSE, GET SLAVE PROCESSOR NUMBER
0137'   1828                    JMPR    ERRCOM  ;CONTINUE
0139'   CBCE            ..RCV:  SET     RR,M    ;SET RECEIVE REQUEST FLAG
013B'   0D                      DCR     C       ;C=SLAVE PROCESSOR DATA PORT
013C'   3E15                    MVI     A,ANAK  ;GET ASCII NACK
013E'   ED79                    OUTP    A       ;OUTPUT NACK TO SLAVE PROCESSOR
0140'   CBDE            ..SWT:  SET     SW,M    ;SET SEND WAITING FLAG
0142'   21 0062"                LXI     H,SNDSPH  ;GET SEND MESSAGE SEMAPHORE
0145'   3A 0023"                LDA     SNDSLV  ;GET SEND SLAVE PROCESSOR NUMBER
0148'   C3 00A6'                JMP     ..SWTL  ;CONTINUE
014B'   21 005C"        ..RMXC: LXI     H,SMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
014E'   C3 0000:07              JMP     SIGNAL# ;RELEASE MUTUAL EXCLUSION
0151'   E1              ..ERR1: POP     H       ;RESTORE MESSAGE BUFFER ADDRESS
0152'   CD 014B'                CALL    ..RMXC  ;RELEASE MUTUAL EXCLUSION
```

```
0155'   CD 02BC'                CALL    SMCOM   ;DO COMMON SETUP
0158'   1818                    JMPR    ERRXIT  ;CONTINUE
015A'   E1              ..ERR2: POP     H       ;RESTORE MESSAGE BUFFER ADDRESS
015B'   CD 014B'        ..ERR3: CALL    ..RMXC  ;RELEASE MUTUAL EXCLUSION
015E'   3A 0023"                LDA     SNDSLV  ;GET SEND SLAVE PROCESSOR NUMBER
                        ;
0161'   CD 02BF'        ERRCOM: CALL    SRMCOM  ;DO COMMON SETUP
0164'   CBC6                    SET     RB,M    ;SET RECEIVE BLINDED FLAG
0166'   CB8E                    RES     RR,M    ;RESET RECEIVE REQUEST FLAG
0168'   CBEE                    SET     DN,M    ;SET SLAVE IS DOWN FLAG
016A'   CBB6                    RES     DL,M    ;RESET DOWNLOAD IN PROGRESS FLAG
016C'   CBBE                    RES     ED,M    ;RESET END OF DOWNLOAD FLAG
016E'   CD 02D6'                CALL    GETECA  ;GET ERROR COUNT ADDRESS
0171'   34                      INR     M       ;INCREMENT ERROR COUNT
                        ;
0172'   1C              ERRXIT: INR     E       ;INCREMENT SLAVE PROCESSOR NUMBER
0173'   3A 0001"                LDA     CKTSLP  ;GET SLAVE CIRCUIT NUMBER
0176'   57                      MOV     D,A     ;SLAVE CIRCUIT NUMBER TO D-REG
0177'   3EFF                    MVI     A,0FFH  ;SET RETURN CODE=0FFH
0179'   C9                      RET             ;DONE
                        ;
017A'   32 0023"        RESSLV: STA     SNDSLV  ;SET SEND SLAVE PROCESSOR NUMBER
017D'   CD 02BF'                CALL    SRMCOM  ;DO COMMON SETUP
0180'   7E                      MOV     A,M     ;GET FLAGS
0181'   E640                    ANI     1<DL    ;EXTRACT DOWNLOAD IN PROGRESS FLAG
0183'   EE40                    XRI     1<DL    ;DOWNLOAD IN PROGRESS FLAG SET?
0185'   C8                      RZ              ;IF SO, DONE
0186'   E5                      PUSH    H       ;ELSE, SAVE FLAGS ADDRESS
0187'   CD 02D6'                CALL    GETECA  ;GET ERROR COUNT ADDRESS
018A'   7E                      MOV     A,M     ;GET ERROR COUNT
018B'   3C                      INR     A       ;ERROR COUNT=255?
018C'   E1                      POP     H       ;RESTORE FLAGS ADDRESS
018D'   C8                      RZ              ;IF ERROR COUNT=255, DONE
018E'   CBF6                    SET     DL,M    ;SET DOWNLOAD IN PROGRESS FLAG
0190'   CB86                    RES     RB,M    ;RESET RECEIVE BLINDED FLAG
0192'   3A 0001"                LDA     CKTSLP  ;GET SLAVE CIRCUIT NUMBER
0195'   67                      MOV     H,A     ;SLAVE CIRCUIT NUMBER TO H-REG
0196'   2E00                    MVI     L,0     ;SET CIRCUIT NODE ADDRESS TO 0
0198'   22 0000:08              SHLD    DIDSLP# ;SET SLAVE PROCESSOR DESTINATION ID
019B'   7B                      MOV     A,E     ;GET SLAVE PROCESSOR NUMBER
019C'   3C                      INR     A       ;INCREMENT SLAVE PROCESSOR NUMBER
019D'   6F                      MOV     L,A     ;SLAVE PROCESSOR NUMBER TO L-REG
019E'   22 0000:09              SHLD    SIDSLP# ;SET SLAVE PROCESSOR SOURCE ID
01A1'   21 0012"                LXI     H,SSTSLP  ;GET SLAVE SUFFIX LETTER TABLE
01A4'   19                      DAD     D       ;INDEX INTO SLAVE SUFFIX TABLE
01A5'   7E                      MOV     A,M     ;GET SLAVE O/S SUFFIX LETTER
01A6'   32 0000:0A              STA     SSLSLP# ;SET SLAVE SUFFIX LETTER
01A9'   3E01                    MVI     A,1<RESET  ;GET RESET BIT
01AB'   ED79                    OUTP    A       ;RESET SLAVE PROCESSOR
01AD'   0600                    MVI     B,0     ;GET DELAY COUNT
01AF'   10FE            ..DLYL: DJNZ    ..DLYL  ;DELAY
01B1'   AF                      XRA     A
01B2'   ED79                    OUTP    A       ;CLEAR RESET BIT
01B4'   21 0068"                LXI     H,REQBLK  ;GET REQUEST BLOCK AREA
```

```
01B7'   060A                    MVI     B,10      ;GET REQUEST BLOCK LENGTH
01B9'   CD 0279'                CALL    RCVSLP    ;RECEIVE REQUEST BLOCK
01BC'   C0                      RNZ               ;IF ERROR, DONE
01BD'   3E08                    MVI     A,1<RESREQ  ;ELSE, GET RESET REQUEST BIT
01BF'   ED79                    OUTP    A         ;RESET SLAVE PROCESSOR REQUEST BIT
01C1'   21 0000:0B              LXI     H,LD1SLP#  ;GET INTERMEDIATE BOOT CODE
01C4'   0680                    MVI     B,128     ;GET INTERMEDIATE BOOT CODE LENGTH
01C6'   CD 0268'                CALL    SNDSLP    ;SEND INTERMEDIATE BOOT CODE
01C9'   C0                      RNZ               ;IF ERROR, DONE
01CA'   21 0000:0C              LXI     H,LADSLP#  ;ELSE, GET LOAD ADDRESS/LENGTH
01CD'   0604                    MVI     B,4       ;GET LENGTH OF LOAD ADDRESS/LENGTH
01CF'   CD 0268'                CALL    SNDSLP    ;SEND LOAD ADDRESS/LENGTH
01D2'   C0                      RNZ               ;IF ERROR, DONE
01D3'   ED5B 0000:0D            LDED    LENSLP#   ;ELSE, GET LOAD LENGTH
01D7'   1B                      DCX     D         ;DECREMENT LOAD LENGTH
01D8'   1C                      INR     E         ;INCREMENT LSB OF LOAD LENGTH
01D9'   43                      MOV     B,E       ;LSB OF LOAD LENGTH TO B-REG
01DA'   D5              ..LDL:  PUSH    D         ;SAVE MSB OF LOAD LENGTH
01DB'   CD 0268'                CALL    SNDSLP    ;SEND UP TO 256 BYTES OF BOOT CODE
01DE'   D1                      POP     D         ;RESTORE MSB OF LOAD LENGTH
01DF'   C0                      RNZ               ;IF ERROR, DONE
01E0'   15                      DCR     D         ;ELSE, DECREMENT MSB OF LOAD LENGTH
01E1'   F2 01DA'                JP      ..LDL     ;IF MORE TO SEND, CONTINUE
01E4'   3A 0023"                LDA     SNDSLV    ;GET SEND SLAVE PROCESSOR NUMBER
01E7'   CD 029E'                CALL    SETTC     ;SET TICK COUNT
01EA'   AF                      XRA     A         ;SET ZERO FLAG
01EB'   C9                      RET               ;DONE
                        ;
01EC'   0000            RCVPOL: .WORD   0         ;SUCCESSOR LINK POINTER
01EE'   0000                    .WORD   0         ;PREDECESSOR LINK POINTER
                        ;
01F0'   3A 0000"                LDA     NMBSLP    ;GET NUMBER OF SLAVES
01F3'   B7                      ORA     A         ;NUMBER OF SLAVES=0?
01F4'   C8                      RZ                ;IF SO, DONE
01F5'   3D                      DCR     A         ;ELSE, CALCULATE MAX SLAVE NUMBER
01F6'   E60F                    ANI     0FH       ;LIMIT TO 16 SLAVE PROCESSORS
01F8'   21 0022"                LXI     H,RCVSLV  ;GET CURRENT SLAVE NUMBER
01FB'   34                      INR     M         ;INCREMENT CURRENT SLAVE NUMBER
01FC'   BE                      CMP     M         ;VALID SLAVE PROCESSOR NUMBER?
01FD'   3002                    JRNC    ..VSPN    ;IF SO, CONTINUE
01FF'   3600                    MVI     M,0       ;ELSE, SET SLAVE PROCESSOR NUMBER=0
0201'   CD 02B7'        ..VSPN: CALL    RMCOM     ;DO COMMON SETUP
0204'   CB46                    BIT     RB,M      ;RECEIVE BLINDED FLAG SET?
0206'   C0                      RNZ               ;IF SO, DONE
0207'   CB4E                    BIT     RR,M      ;RECEIVE REQUEST FLAG SET?
0209'   2006                    JRNZ    ..RR      ;IF SO, CONTINUE
020B'   ED78                    INP     A         ;ELSE, GET SLAVE PROCESSOR STATUS
020D'   CB5F                    BIT     REQEST,A  ;REQUEST BIT SET?
020F'   2806                    JRZ     ..CTC     ;IF SO, CONTINUE
0211'   CB8E            ..RR:   RES     RR,M      ;ELSE, RESET RECEIVE REQUEST FLAG
0213'   CBC6                    SET     RB,M      ;SET RECEIVE BLINDED FLAG
0215'   1804                    JMPR    ..SIG     ;CONTINUE
0217'   CD 02AA'        ..CTC:  CALL    CHKCTC    ;CHECK CURRENT TICK COUNT
021A'   D8                      RC                ;IF TIME NOT EXPIRED, DONE
```

```
021B'   CBD6            ..SIG:  SET     RA,M     ;ELSE, SET RECEIVE ACTIVATED FLAG
021D'   21 01EC'                LXI     H,RCVPOL  ;GET RECEIVE POLL ROUTINE
0220'   CD 0000:0E              CALL    UNLINK# ;UNLINK POLL ROUTINE FROM POLL LIST
0223'   21 0056"                LXI     H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
0226'   C3 0000:07              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
0229'   0000            SNDPOL: .WORD   0        ;SUCCESSOR LINK POINTER
022B'   0000                    .WORD   0        ;PREDECESSOR LINK POINTER
                        ;
022D'   CD 02BC'                CALL    SMCOM    ;DO COMMON SETUP
0230'   ED78                    INP     A        ;GET SLAVE PROCESSOR STATUS
0232'   CB5F                    BIT     REQEST,A  ;REQUEST BIT SET?
0234'   2006                    JRNZ    ..SIG    ;IF SO, CONTINUE
0236'   CD 02AA'                CALL    CHKCTC   ;ELSE, CHECK CURRENT TICK COUNT
0239'   D8                      RC               ;IF TIME NOT EXPIRED, DONE
023A'   CBE6                    SET     TO,M     ;ELSE, SET TIMEOUT FLAG
023C'   21 0229'        ..SIG:  LXI     H,SNDPOL  ;GET SEND POLL ROUTINE
023F'   CD 0000:0E              CALL    UNLINK# ;UNLINK POLL ROUTINE FROM POLL LIST
0242'   21 0062"                LXI     H,SNDSPH  ;GET SEND MESSAGE SEMAPHORE
0245'   C3 0000:07              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
0248'   3E02            INTSLP: MVI     A,1<INT ;GET INTERRUPT BIT
024A'   ED79                    OUTP    A        ;INTERRUPT SLAVE PROCESSOR
024C'   AF                      XRA     A
024D'   ED79                    OUTP    A        ;CLEAR INTERRUPT BIT
024F'   0600                    MVI     B,0      ;GET TIMEOUT LOOP COUNT
0251'   ED78            ..WTIL: INP     A        ;GET SLAVE PROCESSOR STATUS
0253'   CB47                    BIT     SPWOUT,A  ;WAITING ON OUTPUT BIT SET?
0255'   2004                    JRNZ    ..INT    ;IF SO, CONTINUE
0257'   10F8                    DJNZ    ..WTIL   ;ELSE, WAIT
0259'   37                      STC              ;SET CARRY FLAG
025A'   C9                      RET              ;DONE
025B'   0D              ..INT:  DCR     C        ;C=SLAVE PROCESSOR DATA PORT
025C'   ED78                    INP     A        ;GET DATA BYTE FROM SLAVE PROCESSOR
025E'   0C                      INR     C        ;C=SLAVE PROCESSOR STATUS PORT
025F'   FE06                    CPI     AACK     ;RESPONSE=ACK?
0261'   37                      STC              ;PRESET CARRY FLAG
0262'   C0                      RNZ              ;IF RESPONSE NOT=ACK, DONE
0263'   ED78                    INP     A        ;ELSE, GET SLAVE PROCESSOR STATUS
0265'   E608                    ANI     1<REQEST  ;REQUEST BIT SET?
0267'   C9                      RET              ;DONE
                        ;
0268'   1602            SNDSLP: MVI     D,1<SPWIN  ;GET WAITING ON INPUT BIT
026A'   CD 0291'                CALL    WTRDY    ;WAIT FOR SLAVE PROCESSOR READY
026D'   C0                      RNZ              ;IF ERROR, DONE
026E'   0D                      DCR     C        ;C=SLAVE PROCESSOR DATA PORT
026F'   00              ..SNDL: NOP              ;DELAY
0270'   00                      NOP
0271'   00                      NOP
0272'   00                      NOP
0273'   EDA3                    OUTI             ;OUTPUT MESSAGE BYTE
0275'   20F8                    JRNZ    ..SNDL   ;CONTINUE
0277'   180F                    JMPR    SRCXIT   ;CONTINUE
                        ;
```

