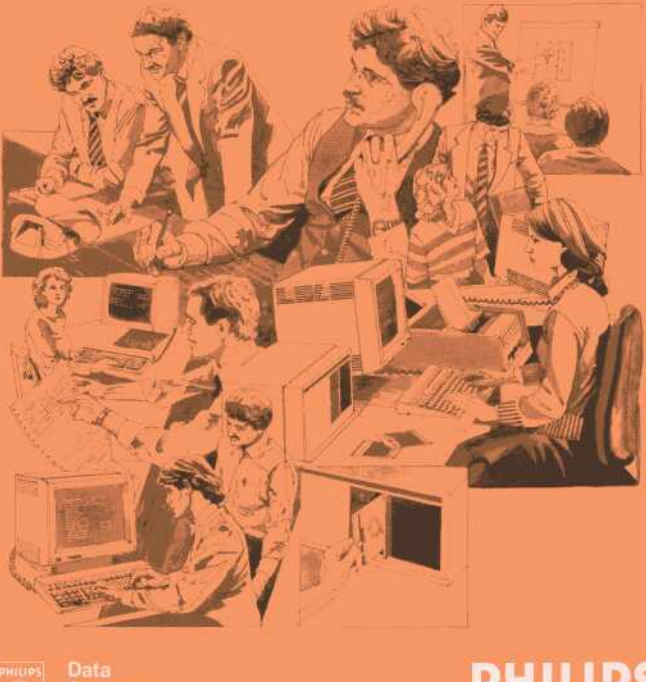
P3500

# TURBODOS configuration guide

F52H internal library



Data Systems

PHILIPS

Module Reference Number : F052H-00A-00E

Office Micro Systems ........ F

Module Number ........ 0

Internal Library ........ 5

Not Relevant ........ 2

Not Relevant ........ H

Complete Module ........ 00A

Initial Edition ........ 00

English Version ........ E

A publication of

Philips Data Systems
SSS - Training & Documentation
Apeldoorn, The Netherlands

Order Number 5122 993 62931

# Configuration Guide to TurboDOS 1.2

May, 1982

## TABLE OF CONTENTS

**Configuration Guide to TurboDOS 1.2**

Table of Contents

SECTION 3 -- SYSTEM IMPLEMENTATION

# INTRODUCTION

This Configuration Guide to TurboDOS provides the information that OEMs, dealers, and sophisticated end-users need to generate various operating system configurations and to implement driver modules for various peripheral components.

A companion document, entitled User's Guide to TurboDOS, provides the information that users need to write and run programs under the TurboDOS operating system. It includes an overview of operating system features, a discussion of architecture and theory of operation, a description of each command, and a definition of each user-callable function.

## Generating TurboDOS Configurations

TurboDOS is a modular operating system consisting of more than 40 separate functional modules. These modules are "building blocks" which can be combined in various ways to produce a family of compatible operating systems. TurboDOS configurations include single-task, spooling, network master and network slave, with numerous subtle variations possible in each of these broad categories.

Functional modules of TurboDOS are distributed in relocatable form. Hardware-dependent device drivers are packaged in the same fashion. The GEN command is a specialized linkage editor which may be used to combine the desired combination of modules into an executable version of TurboDOS configured with the desired set of functions and device drivers. The GEN command also includes a symbolic patch facility which may be used to alter a variety of operating system parameters.

Section 2 describes each functional module of TurboDOS in detail, illustrates how these modules can be combined in various configurations, and provides step-by-step system generation procedures.

## Implementating Driver Modules

TurboDOS has been designed to run on any Z80-based microcomputer with at least 48K of RAM, a random-access mass storage device, and a full-duplex character-oriented console device (or on an interconnected network of such microcomputers). The functional modules of TurboDOS are not dependent upon the specific peripheral devices to be used. Rather, a set of hardware-dependent device driver modules must be included in each TurboDOS configuration in order to adapt the operating system to the specific hardware environment.

Typical hardware-dependent device driver modules include:

- Console driver
- Printer driver
- Disk driver
- Network circuit driver
- Real-time clock driver
- Communications driver

Although Software 2000 Inc. can supply TurboDOS pre-configured for certain specific hardware configurations, most OEMs and many dealers and end-users will want to implement their own hardware-dependent drivers. Driver modules may be readily written by any competant assembly-language programmer, using a relocating Z80 assembler such as Digital Research's RMAC, Microsoft's MACRO-80, or Phoenix Software Associates' PASM. Section 3 provides detailed instructions to programmers for implementing such driver modules, and the Appendix includes assembly listings of various sample drivers.

## Licensing Requirements

TurboDOS is a proprietary software product of Software 2000 Inc. TurboDOS may be used only after the user has paid the required license fee, signed a copy of the TurboDOS software license agreement, and returned the signed agreement to Software 2000 Inc. Then it may be used only in strict conformance with the terms of the software license. Each TurboDOS software license agreement must be filled-out and signed by the end-user (not by an OEM or dealer on his customer's behalf).

Each software license permits the use of TurboDOS only on one specific computer system identified by make, model and serial number. A separate license fee must be paid and a separate license signed for each computer system on which TurboDOS is used. Network slave computers which are also capable of stand-alone operation under TurboDOS must each be licensed separately, but slave computers which cannot be used stand-alone (e.g., because they have no mass storage) do not.

Software 2000 Inc. intends to initiate vigorous legal action against anyone who uses or reproduces TurboDOS software in a manner which is not in strict conformance with the terms of the TurboDOS software license agreement.

## Serialization

Each copy of TurboDOS is magnetically serialized with a unique serial number in order to facilitate tracing of unlicensed copies of TurboDOS.

Each relocatable TurboDOS module which is distributed to a dealer or end-user is magnetically serialized with a unique serial number. The serial number consists of two components: an origin number (which identifies the issuing OEM) and a unit number (which uniquely identifies each copy of TurboDOS issued by that OEM). The GEN command verifies that all functional modules which make up a TurboDOS configuration are serialized consistently, and magnetically serializes the resulting executable version of TurboDOS accordingly.

Each relocatable TurboDOS module which is distributed to an OEM is partially serialized with an origin number only. Each OEM is provided with a SERIAL command which must be used to add a unique unit number to the relocatable modules of each copy of TurboDOS issued by that OEM. The GEN command will not accept partially serialized modules that have not been uniquely serialized by the OEM. Conversely, the SERIAL command will not re-serialize modules which have already been fully serialized.

## OEM Responsibilities

Each OEM is provided with a master copy of TurboDOS relocatable modules and command processors on diskette. An OEM is authorized to reproduce and distribute copies of TurboDOS to dealers and end-users for use on specifically authorized hardware configurations manufactured or distributed by the OEM. The OEM is required to serialize each copy of TurboDOS with a unique sequential magnetic serial number, and to register each serial number promptly by returning a registration card to Software 2000 Inc. This registration requirement for OEMs is in addition to (not in lieu of) the requirement for licensing of each end-user.

Each OEM is provided with a master copy of TurboDOS documentation either in camera-ready form or in ASCII files on diskette. The OEM is responsible for reproducing the documentation and providing it with each copy of TurboDOS issued by that OEM.

An OEM must require a dealer to sign the TurboDOS dealer agreement and return it to Software 2000 Inc. before the OEM may issue copies of TurboDOS to that dealer. An OEM must require an end-user to sign the TurboDOS software license and return it to Software 2000 Inc. before the OEM may issue a copy of TurboDOS directly to that end-user.

## Dealer Responsibilities

A TurboDOS dealer is permitted to purchase individual serialized copies of TurboDOS software and documentation from an authorized TurboDOS OEM, and to resell them to end-users. Dealers are not authorized to make copies of TurboDOS software or documentation for any purpose whatever.

A TurboDOS dealer must require each end-user to sign the TurboDOS software license and return it to Software 2000 Inc. before issuing a copy of TurboDOS software or documentation to the end-user.

## TurboDOS Support

Software 2000 maintains a telephone "hot-line" to provide TurboDOS-related technical assistance to its OEMs. Authorized TurboDOS OEMs should feel free to take advantage of this service whenever technical questions arise concerning the use or configuration of TurboDOS.

It is the responsibility of each OEM 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 with Software 2000 by the OEM.

## SYSTEM GENERATION

TurboDOS is a modular operating system consisting of more than 40 separate functional modules. These modules are "building blocks" which can be combined in various ways to produce a family of compatible operating systems. TurboDOS configurations include single-task, spooling, time-sharing and networking, with numerous subtle variations possible in each of these broad categories. This section describes each functional module of TurboDOS in detail, illustrates how these modules can be combined in various configurations, and provides step-by-step system generation procedures.

Functional modules of TurboDOS are distributed in relocatable form. Hardware-dependent device drivers are packaged in the same fashion. The GEN command processor is a specialized linkage editor which may be used to bind together the desired combination of modules into an executable version of TurboDOS configured with the desired set of functions and device drivers. GEN also includes a symbolic patch facility which may be used to alter a variety of operating system parameters.

To simplify the the system generation process, the most commonly used combinations of TurboDOS functional modules are pre-packaged into several standard configurations. Most requirements for TurboDOS can be satisfied by linking the appropriate standard package together with the requisite hardware-dependent drivers.

## Module Hierarchy

The flow diagram on the facing page illustrates the functional inter-relationship of TurboDOS modules. As the diagram shows, the software elements of TurboDOS can be viewed as a three-level hierarchy.

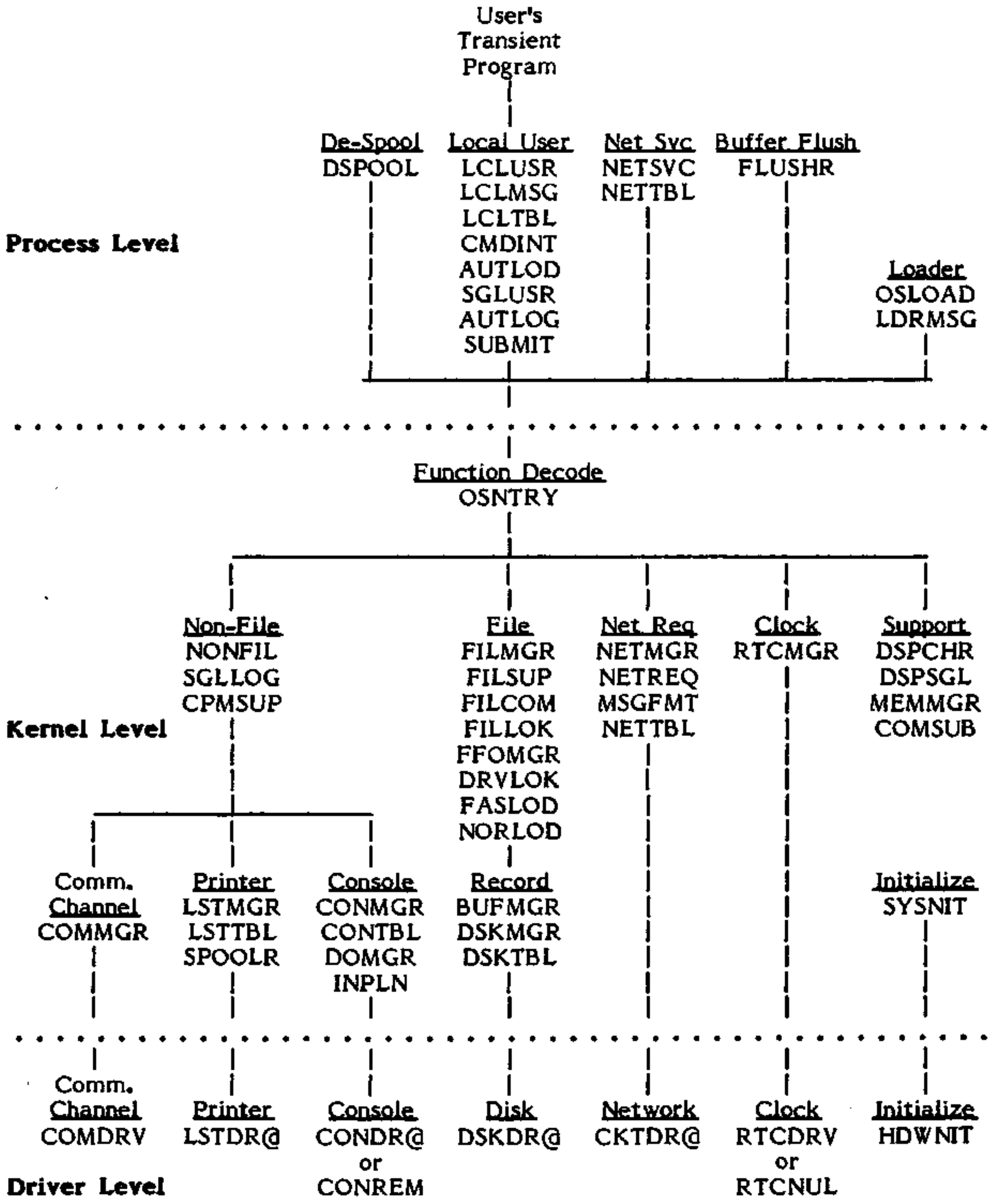
The highest level is known as the "process" level. TurboDOS can support many concurrent processes at this level, and can share the resources of the local computer among them. There are active processes for users who are executing commands and/or transient programs on the local computer. There are also processes for users who are running on remote computers but making network requests of the local computer. There are processes to support de-spooling on local printers. Finally, there is a process which periodically causes buffered disk records to be flushed (i.e., written out) to disk.

The intermediate level is known as the "kernel" level. The kernel supports the various numbered TurboDOS functions (about 100 of them), and controls the sharing of microcomputer resources such as processor time, memory, peripheral devices, and disk files. Processes make requests of the kernel through a single entrypoint (OSNTRY) which decodes each function by number and invokes the appropriate module in the kernel.

The lowest level is known as the "driver" level, and contains all of the device-dependent drivers necessary to interface TurboDOS to a particular configuration of microcomputer hardware. Drivers must be provided for each printer, console, disk controller, and network interface. A driver is also required for the real-time clock or other periodic interrupt source (used for time-slicing among processes and for timing of delays). TurboDOS operates most efficiently with interrupt-driven, buffered or DMA-type devices, but can also work satisfactorily with polled and programmed-I/O devices.

The TurboDOS loader OSLOAD.COM is a special program which contains an abbreviated version of the kernel and drivers. Its purpose is to load the full operating system into memory at each system start-up.

All TurboDOS process-level and kernel-level modules permit re-entrant execution in multi-process situations. Most driver-level modules are not re-entrantly coded, and must utilize a mutual-exclusion mechanism to prevent re-entrant execution.



**TurboDOS Module Hierarchy**

## Process-Level Modules

**LCLUSR** -- Supports a transient program area for a user of the local microcomputer.

**LCLMSG** -- Contains all operating system messages, which are segregated into a separate module to facilitate adaptation to other languages.

**LCLTBL** -- Local user initialization tables.

**CMDINT** -- Command interpreter routine called by LCLUSR to process local user commands and multi-command strings.

**AUTLOD** -- Automatic program load routine called by LCLUSR to process COLDSTRT.AUT and WARMSTRT.AUT files if they are present.

**SGLUSR** -- Buffer flushing routine called by LCLUSR to flush and unlink all disk buffers at every console input. Included in single-user configurations only.

**AUTLOG** -- Automatic log-on routine called by LCLUSR to automatically log-on the local user in configurations where logon/logoff security is not desired. To activate this feature, use the symbolic patch facility to patch the public symbol AUTUSR to the desired user number, with the sign-bit set for a privileged log-on (typically AUTUSR = 80).

**SUBMIT** -- Optional module which emulates the processing by CP/M of $$$.SUB files (not recommended due to significant performance penalty).

**NETSVC** -- Network service process which receives and services network requests from other microcomputers.

**NETTBL** -- Tables which define the topology of the network as seen from a particular processor.

**DSPOOL** -- De-spool process which supports printing of spooled print jobs concurrent with other system activities. In multi-printer configurations, there is a separate re-entrant instance of the DSPOOL process for each printer.

**FLUSHR** -- Buffer flusher process which causes memory-resident disk buffers to be flushed (i.e., written out) to disk periodically.

## Kernel-Level Modules

**OSNTRY** -- Common kernel entrypoint which decodes each function by number and invokes the appropriate module in the kernel.

**FILMGR** -- File manager which processes requests involving local files.

**FILSUP** -- File support routines used by FILMGR.

**FILCOM** -- Processors for common file-oriented functions which are never sent over the network.

**FILLOK** -- File- and record-level interlock routines called by FILMGR.

**FFOMGR** -- FIFO management routines called by FILLOK.

**DRVLOK** -- Drive interlock routines.

**FASLOD** -- Program load optimizer.

**NORLOD** -- Non-optimized program load routine which may be used instead of FASLOD when memory space is at a premium.

**BUFMGR** -- Buffer manager called by FILMGR. It maintains a pool of memory-resident record buffers used for all record-oriented access to local disk storage.

**DSKMGR** -- Disk manager called by BUFMGR and FASLOD to perform physical accesses to local disk storage.

**DSKTBL** -- Tables which define how drive letters are mapped into local and remote drives, and the location on the network of any remote drives.

**NONFIL** -- Non-file request manager which handles kernel requests which are not file-oriented.

**CPMSUP** -- Optional function processors for little-used functions (7, 8, 24, 28, 29, 31, 37, and 107) included in TurboDOS solely for compatibility with CP/M.

**SGLLOG** -- Optional module which may be included in multi-user configurations to prevent two or more non-privileged users from logging-on to the same user number concurrently.

**CONMGR** -- Console manager which handles local console input/output.

**CONTBL** -- Table which defines how the local console is interfaced.

**DOMGR** -- DO-file manager which handles activation of DO-files. When a DO-file is active, this module is called by CONMGR to satisfy console input requests from the DO-file.

**INPLN** -- Console input line editor used for buffered console input (function 10), and used by CMDINT.

**LSTMGR** -- List manager which handles printer-oriented functions.

**LSTTBL** -- Tables which define how printer and queue letters are mapped into local and remote printers and queues, and the location on the network of any remote printers and queues.

**SPOOLR** -- Spooler routine which diverts print output to spool files when the spooler is activated. Also handles direct printing to remote printers.

**COMMGR** -- Comm channel manager which handles the communications channel.

**NETREQ** -- Network request processor which creates network request messages to be passed to remote processors for service.

**MSGFMT** -- Network message format table used by NETREQ.

**NETMGR** -- Network message routing mechanism used by NETSVC and NETREQ.

**NETTBL** -- Tables which define the topology of the network as seen from a particular processor.

**RTCMGR** -- Real-time clock manager which maintains system date and time.

**DSPCHR** -- Multi-process dispatcher which controls the sharing of local processor time among multiple competing processes.

**DSPSGL** -- Null dispatcher used as an alternative to DSPCHR when only one process is required (e.g., in OSLOAD.COM and in minimal single-user configurations without spooling).

**MEMMGR** -- Memory manager which controls the dynamic allocation and deallocation of memory segments.

**COMSUB** -- Common subroutines utilized in all configurations.

**SYSNIT** -- System initialization routine which is executed at system start-up.

**PATCH** -- Optional module consisting of 64 bytes of zeroes which may be included to provide space for any required operating system patches.

## Universal Driver-Level Modules

**RTCNUL** -- Null real-time clock driver for use in configurations in which there is no periodic interrupt source.

**CONREM** -- Remote console driver for network master to allow access from slave consoles by means of the MASTER command.

## Hardware-Dependent Driver-Level Modules

Driver modules are hardware-dependent, and may vary significantly from one TurboDOS implementation to another. In general, the following drivers are required as a minimum:

**CONDR@** -- Console driver allows character-by-character input from a console keyboard and output to a console display.

**LSTDR@** -- Printer driver allows character-by-character output to a hardcopy peripheral. TurboDOS supports multiple printer drivers.

**COMDRV** -- Comm. channel driver allows character-by-character input and output over one or more communications channels.

**DSKDR@** -- Disk controller driver allows input and output of physical-records on a random-access mass storage device (usually flexible or hard disk). TurboDOS supports multiple disk controller drivers, each of which may support multiple drives.

**CKTDR@** -- Network circuit driver allows sending and receiving messages to or from remote processors. TurboDOS supports multiple network circuit drivers, each of which may communicate with multiple remote processors.

**RTCDRV** -- Real-time clock driver services interrupts from a periodic interrupt source, used for time-slicing, delay measurement, and updating the system date and time.

**HDWNIT** -- Hardware initialization routine called by SYSNIT. This module usually consists of calls to initialization entrypoints in other drivers.

## Standard Configurations

To simplify the the system generation process, the most commonly used combinations of TurboDOS functional modules are pre-packaged into the standard configurations shown in the table on the facing page: STDLOADR, STDSINGL, STDSPOOL, STDMASTR, STDSLAVE and STDSLAVX. Most requirements for TurboDOS can be satisfied by linking the appropriate standard package together with the requisite driver modules.

| | | | | Single User | Network | Simple | Complex |
|---|---|---|---|---|---|---|---|
| | Size | O/S Loader | Single User | w/Spooling | Master | Slave | Slave |
| Module | Kb | STDLOADR | STDSINGL | STDSPOOL | STDMASTR | STDSLAVE | STDSLAVX |
| LCLUSR | .9 | - | LCLUSR | LCLUSR | LCLUSR | LCLUSR | LCLUSR |
| LCLMSG | .1 | - | LCLMSG | LCLMSG | LCLMSG | LCLMSG | LCLMSG |
| LCLTBL | .0 | - | LCLTBL | LCLTBL | LCLTBL | LCLTBL | LCLTBL |
| CMDINT | .9 | - | CMDINT | CMDINT | CMDINT | CMDINT | CMDINT |
| AUTLOD | .2 | - | AUTLOD | AUTLOD | AUTLOD | AUTLOD | AUTLOD |
| SGLUSR | .1 | - | SGLUSR | SGLUSR | - | - | SGLUSR |
| AUTLOG | .0 | - | AUTLOG | AUTLOG | AUTLOG | AUTLOG | AUTLOG |
| SUBMIT | .1 | - | - | - | - | - | - |
| NETSVC | 1.0 | - | - | - | NETSVC | - | - |
| DSPOOL | .8 | - | - | DSPOOL | DSPOOL | - | DSPOOL |
| FLUSHR | .2 | - | - | - | FLUSHR | - | - |
| OSLOAD | 1.4 | OSLOAD | - | - | - | - | - |
| LDRMSG | .1 | LDRMSG | - | - | - | - | - |
| OSNTRY | .4 | OSNTRY | OSNTRY | OSNTRY | OSNTRY | OSNTRY | OSNTRY |
| FILMGR | 1.4 | FILMGR | FILMGR | FILMGR | FILMGR | - | FILMGR |
| FILSUP | 1.9 | FILSUP | FILSUP | FILSUP | FILSUP | - | FILSUP |
| FILCOM | .2 | FILCOM | FILCOM | FILCOM | FILCOM | FILCOM | FILCOM |
| FILLOK | 1.3 | - | - | - | FILLOK | - | - |
| FFOMGR | .7 | - | - | - | FFOMGR | - | - |
| DRVLOK | .2 | - | - | - | DRVLOK | - | - |
| FASLOD | .3 | - | FASLOD | FASLOD | FASLOD | - | FASLOD |
| NORLOD | .1 | - | - | - | - | - | - |
| BUFMGR | 1.0 | BUFMGR | BUFMGR | BUFMGR | BUFMGR | - | BUFMGR |
| DSKMGR | .6 | DSKMGR | DSKMGR | DSKMGR | DSKMGR | - | DSKMGR |
| DSKTBL | .0 | DSKTBL | DSKTBL | DSKTBL | DSKTBL | DSKTBL | DSKTBL |
| NONFIL | .1 | NONFIL | NONFIL | NONFIL | NONFIL | NONFIL | NONFIL |
| SGLLOG | .1 | - | - | - | - | - | - |
| CPMSUP | .2 | - | - | - | - | - | - |
| CONMGR | .3 | CONMGR | CONMGR | CONMGR | CONMGR | CONMGR | CONMGR |
| CONTBL | .0 | CONTBL | CONTBL | CONTBL | CONTBL | CONTBL | CONTBL |
| DOMGR | .3 | - | DOMGR | DOMGR | DOMGR | DOMGR | DOMGR |
| INPLN | .1 | - | INPLN | INPLN | INPLN | INPLN | INPLN |
| LSTMGR | .1 | - | LSTMGR | LSTMGR | LSTMGR | LSTMGR | LSTMGR |
| LSTTBL | .1 | - | LSTTBL | LSTTBL | LSTTBL | LSTTBL | LSTTBL |
| SPOOLR | .4 | - | - | SPOOLR | SPOOLR | SPOOLR | SPOOLR |
| COMMGR | .1 | - | COMMGR | COMMGR | COMMGR | COMMGR | COMMGR |
| NETREQ | .9 | - | - | - | - | NETREQ | NETREQ |
| MSGFMT | .1 | - | - | - | - | MSGFMT | MSGFMT |
| NETMGR | .3 | - | - | - | NETMGR | NETMGR | NETMGR |
| NETTBL | .0 | - | - | - | NETTBL | NETTBL | NETTBL |
| RTCMGR | .1 | - | RTCMGR | RTCMGR | RTCMGR | - | RTCMGR |
| DSPCHR | .6 | - | - | DSPCHR | DSPCHR | DSPCHR | DSPCHR |
| DSPSGL | .1 | DSPSGL | DSPSGL | - | - | - | - |
| MEMMGR | .3 | - | MEMMGR | MEMMGR | MEMMGR | MEMMGR | MEMMGR |
| COMSUB | .2 | COMSUB | COMSUB | COMSUB | COMSUB | COMSUB | COMSUB |
| SYSNIT | .0 | - | SYSNIT | SYSNIT | SYSNIT | SYSNIT | SYSNIT |

## Estimating Memory Requirements

To estimate memory requirements for a particular TurboDOS configuration, it is necessary to take into account the combined size of functional modules (see table on previous page), hardware-dependent driver modules, disk buffers and other dynamically allocated storage segments.

Hardware-dependent drivers typically require 1K to 3K of memory, depending on the complexity of the hardware involved. Disk buffer space should be as large as possible for optimum performance, especially in a network master. About 4K of disk buffer space is acceptable for a single-user system, although less can be used in a pinch. Other dynamic storage usually doesn't exceed 1K in a single-user system, 2K in a networking system.

The following table gives typical memory requirements of standard TurboDOS configurations:

| | O/S Loader | Single User | Single User w/Spooling | Network Master | Simple Slave | Complex Slave |
|---|---|---|---|---|---|---|
| | STDLOADR | STDSINGL | STDSPOOL | STDMASTR | STDSLAVE | STDSLAVX |
| TurboDOS | 8K | 10K | 12K | 15K | 7K | 14K |
| Drivers | 2K | 2K | 2K | 3K | 1K | 3K |
| Disk Buffers | 4K | 4K | 4K | 16K | 0K | 4K |
| Dynamic Space | ±1K | ±1K | ±1K | ±3K | ±2K | ±2K |
| Total Size | 15K | 17K | 19K | 37K | 10K | 23K |
| TPA (in 64K) | n/a | 47K | 45K | 27K | 54K | 41K |

Typical TurboDOS Memory Requirements

## Linking and Loading

Functional modules of TurboDOS are distributed in relocatable form. Hardware-dependent device drivers are packaged in the same fashion. The GEN command processor is a specialized linkage editor which may be used to bind together the desired combination of modules into an executable version of TurboDOS configured with the desired set of functions and device drivers. GEN also includes a symbolic patch facility which may be used to alter a variety of operating system parameters.

To generate a TurboDOS system, the GEN command must be used to create both an executable loader OSLOAD.COM and an executable master operating system OSMASTER.SYS. In networking configurations, the GEN command must also be used to create a slave operating system OSSLAVE.SYS. The GEN command can also be used to generate the code for a start-up PROM (or boot track).

At system start-up, the start-up PROM (or boot track) loads the loader program OSLOAD.COM into the TPA of the master computer and executes it. OSLOAD loads the master operating system OSMASTER.SYS into the topmost portion of memory. In networking configurations, the master operating system down-loads the slave operating system OSSLAVE.SYS into the slave computers on the network.

## GEN Command

The GEN command is used for TurboDOS system generation (and may also be used as a general purpose linker for Microsoft-format .REL modules). It links a collection of relocatable modules together into a single executable file. The command format is:

```
GEN filename1 filename2 ;options
```

where "filename1" specifies the name of the configuration file (type .GEN) and parameter file (type .PAR) to be used, and "filename2" specifies the name of the executable file (normally type .COM or .SYS) to be created. If "filename2" is omitted from the command line, then "filename1" is used for the executable file and should include an explicit file type (.COM or .SYS).

If the configuration file (type .GEN) is found, it must contain the list of relocatable files to be linked together. If the configuration file is not found, then the GEN command operates in an interactive mode, reading successive directives from the console until terminated by a null directive. The format of each directive (or each line of the configuration file) is:

```
relfile1, relfile2, ..., relfileN
```

The GEN command links together all of the specified modules, a two-pass process which displays the name of each module as it is encountered. At the end of the second pass, the GEN command looks for a parameter file (type .PAR) and processes it (if found). Finally, the executable file is written out to disk.

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 OEM) and a unit number (which uniquely identifies each copy of TurboDOS issued by that OEM). The GEN command verifies that all modules to be linked are serialized consistently, and serializes the executable file accordingly.

The ";options" argument may contain either ";Lxxxx" or ";Uxxxx" to define either the lower or upper boundary of the executable program ("xxxx" is a hexadecimal memory address). The default boundary is ";L0100" if the output file is of type .COM, and ";UFFFF" if the output file is of type .SYS.

The ";options" argument may also contain ";X" to display undefined symbol references (quite normal in TurboDOS system generation), ";M" to print a load map on the printer, and ";S" to print a full symbol table on the printer.

Example:

The following example uses the GEN command to link the modules listed in OSMASTER.GEN and the patch parameters in OSMASTER.PAR, creating the executable file OSMASTER.SYS.

```
0A}GEN OSMASTER.SYS ;UBFFF
* STDSINGL, CON192, LSTCTS, SPD442
* SER480, BRT442O, RTC442
* DSK401, DSTBL8, HDWNIT
Pass 1.
LCLUSR    LCLTBL    CMDINT    AUTLOD    SGLUSR    AUTLOG
OSNTRY    FILMGR    FILSUP    FASLOD    BUFMGR    DSKMGR
DSKTBL    NONFIL    CONMGR    CONTBL    DOMGR     INPLN
LSTMGR    LSTTBL    COMMGR    RTCMGR    DSPSGL    MEMMGR
COMSUB    SYSNIT    CON192    LSTCTS    SPD442    SER480
BRT442    RTC442    DSK401    DSTBL8    HDWNIT
Pass 2.
LCLUSR    LCLTBL    CMDINT    AUTLOD    SGLUSR    AUTLOG
OSNTRY    FILMGR    FILSUP    FASLOD    BUFMGR    DSKMGR
DSKTBL    NONFIL    CONMGR    CONTBL    DOMGR     INPLN
LSTMGR    LSTTBL    COMMGR    RTCMGR    DSPSGL    MEMMGR
COMSUB    SYSNIT    CON192    LSTCTS    SPD442    SER480
BRT442    RTC442    DSK401    DSTBL8    HDWNIT
Processing parameter file:
AUTUSR = 80
NMBUFS = 8
PTRAST = 2
EOPCHR = 1A
SRHDRV = 1
Writing output file.
0A}
```

## Symbolic Patch Facility

The GEN command supports a symbolic patch facility which may be used to override various operating system parameters as well as to effect necessary software corrections. Symbolic patches must be stored in a parameter file (type .PAR), which may be built using any ordinary file editor. The format of each .PAR file entry is:

```
        location = value1, value2, ..., valueN   ;comments
```

where "value1" through "valueN" are to be loaded into consecutive memory locations starting with "location".

The argument "location" may be a public symbol name, a hexadecimal number, or an expression composed of names and hexadecimal numbers connected by "+" or "-". Hexadecimal numbers must begin with a decimal digit (e.g., "0FFFF"). The location expression must be followed by an equal-sign character.

The arguments "value1" through "valueN" may be expressions (as defined above) or quoted ASCII strings, and must be separated by commas. An expression is stored as a 16-bit word if its value exceeds 255 or if it is enclosed in parentheses; otherwise, an expression 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 the circumflex (e.g., "^X" denotes CTRL-X).

Example:

```
        CLBLEN = 9D        ;Command line buffer length (157)
        CLSCHR = "\"       ;Command line separator character
        ATNCHR = "^S"      ;Attention character
        LOADFN = 0,"OSMASTER","SYS"   ;File name to load
        DSKAST = 00,DSKDRA,01,DSKDRA,00,DSKDRB,80,(0000),81,(0000)
```

## TurboDOS Patch Points

Public symbols in the hardware-independent portion of TurboDOS which may be useful to patch are shown below, together with their default (unpatched) values. (Other patchable symbols my exist in hardware-dependent drivers, and are beyond the scope of this document.)

In AUTLOD Module:

| | | |
|---|---|---|
| LDCOLD | = 0FF | Cold-start autoload enable flag (0 to disable) |
| LDWARM | = 0FF | Warm-start autoload enable flag (0 to disable) |
| COLDFN | = 0,"COLDSTRT","AUT" | Cold-start autoload file name (12 bytes) |
| WARMFN | = 0,"WARMSTRT","AUT" | Warm-start autoload file name (12 bytes) |

In AUTLOG Module:

| | | |
|---|---|---|
| AUTUSR | = 0FF | Automatic log-on user number (sign-bit if privileged) |

In BUFMGR Module:

| | | |
|---|---|---|
| BUFSIZ | = 3 | Default buffer size (0=128, 1=256, 2=512,..., 7=16K) |
| NMBUFS | = 4 | Default number of buffers |

In CMDINT Module:

| | | |
|---|---|---|
| CLBLEN | = 9D | Command line buffer length (default 2*80-3 = 157) |
| CLSCHR | = "\" | Command line separator character |
| CLPCHR | = "]" | Command line prompt character |
| SRHDRV | = 0 | Search drive (0=off, 1="A", 2="B",..., 0FF=sysdsk) |

In CONTBL Module:

| | | |
|---|---|---|
| ATNCHR | = "^S" | Attention character |
| ATNBEL | = "^G" | Attention-received warning |
| RESCHR | = "^Q" | Resume character (attention response) |
| ABTCHR | = "^C" | Abort character (attention response) |
| ECOCHR | = "^P" | Echo character (attention response) |
| PRTCHR | = "^L" | End-print character (attention response) |
| CONAST | = 00,CONDRA | Console assignment table |

In DSKTBL Module:

```
DSKAST    = 00,DSKDRA,01,DSKDRA,0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
                          Disk assignment table (master, 16 3-byte entries)
```

In FILCOM Module:

```
LOGUSR    = 1F            User number for log-off (standard is 31)
```

In FILLOK Module:

```
COMPAT    = 00            File/record locking compatibility flags
```

In FLUSHR Module:

```
BFLDLY    = (012C)        Buffer flush delay (in ticks, no flush if zero)
```

In LCLTBL Module:

```
PRTMOD    = 1             Default print mode (0=direct, 1=spooled, 2=console)
QUEPTR    = 1             Default queue/printer (0=off, 1="A", 2="B", ...)
SPLDRV    = 0FF           Default spool drive (0="A", 1="B", ..., 0FF=sysdsk)
```

In LCLUSR Module:

```
MEMRES    = (0100)        Reserved memory above TPA
```

In LSTTBL Module:

```
EOPCHR    = 0             End-of-print character (if nonzero)
DSPPAT    = 1,...,1       De-spool printer assignment table (16 bytes)
PTRAST    = 00,LSTDRA,0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
                          Printer assignment table (master, 16 3-byte entries)
QUEAST    = 00,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
            0FF,(0),0FF,(0),0FF,(0),0FF,(0)
                          Queue assignment table (master, 16 3-byte entries)
```

In MEMMGR Module:

```
MEMBLL    = (1103)      Memory base lower limit (standard assures 4K TPA)
```

In NETMGR Module:

```
NMBMBS    = 0           Number of pre-allocated message buffers
```

In NETSVC Module:

```
NMBSVC    = 2           Number of NETSVC server processes activated
SLVFN     = "OSSLAVE ","SYS"
                        Name of .SYS file to download
```

In NETTBL Module:

```
NMBCKT    = 1           Number of network circuits
DEFDID    = (0)         Default network destination ID
CKTAST    = (0000),CKTDRA,(0100),CKTDRB,(0200),CKTDRC,(0300),CKTDRD
                        Circuit assignment table (NMBCKT 4-byte entries)
FWDTBL    = 0FF,0FF,0FF,0FF,0FF,0FF,0FF,0FF,0FF
                        Forwarding table (2-byte entries)
```

In NONFIL Module:

```
CPMVER    = 30          CP/M BDOS version number (returned by function 12)
```

In OSLOAD Module:

```
LOADFN    = 0,"OSMASTER","SYS"
                        Default drive and filename for OSLOAD (12 bytes)
MEMTOP    = (0FFFF)     Top limit of OSLOAD RAM test (don't test if zero)
```

In SUBMIT Module:

```
SUBFN     = 0,"$$$     ","SUB"
                        Submit file name
```

**NOTE:** In slave configurations STDSLAVE and STDSLAVX, the following default values change:

In DSKTBL Module:

```
DSKAST    = 80,(0),81,(0),82,(0),83,(0)
            84,(0),85,(0),86,(0),87,(0)
            88,(0),89,(0),8A,(0),8B,(0)
            8C,(0),8D,(0),8E,(0),8F,(0)
                      Disk assignment table (slave, 16 3-byte entries)
```

In LSTTBL Module:

```
PTRAST    = 80,(0),81,(0),82,(0),83,(0)
            84,(0),85,(0),86,(0),87,(0)
            88,(0),89,(0),8A,(0),8B,(0)
            8C,(0),8D,(0),8E,(0),8F,(0)
                      Printer assignment table (slave, 16 3-byte entries)
QUEAST    = 80,(0),81,(0),82,(0),83,(0)
            84,(0),85,(0),86,(0),87,(0)
            88,(0),89,(0),8A,(0),8B,(0)
            8C,(0),8D,(0),8E,(0),8F,(0)
                      Queue assignment table (slave, 16 3-byte entries)
```

## Explanations of Certain TurboDOS Patch Points

AUTUSR may be patched to cause an automatic log-on at cold-start time (rather than the usual password-protected log-on procedure). If automatic log-on is desired, patch AUTUSR to the desired user number (00...1F) and set the sign-bit if a privileged log-on is desired. The patch "AUTUSR = 80" should generally be included in single-user configurations to cause an automatic privileged log-on to user number zero.

