Configuration Guide to TurboDOS

September, 1981

Copyright (C) 1981 by Software 2000 Inc.
Copyright (C) 1981 by Software 2000 Inc.
All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Software 2000 Inc., P.O. Box 945, Los Alamitos, California 90720.

Software 2000 Inc. makes no representations or warranties with respect to the contents of this publication, and specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Software 2000 Inc. shall under no circumstances be liable for consequential damages or related expenses, even if it has been notified of the possibility of such damages. Software 2000 Inc. reserves the right to revise this publication from time to time without obligation to notify any person of such revision.

NOTE: CP/M, MP/M and CP/NET are trademarks of Digital Research, Inc.
**TABLE OF CONTENTS**

**SECTION 1 — INTRODUCTION**
- Generating TurboDOS Configurations .................................................. 1-1
- Implementing Driver Modules ................................................................. 1-1
- Licensing Requirements ............................................................................. 1-2
- Serialization ............................................................................................... 1-3
- OEM Responsibilities ................................................................................... 1-3
- Dealer Responsibilities ............................................................................... 1-4
- TurboDOS Support ...................................................................................... 1-4

**SECTION 2 — SYSTEM GENERATION**
- Module Hierarchy ......................................................................................... 2-2
- Process-Level Modules ................................................................................ 2-4
- Kernel-Level Modules ................................................................................... 2-5
- Universal Driver-Level Modules ................................................................. 2-7
- Hardware-Dependent Driver-Level Modules ............................................. 2-8
- Standard Configurations ............................................................................. 2-8
- Estimating Memory Requirements ............................................................... 2-10
- Linking and Loading ..................................................................................... 2-11
- GEN Command ............................................................................................ 2-12
- Symbolic Patch Facility ............................................................................... 2-14
- Step-by-Step Procedure for System Generation .......................................... 2-18
- SERIAL Command ......................................................................................... 2-19
- Step-by-Step Procedure for OEM Re-Distribution ....................................... 2-20
SECTION 3 — SYSTEM IMPLEMENTATION

Assembler Requirements .............................................. 3-1

Programming Conventions:
  Dynamic Memory Allocation ........................................ 3-2
  Threaded Lists ...................................................... 3-4
  Dispatching .......................................................... 3-5
  Interrupt Service Routines ........................................ 3-7
  Poll Routines ...................................................... 3-8
  Re-Entrancy and Mutual Exclusion ................................ 3-9
  Sample Interrupt-Driven Device Driver ......................... 3-10
  Sample Polled Device Driver ..................................... 3-11

Driver Interface Specifications:
  Initialization .......................................................... 3-12
  Console Drivers ..................................................... 3-13
  Printer Drivers .................................................... 3-13
  Network Drivers .................................................... 3-14
  Disk Drivers ........................................................ 3-15
  Real-Time Clock Driver ............................................ 3-17
  Comm Channel Driver ............................................... 3-18

APPENDIX A — Implementation on IMS Equipment .................. A-1

APPENDIX B — Implementation on TRS-80 Model II .................. B-1

APPENDIX C — Sample Driver Listings ................................ C-1
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, time-sharing and networking, 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.

Implementing 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. 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 interface 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 competent 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 in both camera-ready form and 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
Introduction

that end-user.

Dealer Responsibilities

A TurboDOS dealer is permitted to purchase individual serialized copies of TurboDOS software and documentation from Software 2000 Inc. or from an authorized 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 technical assistance in the use of TurboDOS to its customers. OEMs and dealers 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 and dealer to provide technical support to its end-user customers. Software 2000 cannot assist end-users directly. Where exceptional circumstances seem to require direct contact between Software 2000 technical personnel and an end-user, this must be handled strictly by prior arrangement with Software 2000 by the OEM or dealer.
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 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 each local printer. 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 (more than 80 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.
Configuration Guide to TurboDOS
Copyright (C) 1981 by Software 2000 Inc.
System Generation

User's
Transient
Program

De-Spool
DSPOOL

Local User
LCLUSR
LCLTBL
CMDINT
AUTLOD
SGLUSR
AUTLOG

Process Level

Function Decode

OSNTRY

OSLOAD

Loader

Support

Clock

RTCNGR
DSPCHR
DSPSGL
MEMMGR
COMSUB

Net Svc
NETSVC
SLYTBL

Buffer Flush
FLUSHR

MSGMT

Function Decode

Non-File
NONFIL
SGLLOG

File
FILMGR
FILSUP
FILLOK
FFOMGR
DEVLOK
FASLOD
NORLOD

Net Req
NETREQ
MSGFMT
NETTBL

Kernel Level

Clock

RTCNGR

Support

Initialize

SYSNIT

Driver Level

Comm.
COMMGR
COMMDEV
CONREM

Printer
LSTDPR
CONDR@ or CONREM

Console
DOMGR
CONTRB
DSKDR@

Disk
DSKDR@

Network
NETDR@

Clock
RTCNGR
RTCNUL

TurboDOS Module Hierarchy

Comm.
COMMGR
COMMDEV
CONREM

Printer
LSTDPR
CONDR@ or CONREM

Console
DOMGR
CONTRB
DSKDR@

Disk
DSKDR@

Network
NETDR@

Clock
RTCNGR
RTCNUL

Initialize
HDWINIT

2-3
**Process-Level Modules**

**LCLUSR** — Supports a transient program area for a user of the local microcomputer. In multi-user configurations, there is a separate re-entrant instance of the LCLUSR process for each local user. This module may be omitted from a network master configuration where only remote (i.e., slave) users are desired.