```
0279'   1601            RCVSLP: MVI     D,1<SPWOUT  ;GET WAITING ON OUTPUT BIT
027B'   CD 0291'                CALL    WTRDY   ;WAIT FOR SLAVE PROCESSOR READY
027E'   C0                      RNZ             ;IF ERROR, DONE
027F'   0D                      DCR     C       ;C=SLAVE PROCESSOR DATA PORT
0280'   00              ..RCVL: NOP             ;DELAY
0281'   00                      NOP
0282'   00                      NOP
0283'   00                      NOP
0284'   EDA2                    INI             ;INPUT MESSAGE BYTE
0286'   20F8                    JRNZ    ..RCVL  ;CONTINUE
                        ;
0288'   0C              SRCXIT: INR     C       ;C=SLAVE PROCESSOR STATUS PORT
0289'   ED78                    INP     A       ;GET SLAVE PROCESSOR STATUS
028B'   E604                    ANI     1<OVRRUN  ;OVERRUN ERROR?
028D'   C8                      RZ              ;IF NOT, DONE
028E'   3EFF                    MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
0290'   C9                      RET             ;DONE
                        ;
0291'   1E00            WTRDY:  MVI     E,0     ;GET TIMEOUT LOOP COUNT
0293'   ED78            ..WTRL: INP     A       ;GET SLAVE PROCESSOR STATUS
0295'   A2                      ANA     D       ;SLAVE PROCESSOR READY?
0296'   AA                      XRA     D
0297'   C8                      RZ              ;IF SO, DONE
0298'   1D                      DCR     E       ;ELSE, DECREMENT TIMEOUT COUNT
0299'   20F8                    JRNZ    ..WTRL  ;IF COUNT REMAINS, CONTINUE
029B'   AF                      XRA     A       ;ELSE, SET RETURN CODE=0FFH
029C'   3D                      DCR     A       ;WITH ZERO FLAG CLEARED
029D'   C9                      RET             ;DONE
                        ;
029E'   5F              SETTC:  MOV     E,A     ;SLAVE PROCESSOR NUMBER TO DE-REG
029F'   1600                    MVI     D,0     ;DOUBLE LENGTH
02A1'   21 0036"                LXI     H,TICTBL  ;GET TICK COUNT TABLE
02A4'   19                      DAD     D       ;INDEX INTO TICK COUNT TABLE
02A5'   3A 0000:0F              LDA     TICCNT# ;GET CURRENT TICK COUNT
02A8'   77                      MOV     M,A     ;SET TICK COUNT
02A9'   C9                      RET             ;DONE
                        ;
02AA'   E5              CHKCTC: PUSH    H       ;SAVE FLAGS ADDRESS
02AB'   21 0036"                LXI     H,TICTBL  ;GET TICK COUNT TABLE
02AE'   19                      DAD     D       ;INDEX INTO TICK COUNT TABLE
02AF'   3A 0000:0F              LDA     TICCNT# ;GET CURRENT TICK COUNT
02B2'   96                      SUB     M       ;CALC ELAPSED NUMBER OF TICKS
02B3'   E1                      POP     H       ;RESTORE FLAGS ADDRESS
02B4'   FE3C                    CPI     60      ;MINIMUM NUMBER OF TICKS ELAPSED?
02B6'   C9                      RET             ;DONE
                        ;
02B7'   3A 0022"        RMCOM:  LDA     RCVSLV  ;GET RECEIVE SLAVE PROCESSOR NUMBER
02BA'   1803                    JMPR    SRMCOM  ;CONTINUE
                        ;
02BC'   3A 0023"        SMCOM:  LDA     SNDSLV  ;GET SEND SLAVE PROCESSOR NUMBER
                        ;
02BF'   5F              SRMCOM: MOV     E,A     ;SLAVE PROCESSOR NUMBER TO DE-REG
02C0'   1600                    MVI     D,0     ;DOUBLE LENGTH
02C2'   21 0002"                LXI     H,PATSLP  ;GET SLAVE PORT ADDRESS TABLE
```

```
   02C5'   19                        DAD     D         ;CALC SLAVE PORT ADDRESS
   02C6'   4E                        MOV     C,M       ;DATA PORT ADDRESS TO C-REG
   02C7'   0C                        INR     C         ;C=STATUS PORT ADDRESS
                             ;
   02C8'   21 0026"          GETFTA: LXI     H,FLGTBL  ;GET FLAG TABLE
   02CB'   19                        DAD     D         ;INDEX INTO FLAG TABLE
   02CC'   C9                        RET               ;DONE
                             ;
   02CD'   5F                RERCNT: MOV     E,A       ;SLAVE PROCESSOR NUMBER TO DE-REG
   02CE'   1600                      MVI     D,0       ;DOUBLE LENGTH
   02D0'   CD 02D6'                  CALL    GETECA    ;GET ERROR COUNT ADDRESS
   02D3'   3600                      MVI     M,0       ;SET ERROR COUNT=0
   02D5'   C9                        RET               ;DONE
                             ;
   02D6'   21 0046"          GETECA: LXI     H,ERRTBL  ;GET ERROR COUNT TABLE
   02D9'   19                        DAD     D         ;INDEX INTO ERROR COUNT TABLE
   02DA'   C9                        RET               ;DONE
                             ;
                             .PRGEND
```

```
                        ;
                        ; VERSION: 05/01/84
                        ;
                        .IDENT  LD1SLP          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
001E                    SPDATA  = 1EH           ;SLAVE PROCESSOR DATA PORT ADDRESS
001F                    SPSTAT  = 1FH           ;SLAVE PROCESSOR STATUS/CTRL PORT
                        ;
0000                    PROMON  = 0             ;PROM ON BIT
0001                    RESPES  = 1             ;RESET PARITY ERROR STATUS BIT
0002                    RESORS  = 2             ;RESET OVERRUN ERROR STATUS BIT
0003                    REQEST  = 3             ;REQUEST BIT
                        ;
                        .DEFINE RELOC[ADDR]=[(ADDR-LD1SLP)+1000H] ;RELOCATION MACRO
                        ;
0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
0000'   F3              LD1SLP::DI              ;DISABLE INTERRUPTS
0001'   31 1000                 LXI     SP,RELOC[LD1SLP]  ;INITIALIZE STACK POINTER
0004'   3E06                    MVI     A,6
0006'   D31F                    OUT     SPSTAT  ;DISABLE PROM/CLEAR OVERRUN/REQUEST
0008'   0E1E                    MVI     C,SPDATA  ;C=SLAVE PROCESSOR DATA PORT
000A'   21 0100                 LXI     H,0100H ;GET INTERMEDIATE BOOT LOAD ADDRESS
000D'   0604                    MVI     B,4     ;GET LENGTH OF LOAD ADDRESS/LENGTH
000F'   EDB2                    INIR            ;RECEIVE LOAD ADDRESS/LENGTH
0011'   2A 0100                 LHLD    0100H   ;GET LOAD ADDRESS
0014'   E5                      PUSH    H       ;SAVE LOAD ADDRESS
0015'   ED5B 0102               LDED    0102H   ;GET LENGTH
0019'   43                      MOV     B,E     ;LSB OF LOAD LENGTH TO B-REG
001A'   7B                      MOV     A,E     ;GET LSB OF LOAD LENGTH
001B'   B7                      ORA     A       ;LSB OF LOAD LENGTH=0?
001C'   2001                    JRNZ    ..LDL   ;IF NOT, CONTINUE
001E'   15                      DCR     D       ;ELSE, DECREMENT MSB OF LOAD LENGTH
001F'   EDB2            ..LDL:  INIR            ;INPUT UP TO 256 BYTES OF BOOT CODE
0021'   15                      DCR     D       ;DECREMENT MSB OF LOAD LENGTH
0022'   F2 101F                 JP      RELOC[..LDL]  ;IF MORE TO SEND, CONTINUE
0025'   E1                      POP     H       ;ELSE, RESTORE LOAD ADRESS
0026'   E9                      PCHL            ;TRANSFER TO LOAD ADDRESS
                        ;
                                .IFL    (LD1SLP+128)-.,[
                        ;
                                .ERROR  "INTERMEDIATE BOOT CODE LENGTH > 128"
                        ;
                                        ]
                        ;
                        .PRGEND
                        ;
                        ; VERSION: 05/01/84
                        ;
                        .IDENT  LD2SLP          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
 0018                   INTD    = 18H           ;INTERRUPT CONTROLLER DATA PORT
 0019                   INTC    = 19H           ;INTERRUPT CONTROLLER CONTROL PORT
                        ;
 0010                   INTVEC  = 0010H         ;INTERRUPT VECTOR ADDRESS
                        ;
 001E                   SPDATA  = 1EH           ;SLAVE PROCESSOR DATA PORT ADDRESS
 001F                   SPSTAT  = 1FH           ;SLAVE PROCESSOR STATUS/CTRL PORT
                        ;
 0000                   PROMON  = 0             ;PROM ON BIT
 0001                   RESPES  = 1             ;RESET PARITY ERROR STATUS BIT
 0002                   RESORS  = 2             ;RESET OVERRUN ERROR STATUS BIT
 0003                   REQEST  = 3             ;REQUEST BIT
                        ;
 0007                   RCVMSG  = 7             ;RECEIVE MESSAGE FLAG (FLAGS)
                        ;
 0000'                          .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
 0000'  0100            LADSLP::.WORD   0100H   ;SLAVE PROCESSOR BOOT LOAD ADDRESS
 0002'  00C8            LENSLP::.WORD   LD2LEN  ;SLAVE PROCESSOR BOOT LOAD LENGTH
                        ;
                        .DEFINE RELOC[ADDR]=[(ADDR-LD2SLP)+0100H] ;RELOCATION MACRO
                        ;
 0004'  F3              LD2SLP: DI              ;DISABLE INTERRUPTS
 0005'  31 0100                 LXI     SP,RELOC[LD2SLP]  ;INITIALIZE STACK POINTER
 0008'  3E04                    MVI     A,1<RESORS  ;GET RESET OVERRUN ERROR STATUS
 000A'  D31F                    OUT     SPSTAT  ;DISABLE PROM/CLEAR OVERRUN/REQUEST
 000C'  ED5E                    IM2             ;SET Z80 INTERRUPT MODE 2
 000E'  AF                      XRA     A
 000F'  ED47                    STAI            ;SET INTERRUPT VECTORS TO PAGE 0
 0011'  D319                    OUT     INTC    ;RESET INT CONTROLLER
 0013'  3E80                    MVI     A,80H   ;DEFINE INT CONT MODE:
 0015'  D319                    OUT     INTC    ; FIXED PRIORITY
                                                ; INDIVIDUAL VECTOR
                                                ; INTERRUPT MODE
                                                ; GINT ACTIVE-LOW
                                                ; IREQ ACTIVE-LOW
 0017'  3EC0                    MVI     A,0C0H  ;LOAD AUTO-CLEAR REGISTER
 0019'  D319                    OUT     INTC
 001B'  3EFF                    MVI     A,0FFH  ;AUTO-CLEAR ALL CHANNELS
 001D'  D318                    OUT     INTD
 001F'  3EA9                    MVI     A,0A9H  ;ENABLE INTERRUPT CONTROLLER
 0021'  D319                    OUT     INTC
 0023'  21 01A2                 LXI     H,RELOC[LD2ISR]  ;GET INT SERVICE ADDRESS
 0026'  22 0010                 SHLD    INTVEC  ;SET INTERRUPT VECTOR
 0029'  3EE0                    MVI     A,0E0H  ;GET LOAD RESPONSE MEMORY COMMAND
 002B'  D319                    OUT     INTC    ;SEND TO INTERRUPT CONTROLLER
 002D'  3E10                    MVI     A,INTVEC  ;GET LSB OF VECTOR ADDRESS
```