SRHDRV may be patched to cause TurboDOS to automatically search another drive for a command processor (.COM file) if it is not found on the current default drive. Patch SRHDRV to 1 to search drive "A", 2 to search drive "B", etc. Alternatively, patch SRHDRV to 0FF to search the system disk drive (i.e., whichever drive was used to boot up TurboDOS).

ATNCHR defines the keyboard character interpreted by TurboDOS as an "attention" request. It should be patched if the usual "^S" conflicts with the needs of applications (such as WordStar). A common alternative choice is ATNCHR = "^@" which allows the BREAK key to be used for attention on many systems.

CONAST is a 3-byte entry which defines how console input/output is handled. The first byte is passed to the console driver module, and commonly defines the channel number (e.g., serial port) to be used for the console. The next word specifies the entrypoint address of the console driver to be used.

DSKAST is an array of sixteen 3-byte entries, one for each drive letter A...P, which define whether the corresponding drive is local, remote, or invalid.

- For a local drive, the first byte must not have the sign-bit set. That byte is passed to the disk driver module, and is commonly used to differentiate between multiple drives connected to a single controller. The next word specifies the entrypoint 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 specifies the drive letter to be accessed on the remote processor. The next word specifies the network address of the remote processor.

- For an invalid drive, the first byte must be 0FF, and the next word should be zero.

DSPPAT is an array of sixteen bytes, one for each printer letter A...P, which defines the initial de-spool queue to which each printer is assigned. Byte values 1...16 correspond to queues A...P, and zero means that the corresponding printer is initialized to off-line.

PTRAST is an array of sixteen 3-byte entries, one for each printer letter A...P, which define whether the corresponding printer is local, remote, or invalid.

- For a local printer, the first byte must not have the sign-bit set. That byte is passed to the list driver module, and commonly defines the channel number (e.g., serial port) to be used for the printer. The next word specifies the entrypoint address of the list driver to be used.

- For a remote printer, the first byte must have the sign-bit set. The low-order bits of that byte specifies the printer letter to be accessed on the remote processor. The next word specifies the network address of the remote processor.

- For an invalid printer, the first byte must be 0FF, and the next word should be zero.

<u>QUEAST</u> is an array of sixteen 3-byte entries, one for each queue letter A...P, which define whether the corresponding queue is local, remote, or 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 specifies the queue letter to be accessed on the remote processor. The next word specifies the network address of the remote processor.
- For an invalid queue, the first byte must be 0FF, and the next word should be zero.

<u>EOPCHR</u> may be patched to any non-null ASCII character, in which case the presence of that character in the print output stream will automatically signal an end-of-print-job condition.

<u>NMBMBS</u> is a byte value that specifies the number of network message buffers to pre-allocate at cold-start time. This value may be left at zero, but memory fragmentation may be reduced by assigning a positive value (NMBCKT plus two is a good value to try).

<u>NMBSVC</u> is a byte value that specifies the number of NETSVC server processes to be created (the number of slave processors is a good value to try).

<u>CKTAST</u> is a table of 4-byte (2-word) entries. There are NMBCKT 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.)

<u>FWDTBL</u> is a table of 2-byte entries which define any explicit message forwarding routes which this processor may utilize. The first byte of each entry specifies a circuit number "N" which is <u>not</u> directly connected to this processor, and the second byte specifies a corresponding circuit number "C" which is directly-connected. Any network messages destined for circuit "N" will be routed via circuit "C". This table is variable length (possibly empty), and must be terminated with a byte 0FF.

## Examples of Disk, Printer and Network Tables

Suppose we wish to generate a TurboDOS configuration in which disk drives A...E are to be defined as follows: Drive A is a local hard disk. Drives B and C are local floppies. Drives D and E are remote drives belonging to another processor whose network address is 05-09 (circuit 5, node 9), and are known to that processor as its drives A and B. The .GEN file would include disk driver modules for both hard disk and floppy controllers:

```
..., DSKHARD, DSKFLOP, ...
```

and the .PAR file would set up the disk assignment table for drives A...E as:

```
DSKAST    = 00,DSKDRA, 00,DSKDRB, 01,DSKDRB, 80,(0509), 81,(0509)
```

Furthermore, suppose printers A...E are to be defined as follows: Printers A and B are serial printers using XON/XOFF protocol and connected to serial channels 1 and 2. Printers C and D are serial printers using CTS handshaking and connected to serial channels 3 and 4. Printer E is a remote printer belonging to the processor at network address 02-01 (circuit 2, node 1) and known to that processor as printer B. The .GEN file would include printer drivers for both XON/XOFF and CTS:

```
..., LSTXON, LSTCTS, ...
```

and the .PAR file would set up the printer assignment table for printers A...E as:

```
PTRAST    = 01,LSTDRA, 02,LSTDRA, 03,LSTDRB, 04,LSTDRB, 81,(0201)
```

If de-spooled printing is desired on all five printers, with one queue defined per printer (four local queues and one remote queue), then the .PAR file might set up the queue and de-spool printer assignment tables as:

```
QUEAST    = 0,(0), 0,(0), 0,(0), 0,(0), 81,(0201)
DSPPAT    = 1,2,3,4,5
```

Finally, suppose this processor is connected to two network circuits, a high-speed circuit (#5) and a low-speed circuit (#2). Suppose further that this processor is known on the high-speed circuit as network address 05-07, and on the low-speed circuit as network address 02-06. Suppose also that one of the other processors on circuit #5 can forward messages to circuits #8 and #9. The .GEN file would include network circuit drivers for both high- and low-speed circuits:

```
    ..., CKTHIGH, CKTLOW, ...
```

and the .PAR file might set up the network tables as follows:

```
    NMBCKT   = 2
    CKTAST   = (0507),CKTDRA, (0206),CKTDRB
    DEFDID   = (0509)
    FWDTBL   = 08,05, 09,05, 0FF
```

## Step-by-Step Procedure for System Generation

To generate a new version of TurboDOS, the following steps may be followed:

1. Bring up a single-user operating system, either CP/M or (preferably) a previous version of TurboDOS. If you are using CP/M, all diskettes will have to be in a format compatible with both CP/M and TurboDOS (e.g., eight-inch, one-sided, single-density, 128-byte sector size).

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

3. Using an editor, create or revise the file OSMASTER.GEN containing the names of the relocatable files to be linked together. In most cases, this will consist of the appropriate STDxxxxx file plus all required device drivers.

4. Using an editor, 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 file. If the target machine has less than 64K of memory installed, don't forget to specify a ";Uxxxx" option on the GEN command.

6. If you need to generate a new O/S loader, create or revise the files OSLOAD.GEN and OSLOAD.PAR, and use the command "GEN OSLOAD.COM" to generate an executable loader file.

6. If you need to generate a new slave O/S for a networking configuration, create or revise the files OSSLAVE.GEN and OSSLAVE.PAR, and use the command "GEN OSSLAVE.SYS" to generate an executable down-load file.

7. To test the newly generated system, log onto your working diskette, eject all other diskettes, and enter the command "OSLOAD". If the new system fails to come up or to function properly, you will have to start over at step 1; there is most likely an error in one of your .GEN or .PAR files.

## SERIAL Command

Each relocatable TurboDOS module which is distributed to an OEM is partially serialized with an origin number only. Each OEM is provided with a SERIAL command processor which must be used to add a unique unit number to the relocatable modules of each copy of TurboDOS issued by that OEM.

The format of the SERIAL command is:

```
SERIAL srcfile destfile ;Unnn options
```

where "srcfile", "destfile" and "options" have exactly the same meanings as in the COPY command, and "nnn" is the unit number expressed as a decimal integer. The SERIAL command works exactly like the COPY command, except that it has the additional function of magnetically serializing .REL files.

The GEN command will not accept partially serialized modules that have not been uniquely serialized by the OEM. Conversely, the SERIAL command will not re-serialize modules which have already been fully serialized.

Example:

```
0A}SERIAL  A:  B:  ;U289 N
A:ASSIGN.COM   copied to B:ASSIGN.COM
.
.
.
A:USER.COM   copied to B:USER.COM
0A}
```

## Step-by-Step Procedure for OEM Re-Distribution

To generate a serialized copy of TurboDOS for re-distribution by an OEM to a dealer or end-user, the following steps must be followed:

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

2. Initialize a new diskette, and label it with the TurboDOS version number, the origin and unit numbers, and the required notice "Copyright (C) 1982 by Software 2000, Inc.".

3. Using the SERIAL command, copy and serialize the following files from your OEM redistribution master to the new diskette:

   single-user without spooling: .REL files for STDLOADR and STDSINGL; .COM files for AUTOLOAD, BACKUP, BOOT, BUFFERS, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MONITOR, PRINT, RENAME, SET, SHOW, TYPE, USER, and VERIFY; and .REL files for all necessary driver modules.

   single-user with spooling: .REL files for STDLOADR, STDSINGL, and STDSPOOL; .COM files for AUTOLOAD, BACKUP, BOOT, BUFFERS, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MONITOR, PRINT, PRINTER, QUEUE, RENAME, SET, SHOW, TYPE, USER, and VERIFY; and .REL files for all necessary driver modules.

   multi-user networking: .REL files for STDLOADR, STDSINGL, STDSPOOL, STDMASTR, and STDSLAVE; .COM files for AUTOLOAD, BACKUP, BATCH, BOOT, BUFFERS, CHANGE, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIFO, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MASTER, MONITOR, PRINT, PRINTER, QUEUE, RECEIVE, RENAME, SEND, SET, SHOW, TYPE, USER, and VERIFY; and .REL files for all necessary driver modules.

   **Important Note:** Be certain that the new diskette does not contain unserialized modules or SERIAL.COM.

4. Using the new serialized diskette, generate an executable loader and operating system, using the system generation procedure described earlier in this section.

5. In addition to the serialized diskette, the dealer or end-user should receive a TurboDOS start-up PROM (if applicable) and copies of the User's Guide and Configuration Guide.

# SYSTEM IMPLEMENTATION

TurboDOS has been designed to run on any Z80-based microcomputer with at least 48K of RAM, a random-access mass storage device, and a full-duplex character-oriented console device (or on an interconnected network of such microcomputers). The process-level and kernel-level modules of TurboDOS do not depend upon the specific peripheral devices to be used. Rather, a set of hardware-dependent device driver modules must be included in each TurboDOS configuration in order to adapt the operating system to a particular hardware environment. Device drivers are typically required for consoles, printers, disk controllers, network interfaces, real-time clock, and communications.

Although Software 2000 Inc. can supply TurboDOS pre-configured for certain specific hardware configurations, most OEMs and many dealers and end-users will want to implement their own hardware-dependent drivers. Driver modules may be readily written by any programmer competant in Z80 assembly-language. This section provides detailed instructions to programmers for implementing such driver modules, and the Appendix includes assembly listings of various sample drivers.

## Assembler Requirements

Drivers 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 MACRO-80 and Digital Research's RMAC assemblers have these characteristics, and are well suited for implementing TurboDOS drivers.

Phoenix Software Associates' (PSA) assembler (formerly TDL and Xitan) is an excellent relocatable Z80 assembler, but it produces object modules in a non-standard format. To alleviate this problem, a conversion utility (RELCVT.COM) is available from Software 2000 Inc. for converting PSA-format object modules to standard Microsoft format. The command

```
RELCVT  filename
```

converts the PSA-format .REL file specified by "filename" into standard Microsoft .REL format. Wherever the characters "." and "%" appear in names in the PSA-format module, they are replaced by the characters "?" and "@" (respectively) in the Microsoft-format module.

## Programming Conventions

Assembly-language examples in this section and in the Appendix are all coded for the PSA assembler. In the examples, the name suffix "#" is used to reference an external name that is defined in another module. The label suffix "::" is used to define a public name that is available for reference in other modules. Some assemblers require that such names be declared in an EXTERN or PUBLIC statement. Program, data, and common segments are introduced with a .LOC statement in the examples. Some assemblers use different statements (e.g., CSEG, DSEG, COMMON) to accomplish the same thing. Also, the symbol "." represents the current location counter value; some assemblers use "$" or "*" instead.

### Dynamic Memory Allocation

The resident portion of TurboDOS resides in the topmost portion of system memory. TurboDOS uses a common memory management module (MEMMGR) to provide dynamic allocation and de-allocation of memory space required for disk buffers, despool requests, file interlocks, DO-file nesting, etc. Dynamic memory segments are allocated downward from the base of the TurboDOS resident area, thereby reducing the space available for the transient program area (TPA). Deallocated segments are concatenated with any neighbors and threaded on a free list. A best-fit algorithm is used to reduce memory fragmentation.

Allocation and de-allocation of memory segments is accomplished in this manner:

```
LXI     H,36        ;get size of requested segment in HL
CALL    ALLOC#      ;allocate segment
ORA     A           ;was segment allocated successfully?
JNZ     ERROR       ;if not, error
PUSH    H           ;else, segment base address in HL
.
.
.
POP     H           ;get address of memory segment in HL
CALL    DEALOC#     ;de-allocate segment
```

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

**Threaded Lists**

All dynamic structures in TurboDOS are maintained as threaded lists with bidirectional linkages. This technique permits a node to be easily added or deleted anywhere in a threaded list without searching. The list head and each list node must contain a two-word linkage (forward pointer and backward pointer).

Manipulation of threaded lists is accomplished in this manner:

```
LSTHED:                         ;list head (initialized to empty)
        .WORD   .               ;forward pointer
        .WORD   .-2             ;backward pointer
;
LSTNOD:                         ;list node
        .WORD   0               ;forward pointer
        .WORD   0               ;backward pointer
        .BLKB   128             ;node body
          .
          .
          .
        LXI     H,LSTHED        ;get list head address in HL
        LXI     D,LSTNOD        ;get new node address in DE
        CALL    LNKEND#         ;link node to end of list
          .
          .
          .
        LXI     H,LSTNOD        ;get node address in HL
        CALL    UNLINK#         ;unlink node from list
          .
          .
          .
        LXI     H,LSTHED        ;get list head address in HL
        LXI     D,LSTNOD        ;get new node address in DE
        CALL    LNKBEG#         ;link node to beginning of list
```

## Dispatching

TurboDOS incorporates an extremely efficient and flexible mechanism for dispatching the Z80 microprocessor among various competing processes. In writing device drivers for TurboDOS, the programmer must take extreme care to use the dispatcher correctly in order to attain maximum performance.

Basically, the dispatcher enables one process to wait for some event (e.g., character available, operation complete) while allowing other processes to utilize the microprocessor. For each such event, the programmer must define a three-word structure called an "event semaphore". A semaphore consists of a count-word followed by a two-word list head. The count-word is used by the dispacher to keep track of the status of the event, while the list head defines a threaded list of processes waiting for the event.

There are two fundamental operations which affect an event semaphore: waiting for the event to occur (WAIT#), and signalling that the event has occurred (SIGNAL#). These are coded in the following manner:

```
EVENT:                               ;event semaphore
            .WORD    0               ;semaphore count
            .WORD    .               ;semaphore list forward pointer
            .WORD    .-2             ;semaphore list backward pointer
              .
              .
              .
            LXI      H,EVENT         ;get event semaphore address
            CALL     WAIT#           ;wait until event occurs
              .
              .
              .
            LXI      H,EVENT         ;get event semaphore address
            CALL     SIGNAL#         ;signal that event has occured
```

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

If the occurrence of an event is signalled but no process is waiting for it, then SIGNAL# simply 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 (e.g., head-settle delay, carriage-return delay) rather than for the occurrence of a specific event. The TurboDOS dispatcher provides a delay facility (DELAY#) which permits other processes to use the microprocessor while one process is waiting for such a time interval to expire. Delay intervals are measured in an implementation-defined unit called a "tick"; in most implementations, ticks occur 50 or 60 times per second. Delays may be coded in the following manner:

```
 .
 .
 .
LXI     H,6         ;get number of ticks to delay
CALL    DELAY#      ;delay for specified interval
 .
 .
 .
```

A delay of zero ticks may be specified to effect a very short delay, or simply to relinquish the processor to other processes on a "courtesy" basis.

For best performance, all driver delays should be accomplished by means of WAIT# (wait for an event to be signalled) or DELAY# (wait for a given interval of time to elapse). Drivers should never be coded to spin in a wait loop.

## Interrupt Service Routines

The TurboDOS dispatching mechanism is especially efficient when used with interrupt-driven peripheral devices. In most situations, the interrupt service routine simply calls SIGNAL# to indicate that the event associated with the interrupt has occurred.

Service routines for low-frequency interrupts (no more than 100 times per second) should exit by means of the standard interrupt service routine exit ISRXIT# in order to provide frequent time-slicing of processes. Service routines for high-frequency interrupts (occurring more than 100 times per second) should simply enable interrupts and return, in order to avoid excessive dispatch overhead.

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 a user's program. TurboDOS provides a standard interrupt stack area (INTSTK#) and stack pointer save location (INTSP#) for this purpose.

A simple interrupt service routine for a low-frequency interrupt could be coded in this manner:

```
DEVISR: SSPD    INTSP#          ;save user's stack pointer
        LXI     SP,INTSTK#      ;set up auxilliary stack
        PUSH    PSW             ;save all registers
        PUSH    B
        PUSH    D
        PUSH    H
        IN      STATUS          ;reset the interrupt condition
        LXI     H,EVENT         ;get event semaphore address
        CALL    SIGNAL#         ;signal that event has occured
        POP     H               ;restore all registers
        POP     D
        POP     B
        POP     PSW
        LSPD    INTSP#          ;restore user's stack pointer
        JMP     ISRXIT#         ;exit through dispacher
```

In more complex interrupt situations, it may be necessary for an interrupt service routine to determine which of several possible events occurred, and to signal one of several alternative semaphores. Sometimes it may be desirable for an interrupt service routine to perform a data buffering function (e.g., to provide keyboard type-ahead).

## Poll Routines

Peripheral devices which are not capable of interrupting the processor must be polled by the device driver. To facilitate this, the TurboDOS dispatcher maintains a threaded list of poll routines, and executes the routines on the list at every dispatch. The function of each poll routine is to check the status of its peripheral device, and to signal the occurrence of an event (e.g., character available, operation complete) when it occurs. The routine LNKPOL# can be called at any time to link a new poll routine onto the poll list.

The only tricky thing about a poll routine is that it must be coded in such a fashion that it will not signal the occurrence a particular event more than once. This can be accomplished in various ways, but a most efficient method is for the poll routine to simply unlink itself from the dispatcher's poll list as soon as it has signalled the occurrence of an event. This can be accomplished in the following manner:

```
EVENT:                          ;event semaphore
        .WORD   0               ;semaphore count
        .WORD   .               ;semaphore list forward pointer
        .WORD   .-2             ;semaphore list backward pointer
          .
          .
          .
        LXI     D,POLNOD        ;get poll routine node address
        CALL    LNKPOL#         ;link poll routine onto poll list
        CALL    POLRTN          ;pre-test peripheral status (optional)
        LXI     H,EVENT         ;get event semaphore address
        CALL    WAIT#           ;wait until event occurs
          .
          .
          .
POLNOD: .WORD   0               ;poll routine node linkage
        .WORD   0
POLRTN: IN      STATUS          ;get peripheral status
        ANI     MASK            ;is input character available?
        RZ                      ;if not, exit
        LXI     H,EVENT         ;else, get event semaphore address
        CALL    SIGNAL#         ;signal that event has occurred
        LXI     H,POLNOD        ;get poll routine node address
        CALL    UNLINK#         ;unlink poll routine from poll list
        RET                     ;done
```

### Re-Entrancy and Mutual Exclusion

All TurboDOS process-level and kernel-level modules permit re-entrant execution by multiple processes. However, most driver-level 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 re-entrant execution.

Using the TurboDOS event semaphore mechanism, such a mutual-exclusion interlock can be implemented very simply in the following manner:

```
MXLOCK:                         ;mutual-exclusion interlock semaphore
         .WORD   1              ;semaphore count (initialized to 1!)
         .WORD   .              ;semaphore list head forward pointer
         .WORD   .-2            ;semaphore list head backward pointer
;
DRIVER:  LXI     H,MXLOCK       ;get interlock semaphore address
         CALL    WAIT#          ;wait if driver is already in use
           .
           .
           .
         LXI     H,MXLOCK       ;get interlock semaphore address
         CALL    SIGNAL#        ;signal driver no longer in use
         RET                    ;done
```

Note that the interlock semaphore count-word must be initialized to 1 (instead of 0) for this scheme to work properly.

## Sample Interrupt-Driven Device Driver

The following is a simple device driver for an interrupt-driven serial input device. It illustrates the coding techniques described previously:

```
MXLOCK:                         ;mutual-exclusion interlock semaphore
         .WORD    1             ;semaphore count (initialized to 1!)
         .WORD    .             ;semaphore list head forward pointer
         .WORD    .-2           ;semaphore list head backward pointer
;
EVENT:                          ;event semaphore
         .WORD    0             ;semaphore count
         .WORD    .             ;semaphore list forward pointer
         .WORD    .-2           ;semaphore list backward pointer
;
CHRSAV:  .BYTE    0             ;input character save location
;
DRIVER:  LXI      H,MXLOCK      ;get interlock semaphore address
         CALL     WAIT#         ;wait if driver is already in use
         EI                     ;ensure that interrupts are enabled
         LXI      H,EVENT       ;get event semaphore
         CALL     WAIT#         ;wait for event to occur
         LDA      CHRSAV        ;get input character
         PUSH     PSW           ;save on stack
         LXI      H,MXLOCK      ;get interlock semaphore address
         CALL     SIGNAL#       ;signal driver no longer in use
         POP      PSW           ;return input character in A-register
         RET                    ;done
;
DEVISR:  SSPD     INTSP#        ;save user's stack pointer
         LXI      SP,INTSTK#    ;set up auxilliary stack
         PUSH     PSW           ;save all registers
         PUSH     B
         PUSH     D
         PUSH     H
         IN       STATUS        ;get peripheral status
         ANI      MASK          ;is input character available?
         JRZ      ..X           ;if not, exit
         IN       DATA          ;else, get input character
         STA      CHRSAV        ;save input character
         LXI      H,EVENT       ;get event semaphore address
         CALL     SIGNAL#       ;signal that event has occured
..X:     POP      H             ;restore all registers
         POP      D
         POP      B
         POP      PSW
         LSPD     INTSP#        ;restore user's stack pointer
         JMP      ISRXIT#       ;exit through dispacher
```

### Sample Polled Device Driver

The following is a simple device driver for a polled serial input device. It illustrates the coding techniques described previously:

```
MXLOCK:                               ;mutual-exclusion interlock semaphore
         .WORD    1                   ;semaphore count (initialized to 1!)
         .WORD    .                   ;semaphore list head forward pointer
         .WORD    .-2                 ;semaphore list head backward pointer
;
EVENT:                                ;event semaphore
         .WORD    0                   ;semaphore count
         .WORD    .                   ;semaphore list forward pointer
         .WORD    .-2                 ;semaphore list backward pointer
;
CHRSAV:  .BYTE    0                   ;input character save location
;
DRIVER:  LXI      H,MXLOCK            ;get interlock semaphore address
         CALL     WAIT#               ;wait if driver is already in use
         LXI      D,POLNOD            ;get poll routine node address
         CALL     LNKPOL#             ;link poll routine onto poll list
         CALL     POLRTN              ;pre-test peripheral status (optional)
         LXI      H,EVENT             ;get event semaphore address
         CALL     WAIT#               ;wait until event occurs
         LDA      CHRSAV              ;get input character
         PUSH     PSW                 ;save on stack
         LXI      H,MXLOCK            ;get interlock semaphore address
         CALL     SIGNAL#             ;signal driver no longer in use
         POP      PSW                 ;return input character in A-register
         RET                          ;done
;
POLNOD:  .WORD    0                   ;poll routine node linkage
         .WORD    0
POLRTN:  IN       STATUS              ;get peripheral status
         ANI      MASK                ;is input character available?
         RZ                           ;if not, exit
         IN       DATA                ;else, get input character
         STA      CHRSAV              ;save input character
         LXI      H,EVENT             ;else, get event semaphore address
         CALL     SIGNAL#             ;signal that event has occurred
         LXI      H,POLNOD            ;get poll routine node address
         CALL     UNLINK#             ;unlink poll routine from poll list
         RET                          ;done
```

### Initialization Segments

In programming hardware-dependent driver modules, it is frequently necessary to include a considerable amount of initialization code which is executed only once (at system start-up) and never needed again. TurboDOS provides a space-saving mechanism whereby such initialization code may be loaded and executed in lower memory (TPA), instead of becoming part of the resident operating system. To use this feature, each initialization segment must be assembled under a special location counter (i.e., common block) named ?INIT? (or .INIT.# if PSA assembler and RELCVT are used):

```
         .LOC     .INIT.#     ;initialization segment follows
;
HDWNIT:: XRA      A           ;start of initialization code
          .
          .
          .
         RET                  ;end of initialization code
;
```

### Page-Oriented Segments

Sometimes, a programmer must force a segment of code or data to begin on a 256-byte page boundary. Examples are interrupt vectors for Z80 IM2, and the simulated CP/M BIOS branch table. To do this, a page-oriented segment must be assembled under a special location counter (i.e., common block) named ?PAGE? (or .PAGE.# if PSA assembler and RELCVT are used). If several modules utilize the ?PAGE? location counter, then a separate page is allocated for each such module.

### Inter-Process Messages

For passing messages from one process to another, TurboDOS makes use of a five-word structure called a "message node". A message node consists of a three-word event semaphore followed by a two-word message list head. Subroutines are provided for sending messages to a message node (SNDMSG#), and receiving messages from a message node (RCVMSG#). Typically, the sending process allocates a segment of dynamic memory in which to build the message, and the receiving process deallocates the segment after making use of the message:

```
MSGNOD:                                 ;message node
          .WORD      0                  ;semaphore count
          .WORD      .                  ;semaphore list forward pointer
          .WORD      .-2                ;semaphore list backward pointer
          .WORD      .                  ;message list forward pointer
          .WORD      .-2                ;message list backward pointer

SEND:     LXI        H,12               ;get size of message to send
          CALL       ALLOC#             ;allocate message packet
          PUSH       H                  ;save message packet address
          .
          .                             ;build message in packet
          .
          POP        D                  ;get address of message packet in DE
          LXI        H,MSGNOD           ;get address of message node in HL
          CALL       SNDMSG#            ;send inter-process message
          .
          .
          .

REC:      LXI        H,MSGNOD           ;get address of message node in HL
          CALL       RCVMSG#            ;receive inter-process message
          PUSH       H                  ;save message packet address
          .
          .                             ;read message in packet
          .
          POP        H                  ;get address of message packet in HL
          CALL       DEALOC#            ;deallocate message packet
          .
          .
          .
```

### Console Subroutines

The following illustrates the use of various console-related subroutines which may be called from within driver modules:

```
        CALL    CONST#      ;returns console status in A-reg
        ORA     A           ;is console input available?
        RZ                  ;if not, exit
        CALL    CONIN#      ;returns console input in A-reg
        CALL    UPRCAS#     ;convert lower-case letters to upper-case
        MOV     C,A         ;move character to C-reg
        CALL    CONOUT#     ;displays character passed in C-reg


        CALL    DMS#        ;display following message
..MSG:  .ASCIS  "Message"   ;last byte of message has sign-bit set
        LXI     H,..MSG     ;HL -> sign-bit-terminated message
        CALL    DMSHL#      ;display message pointed-to by HL


        LXI     H,31416     ;HL has any 16-bit value
        CALL    DECOUT#     ;displays HL value in decimal
```

## Creating a Resident Process

In some circumstances, it may be desirable to activate a resident process which runs in the background concurrent with other system activities. The create-process subroutine CRPROC# may be called to create such a process at system initialization time as illustrated below:

```
          .LOC    .INIT.#        ;initialization segment follows
;
HDWNIT::                         ;hardware-dependent initialization
          LXI     D,BACKGR       ;get process entrypoint address
          CALL    CRPROC#        ;create process
            .
            .
            .


          .LOC    .PROG.#        ;program segment follows
;
BACKGR:                          ;background process starts here
          LXI     D,60*60        ;number of ticks in one minute
          MVI     C,125          ;number of "delay process" function
          CALL    OSNTRY#        ;delay for one minute
          LXI     D,..FCB        ;get FCB address
          MVI     C,15           ;number of "open file" function
          CALL    OSNTRY#        ;open the file
          ORA     A              ;was file opened successfully?
          JRNZ    BACKGR         ;if not, try again in a minute
            .                    ;otherwise, ...
            .
            .
..FCB:    .BYTE   0              ;file control block
          .ASCII  "FILENAME"
          .ASCII  "TYP"
          .BYTE   [24]0
```

The CRPROC# routine automatically assigns a TurboDOS work area whose address appears to the new process in the X-register, and a 64-word stack area whose address appears in the SP-register. If the process requires a re-entrant work area (usually dynamically allocated), its address should be passed to CRPROC# in the HL-register and will appear to the new process in the Y-register.

Note that the resident process must make all its operating system requests in exactly the same fashion as a transient program would, except that the operating system entrypoint OSNTRY# is used instead of location 0005H. This is very important. A resident process must not make direct calls to kernel-level subroutines such as WAIT#, SIGNAL#, DELAY#, SNDMSG#, RCVMSG#, ALLOC#, DEALOC#, DMS#, etc. Also, a resident process is not attached to a console, so any console input/output operations it issues will be ignored.

A resident process must preserve index register X, but may use other registers as desired.

## Driver Interface Specifications

The interface specifications for various kinds of device drivers are described below. Drivers may be packaged into as many or few separate modules as desired by the programmer. In general, it is easier to reconfigure TurboDOS for a wide variety of peripheral devices if the driver for each device is packaged as a separate module.

TurboDOS may be configured with multiple disk, console, printer and network drivers. The disk driver entrypoint table refers to disk driver entrypoints DSKDRA#, DSKDRB#, DSKDRC#, etc. Each disk driver should be coded with a public entrypoint DSKDR@:: (or DSKDR%:: if PSA assembler and RELCVT are used). The GEN command automatically maps successive definitions of such names by replacing the trailing @ by A, B, C, etc. The same technique should be used for console, printer, and network drivers.

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 ?UND?#. The TurboDOS common subroutine module COMSUB contains the following stub routine:

```
?UND?::  NOP                         ;single- or double-length load
         NOP                         ;of undefined returns zero
         XRA      A                  ;call of undefined returns A=0
         RET                         ;done
```

Thus, it is always safe to load or call an external name, whether or not it is defined.

Driver routines must preserve the stack and the index registers X and Y, but may use other registers as desired.

### Initialization

All necessary hardware initialization and interrupt vector setup should be performed by an initialization routine that begins with the public entry name HDWNIT::. This routine is called by TurboDOS at system start-up with interrupts disabled. The hardware initialization procedure must not enable interrupts or make calls to WAIT# or DELAY#. In most cases, the HDWNIT:: routine should contain a series of calls to individual driver initialization subroutines. All initialization code which is not needed again should be assembled under the location counter (i.e., common block) ?INIT?.

## Console Drivers

Each console driver routine should begin with the public entry name CONDR@::, and should perform a console operation in accordance with the operation code (0, 1, 2, 8 or 9) passed by TurboDOS in the E-register. A console number is passed in the B-register (obtained from the least-significant nibble of the console assignment table entry CONAST#).

If E=0, the driver must determine if a console input character is available. It must return with A=-1 if a character is available, or with A=0 if no character is available. If a character is available, the driver must return it in the C-register, but must not "consume" the character. (This look-ahead capability is used by TurboDOS to detect attention requests.)

If E=1, the driver must obtain a console input character (waiting for one if necessary), and return it in the A-register.

If E=2, the driver must output to the console the character passed by TurboDOS in the C-register.

If E=8, the driver should prepare to display a TurboDOS error message; if E=9, the driver should revert to normal display. Error message displays issued b· TurboDOS are always preceded by an E=8 call and followed by an E=9 call. This gives the console driver the opportunity to take special action for system error messages (e.g., 25th line, reverse video). For simple console devices, the driver should perform a carriage-return and line-feed in response to E=8 and E=9 calls.

## Printer Drivers

Each printer driver routine should begin with the public entry name LSTDR@::, and should perform a printer operation in accordance with the operation code (2 or 7) passed by TurboDOS in the E-register. A printer number is passed in the B-register (obtained from the least-significant nibble of the printer assignment table entry LSTAST#).

If E=2, the driver must output to the printer the character passed by TurboDOS in the C-register.

If E=7, the driver should take any appropriate end-of-print-job action (e.g., re-align forms, drop ribbon, home print head).

## Network Circuit Drivers

Each network circuit driver should begin with the public entry name CKTDR@::, and should send or receive a network message, according to the operation code (0 or 1) passed by TurboDOS in the C-register.

If C=0, the driver must receive a network message into the message buffer whose address is passed by TurboDOS in the DE-registers. If a message is received successfully, the driver must return with A=0. On the other hand, a malfunction of any remote processor is detected, the driver must return with A=-1 and must return the network address of the crashed processor in the DE-registers.

If C=1, the driver must send the network message from the message buffer whose address is passed by TurboDOS in the DE-registers. If a message is sent successfully, the driver must return with A=0. On the other hand, if the message could not be sent because of a malfunction of the destination processor, the driver must return with A=-1 and must return the network address of the crashed processor in the DE-registers.

The format of a network message buffer is:

```
Linkage: .WORD   .         ;buffer linkage
         .WORD   .

Header:  .BYTE   MSGLEN    ;message length (excludes linkage)
         .WORD   MSGDID    ;network address of message destination
         .BYTE   MSGPID    ;process ID
         .WORD   MSGSID    ;network address of message source
         .WORD   MSGOID    ;network address of originator
         .BYTE   MSGOPR    ;process ID of originator
         .BYTE   MSGLVL    ;forwarding level number
         .BYTE   MSGFCD    ;message format code

Body:    .BLKB   7         ;registers: A,C,B,E,D,L,H
         .BLKB   38        ;FCB data and related info (optional)
         .BLKB   128       ;record data (optional)
```

The first four bytes of a message buffer contains a linkage used by TurboDOS, and should not be sent or received by the driver. The eleven-byte header and variable-length body should be passed over the network. The driver should only need to look at the message length and destination ID in order to do its job. On a receive request (C=0), TurboDOS presets the message length byte to the maximum allowable message length, and expects that byte to contain the actual message length upon return. On a send request (C=1), TurboDOS presets the message length byte to the actual length of the message to be sent (header plus body).

In simple master-slave network situations, it is often desirable for the circuit driver in the master processor to periodically "poll" the slave processors on its circuit in order to detect any slave malfunctions in a timely fashion and to effect recovery. If the circuit driver reports that a slave has crashed (by returning A=-1 and DE=net-address), then the circuit driver must not accept any further messages from that slave until TurboDOS has completed its recovery for that slave. TurboDOS signals the driver that such recovery is complete by sending a dummy message destined for that slave 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 initiate a new down-load of the slave operating system.

For a slave processor to have its operating system downloaded over the network, it must send a special download-request message consisting of a standard eleven-byte header (with MSGPID, MSGOID and MSGFCD set to zero) followed by a one-byte body containing a "download suffix" character. The master processor specified by MSGDID will return a reply message whose body contains the first 128-byte record of the file OSSLAVEx.SYS (where "x" is the specified download suffix). The slave should continue to send download-request messages and receive successive records of the .SYS file until it receives a "short" reply message signifying end-of-file. The first word of the OSSLAVEx.SYS file specifies the base address to which the system should be moved, and the second word specifies the total length of the system. The single byte passed as the body of the "short" reply message identifies the system disk, and should be passed to the system in the A-register.