**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).

**NETSVC** — Network service process which receives and services network requests from slave microcomputers. In network master configurations, there is a separate re-entrant instance of the NETSVC process for each attached slave.

**SLVTBL** — Table which controls down-loading of network slaves.

**MSGFMT** — Network message format tables used by NETSVC and NETREQ modules.

**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. Not required in single-user configurations in which SGLUSR is present.
**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. Not required in slave configurations which lack local disk storage.

**FILSUP** — File support routines required by FILMGR.

**FILLOK** — Multi-user file interlock routines called by FILMGR. Not required in single-user configurations.

**FFOMGR** — FIFO management routines called by FILLOK. Not required in single-user configurations.

**DEVLOK** — Multi-user device interlock routines called by FILMGR. Not required in single-user configurations.

**FASLOD** — Program load optimizer routine called by FILMGR.

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

**BUFMR** — 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** — Table of disk driver entrypoints and drive-letter-to-disk-number equivalences.
NONFIL — Non-file request manager which handles kernel requests which are not file-oriented.

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 of console driver entrypoints.

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 required by CMDINT.

LSTMGR — List manager which handles local printed output.

LSTTBL — Table of printer driver entrypoints.

SPOOLR — Spooler routine which diverts print output to spool files when the spooler is activated.

COMMGR — Comm channel manager which handles the communications channel.

NETREQ — Network request manager which passes appropriate kernel requests to the network to be satisfied by a network master. Required in network slave configurations.

MSGFMT — Network message format tables used by NETSVC and NETREQ modules. Required in both master and slave network configurations.

NETTBL — Table of network driver entrypoints.

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 required 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:

**CONDRV** -- Console driver allows character-by-character input from a console keyboard and output to a console display. TurboDOS supports multiple console drivers.

**LSTDVR** -- 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.

**DSKDRV** -- 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.

**NETDRV** -- Network interface driver allows sending and receiving messages to or from a remote microcomputer. TurboDOS supports multiple network interface drivers, each of which may communicate with multiple remote computers.

**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.

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

Standard Configurations

To simplify 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 and STDSLAVE. Most requirements for TurboDOS can be satisfied by linking the appropriate standard package together with the requisite driver modules.
## Standard TurboDOS Configurations