```
0002F'  D318                OUT    INTD     ;LOAD INTO RESPONSE MEMORY
0031'   3E18                MVI    A,18H    ;GET CLEAR IRR & IMR CMD
0033'   D319                OUT    INTC     ;SEND TO INTERRUPT CONTROLLER
0035'   FB                  EI              ;ENABLE INTERRUPTS
0036'   21 0000             LXI    H,0      ;INITIALIZE MEMORY PARITY
0039'   11 0000             LXI    D,0
003C'   01 0000             LXI    B,0
003F'   EDB0                LDIR
0041'   3E02                MVI    A,1<RESPES  ;GET RESET PARITY ERROR STATUS
0043'   D31F                OUT    SPSTAT  ;RESET PARITY ERROR STATUS
0045'   21 01BC     ..LD2L: LXI    H,RELOC[REQMSG]  ;GET REQUEST MESSAGE
0048'   46                  MOV    B,M      ;GET MESSAGE LENGTH
0049'   CD 0195             CALL   RELOC[..WAIT]  ;WAIT FOR REQUEST TO FALL
004C'   EDB3                OUTIR           ;SEND MESSAGE
004E'   21 01BB             LXI    H,RELOC[FLAGS]  ;GET FLAGS
0051'   CBBE                RES    RCVMSG,M  ;RESET RECEIVE MESSAGE FLAG
0053'   CB7E        ..WTL1: BIT    RCVMSG,M  ;RECEIVE MESSAGE FLAG SET?
0055'   28FC                JRZ    ..WTL1   ;IF NOT, CONTINUE
0057'   21 01C8             LXI    H,RELOC[REPMSG]  ;GET REPLY MESSAGE BUFFER
005A'   CD 0195             CALL   RELOC[..WAIT]  ;WAIT FOR REQUEST TO FALL
005D'   ED40                INP    B        ;INPUT MESSAGE LENGTH
005F'   70                  MOV    M,B      ;STORE MESSAGE LENGTH IN HEADER
0060'   23                  INX    H
0061'   05                  DCR    B        ;DECREMENT MESSAGE LENGTH
0062'   EDB2                INIR            ;RECEIVE MESSAGE
0064'   3A 01C8             LDA    RELOC[REPMSG]  ;GET REPLY MESSAGE LENGTH
0067'   FE0C                CPI    MSGHL+1  ;MESSAGE LENGTH=HEADER LENGTH+1?
0069'   2812                JRZ    ..EOF    ;IF SO, CONTINUE
006B'   21 01D3             LXI    H,RELOC[REPMSG+MSGHL]  ;GET DATA ADDRESS
006E'   ED5B 01B9           LDED   RELOC[DMADDR]  ;GET DMA ADDRESS
0072'   01 0080             LXI    B,128    ;GET DATA RECORD LENGTH
0075'   EDB0                LDIR            ;MOVE DOWNLOAD RECORD INTO DMA ADDF
0077'   ED53 01B9           SDED   RELOC[DMADDR]  ;UPDATE DMA ADDRESS
007B'   18C8     .          JMPR   ..LD2L   ;CONTINUE
007D'   2B          ..EOF:  DCX    H        ;BACK UP TO O/S ID
007E'   7E                  MOV    A,M      ;GET SYSTEM DISK
007F'   21 0253             LXI    H,RELOC[OSLOAD]  ;GET SLAVE PROCESSOR O/S
0082'   5E                  MOV    E,M      ;GET O/S LOAD ADDRESS
0083'   23                  INX    H
0084'   56                  MOV    D,M
0085'   23                  INX    H
0086'   4E                  MOV    C,M      ;GET SLAVE PROCESSOR O/S LENGTH
0087'   23                  INX    H
0088'   46                  MOV    B,M
0089'   09                  DAD    B        ;CALC LAST BYTE OF O/S
008A'   EB                  XCHG            ;O/S LOAD ADDRESS TO HL-REG
008B'   09                  DAD    B        ;CALC LAST BYTE LOAD ADDRESS
008C'   2B                  DCX    H
008D'   EB                  XCHG            ;HL=END OF O/S-HL=LAST LOAD ADDRES
008E'   EDB8                LDDR            ;MOVE O/S INTO LOAD ADDRESS
0090'   13                  INX    D        ;ADVANCE TO O/S ENTRYPOINT
0091'   2A 01C0             LHLD   RELOC[SIDSLP]  ;GET SLAVE SOURCE ID
0094'   22 0080             SHLD   TBUF     ;STORE ID IN DEFAULT BUFFER
0097'   EB                  XCHG            ;O/S ENTRYPOINT TO HL-REG
```

```
0098'   E9                PCHL                ;TRANSFER TO SLAVE PROCESSOR O/S
0099'   0E1E      ..WAIT: MVI     C,SPDATA  ;C=SLAVE PROCESSOR DATA PORT
009B'   3E08              MVI     A,1<REQEST  ;GET SLAVE REQUEST BIT
009D'   D31F              OUT     SPSTAT  ;SET SLAVE PROCESSOR REQUEST BIT
009F'   DB1F      ..WTL2: IN      SPSTAT  ;GET SLAVE PROCESSOR STATUS
00A1'   E608              ANI     1<REQEST  ;REQUEST BIT SET?
00A3'   20FA              JRNZ    ..WTL2  ;IF SO, WAIT
00A5'   C9                RET             ;ELSE, DONE
                  ;
00A6'   F5        LD2ISR: PUSH    PSW     ;SAVE AF-REG
00A7'   3E06              MVI     A,AACK  ;GET ASCII ACK
00A9'   D31E              OUT     SPDATA  ;OUTPUT ACK
00AB'   DB1E              IN      SPDATA  ;INPUT BYTE
00AD'   FE06              CPI     AACK    ;RESPONSE=ACK?
00AF'   2008              JRNZ    ..X     ;IF NOT, CONTINUE
00B1'   3A 01BB           LDA     RELOC[FLAGS]  ;ELSE, GET FLAGS
00B4'   CBFF              SET     RCVMSG,A  ;SET RECEIVE MESSAGE FLAG
00B6'   32 01BB           STA     RELOC[FLAGS]  ;UPDATE FLAGS
00B9'   F1        ..X:    POP     PSW     ;RESTORE AF-REG
00BA'   FB                EI              ;ENABLE INTERRUPTS
00BB'   ED4D              RETI            ;DONE
                  ;
00BD'   0253      DMADDR: .WORD   RELOC[OSLOAD]  ;DMA ADDRESS
00BF'   00        FLAGS:  .BYTE   0       ;FLAGS
                  ;
00C0'             REQMSG:                 ;DOWNLOAD REQUEST MESSAGE
00C0'             MSGHDR:                 ;MESSAGE HEADER
00C0'   0C        MSGLEN: .BYTE   MSGHBL  ;MESSAGE LENGTH
00C1'             DIDSLP::                ;SLAVE PROCESSOR DESTINATION ID
00C1'   0000      MSGDID: .WORD   0       ;MESSAGE DESTINATION ID
00C3'   00        MSGPID: .BYTE   0       ;MESSAGE PROCESS ID
00C4'             SIDSLP::                ;SLAVE PROCESSOR SOURCE ID
00C4'   0000      MSGSID: .WORD   0       ;MESSAGE SOURCE ID
00C6'   000000    MSGORG: .BYTE   [3]0    ;MESSAGE ORIGIN
00C9'   00        MSFLVL: .BYTE   0       ;MESSAGE LEVEL
00CA'   00        MSGFCD: .BYTE   0       ;MESSAGE FORMAT CODE
                  ;
000B              MSGHL   = .-MSGHDR      ;MESSAGE HEADER LENGTH
                  ;
00CB'   20        SSLSLP::.BYTE   ASP     ;SLAVE PROCESSOR O/S SUFFIX LETTER
                  ;
000C              MSGHBL  = .-MSGHDR      ;MESSAGE HEADER/BUFFER LENGTH
                  ;
00C8              LD2LEN  = .-LD2SLP      ;SLAVE PROCESSOR BOOT CODE LENGTH
                  ;
00CC'             REPMSG: .BLKB   MSGHL+128  ;REPLY MESSAGE BUFFER
                  :
0157'             OSLOAD  = .             ;O/S LOAD ADDRESS
                  ;
                  .PRGEND
```

```
                          ;
                          ; VERSION: 01/05/84
                          ;
                          .IDENT  NITSLV          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
0010                      INTBAS  = 0010H         ;INTERRUPT VECTOR BASE
                          ;
0018                      INTD    = 18H           ;INTERRUPT CONTROLLER DATA PORT
0019                      INTC    = 19H           ;INTERRUPT CONTROLLER CONTROL PORT
                          ;
001E                      SPDATA  = 1EH           ;SLAVE PROCESSOR DATA PORT
001F                      SPSTAT  = 1FH           ;SLAVE PROCESSOR STATUS/CTRL PORT
                          ;
0000                      PROMON  = 0             ;PROM ON
0001                      PARERR  = 1             ;PARITY ERROR
0002                      OVRRUN  = 2             ;OVERRUN
0003                      REQEST  = 3             ;REQUEST
                          ;
0000:04                           .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                          ;
0000:04 3E06              HDWNIT::MVI     A,6     ;DISABLE PROM/CLEAR OVERRUN/REQUEST
0002:04 D31F                      OUT     SPSTAT
0004:04 AF                        XRA     A
0005:04 32 0003                   STA     IOBYTE  ;SET I/O BYTE=0
0008:04 CD 001D:04                CALL    INCNIT  ;INITIALIZE INTERRUPT CONTROLLER
000B:04 CD 0000:05                CALL    BNKNIT# ;INITIALIZE BANKED TPA DRIVER
000E:04 CD 0000:06                CALL    SPINIT# ;INITIALIZE SERIAL/PARALLEL DRIVER
0011:04 CD 0000:07                CALL    RTCNIT# ;INITIALIZE REAL TIME CLOCK DRIVER

0014:04 CD 0000:08                CALL    DSKINA# ;INITIALIZE DISK DRIVER A

0017:04 CD 0000:09                CALL    CKTINA# ;INITIALIZE CIRCUIT DRIVER A
001A:04 C3 0000:0A                JMP     CKTINB# ;INITIALIZE CIRCUIT DRIVER B
                          ;
001D:04 ED5E              INCNIT: IM2             ;SET Z80 INTERRUPT MODE 2
001F:04 AF                        XRA     A
0020:04 ED47                      STAI            ;SET INTERRUPT VECTORS TO PAGE 0
0022:04 D319                      OUT     INTC    ;RESET INT CONTROLLER
0024:04 3E80                      MVI     A,80H   ;DEFINE INT CONT MODE:
0026:04 D319                      OUT     INTC    ; FIXED PRIORITY
                                                  ; INDIVIDUAL VECTOR
                                                  ; INTERRUPT MODE
                                                  ; GINT ACTIVE-LOW
                                                  ; IREQ ACTIVE-LOW
0028:04 3EC0                      MVI     A,0C0H  ;LOAD AUTO-CLEAR REGISTER
002A:04 D319                      OUT     INTC
002C:04 3EFF                      MVI     A,0FFH  ;AUTO-CLEAR ALL CHANNELS
002E:04 D318                      OUT     INTD
0030:04 3EA9                      MVI     A,0A9H  ;ENABLE INTERRUPT CONTROLLER
0032:04 D319                      OUT     INTC
0034:04 C9                        RET             ;DONE
                          ;
```

```
                        ; DRIVERS SET UP INTERRUPT CONTROLLER AND VECTORS
                        ; BY CALLING THE FOLLOWING SUBROUTINE "INTNIT" WITH:
                        ;     A  = INTERRUPT NUMBER (0...7)
                        ;     HL = INTERRUPT SVC ROUTINE ADDRESS
                        ;
0035:04 F5              INTNIT::PUSH    PSW     ;SAVE INTERRUPT NUMBER
0036:04 EB                      XCHG            ;ISR ADDRESS TO DE
0037:04 87                      ADD     A       ;VECTOR = INTERRUPT BASE + 2*N
0038:04 4F                      MOV     C,A
0039:04 0600                    MVI     B,0
003B:04 21 0010                 LXI     H,INTBAS
003E:04 09                      DAD     B       ;HL = VECTOR ADDRESS
003F:04 73                      MOV     M,E     ;STORE ISR ADDRESS IN VECTOR
0040:04 23                      INX     H
0041:04 72                      MOV     M,D
0042:04 2B                      DCX     H
0043:04 F1                      POP     PSW     ;RESTORE INTERRUPT NUMBER
0044:04 F5                      PUSH    PSW     ;SAVE INTERRUPT NUMBER
0045:04 F6E0                    ORI     0E0H    ;CONSTRUCT LOAD RESPONSE MEMORY CMD
0047:04 D319                    OUT     INTC    ;SEND TO INTERRUPT CONTROLLER
0049:04 7D                      MOV     A,L     ;GET LSB OF VECTOR ADDRESS
004A:04 D318                    OUT     INTD    ;LOAD INTO RESPONSE MEMORY
004C:04 F1                      POP     PSW     ;RESTORE INTERRUPT NUMBER
004D:04 F618                    ORI     18H     ;CONSTRUCT CLEAR IRR & IMR CMD
004F:04 D319                    OUT     INTC    ;SEND TO INTERRUPT CONTROLLER
0051:04 C9                      RET             ;DONE
                        ;
                        .PRGEND
```