## Disk Drivers

Each disk driver routine should begin with the public entry name DSKDR@::, and should perform a physical disk operation as specified by the physical disk request packet whose address is passed by TurboDOS in the X-register. The format of the physical disk request packet is:

```
X+0:    .BYTE   OPCODE    ;disk operation code
X+1:    .BYTE   DRIVE     ;drive number on controller (base 0)
X+2:    .WORD   TRACK     ;physical track number (base 0)
X+4:    .WORD   SECTOR    ;physical sector number (base 0)
X+6:    .WORD   SECCNT    ;number of sectors to read or write
X+8:    .WORD   BYTCNT    ;number of bytes to read or write
X+10:   .WORD   DMAADR    ;DMA address for read or write
X+12:   .WORD   DSTADR    ;disk specification table address
;
;copy of disk specification table follows
;
X+14:   .BYTE   BLKSIZ    ;block size (3=1K, 4=2K,..., 7=16K)
X+15:   .WORD   NMBLKS    ;number of blocks, total
X+17:   .BYTE   NMBDIR    ;number of directory blocks
X+18:   .BYTE   SECSIZ    ;sector size (0=128, 1=256, 2=512,..., 7=16K)
X+19:   .WORD   SECTRK    ;sectors per track
X+21:   .WORD   TRKDSK    ;total tracks on disk
X+23:   .WORD   RESTRK    ;reserved tracks on disk
```

If OPCODE=0, then the driver must read SECCNT physical sectors (or BYTCNT bytes) into DMAADR, starting at TRACK and SECTOR on DRIVE. Return with A=-1 if an unrecoverable error occurs, otherwise return with A=0. Although TurboDOS may request many consecutive sectors to be read, it will never request an operation which extends past the end of the specified track.

If OPCODE=1, then the driver must write SECCNT physical sectors (or BYTCNT bytes) from DMAADR, starting at TRACK and SECTOR on DRIVE. Return with A=-1 if an unrecoverable error occurs, otherwise return with A=0. Although TurboDOS may request many consecutive sectors to be written, it will never request an operation which extends past the end of the specified track.

If OPCODE=2, then the driver must determine the type of disk mounted in the specified drive, and must return in DSTADR the address of an 11-byte disk specification table structured as follows:

```
DST:    .BYTE   BLKSIZ   ;block size (3=1K, 4=2K,..., 7=16K)
        .WORD   NMBLKS   ;number of blocks, total
        .BYTE   NMBDIR   ;number of directory blocks
        .BYTE   SECSIZ   ;sector size (0=128, 1=256, 2=512,..., 7=16K)
        .WORD   SECTRK   ;sectors per track
        .WORD   TRKDSK   ;total tracks on disk
        .WORD   RESTRK   ;reserved tracks on disk
```

On return, TurboDOS moves a copy of the disk specification table into X+14 through X+24, where it is available for subsequent read and write operations on that drive. If the drive is not ready or the type is unrecognizable, the driver must return A=0, otherwise it must return A=-1.

If OPCODE=3, then the driver must determine whether or not the specified drive is ready. Return A=-1 if the drive is ready, otherwise return A=0.

If OPCODE=4, then the driver must format (i.e., initialize) the specified TRACK on DRIVE. Hardware-dependent formatting information will be provided at DMAADR. Return with A=-1 if an unrecoverable error occurs, otherwise return with A=0.

### Real-Time Clock Driver

The real-time clock driver normally consists of an interrupt service routine which responds to interrupts from a periodic interrupt source (preferably 50 to 60 times per second). The interrupt service routine should call DLYTIC# once per system tick to synchronize process delay requests. It should also call RTCSEC# once per second (i.e., every 50 or 60 ticks) to update the system time and date. Finally, it should exit through ISRXIT# to provide a periodic system time-slice.

Excluding necessary initialization code, a typical real-time clock driver might look like this:

```
RTCCNT: .BYTE   1               ;divide-by-60 counter
;
RTCISR: SSPD    INTSP#          ;save user's stack pointer
        LXI     SP,INTSTK#      ;set up auxilliary stack
        PUSH    PSW             ;save all registers
        PUSH    B
        PUSH    D
        PUSH    H
        IN      STATUS          ;reset the interrupt condition
        CALL    DLYTIC#         ;signal one tick elapsed time
        LXI     H,RTCCNT        ;get divide-by-60 counter
        DCR     M               ;decrement counter
        JRNZ    ..X             ;not 60 ticks yet, exit
        MVI     M,60            ;else, reset counter to 60 ticks
        CALL    RTCSEC#         ;signal one second elapsed time
..X:    POP     H               ;restore all registers
        POP     D
        POP     B
        POP     PSW
        LSPD    INTSP#          ;restore user's stack pointer
        JMP     ISRXIT#         ;exit through dispacher
```

If it is possible to determine the date and/or time-of-day at cold-start (e.g., by means of a battery-powered clock board), then the driver may initialize the following public symbols in RTCMGR:

```
SECS::  .BYTE   0               ;0...59
MINS::  .BYTE   0               ;0...59
HOURS:: .BYTE   0               ;0...23
JDATE:: .WORD   8001H           ;Julian date, based 31 Dec 47
```

## Comm Channel Drivers

The comm channel driver supports the TurboDOS communications extensions (functions 87...93), and is not required if these functions are not used. The comm channel driver routine should begin with the public entry name COMDRV::, and should perform a comm channel operation in accordance with the operation code passed by TurboDOS in the E-register. A channel number is passed in the B-register.

I˙ E=0, the driver must determine if an input character is available on the specified channel. It must return with A=-1 if a character is available, or with A=0 if no character is available.

If E=1, the driver must obtain an input character from the specified channel (waiting for one if necessary), and return it in the A-register.

If E=2, the driver must output to the specified channel the character passed by TurboDOS in the C-register.

If E=3, the driver must set the baud rate of the specified channel according to the baud rate code passed by TurboDOS in the C-register. (See function 90 in the User's Guide for definition of the codes.)

If E=4, the driver must obtain the current baud rate code for the specified channel, and return it in the A-register.

If E=5, the driver must set the modem controls of the specified channel according to the modem control vector passed by TurboDOS in the C-register. (See function 92 in the User's Guide for definition of the vector.)

If E=6, the driver must obtain the current modem status vector for the specified channel, and return it in the A-register. (See function 93 in the User's Guide for definition of the vector.)

## Bootstrap ROM

Implementation of a TurboDOS bootstrap ROM involves linking the standard bootstrap module OSBOOT with a hardware-dependent driver OSBDRV. This should be accomplished with the GEN command, using the ";Lxxxx" option to establish the desired ROM base address. Since the OSBOOT module requires only 0.4K, the completed bootstrap can fit in a 1K ROM (e.g., 2708) if the driver is kept simple enough. The driver module OSBDRV must define five public entry names: INIT::, SELECT::, READ::, XFER::, and RAM::.

INIT:: is called at the beginning of the bootstrap process, and performs any required hardware initialization (e.g., of the disk controller). It must return with the load base address in the HL-registers. The load base address determines the RAM where loading of the file OSLOAD.COM will begin. It should normally be 0100H, but may have to be a higher address if low RAM cannot be written while the ROM is enabled.

SELECT:: selects the disk drive according to the drive number 0...15 passed in the A-register. If the selected drive is not ready or non-existent, then this routine must return A=0. Otherwise, it must return A=-1, and must return the address of an appropriate disk specification table in the HL-registers. The disk specification table is an 11-byte table whose format is the same as described earlier for the normal disk driver.

READ:: reads one physical sector from the last selected drive into RAM. On entry, the physical track is passed in the BC-registers, the physical sector is passed in the DE-registers, and the starting RAM address is passed in the HL-registers. The routine must return with A=0 if the operation was successful, or with A=-1 if an unrecoverable error occurred.

XFER:: is executed at the end of the bootstrap process, and transfers control to the loader program OSLOAD.COM which has been loaded into RAM. In most cases, this involves simply setting location 0080H to zero (to simulate a null command tail), and jumping to 0100H. However, i` INIT returned a loader base other than 0100H, then XFER should move the loader program down to 0100H prior to execution.

RAM:: defines the beginning of a 64-byte area of RAM that OSBOOT can use as working storage. Obviously, it should not be located in the area in which OSLOAD.COM will be loaded!

# APPENDIX — SAMPLE DRIVER LISTINGS

```
                        ;
                        ; *****************************************************
                        ; *                                                   *
                        ; *  The following are common symbolic equates used in  *
                        ; *  the various drivers which follow.  They are refer-  *
                        ; *  enced by a .INSERT DREQUATE in each driver module.  *
                        ; *                                                   *
                        ; *****************************************************
                        ;
                        ; COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.
                        ;
                        ; AUTHORS: RONALD E. RAIKES
                        ;          MICHAEL D. BUSCH
                        ;
                        ; VERSION: EXAMPLE
                        ;
                        .IDENT  EQUATE          ;MODULE ID
                        ;
                        ; 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                   OPSYS   == 0005H           ;OPERATING SYSTEM ENTRYPOINT
  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 LENG

  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
                         ;
                         .END
```