<table>
<thead>
<tr>
<th>Module</th>
<th>Approx. Size (K)</th>
<th>O/S Loader</th>
<th>Single User</th>
<th>Single User w/Spooling</th>
<th>Network Master</th>
<th>Network Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCLUSR</td>
<td>.9</td>
<td>-</td>
<td>LCLUSR</td>
<td>LCLUSR</td>
<td>LCLUSR</td>
<td>LCLUSR</td>
</tr>
<tr>
<td>LCLTBL</td>
<td>.0</td>
<td>-</td>
<td>LCLTBL</td>
<td>LCLTBL</td>
<td>LCLTBL</td>
<td>LCLTBL</td>
</tr>
<tr>
<td>CMDINT</td>
<td>.9</td>
<td>-</td>
<td>CMDINT</td>
<td>CMDINT</td>
<td>CMDINT</td>
<td>CMDINT</td>
</tr>
<tr>
<td>AUTLOD</td>
<td>.2</td>
<td>-</td>
<td>AUTLOD</td>
<td>AUTLOD</td>
<td>AUTLOD</td>
<td>AUTLOD</td>
</tr>
<tr>
<td>SGLUSR</td>
<td>.1</td>
<td>-</td>
<td>SGLUSR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AUTLOG</td>
<td>.0</td>
<td>-</td>
<td>AUTLOG</td>
<td>-</td>
<td>NETSVC</td>
<td>-</td>
</tr>
<tr>
<td>NETSVC</td>
<td>1.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SLYTBL</td>
<td>.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SLYTBL</td>
<td>-</td>
</tr>
<tr>
<td>DSPPOOL</td>
<td>.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DSPPOOL</td>
<td>DSPPOOL</td>
</tr>
<tr>
<td>FLUSR</td>
<td>.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLUSR</td>
</tr>
<tr>
<td>OSLOAD</td>
<td>1.2</td>
<td>OSLOAD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OSNTRY</td>
<td>.3</td>
<td>OSNTRY</td>
<td>OSNTRY</td>
<td>OSNTRY</td>
<td>OSNTRY</td>
<td>OSNTRY</td>
</tr>
<tr>
<td>FILMGR</td>
<td>1.2</td>
<td>FILMGR</td>
<td>FILMGR</td>
<td>FILMGR</td>
<td>FILMGR</td>
<td>-</td>
</tr>
<tr>
<td>FILSUP</td>
<td>2.1</td>
<td>FILSUP</td>
<td>FILSUP</td>
<td>FILSUP</td>
<td>FILSUP</td>
<td>-</td>
</tr>
<tr>
<td>FILLOK</td>
<td>.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FILLOK</td>
<td>-</td>
</tr>
<tr>
<td>FFOMGR</td>
<td>.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FFOMGR</td>
<td>-</td>
</tr>
<tr>
<td>DEVLOG</td>
<td>.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DEVLOG</td>
<td>-</td>
</tr>
<tr>
<td>FASLOD</td>
<td>.3</td>
<td>FASLOD</td>
<td>FASLOD</td>
<td>FASLOD</td>
<td>FASLOD</td>
<td>FASLOD</td>
</tr>
<tr>
<td>NORLOD</td>
<td>.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BUFMGR</td>
<td>1.0</td>
<td>BUFMGR</td>
<td>BUFMGR</td>
<td>BUFMGR</td>
<td>BUFMGR</td>
<td>-</td>
</tr>
<tr>
<td>DSKMGR</td>
<td>.5</td>
<td>DSKMGR</td>
<td>DSKMGR</td>
<td>DSKMGR</td>
<td>DSKMGR</td>
<td>-</td>
</tr>
<tr>
<td>DSKTBL</td>
<td>.0</td>
<td>DSKTBL</td>
<td>DSKTBL</td>
<td>DSKTBL</td>
<td>DSKTBL</td>
<td>DSKTBL</td>
</tr>
<tr>
<td>NONFIL</td>
<td>.2</td>
<td>NONFIL</td>
<td>NONFIL</td>
<td>NONFIL</td>
<td>NONFIL</td>
<td>NONFIL</td>
</tr>
<tr>
<td>SGLLOG</td>
<td>.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CONMGR</td>
<td>.1</td>
<td>CONMGR</td>
<td>CONMGR</td>
<td>CONMGR</td>
<td>CONMGR</td>
<td>CONMGR</td>
</tr>
<tr>
<td>CONTBL</td>
<td>.0</td>
<td>CONTBL</td>
<td>CONTBL</td>
<td>CONTBL</td>
<td>CONTBL</td>
<td>CONTBL</td>
</tr>
<tr>
<td>DOMGR</td>
<td>.5</td>
<td>DOMGR</td>
<td>DOMGR</td>
<td>DOMGR</td>
<td>DOMGR</td>
<td>DOMGR</td>
</tr>
<tr>
<td>INPLN</td>
<td>.1</td>
<td>INPLN</td>
<td>INPLN</td>
<td>INPLN</td>
<td>INPLN</td>
<td>INPLN</td>
</tr>
<tr>
<td>LSTMGR</td>
<td>.1</td>
<td>LSTMGR</td>
<td>LSTMGR</td>
<td>LSTMGR</td>
<td>LSTMGR</td>
<td>LSTMGR</td>
</tr>
<tr>
<td>LSTTBL</td>
<td>.0</td>
<td>LSTTBL</td>
<td>LSTTBL</td>
<td>LSTTBL</td>
<td>LSTTBL</td>
<td>LSTTBL</td>
</tr>
<tr>
<td>SPOOLR</td>
<td>.5</td>
<td>SPOOLR</td>
<td>SPOOLR</td>
<td>SPOOLR</td>
<td>SPOOLR</td>
<td>SPOOLR</td>
</tr>
<tr>
<td>COMMGR</td>
<td>.1</td>
<td>COMMGR</td>
<td>COMMGR</td>
<td>COMMGR</td>
<td>COMMGR</td>
<td>COMMGR</td>
</tr>
<tr>
<td>NETREQ</td>
<td>1.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NETREQ</td>
<td>-</td>
</tr>
<tr>
<td>MSGFMT</td>
<td>.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MSGFMT</td>
<td>MSGFMT</td>
</tr>
<tr>
<td>RTCMGR</td>
<td>.1</td>
<td>RTCMGR</td>
<td>RTCMGR</td>
<td>RTCMGR</td>
<td>RTCMGR</td>
<td>RTCMGR</td>
</tr>
<tr>
<td>RTCNUL</td>
<td>.1</td>
<td>RTCNUL</td>
<td>-</td>
<td>-</td>
<td>RTCNUL</td>
<td>-</td>
</tr>
<tr>
<td>DSPCHR</td>
<td>.6</td>
<td>DSPCHR</td>
<td>DSPCHR</td>
<td>DSPCHR</td>
<td>DSPCHR</td>
<td>DSPCHR</td>
</tr>
<tr>
<td>DSPSGL</td>
<td>.1</td>
<td>DSPSGL</td>
<td>DSPSGL</td>
<td>-</td>
<td>-</td>
<td>DSPSGL</td>
</tr>
<tr>
<td>MEMMGR</td>
<td>.3</td>
<td>MEMMGR</td>
<td>MEMMGR</td>
<td>MEMMGR</td>
<td>MEMMGR</td>
<td>MEMMGR</td>
</tr>
<tr>
<td>COMSUB</td>
<td>.3</td>
<td>COMSUB</td>
<td>COMSUB</td>
<td>COMSUB</td>
<td>COMSUB</td>
<td>COMSUB</td>
</tr>
<tr>
<td>SYSNIT</td>
<td>.1</td>
<td>SYSNIT</td>
<td>SYSNIT</td>
<td>SYSNIT</td>
<td>SYSNIT</td>
<td>SYSNIT</td>
</tr>
</tbody>
</table>