```
                      ;
                      ; VERSION: 01/05/84
                      ;
                      .IDENT  BNKSLV          ;MODULE ID
                      ;
                      .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                      ;
0010                  INTBAS  = 0010H         ;INTERRUPT VECTOR BASE ADDRESS
0020                  SIOVEC  = 0020H         ;SIO INTERRUPT VECTOR
                      ;
001D                  MEMCTL  = 1DH           ;MEMORY CONTROL REGISTER
                      ;
001F                  SPSTAT  = 1FH           ;SLAVE PROCESSOR STATUS/CTRL PORT
                      ;
0001                  RESPES  = 1             ;RESET PARITY ERROR STATUS BIT
                      ;
0000:04                       .LOC    .BANK.# ;LOCATE IN COMMON AREA
                      ;
0000:04 ED73 008F:04  BNKNIT::SSPD    SPSAVE  ;SAVE STACK POINTER
0004:04 31 00A1:04            LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
0007:04 3E01                  MVI     A,1     ;GET BANK 1
0009:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 1
000C:04 21 0000               LXI     H,0     ;INITIALIZE MEMORY PARITY
000F:04 11 0000               LXI     D,0
0012:04 01 0000               LXI     B,0
0015:04 EDB0                  LDIR
0017:04 3E02                  MVI     A,1<RESPES  ;GET RESET PARITY ERROR STATUS
0019:04 D31F                  OUT     SPSTAT  ;RESET PARITY ERROR STATUS
001B:04 21 0036:04            LXI     H,NETINT  ;GET INTERRUPT SERVICE ADDRESS
001E:04 22 0010               SHLD    INTBAS+(2*0)  ;SET INTERRUPT VECTOR
0021:04 21 0052:04            LXI     H,RTCINT  ;GET INTERRUPT SERVICE ADDRESS
0024:04 22 0016               SHLD    INTBAS+(2*3)  ;SET INTERRUPT VECTOR
0027:04 21 006E:04            LXI     H,SIOINT  ;GET INTERRUPT SERVICE ADDRESS
002A:04 22 0020               SHLD    SIOVEC  ;SET INTERRUPT VECTOR
002D:04 AF                    XRA     A       ;GET BANK 0
002E:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 0
0031:04 ED7B 008F:04          LSPD    SPSAVE  ;RESTORE STACK POINTER
0035:04 C9                    RET             ;DONE
                      ;
0036:04 ED73 008F:04  NETINT: SSPD    SPSAVE  ;SAVE STACK POINTER
003A:04 31 00A1:04            LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
003D:04 F5                    PUSH    PSW     ;SAVE AF-REG
003E:04 AF                    XRA     A       ;GET BANK 0
003F:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 0
0042:04 CD 0000:05            CALL    NETISR# ;PROCESS NETWORK INTERRUPT
0045:04 F3                    DI              ;DISABLE INTERRUPTS
0046:04 3E01                  MVI     A,1     ;GET BANK 1
0048:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 1
004B:04 F1                    POP     PSW     ;RESTORE AF-REG
004C:04 ED7B 008F:04          LSPD    SPSAVE  ;RESTORE STACK POINTER
0050:04 FB                    EI              ;ENABLE INTERRUPTS
0051:04 C9                    RET             ;DONE
                      ;
0052:04 ED73 008F:04  RTCINT: SSPD    SPSAVE  ;SAVE STACK POINTER
```

```
0056:04 31 00A1:04            LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
0059:04 F5                    PUSH    PSW     ;SAVE AF-REG
005A:04 AF                    XRA     A       ;GET BANK 0
005B:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 0
005E:04 CD 0000:06            CALL    RTCISR# ;PROCESS REAL TIME CLOCK INTERRUPT
0061:04 F3                    DI              ;DISABLE INTERRUPTS
0062:04 3E01                  MVI     A,1     ;GET BANK 1
0064:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 1
0067:04 F1                    POP     PSW     ;RESTORE AF-REG
0068:04 ED7B 008F:04          LSPD    SPSAVE  ;RESTORE STACK POINTER
006C:04 FB                    EI              ;ENABLE INTERRUPTS
006D:04 C9                    RET             ;DONE
                      ;
006E:04 ED73 008F:04  SIOINT: SSPD    SPSAVE  ;SAVE STACK POINTER
0072:04 31 00A1:04            LXI     SP,AUXSTK  ;SET UP AUXILLIARY STACK
0075:04 F5                    PUSH    PSW     ;SAVE AF-REG
0076:04 AF                    XRA     A       ;GET BANK 0
0077:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 0
007A:04 CD 0000:07            CALL    SIOISR# ;PROCESS SERIAL I/O INTERRUPT
007D:04 F3                    DI              ;DISABLE INTERRUPTS
007E:04 3E01                  MVI     A,1     ;GET BANK 1
0080:04 CD 008A:04            CALL    SELBNK  ;SELECT BANK 1
0083:04 F1                    POP     PSW     ;RESTORE AF-REG
0084:04 ED7B 008F:04          LSPD    SPSAVE  ;RESTORE STACK POINTER
0088:04 FB                    EI              ;ENABLE INTERRUPTS
0089:04 C9                    RET             ;DONE
                      ;
008A:04 C6F1          SELBNK::ADI     0F1H    ;CONSTRUCT BANK SELECT COMMAND
008C:04 D31D                  OUT     MEMCTL  ;SELECT REQUESTED BANK
008E:04 C9                    RET             ;DONE
                      ;
008F:04 0000          SPSAVE: .WORD   0       ;STACK POINTER SAVE AREA
0091:04 000000000000          .BYTE   [8*2]0  ;AUXILLIARY STACK AREA
00A1:04               AUXSTK  = .             ;TOP OF AUXILLIARY STACK AREA
                      ;
                      .PRGEND
                      ;
                      ; VERSION: 01/05/84
                      ;
                      .IDENT  SCDSLV          ;MODULE ID
                      ;
                      .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                      ;
001E                  SPDATA  = 1EH           ;SLAVE PROCESSOR DATA PORT
001F                  SPSTAT  = 1FH           ;SLAVE PROCESSOR STATUS/CTRL PORT
                      ;
0000                  PROMON  = 0             ;PROM ON BIT
0001                  RESPES  = 1             ;RESET PARITY ERROR STATUS BIT
0002                  RESORS  = 2             ;RESET OVERRUN ERROR STATUS BIT
0003                  REQEST  = 3             ;REQUEST BIT
                      ;
0006                  SW      = 6             ;SEND WAITING FLAG (FLAGS)
0007                  RA      = 7             ;RECEIVE ACTIVATED FLAG (FLAGS)
                      ;
0000"                         .LOC    .DATA.# ;LOCATE IN DATA AREA
                      ;
0000"   00            FLAGS:  .BYTE   0       ;FLAGS
                      ;
0001"   0000          RCVSPH: .WORD   0       ;RECEIVE MESSAGE SEMAPHORE
0003"   0003"         ..RMHD: .WORD   ..RMHD
0005"   0003"                 .WORD   ..RMHD
                      ;
0007"   0001          SMXSPH: .WORD   1       ;SEND MUTUAL EXCLUSION SEMAPHORE
0009"   0009"         ..MXHD: .WORD   ..MXHD
000B"   0009"                 .WORD   ..MXHD
                      ;
000D"   0000          SNDSPH: .WORD   0       ;SEND MESSAGE SEMAPHORE
000F"   000F"         ..SMHD: .WORD   ..SMHD
0011"   000F"                 .WORD   ..SMHD
                      ;
0000:04                       .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                      ;
0000:04 AF            CKTIN$::XRA     A       ;GET INTERRUPT VECTOR NUMBER
0001:04 21 0068'              LXI     H,NETISR  ;GET INTERRUPT SERVICE ADDRESS
0004:04 CD 0000:05            CALL    INTNIT# ;INITIALIZE INTERRUPT VECTOR
0007:04 3A 0000:06            LDA     NMBCKT# ;GET NUMBER OF CIRCUITS
000A:04 47                    MOV     B,A     ;NUMBER OF CIRCUITS TO B-REG
000B:04 21 0001:07            LXI     H,CKTAST#+1  ;GET CIRCUIT ASSIGNMENT TABLE
000E:04 3A 0081               LDA     TBUF+1  ;GET MSB OF PASSED DESTINATION ID
0011:04 BE            ..SL:   CMP     M       ;CIRCUIT NUMBERS EQUAL?
0012:04 2807                  JRZ     ..DIDF  ;IF SO, CONTINUE
0014:04 23                    INX     H       ;ELSE, ADVANCE TO NEXT TABLE ENTRY
0015:04 23                    INX     H
0016:04 23                    INX     H
0017:04 23                    INX     H
0018:04 10F7                  DJNZ    ..SL    ;CONTINUE
001A:04 C9                    RET             ;DONE
001B:04 2B            ..DIDF: DCX     H       ;BACK UP TO LSB OF DESTINATION ID
001C:04 3A 0080               LDA     TBUF    ;GET LSB OF PASSED DESTINATION ID
001F:04 77                    MOV     M,A     ;SET LSB OF DESTINATION ID
```

```
0020:04 C9                    RET              ;DONE
                      ;
0000'                         .LOC     .PROG.# ;LOCATE IN PROGRAM AREA
                      ;
0000'    13           CKTDR%::INX      D       ;ADVANCE PAST LINK POINTERS
0001'    13                   INX      D
0002'    13                   INX      D
0003'    13                   INX      D
0004'    79                   MOV      A,C     ;GET FUNCTION NUMBER
0005'    B7                   ORA      A       ;FUNCTION NUMBER=0?
0006'    2804                 JRZ      RCVMSG  ;IF SO, CONTINUE
0008'    3D                   DCR      A       ;FUNCTION NUMBER=1?
0009'    2825                 JRZ      SNDMSG  ;IF SO, CONTINUE
000B'    C9                   RET              ;ELSE, DONE
                      ;
000C'    21 0001"     RCVMSG: LXI      H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
000F'    CD 0000:08           CALL     WAIT#   ;WAIT FOR RECEIVE MESSAGE
0012'    CD 0058'             CALL     SRMCOM  ;DO COMMON SETUP
0015'    ED40                 INP      B       ;GET MESSAGE LENGTH
0017'    70                   MOV      M,B     ;STORE MESSAGE LENGTH IN BUFFER
0018'    23                   INX      H       ;INCREMENT BUFFER POINTER
0019'    05                   DCR      B       ;DECREMENT MESSAGE LENGTH
001A'    EDB2                 INIR             ;RECEIVE REMAINDER OF MESSAGE
001C'    FB                   EI               ;ENABLE INTERRUPTS
001D'    21 0000"             LXI      H,FLAGS ;GET FLAGS
0020'    CBBE                 RES      RA,M    ;RESET RECEIVE ACTIVATED FLAG
0022'    CB76                 BIT      SW,M    ;SEND WAITING FLAG SET?
0024'    2808                 JRZ      ..X     ;IF NOT, CONTINUE
0026'    CBB6                 RES      SW,M    ;ELSE, RESET SEND WAITING FLAG
0028'    21 000D"             LXI      H,SNDSPH  ;GET SEND MESSAGE SEMAPHORE
002B'    CD 0000:09           CALL     SIGNAL# ;SIGNAL PROCESS AS READY
002E'    AF           ..X:    XRA      A       ;SET RETURN CODE=0
002F'    C9                   RET              ;DONE
                      ;
0030'    21 0007"     SNDMSG: LXI      H,SMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
0033'    CD 0000:08           CALL     WAIT#   ;WAIT ON MUTUAL EXCLUSION
0036'    21 0000"     ..WTL:  LXI      H,FLAGS ;GET FLAGS
0039'    F3                   DI               ;DISABLE INTERRUPTS
003A'    CB7E                 BIT      RA,M    ;RECEIVE ACTIVATED FLAG SET?
003C'    280B                 JRZ      ..SEND  ;IF NOT, CONTINUE
003E'    FB                   EI               ;ELSE, ENABLE INTERRUPTS
003F'    CBF6                 SET      SW,M    ;SET SEND WAITING FLAG
0041'    21 000D"             LXI      H,SNDSPH  ;GET SEND MESSAGE SEMAPHORE
0044'    CD 0000:08           CALL     WAIT#   ;WAIT FOR RECEIVE COMPLETION
0047'    18ED                 JMPR     ..WTL   ;CONTINUE
0049'    CD 0058'     ..SEND: CALL     SRMCOM  ;DO COMMON SETUP
004C'    46                   MOV      B,M     ;GET MESSAGE LENGTH
004D'    EDB3                 OUTIR            ;SEND MESSAGE
004F'    FB                   EI               ;ENABLE INTERRUPTS
0050'    21 0007"             LXI      H,SMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
0053'    CD 0000:09           CALL     SIGNAL# ;RELEASE MUTUAL EXCLUSION
0056'    AF                   XRA      A       ;SET RETURN CODE=0
0057'    C9                   RET              ;DONE
                      ;
```