MPENIT - TURBODOS OPERATING SYSTEM MEMORY PARITY INITIALIZATION
COPYRIGHT (C) 1982, SOFTWARE 2000, INC.

```
                         ;
                         ; ********************************************************
                         ; *                                                      *
                         ; *  This module initializes the parity of the RAM on    *
                         ; *  an IMS 64K RAM board or an IMS 740 slave proces-    *
                         ; *  sor board.  The procedure is to do a 64K block      *
                         ; *  move of all memory which causes parity generation   *
                         ; *  to take place in all memory locations.              *
                         ; *                                                      *
                         ; ********************************************************
                         ;
                         ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                         ;
                         ; AUTHORS: RONALD E. RAIKES
                         ;          MICHAEL D. BUSCH
                         ;
                         ; VERSION: EXAMPLE
                         ;
                         .IDENT  MPENIT          ;MODULE ID
                         ;
                         .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                         ;
   0000:04                      .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                         ;
   0000:04 21 0000       MPENIT::LXI     H,0     ;INITIALIZE MEMORY PARITY
   0003:04 11 0000              LXI     D,0
   0006:04 01 0000              LXI     B,0
   0009:04 EDBO                 LDIR
   000B:04 AF                   XRA     A       ;CLEAR START-UP PARITY ERROR
   000C:04 D300                 OUT     0
   000E:04 C9                   RET             ;DONE
                         ;
                         .END
```

```
NITIMS - TURBODOS OPERATING SYSTEM HARDWARE INITIALIZATION (IMS)
COPYRIGHT (C) 1982, SOFTWARE 2000, INC.

                         ;
                         ; ********************************************************
                         ; *                                                      *
                         ; *  This is the hardware initialization routine for     *
                         ; *  an IMS 8000 master processor.  It consists only     *
                         ; *  of calls to the initialization entrypoints of       *
                         ; *  other driver modules.                               *
                         ; *                                                      *
                         ; ********************************************************
                         ;
                         ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                         ;
                         ; AUTHORS: RONALD E. RAIKES
                         ;          MICHAEL D. BUSCH
                         ;
                         ; VERSION: EXAMPLE
                         ;
                         .IDENT  NITIMS          ;MODULE ID
                         ;
                         .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                         ;
   0000:04                       .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                         ;
   0000:04 CD 0000:05    HDWNIT::CALL    MPENIT# ;INITIALIZE MEMORY PARITY
   0003:04 CD 0000:06            CALL    SPINIT# ;INITIALIZE SERIAL/PARALLEL I/O
   0006:04 CD 0000:07            CALL    RTCNIT# ;INITIALIZE REAL TIME CLOCK
   0009:04 CD 0000:08            CALL    DSKINA# ;INITIALIZE DISK DRIVER A
   000C:04 CD 0000:09            CALL    DSKINB# ;INITIALIZE DISK DRIVER B
   000F:04 CD 0000:0A            CALL    DSKINC# ;INITIALIZE DISK DRIVER C
   0012:04 CD 0000:0B            CALL    DSKIND# ;INITIALIZE DISK DRIVER D
   0015:04 CD 0000:0C            CALL    CKTINA# ;INITIALIZE CIRCUIT DRIVER A
   0018:04 CD 0000:0D            CALL    CKTINB# ;INITIALIZE CIRCUIT DRIVER B
   001B:04 CD 0000:0E            CALL    CKTINC# ;INITIALIZE CIRCUIT DRIVER C
   001E:04 C3 0000:0F            JMP     CKTIND# ;INITIALIZE CIRCUIT DRIVER D
                         ;
                         .END
```

```
                        ;
                        ; ******************************************************
                        ; *                                                    *
                        ; *  This is a disk driver for the IMS 401 eight-inch  *
                        ; *  floppy disk controller board.  This board makes   *
                        ; *  use of a NEC uPD-765 floppy disk controller chip  *
                        ; *  coupled with an Intel 8257 DMA controller chip.   *
                        ; *                                                    *
                        ; *  This driver is fairly long and complex because:   *
                        ; *                                                    *
                        ; *  (1) The NEC uPD-765 is not as easy to get along   *
                        ; *      with as other chips (such as the Western      *
                        ; *      Digital 1791 or 1793).                        *
                        ; *                                                    *
                        ; *  (2) The driver accomodates a wide variety of      *
                        ; *      disk formats, including both TurboDOS non-    *
                        ; *      interleaved and CP/M interleaved formats.     *
                        ; *      The various formats are defined in a separate *
                        ; *      module DST8F.                                 *
                        ; *                                                    *
                        ; ******************************************************
                        ;
                        ;
                        ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                        ;
                        ; AUTHORS: RONALD E. RAIKES
                        ;          MICHAEL D. BUSCH
                        ;
                        ; VERSION: EXAMPLE
                        ;
                        .IDENT  DSK401          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
  0082                  CH1DMA  = 82H           ;CHANNEL 1 DMA REGISTER (FDC)
  0083                  CH1TC   = 83H           ;CHANNEL 1 TERMINAL COUNT (FDC)
  0088                  DMACTL  = 88H           ;DMA COMMAND AND STATUS REGISTERS
  008A                  DSKSEL  = 8AH           ;DISK SELECT PORT
  008C                  DSKCTL  = 8CH           ;STATUS AND INT MASK (BOARD)
  008E                  FDCST   = 8EH           ;DISK CONTROLLER STATUS (uPD-765)
  008F                  FDCDAT  = 8FH           ;DISK CONTROLLER DATA (uPD-765)
                        ;
  0042                  CH1ENA  = 42H           ;DMA CHANNEL 1 ENABLE COMMAND
  0000                  DMAVFY  = 00H           ;DMA VERIFY COMMAND
  0040                  DMARD   = 40H           ;DMA READ COMMAND
  0080                  DMAWR   = 80H           ;DMA WRITE COMMAND
                        ;
  0003                  FDCSFY  = 03H           ;FDC SPECIFY COMMAND
  0004                  FDCSDS  = 04H           ;FDC SENSE DRIVE STATUS COMMAND
  0007                  FDCRCL  = 07H           ;FDC RECALIBRATE COMMAND
  0008                  FDCSIS  = 08H           ;FDC SENSE INTERRUPT STATUS COMMAN
  000A                  FDCRID  = 0AH           ;FDC READ ID COMMAND
  000D                  FDCFMT  = 0DH           ;FDC FORMAT TRACK COMMAND
  000F                  FDCSK   = 0FH           ;FDC SEEK COMMAND
  0005                  FDCWR   = 05H           ;FDC WRITE COMMAND
  0006                    FDCRD   = 06H              ;FDC READ COMMAND
                          ;
  0000                    DSKENI  = 0                ;DISK CONTROLLER ENABLE INTERRUPTS
  0007                    DSKDLC  = 7                ;DISK CONTROLLER DELAY COMPLETE
                          ;
  0006                    FDCMFM  = 6                ;FDC DOUBLE-DENSITY BIT
  0007                    FDCMT   = 7                ;FDC MULTI-TRACK BIT
                          ;
  0004                    FDCBSY  = 4                ;FDC BUSY STATUS
  0005                    FDCSE   = 5                ;FDC SEEK END
  0006                    FDCOUT  = 6                ;FDC OUTPUT MODE
  0007                    FDCRDY  = 7                ;FDC READY FOR DATA
                          ;
  0003                    SRT8Q   = 3                ;8 INCH FDD STEP RATE (3 MS-QUME)
  0008                    SRT8S   = 8                ;8 INCH FDD STEP RATE (8 MS-SHUG)
                          ;
  0024                    HDLT    = 18*2             ;FDD HEAD LOAD TIME (36 MS)
  0001                    HDUT    = 1                ;FDD HEAD UNLOAD TIME (16 MS)
                          ;
  0003                    STONR   = 3                ;STATUS REGISTER 0 NOT READY
  0004                    STOEC   = 4                ;STATUS REGISTER 0 EQUIP CHECK
  0005                    STOSE   = 5                ;STATUS REGISTER 0 SEEK END
                          ;
  0000                    ST1MA   = 0                ;STATUS REGISTER 1 MISSING ADDR MK
  0001                    ST1NW   = 1                ;STATUS REGISTER 1 NOT WRITABLE
  0002                    ST1ND   = 2                ;STATUS REGISTER 1 NO DATA
  0004                    ST1OR   = 4                ;STATUS REGISTER 1 OVER RUN
  0005                    ST1DE   = 5                ;STATUS REGISTER 1 DATA ERROR
                          ;
  0003                    ST3TS   = 3                ;STATUS REGISTER 3 TWO-SIDED
  0004                    ST3TO   = 4                ;STATUS REGISTER 3 TRACK 0
  0005                    ST3RDY  = 5                ;STATUS REGISTER 3 READY
  0006                    ST3WP   = 6                ;STATUS REGISTER 3 WRITE PROTECTED
                          ;
  000A                    MAXTRY  = 10               ;MAX DISK TRY COUNT
                          ;
  0002                    TSD     = 2                ;TWO-SIDED DISK BIT (TYPE CODE)
  0003                    DDD     = 3                ;DOUBLE DENSITY DISK BIT (TYPE COD

  0004                    MINI    = 4                ;MINI-FLOPPY DISK BIT (TYPE CODE)
                          ;
  0000"                           .LOC    .DATA.#    ;LOCATE IN DATA AREA
                          ;
  0000"   08              SRT401::.BYTE   SRT8S      ;STEP RATE
  0001"   00              TRYCNT: .BYTE   0          ;TRY COUNT
  0002"   00              CALTBL: .BYTE   0          ;DRIVE CALIBRATED TABLE
  0003"   00              FLAGS:  .BYTE   0          ;FLAGS
  0004"   00              IORWC:  .BYTE   0          ;I/O READ/WRITE COMMAND
  0005"   00              IODMAC: .BYTE   0          ;I/O DMA COMMAND
  0006"   0000            RETSP:  .WORD   0          ;ERROR RETURN STACK POINTER
  0008"   00              RIDDSK: .BYTE   0          ;READ ID DISK
  0009"   00              CURSEC: .BYTE   0          ;CURRENT SECTOR NUMBER
  000A"   0000            CURADR: .WORD   0          ;CURRENT DMA ADDRESS
  000C"   00              CURSC:  .BYTE   0          ;CURRENT SECTOR COUNT
```

```
000D"   00              IOERR:  .BYTE   0       ;I/O ERROR STATUS
                        ;
000E"                   DMXSPH:                 ;MUTUAL EXCLUSION SEMAPHORE
000E"   0001                    .WORD   1       ;SEMAPHORE COUNT
0010"   0010"           ..DMXH: .WORD   ..DMXH  ;SEMAPHORE P/D HEAD
0012"   0010"                   .WORD   ..DMXH
                        ;
0014"                   DWTSPH:                 ;DISK WAIT SEMAPHORE
0014"   0000                    .WORD   0       ;SEMAPHORE COUNT
0016"   0016"           ..DWTH: .WORD   ..DWTH  ;SEMAPHORE P/D HEAD
0018"   0016"                   .WORD   ..DWTH
                        ;
001A"                   IDINFO:                 ;SECTOR ID INFO LIST
001A"   00              CYL:    .BYTE   0       ;DISK CYLINDER NUMBER
001B"   00              HEAD:   .BYTE   0       ;DISK HEAD NUMBER
001C"   00              REC:    .BYTE   0       ;DISK RECORD NUMBER
001D"   00              SIZE:   .BYTE   0       ;DISK SECTOR SIZE
001E"   00              EOT:    .BYTE   0       ;END OF TRACK SECTOR NUMBER
001F"   00              GPL:    .BYTE   0       ;DISK GAP 3 SIZE
0020"   00              DTL:    .BYTE   0       ;DISK SECTOR SIZE WHEN SIZE=0
                        ;
0021"                   RESULT:                 ;RESULT PHASE LIST
0021"   00              ST0:    .BYTE   0       ;STATUS REGISTER 0
0022"   00              ST1:    .BYTE   0       ;STATUS REGISTER 1
0023"   00              ST2:    .BYTE   0       ;STATUS REGISTER 2
0024"   00              RCYL:   .BYTE   0       ;DISK CYLINDER NUMBER
0025"   00              RHEAD:  .BYTE   0       ;DISK HEAD NUMBER
0026"   00              RREC:   .BYTE   0       ;DISK RECORD NUMBER
0027"   00              RSIZE:  .BYTE   0       ;DISK SECTOR SIZE
0028"   00              MAINST: .BYTE   0       ;MAIN STATUS REGISTER
0029"   00              ST3:    .BYTE   0       ;STATUS REGISTER 3
                        ;
0000:04                         .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                        ;
0000:04 DB8E            DSKIN$::IN      FDCST   ;GET FDC STATUS
0002:04 3C                      INR     A       ;CONTROLLER PRESENT?
0003:04 C8                      RZ              ;IF NOT, DONE
0004:04 3EC3                    MVI     A,JMP   ;ELSE, INITIALIZE INTERRUPT VECTOR
0006:04 32 0028                 STA     5*8     ;(VECTORED INTERRUPT-5)
0009:04 21 0303'                LXI     H,DSKISR
000C:04 22 0029                 SHLD    (5*8)+1
000F:04 AF                      XRA     A
0010:04 D388                    OUT     DMACTL  ;DISABLE DMA CONTROLLER
0012:04 3E03                    MVI     A,FDCSFY  ;GET FDC SPECIFY COMMAND
0014:04 CD 038E'                CALL    CMDRDY  ;OUTPUT COMMAND TO FDC
0017:04 3A 0000"                LDA     SRT401  ;GET STEP RATE
001A:04 ED44                    NEG             ;CALC FDC STEP RATE VALUE
001C:04 C610                    ADI     16
001E:04 87                      ADD     A
001F:04 87                      ADD     A
0020:04 87                      ADD     A
0021:04 87                      ADD     A
0022:04 F601                    ORI     HDUT    ;COMBINE STEP RATE WITH HEAD UNLOA
0024:04 CD 0394'                CALL    DATOUT  ;OUTPUT IT TO FDC
```

```
0027:04 3E24                MVI     A,HDLT  ;GET HEAD LOAD TIME/NON-DMA BIT
0029:04 CD 0394'            CALL    DATOUT  ;OUTPUT IT TO FDC
002C:04 3E01                MVI     A,1<DSKENI
002E:04 D38C                OUT     DSKCTL  ;ENABLE CONTROLLER INTERRUPTS
0030:04 C9                  RET             ;DONE
                    ;
0000'                       .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                    ;
0000'   21 000E"    DSKDR$::LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
0003'   CD 0000:05          CALL    WAIT#   ;DISPATCH IF NECESSARY
0006'   CD 0012'            CALL    ..DD    ;CALL DISK DRIVER
0009'   F5                  PUSH    PSW     ;SAVE RETURN CODE
000A'   21 000E"            LXI     H,DMXSPH  ;GET MUTUAL EXCLUSION SEMAPHORE
000D'   CD 0000:06          CALL    SIGNAL# ;SIGNAL PROCESS AS READY
0010'   F1                  POP     PSW     ;RESTORE RETURN CODE
0011'   C9                  RET             ;DONE
                    ;
0012'   ED73 0006"  ..DD:   SSPD    RETSP   ;SAVE ERROR RETURN STACK POINTER
0016'   DD7E00              MOV     A,PDRFCN(X)  ;GET PD REQ FUNCTION NUMBER
0019'   B7                  ORA     A       ;PD REQ FUNCTION NUMBER=0?
001A'   280F                JRZ     RDDSK   ;IF SO, CONTINUE
001C'   3D                  DCR     A       ;PD REQ FUNCTION NUMBER=1?
001D'   281E                JRZ     WRDSK   ;IF SO, CONTINUE
001F'   3D                  DCR     A       ;PD REQ FUNCTION NUMBER=2?
0020'   CA 0206'            JZ      RETDST  ;IF SO, CONTINUE
0023'   3D                  DCR     A       ;PD REQ FUNCTION NUMBER=3?
0024'   CA 027F'            JZ      RETRDY  ;IF SO, CONTINUE
0027'   3D                  DCR     A       ;PD REQ FUNCTION NUMBER=4?
0028'   282E                JRZ     FMTDSK  ;IF SO, CONTINUE
002A'   C9                  RET             ;ELSE, DONE
                    ;
002B'   3E0A        RDDSK:  MVI     A,MAXTRY  ;GET MAX TRY COUNT
002D'   32 0001"            STA     TRYCNT  ;SET TRY COUNT
0030'   3E06        ..RD:   MVI     A,FDCRD ;GET FDC READ COMMAND
0032'   0E40                MVI     C,DMARD ;GET DMA READ COMMAND
0034'   CD 00C4'            CALL    DSKCOM  ;CALL COMMON CODE
0037'   C8                  RZ              ;NO ERRORS, RET A=0
0038'   CD 011B'            CALL    RETRY   ;ERRORS, RECALIBRATE
003B'   18F3                JMPR    ..RD    ;TRY AGAIN
                    ;
003D'   3E0A        WRDSK:  MVI     A,MAXTRY  ;GET MAX TRY COUNT
003F'   32 0001"            STA     TRYCNT  ;SET TRY COUNT
0042'   3E05        ..WR:   MVI     A,FDCWR ;GET FDC WRITE COMMAND
0044'   0E80                MVI     C,DMAWR ;GET DMA WRITE COMMAND
0046'   CD 00C4'            CALL    DSKCOM  ;CALL COMMON CODE
0049'   2008                JRNZ    ..RT    ;IF ERRORS, RETRY
004B'   3E06                MVI     A,FDCRD ;ELSE, GET FDC READ COMMAND
004D'   0E00                MVI     C,DMAVFY  ;GET DMA VERIFY COMMAND
004F'   CD 00C4'            CALL    DSKCOM  ;CALL COMMON CODE
0052'   C8                  RZ              ;NO ERRORS, RET A=0
0053'   CD 011B'    ..RT:   CALL    RETRY   ;ERRORS, RECALIBRATE
0056'   18EA                JMPR    ..WR    ;TRY AGAIN
                    ;
0058'   DD7E02      FMTDSK: MOV     A,PDRTRK(X)  ;GET PD REQ TRACK NUMBER
```

```
   005B'    B7                     ORA     A         ;PD REQUEST TRACK NUMBER=0?
   005C'    2006                   JRNZ    ..NTRO    ;IF NOT, CONTINUE
   005E'    CD 02CC'               CALL    SELCUR    ;ELSE, SELECT I/O DISK
   0061'    CD 01CB'               CALL    RECAL     ;RECALIBRATE DRIVE
   0064'    3E0A           ..NTRO: MVI     A,MAXTRY  ;GET MAX TRY COUNT
   0066'    32 0001"               STA     TRYCNT    ;SET TRY COUNT
   0069'    CD 01A6'       ..FMT:  CALL    SEEK      ;SELECT DISK AND SEEK
   006C'    3E80                   MVI     A,DMAWR ;GET DMA WRITE COMMAND
   006E'    32 0005"               STA     IODMAC    ;SET DMA COMMAND
   0071'    DD6E08                 MOV     L,PDRTC(X)  ;GET PD REQ TRANSFER COUNT
   0074'    DD6609                 MOV     H,PDRTC+1(X)
   0077'    DD5E0A                 MOV     E,PDRDMA(X)  ;GET PD REQUEST DMA ADDRESS
   007A'    DD560B                 MOV     D,PDRDMA+1(X)
   007D'    CD 013D'               CALL    DMANIT    ;INITIALIZE DMA CONTROLLER
   0080'    3E0D                   MVI     A,FDCFMT  ;GET FORMAT TRACK COMMAND
   0082'    DDCB047E               BIT     7,PDRSEC(X)  ;DOUBLE DENSITY FLAG SET?
   0086'    2802                   JRZ     ..SD      ;IF NOT, CONTINUE
   0088'    CBF7                   SET     FDCMFM,A  ;ELSE, SET DOUBLE DENSITY BIT
   008A'    CD 038E'       ..SD:   CALL    CMDRDY    ;SEND FORMAT COMMAND TO FDC
   008D'    DD7E01                 MOV     A,PDRDRV(X)  ;GET PD REQUEST DRIVE NUMBER
   0090'    DDCB057E               BIT     7,PDRSEC+1(X)  ;HEAD NUMBER ONE FLAG SET?
   0094'    2802                   JRZ     ..HDO     ;IF NOT, CONTINUE
   0096'    CBD7                   SET     2,A       ;ELSE, SET HEAD ONE BIT
   0098'    CD 0394'       ..HDO:  CALL    DATOUT    ;OUTPUT UNIT NUMBER TO FDC
   009B'    DD7E04                 MOV     A,PDRSEC(X)  ;GET PD REQUEST SECTOR (LSB)
   009E'    E603                   ANI     3         ;EXTRACT FORMAT SECTOR SIZE
   00A0'    CD 0394'               CALL    DATOUT    ;OUTPUT FORMAT SECTOR SIZE TO FDC
   00A3'    DD7E06                 MOV     A,PDRSC(X)  ;GET PD REQUEST SECTOR COUNT
   00A6'    CD 0394'               CALL    DATOUT    ;OUTPUT SECTORS/TRACK TO FDC
   00A9'    DD7E05                 MOV     A,PDRSEC+1(X)  ;GET PD REQUEST SECTOR (MSB
   00AC'    E67F                   ANI     7FH       ;EXTRACT FORMAT GAP LENGTH
   00AE'    CD 0394'               CALL    DATOUT    ;OUTPUT FORMAT GAP LENGTH TO FDC
   00B1'    3EE5                   MVI     A,0E5H    ;GET FORMAT FILLER BYTE
   00B3'    CD 0394'               CALL    DATOUT    ;OUTPUT FORMAT FILLER BYTE TO FDC
   00B6'    CD 02FC'               CALL    WTINT     ;WAIT FOR INTERRUPT
   00B9'    3A 0021"               LDA     ST0       ;GET STATUS REGISTER 0
   00BC'    E6C0                   ANI     0C0H      ;ANY ERRORS?
   00BE'    C8                     RZ                ;NO ERRORS, RET A=0
   00BF'    CD 011B'               CALL    RETRY     ;ERRORS, RECALIBRATE
   00C2'    18A5                   JMPR    ..FMT     ;TRY AGAIN
                           ;
   00C4'    32 0004"       DSKCOM: STA     IORWC     ;SET FDC READ/WRITE COMMAND
   00C7'    79                     MOV     A,C       ;GET DMA COMMAND
   00C8'    32 0005"               STA     IODMAC    ;SET DMA COMMAND
   00CB'    DD7E04                 MOV     A,PDRSEC(X)  ;GET PD REQ SECTOR NUMBER
   00CE'    32 0009"               STA     CURSEC    ;SET CURRENT SECTOR
   00D1'    DD6E0A                 MOV     L,PDRDMA(X)  ;GET PD REQUEST DMA ADDRESS
   00D4'    DD660B                 MOV     H,PDRDMA+1(X)
   00D7'    22 000A"               SHLD    CURADR    ;SET CURRENT DMA ADDRESS
   00DA'    DD7E06                 MOV     A,PDRSC(X)  ;GET PD REQ SECTOR COUNT
   00DD'    32 000C"               STA     CURSC     ;SET CURRENT SECTOR COUNT
   00E0'    CD 01A6'               CALL    SEEK      ;SELECT DISK AND SEEK
   00E3'    AF                     XRA     A
   00E4'    32 000D"               STA     IOERR     ;CLEAR I/O ERROR STATUS
```

```
  00E7'   CD 0155'       ..RWL:  CALL     SETID   ;SET UP SECTOR ID INFO
  00EA'   CD 012B'               CALL     SETUP   ;SETUP READ/WRITE DMA
  00ED'   CD 035D'               CALL     CMDOUT  ;SEND SECTOR ID INFO TO FDC
  00F0'   CD 02FC'               CALL     WTINT   ;WAIT FOR INTERRUPT
  00F3'   21 000D"               LXI      H,IOERR ;GET I/O ERROR STATUS
  00F6'   3A 0021"               LDA      ST0     ;GET STATUS REGISTER 0
  00F9'   B6                     ORA      M       ;ADD NEW STATUS
  00FA'   77                     MOV      M,A     ;UPDATE I/O ERROR STATUS
  00FB'   CD 03CF'               CALL     GETXLT  ;GET TRANSLATION TABLE ADDRESS
  00FE'   2815                   JRZ      ..NI    ;IF TRANSLATION NOT REQUIRED, CONT
  0100'   21 0009"               LXI      H,CURSEC  ;ELSE, GET CURRENT SECTOR NUMBER
  0103'   34                     INR      M       ;INCREMENT CURRENT SECTOR
  0104'   CD 03C4'               CALL     CALCSS  ;CALC SECTOR SIZE
  0107'   EB                     XCHG             ;SECTOR SIZE TO DE-REG
  0108'   2A 000A"               LHLD     CURADR  ;GET CURRENT DMA ADDRESS
  010B'   19                     DAD      D       ;CALC NEXT DMA ADDRESS
  010C'   22 000A"               SHLD     CURADR  ;UPDATE CURRENT DMA ADDRESS
  010F'   21 000C"               LXI      H,CURSC ;GET CURRENT SECTOR COUNT
  0112'   35                     DCR      M       ;DECREMENT CURRENT SECTOR COUNT
  0113'   20D2                   JRNZ     ..RWL   ;IF TRANSFER NOT COMPLETE, CONTINU
  0115'   3A 000D"       ..NI:   LDA      IOERR   ;GET I/O ERROR STATUS
  0118'   E6C0                   ANI      0C0H    ;EXTRACT COMPLETION STATUS
  011A'   C9                     RET              ;DONE
                         ;
  011B'   0E07           RETRY:  MVI      C,ABEL  ;GET BELL CHARACTER
  011D'   CD 0000:07             CALL     CONOUT# ;OUTPUT TO CONSOLE
  0120'   CD 01CB'               CALL     RECAL   ;RECALIBRATE DRIVE
  0123'   21 0001"               LXI      H,TRYCNT  ;GET RETRY COUNT
  0126'   35                     DCR      M       ;DECREMENT RETRY COUNT
  0127'   C0                     RNZ              ;IF COUNT NOT EXHAUSTED, TRY AGAIN
  0128'   C3 03EC'               JMP      FATAL   ;CONTINUE
                         ;
  012B'   CD 03CF'       SETUP:  CALL     GETXLT  ;GET TRANSLATION TABLE ADDRESS
  012E'   DD6E08                 MOV      L,PDRTC(X)  ;GET PD REQ TRANSFER COUNT
  0131'   DD6609                 MOV      H,PDRTC+1(X)
  0134'   2803                   JRZ      ..NI    ;IF NO TRANSLATION RQRD, CONTINUE
  0136'   CD 03C4'               CALL     CALCSS  ;ELSE, CALC SECTOR SIZE
  0139'   ED5B 000A"     ..NI:   LDED     CURADR  ;GET CURRENT DMA ADDRESS
                         ;
  013D'   AF             DMANIT: XRA      A
  013E'   D388                   OUT      DMACTL  ;RESET DMA CONTROLLER
  0140'   2B                     DCX      H       ;TERMINAL COUNT-1 FOR 8257
  0141'   7D                     MOV      A,L     ;GET LSB OF TERMINAL COUNT
  0142'   D383                   OUT      CH1TC   ;SEND LSB OF TERMINAL COUNT
  0144'   3A 0005"               LDA      IODMAC  ;GET I/O DMA COMMAND
  0147'   B4                     ORA      H       ;ADD TO MSB OF TERMINAL COUNT
  0148'   D383                   OUT      CH1TC   ;SEND MSB OF TERMINAL COUNT
  014A'   7B                     MOV      A,E     ;GET LSB
  014B'   D382                   OUT      CH1DMA  ;OUTPUT IT TO DMA CONTROLLER
  014D'   7A                     MOV      A,D     ;GET MSB
  014E'   D382                   OUT      CH1DMA  ;OUTPUT IT TO DMA CONTROLLER
  0150'   3E42                   MVI      A,CH1ENA  ;GET CHANNEL 1 ENABLE COMMAND
  0152'   D388                   OUT      DMACTL  ;ENABLE DMA CONTROLLER
  0154'   C9                     RET              ;DONE
```

```
                                 ;
  0155'   DD7E02                 SETID:  MOV     A,PDRTRK(X)  ;GET PD REQ TRACK NUMBER
  0158'   32 001A"                       STA     CYL       ;SET CYLINDER
  015B'   3A 0009"                       LDA     CURSEC    ;GET CURRENT SECTOR
  015E'   4F                             MOV     C,A       ;SECTOR NUMBER TO C-REG
  015F'   CD 03CF'                       CALL    GETXLT    ;GET TRANSLATION TABLE ADDRESS
  0162'   2804                           JRZ     ..NI      ;IF TRANSLATION NOT REQUIRED, CONT
  0164'   0600                           MVI     B,0       ;ELSE, MAKE SECTOR DOUBLE LENGTH
  0166'   09                             DAD     B         ;INDEX INTO TRANSLATION TABLE
  0167'   4E                             MOV     C,M       ;GET TRANSLATED SECTOR NUMBER
  0168'   0C                     ..NI:   INR     C         ;CONVERT SECTOR TO BASE 1
  0169'   DD4613                         MOV     B,SECTRK(X)  ;GET NUMBER OF SECTORS/TRACK
  016C'   CD 03DD'                       CALL    GETTCA    ;GET DISK TYPE CODE ADDRESS
  016F'   CB56                           BIT     TSD,M     ;TWO SIDED DISK?
  0171'   2802                           JRZ     ..SSD     ;IF NOT, CONTINUE
  0173'   CB38                           SRLR    B         ;ELSE, CALC NUMBER OF SECTORS/SIDE
  0175'   78                     ..SSD:  MOV     A,B       ;GET NUMBER OF SECTORS/SIDE
  0176'   32 001E"                       STA     EOT       ;SET END OF TRACK SECTOR NUMBER
  0179'   B9                             CMP     C         ;FRONT SIDE OF DISK?
  017A'   3E00                           MVI     A,0       ;PRESET FOR FRONT SIDE
  017C'   3005                           JRNC    ..FS      ;IF FRONT SIDE, CONTINUE
  017E'   79                             MOV     A,C       ;GET SECTOR NUMBER
  017F'   90                             SUB     B         ;SUBTRACT ONE SIDES WORTH
  0180'   4F                             MOV     C,A       ;TO C-REG
  0181'   3E01                           MVI     A,1       ;GET HEAD #1
  0183'   32 001B"               ..FS:   STA     HEAD      ;SET HEAD NUMBER
  0186'   79                             MOV     A,C       ;GET SECTOR NUMBER
  0187'   32 001C"                       STA     REC       ;SET RECORD NUMBER
  018A'   DD7E12                         MOV     A,SECSIZ(X)  ;GET SECTOR SIZE
  018D'   32 001D"                       STA     SIZE      ;SET RECORD SIZE
  0190'   B7                             ORA     A         ;N=0?
  0191'   3E80                           MVI     A,128     ;PRESET DTL=128
  0193'   2802                           JRZ     ..NO      ;IF N=0, CONTINUE
  0195'   3EFF                           MVI     A,0FFH    ;ELSE, DTL=0FFH
  0197'   32 0020"               ..NO:   STA     DTL       ;SET DATA LENGTH
  019A'   CD 03E5'                       CALL    GETDST    ;GET DST ADDRESS
  019D'   11 0000:08                     LXI     D,GAPLEN#  ;GET OFFSET TO GAP LENGTH
  01A0'   19                             DAD     D         ;CALC GAP LENGTH ADDRESS
  01A1'   7E                             MOV     A,M       ;GET GAP LENGTH
  01A2'   32 001F"                       STA     GPL       ;SET GAP LENGTH
  01A5'   C9                             RET               ;DONE
                                 ;
  01A6'   CD 02CC'               SEEK:   CALL    SELCUR    ;SELECT I/O DISK
  01A9'   DD7E01                         MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
  01AC'   3C                             INR     A         ;INCREMENT IT
  01AD'   47                             MOV     B,A       ;TO B-REG
  01AE'   AF                             XRA     A         ;CLEAR DRIVE VECTOR
  01AF'   37                             STC               ;SET CARRY FLAG
  01B0'   8F                     ..SL:   ADC     A         ;SHIFT CARRY FLAG LEFT
  01B1'   10FD                           DJNZ    ..SL
  01B3'   21 0002"                       LXI     H,CALTBL  ;GET DRIVE CALIBRATED TABLE
  01B6'   B6                             ORA     M         ;COMBINE VECTOR W/CALIBRATED TABLE
  01B7'   BE                             CMP     M         ;DRIVE ALREADY CALIBRATED?
  01B8'   77                             MOV     M,A       ;UPDATE DRIVE CALIBRATED TABLE
  01B9'     C4 01CB'                  CNZ     RECAL    ;RE-CALIBRATE DRIVE, IF NECESSARY
  01BC'     DD7E01                    MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
  01BF'     47                        MOV     B,A      ;CONTROLLER DISK TO B-REG
  01C0'     DD7E02                    MOV     A,PDRTRK(X)  ;GET PD REQ TRACK NUMBER
  01C3'     4F                        MOV     C,A      ;CYLINDER TO C-REG
  01C4'     CD 01EA'                  CALL    SEKCMD   ;SEND SEEK COMMAND
  01C7'     C8                        RZ               ;IF NO ERRORS, DONE
  01C8'     C3 03EC'                  JMP     FATAL    ;CONTINUE
                             ;
  01CB'     DD7E01           RECAL:   MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
  01CE'     CD 01D5'                  CALL    RECCMD   ;SEND RECALIBRATE COMMAND
  01D1'     C8                        RZ               ;IF NO ERRORS, DONE
  01D2'     C3 03EC'                  JMP     FATAL    ;CONTINUE
                             ;
  01D5'     F5               RECCMD:  PUSH    PSW      ;SAVE CONTROLLER DISK
  01D6'     3E07                      MVI     A,FDCRCL  ;GET FDC RECALIBRATE COMMAND
  01D8'     CD 038E'                  CALL    CMDRDY   ;OUTPUT COMMAND TO FDC
  01DB'     F1                        POP     PSW      ;RESTORE CONTROLLER DISK
  01DC'     CD 0394'                  CALL    DATOUT   ;OUTPUT IT TO FDC
  01DF'     CD 02FC'                  CALL    WTINT    ;WAIT FOR INTERRUPT
  01E2'     3A 0021"                  LDA     ST0      ;GET STATUS REGISTER 0
  01E5'     E6E0                      ANI     0C0H!1<FDCSE  ;EXTRACT COMPLETION STATUS
  01E7'     FE20                      CPI     1<FDCSE  ;ANY ERRORS?
  01E9'     C9                        RET              ;DONE
                             ;
  01EA'     C5               SEKCMD:  PUSH    B        ;SAVE DISK/TRACK
  01EB'     3E0F                      MVI     A,FDCSK  ;GET FDC SEEK COMMAND
  01ED'     CD 038E'                  CALL    CMDRDY   ;OUTPUT COMMAND TO FDC
  01F0'     C1                        POP     B        ;RESTORE DISK/TRACK
  01F1'     C5                        PUSH    B        ;SAVE DISK/TRACK
  01F2'     78                        MOV     A,B      ;GET CONTROLLER DISK
  01F3'     CD 0394'                  CALL    DATOUT   ;OUTPUT IT TO FDC
  01F6'     C1                        POP     B        ;RESTORE DISK/TRACK
  01F7'     79                        MOV     A,C      ;GET CYLINDER NUMBER
  01F8'     CD 0394'                  CALL    DATOUT   ;OUTPUT IT TO FDC
  01FB'     CD 02FC'                  CALL    WTINT    ;WAIT FOR INTERRUPT
  01FE'     3A 0021"                  LDA     ST0      ;GET STATUS REGISTER 0
  0201'     E6E0                      ANI     0C0H!1<FDCSE  ;EXTRACT COMPLETION STATUS
  0203'     FE20                      CPI     1<FDCSE  ;ANY ERRORS?
  0205'     C9                        RET              ;DONE
                             ;
  0206'     CD 027F'         RETDST:  CALL    RETRDY   ;RETURN READY STATUS
  0209'     B7                        ORA     A        ;DRIVE READY?
  020A'     C8                        RZ               ;IF NOT, DONE
  020B'     0E00                      MVI     C,0      ;ELSE, GET INITIAL TYPE VALUE
  020D'     3A 0029"                  LDA     ST3      ;GET STATUS REGISTER 3
  0210'     CB5F                      BIT     ST3TS,A  ;ONE-SIDED DISK?
  0212'     2802                      JRZ     ..OS     ;YES
  0214'     CBD1                      SET     TSD,C    ;SET TWO-SIDED DISK BIT
  0216'     DD7E01           ..OS:    MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
  0219'     32 0008"                  STA     RIDDSK   ;SET READ ID DISK
  021C'     CD 026A'                  CALL    ..FD     ;FIND DISK DENSITY
  021F'     280F                      JRZ     ..DF     ;IF DENSITY FOUND, CONTINUE
  0221'     C5                        PUSH    B        ;ELSE, SAVE DISK TYPE CODE
  0222'   DD7E01                  MOV    A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
  0225'   CD 01D5'                CALL   RECCMD  ;RECALIBRATE DRIVE
  0228'   C1                      POP    B       ;RESTORE DISK TYPE CODE
  0229'   2032                    JRNZ   ..NR    ;IF UNABLE TO RECALIBRATE, CONTINU
  022B'   CD 026A'                CALL   ..FD    ;ELSE, ATTEMPT TO FIND DISK DENSIT
  022E'   202D                    JRNZ   ..NR    ;IF DENSITY NOT FOUND, CONTINUE
  0230'   B1             ..DF:    ORA    C       ;ADD SECTOR SIZE TO TYPE CODE
  0231'   4F                      MOV    C,A
  0232'   CB51                    BIT    TSD,C   ;TWO SIDED BIT SET?
  0234'   2814                    JRZ    ..FDI   ;IF NOT, CONTINUE
  0236'   21 0008"                LXI    H,RIDDSK  ;GET READ ID DISK
  0239'   CBD6                    SET    2,M     ;SET HEAD BIT
  023B'   3E4A                    MVI    A,FDCRID|1<FDCMFM  ;GET READ ID CMD (DD)
  023D'   CB59                    BIT    DDD,C   ;DOUBLE DENSITY BIT SET?
  023F'   2002                    JRNZ   ..DD    ;IF SO, CONTINUE
  0241'   CBB7                    RES    FDCMFM,A  ;ELSE, RESET MFM BIT
  0243'   CD 0279'       ..DD:    CALL   ..RID   ;ATTEMP TO READ ID ON BACK SIDE
  0246'   2802                    JRZ    ..FDI   ;IF READABLE, CONTINUE
  0248'   CB91                    RES    TSD,C   ;ELSE, RESET TWO SIDED BIT
  024A'   11 0000:09     ..FDI:   LXI    D,DSTBLS#  ;GET DISK SPEC TABLES
  024D'   79             ..SL2:   MOV    A,C     ;GET DISK TYPE CODE
  024E'   21 0000:0A              LXI    H,DTCO# ;GET OFFSET TO DISK TYPE CODE
  0251'   19                      DAD    D       ;CALC DISK TYPE CODE ADDRESS
  0252'   BE                      CMP    M       ;DISK SPEC TABLE FOUND?
  0253'   280A                    JRZ    ..DSTF  ;IF SO, CONTINUE
  0255'   EB                      XCHG           ;DISK SPEC TABLE ADDRESS TO HL-REG
  0256'   5E                      MOV    E,M     ;GET DISK SPEC TABLE LINK POINTER
  0257'   23                      INX    H
  0258'   56                      MOV    D,M
  0259'   7A                      MOV    A,D
  025A'   B3                      ORA    E       ;END OF LIST?
  025B'   20F0                    JRNZ   ..SL2   ;IF NOT, CONTINUE
  025D'   AF             ..NR:    XRA    A       ;ELSE, SET RETURN CODE=0
  025E'   C9                      RET            ;DONE
  025F'   13             ..DSTF:  INX    D       ;ADVANCE PAST LINK POINTER
  0260'   13                      INX    D
  0261'   DD730C                  MOV    PDRDST(X),E  ;SET DISK SPEC TABLE ADDRESS
  0264'   DD720D                  MOV    PDRDST+1(X),D
  0267'   3EFF                    MVI    A,0FFH  ;SET RETURN CODE=0FFH
  0269'   C9                      RET            ;DONE
  026A'   3E0A           ..FD:    MVI    A,FDCRID  ;GET FDC READ ID COMMAND (SD)
  026C'   CD 0279'                CALL   ..RID   ;ATTEMPT TO READ SINGLE-DENSITY
  026F'   C8                      RZ             ;IF SINGLE-DENSITY, DONE
  0270'   3E4A                    MVI    A,FDCRID|1<FDCMFM  ;GET READ ID CMD (DD)
  0272'   CD 0279'                CALL   ..RID   ;ATTEMPT TO READ DOUBLE-DENSITY
  0275'   C0                      RNZ            ;IF UNABLE, DONE
  0276'   CBD9                    SET    DDD,C   ;SET DOUBLE-DENSITY DISK BIT
  0278'   C9                      RET            ;DONE
  0279'   C5             ..RID:   PUSH   B       ;SAVE BC
  027A'   CD 02A2'                CALL   READID  ;READ DISK ID
  027D'   C1                      POP    B       ;RESTORE BC
  027E'   C9                      RET            ;DONE
                         ;
  027F'   DD7E01         RETRDY:  MOV    A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
```

```
   0282'    FE04                   CPI     4         ;TEST FOR VALID DRIVE NUMBER
   0284'    3E00                   MVI     A,0       ;PRESET RETURN CODE=0
   0286'    D0                     RNC               ;IF INVALID DRIVE, RETURN NOT READ
   0287'    DB8E                   IN      FDCST     ;GET FDC STATUS
   0289'    3C                     INR     A         ;CONTROLLER PRESENT?
   028A'    C8                     RZ                ;IF NOT, DONE
   028B'    CD 02CC'               CALL    SELCUR    ;ELSE, SELECT REQUESTED DRIVE
   028E'    CD 0298'               CALL    ..RDY     ;CHECK IF DRIVE READY
   0291'    C0                     RNZ               ;IF SO, DONE
   0292'    21 0001                LXI     H,1       ;ELSE, DELAY ONE TICK...
   0295'    CD 0000:0B             CALL    DELAY#    ;...SO 765 CAN SCAN
   0298'    CD 02E9'       ..RDY:  CALL    SENSDS    ;SENSE DRIVE STATUS
   029B'    CB6F                   BIT     ST3RDY,A  ;DRIVE READY?
   029D'    3E00                   MVI     A,0       ;PRESET RETURN CODE=0
   029F'    C8                     RZ                ;IF DRIVE NOT READY, DONE
   02A0'    2F                     CMA               ;ELSE, SET RETURN CODE=OFFH
   02A1'    C9                     RET               ;DONE
                           ;
   02A2'    CD 038E'       READID: CALL    CMDRDY    ;OUTPUT COMMAND TO FDC
   02A5'    3A 0008"               LDA     RIDDSK    ;GET READ ID DISK
   02A8'    CD 0394'               CALL    DATOUT    ;OUTPUT IT TO FDC
   02AB'    CD 02FC'               CALL    WTINT     ;WAIT FOR INTERRUPT
   02AE'    3A 0021"               LDA     STO       ;GET STATUS REGISTER 0
   02B1'    E6C0                   ANI     0COH      ;EXTRACT COMPLETION STATUS
   02B3'    3A 0027"               LDA     RSIZE     ;RETURN SECTOR SIZE
   02B6'    C9                     RET               ;DINE
                           ;
   02B7'                   DLCPOL:                   ;DELAY COMPLETE POLL ROUTINE
   02B7'    0000                   .WORD   0
   02B9'    0000                   .WORD   0
                           ;
   02BB'    DB8C           DLCPR:  IN      DSKCTL    ;GET DISK CONTROLLER STATUS
   02BD'    CB7F                   BIT     DSKDLC,A  ;DELAY COMPLETE (MOTORS RUNNING)
   02BF'    C8                     RZ                ;IF NOT, DONE
   02C0'    21 02B7'               LXI     H,DLCPOL  ;ELSE, GET POLL ROUTINE
   02C3'    CD 0000:0C             CALL    UNLINK#   ;UNLINK POLL ROUTINE FROM POLL LIS
   02C6'    21 0014"               LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
   02C9'    C3 0000:06             JMP     SIGNAL#   ;CONTINUE
                           ;
   02CC'    DB8C           SELCUR: IN      DSKCTL    ;GET DISK CONTROLLER STATUS
   02CE'    0F                     RRC               ;EXTRACT SELECTED DRIVE
   02CF'    E603                   ANI     3
   02D1'    4F                     MOV     C,A       ;TO C-REG
   02D2'    DD7E01                 MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
   02D5'    B9                     CMP     C         ;DRIVE ALREADY SELECTED?
   02D6'    2802                   JRZ     ..DAS     ;IF SO, CONTINUE
   02D8'    D38A                   OUT     DSKSEL    ;ELSE, SELECT CONTROLLER DISK
   02DA'    11 02B7'       ..DAS:  LXI     D,DLCPOL  ;GET POLL ROUTINE
   02DD'    CD 0000:0D             CALL    LNKPOL#   ;CREATE POLL ROUTINE
   02E0'    CD 02BB'               CALL    DLCPR     ;EXECUTE POLL ROUTINE
   02E3'    21 0014"               LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
   02E6'    C3 0000:05             JMP     WAIT#     ;DISPATCH IF NECESSARY
                           ;
   02E9'    3E04           SENSDS: MVI     A,FDCSDS  ;GET FDC SENSE DRIVE STATUS CMD
```

```
 02EB'   CD 038E'                 CALL    CMDRDY  ;OUTPUT COMMAND TO FDC
 02EE'   DD7E01                   MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
 02F1'   CD 0394'                 CALL    DATOUT  ;OUTPUT IT TO FDC
 02F4'   CD 039B'                 CALL    DATAIN  ;GET STATUS REGISTER 3
 02F7'   32 0029"                 STA     ST3     ;SAVE STATUS REGISTER 3
 02FA'   FB                       EI              ;ENABLE INTERRUPTS
 02FB'   C9                       RET             ;DONE
                         ;
 02FC'   FB              WTINT:   EI              ;ENABLE INTERRUPTS
 02FD'   21 0014"                 LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
 0300'   C3 0000:05               JMP     WAIT#   ;DISPATCH IF NECESSARY
                         ;
 0303'   ED73 0000:0E    DSKISR:  SSPD    INTSP#  ;SAVE INTERRUPT STACK PCINTER
 0307'   31 0000:0F               LXI     SP,INTSTK#  ;SET UP AUX STACK
 030A'   F5                       PUSH    PSW     ;SAVE REGISTERS
 030B'   C5                       PUSH    B
 030C'   D5                       PUSH    D
 030D'   E5                       PUSH    H
 030E'   DB8E            ..RQML:  IN      FDCST   ;GET FDC STATUS
 0310'   CB7F                     BIT     FDCRDY,A  ;FDC READY FOR CONVERSATION?
 0312'   28FA                     JRZ     ..RQML  ;IF NOT, WAIT
 0314'   32 0028"                 STA     MAINST  ;SAVE MAIN STATUS REGISTER
 0317'   CB77                     BIT     FDCOUT,A  ;FDC IN OUTPUT MODE?
 0319'   2020                     JRNZ    ..RW    ;IF SO, PROCESS
 031B'   3E08                     MVI     A,FDCSIS  ;GET SENSE INTERRUPT STATUS CMD
 031D'   D38F                     OUT     FDCDAT  ;OUTPUT IT TO FDC DATA REGISTER
 031F'   CD 039B'                 CALL    DATAIN  ;GET STATUS REGISTER 0
 0322'   4F                       MOV     C,A     ;SAVE IT IN C-REG
 0323'   E6C0                     ANI     0C0H    ;EXTRACT COMPLETION STATUS
 0325'   FE80                     CPI     80H     ;INTERRUPT STACK EMPTY?
 0327'   2829                     JRZ     ..X     ;IF SO, DONE
 0329'   CD 039B'                 CALL    DATAIN  ;GET PRESENT CYLINDER NUMBER
 032C'   CB69                     BIT     STOSE,C ;READY LINE CHANGE STATE?
 032E'   28DE                     JRZ     ..RQML  ;IF SO, IGNORE
 0330'   32 0024"                 STA     RCYL    ;ELSE, SAVE PCN
 0333'   79                       MOV     A,C     ;GET STATUS REGISTER 0
 0334'   32 0021"                 STA     STO     ;SAVE IT
 0337'   3E01                     MVI     A,1     ;SET INTERRUPT COMPLETION STATUS
 0339'   180F                     JMPR    ..SIGC  ;CONTINUE
 033B'   21 0021"        ..RW:    LXI     H,RESULT  ;GET RESULT TABLE
 033E'   0607                     MVI     B,7     ;GET LENGTH OF RESULT PHASE
 0340'   CD 039B'        ..RL:    CALL    DATAIN  ;GET RESULT BYTE FROM FDC
 0343'   77                       MOV     M,A     ;STORE IN RESULT AREA
 0344'   23                       INX     H       ;INCREMENT POINTER
 0345'   10F9                     DJNZ    ..RL    ;READ ALL SEVEN BYTES
 0347'   AF                       XRA     A
 0348'   D388                     OUT     DMACTL  ;DISABLE DMA CONTROLLER
 034A'   21 0014"        ..SIGC:  LXI     H,DWTSPH  ;GET DISK WAIT SEMAPHORE
 034D'   CD 0000:06               CALL    SIGNAL# ;SIGNAL PROCESS AS READY
 0350'   18BC                     JMPR    ..RQML  ;FLUSH ANY REMAINING INTERRUPTS
 0352'   E1              ..X:     POP     H       ;REGISTERS
 0353'   D1                       POP     D
 0354'   C1                       POP     B
 0355'   F1                       POP     PSW
```

```
0356'   ED7B 0000:0E            LSPD    INTSP#  ;RESTORE STACK POINTER
035A'   C3 0000:10              JMP     ISRXIT# ;CONTINUE
                        ;
035D'   CD 03DD'        CMDOUT: CALL    GETTCA  ;GET DISK TYPE CODE ADDRESS
0360'   3A 0004"                LDA     IORWC   ;GET READ/WRITE COMMAND
0363'   CB5E                    BIT     DDD,M   ;DOUBLE DENSITY DISK?
0365'   2802                    JRZ     ..SD    ;IF NOT, SINGLE DENSITY
0367'   CBF7                    SET     FDCMFM,A  ;ELSE, SET DOUBLE DENSITY BIT
0369'   CB56            ..SD:   BIT     TSD,M   ;TWO-SIDED DISK?
036B'   2802                    JRZ     ..SS    ;IF NOT, SINGLE SIDED
036D'   CBFF                    SET     FDCMT,A  ;ELSE, SET MULTI-TRACK BIT
036F'   CD 038E'        ..SS:   CALL    CMDRDY  ;SEND COMMAND TO FDC
0372'   DD7E01                  MOV     A,PDRDRV(X)  ;GET PD REQ DISK NUMBER
0375'   21 001B"                LXI     H,HEAD  ;GET HEAD NUMBER
0378'   CB46                    BIT     0,M     ;HEAD #0?
037A'   2802                    JRZ     ..FS    ;IF SO, CONTINUE
037C'   CBD7                    SET     2,A     ;ELSE, SET HEAD #1 BIT IN I/O DISK
037E'   CD 0394'        ..FS:   CALL    DATOUT  ;OUTPUT IT TO FDC
0381'   21 001A"                LXI     H,IDINFO  ;GET SECTOR ID INFO
0384'   0607                    MVI     B,7     ;B=LENGTH OF ID INFO
0386'   7E              ..IDL:  MOV     A,M     ;GET BYTE FROM LIST
0387'   23                      INX     H       ;INCREMENT POINTER
0388'   CD 0394'                CALL    DATOUT  ;OUTPUT BYTE TO FDC
038B'   10F9                    DJNZ    ..IDL   ;SEND ENTIRE LIST
038D'   C9                      RET             ;DONE
                        ;
038E'   CD 03A6'        CMDRDY: CALL    OUTRDY  ;WAIT FOR FDC READY
0391'   F3                      DI              ;DISABLE INTERRUPTS
0392'   1803                    JMPR    OUTCOM  ;JOIN COMMON CODE
                        ;
0394'   CD 03A6'        DATOUT: CALL    OUTRDY  ;WAIT FOR FDC READY
                        ;
0397'   79              OUTCOM: MOV     A,C     ;RESTORE OUTPUT BUTE
0398'   D38F                    OUT     FDCDAT  ;OUTPUT BYTE TO FDC DATA REGISTER
039A'   C9                      RET             ;DONE
                        ;
039B'   DB8E            DATAIN: IN      FDCST   ;GET FDC STATUS
039D'   07                      RLC             ;TEST FDC FOR READY
039E'   30FB                    JRNC    DATAIN  ;IF NOT READY, WAIT
03A0'   07                      RLC             ;TEST FDC DIRECTION
03A1'   300B                    JRNC    FDCERR  ;IF WRONG DIRECTION, DIAGNOSE
03A3'   DB8F                    IN      FDCDAT  ;GET FDC DATA BYTE
03A5'   C9                      RET             ;DONE
                        ;
03A6'   4F              OUTRDY: MOV     C,A     ;SAVE OUTPUT BYTE
03A7'   DB8E            ..RW:   IN      FDCST   ;GET FDC STATUS
03A9'   07                      RLC             ;TEST FDC FOR READY
03AA'   30FB                    JRNC    ..RW    ;IF NOT READY, WAIT
03AC'   07                      RLC             ;TEST FDC DIRECTION
03AD'   D0                      RNC             ;IF DIRECTION CORRECT, DONE
                        ;
03AE'   CD 0000:11      FDCERR: CALL    DMS#    ;SOUND BELL
03B1'   87                      .ASCIS  [ABEL]
03B2'   CD 0000:12              CALL    CONSO#  ;SHIFT CONSOLE TO ERROR LINE
```

```
  03B5'    CD 0000:11                  CALL    DMS#     ;DISPLAY ERROR MESSAGE
  03B8'    464443204572                .ASCIS  "FDC Error"
  03C1'    C3 03C1'                    JMP     .        ;HALT
                              ;
  03C4'    21 0080            CALCSS:  LXI     H,128    ;GET 128 BYTE SECTOR LENGTH
  03C7'    DD7E12                      MOV     A,SECSIZ(X)  ;GET SECTOR SIZE
  03CA'    3D                 ..SL:    DCR     A        ;DECREMENT SECTOR SIZE
  03CB'    F8                          RM               ;IF UNDERFLOW, DONE
  03CC'    29                          DAD     H        ;ELSE, SHIFT SECTOR SIZE LEFT
  03CD'    18FB                        JMPR    ..SL     ;CONTINUE
                              ;
  03CF'    CD 03E5'           GETXLT:  CALL    GETDST   ;GET DST ADDRESS
  03D2'    11 0000:13                  LXI     D,XLTBL#  ;GET OFFSET TO TRANSLATION TABLF
  03D5'    19                          DAD     D        ;CALC TRANSL\TION TABLE ADDRESS
  03D6'    5E                          MOV     E,M      ;GET TRANSLATION TABLE ADDRESS
  03D7'    23                          INX     H
  03D8'    56                          MOV     D,M
  03D9'    EB                          XCHG             ;TRANSLATION TABLE ADDRESS TO HL-R

  03DA'    7C                          MOV     A,H
  03DB'    B5                          ORA     L        ;TRANSLATION REQUIRED?
  03DC'    C9                          RET              ;DONE
                              ;
  03DD'    CD 03E5'           GETTCA:  CALL    GETDST   ;GET DST ADDRESS
  03E0'    11 0000:14                  LXI     D,TYPCOD#  ;GET OFFSET TO DISK TYPE CODE
  03E3'    19                          DAD     D        ;CALC DISK TYPE CODE ADDRESS
  03E4'    C9                          RET              ;DONE
                              ;
  03E5'    DD6E0C             GETDST:  MOV     L,PDRDST(X)  ;GET PD REQUEST DST ADDRESS
  03E8'    DD660D                      MOV     H,PDRDST+1(X)
  03EB'    C9                          RET              ;DONE
                              ;
  03EC'    ED7B 0006"         FATAL:   LSPD    RETSP    ;RESTORE STACK POINTER
  03F0'    3EFF                        MVI     A,0FFH   ;RETURN ERROR CODE
  03F2'    C9                          RET              ;DONE
                              ;
                              .END
```

```
                             ;
                             ; ****************************************************
                             ; *                                                  *
                             ; *  This is a list of disk specification tables for *
                             ; *  all of the 8-inch floppy disk formats supported *
                             ; *  by the DSK401 disk driver module and the MPB401 *
                             ; *  boot prom driver module.                        *
                             ; *                                                  *
                             ; ****************************************************
                             ;
                             ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                             ;
                             ; AUTHORS: RONALD E. RAIKES
                             ;          MICHAEL D. BUSCH
                             ;
                             ; VERSION: EXAMPLE
                             ;
                             .IDENT  DSKFMT          ;MODULE ID
                             ;
                             .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                             ;
  0002                       TSD     = 2             ;TWO-SIDED DISK BIT (TYPE CODE)
  0003                       DDD     = 3             ;DOUBLE DENSITY DISK BIT (TYPE COD

                             ;
  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, ONE-SIDED
                             ;
  0011'   0022'                      .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
  0013'   04                         .BYTE   4       ;BLOCK SIZE
  0014'   0134                       .WORD   (77*(8*(1<3)))/(1<4)  ;NUMBER OF BLOCKS
  0016'   03                         .BYTE   3       ;NUMBER OF DIRECTORY BLOCKS
  0017'   03                         .BYTE   3       ;PHYSICAL SECTOR SIZE (2^N*128)
  0018'   0008                       .WORD   8       ;PHYSICAL SECTORS PER TRACK
  001A'   004D                       .WORD   77      ;PHYSICAL TRACKS PER DISK
  001C'   0000                       .WORD   0       ;RESERVED TRACKS
  001E'   0000                       .WORD   0       ;TRANSLATION TABLE ADDRESS
  0020'   0B                         .BYTE   1<DDD!3 ;DISK TYPE CODE
  0021'   35                         .BYTE   35H     ;GAP LENGTH
```

```
                                ;
                                ;       512 BYTE SECTOR, SINGLE-DENSITY, TWO-SIDED
                                ;
   0022'   0033'                        .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
   0024'   04                           .BYTE   4       ;BLOCK SIZE
   0025'   0134                         .WORD   (77*(16*(1<2)))/(1<4)  ;NUMBER OF BLOCKS
   0027'   03                           .BYTE   3       ;NUMBER OF DIRECTORY BLOCKS
   0028'   02                           .BYTE   2       ;PHYSICAL SECTOR SIZE (2^N*128)
   0029'   0010                         .WORD   16      ;PHYSICAL SECTORS PER TRACK
   002B'   004D                         .WORD   77      ;PHYSICAL TRACKS PER DISK
   002D'   0000                         .WORD   0       ;RESERVED TRACKS
   002F'   0000                         .WORD   0       ;TRANSLATION TABLE ADDRESS
   0031'   06                           .BYTE   1<TSD!2 ;DISK TYPE CODE
   0032'   1B                           .BYTE   1BH     ;GAP LENGTH
                                ;
                                ;       512 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED
                                ;
   0033'   0044'                        .WORD   .+DSTL  ;DISK SPEC TABLE LINK POINTER
   0035'   04                           .BYTE   4       ;BLOCK SIZE
   0036'   009A                         .WORD   (77*(8*(1<2)))/(1<4)  ;NUMBER OF BLOCKS
   0038'   02                           .BYTE   2       ;NUMBER OF DIRECTORY BLOCKS
   0039'   02                           .BYTE   2       ;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'   02                           .BYTE   2       ;DISK TYPE CODE
   0043'   1B                           .BYTE   1BH     ;GAP LENGTH
                                ;
                                ;       128 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED
                                ;
   0044'   0000                 DSTA:   .WORD   0       ;DISK SPEC TABLE LINK POINTER
   0046'   03                   DSTB:   .BYTE   3       ;BLOCK SIZE
   0047'   00F3                         .WORD   (75*(26*(1<0)))/(1<3)  ;NUMBER OF BLOCKS
   0049'   02                           .BYTE   2       ;NUMBER OF DIRECTORY BLOCKS
   004A'   00                           .BYTE   0       ;PHYSICAL SECTOR SIZE (2^N*128)
   004B'   001A                         .WORD   26      ;PHYSICAL SECTORS PER TRACK
   004D'   004D                         .WORD   77      ;PHYSICAL TRACKS PER DISK
   004F'   0002                         .WORD   2       ;RESERVED TRACKS
                                ;
   000B                         XLTBL   =: .-DSTB       ;TRANSLATION TABLE ADDRESS OFFSET
                                ;
   0051'   0055'                        .WORD   TRTBL   ;TRANSLATION TABLE ADDRESS
                                ;
   000F                         DTCO    =: .-DSTA       ;DISK TYPE CODE OFFSET
   000D                         TYPCOD  =: .-DSTB       ;DISK TYPE CODE OFFSET
                                ;
   0053'   00                           .BYTE   0       ;DISK TYPE CODE
                                ;
   000E                         GAPLEN  =: .-DSTB       ;GAP LENGTH OFFSET
                                ;
   0054'   07                           .BYTE   7       ;GAP LENGTH
                                ;
   0011                         DSTL    = .-DSTA        ;DISK SPEC TABLE LENGTH
```

```
                            ;
                            ; SINGLE-DENSITY/SINGLE-SIDED SECTOR TRANSLATION TABLE
                            ;
  0055'   00060C121804      TRTBL:  .BYTE   0,6,12,18,24,4,10,16,22
  005E'   02080E140107              .BYTE   2,8,14,20,1,7,13,19,25
  0067'   050B11170309              .BYTE   5,11,17,23,3,9,15,21
                            ;
                            .END
```

CON192 - TURBODOS OPERATING SYSTEM 19.2KB CONSOLE DRIVER
COPYRIGHT (C) 1982, SOFTWARE 2000, INC.

```
                                ;
                                ; ***************************************************
                                ; *                                                 *
                                ; *  This is a trivial console driver for a 19.2KB  *
                                ; *  console device connected to serial channel zero. *
                                ; *  This module is hardware-independent.  All access *
                                ; *  to the serial interface hardware is accomplished *
                                ; *  via calls to the entrypoint SERIAL:: in the hard- *
                                ; *  ware dependent serial driver modules SPD442 or  *
                                ; *  SPD740.                                        *
                                ; *                                                 *
                                ; ***************************************************
                                ;
                                ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                                ;
                                ; AUTHORS: RONALD E. RAIKES
                                ;          MICHAEL D. BUSCH
                                ;
                                ; VERSION: EXAMPLE
                                ;
                                .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 BAU

   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 R
                                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
                            ;
                            .END
```

```
                          ;
                          ; ******************************************************
                          ; *                                                    *
                          ; *  This is a printer driver for a 300 baud TTY-like  *
                          ; *  printer device connected to a serial channel 0-15. *
                          ; *  It is coded to support up to 16 channels at once. *
                          ; *  This module is hardware-independent.  All access  *
                          ; *  to the serial interface hardware is accomplished  *
                          ; *  via calls to the entrypoint SERIAL:: in the hard- *
                          ; *  ware dependent serial driver modules SPD442 or    *
                          ; *  SPD740.                                           *
                          ; *                                                    *
                          ; ******************************************************
                          ;
                          ; COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.
                          ;
                          ; AUTHORS: RONALD E. RAIKES
                          ;          MICHAEL D. BUSCH
                          ;
                          ; VERSION: EXAMPLE
                          ;
                          .IDENT  LST300          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
 0000"                            .LOC    .DATA.# ;LOCATE IN DATA AREA
                          ;
 0000"   25               LST3BR::.BYTE   25H     ;BAUD RATE CODE (300 BAUD)
 0001"   0C               LST3FF::.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     LST3BR  ;GET BAUD RATE CODE
 001E'   4F                       MOV     C,A     ;BAUD RATE CODE TO C-REG
 001F'   1E03                     MVI     E,3     ;SET FUNCTION NUMBER=3
```