**System Generation**

Configuration Guide to TurboDOS

Copyright (C) 1981 by Software 2000 Inc.
**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.

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

<table>
<thead>
<tr>
<th>O/S Loader</th>
<th>Single User</th>
<th>Single User w/Spooling</th>
<th>Network Master</th>
<th>Network Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDLOADR</td>
<td>STDSINGL</td>
<td>STDSPOOL</td>
<td>STDMASTR</td>
<td>STDSLAVE</td>
</tr>
<tr>
<td>Functional Modules</td>
<td>7K</td>
<td>10K</td>
<td>11K</td>
<td>13K</td>
</tr>
<tr>
<td>Device Drivers</td>
<td>2K</td>
<td>2K</td>
<td>2K</td>
<td>3K</td>
</tr>
<tr>
<td>Disk Buffer Space</td>
<td>4K</td>
<td>4K</td>
<td>4K</td>
<td>16K</td>
</tr>
<tr>
<td>Dynamic Storage</td>
<td>±1K</td>
<td>±1K</td>
<td>±1K</td>
<td>±1K</td>
</tr>
<tr>
<td>Total Memory Req'd</td>
<td>14K</td>
<td>17K</td>
<td>18K</td>
<td>33K</td>
</tr>
<tr>
<td>TPA (in 64K system)</td>
<td>n/a</td>
<td>47K</td>
<td>46K</td>
<td>31K</td>
</tr>
</tbody>
</table>

**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 OSSLA VE.SYS. The GEN command can also be used to generate the code for a start-up PROM.

At system start-up, the start-up PROM 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 OSSLA VE.SYS into the slave computers on the network.
**GEN Command**

The GEN command is used for TurboDOS system generation. 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:

```
reelfile1, reelfile2, ..., reelfileN
```

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 magnetically serializes the resulting 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.
Examples:

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 :LIBDEF
* STDSINGLE, CON192, LSTCTS, SP442
* SER480, BRT442, RTC442
* DSK401, DSKFMT3, HDWNIT
Pass 1.
LCLUSR    LCLTBL    CMDINT    AUTLOG    SGLUSR    PRVUSR
OSNTRY    FILMGR    FILSUP    FASLOD    BUFMGR    DSKMGR
DSKTBL    NONFIL    CONMGR    CONTBL    DOMGR    INPLN
LSTMGR    LSTTBL    COMMGR    RTCMGR    DSPSGL    MEMMGR
COMSUB    SYSNIT    CON192    LSTCTS    SP442    SER480
BRT442    RTC442    DSK401    DSKFMT    HDWNIT
Pass 2.
LCLUSR    LCLTBL    CMDINT    AUTLOG    SGLUSR    PRVUSR
OSNTRY    FILMGR    FILSUP    FASLOD    BUFMGR    DSKMGR
DSKTBL    NONFIL    CONMGR    CONTBL    DOMGR    INPLN
LSTMGR    LSTTBL    COMMGR    RTCMGR    DSPSGL    MEMMGR
COMSUB    SYSNIT    CON192    LSTCTS    SP442    SER480
BRT442    RTC442    DSK401    DSKFMT    HDWNIT
```

Processing parameter files:
AUTLOG = 80
NMBUFS = 8
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).

Examples:

```
CLBLLEN = 9D
NMBUFS = 4
BUFSIZE = 3
CBFCHR = "F"
CLCHR = "\n"
ATNCHR = "S"
RESCHR = "Q"
ABTCHR = "C"
DSKAST = 00,01,02,03,10,11,12,13,20,21,22,23,30,31,32,33
```
TurboDOS Patch Points

Parameters in TurboDOS which may be useful to patch include the following, shown with their standard values:

**In AUTLOG Module**

- **COLDFN** = 0,"COLDSTRTAUT"
  Cold-start autoload file (12 bytes)
- **WARMFN** = 0,"WARMSTRTAUT"
  Warm-start autoload file (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
- **CLSCHR** = "\n"
  Command line separator character

**In CNTBL Module**

- **ATNCHR** = "S"
  Attention character
- **ATNBEL** = "G"
  Attention-received response
- **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
  Console assignment table
- **CONTBL** = CONDRA
  Console driver table

**In DSKTBL Module**

- **DSKAST** = 00,01,02,03,10,11,12,13,20,21,22,23,30,31,32,33
  Disk assignment table (16 bytes)
- **DSKTBL** = DSKDRA, DSKDRB, DSKDRC, DSKDRD
  Disk driver table (4 words)

**In FLUSHR Module**

- **BFLDLY** = (012C)
  Buffer flush delay in ticks (no flush if zero)
Configuration Guide to TurboDOS
Copyright (C) 1981 by Software 2000 Inc.
System Generation

In LCLTBL Module:
SPLMOD = 1
Default spool mode
QUEPTR = 1
Default spool queue assignment
SPLDRV = OFF
Default spool drive 00...0F (system drive if OFF)

In LCLIUSR Module:
MEMRES = (0100)
Reserved memory between O/S and TPA
SOMSG = "TurboDOS 1.xx, Copyright (C) 1981, Software 2000, Inc. $"
Sign-on message (56 bytes, must end with "$")

In LSTTBL Module:
LSTAST = 00,10,20,30
List assignment table (4 bytes)
LSTTBL = LSTDRA,LSTDDB,LSTDRE,LSTDRO
List driver table (4 words)
NMBQUE = 1
Number of de-spool queues
DSPPAT = 1,...,1
De-spool printer assignment table (16 bytes)
NMBPTR = 1
Number of printers
LSTREM = OFF
Default print site (0=local, OFF=remote)
EOPCHR = 0
End-of-print character (if nonzero)

In MEMMGR Module:
MEMBLL = (1103)
Memory base lower limit (standard assures 4K TPA)

In NONFIL Module:
LOGUSR = 1F
User number for log-off (standard is 31)

In OSLOAD Module:
LOADFN = 0,"OSMASTERSYS"
Default drive and filename for OSLOAD (12 bytes)
MEMTOP = (0FFFF)
Top limit of OSLOAD RAM test (don't test if zero)

In SLVTBL Module:
NMBSLV = 2
Number of network slaves
SLVTBL = " "
OSSLAVEEX.SYS suffix letters (16 bytes)
Explanatory Notes

The patch "AUTUSR = 80" should generally be included in single-user configurations to cause an automatic privileged log-on to user number zero.

The disk assignment table DSKAST contains an array of byte entries corresponding to drives A...P. The high-order nibble of each entry specifies which disk driver (in DSKTBL) to use, while the low order nibble is a drive number passed to the selected driver. In network slaves, the high-order nibble should be set to 15 to indicate a remote drive.

The list assignment table LSTAST contains an array of byte entries corresponding to printers A...P. The high-order nibble of each entry specifies which printer driver (in LSTTBL) to use, while the low order nibble is a printer number passed to the selected driver in the B-register. The console assignment table CONAST works the same way.

If EOPCHR is patched to any non-null ASCII character, then the presence of that character in the print output stream will automatically signal an end-of-print-job condition.

The slave suffix table SLVTBL contains an array of byte entries corresponding to slaves A...P. Each slave operating system is down-loaded from the file "OSSLAVE\Ex_SYS", where "x" is the proper SLVTBL entry. SLVTBL normally contains all spaces, so that all slaves are down-loaded from "OSSLAVE_SYS".
**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 CP/M-compatible format (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.

7. 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.

8. 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 "unm" 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.

Examples:

```
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) 1981, Software 2000 Inc."

3. Using the SERIAL command, copy and serialize the following files from your OEM redistribution master to the new diskette: the appropriate STDxxxxx files, all necessary driver modules, and plus .COM files for AUTOLOAD, BACKUP, BUFFERS, CHANGE, COPY, DATE, DELETE, DIR, DO, DRIVE, DUMP, ERASEDIR, FIFO, FIXMAP, FORMAT, GEN, LABEL, LOGOFF, LOGON, MASTER, PRINT, PRINTER, QUEUE, RECEIVE, RENAME, RESET, SEND, SET, SHOW, TYPE, USER, and VERIFY. 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 and copies of the User's Guide and Configuration Guide.
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. 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 competent 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 denote an external name that is defined in another module. The label suffix "+" is used to denote 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. 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 at 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

Note that ALLOC# clears each newly-allocated segment to zeroes. Note also that ALLOC# prefixes each dynamic memory segment with a word containing the segment length (including the prefix word itself), so that DEALOC# can tell how much memory is to be de-allocated.
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:**

```assembly
.WORD 1 ; mutual-exclusion interlock semaphore
.WORD . ; semaphore count (initialized to 1)
.WORD -. ; semaphore list head forward pointer
.WORD -.2 ; semaphore list head backward pointer

; EVENT:
.WORD 0 ; event semaphore
.WORD . ; 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 
LXI H,EVENT ; ensure that interrupts are enabled
CALL WAIT# ; get event semaphore
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 auxiliary 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 occurred
POP H ; restore all registers
POP D
POP B
POP PSW
LSPD INTSP# ; restore user's stack pointer
JMP ISRXIT# ; exit through dispatcher
```
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:
    .WORD 1 ; mutual-exclusion interlock semaphore
    .WORD 2 ; semaphore count (initialized to 1)
    .WORD -2 ; semaphore list head forward 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.
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 of 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:
  .WORD 0 ;event semaphore
  .WORD - ;semaphore count
  .WORD -2 ;semaphore list forward 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
```
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 ISRXT# 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 auxiliary 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 auxiliary 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 occurred
         POP      H         ;restore all registers
         POP      D
         POP      B
         POP      PSW
         LSPD     INTSP#    ;restore user's stack pointer
         JMP      ISRXT#    ;exit through dispatcher
```