```
0058'   EB              SRMCOM: XCHG            ;MESSAGE PACKET ADDRESS TO HL-REG
0059'   0E1E                    MVI     C,SPDATA  ;C=SLAVE PROCESSOR DATA PORT
005B'   3E08                    MVI     A,1<REQEST  ;GET SLAVE REQUEST BIT
005D'   D31F                    OUT     SPSTAT  ;SET SLAVE PROCESSOR REQUEST BIT
005F'   FB                      EI              ;ENABLE INTERRUPTS
0060'   DB1F            ..WTL:  IN      SPSTAT  ;GET SLAVE PROCESSOR STATUS
0062'   E608                    ANI     1<REQEST  ;REQUEST BIT SET?
0064'   20FA                    JRNZ    ..WTL   ;IF SO, WAIT
0066'   F3                      DI              ;ELSE, DISABLE INTERRUPTS
0067'   C9                      RET             ;DONE
                        ;
0068'   ED73 0000:0A    NETISR::SSPD    INTSP#  ;SAVE STACK POINTER
006C'   31 0000:0B              LXI     SP,INTSTK#  ;SET UP AUX STACK
006F'   F5                      PUSH    PSW     ;SAVE AF-REG
0070'   3E06                    MVI     A,AACK  ;GET ASCII ACK
0072'   D31E                    OUT     SPDATA  ;OUTPUT ACK
0074'   DB1E                    IN      SPDATA  ;INPUT BYTE
0076'   FE06                    CPI     AACK    ;RESPONSE=ACK?
0078'   2011                    JRNZ    ..X     ;IF NOT, CONTINUE
007A'   C5                      PUSH    B       ;ELSE, SAVE BC-REG
007B'   D5                      PUSH    D       ;SAVE DE-REG
007C'   E5                      PUSH    H       ;SAVE HL-REG
007D'   21 0000"                LXI     H,FLAGS ;GET FLAGS
0080'   CBFE                    SET     RA,M    ;SET RECEIVE ACTIVATED FLAG
0082'   21 0001"                LXI     H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
0085'   CD 0000:09              CALL    SIGNAL# ;SIGNAL PROCESS AS READY
0088'   E1                      POP     H       ;RESTCRE HL-REG
0089'   D1                      POP     D       ;RESTORE DE-REG
008A'   C1                      POP     B       ;RESTORE BC-REG
008B'   F1              ..X:    POP     PSW     ;SAVE AF-REG
008C'   ED7B 0000:0A            LSPD    INTSP#  ;RESTORE STACK POINTER
0090'   C3 0000:0C              JMP     ISRXIT# ;CONTINUE
                        ;
                        .PRGEND
                        ;
                        ; VERSION: 01/05/84
                        ;
                        .IDENT  RTCSLV          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
0012                    TIM2    = 12H           ;TIMER 2 DATA REGISTER
0013                    TIMCTL  = 13H           ;TIMER CONTROL REGISTER
                        ;
00B6                    T2CMD   = 0B6H          ;TIMER 2 COMMAND
                        ;
0000"                           .LOC    .DATA.# ;LOCATE IN DATA AREA
                        ;
0000"   00              TICCTR: .BYTE   0       ;TICK COUNTER
                        ;- -
0000:04                         .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                        ;
0000:04 3E03            RTCNIT::MVI     A,3     ;GET INTERRUPT VECTOR NUMBER
0002:04 21 0000'                LXI     H,RTCISR  ;GET INTERRUPT SERVICE ADDRESS
0005:04 CD 0000:05              CALL    INTNIT# ;INITIALIZE INTERRUPT VECTOR
0008:04 3EB6                    MVI     A,T2CMD ;GET TIMER 2 COMMAND
000A:04 D313                    OUT     TIMCTL  ;SELECT TIMER 2
000C:04 3E80                    MVI     A,9600&0FFH  ;GET LSB OF TIMER VALUE
000E:04 D312                    OUT     TIM2    ;OUTPUT IT TO TIMER 2 DATA REGISTER
0010:04 3E25                    MVI     A,9600>8  ;GET MSB OF TIMER VALUE
0012:04 D312                    OUT     TIM2    ;OUTPUT IT TO TIMER 2 DATA REGISTER
0014:04 C9                      RET             ;DONE
                        ;
0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
0000'   ED73 0000:06    RTCISR::SSPD    INTSP#  ;SAVE STACK POINTER
0004'   31 0000:07              LXI     SP,INTSTK#  ;SET UP AUX STACK POINTER
0007'   F5                      PUSH    PSW     ;SAVE REGISTERS
0008'   C5                      PUSH    B
0009'   D5                      PUSH    D
000A'   E5                      PUSH    H
000B'   21 0000"                LXI     H,TICCTR  ;GET TICK COUNTER
000E'   34                      INR     M       ;INCREMENT TICK COUNTER
000F'   7E                      MOV     A,M     ;GET TICK COUNT
0010'   FE40                    CPI     64      ;SECONDS COUNT REACHED?
0012'   3805                    JRC     ..NSEC  ;IF NOT, CONTINUE
0014'   3600                    MVI     M,0     ;ELSE, RESET TICK COUNTER
0016'   CD 0000:08              CALL    RTCSEC# ;SERVICE REAL TIME CLOCK MANAGER
0019'   CD 0000:09      ..NSEC: CALL    DLYTIC# ;SERVICE DISPATCHER DELAY MANAGER
001C'   E1                      POP     H       ;RESTORE REGISTERS
001D'   D1                      POP     D
001E'   C1                      POP     B
001F'   F1                      POP     PSW
0020'   ED7B 0000:06            LSPD    INTSP#  ;RESTORE STACK POINTER
0024'   C3 0000:0A              JMP     ISRXIT# ;CONTINUE
                        ;
                        .PRGEND
                        ;
                        ; VERSION: 01/22/84
                        ;
                        .IDENT  SPDSLV          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
0020                    SIOVEC  = 20H           ;SIO INTERRUPT VECTOR ADDRESS
                        ;
0000                    SIOADR  = 00H           ;SIO PORT A DATA REGISTER
0001                    SIOACR  = 01H           ;SIO PORT A CONTROL REGISTER
0002                    SIOBDR  = 02H           ;SIO PORT B DATA REGISTER
0003                    SIOBCR  = 03H           ;SIO PORT B CONTROL REGISTER
                        ;
0000                    RDA     = 0             ;RECEIVED DATA AVAILABLE BIT
0002                    TBE     = 2             ;TRANSMIT BUFFER EMPTY BIT
0003                    DCD     = 3             ;DATA CARRIER DETECT BIT
0005                    CTS     = 5             ;CLEAR TO SEND BIT
                        ;
0010                    TIM0    = 10H           ;TIMER 0 DATA REGISTER
0011                    TIM1    = 11H           ;TIMER 1 DATA REGISTER
0012                    TIM2    = 12H           ;TIMER 2 DATA REGISTER
0013                    TIMCTL  = 13H           ;TIMER CONTROL REGISTER
                        ;
0036                    T0CMD   = 36H           ;TIMER 0 COMMAND
0076                    T1CMD   = 76H           ;TIMER 1 COMMAND
00B6                    T2CMD   = 0B6H          ;TIMER 2 COMMAND
                        ;
0000                    PAUSE   = 0             ;PAUSE FLAG (S0OFLG/S01FLG)
                        ;
0000"                           .LOC    .DATA.# ;LOCATE IN DATA AREA
                        ;
0000"   08              CTSMSK::.BYTE   1<DCD   ;CTS HANDSHAKE MASK
                        ;
0001"   0040            S0IBSZ::.WORD   64      ;SERIAL 0 INPUT BUFFER SIZE
0003"   0000            S0IBUF: .WORD   0       ;SERIAL 0 INPUT BUFFER ADDRESS
0005"   0000            S0IPTR: .WORD   0       ;SERIAL 0 INPUT POINTER
0007"   0000            S0OPTR: .WORD   0       ;SERIAL 0 OUTPUT POINTER
0009"   0000            S0ICNT: .WORD   0       ;SERIAL 0 INPUT COUNT
000B"   00              S0IWCT: .BYTE   0       ;SERIAL 0 INPUT WAIT COUNT
000C"   00              S0OCHR: .BYTE   0       ;SERIAL 0 OUTPUT CHARACTER
000D"   00              S0OFLG: .BYTE   0       ;SERIAL 0 OUTPUT XON/XOFF FLAG
000E"   00              S0BR:   .BYTE   0       ;SERIAL 0 BAUD RATE CODE
                        ;
000F"                   S0ISPH:                 ;SERIAL 0 INPUT SEMAPHORE
000F"   0000                    .WORD   0       ;SEMAPHORE COUNT
0011"   0011"           ..S0IH: .WORD   ..S0IH  ;SEMAPHORE P/D HEAD
0013"   0011"                   .WORD   ..S0IH
                        ;
                                                ;SERIAL 0 OUTPUT SEMAPHORE
0015"   0000            S0OSPH: .WORD   0       ;SEMAPHORE COUNT
0017"   0017"           ..S0OH: .WORD   ..S0OH  ;SEMAPHORE P/D HEAD
0019"   0017"                   .WORD   ..S0OH
                        ;
```

```
001B"    0040          S1IBSZ::.WORD   64       ;SERIAL 1 INPUT BUFFER SIZE
001D"    0000          S1IBUF: .WORD   0        ;SERIAL 1 INPUT BUFFER ADDRESS
001F"    0000          S1IPTR: .WORD   0        ;SERIAL 1 INPUT POINTER
0021"    0000          S1OPTR: .WORD   0        ;SERIAL 1 OUTPUT POINTER
0023"    0000          S1ICNT: .WORD   0        ;SERIAL 1 INPUT COUNT
0025"    00            S1IWCT: .BYTE   0        ;SERIAL 1 INPUT WAIT COUNT
0026"    00            S1OCHR: .BYTE   0        ;SERIAL 1 OUTPUT CHARACTER
0027"    00            S1OFLG: .BYTE   0        ;SERIAL 1 OUTPUT XON/XOFF FLAG
0028"    00            S1BR:   .BYTE   0        ;SERIAL 1 BAUD RATE CODE
                       ;
                                                ;SERIAL 1 INPUT SEMAPHORE
0029"    0000          S1ISPH: .WORD   0        ;SEMAPHORE COUNT
002B"    002B"         ..S1IH: .WORD   ..S1IH   ;SEMAPHORE P/D HEAD
002D"    002B"                 .WORD   ..S1IH
                       ;
                                                ;SERIAL 1 OUTPUT SEMAPHORE
002F"    0000          S1OSPH: .WORD   0        ;SEMAPHORE COUNT
0031"    0031"         ..S1OH: .WORD   ..S1OH   ;SEMAPHORE P/D HEAD
0033"    0031"                 .WORD   ..S1OH
                       ;
0000:04                        .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                       ;
0000:04 21 01BE'       SPINIT::LXI     H,SIOISR  ;GET SIO INTERRUPT SERVICE ADDR
0003:04 22 0020                SHLD    SIOVEC  ;SET SIO INTERRUPT VECTOR ADDRESS
0006:04 21 0035:04             LXI     H,SIOPGM  ;GET SIO PROGRAM LIST
0009:04 01 0901                LXI     B,SIOAPL<8!SIOACR  ;B=LENGTH/C=CONTROL REG
000C:04 EDB3                   OUTIR            ;PROGRAM SIO PORT A
000E:04 21 0035:04             LXI     H,SIOPGM  ;GET SIO PROGRAM LIST
0011:04 01 0B03                LXI     B,SIOBPL<8!SIOBCR  ;B=LENGTH/C=CONTROL REG
0014:04 EDB3                   OUTIR            ;PROGRAM SIO PORT B
0016:04 2A 0001"               LHLD    S0IBSZ   ;GET SERIAL 0 INPUT BUFFER SIZE
0019:04 CD 0000:05             CALL    ALLOC#   ;ALLOCATE PACKET FOR SERIAL BUFFER
001C:04 22 0003"               SHLD    S0IBUF   ;SAVE SERIAL 0 INPUT BUFFER ADDRESS
001F:04 22 0005"               SHLD    S0IPTR   ;SET SERIAL 0 INPUT POINTER
0022:04 22 0007"               SHLD    S0OPTR   ;SET SERIAL 0 OUTPUT POINTER
0025:04 2A 001B"               LHLD    S1IBSZ   ;GET SERIAL 1 INPUT BUFFER SIZE
0028:04 CD 0000:05             CALL    ALLOC#   ;ALLOCATE PACKET FOR SERIAL BUFFER
002B:04 22 001D"               SHLD    S1IBUF   ;SAVE SERIAL 1 INPUT BUFFER ADDRESS
002E:04 22 001F"               SHLD    S1IPTR   ;SET SERIAL 1 INPUT POINTER
0031:04 22 0021"               SHLD    S1OPTR   ;SET SERIAL 1 OUTPUT POINTER
0034:04 C9                     RET              ;DONE
                       ;
0035:04 18             SIOPGM: .BYTE   18H      ;RESET CHANNEL
0036:04 04                     .BYTE   4        ;SELECT WR4
0037:04 44                     .BYTE   44H      ;WRITE REGISTER 4 CONTROL WORD
0038:04 05                     .BYTE   5        ;SELECT WR5
0039:04 EA                     .BYTE   0EAH     ;WRITE REGISTER 5 CONTROL WORD
003A:04 03                     .BYTE   3        ;SELECT WR3
003B:04 C1                     .BYTE   0C1H     ;WRITE REGISTER 3 CONTROL WORD
003C:04 01                     .BYTE   1        ;SELECT WR1
003D:04 10                     .BYTE   10H      ;WRITE REGISTER 1 CONTROL WORD
                       ;
0009                   SIOAPL  = .-SIOPGM       ;SIO PORT A PROGRAM LENGTH
                       ;
```