LSTCTS - TURBODOS OPERATING SYSTEM CLEAR-TO-SEND PRINTER DRIVER
COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.

```
                              ;
                              ; *****************************************************
                              ; *                                                   *
                              ; *  This is a printer driver for 9600 baud printers  *
                              ; *  using clear-to-send handshaking (e.g. TI-810).   *
                              ; *  It is coded to support up to 16 channels at once. *
                              ; *  This module is hardware-independent.  All access *
                              ; *  to the serial interface hardware is accomplished *
                              ; *  via calls to the entrypoint SERIAL:: in the hard- *
                              ; *  ware dependent serial driver modules SPD442 or   *
                              ; *  SPD740.                                          *
                              ; *                                                   *
                              ; *****************************************************
                              ;
                              ; COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.
                              ;
                              ; AUTHORS: RONALD E. RAIKES
                              ;          MICHAEL D. BUSCH
                              ;
                              ; VERSION: EXAMPLE
                              ;
                              .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
                              ;
                              .END
```

```
                        ;
                        ; *****************************************************
                        ; *                                                   *
                        ; *  This is a printer driver for 1200 baud printers  *
                        ; *  using ETX/ACK protocol (e.g., Diablo or NEC daisy). *
                        ; *  It is coded to support up to 16 channels at once. *
                        ; *  This module is hardware-independent.  All access  *
                        ; *  to the serial interface hardware is accomplished  *
                        ; *  via calls to the entrypoint SERIAL:: in the hard- *
                        ; *  ware dependent serial driver modules SPD442 or    *
                        ; *  SPD740.                                           *
                        ; *                                                   *
                        ; *****************************************************
                        ;
                        ; COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.
                        ;
                        ; AUTHORS: RONALD E. RAIKES
                        ;          MICHAEL D. BUSCH
                        ;
                        ; VERSION: EXAMPLE
                        ;
                        .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
```

```
   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
                             ;
                             .END
```

```
  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 STATUS
  0058'   D1                ..SSIC: POP     D       ;RESTORE FUNCTION NUMBER
  0059'   C1                        POP     B       ;RESTORE CHANNEL NUMBER/CHARACTER
  005A'   C9                        RET             ;DONE
                            ;
                            .END
```

```
;
; *********************************************************
; *                                                       *
; *  This is a fairly complex network circuit driver      *
; *  which runs in an IMS 8000 master processor and       *
; *  which communicates with a simple circuit driver      *
; *  (SCD740) running in an IMS 740 slave processor       *
; *  board.  This driver accomodates up to sixteen        *
; *  IMS 740 slave boards.                                *
; *                                                       *
; *  The IMS 740 slave communicates with the master       *
; *  processor over the S-100 bus via a byte-parallel     *
; *  transfer which is handled by programmed I/O by       *
; *  both processors.  The IMS 740 slave appears to the   *
; *  master processor as an ordinary port-addressed       *
; *  peripheral device.  The interface on the 740 board   *
; *  is an Intel 8255 parallel I/O controller chip,       *
; *  and coordination between the master and slave is     *
; *  handled by the status bits of the 8255.  The         *
; *  master processor may interrupt the slave to see      *
; *  if it is still healthy, and can also reset and       *
; *  download the slave.                                  *
; *                                                       *
; *  This driver maintains a "watchdog timer" for each    *
; *  of the slave processors.  If it has not received     *
; *  any messages from a particular slave processor       *
; *  after a one-second interval, it polls that slave     *
; *  by interrupting it and looking for an acknowledge-   *
; *  ment from the slave's interrupt service routine.     *
; *  If no acknowledgement is received within a brief     *
; *  timeout interval, then this driver concludes that    *
; *  the slave has crashed.  The driver informs Turbo-    *
; *  DOS of the crash (so TurboDOS can take its re-       *
; *  covery action), and then the driver resets and       *
; *  downloads a fresh copy of the operating system       *
; *  into the failed slave.                               *
; *                                                       *
; *  The IMS 740 slave board has a start-up ROM which     *
; *  is programmed with the contents of the module        *
; *  SPB740 to handle the slave-end of the download       *
; *  procedure.  At the start of a download, this dri-    *
; *  ver sends (and the SPB740 slave ROM receives) a      *
; *  loader module LOD740 to the slave.  This loader      *
; *  module handles the remainder of the downloading      *
; *  task by making standard TurboDOS network download    *
; *  requests to load the full slave operating system     *
; *  from a file OSSLAVE.SYS on the master's disk.        *
; *                                                       *
; *********************************************************
;
; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
;
; AUTHORS: RONALD E. RAIKES
;          MICHAEL D. BUSCH
```

```
                        ;
                        ; VERSION: EXAMPLE
                        ;
                        .IDENT  MCD740          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
  0005                  IBF     = 5             ;INPUT BUFFER FULL BIT
  0007                  OBFN    = 7             ;INPUT BUFFER FULL (NOT) BIT
                        ;
  0000                  RESPCO  = 00H           ;RESET PCO COMMAND
  0001                  SETPCO  = 01H           ;SET PCO COMMAND
  0002                  RESPC1  = 02H           ;RESET PC1 COMMAND
  0003                  SETPC1  = 03H           ;SET PC1 COMMAND
  0008                  RESPC4  = 08H           ;RESET PC4 COMMAND
  0009                  SETPC4  = 09H           ;SET PC4 COMMAND
  000A                  RESPC5  = 0AH           ;RESET PC5 COMMAND
  000B                  SETPC5  = 0BH           ;SET PC5 COMMAND
  000C                  RESPC6  = 0CH           ;RESET PC6 COMMAND
  000D                  SETPC6  = 0DH           ;SET PC6 COMMAND
  000C                  RESPC7  = 0CH           ;RESET PC7 COMMAND
  000D                  SETPC7  = 0DH           ;SET PC7 COMMAND
                        ;
  00C0                  PPMODE  = 0C0H          ;PARALLEL PORT MODE WORD
                                                ;(MODE 2/MODE 0 OUT)
                        ;
  0000"                         .LOC    .DATA.# ;LOCATE IN DATA AREA
                        ;
  0000"   02            NMB740::.BYTE   2       ;NUMBER OF IMS 740 SLAVES
                        ;
  0001"   00            CKT740::.BYTE   00H     ;IMS 740 CIRCUIT NUMBER
                        ;
  0002"                 PAT740::                ;IMS 740 SLAVE PORT ADDRESS TABLE
  0002"   40                    .BYTE   40H
  0003"   44                    .BYTE   44H
  0004"   48                    .BYTE   48H
  0005"   4C                    .BYTE   4CH
  0006"   50                    .BYTE   50H
  0007"   54                    .BYTE   54H
  0008"   58                    .BYTE   58H
  0009"   5C                    .BYTE   5CH
  000A"   60                    .BYTE   60H
  000B"   64                    .BYTE   64H
  000C"   68                    .BYTE   68H
  000D"   6C                    .BYTE   6CH
  000E"   70                    .BYTE   70H
  000F"   74                    .BYTE   74H
  0010"   78                    .BYTE   78H
  0011"   7C                    .BYTE   7CH
                        ;
  0012"                 SST740::                ;IMS 740 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            MAXLEN: .BYTE   0       ;MAXIMUM MESSAGE LENGTH
0023"   000F          CURSLV: .WORD   000FH   ;CURRENT SLAVE PROCESSOR NUMBER
                      ;
0025"   0000          RCVSPH: .WORD   0       ;RECEIVE MESSAGE SEMAPHORE
0027"   0027"         ..RCVH: .WORD   ..RCVH
0029"   0027"                 .WORD   ..RCVH
                      ;
002B"   000000000000  POLCNT: .BYTE   [16]0   ;POLL ROUTINE TICK COUNTS
                      ;
0000:04                       .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                      ;
0000:04 C9            CKTIN%::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 0060'              JZ      SNDMSG  ;IF SO, CONTINUE
0008'   C9                    RET             ;ELSE, DONE
                      ;
0009'   13            RCVMSG: INX     D       ;ADVANCE PAST LINK POINTERS
000A'   13                    INX     D
000B'   13                    INX     D
000C'   13                    INX     D
000D'   D5                    PUSH    D       ;SAVE MESSAGE BUFFER ADDRESS
000E'   1A                    LDAX    D       ;GET MAXIMUM MESSAGE LENGTH
000F'   32 0022"              STA     MAXLEN  ;SAVE MAXIMUM MESSAGE LENGTH
0012'   11 00F0'      ..RCVL: LXI     D,POL740  ;GET IMS 740 POLL ROUTINE
0015'   CD 0000:05            CALL    LNKPOL# ;LINK POLL ROUTINE ON POLL LIST
0018'   21 0025"              LXI     H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
001B'   CD 0000:06            CALL    WAIT#   ;WAIT FOR REQUEST
001E'   CD 0170'              CALL    RMCOM   ;DO COMMON SETUP
0021'   D5                    PUSH    D       ;SAVE SLAVE PROCESSOR NUMBER
0022'   200C                  JRNZ    ..RIP   ;IF REQUEST IN PROGRESS, CONTINUE
0024'   CBC6                  SET     0,M     ;ELSE, SET REQUEST IN PROGRESS FLA
0026'   0C                    INR     C       ;CALC CONTROL PORT
0027'   3E01                  MVI     A,SETPCO  ;SET PCO (INTERRUPT)
0029'   ED79                  OUTP    A       ;ASSERT INTERRUPT
```

```
  002B'    3E00                MVI     A,RESPCO  ;RESET PCO (INTERRUPT)
  002D'    ED79                OUTP    A         ;RELEASE INTERRUPT
  002F'    0D                  DCR     C         ;CALC PORT C ADDRESS
  0030'    CD 013A'    ..RIP:  CALL    INBYT     ;INPUT BYTE FROM SLAVE PROCESSOR
  0033'    D1                  POP     D         ;RESTORE SLAVE PROCESSOR NUMBER
  0034'    380D                JRC     ..IERR    ;IF ERROR, CONTINUE
  0036'    FE06                CPI     AACK      ;RESPONSE=ACK?
  0038'    280C                JRZ     ..RCV     ;IF SO, CONTINUE
  003A'    FE15                CPI     ANAK      ;RESPONSE=NACK?
  003C'    2005                JRNZ    ..IERR    ;IF NOT, CONTINUE
  003E'    CD 0168'            CALL    SETTC     ;ELSE, SET TICK COUNT
  0041'    18CF                JMPR    ..RCVL    ;CONTINUE
  0043'    E1          ..IERR: POP     H         ;RESTORE MESSAGE BUFFER ADDRESS
  0044'    1842                JMPR    ERRCOM    ;CONTINUE
  0046'    E1          ..RCV:  POP     H         ;RESTORE MESSAGE BUFFER ADDRESS
  0047'    0601                MVI     B,1       ;GET LENGTH OF MESSAGE LENGTH
  0049'    CD 0130'            CALL    RCV740    ;RECEIVE MESSAGE LENGTH
  004C'    383A                JRC     ERRCOM    ;IF ERROR, CONTINUE
  004E'    2B                  DCX     H         ;ELSE, BACK UP TO MESSAGE LENGTH
  004F'    3A 0022"            LDA     MAXLEN    ;GET MAXIMUM MESSAGE LENGTH
  0052'    BE                  CMP     M         ;MAXIMUM MESSAGE LENGTH EXCEEDED?
  0053'    3833                JRC     ERRCOM    ;IF SO, CONTINUE
  0055'    46                  MOV     B,M       ;ELSE, GET MESSAGE LENGTH
  0056'    23                  INX     H         ;RESTORE MESSAGE BUFFER ADDRESS
  0057'    05                  DCR     B         ;DECREMENT MESSAGE LENGTH
  0058'    282E                JRZ     ERRCOM    ;IF MESSAGE LENGTH=0, CONTINUE
  005A'    CD 0130'            CALL    RCV740    ;ELSE, RECEIVE REMAINDER OF MESSAG
  005D'    D0                  RNC               ;IF NO ERROR. DONE
  005E'    1828                JMPR    ERRCOM    ;ELSE, CONTINUE
                       ;
  0060'    EB          SNDMSG: XCHG              ;MESSAGE BUFFER ADDRESS TO HL-REG
  0061'    23                  INX     H         ;ADVANCE PAST LINK POINTERS
  0062'    23                  INX     H
  0063'    23                  INX     H
  0064'    23                  INX     H
  0065'    E5                  PUSH    H         ;SAVE MESSAGE BUFFER ADDRESS
  0066'    23                  INX     H         ;ADVANCE TO MESSAGE DESTINATION ID
  0067'    5E                  MOV     E,M       ;GET MESSAGE DESTINATION ID
  0068'    1600                MVI     D,0       ;DOUBLE LENGTH
  006A'    1D                  DCR     E         ;DECREMENT MESSAGE DESTINATION ID
  006B'    CD 0174'            CALL    SMCOM     ;DO COMMON SETUP
  006E'    E1                  POP     H         ;RESTORE MESSAGE BUFFER ADDRESS
  006F'    7E                  MOV     A,M       ;GET MESSAGE LENGTH
  0070'    B7                  ORA     A         ;MESSAGE LENGTH=0?
  0071'    2810                JRZ     ..X       ;IF SO, CONTINUE
  0073'    D5                  PUSH    D         ;ELSE, SAVE SLAVE PROCESSOR NUMBER
  0074'    46                  MOV     B,M       ;GET MESSAGE LENGTH
  0075'    05                  DCR     B         ;DECREMENT MESSAGE LENGTH
  0076'    C5                  PUSH    B         ;SAVE MESSAGE LENGTH
  0077'    0601                MVI     B,1       ;GET LENGTH OF MESSAGE LENGTH
  0079'    CD 0126'            CALL    SND740    ;SEND MESSAGE LENGTH TO SLAVE
  007C'    C1                  POP     B         ;RESTORE MESSAGE LENGTH
  007D'    D4 0126'            CNC     SND740    ;IF NO ERROR. SEND MESSAGE
  0080'    D1                  POP     D         ;RESTORE SLAVE PROCESSOR NUMBER
```

```
   0081'    3805                      JRC     ERRCOM  ;IF ERROR, CONTINUE
   0083'    CD 0168'         ..X:     CALL    SETTC   ;ELSE, SET TICK COUNT
   0086'    AF                        XRA     A       ;SET RETURN CODE=0
   0087'    C9                        RET             ;DONE
                             ;
   0088'    CD 0092'         ERRCOM:  CALL    RESSLV  ;RESET SLAVE PROCESSOR
   008B'    ED5B 0000:07              LDED    SID740# ;GET SLAVE PROCESSOR SOURCE ID
   008F'    3EFF                      MVI     A,0FFH  ;SET RETURN CODE=0FFH
   0091'    C9                        RET             ;DONE
                             ;
   0092'    3A 0001"         RESSLV:  LDA     CKT740  ;GET IMS 740 CIRCUIT NUMBER
   0095'    67                        MOV     H,A     ;IMS 740 CIRCUIT NUMBER TO H-REG
   0096'    2E00                      MVI     L,0     ;SET CIRCUIT NODE ADDRESS TO 0
   0098'    22 0000:08                SHLD    DID740# ;SET SLAVE PPOCESSOR DESTINATION I
   009B'    3A 0023"                  LDA     CURSLV  ;GET CURRENT SLAVE PROCESSOR NUMBE
   009E'    3C                        INR     A       ;INCREMENT SLAVE PROCESSOR NUMBER
   009F'    6F                        MOV     L,A     ;SLAVE PROCESSOR NUMBER TO L-REG
   00A0'    22 0000:07                SHLD    SID740# ;SET SLAVE PROCESSOR SOURCE ID
   00A3'    22 0000:09                SHLD    ORG740# ;SET SLAVE PROCESSOR ORIGIN
   00A6'    2A 0023"                  LHLD    CURSLV  ;GET CURRENT SLAVE PROCESSOR NUMBE
   00A9'    11 0012"                  LXI     D,SST740  ;GET SLAVE SUFFIX LETTER TABLE
   00AC'    19                        DAD     D       ;INDEX INTO SLAVE SUFFIX TABLE
   00AD'    7E                        MOV     A,M     ;GET SLAVE O/S SUFFIX LETTER
   00AE'    32 0000:0A                STA     SSL740# ;SET SLAVE SUFFIX LETTER
   00B1'    0C                        INR     C       ;CALC CONTROL PORT ADDRESS
   00B2'    3EC0                      MVI     A,PPMODE  ;GET PARALLEL PORT MODE WORD
   00B4'    ED79                      OUTP    A       ;INITIALIZE 8255
   00B6'    3E03                      MVI     A,SETPC1  ;SET PC1 (RESET)
   00B8'    ED79                      OUTP    A       ;ASSERT RESET
   00BA'    0D                        DCR     C       ;CALC PORT A ADDRESS
   00BB'    0D                        DCR     C
   00BC'    0D                        DCR     C
   00BD'    ED78                      INP     A       ;CLEAR PORT A INPUT
   00BF'    0C                        INR     C       ;CALC CONTROL PORT ADDRESS
   00C0'    0C                        INR     C
   00C1'    0C                        INR     C
   00C2'    CD 00EB'                  CALL    ..DLY   ;DELAY
   00C5'    3E02                      MVI     A,RESPC1  ;RESET PC1 (RESET)
   00C7'    ED79                      OUTP    A       ;RELEASE RESET
   00C9'    CD 00EB'                  CALL    ..DLY   ;DELAY
   00CC'    0D                        DCR     C       ;CALC PORT C ADDRESS
   00CD'    21 0000:0B                LXI     H,LAD740#  ;GET LOAD ADDRESS/LENGTH
   00D0'    0604                      MVI     B,4     ;GET LENGTH OF LOAD ADDRESS/LENGTH
   00D2'    CD 0126'                  CALL    SND740  ;SEND LOAD ADDRESS/LENGTH
   00D5'    D8                        RC              ;IF ERROR, DONE
   00D6'    ED5B 0000:0C              LDED    LEN740# ;GET LOAD LENGTH
   00DA'    43                        MOV     B,E     ;LSB OF LOAD LENGTH TO B-REG
   00DB'    7B                        MOV     A,E     ;GET LSB OF LOAD LENGTH
   00DC'    B7                        ORA     A       ;LSB OF LOAD LENGTH=0?
   00DD'    2001                      JRNZ    ..LDL   ;IF NOT, CONTINUE
   00DF'    15                        DCR     D       ;ELSE, DECREMENT MSB OF LOAD LENGT
   00E0'    D5               ..LDL:   PUSH    D       ;SAVE LOAD LENGTH
   00E1'    CD 0126'                  CALL    SND740  ;SEND UP TO 256 BYTES OF BOOT CODE
   00E4'    D1                        POP     D       ;RESTORE LOAD LENGTH
   00E5'    D8                          RC                  ;IF ERROR, DONE
   00E6'    15                          DCR       D         ;ELSE, DECREMENT LSB OF LOAD LENGT
   00E7'    F2 00E0'                    JP        ..LDL     ;IF MORE TO SEND, CONTINUE
   00EA'    C9                          RET                 ;ELSE, DONE
   00EB'    0600              ..DLY:    MVI       B,0       ;INITIALIZE DELAY COUNT
   00ED'    10FE              ..DLYL:   DJNZ      ..DLYL    ;DELAY
   00EF'    C9                          RET                 ;DONE
                              ;
   00F0'    0000              POL740:   .WORD     0         ;SUCCESSOR LINK POINTER
   00F2'    0000                        .WORD     0         ;PREDECESSOR LINK POINTER
                              ;
   00F4'    3A 0000"                    LDA       NMB740    ;GET NUMBER OF IMS 740 PROCESSORS
   00F7'    B7                          ORA       A         ;NUMBER OF IMS 740 PROCESSORS=0?
   00F8'    C8                          RZ                  ;IF SO, DONE
   00F9'    3D                          DCR       A         ;DECREMENT NUMBER OF PROCESSORS
   00FA'    E60F                        ANI       0FH       ;LIMIT NUMBER OF PROCESSORS TO 16
   00FC'    21 0023"                    LXI       H,CURSLV  ;GET CURRENT SLAVE NUMBER
   00FF'    34                          INR       M         ;INCREMENT CURRENT SLAVE NUMBER
   0100'    BE                          CMP       M         ;VALID SLAVE PROCESSOR NUMBER?
   0101'    3002                        JRNC      ..VSPN    ;IF SO, CONTINUE
   0103'    3600                        MVI       M,0       ;ELSE, SET SLAVE PROCESSOR NUMBER=
   0105'    CD 0170'          ..VSPN:   CALL      RMCOM     ;DO COMMON SETUP
   0108'    C0                          RNZ                 ;IF REQUEST IN PROGRESS SET, DONE
   0109'    ED78                        INP       A         ;ELSE, GET SLAVE PROCESSOR STATUS
   010B'    CB6F                        BIT       IBF,A     ;INPUT BUFFER FULL SET?
   010D'    2804                        JRZ       ..CTC     ;IF NOT, CONTINUE
   010F'    CBC6                        SET       0,M       ;ELSE, SET REQUEST IN PROGRESS FLA
   0111'    1807                        JMPR      ..SIG     ;CONTINUE
   0113'    CD 0182'          ..CTC:    CALL      GETCTC    ;GET CURRENT TICK COUNT
   0116'    96                          SUB       M         ;CALC ELAPSED NUMBER OF TICKS
   0117'    FE3C                        CPI       60        ;MINIMUM NUMBER OF TICKS ELAPSED?
   0119'    D8                          RC                  ;IF NOT, DONE
   011A'    21 00F0'          ..SIG:    LXI       H,POL740  ;ELSE, GET IMS 740 POLL ROUTINE
   011D'    CD 0000:0D                  CALL      UNLINK#   ;UNLINK POLL ROUTINE FROM POLL LIS
   0120'    21 0025"                    LXI       H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
   0123'    C3 0000:0E                  JMP       SIGNAL#   ;SIGNAL PROCESS AS READY
                              ;
   0126'    7E                SND740:   MOV       A,M       ;GET BYTE FROM BUFFER
   0127'    23                          INX       H
   0128'    CD 0151'                    CALL      OUTBYT    ;OUTPUT BYTE
   012B'    D8                          RC                  ;IF TIME-OUT, DONE
   012C'    10F8                        DJNZ      SND740    ;ELSE, CONTINUE
   012E'    AF                          XRA       A         ;SET RETURN CODE=0
   012F'    C9                          RET                 ;DONE
                              ;
   0130'    CD 013A'          RCV740:   CALL      INBYT     ;INPUT BYTE
   0133'    D8                          RC                  ;IF TIME-OUT, DONE
   0134'    77                          MOV       M,A       ;ELSE, STORE BYTE IN BUFFER
   0135'    23                          INX       H
   0136'    10F8                        DJNZ      RCV740    ;CONTINUE
   0138'    AF                          XRA       A         ;SET RETURN CODE=0
   0139'    C9                          RET                 ;DONE
                              ;
   013A'    1E00              INBYT:    MVI       E,0       ;INITIALIZE TIME OUT COUNTER
```

```
  013C'   ED50          ..WTL:  INP     D       ;GET SLAVE PROCESSOR STATUS
  013E'   CB6A                  BIT     IBF,D   ;INPUT BUFFER FULL SET?
  0140'   2007                  JRNZ    ..IBF   ;IF SO, CONTINUE
  0142'   1D                    DCR     E       ;ELSE, DECREMENT TIME OUT COUNT
  0143'   20F7                  JRNZ    ..WTL   ;CONTINUE
  0145'   3EFF                  MVI     A,OFFH  ;SET RETURN CODE=OFFH
  0147'   37                    STC             ;SET CARRY FLAG
  0148'   C9                    RET             ;DONE
  0149'   0D            ..IBF:  DCR     C       ;CALC PORT A ADDRESS
  014A'   0D                    DCR     C
  014B'   ED78                  INP     A       ;INPUT BYTE FROM PORT A
  014D'   0C                    INR     C       ;CALC PORT C ADDRESS
  014E'   0C                    INR     C
  014F'   B7                    ORA     A       ;CLEAR CARRY FLAG
  0150'   C9                    RET             ;DONE
                        ;
  0151'   1E00          OUTBYT: MVI     E,0     ;INITIALIZE TIME OUT COUNTER
  0153'   ED50          ..WTL:  INP     D       ;GET SLAVE PROCESSOR STATUS
  0155'   CB7A                  BIT     OBFN,D  ;OUTPUT BUFFER FULL (NOT) SET?
  0157'   2007                  JRNZ    ..NOBF  ;IF SO, CONTINUE
  0159'   1D                    DCR     E       ;ELSE, DECREMENT TIME OUT COUNT
  015A'   20F7                  JRNZ    ..WTL   ;CONTINUE
  015C'   3EFF                  MVI     A,OFFH  ;SET RETURN CODE=OFFH
  015E'   37                    STC             ;SET CARRY FLAG
  015F'   C9                    RET             ;DONE
  0160'   0D            ..NOBF: DCR     C       ;CALC PORT A ADDRESS
  0161'   0D                    DCR     C
  0162'   ED79                  OUTP    A       ;OUTPUT BYTE TO PORT A
  0164'   0C                    INR     C       ;CALC PORT C ADDRESS
  0165'   0C                    INR     C
  0166'   B7                    ORA     A       ;CLEAR CARRY FLAG
  0167'   C9                    RET             ;DONE
                        ;
  0168'   CD 0174'      SETTC:  CALL    SMCOM   ;GET TICK COUNT ADDRESS
  016B'   CD 0182'              CALL    GETCTC  ;GET CURRENT TICK COUNT
  016E'   77                    MOV     M,A     ;SET TICK COUNT
  016F'   C9                    RET             ;DONE
                        ;
  0170'   ED5B 0023"    RMCOM:  LDED    CURSLV  ;GET CURRENT SLAVE PROCESSOR NUMBE
                        ;
  0174'   21 0002"      SMCOM:  LXI     H,PAT740  ;GET IMS 740 PORT ADDRESS TABLE
  0177'   19                    DAD     D       ;CALC IMS 740 PORT ADDRESS
  0178'   4E                    MOV     C,M     ;DATA PORT ADDRESS TO C-REG
  0179'   0C                    INR     C       ;CALC PORT C ADDRESS
  017A'   0C                    INR     C
  017B'   21 002B"              LXI     H,POLCNT  ;GET POLL ROUTINE TICK COUNTS
  017E'   19                    DAD     D       ;INDEX INTO TICK COUNTS
  017F'   CB46                  BIT     0,M     ;REQUEST IN PROGRESS FLAG SET?
  0181'   C9                    RET             ;DONE
                        ;
  0182'   3A 0000:0F    GETCTC: LDA     TICCNT# ;GET CURRENT TICK COUNT
  0185'   E6FE                  ANI     OFEH    ;STRIP BIT 0
  0187'   C9                    RET             ;DONE
                        ;
                        .PRGEND
```

```
                          ;
                          ; ******************************************************
                          ; *                                                    *
                          ; *  At the start of a download of an IMS 740 slave    *
                          ; *  processor board, the master circuit driver MCD740 *
                          ; *  sends this loader to the slave.  It is loaded into *
                          ; *  slave memory by the slave ROM module SPB740.  This *
                          ; *  loader module handles the remainder of the down-  *
                          ; *  loading task by making standard TurboDOS network  *
                          ; *  download requests to load the full slave operating *
                          ; *  system from a file OSSLAVE.SYS on the master's    *
                          ; *  disk.                                             *
                          ; *                                                    *
                          ; ******************************************************
                          ;
                          ; COPYRIGHT (C) 1982. SOFTWARE 2000. INC.
                          ;
                          ; AUTHORS: RONALD E. RAIKES
                          ;          MICHAEL D. BUSCH
                          ;
                          ; VERSION: EXAMPLE
                          ;
                          .IDENT  LOD740          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
  0000                    PIOVEC  = 00H           ;PIO INTERRUPT VECTOR ADDRESS
                          ;
  0020                    PORTAD  = 20H           ;8255 PORT A DATA REGISTER
  0024                    PIOADR  = 24H           ;PIO PORT A DATA REGISTER
  0025                    PIOBDR  = 25H           ;PIO PORT B DATA REGISTER
  0026                    PIOACR  = 26H           ;PIO PORT A CONTROL REGISTER
  0027                    PIOBCR  = 27H           ;PIO PORT B CONTROL REGISTER
                          ;
  0004                    PORTC0  = 4             ;PORT C DATA BIT 0 (INT)
  0005                    PORTC5  = 5             ;PORT C DATA BIT 5 (IBF)
  0006                    PORTC7  = 6             ;PORT C DATA BIT 7 (OBF)
  0007                    RAMPER  = 7             ;RAM PARITY ERROR
                          ;
  0006                    ROMENA  = 6             ;ROM ENABLE BIT
  0007                    PERRES  = 7             ;PARITY ERROR RESET
                          ;
  0000'                           .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                          ;
  0000'   0A00            LAD740::.WORD   0A00H   ;SLAVE PROCESSOR BOOT LOAD ADDRESS
  0002'   00CF            LEN740::.WORD   SPBLEN  ;SLAVE PROCESSOR BOOT LOAD LENGTH
                          ;
                          .DEFINE RELOC[ADDR]=[(ADDR-SPB740)+0A00H] ;RELOCATION MACR
                          ;
  0004'   F3              SPB740: DI              ;DISABLE INTERRUPTS
  0005'   31 0900                 LXI     SP,900H ;INITIALIZE STACK POINTER
  0008'   3E09                    MVI     A,09H   ;GET INTERRUPT VECTOR PAGE
  000A'   ED47                    STAI            ;SET INTERRUPT PAGE REGISTER
  000C'   ED5E                    IM2             ;SET INTERRUPT MODE 2
```

```
000E'   21 0A8E                 LXI     H,RELOC[PIOISR]  ;GET PIO INT SERVICE ADDR
0011'   22 0900                 SHLD    0900H   ;SET INTERRUPT VECTOR ADDRESS
0014'   3E00                    MVI     A,PIOVEC  ;GET PIO INTERRUPT SERVICE VECTO
0016'   D326                    OUT     PIOACR  ;OUTPUT INTERRUPT SERVICE VECTOR
0018'   3ECF                    MVI     A,0CFH  ;GET PIO A MODE WORD (MODE 3)
001A'   D326                    OUT     PIOACR  ;OUTPUT PIO A MODE WORD
001C'   3EFF                    MVI     A,0FFH  ;GET PIO A MODE 3 MODE WORD
001E'   D326                    OUT     PIOACR  ;OUTPUT PIO A MODE 3 MODE WORD
0020'   3EB7                    MVI     A,0B7H  ;GET PIO A INTERRUPT CONTROL WORD
0022'   D326                    OUT     PIOACR  ;OUTPUT INTERRUPT CONTROL WORD
0024'   3EEF                    MVI     A,0EFH  ;GET PIO A INTERRUPT MASK
0026'   D326                    OUT     PIOACR  ;OUTPUT INTERRUPT MASK
0028'   3EF0                    MVI     A,0F0H  ;GET PIO B DATA DATA WORD
002A'   D325                    OUT     PIOBDR  ;OUTPUT PIO B DATA WORD
002C'   3E0F                    MVI     A,0FH   ;GET PIO B MODE WORD (MODE 0)
002E'   D327                    OUT     PIOBCR  ;OUTPUT PIO B MODE WORD
0030'   DB20                    IN      PORTAD  ;CLEAR INPUT PORT A
0032'   FB                      EI              ;ENABLE INTERRUPTS
0033'   21 0000                 LXI     H,0     ;INITIALIZE MEMORY PARITY
0036'   11 0000                 LXI     D,0
0039'   01 0000                 LXI     B,0
003C'   EDB0                    LDIR
003E'   3E30                    MVI     A,30H   ;GET PIO B DATA BYTE
0040'   D325                    OUT     PIOBDR  ;RELEASE PARITY ERROR RESET
0042'   3E06            ..SPBL: MVI     A,AACK  ;GET ACK
0044'   CD 0AB6                 CALL    RELOC[OUTBYT]  ;OUTPUT ACK
0047'   21 0AC3                 LXI     H,RELOC[REQMSG]  ;GET REQUEST MESSAGE
004A'   46                      MOV     B,M     ;GET MESSAGE LENGTH
004B'   CD 0A9D                 CALL    RELOC[SND740]  ;SEND MESSAGE
004E'   21 0ACF                 LXI     H,RELOC[REPMSG]  ;GET REPLY MESSAGE BUFFER
0051'   0601                    MVI     B,1     ;GET LENGTH OF MESSAGE LENGTH
0053'   CD 0AA5                 CALL    RELOC[RCV740]  ;RECEIVE MESSAGE LENGTH
0056'   2B                      DCX     H       ;BACK UP TO MESSAGE LENGTH
0057'   46                      MOV     B,M     ;GET MESSAGE LENGTH
0058'   23                      INX     H       ;RESTORE MESSAGE BUFFER ADDRESS
0059'   05                      DCR     B       ;DECREMENT MESSAGE LENGTH
005A'   CD 0AA5                 CALL    RELOC[RCV740]  ;RECEIVE MESSAGE
005D'   3A 0ACF                 LDA     RELOC[REPMSG]  ;GET REPLY MESSAGE LENGTH
0060'   FE0C                    CPI     MSGHL+1 ;MESSAGE LENGTH=HEADER LENGTH+1?
0062'   2812                    JRZ     ..EOF   ;IF SO, CONTINUE
0064'   21 0ADA                 LXI     H,RELOC[REPMSG+MSGHL]  ;GET DATA ADDRESS
0067'   ED5B 0AC1               LDED    RELOC[DMADDR]  ;GET DMA ADDRESS
006B'   01 0080                 LXI     B,128   ;GET DATA RECORD LENGTH
006E'   EDB0                    LDIR            ;MOVE DOWNLOAD RECORD INTO DMA ADD
0070'   ED53 0AC1               SDED    RELOC[DMADDR]  ;UPDATE DMA ADDRESS
0074'   18CC                    JMPR    ..SPBL  ;CONTINUE
0076'   2B              ..EOF:  DCX     H       ;BACK UP TO O/S ID
0077'   7E                      MOV     A,M     ;GET SYSTEM DISK
0078'   21 0B5A                 LXI     H,RELOC[OSLOAD]  ;GET SLAVE PROCESSOR O/S
007B'   5E                      MOV     E,M     ;GET O/S LOAD ADDRESS
007C'   23                      INX     H
007D'   56                      MOV     D,M
007E'   23                      INX     H
007F'   4E                      MOV     C,M     ;GET SLAVE PROCESSOR O/S LENGTH
```

```
LOD740 - TURBODOS OPERATING SYSTEM INTERMEDIATE BOOTSTAP (IMS 740)
COPYRIGHT (C) 1982, SOFTWARE 2000, INC.


  0080'   23                      INX     H
  0081'   46                      MOV     B,M
  0082'   09                      DAD     B       ;CALC LAST BYTE OF O/S
  0083'   EB                      XCHG            ;O/S LOAD ADDRESS TO HL-REG
  0084'   09                      DAD     B       ;CALC LAST BYTE LOAD ADDRESS
  0085'   2B                      DCX     H
  0086'   EB                      XCHG            ;HL=END OF O/S-HL=LAST LOAD ADDRES
  0087'   EDB8                    LDDR            ;MOVE O/S INTO LOAD ADDRESS
  0089'   13                      INX     D       ;ADVANCE TO O/S ENTRYPOINT
  008A'   2A 0AC7                 LHLD    RELOC[SID740]  ;GET IMS 740 SOURCE ID
  008D'   22 0080                 SHLD    TBUF    ;STORE ID IN DEFAULT BUFFER
  0090'   EB                      XCHG            ;O/S ENTRYPOINT TO HL-REG
  0091'   E9                      PCHL            ;TRANSFER TO SLAVE PROCESSOR O/S
                        ;
  0092'   F5            PIOISR:   PUSH    PSW     ;SAVE AF-REG
  0093'   DB24                    IN      PIOADR  ;GET PIO A DATA REGISTER
  0095'   CB6F                    BIT     PORTC5,A  ;PORT C BIT 5 SET? (IBF)
  0097'   2004                    JRNZ    ..X     ;IF SO, CONITNUE
  0099'   3E15                    MVI     A,ANAK  ;ELSE, GET NACK
  009B'   D320                    OUT     PORTAD  ;OUTPUT NACK TO PORT A
  009D'   F1            ..X:      POP     PSW     ;RESTORE AF-REG
  009E'   FB                      EI              ;ENABLE INTERRUPTS
  009F'   ED4D                    RETI            ;DONE
                        ;
  00A1'   7E            SND740:   MOV     A,M     ;GET BYTE FROM MESSAGE
  00A2'   23                      INX     H
  00A3'   CD 0AB6                 CALL    RELOC[OUTBYT]  ;OUTPUT BYTE
  00A6'   10F9                    DJNZ    SND740  ;CONTINUE
  00A8'   C9                      RET             ;DONE
                        ;
  00A9'   CD 0AAD       RCV740:   CALL    RELOC[INBYT]  ;INPUT BYTE
  00AC'   77                      MOV     M,A     ;SAVE BYTE IN MESSAGE
  00AD'   23                      INX     H
  00AE'   10F9                    DJNZ    RCV740  ;CONTINUE
  00B0'   C9                      RET             ;DONE
                        ;
  00B1'   DB24          INBYT:    IN      PIOADR  ;GET PIO A DATA REGISTER
  00B3'   CB77                    BIT     PORTC7,A  ;PORT C BIT 7 SET? (OBF)
  00B5'   20FA                    JRNZ    INBYT   ;IF NOT, WAIT
  00B7'   DB20                    IN      PORTAD  ;INPUT BYTE FROM PORT A
  00B9'   C9                      RET             ;DONE
                        ;
  00BA'   4F            OUTBYT:   MOV     C,A     ;SAVE OUTPUT BYTE IN C-REG
  00BB'   DB24          ..WTL:    IN      PIOADR  ;GET PIO A DATA REGISTER
  00BD'   CB6F                    BIT     PORTC5,A  ;PORT C BIT 5 SET? (IBF)
  00BF'   20FA                    JRNZ    ..WTL   ;IF SO, WAIT
  00C1'   79                      MOV     A,C     ;GET OUTPUT BYTE
  00C2'   D320                    OUT     PORTAD  ;OUTPUT BYTE TO PORT A
  00C4'   C9                      RET             ;DONE
                        ;
  00C5'   0B5A          DMADDR:   .WORD   RELOC[OSLOAD]  ;DMA ADDRESS
                        ;
  00C7'                 REQMSG:                   ;DOWNLOAD REQUEST MESSAGE
  00C7'                 MSGHDR:                   ;MESSAGE HEADER
```