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).
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.
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 dispatcher 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:
.WORD 0     ;event semaphore
.WORD .     ;semaphore count
.WORD .-2   ;semaphore list forward 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 occurred
```

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.
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:
  .WORD   ;list head (initialized to empty)
  .WORD    ;forward pointer
  .WORD    ;backward pointer

LSTNOD:
  .WORD 0   ;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
```
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. Using DEALOC#, the memory space occupied by such initialization code can be made available to satisfy subsequent dynamic memory requirements. To do this, the code segment must be prefixed with a word containing the segment length:

```
.WORD LENGTH+2 ;length to be de-allocated
; HDWINIT: XRA A ;start of initialization code
    ...
    LXI H,HDWINIT ;get beginning of segment
    JMP DEALOC# ;de-allocate segment
; LENGTH = ~HDWINIT ;length of segment
```
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:
.WORD 1 ;mutual-exclusion interlock semaphore
.WORD . ;semaphore count (initialized to 1)
.WORD -2 ;semaphore list head forward pointer
.EVENT:
.WORD 0 ;event semaphore
.WORD . ;semaphore count
.WORD -2 ;semaphore list head 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
```
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 routines:

```assembly
?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.
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, 3 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=3, the driver should prepare to display a TurboDOS error message; if E=9, the driver should revert to normal display. Error message displays issued by TurboDOS are always preceded by an E=3 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=3 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 Drivers

(To be supplied in a future revision.)
Disk Drivers

Each disk driver routine should begin with the public entry name DSKDR@x, 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
RTCSR:
  SSDP INTSP# ;save user's stack pointer
  LXI SP,INTSTK# ;set up auxiliary stack
  PUSH PSW ;save all registers
  PUSH B
  PUSH D
  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
  POP H ;restore all registers
  POP D
  POP B
  POP PSW
  LSPD INTSP# ;restore user's stack pointer
  JMP ISRXIT# ;exit through dispatcher