```
003E:04 02                     .BYTE   2         ;SELECT WR2
003F:04 20                     .BYTE   SIOVEC    ;WRITE REGISTER 1 CONTROL WORD
                       ;
000B                   SIOBPL  = .-SIOPGM        ;SIO PORT B PROGRAM LENGTH
                       ;
0000'                          .LOC    .PROG.#   ;LOCATE IN PROGRAM AREA
                       ;
0000'                  SERIAL::
0000'   7B             COMDRV::MOV     A,E       ;GET FUNCTION NUMBER
0001'   B7                     ORA     A         ;FUNCTION NUMBER=0?
0002'   281D                   JRZ     SERST     ;IF SO, CONTINUE
0004'   FE0A                   CPI     10        ;FUNCTION NUMBER=10?
0006'   CA 00A4'               JZ      SEROPT    ;IF SO, CONTINUE
0009'   3D                     DCR     A         ;FUNCTION NUMBER=1?
000A'   282E                   JRZ     SERIN     ;IF SO, CONTINUE
000C'   3D                     DCR     A         ;FUNCTION NUMBER=2?
000D'   CA 00F0'               JZ      SEROUT    ;IF SO, CONTINUE
0010'   3D                     DCR     A         ;FUNCTION NUMBER=3?
0011'   CA 02E0'               JZ      SERSBR    ;IF SO, CONTINUE
0014'   3D                     DCR     A         ;FUNCTION NUMBER=4?
0015'   CA 0330'               JZ      SERRBR    ;IF SO, CONTINUE
0018'   3D                     DCR     A         ;FUNCTION NUMBER=5?
0019'   CA 033C'               JZ      SERSMC    ;IF SO, CONTINUE
001C'   3D                     DCR     A         ;FUNCTION NUMBER=6?
001D'   CA 035E'               JZ      SERRMC    ;IF SO, CONTINUE
0020'   C9                     RET               ;ELSE, DONE
                       ;
0021'   78             SERST:  MOV     A,B       ;GET CHANNEL NUMBER
0022'   ED4B 0009"             LBCD    S0ICNT    ;GET SERIAL 0 INPUT BUFFER COUNT
0026'   2A 0007"               LHLD    S0OPTR    ;GET SERIAL 0 OUTPUT POINTER
0029'   B7                     ORA     A         ;CHANNEL NUMBER=0
002A'   2807                   JRZ     ..COM     ;IF SO, CONTINUE
002C'   ED4B 0023"             LBCD    S1ICNT    ;GET SERIAL 1 INPUT BUFFER COUNT
0030'   2A 0021"               LHLD    S1OPTR    ;GET SERIAL 1 OUTPUT POINTER
0033'   78             ..COM:  MOV     A,B
0034'   B1                     ORA     C         ;SERIAL INPUT BUFFER COUNT=0?
0035'   C8                     RZ                ;IF SO, DONE
0036'   4E                     MOV     C,M       ;ELSE, GET SERIAL INPUT CHARACTER
0037'   3EFF                   MVI     A,0FFH    ;SET RETURN CODE=0FFH
0039'   C9                     RET               ;DONE
                       ;
003A'   78             SERIN:  MOV     A,B       ;GET CHANNEL NUMBER
003B'   B7                     ORA     A         ;CHANNEL NUMBER=0?
003C'   2033                   JRNZ    ..S1I     ;IF NOT, CONTINUE
003E'   F3             ..S0I:  DI                ;ELSE, DISABLE INTERRUPTS
003F'   2A 0009"               LHLD    S0ICNT    ;GET SERIAL 0 INPUT COUNT
0042'   7C                     MOV     A,H
0043'   B5                     ORA     L         ;SERIAL 0 INPUT COUNT=0?
0044'   281F                   JRZ     ..WT0     ;IF SO, CONTINUE
0046'   2B                     DCX     H         ;DECREMENT SERIAL 0 INPUT COUNT
0047'   22 0009"               SHLD    S0ICNT    ;UPDATE SERIAL 0 INPUT COUNT
004A'   2A 0007"               LHLD    S0OPTR    ;GET SERIAL 0 OUTPUT POINTER
004D'   7E                     MOV     A,M       ;GET CHARACTER FROM BUFFER
004E'   23                     INX     H         ;INCREMENT SERIAL 0 OUTPUT POINTER
```

```
  004F'   EB                      XCHG             ;SERIAL 0 OUTPUT POINTER TO DE-REG
  0050'   2A 0001"                LHLD     S0IBSZ  ;GET SERIAL 0 INPUT BUFFER SIZE
  0053'   2B                      DCX      H       ;DECREMENT INPUT BUFFER SIZE
  0054'   ED4B 0003"              LBCD     S0IBUF  ;GET SERIAL 0 INPUT BUFFER ADDRESS
  0058'   09                      DAD      B       ;CALC LAST INPUT BUFFER ADDRESS
  0059'   ED52                    DSBC     D       ;BUFFER WRAP-AROUND?
  005B'   3002                    JRNC     ..NWA0  ;IF NOT, CONTINUE
  005D'   59                      MOV      E,C     ;GET SERIAL 0 INPUT BUFFER ADDRESS
  005E'   50                      MOV      D,B
  005F'   ED53 0007"     ..NWA0:  SDED     S0OPTR  ;UPDATE SERIAL 0 OUTPUT POINTER
  0063'   FB                      EI               ;ENABLE INTERRUPTS
  0064'   C9                      RET              ;DONE
  0065'   21 000B"       ..WT0:   LXI      H,S0IWCT  ;GET SERIAL 0 INPUT WAIT COUNT
  0068'   34                      INR      M       ;INCREMENT INPUT WAIT COUNT
  0069'   21 000F"                LXI      H,S0ISPH  ;GET SERIAL 0 INPUT SEMAPHORE
  006C'   CD 0000:06              CALL     WAIT#   ;WAIT FOR CONSOLE INPUT
  006F'   18CD                    JMPR     ..S0I   ;CONTINUE
  0071'   F3             ..S1I:   DI               ;DISABLE INTERRUPTS
  0072'   2A 0023"                LHLD     S1ICNT  ;GET SERIAL 1 INPUT COUNT
  0075'   7C                      MOV      A,H
  0076'   B5                      ORA      L       ;SERIAL 1 INPUT COUNT=0?
  0077'   281F                    JRZ      ..WT1   ;IF SO, CONTINUE
  0079'   2B                      DCX      H       ;DECREMENT SERIAL 1 INPUT COUNT
  007A'   22 0023"                SHLD     S1ICNT  ;UPDATE SERIAL 1 INPUT COUNT
  007D'   2A 0021"                LHLD     S1OPTR  ;GET SERIAL 1 OUTPUT POINTER
  0080'   7E                      MOV      A,M     ;GET CHARACTER FROM BUFFER
  0081'   23                      INX      H       ;INCREMENT SERIAL 1 OUTPUT POINTER
  0082'   EB                      XCHG             ;SERIAL 1 OUTPUT POINTER TO DE-REG
  0083'   2A 001B"                LHLD     S1IBSZ  ;GET SERIAL 1 INPUT BUFFER SIZE
  0086'   2B                      DCX      H       ;DECREMENT INPUT BUFFER SIZE
  0087'   ED4B 001D"              LBCD     S1IBUF  ;GET SERIAL 1 INPUT BUFFER ADDRESS
  008B'   09                      DAD      B       ;CALC LAST INPUT BUFFER ADDRESS
  008C'   ED52                    DSBC     D       ;BUFFER WRAP-AROUND?
  008E'   3002                    JRNC     ..NWA1  ;IF NOT, CONTINUE
  0090'   59                      MOV      E,C     ;GET SERIAL 1 INPUT BUFFER ADDRESS
  0091'   50                      MOV      D,B
  0092'   ED53 0021"     ..NWA1:  SDED     S1OPTR  ;UPDATE SERIAL 1 OUTPUT POINTER
  0096'   FB                      EI               ;ENABLE INTERRUPTS
  0097'   C9                      RET              ;DONE
  0098'   21 0025"       ..WT1:   LXI      H,S1IWCT  ;GET SERIAL 1 INPUT WAIT COUNT
  009B'   34                      INR      M       ;INCREMENT INPUT WAIT COUNT
  009C'   21 0029"                LXI      H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
  009F'   CD 0000:06              CALL     WAIT#   ;WAIT FOR CONSOLE INPUT
  00A2'   18CD                    JMPR     ..S1I   ;CONTINUE
                         ;
  00A4'   78             SEROPT:  MOV      A,B     ;GET CHANNEL NUMBER
  00A5'   B7                      ORA      A       ;CHANNEL NUMBER=1?
  00A6'   3E10                    MVI      A,10H   ;GET RESET EXTERNAL STATUS COMMAND
  00A8'   2023                    JRNZ     ..S10   ;IF CHANNEL NUMBER=1, CONTINUE
  00AA'   D301                    OUT      SIOACR  ;ELSE, RESET EXTERNAL STATUS
  00AC'   DB01                    IN       SIOACR  ;GET SIO PORT A STATUS
  00AE'   E604                    ANI      1<TBE   ;TRANSMIT BUFFER EMPTY?
  00B0'   C8                      RZ               ;IF NOT, DONE
  00B1'   3A 000D"                LDA      S0OFLG  ;GET FLAG
```

```
00B4'   CB47                BIT     PAUSE,A ;IN A PAUSE?
00B6'   3E00                MVI     A,0     ;PRESET RETURN CODE
00B8'   C0                  RNZ             ;IF SO, DONE
00B9'   3A 000E"            LDA     S0BR    ;ELSE, GET SERIAL 0 BAUD RATE COD
00BC'   E640                ANI     1<6     ;CTS HANDSHAKING REQUESTED?
00BE'   2807                JRZ     ..NHR0  ;IF NOT REQUIRED, CONTINUE
00C0'   DB01                IN      SIOACR  ;ELSE, GET SIO PORT B STATUS
00C2'   21 0000"            LXI     H,CTSMSK  ;GET CTS MASK
00C5'   A6                  ANA     M       ;CHECK IF CLEAR-TO-SEND
00C6'   C8                  RZ              ;IF CLEAR TO SEND FALSE, DONE
00C7'   79          ..NHR0: MOV     A,C     ;GET SERIAL 0 OUTPUT CHARACTER
00C8'   D300                OUT     SIOADR  ;OUTPUT CHARACTER
00CA'   3EFF                MVI     A,0FFH  ;SET RETURN CODE=0FFH
00CC'   C9                  RET             ;DONE
00CD'   D303        ..S10:  OUT     SIOBCR  ;RESET EXTERNAL STATUS
00CF'   DB03                IN      SIOBCR  ;GET SIO PORT B STATUS
00D1'   E604                ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
00D3'   C8                  RZ              ;IF NOT, DONE
00D4'   3A 0027"            LDA     S1OFLG  ;GET FLAG
00D7'   CB47                BIT     PAUSE,A ;IN A PAUSE?
00D9'   3E00                MVI     A,0     ;PRESET RETURN CODE
00DB'   C0                  RNZ             ;IF SO, DONE
00DC'   3A 0028"            LDA     S1BR    ;ELSE, GET SERIAL 1 BAUD RATE COD
00DF'   E640                ANI     1<6     ;CTS HANDSHAKING REQUESTED?
00E1'   2807                JRZ     ..NHR1  ;IF NOT REQUIRED, CONTINUE
00E3'   DB03                IN      SIOBCR  ;ELSE, GET SIO B PORT STATUS
00E5'   21 0000"            LXI     H,CTSMSK  ;GET CTS MASK
00E8'   A6                  ANA     M       ;CHECK IF CLEAR-TO-SEND
00E9'   C8                  RZ              ;IF CLEAR TO SEND FALSE, DONE
00EA'   79          ..NHR1: MOV     A,C     ;GET SERIAL 1 OUTPUT CHARACTER
00EB'   D302                OUT     SIOBDR  ;OUTPUT CHARACTER
00ED'   3EFF                MVI     A,0FFH  ;SET RETURN CODE=0FFH
00EF'   C9                  RET             ;DONE
                    ;
00F0'   78          SEROUT: MOV     A,B     ;GET CHANNEL NUMBER
00F1'   B7                  ORA     A       ;CHANNEL NUMBER=1?
00F2'   3E10                MVI     A,10H   ;GET RESET EXTERNAL STATUS COMMAN
00F4'   2032                JRNZ    ..S10   ;IF CHANNEL NUMBER=1, CONTINUE
00F6'   D301                OUT     SIOACR  ;ELSE, RESET EXTERNAL STATUS
00F8'   DB01                IN      SIOACR  ;GET SIO PORT A STATUS
00FA'   E604                ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
00FC'   281A                JRZ     ..S0NR  ;IF NOT, CONTINUE
00FE'   3A 000D"            LDA     S0OFLG  ;GET FLAG
0101'   CB47                BIT     PAUSE,A ;IN A PAUSE?
0103'   2013                JRNZ    ..S0NR  ;IF SO, CONTINUE
0105'   3A 000E"            LDA     S0BR    ;ELSE, GET SERIAL 0 BAUD RATE COD
0108'   E640                ANI     1<6     ;CTS HANDSHAKING REQUESTED?
010A'   2808                JRZ     ..HNR0  ;IF NOT REQUIRED, CONTINUE
010C'   DB01                IN      SIOACR  ;ELSE, GET SIO PORT A STATUS
010E'   21 0000"            LXI     H,CTSMSK  ;GET CTS MASK
0111'   A6                  ANA     M       ;CHECK IF CLEAR-TO-SEND
0112'   2804                JRZ     ..S0NR  ;IF CLEAR TO SEND FALSE, CONTINUE
0114'   79          ..HNR0: MOV     A,C     ;GET SERIAL 0 OUTPUT CHARACTER
0115'   D300                OUT     SIOADR  ;OUTPUT CHARACTER
```