LOD740 - TURBODOS OPERATING SYSTEM INTERMEDIATE BOOTSTAP (IMS 740)

```
  00C7'   0C            MSGLEN: .BYTE   MSGHBL  ;MESSAGE LENGTH
  00C8'                 DID740::                ;SLAVE PROCESSOR DESTINATION ID
  00C8'   0000          MSGDID: .WORD   0       ;MESSAGE DESTINATION ID
  00CA'   00            MSGPID: .BYTE   0       ;MESSAGE PROCESS ID
  00CB'                 SID740::                ;SLAVE PROCESSOR SOURCE ID
  00CB'   0000          MSGSID: .WORD   0       ;MESSAGE SOURCE ID
  00CD'                 ORG740::                ;SLAVE PROCESSOR ORIGIN
  00CD'   000000        MSGORG: .BYTE   [3]0    ;MESSAGE ORIGIN
  00D0'   00            MSGLVL: .BYTE   0       ;MESSAGE LEVEL
  00D1'   00            MSGFCD: .BYTE   0       ;MESSAGE FORMAT CODE
                        ;
  000B                  MSGHL   = .-MSGHDR      ;MESSAGE HEADER LENGTH
                        ;
  00D2'   20            SSL740::.BYTE   ASP     ;SLAVE PROCESSOR O/S SUFFIX LETTER
                        ;
  000C                  MSGHBL  = .-MSGHDR      ;MESSAGE HEADER/BUFFER LENGTH
                        ;
  00CF                  SPBLEN  = .-SPB740      ;SLAVE PROCESSOR BOOT CODE LENGTH
                        ;
  00D3'                 REPMSG: .BLKB   MSGHL+128  ;REPLY MESSAGE BUFFER
                        ;
  015E'                 OSLOAD  = .             ;O/S LOAD ADDRESS
                        ;
                        .END
```

```
                              ;
                              ; ********************************************************
                              ; *                                                      *
                              ; *  This is a trivial network circuit driver which      *
                              ; *  may be used for master-to-master networking over    *
                              ; *  an ordinary RS232 serial channel.  It performs      *
                              ; *  absolutely no handshaking, error checking, or       *
                              ; *  error recovery, and consequently is suitable only   *
                              ; *  for very short links which are considered to be     *
                              ; *  completely error-free.  It could easily be en-      *
                              ; *  hanced to include such features, however.          *
                              ; *                                                      *
                              ; ********************************************************
                              ;
                              ;
                              ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                              ;
                              ; AUTHORS: RONALD E. RAIKES
                              ;          MICHAEL D. BUSCH
                              ;
                              ;
                              ; VERSION: EXAMPLE
                              ;
                              .IDENT  CKTSER          ;MODULE ID
                              ;
                              .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                              ;
0000"                                 .LOC    .DATA.# ;LOCATE IN DATA AREA
                              ;
0000"   0F                    CKTSBR::.BYTE   0FH     ;NETWORK BAUD RATE
0001"   01                    CKTSCH::.BYTE   1       ;NETWORK SERIAL CHANNEL
                              ;
0000:04                               .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                              ;
0000:04 1E03                  CKTIN%::MVI     E,3     ;E = SET BAUD RATE FCN
0002:04 ED4B 0000"                    LBCD    CKTSBR  ;C = BAUD RATE, B = CHANNEL NUMBER
0006:04 C3 0000:05                    JMP     SERIAL# ;SET SERIAL CHANNEL BAUD RATE
                              ;
0000'                                 .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                              ;
0000'   79                    CKTDR%::MOV     A,C     ;GET FUNCTION NUMBER
0001'   B7                            ORA     A       ;FUNCTION NUMBER=0?
0002'   2804                          JRZ     RCVMSG  ;IF SO, CONTINUE
0004'   3D                            DCR     A       ;FUNCTION NUMBER=1?
0005'   282E                          JRZ     SNDMSG  ;IF SO, CONTINUE
0007'   C9                            RET             ;ELSE, DONE
                              ;
0008'   EB                    RCVMSG: XCHG            ;ADDRESS OF BUFFER IN HL-REGISTERS
0009'   0603                  ..L1:   MVI     B,3     ;LOOK FOR THREE SYNC CHARACTERS
000B'   CD 0027'              ..L2:   CALL    RCVA    ;RECEIVE SYNC BYTE (?)
000E'   FE16                          CPI     ASYN    ;SYNC?
0010'   20F7                          JRNZ    ..L1    ;NO, RESET COUNT
0012'   10F7                          DJNZ    ..L2    ;ELSE, CONTINUE FOR COUNT
0014'   23                            INX     H       ;ADVANCE PAST BUFFER LINKAGE
0015'   23                            INX     H
0016'   23                            INX     H
```

```
 0017'   23                       INX     H
 0018'   CD 0027'                 CALL    RCVA      ;RECEIVE MESSAGE LENGTH BYTE
 001B'   47                       MOV     B,A       ;B = ACTUAL MESSAGE LENGTH
 001C'   1803                     JMPR    ..L4      ;JOIN COMMON
 001E'   CD 0027'        ..L3:    CALL    RCVA      ;RECEIVE MESSAGE BYTE
 0021'   77              ..L4:    MOV     M,A       ;STORE IN BUFFER
 0022'   23                       INX     H         ;INCREMENT BUFFER POINTER
 0023'   10F9                     DJNZ    ..L3      ;CONTINUE FOR THE COUNT
 0025'   AF                       XRA     A         ;SET RETURN CODE = SUCCESS
 0026'   C9                       RET               ;DONE
                         ;
 0027'   E5              RCVA:    PUSH    H         ;SAVE REGISTERS
 0028'   C5                       PUSH    B
 0029'   3A 0001"                 LDA     CKTSCH    ;B = GET CHANNEL NUMBER
 002C'   47                       MOV     B,A
 002D'   1E01                     MVI     E,1       ;E = SERIAL IN FCN
 002F'   CD 0000:05               CALL    SERIAL#   ;INPUT CHARACTER
 0032'   C1                       POP     B         ;RESTORE REGISTERS
 0033'   E1                       POP     H
 0034'   C9                       RET               ;DONE
                         ;
 0035'   EB              SNDMSG:  XCHG              ;ADDRESS OF BUFFER IN HL-REGISTERS
 0036'   01 0316                  LXI     B,3<8!ASYN  ;SEND THREE SYNC CHARACTERS FI
                         T
 0039'   CD 004C'        ..L1:    CALL    SNDC
 003C'   10FB                     DJNZ    ..L1
 003E'   23                       INX     H         ;ADVANCE PAST BUFFER LINKAGE
 003F'   23                       INX     H
 0040'   23                       INX     H
 0041'   23                       INX     H
 0042'   46                       MOV     B,M       ;B = MESSAGE LENGTH
 0043'   4E              ..L2:    MOV     C,M       ;GET NEXT MESSAGE BYTE
 0044'   23                       INX     H         ;ADVANCE BUFFER POINTER
 0045'   CD 004C'                 CALL    SNDC      ;SEND MESSAGE BYTE
 0048'   10F9                     DJNZ    ..L2      ;CONTINUE FOR THE COUNT
 004A'   AF                       XRA     A         ;SET RETURN CODE = SUCCESS
 004B'   C9                       RET               ;DONE
                         ;
 004C'   E5              SNDC:    PUSH    H         ;SAVE REGISTERS
 004D'   C5                       PUSH    B
 004E'   3A 0001"                 LDA     CKTSCH    ;B = GET CHANNEL NUMBER
 0051'   47                       MOV     B,A
 0052'   1E02                     MVI     E,2       ;E = SERIAL OUT FCN
 0054'   CD 0000:05               CALL    SERIAL#   ;OUTPUT CHARACTER
 0057'   C1                       POP     B         ;RESTORE REGISTERS
 0058'   E1                       POP     H
 0059'   C9                       RET               ;DONE
                         ;
                         .END
```

```
SPD442 - TURBODOS OPERATING SYSTEM SERIAL/PARALLEL I/O DRIVER (IMS 442)
COPYRIGHT (C) 1982. SOFTWARE 2000. INC.

                          ;
                          ; ***********************************************************
                          ; *                                                         *
                          ; *  This module handles all serial and parallel I/O        *
                          ; *  for the IMS 442 I/O board.  Serial channels 0 and      *
                          ; *  1 are handled directly by this module, and cor-        *
                          ; *  respond to the two serial ports on the IMS 442         *
                          ; *  board.  Serial channels 2, 3, 4 and 5 are passed       *
                          ; *  to another module, SER480, and correspond to the       *
                          ; *  four serial ports on the IMS 480 four-channel          *
                          ; *  serial board.  Thus, six serial channels can be        *
                          ; *  supported if both boards are installed.                *
                          ; *                                                         *
                          ; *  Serial input is handled via interrupts and buf-        *
                          ; *  fered in a separate circular buffer for each chan-     *
                          ; *  nel.  Serial output is handled on a polled basis       *
                          ; *  and is not buffered.                                   *
                          ; *                                                         *
                          ; ***********************************************************
                          ;
                          ; COPYRIGHT (C) 1982. SOFTWARE 2000. INC.
                          ;
                          ; AUTHORS: RONALD E. RAIKES
                          ;          MICHAEL D. BUSCH
                          ;
                          ; VERSION: EXAMPLE
                          ;
                          .IDENT  SPD442          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
  0010                    IOBASE  = 10H           ;SERIAL/PARALLEL I/O PORT BASE
                          ;
  0010                    S0CTRL  = IOBASE+00H    ;SERIAL 0 CONTROL/STATUS REGISTER
  0011                    S0DATA  = IOBASE+01H    ;SERIAL 0 DATA REGISTER
  0012                    S1CTRL  = IOBASE+02H    ;SERIAL 1 CONTROL/STATUS REGISTER
  0013                    S1DATA  = IOBASE+03H    ;SERIAL 1 DATA REGISTER
  0014                    TIM0    = IOBASE+04H    ;TIMER 0 DATA REGISTER
  0015                    TIM1    = IOBASE+05H    ;TIMER 1 DATA REGISTER
  0016                    TIM2    = IOBASE+06H    ;TIMER 2 DATA REGISTER
  0017                    TIMCTL  = IOBASE+07H    ;TIMER CONTROL REGISTER
  0018                    SINTE   = IOBASE+08H    ;SERIAL INTERRUPT ENABLE REGISTER
  0019                    T2RES   = IOBASE+09H    ;TIMER 2 INTERRUPT RESET
  001C                    P0DATA  = IOBASE+0CH    ;PARALLEL 0 DATA REGISTER
  001D                    P1DATA  = IOBASE+0DH    ;PARALLEL 1 DATA REGISTER
  001E                    P2DATA  = IOBASE+0EH    ;PARALLEL 2 DATA REGISTER
  001F                    PPCTL   = IOBASE+0FH    ;PARALLEL PORT CONTROL REGISTER
                          ;
  0000                    RDA     = 0             ;RECEIVED DATA AVAILABLE BIT
  0001                    TBE     = 1             ;TRANSMIT BUFFER EMPTY BIT
  0007                    CTSN    = 7             ;CLEAR TO SEND (NOT) BIT
                          ;
  0000                    ROMDIS  = 0             ;ROM DISABLE BIT
  0001                    RTCENA  = 1             ;REAL TIME CLOCK ENABLE BIT
  0002                     S1TXIE  = 2              ;SERIAL 1 TX INTERRUPT ENABLE BIT
  0003                     S1RXIE  = 3              ;SERIAL 1 RX INTERRUPT ENABLE BIT
  0004                     S1RTSN  = 4              ;SERIAL 1 REQ TO SEND (NOT) BIT
  0005                     S0TXIE  = 5              ;SERIAL 0 TX INTERRUPT ENABLE BIT
  0006                     S0RXIE  = 6              ;SERIAL 0 RX INTERRUPT ENABLE BIT
  0007                     S0RTSN  = 7              ;SERIAL 0 REQ TO SEND (NOT) BIT
                           ;
  0036                     T0CMD   = 36H            ;TIMER 0 COMMAND
  0076                     T1CMD   = 76H            ;TIMER 1 COMMAND
  00B6                     T2CMD   = 0B6H           ;TIMER 2 COMMAND
                           ;
  0089                     PPMODE  = 89H            ;PARALLEL PORT MODE WORD
  0019                     SPMODE  = 19H            ;SERIAL PORT MODE WORD
                                                    ;PARITY INHIBIT/1 STOP BIT/8 BITS
                           ;
  0000"                            .LOC    .DATA.#  ;LOCATE IN DATA AREA
                           ;
  0000"   0040             S0IBSZ::.WORD   64       ;SERIAL 0 INPUT BUFFER SIZE
  0002"   0000             S0IBUF: .WORD   0        ;SERIAL 0 INPUT BUFFER ADDRESS
  0004"   0000             S0IPTR: .WORD   0        ;SERIAL 0 INPUT POINTER
  0006"   0000             S0OPTR: .WORD   0        ;SERIAL 0 OUTPUT POINTER
  0008"   0000             S0ICNT: .WORD   0        ;SERIAL 0 INPUT COUNT
  000A"   00               S0OCHR: .BYTE   0        ;SERIAL 0 OUTPUT CHARACTER
  000B"   00               S0BR:   .BYTE   0        ;SERIAL 0 BAUD RATE CODE
                           ;
  000C"                    S0ISPH:                  ;SERIAL 0 INPUT SEMAPHORE
  000C"   0000                     .WORD   0        ;SEMAPHORE COUNT
  000E"   000E"            ..S0IH: .WORD   ..S0IH   ;SEMAPHORE P/D HEAD
  0010"   000E"                    .WORD   ..S0IH
                           ;
                                                    ;SERIAL 0 OUTPUT SEMAPHORE
  0012"   0000             S0OSPH: .WORD   0        ;SEMAPHORE COUNT
  0014"   0014"            ..S0OH: .WORD   ..S0OH   ;SEMAPHORE P/D HEAD
  0016"   0014"                    .WORD   ..S0OH
                           ;
                                                    ;SERIAL 0 OUTPUT SEMAPHORE
  0018"   0001             S0XSPH: .WORD   1        ;SEMAPHORE COUNT
  001A"   001A"            ..S0XH: .WORD   ..S0XH   ;SEMAPHORE P/D HEAD
  001C"   001A"                    .WORD   ..S0XH
                           ;
  001E"   0010             S1IBSZ::.WORD   16       ;SERIAL 1 INPUT BUFFER SIZE
  0020"   0000             S1IBUF: .WORD   0        ;SERIAL 1 INPUT BUFFER ADDRESS
  0022"   0000             S1IPTR: .WORD   0        ;SERIAL 1 INPUT POINTER
  0024"   0000             S1OPTR: .WORD   0        ;SERIAL 1 OUTPUT POINTER
  0026"   0000             S1ICNT: .WORD   0        ;SERIAL 1 INPUT COUNT
  0028"   00               S1OCHR: .BYTE   0        ;SERIAL 1 OUTPUT CHARACTER
  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
                        ;
                                                  ;SERIAL 1 OUTPUT SEMAPHORE
0036"    0001           S1XSPH: .WORD   1         ;SEMAPHORE COUNT
0038"    0038"          ..S1XH: .WORD   ..S1XH    ;SEMAPHORE P/D HEAD
003A"    0038"                  .WORD   ..S1XH
                        ;
003C"    49             INTMSK::.BYTE   1<ROMDIS!1<S0RXIE!1<S1RXIE  ;INTERRUPT MAS
                        ;
0000:04                         .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                        ;
0000:04 3E89            SPINIT::MVI     A,PPMODE  ;INITIALIZE 8255
0002:04 D31F                    OUT     PPCTL
0004:04 3EFF                    MVI     A,0FFH    ;CLEAR PARALLEL PORTS
0006:04 D31C                    OUT     P0DATA
0008:04 D31D                    OUT     P1DATA
000A:04 3E19                    MVI     A,SPMODE  ;INITIALIZE UARTS
000C:04 D310                    OUT     S0CTRL
000E:04 D312                    OUT     S1CTRL
0010:04 3EC3                    MVI     A,JMP     ;SET UP SERIAL INTERRUPT VECTOR
0012:04 32 0018                 STA     3*8
0015:04 21 0131'                LXI     H,SERISR
0018:04 22 0019                 SHLD    (3*8)+1
001B:04 3A 003C"                LDA     INTMSK    ;GET INTERRUPT MASK
001E:04 D318                    OUT     SINTE     ;ENABLE INTERRUPT MASKS
0020:04 2A 0000"                LHLD    S0IBSZ    ;GET SERIAL 0 INPUT BUFFER SIZE
0023:04 CD 0000:05              CALL    ALLOC#    ;ALLOCATE PACKET FOR SERIAL BUFFER
0026:04 22 0002"                SHLD    S0IBUF    ;SAVE SERIAL 0 INPUT BUFFER ADDRES
0029:04 22 0004"                SHLD    S0IPTR    ;SET SERIAL 0 INPUT POINTER
002C:04 22 0006"                SHLD    S0OPTR    ;SET SERIAL 0 OUTPUT POINTER
002F:04 2A 001E"                LHLD    S1IBSZ    ;GET SERIAL 1 INPUT BUFFER SIZE
0032:04 CD 0000:05              CALL    ALLOC#    ;ALLOCATE PACKET FOR SERIAL BUFFER
0035:04 22 0020"                SHLD    S1IBUF    ;SAVE SERIAL 1 INPUT BUFFER ADDRES
0038:04 22 0022"                SHLD    S1IPTR    ;SET SERIAL 1 INPUT POINTER
003B:04 22 0024"                SHLD    S1OPTR    ;SET SERIAL 1 OUTPUT POINTER
003E:04 C3 0000:06              JMP     NIT480#   ;INITIALIZE IMS 480 SERIAL PORTS
                        ;
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'    2818                   JRZ     SERST     ;IF SO, CONTINUE
0004'    3D                     DCR     A         ;FUNCTION NUMBER=1?
0005'    2834                   JRZ     SERIN     ;IF SO, CONTINUE
0007'    3D                     DCR     A         ;FUNCTION NUMBER=2?
0008'    CA 00A4'               JZ      SEROUT    ;IF SO, CONTINUE
000B'    3D                     DCR     A         ;FUNCTION NUMBER=3?
000C'    CA 0200'               JZ      SERSBR    ;IF SO, CONTINUE
000F'    3D                     DCR     A         ;FUNCTION NUMBER=4?
0010'    CA 0235'               JZ      SERRBR    ;IF SO, CONTINUE
0013'    3D                     DCR     A         ;FUNCTION NUMBER=5?
```

```
   0014'    CA 0246'                JZ      SERSMC  ;IF SO, CONTINUE
   0017'    3D                      DCR     A       ;FUNCTION NUMBER=6?
   0018'    CA 026A'                JZ      SERRMC  ;IF SO, CONTINUE
   001B'    C9                      RET             ;ELSE, DONE
                            ;
   001C'    78              SERST:  MOV     A,B     ;GET CHANNEL NUMBER
   001D'    FE02                    CPI     2       ;CHANNEL NUMBER=0/1?
   001F'    D2 0000:07              JNC     ST480#  ;IF NOT, CONTINUE
                            ;
   0022'    ED5B 0008"      ST442:  LDED    S0ICNT  ;GET SERIAL 0 INPUT BUFFER COUNT
   0026'    2A 0006"                LHLD    S0OPTR  ;GET SERIAL 0 OUTPUT POINTER
   0029'    78                      MOV     A,B     ;GET CHANNEL NUMBER
   002A'    B7                      ORA     A       ;CHANNEL NUMBER=0
   002B'    2807                    JRZ     ..STC   ;IF SO, CONTINUE
   002D'    ED5B 0026"              LDED    S1ICNT  ;GET SERIAL 1 INPUT BUFFER COUNT
   0031'    2A 0024"                LHLD    S1OPTR  ;GET SERIAL 1 OUTPUT POINTER
   0034'    7A              ..STC:  MOV     A,D
   0035'    B3                      ORA     E       ;SERIAL INPUT BUFFER COUNT=0?
   0036'    C8                      RZ              ;IF SO, DONE
   0037'    3EFF                    MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
   0039'    4E                      MOV     C,M     ;GET SERIAL INPUT CHARACTER
   003A'    C9                      RET             ;DONE
                            ;
   003B'    78              SERIN:  MOV     A,B     ;GET CHANNEL NUMBER
   003C'    FE02                    CPI     2       ;CHANNEL NUMBER=0/1?
   003E'    D2 0000:08              JNC     IN480#  ;IF NOT, CONTINUE
   0041'    CD 0022'        ..SINL: CALL    ST442   ;ELSE, GET SERIAL STATUS
   0044'    B7                      ORA     A       ;CHARACTER AVAILABLE?
   0045'    200F                    JRNZ    ..SIN   ;IF SO, CONTINUE
   0047'    78                      MOV     A,B     ;ELSE, GET CHANNEL NUMBER
   0048'    21 000C"                LXI     H,S0ISPH  ;GET SERIAL 0 INPUT SEMAPHORE
   004B'    B7                      ORA     A       ;CHANNEL NUMBER=0?
   004C'    2803                    JRZ     ..INC   ;IF SO, CONTINUE
   004E'    21 002A"                LXI     H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
   0051'    CD 0000:09      ..INC:  CALL    WAIT#   ;WAIT FOR CONSOLE INPUT
   0054'    18EB                    JMPR    ..SINL  ;CONTINUE
   0056'    78              ..SIN:  MOV     A,B     ;GET CHANNEL NUMBER
   0057'    B7                      ORA     A       ;CHANNEL NUMBER=0?
   0058'    2025                    JRNZ    ..S1I   ;IF NOT, CONTINUE
   005A'    F3                      DI              ;ELSE, DISABLE INTERRUPTS
   005B'    2A 0008"                LHLD    S0ICNT  ;GET SERIAL 0 INPUT COUNT
   005E'    2B                      DCX     H       ;DECREMENT SERIAL 0 INPUT COUNT
   005F'    22 0008"                SHLD    S0ICNT  ;UPDATE SERIAL 0 INPUT COUNT
   0062'    2A 0006"                LHLD    S0OPTR  ;GET SERIAL 0 OUTPUT POINTER
   0065'    7E                      MOV     A,M     ;GET CHARACTER FROM BUFFER
   0066'    23                      INX     H       ;INCREMENT SERIAL 0 OUTPUT POINTER
   0067'    EB                      XCHG            ;SERIAL 0 OUTPUT POINTER TO DE-REG
   0068'    2A 0002"                LHLD    S0IBUF  ;GET SERIAL 0 INPUT BUFFER ADDRESS
   006B'    ED4B 0000"              LBCD    S0IBSZ  ;GET SERIAL 0 INPUT BUFFER SIZE
   006F'    0B                      DCX     B       ;DECREMENT INPUT BUFFER SIZE
   0070'    09                      DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
   0071'    ED52                    DSBC    D       ;BUFFER WRAP-AROUND?
   0073'    3004                    JRNC    ..NWAO  ;IF NOT, CONTINUE
   0075'    ED5B 0002"              LDED    S0IBUF  ;GET SERIAL 0 INPUT BUFFER ADDRESS
```

```
  00EB'   0000                     .WORD   0        ;PREDECESSOR LINK POINTER
                          ;
  00ED'   DB10            SOOPR:   IN      SOCTRL   ;GET SERIAL 0 STATUS
  00EF'   CB4F                     BIT     TBE,A    ;TRANSMIT BUFFER EMPTY?
  00F1'   C8                       RZ               ;IF NOT, DONE
  00F2'   21 000B"                 LXI     H,SOBR   ;ELSE, GET SERIAL 0 BAUD RATE CODE
  00F5'   CB76                     BIT     6,M      ;CTS HANDSHAKING REQUESTED?
  00F7'   2803                     JRZ     ..NCTS   ;IF NOT, CONTINUE
  00F9'   CB7F                     BIT     CTSN,A   ;CHECK CLEAR TO SEND (NOT) STATUS
  00FB'   C0                       RNZ              ;IF CLEAR TO SEND FALSE, DONE
  00FC'   3A 000A"        ..NCTS:  LDA     SOOCHR   ;GET SERIAL 0 OUTPUT CHARACTER
  00FF'   D311                     OUT     SODATA   ;OUTPUT CHARACTER
  0101'   21 00E9'                 LXI     H,SOOPOL  ;GET SERIAL 0 OUT POLL ROUTINE
  0104'   CD 0000:0D               CALL    UNLINK# ;UNLINK POLL ROUTINE
  0107'   21 0012"                 LXI     H,SOOSPH  ;GET SERIAL 0 OUT SEMAPHORE
  010A'   C3 0000:0C               JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                          ;
  010D'                   S1OPOL:                   ;SERIAL 1 OUTPUT POLL ROUTINE
  010D'   0000                     .WORD   0        ;SUCCESSOR LINK POINTER
  010F'   0000                     .WORD   0        ;PREDECESSOR LINK POINTER
                          ;
  0111'   DB12            S1OPR:   IN      S1CTRL   ;GET SERIAL 1 STATUS
  0113'   CB4F                     BIT     TBE,A    ;TRANSMIT BUFFER EMPTY?
  0115'   C8                       RZ               ;IF NOT, DONE
  0116'   21 0029"                 LXI     H,S1BR   ;ELSE, GET SERIAL 1 BAUD RATE CODE
  0119'   CB76                     BIT     6,M      ;CTS HANDSHAKING REQUESTED?
  011B'   2803                     JRZ     ..NCTS   ;IF NOT, CONTINUE
  011D'   CB7F                     BIT     CTSN,A   ;CHECK CLEAR TO SEND (NOT) STATUS
  011F'   C0                       RNZ              ;IF CLEAR TO SEND FALSE, DONE
  0120'   3A 0028"        ..NCTS:  LDA     S1OCHR   ;GET SERIAL 1 OUTPUT CHARACTER
  0123'   D313                     OUT     S1DATA   ;OUTPUT CHARACTER
  0125'   21 010D'                 LXI     H,S1OPOL  ;GET SERIAL 1 OUT POLL ROUTINE
  0128'   CD 0000:0D               CALL    UNLINK# ;UNLINK POLL ROUTINE
  012B'   21 0030"                 LXI     H,S1OSPH  ;GET SERIAL 1 OUT SEMAPHORE
  012E'   C3 0000:0C               JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                          ;
  0131'   ED73 0000:0E    SERISR:  SSPD    INTSP#   ;SAVE STACK POINTER
  0135'   31 0000:0F               LXI     SP,INTSTK#  ;SET UP AUX STACK POINTER
  0138'   F5                       PUSH    PSW      ;SAVE REGISTERS
  0139'   C5                       PUSH    B
  013A'   D5                       PUSH    D
  013B'   E5                       PUSH    H
  013C'   CD 0150'                 CALL    ..SOI    ;CHECK FOR SERIAL 0 INPUT
  013F'   CD 01A6'                 CALL    ..S1I    ;CHECK FOR SERIAL 1 INPUT
  0142'   CD 0000:10               CALL    ISR480# ;CHECK FOR IMS 480 INPUT
  0145'   E1                       POP     H        ;RESTORE REGISTERS
  0146'   D1                       POP     D
  0147'   C1                       POP     B
  0148'   F1                       POP     PSW
  0149'   ED7B 0000:0E             LSPD    INTSP#   ;RESTORE STACK POINTER
  014D'   C3 0000:11               JMP     ISRXIT# ;CONTINUE
  0150'   DB10            ..SOI:   IN      SOCTRL   ;GET SERIAL 0 STATUS
  0152'   CB47                     BIT     RDA,A    ;CHARACTER AVAILABLE
  0154'   C8                       RZ               ;IF NOT, DONE
  01CC'   2A 001E"       ..NAD1: LHLD    S1IBSZ  ;GET SERIAL 1 INPUT BUFFER SIZE
  01CF'   ED5B 0026"             LDED    S1ICNT  ;GET SERIAL 1 INPUT COUNT
  01D3'   13                     INX     D       ;INCREMENT SERIAL 1 INPUT COUNT
  01D4'   B7                     ORA     A       ;CLEAR CARRY FLAG
  01D5'   ED52                   DSBC    D       ;SERIAL 1 INPUT BUFFER FULL?
  01D7'   D8                     RC              ;IF SO, DONE
  01D8'   ED53 0026"             SDED    S1ICNT  ;ELSE, UPDATE SERIAL 1 INPUT COUNT
  01DC'   2A 0022"               LHLD    S1IPTR  ;GET SERIAL 1 INPUT POINTER
  01DF'   71                     MOV     M,C     ;STORE INPUT CHARACTER IN BUFFER
  01E0'   23                     INX     H       ;INCREMENT INPUT POINTER
  01E1'   EB                     XCHG            ;INPUT BUFFER POINTER TO DE-REG
  01E2'   2A 001E"               LHLD    S1IBSZ  ;GET SERIAL 1 INPUT BUFFER SIZE
  01E5'   2B                     DCX     H       ;DECREMENT INPUT BUFFER SIZE
  01E6'   ED4B 0020"             LBCD    S1IBUF  ;GET SERIAL 1 INPUT BUFFER ADDRESS
  01EA'   09                     DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
  01EB'   ED52                   DSBC    D       ;BUFFER WRAP-AROUND?
  01ED'   3004                   JRNC    ..NWA1  ;IF NOT, CONTINUE
  01EF'   ED5B 0020"             LDED    S1IBUF  ;GET SERIAL 1 INPUT BUFFER ADDRESS
  01F3'   ED53 0022"     ..NWA1: SDED    S1IPTR  ;UPDATE SERIAL 1 INPUT POINTER
  01F7'   21 002A"               LXI     H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
  01FA'   7E             ..X:    MOV     A,M     ;GET SEMAPHORE COUNT
  01FB'   B7                     ORA     A       ;SEMAPHORE COUNT=0?
  01FC'   C8                     RZ              ;IF SO, DONE
  01FD'   C3 0000:0C             JMP     SIGNAL# ;ELSE, SIGNAL PROCESS AS READY
                                 ;
  0200'   78             SERSBR: MOV     A,B     ;GET CHANNEL NUMBER
  0201'   FE02                   CPI     2       ;CHANNEL NUMBER=0/1?
  0203'   D2 0000:13             JNC     SBR480# ;IF NOT, CONTINUE
  0206'   21 000B"               LXI     H,S0BR  ;ELSE, GET SERIAL 0 BAUD RATE CODE
  0209'   B7                     ORA     A       ;CHANNEL NUMBER=0?
  020A'   2803                   JRZ     ..COM1  ;IF SO, CONTINUE
  020C'   21 0029"               LXI     H,S1BR  ;ELSE, GET SERIAL 1 BAUD RATE CODE
  020F'   71             ..COM1: MOV     M,C     ;SAVE BAUD RATE CODE
  0210'   CD 0226'               CALL    GETBTV  ;GET BAUD RATE TIMER VALUE
  0213'   78                     MOV     A,B     ;GET CHANNEL NUMBER
  0214'   B7                     ORA     A       ;CHANNEL NUMBER=0?
  0215'   3E36                   MVI     A,T0CMD ;GET TIMER 0 COMMAND
  0217'   0E14                   MVI     C,TIM0  ;GET TIMER 0 DATA REGISTER
  0219'   2804                   JRZ     ..COM2  ;IF CHANNEL NUMBER=0, CONTINUE
  021B'   3E76                   MVI     A,T1CMD ;ELSE, GET TIMER 1 COMMAND
  021D'   0E15                   MVI     C,TIM1  ;GET TIMER 1 DATA REGISTER
  021F'   D317           ..COM2: OUT     TIMCTL  ;SELECT TIMER
  0221'   ED59                   OUTP    E       ;OUTPUT LSB OF TIMER VALUE
  0223'   ED51                   OUTP    D       ;OUTPUT MSB OF TIMER VALUE
  0225'   C9                     RET             ;DONE
                                 ;
  0226'   79             GETBTV::MOV     A,C     ;GET REQUESTED BAUD RATE CODE
  0227'   E60F                   ANI     0FH     ;EXTRACT RELEVANT BITS
  0229'   87                     ADD     A       ;X2
  022A'   5F                     MOV     E,A     ;TO E-REG
  022B'   1600                   MVI     D,0     ;MAKE IT DOUBLE LENGTH
  022D'   21 0000:14             LXI     H,BRTBL#  ;GET BAUD RATE TABLE
  0230'   19                     DAD     D       ;INDEX INTO TABLE
  0231'   5E                     MOV     E,M     ;GET TIMER VALUE
```

```
                         ;
                         ; *********************************************************
                         ; *                                                       *
                         ; *  This module is called by SPD442, and handles the     *
                         ; *  four serial ports on the IMS 480 four-port serial    *
                         ; *  board.  Input is handled on an interrupt-driven      *
                         ; *  basis and buffered via a circular buffer.  Output    *
                         ; *  is handled on a polled basis and not buffered.       *
                         ; *                                                       *
                         ; *                                                       *
                         ; *********************************************************
                         ;
                         ;
                         ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                         ;
                         ; AUTHORS: RONALD E. RAIKES
                         ;          MICHAEL D. BUSCH
                         ;
                         ; VERSION: EXAMPLE
                         ;
                         .IDENT  SER480          ;MODULE ID
                         ;
                         .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                         ;
   00E0                  IOBASE  = 0E0H          ;I/O PORT BASE
                         ;
   00E0                  S2DATA  = IOBASE+00H    ;SERIAL 2 DATA REGISTER
   00E1                  S2IER   = IOBASE+01H    ;SERIAL 2 INTERRUPT ENABLE REGISTE
   00E2                  S2IIDR  = IOBASE+02H    ;SERIAL 2 INTERRUPT ID REGISTER
   00E3                  S2LCR   = IOBASE+03H    ;SERIAL 2 LINE CONTROL REGISTER
   00E4                  S2MCR   = IOBASE+04H    ;SERIAL 2 MODEM CONTROL REGISTER
   00E5                  S2LSR   = IOBASE+05H    ;SERIAL 2 LINE STATUS REGISTER
   00E6                  S2MSR   = IOBASE+06H    ;SERIAL 2 MODEM STATUS REGISTER
                         ;
   00E8                  S3DATA  = IOBASE+08H    ;SERIAL 3 DATA REGISTER
   00E9                  S3IER   = IOBASE+09H    ;SERIAL 3 INTERRUPT ENABLE REGIST'
   00EA                  S3IIDR  = IOBASE+0AH    ;SERIAL 3 INTERRUPT ID REGISTER
   00EB                  S3LCR   = IOBASE+0BH    ;SERIAL 3 LINE CONTROL REGISTER
   00EC                  S3MCR   = IOBASE+0CH    ;SERIAL 3 MODEM CONTROL REGISTER
   00ED                  S3LSR   = IOBASE+0DH    ;SERIAL 3 LINE STATUS REGISTER
   00EE                  S3MSR   = IOBASE+0EH    ;SERIAL 3 MODEM STATUS REGISTER
                         ;
   00F0                  S4DATA  = IOBASE+10H    ;SERIAL 4 DATA REGISTER
   00F1                  S4IER   = IOBASE+11H    ;SERIAL 4 INTERRUPT ENABLE REGISTE
   00F2                  S4IIDR  = IOBASE+12H    ;SERIAL 4 INTERRUPT ID REGISTER
   00F3                  S4LCR   = IOBASE+13H    ;SERIAL 4 LINE CONTROL REGISTER
   00F4                  S4MCR   = IOBASE+14H    ;SERIAL 4 MODEM CONTROL REGISTER
   00F5                  S4LSR   = IOBASE+15H    ;SERIAL 4 LINE STATUS REGISTER
   00F6                  S4MSR   = IOBASE+16H    ;SERIAL 4 MODEM STATUS REGISTER
                         ;
   00F8                  S5DATA  = IOBASE+18H    ;SERIAL 5 DATA REGISTER
   00F9                  S5IER   = IOBASE+19H    ;SERIAL 5 INTERRUPT ENABLE REGISTE
   00FA                  S5IIDR  = IOBASE+1AH    ;SERIAL 5 INTERRUPT ID REGISTER
   00FB                  S5LCR   = IOBASE+1BH    ;SERIAL 5 LINE CONTROL REGISTER
   00FC                  S5MCR   = IOBASE+1CH    ;SERIAL 5 MODEM CONTROL REGISTER
```

```
                                                    ;SERIAL 3 OUTPUT SEMAPHORE
  0036"    0001          S3XSPH: .WORD    1         ;SEMAPHORE COUNT
  0038"    0038"         ..S3XH: .WORD    ..S3XH    ;SEMAPHORE P/D HEAD
  003A"    0038"                 .WORD    ..S3XH
                         ;
  003C"    0010          S4IBSZ::.WORD    16        ;SERIAL 4 INPUT BUFFER SIZE
  003E"    0000          S4IBUF: .WORD    0         ;SERIAL 4 INPUT BUFFER ADDRESS
  0040"    0000          S4IPTR: .WORD    0         ;SERIAL 4 INPUT POINTER
  0042"    0000          S4OPTR: .WORD    0         ;SERIAL 4 OUTPUT POINTER
  0044"    0000          S4ICNT: .WORD    0         ;SERIAL 4 INPUT COUNT
  0046"    00            S4OCHR: .BYTE    0         ;SERIAL 4 OUTPUT CHARACTER
  0047"    00            S4BR:   .BYTE    0         ;SERIAL 4 BAUD RATE CODE
                         ;
  0048"                  S4ISPH:                    ;SERIAL 4 INPUT SEMAPHORE
  0048"    0000                  .WORD    0         ;SEMAPHORE COUNT
  004A"    004A"         ..S4IH: .WORD    ..S4IH    ;SEMAPHORE P/D HEAD
  004C"    004A"                 .WORD    ..S4IH
                         ;
                                                    ;SERIAL 4 OUTPUT SEMAPHORE
  004E"    0000          S4OSPH: .WORD    0         ;SEMAPHORE COUNT
  0050"    0050"         ..S4OH: .WORD    ..S4OH    ;SEMAPHORE P/D HEAD
  0052"    0050"                 .WORD    ..S4OH
                         ;
                                                    ;SERIAL 4 OUTPUT SEMAPHORE
  0054"    0001          S4XSPH: .WORD    1         ;SEMAPHORE COUNT
  0056"    0056"         ..S4XH: .WORD    ..S4XH    ;SEMAPHORE P/D HEAD
  0058"    0056"                 .WORD    ..S4XH
                         ;
  005A"    0010          S5IBSZ::.WORD    16        ;SERIAL 5 INPUT BUFFER SIZE
  005C"    0000          S5IBUF: .WORD    0         ;SERIAL 5 INPUT BUFFER ADDRESS
  005E"    0000          S5IPTR: .WORD    0         ;SERIAL 5 INPUT POINTER
  0060"    0000          S5OPTR: .WORD    0         ;SERIAL 5 OUTPUT POINTER
  0062"    0000          S5ICNT: .WORD    0         ;SERIAL 5 INPUT COUNT
  0064"    00            S5OCHR: .BYTE    0         ;SERIAL 5 OUTPUT CHARACTER
  0065"    00            S5BR:   .BYTE    0         ;SERIAL 5 BAUD RATE CODE
                         ;
                                                    ;SERIAL 5 INPUT SEMAPHORE
  0066"    0000          S5ISPH: .WORD    0         ;SEMAPHORE COUNT
  0068"    0068"         ..S5IH: .WORD    ..S5IH    ;SEMAPHORE P/D HEAD
  006A"    0068"                 .WORD    ..S5IH
                         ;
                                                    ;SERIAL 5 OUTPUT SEMAPHORE
  006C"    0000          S5OSPH: .WORD    0         ;SEMAPHORE COUNT
  006E"    006E"         ..S5OH: .WORD    ..S5OH    ;SEMAPHORE P/D HEAD
  0070"    006E"                 .WORD    ..S5OH
                         ;
                                                    ;SERIAL 5 OUTPUT SEMAPHORE
  0072"    0001          S5XSPH: .WORD    1         ;SEMAPHORE COUNT
  0074"    0074"         ..S5XH: .WORD    ..S5XH    ;SEMAPHORE P/D HEAD
  0076"    0074"                 .WORD    ..S5XH
                         ;
                         ;
  0000:04                        .LOC     .INIT.#   ;LOCATE IN INITIALIZATION AREA
                         ;
  001F'   ED5B 0044"    ..ST4:  LDED    S4ICNT  ;GET SERIAL 4 INPUT BUFFER COUNT
  0023'   2A 0042"              LHLD    S4OPTR  ;GET SERIAL 4 OUTPUT POINTER
  0026'   1807                  JMPR    ..COM   ;CONTINUE
  0028'   ED5B 0062"    ..ST5:  LDED    S5ICNT  ;GET SERIAL 5 INPUT BUFFER COUNT
  002C'   2A 0060"              LHLD    S5OPTR  ;GET SERIAL 5 OUTPUT POINTER
  002F'   7A            ..COM:  MOV     A,D
  0030'   B3                    ORA     E       ;SERIAL INPUT BUFFER COUNT=0?
  0031'   C8                    RZ              ;IF SO, DONE
  0032'   3EFF                  MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
  0034'   4E                    MOV     C,M     ;GET SERIAL INPUT CHARACTER
  0035'   C9                    RET             ;DONE
                        ;
  0036'   CD 0000'      IN480:: CALL    ST480   ;GET SERIAL STATUS
  0039'   B7                    ORA     A       ;CHARACTER AVAILABLE?
  003A'   201C                  JRNZ    ..SIN   ;IF SO, CONTINUE
  003C'   78                    MOV     A,B     ;ELSE, GET CHANNEL NUMBER
  003D'   21 000C"              LXI     H,S2ISPH  ;GET SERIAL 2 INPUT SEMAPHORE
  0040'   D602                  SUI     2       ;CHANNEL NUMBER=2?
  0042'   280F                  JRZ     ..INC   ;IF SO, CONTINUE
  0044'   21 002A"              LXI     H,S3ISPH  ;GET SERIAL 3 INPUT SEMAPHORE
  0047'   3D                    DCR     A       ;CHANNEL NUMBER=3?
  0048'   2809                  JRZ     ..INC   ;IF SO, CONTINUE
  004A'   21 0048"              LXI     H,S4ISPH  ;GET SERIAL 4 INPUT SEMAPHORE
  004D'   3D                    DCR     A       ;CHANNEL NUMBER=4?
  004E'   2803                  JRZ     ..INC   ;IF SO, CONTINUE
  0050'   21 0066"              LXI     H,S5ISPH  ;GET SERIAL 5 INPUT SEMAPHORE
                        ;
  0053'   CD 0000:06    ..INC:  CALL    WAIT#   ;WAIT FOR INPUT CHARACTER
  0056'   18DE                  JMPR    IN480   ;CONTINUE
  0058'   78            ..SIN:  MOV     A,B     ;GET CHANNEL NUMBER
  0059'   D602                  SUI     2       ;CHANNEL NUMBER=2?
  005B'   2808                  JRZ     ..IN2   ;IF SO, CONTINUE
  005D'   3D                    DCR     A       ;CHANNEL NUMBER=3?
  005E'   282A                  JRZ     ..IN3   ;IF SO, CONTINUE
  0060'   3D                    DCR     A       ;CHANNEL NUMBER=4?
  0061'   284C                  JRZ     ..IN4   ;IF SO, CONTINUE
  0063'   186F                  JMPR    ..IN5   ;ELSE, CONTINUE
                        ;
  0065'   F3            ..IN2:  DI              ;DISABLE INTERRUPTS
  0066'   2A 0008"              LHLD    S2ICNT  ;GET SERIAL 2 INPUT COUNT
  0069'   2B                    DCX     H       ;DECREMENT SERIAL 2 INPUT COUNT
  006A'   22 0008"              SHLD    S2ICNT  ;UPDATE SERIAL 2 INPUT COUNT
  006D'   2A 0006"              LHLD    S2OPTR  ;GET SERIAL 2 OUTPUT POINTER
  0070'   7E                    MOV     A,M     ;GET CHARACTER FROM BUFFER
  0071'   23                    INX     H       ;INCREMENT SERIAL 2 OUTPUT POINTER
  0072'   EB                    XCHG            ;SERIAL 2 OUTPUT POINTER TO DE-REG
  0073'   2A 0002"              LHLD    S2IBUF  ;GET SERIAL 2 INPUT BUFFER ADDRESS
  0076'   ED4B 0000"            LBCD    S2IBSZ  ;GET SERIAL 2 INPUT BUFFER SIZE
  007A'   0B                    DCX     B       ;DECREMENT INPUT BUFFER SIZE
  007B'   09                    DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
  007C'   ED52                  DSBC    D       ;BUFFER WRAP-AROUND?
  007E'   3004                  JRNC    ..NWA2  ;IF NOT, CONTINUE
  0080'   ED5B 0002"            LDED    S2IBUF  ;GET SERIAL 2 INPUT BUFFER ADDRESS
  0084'   ED53 0006"    ..NWA2: SDED    S2OPTR  ;UPDATE SERIAL 2 OUTPUT POINTER
  00ED'   3004                    JRNC    ..NWA5  ;IF NOT, CONTINUE
  00EF'   ED5B 005C"              LDED    S5IBUF  ;GET SERIAL 5 INPUT BUFFER ADDRESS
  00F3'   ED53 0060"     ..NWA5:  SDED    S5OPTR  ;UPDATE SERIAL 5 OUTPUT POINTER
  00F7'   FB                      EI              ;ENABLE INTERRUPTS
  00F8'   C9                      RET             ;DONE
                         ;
  00F9'   D602           OUT480::SUI      2       ;CHANNEL NUMBER=2?
  00FB'   2808                    JRZ     ..OUT2  ;IF SO, CONTINUE
  00FD'   3D                      DCR     A       ;CHANNEL NUMBER=3?
  00FE'   2823                    JRZ     ..OUT3  ;IF SO, CONTINUE
  0100'   3D                      DCR     A       ;CHANNEL NUMBER=4?
  0101'   283E                    JRZ     ..OUT4  ;IF SO, CONTINUE
  0103'   185A                    JMPR    ..OUT5  ;ELSE, CONTINUE
                         ;
  0105'   21 0018"       ..OUT2:  LXI     H,S2XSPH  ;GET SERIAL 2 OUT SEMAPHORE
  0108'   E5                      PUSH    H       ;SAVE SERIAL 2 OUT SEMAPHORE
  0109'   CD 0000:06              CALL    WAIT#   ;WAIT ON MUTUAL EXCLUSION
  010C'   21 000A"                LXI     H,S2OCHR  ;GET SERIAL 2 OUTPUT CHARACTER
  010F'   71                      MOV     M,C     ;SAVE OUTPUT CHARACTER
  0110'   11 017D'                LXI     D,S2OPOL  ;GET SERIAL 2 OUT POLL ROUTINE
  0113'   CD 0000:07              CALL    LNKPOL# ;CREATE POLL ROUTINE
  0116'   CD 0181'                CALL    S2OPR   ;EXECUTE POLL ROUTINE
  0119'   21 0012"                LXI     H,S2OSPH  ;GET SERIAL 2 OUT SEMAPHORE
  011C'   CD 0000:06              CALL    WAIT#   ;DISPATCH IF NECESSARY
  011F'   E1                      POP     H       ;GET MUTUTAL EXCLUSION SEMAPHORE
  0120'   C3 0000:08              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                         ;
  0123'   21 0036"       ..OUT3:  LXI     H,S3XSPH  ;GET SERIAL 3 OUT SEMAPHORE
  0126'   E5                      PUSH    H       ;SAVE SERIAL 3 OUT SEMAPHORE
  0127'   CD 0000:06              CALL    WAIT#   ;WAIT ON MUTUAL EXCLUSION
  012A'   21 0028"                LXI     H,S3OCHR  ;GET SERIAL 3 OUTPUT CHARACTER
  012D'   71                      MOV     M,C     ;SAVE OUTPUT CHARACTER
  012E'   11 01A3'                LXI     D,S3OPOL  ;GET SERIAL 3 OUT POLL ROUTINE
  0131'   CD 0000:07              CALL    LNKPOL# ;CREATE POLL ROUTINE
  0134'   CD 01A7'                CALL    S3OPR   ;EXECUTE POLL ROUTINE
  0137'   21 0030"                LXI     H,S3OSPH  ;GET SERIAL 3 OUT SEMAPHORE
  013A'   CD 0000:06              CALL    WAIT#   ;DISPATCH IF NECESSARY
  013D'   E1                      POP     H       ;GET MUTUTAL EXCLUSION SEMAPHORE
  013E'   C3 0000:08              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                         ;
  0141'   21 0054"       ..OUT4:  LXI     H,S4XSPH  ;GET SERIAL 4 OUT SEMAPHORE
  0144'   E5                      PUSH    H       ;SAVE SERIAL 4 OUT SEMAPHORE
  0145'   CD 0000:06              CALL    WAIT#   ;WAIT ON MUTUAL EXCLUSION
  0148'   21 0046"                LXI     H,S4OCHR  ;GET SERIAL 4 OUTPUT CHARACTER
  014B'   71                      MOV     M,C     ;SAVE OUTPUT CHARACTER
  014C'   11 01C9'                LXI     D,S4OPOL  ;GET SERIAL 4 OUT POLL ROUTINE
  014F'   CD 0000:07              CALL    LNKPOL# ;CREATE POLL ROUTINE
  0152'   CD 01CD'                CALL    S4OPR   ;EXECUTE POLL ROUTINE
  0155'   21 004E"                LXI     H,S4OSPH  ;GET SERIAL 4 OUT SEMAPHORE
  0158'   CD 0000:06              CALL    WAIT#   ;DISPATCH IF NECESSARY
  015B'   E1                      POP     H       ;GET MUTUTAL EXCLUSION SEMAPHORE
  015C'   C3 0000:08              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                         ;
  015F'   21 0072"       ..OUT5:  LXI     H,S5XSPH  ;GET SERIAL 5 OUT SEMAPHORE
```

```
   01CB'    0000                      .WORD    0         ;PREDECESSOR LINK POINTER
                              ;
   01CD'    DBF5              S4OPR:  IN       S4LSR     ;GET SERIAL 4 LINE STATUS REGISTER
   01CF'    CB6F                      BIT      TBE,A     ;TRANSMIT BUFFER EMPTY?
   01D1'    C8                        RZ                 ;IF NOT, DONE
   01D2'    21 0047"                  LXI      H,S4BR    ;ELSE, GET SERIAL 4 BAUD RATE CODE
   01D5'    CB76                      BIT      6,M       ;CLEAR TO SEND HANDSHAKING?
   01D7'    2805                      JRZ      ..NCTS    ;IF NOT, CONTINUE
   01D9'    DBF6                      IN       S4MSR     ;GET SERIAL 4 MODEM STATUS REGISTE
   01DB'    CB67                      BIT      CTS,A     ;CLEAR TO SEND STATUS TRUE?
   01DD'    C8                        RZ                 ;IF NOT, DONE
   01DE'    3A 0046"          ..NCTS: LDA      S4OCHR    ;GET SERIAL 4 OUTPUT CHARACTER
   01E1'    D3F0                      OUT      S4DATA    ;OUTPUT CHARACTER
   01E3'    21 01C9'                  LXI      H,S4OPOL  ;GET SERIAL 4 OUT POLL ROUTINE
   01E6'    CD 0000:09                CALL     UNLINK# ;UNLINK POLL ROUTINE
   01E9'    21 004E"                  LXI      H,S4OSPH  ;GET SERIAL 4 OUT SEMAPHORE
   01EC'    C3 0000:08                JMP      SIGNAL# ;SIGNAL PROCESS AS READY
                              ;
   01EF'                      S5OPOL:                    ;SERIAL 5 OUTPUT POLL ROUTINE
   01EF'    0000                      .WORD    0         ;SUCCESSOR LINK POINTER
   01F1'    0000                      .WORD    0         ;PREDECESSOR LINK POINTER
                              ;
   01F3'    DBFD              S5OPR:  IN       S5LSR     ;GET SERIAL 5 LINE STATUS REGISTER
   01F5'    CB6F                      BIT      TBE,A     ;TRANSMIT BUFFER EMPTY?
   01F7'    C8                        RZ                 ;IF NOT, DONE
   01F8'    21 0065"                  LXI      H,S5BR    ;ELSE, GET SERIAL 5 BAUD RATE CODE
   01FB'    CB76                      BIT      6,M       ;CLEAR TO SEND HANDSHAKING?
   01FD'    2805                      JRZ      ..NCTS    ;IF NOT, CONTINUE
   01FF'    DBFE                      IN       S5MSR     ;GET SERIAL 5 MODEM STATUS REGISTE
   0201'    CB67                      BIT      CTS,A     ;CLEAR TO SEND STATUS TRUE?
   0203'    C8                        RZ                 ;IF NOT, DONE
   0204'    3A 0064"          ..NCTS: LDA      S5OCHR    ;GET SERIAL 5 OUTPUT CHARACTER
   0207'    D3F8                      OUT      S5DATA    ;OUTPUT CHARACTER
   0209'    21 01EF'                  LXI      H,S5OPOL  ;GET SERIAL 5 OUT POLL ROUTINE
   020C'    CD 0000:09                CALL     UNLINK# ;UNLINK POLL ROUTINE
   020F'    21 006C"                  LXI      H,S5OSPH  ;GET SERIAL 5 OUT SEMAPHORE
   0212'    C3 0000:08                JMP      SIGNAL# ;SIGNAL PROCESS AS READY
                              ;
   0215'    CD 0222'          ISR480::CALL     ..S2I     ;CHECK FOR SERIAL 2 INPUT
   0218'    CD 0276'                  CALL     ..S3I     ;CHECK FOR SERIAL 3 INPUT
   021B'    CD 02CA'                  CALL     ..S4I     ;CHECK FOR SERIAL 4 INPUT
   021E'    CD 0319'                  CALL     ..S5I     ;CHECK FOR SERIAL 5 INPUT
   0221'    C9                        RET                ;DONE
                              ;
   0222'    DBE5              ..S2I:  IN       S2LSR     ;GET SERIAL 2 STATUS
   0224'    CB47                      BIT      RDA,A     ;CHARACTER AVAILABLE
   0226'    C8                        RZ                 ;IF NOT, DONE
   0227'    DBE0                      IN       S2DATA    ;ELSE, GET SERIAL 2 DATA CHARACTER
   0229'    21 000B"                  LXI      H,S2BR    ;GET SERIAL 2 BAUD RATE CODE
   022C'    CB6E                      BIT      5,M       ;INHIBIT INPUT FLAG SET?