..X:
```

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.

If 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.)
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, if 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 A — IMPLEMENTATION ON IMS EQUIPMENT**

**Drivers furnished for IMS Equipment**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRT442O</td>
<td>Baud-rate tables for IMS 442/480 with baud-rate oscillator.</td>
</tr>
<tr>
<td>BRT442S</td>
<td>Baud-rate tables for IMS 442 without baud-rate oscillator.</td>
</tr>
<tr>
<td>BRT740</td>
<td>Baud-rate tables for IMS 740 slave board.</td>
</tr>
<tr>
<td>CON192</td>
<td>Console driver for ASCII CRT at 19,200 baud.</td>
</tr>
<tr>
<td>CON96</td>
<td>Console driver for ASCII CRT at 9,600 baud.</td>
</tr>
<tr>
<td>DSK401</td>
<td>Disk driver for IMS 401 8-inch floppy disk controller.</td>
</tr>
<tr>
<td>DSK431</td>
<td>Disk driver for IMS 431 5-inch floppy disk controller.</td>
</tr>
<tr>
<td>DSK490</td>
<td>Disk driver for IMS 490 hard disk controller.</td>
</tr>
<tr>
<td>DSKIMS58</td>
<td>Disk driver for both IMS 401 and 431 floppy disk controllers.</td>
</tr>
<tr>
<td>DSKM10</td>
<td>Disk driver for Morrow 10 Mb Winchester.</td>
</tr>
<tr>
<td>DSKM20</td>
<td>Disk driver for Morrow 20 Mb Winchester.</td>
</tr>
<tr>
<td>DSKM26</td>
<td>Disk driver for Morrow 26 Mb Winchester.</td>
</tr>
<tr>
<td>DSKFMT5</td>
<td>Disk specification tables for 5-inch floppy disks.</td>
</tr>
<tr>
<td>DSKFMT8</td>
<td>Disk specification tables for 8-inch floppy disks.</td>
</tr>
<tr>
<td>DSKFMTA</td>
<td>Disk specification tables for both 5-inch and 8-inch floppy disks.</td>
</tr>
<tr>
<td>HDWNIT</td>
<td>Hardware initialization.</td>
</tr>
<tr>
<td>LST300</td>
<td>Printer driver for no handshaking, 300 baud (e.g., Teletype 43).</td>
</tr>
<tr>
<td>LSTCTS</td>
<td>Printer driver for CTS handshaking, 9600 baud (e.g., TI-810).</td>
</tr>
<tr>
<td>LSTETX</td>
<td>Printer driver for ETX/ACK handshaking, 1200 baud (e.g., NEC 5510).</td>
</tr>
<tr>
<td>LSTXON</td>
<td>Printer driver for XON/XOFF handshaking, 1200 baud (e.g., Diablo 630).</td>
</tr>
<tr>
<td>LSTIMS</td>
<td>Printer driver for Centronics parallel.</td>
</tr>
<tr>
<td>LSTNEC</td>
<td>Printer driver for parallel NEC 5500D.</td>
</tr>
<tr>
<td>MPENIT</td>
<td>Memory parity initialization for IMS 461 64K RAM board.</td>
</tr>
<tr>
<td>N740M</td>
<td>Network driver for master, using IMS 740 slave boards.</td>
</tr>
<tr>
<td>N740S</td>
<td>Network driver for slaves, using IMS 740 slave boards.</td>
</tr>
<tr>
<td>NET80M</td>
<td>Network driver for master, using MuSYS NET/80 slave boards.</td>
</tr>
<tr>
<td>NET80S</td>
<td>Network driver for slaves, using MuSYS NET/80 slave boards.</td>
</tr>
<tr>
<td>RTC442</td>
<td>Real-time clock driver for IMS 442 ROM-I/O board.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>SER480</td>
<td>Serial subroutines for IMS 480 4-port serial board.</td>
</tr>
<tr>
<td>SP442</td>
<td>Serial and parallel subroutines for IMS 442 ROM-I/O board.</td>
</tr>
<tr>
<td>SP740</td>
<td>Serial and parallel subroutines for IMS 740 slave board.</td>
</tr>
<tr>
<td>SPN80</td>
<td>Serial and parallel subroutines for MuSYS NET/80.</td>
</tr>
</tbody>
</table>
**Symbolic Patch Points in IMS Driver Modules**

Parameters in the IMS hardware-dependent driver modules which may be useful to patch include the following, shown with their standard values:

**In CON96 Modules**
- **CONBR** = 8E
- **FFCHR** = 0C
  - Baud rate code (9,600 baud w/attention detect)
  - Clear-the-screen character at cold-start

**In CON192 Modules**
- **CONBR** = 8F
- **FFCHR** = 0C
  - Baud rate code (19,200 baud w/attention detect)
  - Clear-the-screen character at cold-start

**In LST300 Modules**
- **LST3BR** = 25
- **LST3FF** = 0C
  - Baud rate code (300 baud, output-only)
  - Top-of-form character at end-of-print

**In LSTCTCS Modules**
- **CTSBR** = 6E
- **CTSF** = 0C
  - Baud rate code (9,600 baud, CTS handshaking)
  - Top-of-form character at end-of-print

**In LSTETX Modules**
- **ETXBR** = 07
- **ETXFF** = 0C
  - Top-of-form character at end-of-print
- **ETXLEN** = 8C
  - Length of output between ETXs
- **ETXSEQ** = 03
  - Length of maximal escape sequence

**In LSTIMS Modules**
- **IMSFF** = 0C
  - Top-of-form character at end-of-print

**In LSTXON Modules**
- **XONBR** = 07
- **XONFF** = 0C
  - Baud rate code (1,200 baud, input/output)
  - Top-of-form character at end-of-print

**In NZ40M Modules**
- **SLVPAT** = 40,44,48,4C,50,54,58,5C,60,64,68,6C,70,74,78,7C
  - Slave board port assignment table

**In NET20M Modules**
- **N80PAT** = 20,22,24,26,28,2A,2C,2E,30,32,34,36,38,3A,3C,3E
  - Slave board port assignment table
### Implementation on IMS Equipment

**In SP442 Modules**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERBSZ</td>
<td>40</td>
<td>Type-ahead buffer size</td>
</tr>
</tbody>
</table>

**In SP740 Modules**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERBSZ</td>
<td>40</td>
<td>Type-ahead buffer size</td>
</tr>
</tbody>
</table>