```
 0117'   C9                         RET               ;DONE
 0118'   79               ..S0NR:   MOV     A,C       ;GET SERIAL 0 OUTPUT CHARACTER
 0119'   32 000C"                   STA     S0OCHR    ;SAVE OUTPUT CHARACTER
 011C'   11 015A'                   LXI     D,S0OPOL  ;GET SERIAL 0 OUT POLL ROUTINE
 011F'   CD 0000:07                 CALL    LNKPOL#   ;CREATE POLL ROUTINE
 0122'   21 0015"                   LXI     H,S0OSPH  ;GET SERIAL 0 OUT SEMAPHORE
 0125'   C3 0000:06                 JMP     WAIT#     ;DISPATCH IF NECESSARY
 0128'   D303             ..S1O:    OUT     SIOBCR    ;RESET EXTERNAL STATUS
 012A'   DB03                       IN      SIOBCR    ;GET SIO PORT B STATUS
 012C'   E604                       ANI     1<TBE     ;TRANSMIT BUFFER EMPTY?
 012E'   281A                       JRZ     ..S1NR    ;IF NOT, CONTINUE
 0130'   3A 0027"                   LDA     S1OFLG    ;GET FLAG
 0133'   CB47                       BIT     PAUSE,A   ;IN A PAUSE?
 0135'   2013                       JRNZ    ..S1NR    ;IF SO, CONTINUE
 0137'   3A 0028"                   LDA     S1BR      ;ELSE, GET SERIAL 1 BAUD RATE CODE
 013A'   E640                       ANI     1<6       ;CTS HANDSHAKING REQUESTED?
 013C'   2808                       JRZ     ..HNR1    ;IF NOT REQUIRED, CONTINUE
 013E'   DB03                       IN      SIOBCR    ;ELSE, GET SIO PORT B STATUS
 0140'   21 0000"                   LXI     H,CTSMSK  ;GET CTS MASK
 0143'   A6                         ANA     M         ;CHECK IF CLEAR-TO-SEND
 0144'   2804                       JRZ     ..S1NR    ;IF CLEAR TO SEND FALSE, CONTINUE
 0146'   79               ..HNR1:   MOV     A,C       ;GET SERIAL 1 OUTPUT CHARACTER
 0147'   D302                       OUT     SIOBDR    ;OUTPUT CHARACTER
 0149'   C9                         RET               ;DONE
 014A'   79               ..S1NR:   MOV     A,C       ;GET SERIAL 1 OUTPUT CHARACTER
 014B'   32 0026"                   STA     S1OCHR    ;SAVE OUTPUT CHARACTER
 014E'   11 018C'                   LXI     D,S1OPOL  ;GET SERIAL 1 OUT POLL ROUTINE
 0151'   CD 0000:07                 CALL    LNKPOL#   ;CREATE POLL ROUTINE
 0154'   21 002F"                   LXI     H,S1OSPH  ;GET SERIAL 1 OUT SEMAPHORE
 0157'   C3 0000:06                 JMP     WAIT#     ;DISPATCH IF NECESSARY
                          ;
 015A'                    S0OPOL:                     ;SERIAL 0 OUTPUT POLL ROUTINE
 015A'   0000                       .WORD   0         ;SUCCESSOR LINK POINTER
 015C'   0000                       .WORD   0         ;PREDECESSOR LINK POINTER
                          ;
 015E'   3E10                       MVI     A,10H     ;GET RESET EXTERNAL STATUS COMMAND
 0160'   D301                       OUT     SIOACR    ;RESET EXTERNAL STATUS
 0162'   DB01                       IN      SIOACR    ;GET SIO PORT A STATUS
 0164'   E604                       ANI     1<TBE     ;TRANSMIT BUFFER EMPTY?
 0166'   C8                         RZ                ;IF NOT, DONE
 0167'   3A 000D"                   LDA     S0OFLG    ;GET FLAG
 016A'   CB47                       BIT     PAUSE,A   ;IN A PAUSE?
 016C'   C0                         RNZ               ;IF SO, DONE
 016D'   3A 000E"                   LDA     S0BR      ;ELSE, GET SERIAL 0 BAUD RATE CODE
 0170'   E640                       ANI     1<6       ;CTS HANDSHAKING REQUESTED?
 0172'   2807                       JRZ     ..HNR     ;IF NOT, CONTINUE
 0174'   DB01                       IN      SIOACR    ;ELSE, GET SIO PORT A STATUS
 0176'   21 0000"                   LXI     H,CTSMSK  ;GET CTS MASK
 0179'   A6                         ANA     M         ;CHECK IF CLEAR-TO-SEND
 017A'   C8                         RZ                ;IF CLEAR TO SEND FALSE, DONE
 017B'   3A 000C"         ..HNR:    LDA     S0OCHR    ;GET SERIAL 0 OUTPUT CHARACTER
 017E'   D300                       OUT     SIOADR    ;OUTPUT CHARACTER
 0180'   21 015A'                   LXI     H,S0OPOL  ;GET SERIAL 0 OUT POLL ROUTINE
 0183'   CD 0000:08                 CALL    UNLINK#   ;UNLINK POLL ROUTINE
```

```
0186'   21 0015"                LXI     H,S0OSPH  ;GET SERIAL 0 OUT SEMAPHORE
0189'   C3 0000:09              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
018C'                   S1OPOL:                 ;SERIAL 1 OUTPUT POLL ROUTINE
018C'   0000                    .WORD   0       ;SUCCESSOR LINK POINTER
018E'   0000                    .WORD   0       ;PREDECESSOR LINK POINTER
                        ;
0190'   3E10                    MVI     A,10H   ;GET RESET EXTERNAL STATUS COMMAND
0192'   D303                    OUT     SIOBCR  ;RESET EXTERNAL STATUS
0194'   DB03                    IN      SIOBCR  ;GET SIO PORT B STATUS
0196'   E604                    ANI     1<TBE   ;TRANSMIT BUFFER EMPTY?
0198'   C8                      RZ              ;IF NOT, DONE
0199'   3A 0027"                LDA     S1OFLG  ;GET FLAG
019C'   CB47                    BIT     PAUSE,A ;IN A PAUSE?
019E'   C0                      RNZ             ;IF SO, DONE
019F'   3A 0028"                LDA     S1BR    ;ELSE, GET SERIAL 1 BAUD RATE CODE
01A2'   E640                    ANI     1<6     ;CTS HANDSHAKING REQUESTED?
01A4'   2807                    JRZ     ..HNR   ;IF NOT, CONTINUE
01A6'   DB03                    IN      SIOBCR  ;ELSE, GET SIO PORT B STATUS
01A8'   21 0000"                LXI     H,CTSMSK  ;GET CTS MASK
01AB'   A6                      ANA     M       ;CHECK IF CLEAR-TO-SEND
01AC'   C8                      RZ              ;IF CLEAR TO SEND FALSE, DONE
01AD'   3A 0026"        ..HNR:  LDA     S1OCHR  ;GET SERIAL 1 OUTPUT CHARACTER
01B0'   D302                    OUT     SIOBDR  ;OUTPUT CHARACTER
01B2'   21 018C'                LXI     H,S1OPOL  ;GET SERIAL 1 OUT POLL ROUTINE
01B5'   CD 0000:08              CALL    UNLINK# ;UNLINK POLL ROUTINE
01B8'   21 002F"                LXI     H,S1OSPH  ;GET SERIAL 1 OUT SEMAPHORE
01BB'   C3 0000:09              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                        ;
01BE'   ED73 0000:0A    SIOISR::SSPD    INTSP#  ;SAVE STACK POINTER
01C2'   31 0000:0B              LXI     SP,INTSTK#  ;SET UP AUX STACK POINTER
01C5'   F5                      PUSH    PSW     ;SAVE REGISTERS
01C6'   C5                      PUSH    B
01C7'   D5                      PUSH    D
01C8'   E5                      PUSH    H
01C9'   CD 01DA'                CALL    ..S0I   ;CHECK FOR SERIAL 0 INPUT
01CC'   CD 0259'                CALL    ..S1I   ;CHECK FOR SERIAL 1 INPUT
01CF'   E1                      POP     H       ;RESTORE REGISTERS
01D0'   D1                      POP     D
01D1'   C1                      POP     B
01D2'   F1                      POP     PSW
01D3'   ED7B 0000:0A            LSPD    INTSP#  ;RESTORE STACK POINTER
01D7'   FB                      EI              ;ENABLE INTERRUPTS
01D8'   ED4D                    RETI            ;DONE
01DA'   DB01            ..S0I:  IN      SIOACR  ;GET SIO PORT A STATUS
01DC'   CB47                    BIT     RDA,A   ;CHARACTER AVAILABLE
01DE'   C8                      RZ              ;IF NOT, DONE
01DF'   DB00                    IN      SIOADR  ;GET SIO PORT A DATA CHARACTER
01E1'   21 000E"                LXI     H,S0BR  ;GET SERIAL 0 BAUD RATE CODE
01E4'   CB6E                    BIT     5,M     ;INHIBIT INPUT FLAG SET?
01E6'   C0                      RNZ             ;IF SO, DONE
01E7'   4F                      MOV     C,A     ;SERIAL 0 DATA CHARACTER TO C-REG
01E8'   CB66                    BIT     4,M     ;XON/XOFF HANDSHAKING?
01EA'   281B                    JRZ     ..NX0   ;NO, CONTINUE
```

```
 01EC'   21 000D"              LXI     H,SOOFLG  ;GET FLAG
 01EF'   E67F                  ANI     7FH      ;STRIP SIGN BIT
 01F1'   FE13                  CPI     ADC3     ;XOFF?
 01F3'   2007                  JRNZ    ..NXOO   ;NO, CONTINUE
 01F5'   CB46                  BIT     PAUSE,M  ;IN A PAUSE?
 01F7'   200E                  JRNZ    ..NXO    ;YES, ACCEPT XOFF AS DATA
 01F9'   CBC6                  SET     PAUSE,M  ;ELSE, SET PAUSE FLAG
 01FB'   C9                    RET              ;AND INHIBIT INPUT OF XOFF
 01FC'   FE11         ..NXOO:  CPI     ADC1     ;XON?
 01FE'   2007                  JRNZ    ..NXO    ;NO, CONTINUE
 0200'   CB46                  BIT     PAUSE,M  ;IN A PAUSE?
 0202'   2803                  JRZ     ..NXO    ;NO, ACCEPT XON AS DATA
 0204'   CB86                  RES     PAUSE,M  ;ELSE, RESET PAUSE FLAG
 0206'   C9                    RET              ;AND INHIBIT INPUT OF XON
 0207'   21 000E"     ..NXO:   LXI     H,SOBR   ;GET SERIAL 0 BAUD RATE CODE
 020A'   CB7E                  BIT     7,M      ;SIGN BIT ON BAUD RATE CODE?
 020C'   2817                  JRZ     ..NADO   ;IF NOT, CONTINUE
 020E'   CBB9                  RES     7,C      ;ELSE, STRIP SIGN BIT ON CHARACTER
 0210'   CD 0000:0C            CALL    SLVRES#  ;CHECK FOR SLAVE RESET
 0213'   3A 0000:0D            LDA     ATNCHR#  ;GET ATTENTION CHARACTER
 0216'   B9                    CMP     C        ;CHARACTER=ATTENTION CHARACTER?
 0217'   200C                  JRNZ    ..NADO   ;IF NOT, CONTINUE
 0219'   2A 0005"              LHLD    SOIPTR   ;ELSE, GET SERIAL 0 INPUT POINTER
 021C'   22 0007"              SHLD    SOOPTR   ;RESET SERIAL 0 OUTPUT POINTER
 021F'   21 0000               LXI     H,0
 0222'   22 0009"              SHLD    SOICNT   ;SET SERIAL 0 INPUT COUNT=0
 0225'   2A 0001"     ..NADO:  LHLD    SOIBSZ   ;GET SERIAL 0 INPUT BUFFER SIZE
 0228'   ED5B 0009"            LDED    SOICNT   ;GET SERIAL 0 INPUT COUNT
 022C'   13                    INX     D        ;INCREMENT SERIAL 0 INPUT COUNT
 022D'   B7                    ORA     A        ;CLEAR CARRY FLAG
 022E'   ED52                  DSBC    D        ;SERIAL 0 INPUT BUFFER FULL?
 0230'   D8                    RC               ;IF SO, DONE
 0231'   ED53 0009"            SDED    SOICNT   ;ELSE, UPDATE SERIAL 0 INPUT COUNT
 0235'   2A 0005"              LHLD    SOIPTR   ;GET SERIAL 0 INPUT POINTER
 0238'   71                    MOV     M,C      ;STORE INPUT CHARACTER IN BUFFER
 0239'   23                    INX     H        ;INCREMENT INPUT POINTER
 023A'   EB                    XCHG             ;DE=INPUT POINTER/HL=BUFFER SIZE
 023B'   2A 0001"              LHLD    SOIBSZ   ;GET SERIAL 0 INPUT BUFFER SIZE
 023E'   2B                    DCX     H        ;DECREMENT INPUT BUFFER SIZE
 023F'   ED4B 0003"            LBCD    SOIBUF   ;GET SERIAL 0 INPUT BUFFER ADDRESS
 0243'   09                    DAD     B        ;CALC LAST INPUT BUFFER ADDRESS
 0244'   ED52                  DSBC    D        ;BUFFER WRAP-AROUND?
 0246'   3002                  JRNC    ..NWAO   ;IF NOT, CONTINUE
 0248'   59                    MOV     E,C      ;GET SERIAL 0 INPUT BUFFER ADDRESS
 0249'   50                    MOV     D,B
 024A'   ED53 0005"   ..NWAO:  SDED    SOIPTR   ;UPDATE SERIAL 0 INPUT POINTER
 024E'   11 000B"              LXI     D,SOIWCT  ;GET SERIAL 0 INPUT WAIT COUNT
 0251'   21 000F"              LXI     H,SOISPH  ;GET SERIAL 0 INPUT SEMAPHORE
 0254'   CD 02D8'              CALL    ..SIGC   ;SIGNAL IF NECESSARY
 0257'   1881                  JMPR    ..SOI    ;CONTINUE
 0259'   DB03         ..S1I:   IN      SIOBCR   ;GET SIO PORT B STATUS
 025B'   CB47                  BIT     RDA,A    ;CHARACTER AVAILABLE
 025D'   C8                    RZ               ;IF NOT, DONE
 025E'   DB02                  IN      SIOBDR   ;GET SIO PORT B DATA CHARACTER
```