   022E'    C0                        RNZ                ;IF SO, DONE
   022F'    4F                        MOV      C,A       ;SERIAL 2 DATA CHARACTER TO C-REG
   0230'    CB7E                      BIT      7,M       ;ATTENTION DETECTION FLAG SET?
   0232'    2814                      JRZ      ..NAD2    ;IF NOT, CONTINUE
```

```
    02AC'   2A 0022"                 LHLD    S3IPTR  ;GET SERIAL 3 INPUT POINTER
    02AF'   71                       MOV     M,C     ;STORE INPUT CHARACTER IN BUFFER
    02B0'   23                       INX     H       ;INCREMENT INPUT POINTER
    02B1'   EB                       XCHG            ;INPUT BUFFER POINTER TO DE-REG
    02B2'   2A 001E"                 LHLD    S3IBSZ  ;GET SERIAL 3 INPUT BUFFER SIZE
    02B5'   2B                       DCX     H       ;DECREMENT INPUT BUFFER SIZE
    02B6'   ED4B 0020"               LBCD    S3IBUF  ;GET SERIAL 3 INPUT BUFFER ADDRESS
    02BA'   09                       DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
    02BB'   ED52                     DSBC    D       ;BUFFER WRAP-AROUND?
    02BD'   3004                     JRNC    ..NWA3  ;IF NOT, CONTINUE
    02BF'   ED5B 0020"               LDED    S3IBUF  ;GET SERIAL 3 INPUT BUFFER ADDRESS
    02C3'   ED53 0022"       ..NWA3: SDED    S3IPTR  ;UPDATE SERIAL 3 INPUT POINTER
    02C7'   C3 036D'                 JMP     ..X     ;CONTINUE
                             ;
    02CA'   DBF5             ..S4I:  IN      S4LSR   ;GET SERIAL 4 STATUS
    02CC'   CB47                     BIT     RDA,A   ;CHARACTER AVAILABLE
    02CE'   C8                       RZ              ;IF NOT, DONE
    02CF'   DBF0                     IN      S4DATA  ;ELSE, GET SERIAL 4 DATA CHARACTER
    02D1'   21 0047"                 LXI     H,S4BR  ;GET SERIAL 4 BAUD RATE CODE
    02D4'   CB6E                     BIT     5,M     ;INHIBIT INPUT FLAG SET?
    02D6'   C0                       RNZ             ;IF SO, DONE
    02D7'   4F                       MOV     C,A     ;SERIAL 4 DATA CHARACTER TO C-REG
    02D8'   CB7E                     BIT     7,M     ;ATTENTION DETECTION FLAG SET?
    02DA'   2814                     JRZ     ..NAD4  ;IF NOT, CONTINUE
    02DC'   CBB9                     RES     7,C     ;STRIP SIGN BIT ON INPUT CHARACTER
    02DE'   3A 0000:0A               LDA     ATNCHR# ;GET ATTENTION CHARACTER
    02E1'   B9                       CMP     C       ;CHARACTER=ATTENTION CHARACTER?
    02E2'   200C                     JRNZ    ..NAD4  ;IF NOT, CONTINUE
    02E4'   2A 0040"                 LHLD    S4IPTR  ;ELSE, GET SERIAL 4 INPUT POINTER
    02E7'   22 0042"                 SHLD    S4OPTR  ;RESET SERIAL 4 OUTPUT POINTER
    02EA'   21 0000                  LXI     H,0
    02ED'   22 0044"                 SHLD    S4ICNT  ;SET SERIAL 4 INPUT COUNT=0
    02F0'   2A 003C"         ..NAD4: LHLD    S4IBSZ  ;GET SERIAL 4 INPUT BUFFER SIZE
    02F3'   ED5B 0044"               LDED    S4ICNT  ;GET SERIAL 4 INPUT COUNT
    02F7'   13                       INX     D       ;INCREMENT SERIAL 4 INPUT COUNT
    02F8'   B7                       ORA     A       ;CLEAR CARRY FLAG
    02F9'   ED52                     DSBC    D       ;SERIAL 4 INPUT BUFFER FULL?
    02FB'   D0                       RNC             ;IF SO, DONE
    02FC'   ED53 0044"               SDED    S4ICNT  ;ELSE, UPDATE SERIAL 4 INPUT COUNT
    0300'   2A 0040"                 LHLD    S4IPTR  ;GET SERIAL 4 INPUT POINTER
    0303'   71                       MOV     M,C     ;STORE INPUT CHARACTER IN BUFFER
    0304'   23                       INX     H       ;INCREMENT INPUT POINTER
    0305'   2B                       DCX     H       ;DECREMENT INPUT BUFFER SIZE
    0306'   ED4B 003E"               LBCD    S4IBUF  ;GET SERIAL 4 INPUT BUFFER ADDRESS
    030A'   09                       DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
    030B'   ED52                     DSBC    D       ;BUFFER WRAP-AROUND?
    030D'   3004                     JRNC    ..NWA4  ;IF NOT, CONTINUE
    030F'   ED5B 003E"               LDED    S4IBUF  ;GET SERIAL 4 INPUT BUFFER ADDRESS
    0313'   ED53 0040"       ..NWA4: SDED    S4IPTR  ;UPDATE SERIAL 4 INPUT POINTER
    0317'   1854                     JMPR    ..X     ;CONTINUE
                             ;
    0319'   DBFD             ..S5I:  IN      S5LSR   ;GET SERIAL 5 STATUS
    031B'   CB47                     BIT     RDA,A   ;CHARACTER AVAILABLE
    031D'   C8                       RZ              ;IF NOT, DONE
```

```
  0386'   0D                      DCR     C
  0387'   0D                      DCR     C
  0388'   ED59                    OUTP    E       ;OUTPUT LSB OF TIMER VALUE
  038A'   0C                      INR     C       ;CALC DATA REGISTER+1
  038B'   ED51                    OUTP    D       ;OUTPUT MSB OF TIMER VALUE
  038D'   0C                      INR     C       ;CALC LINE CONTROL REGISTER
  038E'   0C                      INR     C
  038F'   3E03                    MVI     A,LCRCW ;GET LINE CONTROL REGISTER VALUE
  0391'   ED79                    OUTP    A       ;DE-SELECT DIVISOR LATCH
  0393'   C1                      POP     B       ;RESTORE BAUD RATE CODE
  0394'   F1                      POP     PSW     ;RESTORE CHANNEL NUMBER
  0395'   2808                    JRZ     ..SBR2  ;IF CHANNEL NUMBER=2, CONTINUE
  0397'   3D                      DCR     A       ;CHANNEL NUMBER=3?
  0398'   280A                    JRZ     ..SBR3  ;IF SO, CONTINUE
  039A'   3D                      DCR     A       ;CHANNEL NUMBER=4?
  039B'   280C                    JRZ     ..SBR4  ;IF SO, CONTINUE
  039D'   180F                    JMPR    ..SBR5  ;ELSE, CONTINUE
                          ;
  039F'   79              ..SBR2: MOV     A,C     ;GET SERIAL 2 BAUD RATE CODE
  03A0'   32 000B"                STA     S2BR    ;SET SERIAL 2 BAUD RATE CODE
  03A3'   C9                      RET             ;DONE
                          ;
  03A4'   79              ..SBR3: MOV     A,C     ;GET SERIAL 3 BAUD RATE CODE
  03A5'   32 0029"                STA     S3BR    ;SET SERIAL 3 BAUD RATE CODE
  03A8'   C9                      RET             ;DONE
                          ;
  03A9'   79              ..SBR4: MOV     A,C     ;GET SERIAL 4 BAUD RATE CODE
  03AA'   32 0047"                STA     S4BR    ;SET SERIAL 4 BAUD RATE CODE
  03AD'   C9                      RET             ;DONE
                          ;
  03AE'   79              ..SBR5: MOV     A,C     ;GET SERIAL 5 BAUD RATE CODE
  03AF'   32 0065"                STA     S5BR    ;SET SERIAL 5 BAUD RATE CODE
  03B2'   C9                      RET             ;DONE
                          ;
  03B3'   D602            RBR480::SUI     2       ;CHANNEL NUMBER=2?
  03B5'   2808                    JRZ     ..RBR2  ;IF SO, CONTINUE
  03B7'   3D                      DCR     A       ;CHANNEL NUMBER=3?
  03B8'   2809                    JRZ     ..RBR3  ;IF SO, CONTINUE
  03BA'   3D                      DCR     A       ;CHANNEL NUMBER=4?
  03BB'   280A                    JRZ     ..RBR4  ;IF SO, CONTINUE
  03BD'   180C                    JMPR    ..RBR5  ;ELSE, CONTINUE
                          ;
  03BF'   3A 000B"        ..RBR2: LDA     S2BR    ;GET SERIAL 2 BAUD RATE CODE
  03C2'   C9                      RET             ;DONE
                          ;
  03C3'   3A 0029"        ..RBR3: LDA     S3BR    ;GET SERIAL 3 BAUD RATE CODE
  03C6'   C9                      RET             ;DONE
                          ;
  03C7'   3A 0047"        ..RBR4: LDA     S4BR    ;GET SERIAL 4 BAUD RATE CODE
  03CA'   C9                      RET             ;DONE
                          ;
  03CB'   3A 0065"        ..RBR5: LDA     S5BR    ;GET SERIAL 5 BAUD RATE CODE
  03CE'   C9                      RET             ;DONE
                          ;
```

```
BRT442 - TURBODOS OPERATING SYSTEM BAUD RATE TABLE (IMS 442 OPTIONAL OSC)
COPYRIGHT (C) 1982 BY SOFTWARE 2000. INC.

                        ;
                        ; **************************************************
                        ; *                                                *
                        ; *  This module contains the table of baud-rate divi-  *
                        ; *  sor values used by the IMS 442 and IMS 480 I/O     *
                        ; *  boards.  It is used by the SPD442, SER480, and     *
                        ; *  RTC442 modules.                                *
                        ; *                                                *
                        ; **************************************************
                        ;
                        ;
                        ; COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.
                        ;
                        ; AUTHORS: RONALD E. RAIKES
                        ;          MICHAEL D. BUSCH
                        ;
                        ;
                        ; VERSION: EXAMPLE2
                        ;
                        .IDENT  BRT442          ;MODULE ID
                        ;
                        .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                        ;
  0600                  BR50    = 1536          ;50 BAUD TIMER VALUE
  0400                  BR75    = 1024          ;75 BAUD TIMER VALUE
  02BA                  BR110   = 698           ;110 BAUD TIMER VALUE
  023B                  BR1345  = 571           ;134.5 BAUD TIMER VALUE
  0200                  BR150   = 512           ;150 BAUD TIMER VALUE
  0100                  BR300   = 256           ;300 BAUD TIMER VALUE
  0080                  BR600   = 128           ;600 BAUD TIMER VALUE
  0040                  BR1200  = 64            ;1200 BAUD TIMER VALUE
  002B                  BR1800  = 43            ;1800 BAUD TIMER VALUE
  0026                  BR2000  = 38            ;2000 BAUD TIMER VALUE
  0020                  BR2400  = 32            ;2400 BAUD TIMER VALUE
  0015                  BR3600  = 21            ;3600 BAUD TIMER VALUE
  0010                  BR4800  = 16            ;4800 BAUD TIMER VALUE
  000B                  BR7200  = 11            ;7200 BAUD TIMER VALUE
  0008                  BR9600  = 8             ;9600 BAUD TIMER VALUE
  0004                  BR192K  = 4             ;19200 BAUD TIMER VALUE
                        ;
  5000                  RTCCNT  =: 20480        ;RTC COUNT (1/60 SECOND TICK)
  003C                  TICSEC  =: 60           ;RTC TICKS PER SECOND
                        ;
  0000'                         .LOC    .PROG.# ;LOCATE IN PROGRAM AREA
                        ;
  0000'   0600          BRTBL:: .WORD   BR50    ;50 BAUD TIMER VALUE
  0002'   0400                  .WORD   BR75    ;75 BAUD TIMER VALUE
  0004'   02BA                  .WORD   BR110   ;110 BAUD TIMER VALUE
  0006'   023B                  .WORD   BR1345  ;134.5 BAUD TIMER VALUE
  0008'   0200                  .WORD   BR150   ;150 BAUD TIMER VALUE
  000A'   0100                  .WORD   BR300   ;300 BAUD TIMER VALUE
  000C'   0080                  .WORD   BR600   ;600 BAUD TIMER VALUE
  000E'   0040                  .WORD   BR1200  ;1200 BAUD TIMER VALUE
  0010'   002B                  .WORD   BR1800  ;1800 BAUD TIMER VALUE
  0012'   0026                  .WORD   BR2000  ;2000 BAUD TIMER VALUE
  0014'   0020                  .WORD   BR2400  ;2400 BAUD TIMER VALUE
```

```
                          ;
                          ; ********************************************************
                          ; *                                                      *
                          ; *  This is a real-time clock driver, which utilizes    *
                          ; *  a counter on the IMS 442 I/O board to provide a     *
                          ; *  periodic "tick" interrupt at 60 hertz.              *
                          ; *                                                      *
                          ; ********************************************************
                          ;
                          ; COPYRIGHT (C) 1982 BY SOFTWARE 2000, INC.
                          ;
                          ; AUTHORS: RONALD E. RAIKES
                          ;          MICHAEL D. BUSCH
                          ;
                          ; VERSION: EXAMPLE
                          ;
                          .IDENT  RTC442          ;MODULE ID
                          ;
                          .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                          ;
    0010                  IOBASE  = 10H           ;SERIAL/PARALLEL I/O PORT BASE
                          ;
    0016                  TIM2    = IOBASE+06H    ;TIMER 2 DATA REGISTER
    0017                  TIMCTL  = IOBASE+07H    ;TIMER CONTROL REGISTER
    0018                  SINTE   = IOBASE+08H    ;SERIAL INTERRUPT ENABLE REGISTER
    0019                  T2RES   = IOBASE+09H    ;TIMER 2 INTERRUPT RESET
                          ;
    0001                  RTCENA  = 1             ;REAL TIME CLOCK ENABLE BIT
                          ;
    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 3EC3          RTCNIT::MVI     A,JMP   ;INIT RTC INTERRUPT VECTOR ADDR
    0002:04 32 0008               STA     1*8
    0005:04 21 0000'              LXI     H,RTCISR
    0008:04 22 0009               SHLD    (1*8)+1
    000B:04 3EB6                  MVI     A,T2CMD ;GET TIMER 2 COMMAND
    000D:04 D317                  OUT     TIMCTL  ;SELECT TIMER 2
    000F:04 21 0000:05            LXI     H,RTCCNT#  ;GET RTC COUNTER VALUE
    0012:04 7D                    MOV     A,L     ;GET LSB OF TIMER VALUE
    0013:04 D316                  OUT     TIM2    ;OUTPUT IT TO TIMER 2 DATA REGISTE
    0015:04 7C                    MOV     A,H     ;GET MSB OF TIMER VALUE
    0016:04 D316                  OUT     TIM2    ;OUTPUT IT TO TIMER 2 DATA REGISTE
    0018:04 21 0000:06            LXI     H,INTMSK#  ;GET INTERRUPT MASK
    001B:04 CBCE                  SET     RTCENA,M  ;SET RTC INTERRUPT ENABLE BIT
    001D:04 3A 0000:06            LDA     INTMSK  ;GET INTERRUPT MASK
    0020:04 D318                  OUT     SINTE   ;ENABLE RTC INTERRUPT MASK
    0022:04 C9                    RET             ;DONE
                          ;
```

```
NIT740 - TURBODOS OPERATING SYSTEM HARDWARE INITIALIZATION (IMS 740)
COPYRIGHT (C) 1982, SOFTWARE 2000. INC.

                           ;
                           ; ***************************************************
                           ; *                                                 *
                           ; *  This is the hardware initialization routine for *
                           ; *  an IMS 740 slave processor board.  It consists  *
                           ; *  of calls to the initialization entrypoints of   *
                           ; *  other driver modules.                           *
                           ; *                                                 *
                           ; ***************************************************
                           ;
                           ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                           ;
                           ; AUTHORS: RONALD E. RAIKES
                           ;          MICHAEL D. BUSCH
                           ;
                           ; VERSION: EXAMPLE
                           ;
                           .IDENT  NIT740          ;MODULE ID
                           ;
                           .INSERT DREQUATE        ;DRIVER SYMBOLIC EQUIVALENCES
                           ;
  0000:04                         .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                           ;
  0000:04 CD 0000:05       HDWNIT::CALL    SPINIT# ;INITIALIZE SERIAL/PARALLEL I/O
  0003:04 C3 0000:06              JMP     CKTINA# ;INITIALIZE CIRCUIT DRIVER A
                           ;
                           .END
```

```
                             ;
  0000:04                            .LOC     .INIT.# ;LOCATE IN INITIALIZATION AREA
                             ;
  0000:04 21 0041'           CKTIN$::LXI      H,PIOISR  ;GET PIO A INTERRUPT SERVICE ADD
  0003:04 22 0010                    SHLD     PIOVEC  ;SET INTERRUPT VECTOR ADDRESS
  0006:04 3E10                       MVI      A,PIOVEC  ;GET PIO A INTERRUPT VECTOR
  0008:04 D326                       OUT      PIOACR  ;OUTPUT INTERRUPT SERVICE VECTOR
  000A:04 3ECF                       MVI      A,0CFH  ;GET PIO A MODE WORD (MODE 3)
  000C:04 D326                       OUT      PIOACR  ;OUTPUT PIO A MODE WORD
  000E:04 3EFF                       MVI      A,0FFH  ;GET PIO A MODE 3 MODE WORD
  0010:04 D326                       OUT      PIOACR  ;OUTPUT PIO A MODE 3 MODE WORD
  0012:04 3EB7                       MVI      A,0B7H  ;GET PIO A INTERRUPT CONTROL WORD
  0014:04 D326                       OUT      PIOACR  ;OUTPUT INTERRUPT CONTROL WORD
  0016:04 3EEF                       MVI      A,0EFH  ;GET PIO A INTERRUPT MASK
  0018:04 D326                       OUT      PIOACR  ;OUTPUT INTERRUPT MASK
  001A:04 3E30                       MVI      A,30H   ;GET PIO B DATA DATA WORD
  001C:04 D325                       OUT      PIOBDR  ;OUTPUT PIO B DATA WORD
  001E:04 3E0F                       MVI      A,0FH   ;GET PIO B MODE WORD (MODE 0)
  0020:04 D327                       OUT      PIOBCR  ;OUTPUT PIO B MODE WORD
  0022:04 DB20                       IN       PORTAD  ;CLEAR INPUT PORT A
  0024:04 3A 0000:05                 LDA      NMBCKT# ;GET NUMBER OF CIRCUITS
  0027:04 47                         MOV      B,A     ;NUMBER OF CIRCUITS TO B-REG
  0028:04 21 0001:06                 LXI      H,CKTAST#+1  ;GET CIRCUIT ASSIGNMENT TABLE
  002B:04 3A 0081                    LDA      TBUF+1  ;GET MSB OF PASSED DESTINATION ID
  002E:04 BE                 ..SL:   CMP      M       ;CIRCUIT NUMBERS EQUAL?
  002F:04 2807                       JRZ      ..DIDF  ;IF SO, CONTINUE
  0031:04 23                         INX      H       ;ELSE, ADVANCE TO NEXT TABLE ENTRY
  0032:04 23                         INX      H
  0033:04 23                         INX      H
  0034:04 23                         INX      H
  0035:04 10F7                       DJNZ     ..SL    ;CONTINUE
  0037:04 C9                         RET              ;DONE
  0038:04 2B                 ..DIDF: DCX      H       ;BACK UP TO LSB OF DESTINATION ID
  0039:04 3A 0080                    LDA      TBUF    ;GET LSB OF PASSED DESTINATION ID
  003C:04 77                         MOV      M,A     ;SET LSB OF DESTINATION ID
  003D: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'   2804                       JRZ      SLVRCV  ;IF SO, CONTINUE
  0004'   3D                         DCR      A       ;FUNCTION NUMBER=1?
  0005'   281C                       JRZ      SLVSND  ;IF SO, CONTINUE
  0007'   C9                         RET              ;ELSE, DONE
                             ;
  0008'   21 0000"           SLVRCV: LXI      H,RCVSPH  ;GET RECEIVE MESSAGE SEMAPHORE
  000B'   CD 0000:07                 CALL     WAIT#   ;WAIT FOR RECEIVE MESSAGE
  000E'   CD 003B'                   CALL     SRMCOM  ;DO COMMON SETUP
  0011'   0601                       MVI      B,1     ;GET LENGTH OF MESSAGE LENGTH
  0013'   CD 001A'                   CALL     ..RCVL  ;RECEIVE MESSAGE LENGTH
  0016'   2B                         DCX      H       ;BACK UP TO MESSAGE LENGTH
  0017'   46                         MOV      B,M     ;GET MESSAGE LENGTH
  0018'   23                         INX      H       ;RESTORE MESSAGE BUFFER ADDRESS
```

```
                    ;
                    .END
```

```
0003                  DCD     = 3             ;DATA CARRIER DETECT BIT
0005                  CTS     = 5             ;CLEAR TO SEND BIT
                      ;
0000"                         .LOC    .DATA.# ;LOCATE IN DATA AREA
                      ;
0000"   0040          S0IBSZ::.WORD   64      ;SERIAL 0 INPUT BUFFER SIZE
0002"   0000          S0IBUF: .WORD   0       ;SERIAL 0 INPUT BUFFER ADDRESS
0004"   0000          S0IPTR: .WORD   0       ;SERIAL 0 INPUT POINTER
0006"   0000          S0OPTR: .WORD   0       ;SERIAL 0 OUTPUT POINTER
0008"   0000          S0ICNT: .WORD   0       ;SERIAL 0 INPUT COUNT
000A"   00            S0OCHR: .BYTE   0       ;SERIAL 0 OUTPUT CHARACTER
000B"   00            S0BR:   .BYTE   0       ;SERIAL 0 BAUD RATE CODE
                      ;
000C"                 S0ISPH:                 ;SERIAL 0 INPUT SEMAPHORE
000C"   0000                  .WORD   0       ;SEMAPHORE COUNT
000E"   000E"         ..S0IH: .WORD   ..S0IH  ;SEMAPHORE P/D HEAD
0010"   000E"                 .WORD   ..S0IH
                      ;
                                              ;SERIAL 0 OUTPUT SEMAPHORE
0012"   0000          S0OSPH: .WORD   0       ;SEMAPHORE COUNT
0014"   0014"         ..S0OH: .WORD   ..S0OH  ;SEMAPHORE P/D HEAD
0016"   0014"                 .WORD   ..S0OH
                      ;
                                              ;SERIAL 0 OUTPUT SEMAPHORE
0018"   0001          S0XSPH: .WORD   1       ;SEMAPHORE COUNT
001A"   001A"         ..S0XH: .WORD   ..S0XH  ;SEMAPHORE P/D HEAD
001C"   001A"                 .WORD   ..S0XH
                      ;
001E"   0010          S1IBSZ::.WORD   16      ;SERIAL 1 INPUT BUFFER SIZE
0020"   0000          S1IBUF: .WORD   0       ;SERIAL 1 INPUT BUFFER ADDRESS
0022"   0000          S1IPTR: .WORD   0       ;SERIAL 1 INPUT POINTER
0024"   0000          S1OPTR: .WORD   0       ;SERIAL 1 OUTPUT POINTER
0026"   0000          S1ICNT: .WORD   0       ;SERIAL 1 INPUT COUNT
0028"   00            S1OCHR: .BYTE   0       ;SERIAL 1 OUTPUT CHARACTER
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
                      ;
                                              ;SERIAL 1 OUTPUT SEMAPHORE
0036"   0001          S1XSPH: .WORD   1       ;SEMAPHORE COUNT
0038"   0038"         ..S1XH: .WORD   ..S1XH  ;SEMAPHORE P/D HEAD
003A"   0038"                 .WORD   ..S1XH
                      ;
0000:04                       .LOC    .INIT.# ;LOCATE IN INITIALIZATION AREA
                      ;
0000:04 AF            SPINIT::XRA     A
```

```
0005'   282E                    JRZ     SERIN   ;IF SO, CONTINUE
0007'   3D                      DCR     A       ;FUNCTION NUMBER=2?
0008'   CA 0097'                JZ      SEROUT  ;IF SO, CONTINUE
000B'   3D                      DCR     A       ;FUNCTION NUMBER=3?
000C'   CA 01F9'                JZ      SERSBR  ;IF SO, CONTINUE
000F'   3D                      DCR     A       ;FUNCTION NUMBER=4?
0010'   CA 022B'                JZ      SERRBR  ;IF SO, CONTINUE
0013'   3D                      DCR     A       ;FUNCTION NUMBER=5?
0014'   CA 0237'                JZ      SERSMC  ;IF SO, CONTINUE
0017'   3D                      DCR     A       ;FUNCTION NUMBER=6?
0018'   CA 0257'                JZ      SERRMC  ;IF SO, CONTINUE
001B'   C9                      RET             ;ELSE, DCNE
                        ;
001C'   ED5B 0008"      SERST:  LDED    S0ICNT  ;GET SERIAL 0 INPUT BUFFER COUNT
0020'   2A 0006"                LHLD    S0OPTR  ;GET SERIAL 0 OUTPUT POINTER
0023'   78                      MOV     A,B     ;GET CHANNEL NUMBER
0024'   B7                      ORA     A       ;CHANNEL NUMBER=0
0025'   2807                    JRZ     ..COM   ;IF SO, CONTINUE
0027'   ED5B 0026"              LDED    S1ICNT  ;GET SERIAL 1 INPUT BUFFER COUNT
002B'   2A 0024"                LHLD    S1OPTR  ;GET SERIAL 1 OUTPUT POINTER
002E'   7A              ..COM:  MOV     A,D
002F'   B3                      ORA     E       ;SERIAL INPUT BUFFER COUNT=0?
0030'   C8                      RZ              ;IF SO, DONE
0031'   3EFF                    MVI     A,0FFH  ;ELSE, SET RETURN CODE=0FFH
0033'   4E                      MOV     C,M     ;GET SERIAL INPUT CHARACTER
0034'   C9                      RET             ;DONE
                        ;
0035'   CD 001C'        SERIN:  CALL    SERST   ;GET SERIAL STATUS
0038'   B7                      ORA     A       ;CHARACTER AVAILABLE?
0039'   78                      MOV     A,B     ;GET CHANNEL NUMBER
003A'   200E                    JRNZ    ..SIN   ;IF CHARACTER AVAILABLE, CONTINUE
003C'   21 000C"                LXI     H,S0ISPH  ;GET SERIAL 0 INPUT SEMAPHORE
003F'   B7                      ORA     A       ;CHANNEL NUMBER=0?
0040'   2803                    JRZ     ..INC   ;IF SO, CONTINUE
0042'   21 002A"                LXI     H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
0045'   CD 0000:06      ..INC:  CALL    WAIT#   ;WAIT FOR INPUT CHARACTER
0048'   18EB                    JMPR    SERIN   ;CONTINUE
004A'   B7              ..SIN:  ORA     A       ;CHANNEL NUMBER=0?
004B'   2025                    JRNZ    ..S1I   ;IF NOT, CONTINUE
004D'   F3                      DI              ;DISABLE INTERRUPTS
004E'   2A 0008"                LHLD    S0ICNT  ;GET SERIAL 0 INPUT COUNT
0051'   2B                      DCX     H       ;DECREMENT SERIAL 0 INPUT COUNT
0052'   22 0008"                SHLD    S0ICNT  ;UPDATE SERIAL 0 INPUT COUNT
0055'   2A 0006"                LHLD    S0OPTR  ;GET SERIAL 0 OUTPUT POINTER
0058'   7E                      MOV     A,M     ;GET CHARACTER FROM BUFFER
0059'   23                      INX     H       ;INCREMENT SERIAL 0 OUTPUT POINTER
005A'   EB                      XCHG            ;SERIAL 0 OUTPUT POINTER TO DE-REG
005B'   2A 0002"                LHLD    S0IBUF  ;GET SERIAL 0 INPUT BUFFER ADDRESS
005E'   ED4B 0000"              LBCD    S0IBSZ  ;GET SERIAL 0 INPUT BUFFER SIZE
0062'   0B                      DCX     B       ;DECREMENT INPUT BUFFER SIZE
0063'   09                      DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
0064'   ED52                    DSBC    D       ;BUFFER WRAP-AROUND?
0066'   3004                    JRNC    ..NWA0  ;IF NOT, CONTINUE
0068'   ED5B 0002"              LDED    S0IBUF  ;GET SERIAL 0 INPUT BUFFER ADDRESS
```

```
  00DB'   3E10            SOOPR:  MVI     A,10H    ;GET RESET EXTERNAL STATUS COMMAND
  00DD'   D32E                    OUT     SIOACR   ;RESET EXTERNAL STATUS
  00DF'   DB2E                    IN      SIOACR   ;GET SIO PORT A STATUS
  00E1'   CB57                    BIT     TBE,A    ;TRANSMIT BUFFER EMPTY?
  00E3'   C8                      RZ               ;IF NOT, DONE
  00E4'   21 000B"                LXI     H,SOBR   ;ELSE, GET SERIAL 0 BAUD RATE CODE
  00E7'   CB76                    BIT     6,M      ;CTS HANDSHAKING REQUESTED?
  00E9'   2803                    JRZ     ..NCTS   ;IF NOT, CONTINUE
  00EB'   CB6F                    BIT     CTS,A    ;ELSE, CHECK CLEAR TO SEND STATUS
  00ED'   C8                      RZ               ;IF CLEAR TO SEND FALSE, DONE
  00EE'   3A 000A"        ..NCTS: LDA     SOOCHR   ;GET SERIAL 0 OUTPUT CHARACTER
  00F1'   D32C                    OUT     SIOADR   ;OUTPUT CHARACTER
  00F3'   21 00D7'                LXI     H,SOOPOL  ;GET SERIAL 0 OUT POLL ROUTINE
  00F6'   CD 0000:09              CALL    UNLINK# ;UNLINK POLL ROUTINE
  00F9'   21 0012"                LXI     H,SOOSPH  ;GET SERIAL 0 OUT SEMAPHORE
  00FC'   C3 0000:08              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                          ;
  00FF'                   S1OPOL:                  ;SERIAL 1 OUTPUT POLL ROUTINE
  00FF'   0000                    .WORD   0        ;SUCCESSOR LINK POINTER
  0101'   0000                    .WORD   0        ;PREDECESSOR LINK POINTER
                          ;
  0103'   3E10            S1OPR:  MVI     A,10H    ;GET RESET EXTERNAL STATUS COMMAND
  0105'   D32F                    OUT     SIOBCR   ;RESET EXTERNAL STATUS
  0107'   DB2F                    IN      SIOBCR   ;GET SIO PORT B STATUS
  0109'   CB57                    BIT     TBE,A    ;TRANSMIT BUFFER EMPTY?
  010B'   C8                      RZ               ;IF NOT, DONE
  010C'   21 0029"                LXI     H,S1BR   ;ELSE, GET SERIAL 1 BAUD RATE CODE
  010F'   CB76                    BIT     6,M      ;CTS HANDSHAKING REQUESTED?
  0111'   2803                    JRZ     ..NCTS   ;IF NOT, CONTINUE
  0113'   CB6F                    BIT     CTS,A    ;ELSE, CHECK CLEAR TO SEND STATUS
  0115'   C8                      RZ               ;IF CLEAR TO SEND FALSE, DONE
  0116'   3A 0028"        ..NCTS: LDA     S1OCHR   ;GET SERIAL 1 OUTPUT CHARACTER
  0119'   D32D                    OUT     SIOBDR   ;OUTPUT CHARACTER
  011B'   21 00FF'                LXI     H,S1OPOL  ;GET SERIAL 1 OUT POLL ROUTINE
  011E'   CD 0000:09              CALL    UNLINK# ;UNLINK POLL ROUTINE
  0121'   21 0030"                LXI     H,S1OSPH  ;GET SERIAL 1 OUT SEMAPHORE
  0124'   C3 0000:08              JMP     SIGNAL# ;SIGNAL PROCESS AS READY
                          ;
  0127'   ED73 0000:0A    SIOISR: SSPD    INTSP#   ;SAVE STACK POINTER
  012B'   31 0000:0B              LXI     SP,INTSTK#  ;SET UP AUX STACK POINTER
  012E'   F5                      PUSH    PSW      ;SAVE REGISTERS
  012F'   C5                      PUSH    B
  0130'   D5                      PUSH    D
  0131'   E5                      PUSH    H
  0132'   CD 0143'                CALL    ..SOI    ;CHECK FOR SERIAL 0 INPUT
  0135'   CD 019C'                CALL    ..S1I    ;CHECK FOR SERIAL 1 INPUT
  0138'   E1                      POP     H        ;RESTORE REGISTERS
  0139'   D1                      POP     D
  013A'   C1                      POP     B
  013B'   F1                      POP     PSW
  013C'   ED7B 0000:0A            LSPD    INTSP#   ;RESTORE STACK POINTER
  0140'   C3 0000:0C              JMP     ISRXIT# ;CONTINUE
  0143'   DB2E            ..SOI:  IN      SIOACR   ;GET SIO PORT A STATUS
  0145'   CB47                    BIT     RDA,A    ;CHARACTER AVAILABLE
```

```
   01BC'    22 0024"             SHLD    S1OPTR  ;RESET SERIAL 1 OUTPUT POINTER
   01BF'    21 0000              LXI     H,0
   01C2'    22 0026"             SHLD    S1ICNT  ;SET SERIAL 1 INPUT COUNT=0
   01C5'    2A 001E"     ..NAD1: LHLD    S1IBSZ  ;GET SERIAL 1 INPUT BUFFER SIZE
   01C8'    ED5B 0026"           LDED    S1ICNT  ;GET SERIAL 1 INPUT COUNT
   01CC'    13                   INX     D       ;INCREMENT SERIAL 1 INPUT COUNT
   01CD'    B7                   ORA     A       ;CLEAR CARRY FLAG
   01CE'    ED52                 DSBC    D       ;SERIAL 1 INPUT BUFFER FULL?
   01D0'    D8                   RC              ;IF SO, DONE
   01D1'    ED53 0026"           SDED    S1ICNT  ;ELSE, UPDATE SERIAL 1 INPUT COUNT
   01D5'    2A 0022"             LHLD    S1IPTR  ;GET SERIAL 1 INPUT POINTER
   01D8'    71                   MOV     M,C     ;STORE INPUT CHARACTER IN BUFFER
   C1D9'    23                   INX     H       ;INCREMENT INPUT POINTER
   01DA'    EB                   XCHG            ;INPUT POINTER TO DE-REG
   01DB'    2A 001E"             LHLD    S1IBSZ  ;GET SERIAL 1 INPUT BUFFER SIZE
   01DE'    2B                   DCX     H       ;DECREMENT INPUT BUFFER SIZE
   01DF'    ED4B 0020"           LBCD    S1IBUF  ;GET SERIAL 1 INPUT BUFFER ADDRESS
   01E3'    09                   DAD     B       ;CALC LAST INPUT BUFFER ADDRESS
   01E4'    ED52                 DSBC    D       ;BUFFER WRAP-AROUND?
   01E6'    3004                 JRNC    ..NWA1  ;IF NOT, CONTINUE
   01E8'    ED5B 0020"           LDED    S1IBUF  ;GET SERIAL 1 INPUT BUFFER ADDRESS
   01EC'    ED53 0022"   ..NWA1: SDED    S1IPTR  ;UPDATE SERIAL 1 INPUT POINTER
   01F0'    21 002A"             LXI     H,S1ISPH  ;GET SERIAL 1 INPUT SEMAPHORE
   01F3'    7E           ..X:    MOV     A,M     ;GET SEMAPHORE COUNT
   01F4'    B7                   ORA     A       ;SEMAPHORE COUNT=0?
   01F5'    C8                   RZ              ;IF SO, DONE
   01F6'    C3 0000:08           JMP     SIGNAL# ;ELSE, SIGNAL PROCESS AS READY
                         ;
   01F9'    51           SERSBR: MOV     D,C     ;REQUESTED BAUD RATE TO D-REG
   01FA'    21 000B"             LXI     H,S0BR  ;GET SERIAL 0 BAUD RATE
   01FD'    0E28                 MVI     C,CTCCH0  ;GET CTC CHANNEL 0 REGISTERS
   01FF'    78                   MOV     A,B     ;GET CHANNEL NUMBER
   0200'    B7                   ORA     A       ;CHANNEL NUMBER=0?
   0201'    2805                 JRZ     ..COM   ;IF SO, CONTINUE
   0203'    21 0029"             LXI     H,S1BR  ;ELSE, GET SERIAL 1 BAUD RATE
   0206'    0E29                 MVI     C,CTCCH1  ;GET CTC CHANNEL 1 REGISTERS
   0208'    72           ..COM:  MOV     M,D     ;SAVE BAUD RATE CODE
   0209'    3E47                 MVI     A,47H   ;GET CTC CHANNEL CONTROL WORD
   020B'    ED79                 OUTP    A       ;INITIALIZE CTC CHANNEL
   020D'    7A                   MOV     A,D     ;GET REQUESTED BAUD RATE CODE
   020E'    E60F                 ANI     0FH     ;LIMIT TO 16 BAUD RATES
   0210'    5F                   MOV     E,A     ;TO E-REG
   0211'    1600                 MVI     D,0     ;MAKE IT DOUBLE LENGTH
   0213'    21 021B'             LXI     H,BRTBL  ;GET BAUD RATE TABLE
   0216'    19                   DAD     D       ;INDEX INTO TABLE
   0217'    7E                   MOV     A,M     ;GET TIMER VALUE
   0218'    ED79                 OUTP    A       ;SET CTC CHANNEL TIME CONSTANT
   021A'    C9                   RET             ;DONE
                         ;
   021B'    00           BRTBL:  .BYTE   0       ;50 BAUD TIMER VALUE
   021C'    00                   .BYTE   0       ;75 BAUD TIMER VALUE
   021D'    AF                   .BYTE   175     ;110 BAUD TIMER VALUE
   021E'    8F                   .BYTE   143     ;134.5 BAUD TIMER VALUE
   021F'    80                   .BYTE   128     ;150 BAUD TIMER VALUE
   026F'   CBEF                        SET      5,A       ;ELSE, SET DCD BIT
   0271'   C9                          RET                ;DONE
                            ;
                            .END
```

```
MPB401 - TURBODOS OPERATING SYSTEM MASTER PROCESSOR BOOT FOR IMS 401
COPYRIGHT (C) 1982, SOFTWARE 2000, INC.

                    ;
                    ; ********************************************************
                    ; *                                                      *
                    ; *  This is a boot ROM driver module which interfaces   *
                    ; *  with the IMS 401 eight-inch floppy disk controller  *
                    ; *  board.  This driver is linked together with two     *
                    ; *  other modules, OSBOOT.REL and DST8F.REL, to create  *
                    ; *  a boot ROM for an IMS 8000 system.                  *
                    ; *                                                      *
                    ; ********************************************************
                    ;
                    ; COPYRIGHT (C) 1982, SOFTWARE 2000, INC.
                    ;
                    ; AUTHORS: RONALD E. RAIKES
                    ;          MICHAEL D. BUSCH
                    ;
                    ; VERSION: EXAMPLE
                    ;
                    .IDENT  MPB401          ;MODULE ID
                    ;
                    .INSERT EQUATE          ;DRIVER SYMBOLIC EQUIVALENCES
                    ;
  0080              RAM     =: TBUF         ;WORKING STORAGE ADDRESS
  0040              RAMLEN  = 64            ;WORKING STORAGE LENGTH
                    ;
  0082              CH1DMA  = 82H           ;CHANNEL 1 DMA REGISTER (FDC)
  0083              CH1TC   = 83H           ;CHANNEL 1 TERMINAL COUNT (FDC)
  0088              DMACTL  = 88H           ;DMA COMMAND AND STATUS REGISTERS
  008A              DSKSEL  = 8AH           ;DISK SELECT PORT
  008C              DSKCTL  = 8CH           ;STATUS AND INT MASK (BOARD)
  008E              FDCST   = 8EH           ;DISK CONTROLLER STATUS (uPD-765)
  008F              FDCDAT  = 8FH           ;DISK CONTROLLER DATA (uPD-765)
                    ;
  0042              CH1ENA  = 42H           ;DMA CHANNEL 1 ENABLE COMMAND
  0000              DMAVFY  = 00H           ;DMA VERIFY COMMAND
  0040              DMARD   = 40H           ;DMA READ COMMAND
  0080              DMAWR   = 80H           ;DMA WRITE COMMAND
                    ;
  0003              FDCSFY  = 03H           ;FDC SPECIFY COMMAND
  0004              FDCSDS  = 04H           ;FDC SENSE DRIVE STATUS COMMAND
  0007              FDCRCL  = 07H           ;FDC RECALIBRATE COMMAND
  0008              FDCSIS  = 08H           ;FDC SENSE INTERRUPT STATUS COMMAN
  000A              FDCRID  = 0AH           ;FDC READ ID COMMAND
  000F              FDCSK   = 0FH           ;FDC SEEK COMMAND
  0085              FDCWR   = 85H           ;FDC WRITE COMMAND
  0086              FDCRD   = 86H           ;FDC READ COMMAND
                    ;
  0000              DSKENI  = 0             ;DISK CONTROLLER ENABLE INTERRUPTS
  0007              DSKDLC  = 7             ;DISK CONTROLLER DELAY COMPLETE
                    ;
  0006              FDCMFM  = 6             ;FDC DOUBLE-DENSITY BIT
                    ;
  0004              FDCBSY  = 4             ;FDC BUSY STATUS
  0005              FDCSE   = 5             ;FDC SEEK END
```

```
                              ;
 0000'                                .LOC     .PROG.# ;LOCATE IN PROGRAM AREA
                              ;
 0000'    3E03                INIT::  MVI      A,FDCSFY  ;GET FDC SPECIFY COMMAND
 0002'    CD 01B3'                    CALL     DATOUT  ;OUTPUT FDC SPECIFY COMMAND
 0005'    3EA1                        MVI      A,SRT8S!HUT  ;GET STEP RATE/HEAD UNLD TIME
 0007'    CD 01B3'                    CALL     DATOUT  ;OUTPUT STEP RATE/HEAD UNLD TIME
 000A'    3E24                        MVI      A,HLT   ;GET HEAD LOAD TIME/NON-DMA BIT
 000C'    CD 01B3'                    CALL     DATOUT  ;OUTPUT HEAD LOAD TIME/NON-DMA BIT
 000F'    21 0100                     LXI      H,TPA   ;GET LOAD BASE ADDRESS
 0012'    C9                          RET              ;DONE
                              ;
 0013'    FE04                SELECT::CPI      4       ;TEST FOR VALID DRIVE
 0015'    3061                        JRNC     ..NR    ;IF INVALID DRIVE, CONTINUE
 0017'    32 00C0                     STA      IODSK   ;ELSE, SET DISK NUMBER
 001A'    4F                          MOV      C,A     ;DISK NUMBER TO C-REG
 001B'    DB8C                        IN       DSKCTL  ;GET DISK CONTROLLER STATUS
 001D'    0F                          RRC              ;EXTRACT SELECTED DRIVE
 001E'    E603                        ANI      3
 0020'    B9                          CMP      C       ;DRIVE ALREADY SELECTED?
 0021'    2803                        JRZ      ..DS    ;IF SO, CONTINUE
 0023'    79                          MOV      A,C     ;ELSE, GET DISK NUMBER
 0024'    D38A                        OUT      DSKSEL  ;SELECT DISK NUMBER
 0026'    01 0014             ..DS:   LXI      B,20    ;DELAY 10 MILLISECONDS
 0029'    10FE                ..DLY:  DJNZ     ..DLY
 002B'    0D                          DCR      C
 002C'    20FB                        JRNZ     ..DLY
 002E'    3E04                        MVI      A,FDCSDS  ;GET FDC SENSE DRIVE STATUS CMD
 0030'    CD 01B3'                    CALL     DATOUT  ;OUTPUT COMMAND TO FDC
 0033'    3A 00C0                     LDA      IODSK   ;GET DISK NUMBER
 0036'    CD 01B3'                    CALL     DATOUT  ;OUTPUT IT TO FDC
 0039'    CD 01A7'                    CALL     DATAIN  ;GET STATUS REGISTER 3
 003C'    CB6F                        BIT      ST3RDY,A  ;DRIVE READY?
 003E'    2838                        JRZ      ..NR    ;IF NOT READY, CONTINUE
 0040'    32 00C7                     STA      ST3REG  ;ELSE, SAVE STATUS REGISTER 3
 0043'    CD 0165'                    CALL     RECAL   ;RECALIBRATE DRIVE
 0046'    2030                        JRNZ     ..NR    ;IF ERRORS, CONTINUE
 0048'    0E00                        MVI      C,0     ;ELSE, GET INITIAL TYPE VALUE
 004A'    21 00C7                     LXI      H,ST3REG  ;GET STATUS REGISTER 3
 004D'    CB5E                        BIT      ST3TS,M ;ONE-SIDED DISK?
 004F'    2802                        JRZ      ..OSD   ;YES
 0051'    CBD1                        SET      TSD,C   ;SET TWO-SIDED DISK BIT
 0053'    3E0A                ..OSD:  MVI      A,FDCRID  ;GET FDC READ ID COMMAND (SD)
 0055'    CD 0089'                    CALL     ..RID   ;ATTEMPT TO READ SINGLE-DENSITY
 0058'    2809                        JRZ      ..TPC   ;IF SINGLE-DENSITY, DONE
 005A'    3E4A                        MVI      A,FDCRID!1<FDCMFM  ;GET READ ID CMD (DD)
 005C'    CD 0089'                    CALL     ..RID   ;ATTEMPT TO READ DOUBLE-DENSITY
 005F'    2017                        JRNZ     ..NR    ;IF NOT DOUBLE-DENSITY, DONE
 0061'    CBD9                        SET      DDD,C   ;SET DOUBLE-DENSITY DISK BIT
 0063'    B1                  ..TPC:  ORA      C       ;ADD SECTOR SIZE TO TYPE CODE
 0064'    4F                          MOV      C,A     ;SAVE TYPE CODE IN C-REG
 0065'    11 0000:04                  LXI      D,DSTBLS#  ;GET DST BASE ADDRESS
 0068'    79                  ..SL:   MOV      A,C     ;GET DISK TYPE CODE
 0069'    21 0000:05                  LXI      H,DTCO# ;GET OFFSET TO DISK TYPE CODE
```

```
   00CC'   D382                OUT     CH1DMA  ;OUTPUT LSB OF DMA ADDRESS
   00CE'   7C                  MOV     A,H     ;GET MSB OF DMA ADDRESS
   00CF'   D382                OUT     CH1DMA  ;OUTPUT MSB OF DMA ADDRESS
   00D1'   3E42                MVI     A,CH1ENA  ;GET CHANNEL 1 ENABLE COMMAND
   00D3'   D388                OUT     DMACTL  ;ENABLE DMA CONTROLLER
   00D5'   3E86                MVI     A,FDCRD ;GET FDC READ COMMAND
   00D7'   21 00D6             LXI     H,TYPCOD  ;GET DISK TYPE CODE
   00DA'   CB5E                BIT     DDD,M   ;SINGLE DENSITY DISK?
   00DC'   2802                JRZ     ..SD    ;IF SO, CONTINUE
   00DE'   CBF7                SET     FDCMFM,A  ;ELSE, SET FDC MFM BIT
   00E0'   CD 01B3'    ..SD:   CALL    DATOUT  ;OUTPUT FDC READ COMMAND
   00E3'   3A 00C3             LDA     IOSEC   ;GET SECTOR NUMBER
   00E6'   5F                  MOV     E,A     ;SECTOR NUMBER TO E-REG
   00E7'   2A 00D4             LHLD    XLTBL   ;GET TRANSLATION TABLE ADDRESS
   00EA'   7C                  MOV     A,H
   00EB'   B5                  ORA     L       ;SECTOR TRANSLATION REQUIRED?
   00EC'   2804                JRZ     ..NI    ;IF NOT, CONTINUE
   00EE'   1600                MVI     D,0     ;ELSE, MAKE SECTOR DOUBLE LENGTH
   00F0'   19                  DAD     D       ;INDEX INTO TRANSLATION TABLE
   00F1'   5E                  MOV     E,M     ;GET TRANSLATED SECTOR NUMBER
   00F2'   1C          ..NI:   INR     E       ;CONVERT SECTOR TO BASE 1
   00F3'   3A 00CE             LDA     SECTRK  ;GET NUMBER OF SECTORS/TRACK
   00F6'   21 00D6             LXI     H,TYPCOD  ;GET DISK TYPE CODE ADDRESS
   00F9'   CB56                BIT     TSD,M   ;TWO SIDED DISK?
   00FB'   2802                JRZ     ..SSD   ;IF NOT, CONTINUE
   00FD'   CB3F                SRLR    A       ;ELSE, CALC NUMBER OF SECTORS/SIDE
   00FF'   57          ..SSD:  MOV     D,A     ;SAVE NUMBER OF SECTORS/SIDE
   0100'   0600                MVI     B,0     ;PRESET FOR FRONT SIDE
   0102'   BB                  CMP     E       ;FRONT SIDE OF DISK?
   0103'   3004                JRNC    ..FS1   ;IF SO, CONTINUE
   0105'   7B                  MOV     A,E     ;ELSE, GET SECTOR NUMBER
   0106'   92                  SUB     D       ;SUBTRACT ONE SIDES WORTH
   0107'   5F                  MOV     E,A     ;SECTOR NUMBER TO C-REG
   0108'   04                  INR     B       ;SET HEAD NUMBER=1
   0109'   3A 00C0     ..FS1:  LDA     IODSK   ;GET DISK NUMBER
   010C'   04                  INR     B
   010D'   05                  DCR     B       ;HEAD=0?
   010E'   2802                JRZ     ..FS2   ;IF SO, CONTINUE
   0110'   CBD7                SET     2,A     ;ELSE, SET HEAD BIT
   0112'   CD 01B3'    ..FS2:  CALL    DATOUT  ;OUTPUT UNIT NUMBER
   0115'   3A 00C1             LDA     IOTRK   ;GET TRACK NUMBER
   0118'   CD 01B3'            CALL    DATOUT  ;OUTPUT TRACK NUMBER
   011B'   78                  MOV     A,B     ;GET HEAD NUMBER
   011C'   CD 01B3'            CALL    DATOUT  ;OUTPUT HEAD NUMBER
   011F'   7B                  MOV     A,E     ;GET SECTOR NUMBER
   0120'   CD 01B3'            CALL    DATOUT  ;OUTPUT SECTOR NUMBER
   0123'   3A 00CD             LDA     SECSIZ  ;GET SECTOR SIZE
   0126'   F5                  PUSH    PSW     ;SAVE SECTOR SIZE
   0127'   CD 01B3'            CALL    DATOUT  ;OUTPUT SECTOR SIZE
   012A'   7A                  MOV     A,D     ;GET EOT
   012B'   CD 01B3'            CALL    DATOUT  ;OUTPUT EOT
   012E'   3A 00D7             LDA     GAPLEN  ;GET GAP LENGTH
   0131'   CD 01B3'            CALL    DATOUT  ;OUTPUT GAP LENGTH
   0134'   F1                  POP     PSW     ;RESTORE SECTOR SIZE
```

```
  01A1'    18D2                    JMPR    ..RQML  ;FLUSH ANY REMAINING INTERRUPTS
  01A3'    7A              ..X:    MOV     A,D     ;GET STATUS REGISTER 0
  01A4'    E6C0                    ANI     0C0H    ;EXTRACT COMPLETION STATUS
  01A6'    C9                      RET             ;DONE
                           ;
  01A7'    DB8E            DATAIN: IN      FDCST   ;GET FDC STATUS
  01A9'    07                      RLC             ;TEST FDC FOR READY
  01AA'    30FB                    JRNC    DATAIN  ;IF NOT READY, WAIT
  01AC'    07                      RLC             ;TEST FDC DIRECTION
  01AD'    D2 0000:06              JNC     .BEG.#  ;IF WRONG DIRECTION, CONTINUE
  01B0'    DB8F                    IN      FDCDAT  ;GET FDC DATA BYTE
  01B2'    C9                      RET             ;DONE
                           ;
  01B3'    4F              DATOUT: MOV     C,A     ;SAVE OUTPUT BYTE
  01B4'    DB8E            ..RW:   IN      FDCST   ;GET FDC STATUS
  01B6'    07                      RLC             ;TEST FDC FOR READY
  01B7'    30FB                    JRNC    ..RW    ;IF NOT READY, WAIT
  01B9'    07                      RLC             ;TEST FDC DIRECTION
  01BA'    DA 0000:06              JC      .BEG.#  ;IF WRONG DIRECTION, CONTINUE
  01BD'    79                      MOV     A,C     ;RESTORE OUTPUT BUTE
  01BE'    D38F                    OUT     FDCDAT  ;OUTPUT BYTE TO FDC DATA REGISTER
  01C0'    C9                      RET             ;DONE
                           ;
  01C1'    AF              XFER::  XRA     A
  01C2'    32 0080                 STA     TBUF    ;MAKE DEFAULT BUFFER EMPTY
  01C5'    C3 0100                 JMP     TPA     ;TRANSFER TO O/S LOADER
                           ;
                           .END
```

```
0O2C'   21 0035'              LXI     H,XFRCOD  ;GET TRANSFER CODE
002F'   01 0006               LXI     B,XFRLEN  ;GET TRANSFER CODE LENGTH
0032'   EDBO                  LDIR              ;MOVE TRANSFER CODE TO DMA ADDRESS
0034'   C9                    RET               ;TRANSFER TO TRANSFER CODE
                      ;
0035'   F3            XFRCOD: DI                ;DISABLE INTERRUPTS
0036'   3EB0                  MVI     A,0B0H    ;GET PIO B DATA BYTE
0038'   D325                  OUT     PIOBDR    ;DIABLE BOOTSTRAP EPROM
003A'   C9                    RET               ;DONE
                      ;
0006                  XFRLEN  = .-XFRCOD        ;TRANSFER CODE LENGTH
                      ;
003B'   CD 0046'      RCV740: CALL    INBYT     ;INPUT BYTE
003E'   77                    MOV     M,A       ;SAVE BYTE IN MESSAGE
003F'   23                    INX     H
0040'   1B                    DCX     D         ;DECREMENT COUNT
0041'   7A                    MOV     A,D
0042'   B3                    ORA     E         ;COUNT=0?
0043'   20F6                  JRNZ    RCV740    ;IF NOT, CONTINUE
0045'   C9                    RET               ;ELSE, DONE
                      ;
0046'   DB24          INBYT:  IN      PIOADR    ;GET PIO A DATA REGISTER
0048'   CB77                  BIT     PORTC7,A  ;PORT C BIT 7 SET? (OBF)
004A'   20FA                  JRNZ    INBYT     ;IF NOT, WAIT
004C'   DB20                  IN      PORTAD    ;INPUT BYTE FROM PORT A
004E'   C9                    RET               ;DONE
                      ;
0000'                 .END    SPBOOT
```