```
  0260'   21 0028"                LXI    H,S1BR   ;GET SERIAL 1 BAUD RATE CODE
  0263'   CB6E                    BIT    5,M      ;INHIBIT INPUT FLAG SET?
  0265'   C0                      RNZ             ;IF SO, DONE
  0266'   4F                      MOV    C,A      ;SERIAL 1 DATA CHARACTER TO C-REG
  0267'   CB66                    BIT    4,M      ;XON/XOFF HANDSHAKING?
  0269'   281B                    JRZ    ..NX1    ;NO, CONTINUE
  026B'   21 0027"                LXI    H,S1OFLG  ;GET FLAG
  026E'   E67F                    ANI    7FH      ;STRIP SIGN BIT
  0270'   FE13                    CPI    ADC3     ;XOFF?
  0272'   2007                    JRNZ   ..NXO1   ;NO, CONTINUE
  0274'   CB46                    BIT    PAUSE,M  ;IN A PAUSE?
  0276'   200E                    JRNZ   ..NX1    ;YES, ACCEPT XOFF AS DATA
  0278'   CBC6                    SET    PAUSE,M  ;ELSE, SET PAUSE FLAG
  027A'   C9                      RET             ;AND INHIBIT INPUT OF XOFF
  027B'   FE11           ..NXO1:  CPI    ADC1     ;XON?
  027D'   2007                    JRNZ   ..NX1    ;NO, CONTINUE
  027F'   CB46                    BIT    PAUSE,M  ;IN A PAUSE?
  0281'   2803                    JRZ    ..NX1    ;NO, ACCEPT XON AS DATA
  0283'   CB86                    RES    PAUSE,M  ;ELSE, RESET PAUSE FLAG
  0285'   C9                      RET             ;AND INHIBIT INPUT OF XON
  0286'   21 0028"       ..NX1:   LXI    H,S1BR   ;GET SERIAL 1 BAUD RATE CODE
  0289'   CB7E                    BIT    7,M      ;ATTENTION DETECTION FLAG SET?
  028B'   2817                    JRZ    ..NAD1   ;IF NOT, CONTINUE
  028D'   CBB9                    RES    7,C      ;ELSE, STRIP SIGN BIT ON CHARACTER
  028F'   CD 0000:0C              CALL   SLVRES#  ;CHECK FOR SLAVE RESET
  0292'   3A 0000:0D              LDA    ATNCHR#  ;GET ATTENTION CHARACTER
  0295'   B9                      CMP    C        ;CHARACTER=ATTENTION CHARACTER?
  0296'   200C                    JRNZ   ..NAD1   ;IF NOT, CONTINUE
  0298'   2A 001F"                LHLD   S1IPTR   ;ELSE, GET SERIAL 1 INPUT POINTER
  029B'   22 0021"                SHLD   S1OPTR   ;RESET SERIAL 1 OUTPUT POINTER
  029E'   21 0000                 LXI    H,0
  02A1'   22 0023"                SHLD   S1ICNT   ;SET SERIAL 1 INPUT COUNT=1
  02A4'   2A 001B"       ..NAD1:  LHLD   S1IBSZ   ;GET SERIAL 1 INPUT BUFFER SIZE
  02A7'   ED5B 0023"              LDED   S1ICNT   ;GET SERIAL 1 INPUT COUNT
  02AB'   13                      INX    D        ;INCREMENT SERIAL 1 INPUT COUNT
  02AC'   B7                      ORA    A        ;CLEAR CARRY FLAG
  02AD'   ED52                    DSBC   D        ;SERIAL 1 INPUT BUFFER FULL?
  02AF'   D8                      RC              ;IF SO, DONE
  02B0'   ED53 0023"              SDED   S1ICNT   ;ELSE, UPDATE SERIAL 1 INPUT COUNT
  02B4'   2A 001F"                LHLD   S1IPTR   ;GET SERIAL 1 INPUT POINTER
  02B7'   71                      MOV    M,C      ;STORE INPUT CHARACTER IN BUFFER
  02B8'   23                      INX    H        ;INCREMENT INPUT POINTER
  02B9'   EB                      XCHG            ;DE=INPUT POINTER/HL=BUFFER SIZE
  02BA'   2A 001B"                LHLD   S1IBSZ   ;GET SERIAL 1 INPUT BUFFER SIZE
  02BD'   2B                      DCX    H        ;DECREMENT INPUT BUFFER SIZE
  02BE'   ED4B 001D"              LBCD   S1IBUF   ;GET SERIAL 1 INPUT BUFFER ADDRESS
  02C2'   09                      DAD    B        ;CALC LAST INPUT BUFFER ADDRESS
  02C3'   ED52                    DSBC   D        ;BUFFER WRAP-AROUND?
  02C5'   3002                    JRNC   ..NWA1   ;IF NOT, CONTINUE
  02C7'   59                      MOV    E,C      ;GET SERIAL 1 INPUT BUFFER ADDRESS
  02C8'   50                      MOV    D,B
  02C9'   ED53 001F"     ..NWA1:  SDED   S1IPTR   ;UPDATE SERIAL 1 INPUT POINTER
  02CD'   11 0025"                LXI    D,S1IWCT  ;GET SERIAL 1 INPUT WAIT COUNT
  02D0'   21 0029"                LXI    H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
```

```
02D3'   CD 02D8'            CALL    ..SIGC  ;SIGNAL IF NECESSARY
02D6'   1881                JMPR    ..S1I   ;CONTINUE
02D8'   1A          ..SIGC: LDAX    D       ;GET SERIAL INPUT WAIT COUNT
02D9'   B7                  ORA     A       ;SERIAL INPUT WAIT COUNT=0?
02DA'   C8                  RZ              ;IF SO, DONE
02DB'   3D                  DCR     A       ;DECREMENT SERIAL INPUT WAIT COUNT
02DC'   12                  STAX    D       ;UPDATE SERIAL INPUT WAIT COUNT
02DD'   C3 0000:09          JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                    ;
02E0'   78          SERSBR: MOV     A,B     ;GET CHANNEL NUMBER
02E1'   21 000E"            LXI     H,SOBR  ;GET SERIAL 0 BAUD RATE CODE
02E4'   B7                  ORA     A       ;CHANNEL NUMBER=0?
02E5'   2803                JRZ     ..COM1  ;IF SO, CONTINUE
02E7'   21 0028"            LXI     H,S1BR  ;ELSE, GET SERIAL 1 BAUD RATE CODE
02EA'   71          ..COM1: MOV     M,C     ;SAVE BAUD RATE CODE
02EB'   CD 0301'            CALL    GETBTV  ;GET BAUD RATE TIMER VALUE
02EE'   78                  MOV     A,B     ;GET CHANNEL NUMBER
02EF'   B7                  ORA     A       ;CHANNEL NUMBER=0?
02F0'   3E36                MVI     A,TOCMD ;GET TIMER 0 COMMAND
02F2'   0E10                MVI     C,TIM0  ;GET TIMER 0 DATA REGISTER
02F4'   2804                JRZ     ..COM2  ;IF CHANNEL NUMBER=0, CONTINUE
02F6'   3E76                MVI     A,T1CMD ;ELSE, GET TIMER 1 COMMAND
02F8'   0E11                MVI     C,TIM1  ;GET TIMER 1 DATA REGISTER
02FA'   D313        ..COM2: OUT     TIMCTL  ;SELECT TIMER
02FC'   ED59                OUTP    E       ;OUTPUT LSB OF TIMER VALUE
02FE'   ED51                OUTP    D       ;OUTPUT MSB OF TIMER VALUE
0300'   C9                  RET             ;DONE
                    ;
0301'   79          GETBTV::MOV     A,C     ;GET REQUESTED BAUD RATE CODE
0302'   E60F                ANI     0FH     ;EXTRACT RELEVANT BITS
0304'   87                  ADD     A       ;X2
0305'   5F                  MOV     E,A     ;TO E-REG
0306'   1600                MVI     D,0     ;MAKE IT DOUBLE LENGTH
0308'   21 0310'            LXI     H,BRTBL ;GET BAUD RATE TABLE
030B'   19                  DAD     D       ;INDEX INTO TABLE
030C'   5E                  MOV     E,M     ;GET TIMER VALUE
030D'   23                  INX     H
030E'   56                  MOV     D,M
030F'   C9                  RET             ;DONE
                    ;
0310'   0C00        BRTBL:  .WORD   3072    ;50 BAUD TIMER VALUE
0312'   0800                .WORD   2048    ;75 BAUD TIMER VALUE
0314'   0574                .WORD   1396    ;110 BAUD TIMER VALUE
0316'   0476                .WORD   1142    ;134.5 BAUD TIMER VALUE
0318'   0400                .WORD   1024    ;150 BAUD TIMER VALUE
031A'   0200                .WORD   512     ;300 BAUD TIMER VALUE
031C'   0100                .WORD   256     ;600 BAUD TIMER VALUE
031E'   0080                .WORD   128     ;1200 BAUD TIMER VALUE
0320'   0055                .WORD   85      ;1800 BAUD TIMER VALUE
0322'   004D                .WORD   77      ;2000 BAUD TIMER VALUE
0324'   0040                .WORD   64      ;2400 BAUD TIMER VALUE
0326'   002B                .WORD   43      ;3600 BAUD TIMER VALUE
0328'   0020                .WORD   32      ;4800 BAUD TIMER VALUE
032A'   0015                .WORD   21      ;7200 BAUD TIMER VALUE
```

```
032C'   0010                    .WORD   16       ;9600 BAUD TIMER VALUE
032E'   0008                    .WORD   8        ;19200 BAUD TIMER VALUE
                        ;
0330'   21 000E"        SERRBR: LXI     H,S0BR   ;GET SERIAL 0 BAUD RATE
0333'   78                      MOV     A,B      ;GET CHANNEL NUMBER
0334'   B7                      ORA     A        ;CHANNEL NUMBER=0?
0335'   2803                    JRZ     ..COM    ;IF SO, CONTINUE
0337'   21 0028"                LXI     H,S1BR   ;ELSE, GET SERIAL 1 BAUD RATE
033A'   7E              ..COM:  MOV     A,M      ;GET CURRENT BAUD RATE CODE
033B'   C9                      RET              ;DONE
                        ;
033C'   3EEA            SERSMC: MVI     A,0EAH   ;GET WRITE REGISTER 5 CONTROL WORD
033E'   E67D                    ANI     #82H     ;STRIP RTS/CTS CONTROL BITS
0340'   CB79                    BIT     7,C      ;RTS REQUESTED?
0342'   2802                    JRZ     ..NRTS
0344'   CBCF                    SET     1,A      ;IF SO, SET RTS BIT
0346'   CB71            ..NRTS: BIT     6,C      ;DTR REQUESTED?
0348'   2802                    JRZ     ..NDTR
034A'   CBFF                    SET     7,A      ;IF SO, SET DTR BIT
034C'   57              ..NDTR: MOV     D,A      ;REQUESTED MODEM CONTROLS TO D-REG
034D'   0E01                    MVI     C,SIOACR  ;GET SIO PORT A CONTROL REGISTER
034F'   78                      MOV     A,B      ;GET CHANNEL NUMBER
0350'   B7                      ORA     A        ;CHANNEL NUMBER=0?
0351'   2802                    JRZ     ..COM    ;IF SO, CONTINUE
0353'   0E03                    MVI     C,SIOBCR  ;GET SIO PORT B CONTROL REGISTER
0355'   3E05            ..COM:  MVI     A,5      ;GET WRITE REGISTER 5
0357'   F3                      DI               ;DISABLE INTERRUPTS
0358'   ED79                    OUTP    A        ;SELECT WRITE REGISTER 5
035A'   ED51                    OUTP    D        ;OUTPUT CONTROL WORD
035C'   FB                      EI               ;ENABLE INTERRUPTS
035D'   C9                      RET              ;DONE
                        ;
035E'   0E01            SERRMC: MVI     C,SIOACR  ;GET SIO PORT A CONTROL REGISTER
0360'   78                      MOV     A,B      ;GET CHANNEL NUMBER
0361'   B7                      ORA     A        ;CHANNEL NUMBER=0?
0362'   2802                    JRZ     ..COM    ;IF SO, CONTINUE
0364'   0E03                    MVI     C,SIOBCR  ;GET SIO PORT B CONTROL REGISTER
0366'   3E10            ..COM:  MVI     A,10H    ;GET RESET EXTERNAL STATUS COMMAND
0368'   ED79                    OUTP    A        ;RESET EXTERNAL STATUS
036A'   ED50                    INP     D        ;GET SIO MODEM STATUS
036C'   AF                      XRA     A        ;CLEAR RETURN VECTOR
036D'   CB6A                    BIT     CTS,D    ;CTS SET?
036F'   2802                    JRZ     ..NCTS   ;IF NOT, CONTINUE
0371'   CBFF                    SET     7,A      ;ELSE, SET CTS BIT
0373'   CB5A            ..NCTS: BIT     DCD,D    ;DCD SET?
0375'   C8                      RZ               ;IF NOT, DONE
0376'   CBEF                    SET     5,A      ;ELSE, SET DCD BIT
0378'   C9                      RET              ;DONE
                        ;
                        .PRGEND
```

SLVRES - SAMPLE SLAVE RESET MODULE
FOR TURBODOS ON SLAVE Z80 PROCESSOR

```
                         ;
                         ; VERSION: 01/22/84
                         ;
                         .IDENT  SLVRES          ;MODULE ID
                         ;
                         .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                         ;
0000"                           .LOC    .DATA.# ;LOCATE IN DATA AREA
                         ;
0000"   1F1F             SLRSEQ::.BYTE  AUS,AUS ;SLAVE RESET SEQUENCE
0002"   FFFFFF                  .BYTE   -1,-1,-1
0005"   0000"            SLRPTR: .WORD  SLRSEQ  ;SEQUENCE POINTER
                         ;
0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                         ;
0000'   2A 0005"         SLVRES::LHLD   SLRPTR  ;GET SEQUENCE POINTER
0003'   79                      MOV     A,C     ;GET CONSOLE CHARACTER
0004'   E67F                    ANI     7FH     ;STRIP PARITY
0006'   BE                      CMP     M       ;DOES IT MATCH SEQUENCE?
0007'   2009                    JRNZ    ..NE    ;IF NOT, CONTINUE
0009'   23                      INX     H       ;ELSE, INCREMENT POINTER
000A'   22 0005"                SHLD    SLRPTR  ;UPDATE POINTER
000D'   7E                      MOV     A,M     ;GET NEXT CHAR IN SEQUENCE
000E'   3C                      INR     A       ;END-OF-SEQUENCE?
000F'   C0                      RNZ             ;IF NOT, RETURN
0010'   F3                      DI              ;ELSE, DISABLE INTERRUPTS
0011'   76                      HLT             ;HALT
0012'   21 0000"         ..NE:  LXI     H,SLRSEQ  ;RESET SEQUENCE POINTER
0015'   22 0005"                SHLD    SLRPTR
0018'   C9                      RET             ;DONE
                         ;
                         .END
```