On page 2-20 (after NMBMBS paragraph), add:

> NMBRPS is a byte value that specifies the number of network reply packets to pre-allocate at cold-start time. This value may be left zero, but memory fragmentation may be reduced by assigning a positive value (see NMBMBS above).

On page 2-25, change item 3 as follows:

> single-user without spooling: .REL files for STDLOADR and STDSINGL; .COM files for AUTOLOAD, BACKUP, BOOT, BUFFERS, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MONITOR, PACKAGE, PRINT, RELCVT, RENAME, SET, SHOW, TYPE, USER and VERIFY; and .REL files for CPMSUP, OSBOOT, PATCH, RTCNUL, SUBMIT, and all necessary driver modules.
>
> single-user with spooling: .REL files for STDLOADR, STDSINGL and STDSPOOL; .COM files for AUTOLOAD, BACKUP, BOOT, BUFFERS, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MONITOR, PACKAGE, PRINT, PRINTER, QUEUE, RELCVT, RENAME, SET, SHOW, TYPE, USER and VERIFY; and .REL files for CPMSUP, OSBOOT, PATCH, RTCNUL, SUBMIT, and all necessary driver modules.
>
> multi-user networking: .REL files for STDLOADR, STDSINGL, STDSPOOL, STDMASTR, STDSLAVE and STDSLAVX; .COM files for AUTOLOAD, BACKUP, BATCH, BOOT, BUFFERS, CHANGE, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIFO, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MASTER, MONITOR, PACKAGE, PRINT, PRINTER, QUEUE, RECEIVE, RELCVT, RENAME, SEND, SET, SHOW, TYPE, USER and VERIFY; and .REL files for CONREM, CPMSUP, OSBOOT, PATCH, RTCNUL, SUBMIT, and all necessary driver modules.