IMS Logic Board Set-Up for TurboDOS

401 8" Floppy Disk Controller

Shunt  
JA  Address to port 80-8F hex (standard etch)  
JB  Install horizontal shunt at "5" (interrupts)  
JC  Install two horizontal shunts at top and bottom (precomp)  
JD  Install horizontal shunt (delay complete)

431 5" Floppy Disk Controller

Shunt  
JA  Address to port C0-CF hex (cut trace, install jumper)  
JB  Install horizontal shunt at "5" (interrupts)  
JC  Install vertical shunt (two-sided) if MPI 52 drives installed

Remove IC at "5-B" (74LS74), lift pin 13 clear of socket, connect to pin 12 on the IC itself, and reinstall in socket (ready logic modification).

442 I/O Board

Shunt  
JA  Install horizontal shunt on topmost pair  
JB  Install vertical shunt  
JC  For parallel NEC printer, install horizontal shunts on all 16 pairs  
JD  Standard etch  
JE  Standard etch  
JF  Standard etch  
JG  Install only 1 horizontal shunt on bottom pair (ROM enable at FC00 hex)  
JH  Cut trace between top pair, jumper bottom pair (timer enable)  
JJ  Install horizontal shunt at "VI1"  
JK  Install horizontal shunt at "VB3"  
JL  Install horizontal shunt at "VI3"  
JM  No shunt  
JN  No shunt  
JP  Install 3 horizontal shunts at "A7", "A6", and "A5" (not "A4")

Install oscillator at "13-D" and install 74LS161 IC at "13-C". Install TurboDOS boot PROM at "65-B". For parallel NEC printer, install 220/330 resistor pack at IC "12-A".

450 730 CPU Board

Shunt  
JA  No shunt (wait states)  
JB  Install 2 horizontal shunts on bottom (Jump to FC00 hex)  
JC  Standard etch (4 MHz)  
JD  Standard etch

For accurate real-time clock performance, add 10 pf capacitor (mica or NPO) in series with one leg of 16 Mhz crystal.
**461 64K Dynamic RAM Board**

- **Shunt JA** Install vertical shunt on top pair (high-speed RAM)
- **Shunt JB** Install horizontal shunt (I/O port enabled)
- **Shunt JC** Install 8 horizontal shunts (port 00)
- **Shunt JD** Install shunt (phantom)
- **Shunt JE** No shunt (unmapped)
- **Shunt JF** No shunt (Z80 timing)
- **Shunt JG** No shunt (vectored interrupt)
- **Shunt JH** Install horizontal shunt on left pair (no front panel)

**480 Four-Port Serial Board**

- **Shunt JA** No shunts (port address E0-FF hex)
- **Shunt JB** Install horizontal shunt on bottom pair (use oscillator)
- **Shunt JC** Install horizontal shunt at "V13" (vectored interrupt)
- **Shunt JD** Install horizontal shunt at "V13" (vectored interrupt)
- **Shunt JE** Install horizontal shunt at "V13" (vectored interrupt)
- **Shunt JF** Install horizontal shunt at "V13" (vectored interrupt)

**490 Cartridge Module Drive Controller**

- **Shunt JA** Standard etch (DMA channel 2)
- **Shunt JC** Standard etch (address port 90-97 hex)
- **Shunt JD** Install vertical shunt at "V14" (vectored interrupt)

**750 I/O Processor Board (Slave up)**

- **Shunt JA** Address boards to hex 40, 44, 48, 4C, etc.
- **Shunt JB** No shunt (disable vectored interrupt)
APPENDIX B — IMPLEMENTATION ON TRS-80 MODEL II

Drivers furnished for TRS-80 Model II

CONTRS  Console driver for TRS-80 keyboard/display.

DSKTR5  Disk driver for TRS-80 8-inch floppy disk controller.

DSKFMT3 Disk specification tables for 8-inch floppy disks.

HDWNIT  Hardware initialization.

LSTTRS  Printer driver for TRS-80 Centronics parallel interface.

LST300  Printer driver for no handshaking, 300 baud (e.g., Teletype 43).

LSTCTS  Printer driver for CTS handshaking, 9600 baud (e.g., TI-810).

LSTETX  Printer driver for ETX/ACK handshaking, 1200 baud (e.g., NEC 5510).

LSTXON  Printer driver for XON/XOFF handshaking, 1200 baud (e.g., Diablo 630).

RTCTRS  Real-time clock driver for TRS-80 CTC.

SPTRS  Serial and parallel subroutines for TRS-80.
Symbolic Patch Points in TRS-80 Model II Driver Modules

Parameters in the TRS-80 Model II hardware-dependent driver modules which may be useful to patch include the following, shown with their standard values:

**In LSTTRS Module:**
- **TRSF** = 0C  
  Top-of-form character at end-of-print

**In LST300 Module:**
- **LSTSBR** = 25  
  Baud rate code (300 baud, output-only)
- **LSTSFF** = 0C  
  Top-of-form character at end-of-print

**In LSTCTS Module:**
- **CTSBR** = 6E  
  Baud rate code (9,600 baud, CTS handshaking)
- **CTSFF** = 0C  
  Top-of-form character at end-of-print

**In LSTETX Module:**
- **ETXBR** = 07  
  Baud rate code (1,200 baud, input/output)
- **ETXFF** = 0C  
  Top-of-form character at end-of-print
- **ETXLEN** = 8C  
  Length of output between ETXs
- **ETXSEQ** = 03  
  Length of maximal escape sequence

**In LSTXON Module:**
- **XONBR** = 07  
  Baud rate code (1,200 baud, input/output)
- **XONFF** = 0C  
  Top-of-form character at end-of-print

**In SPTRS Module:**
- **SERBSZ** = 40  
  Type-ahead buffer size
APPENDIX C — SAMPLE DRIVER LISTINGS
; IDENT EQUATE
;
; ASCII EQUIVALENCES
;
0000  ABUL  ==  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  ABEI  ==  07H ; BEL
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  AANAK  ==  15H ; NAK
0016  ASYN  ==  16H ; SYN
0017  AETB  ==  17H ; ETEB
0018  ACRE  ==  18H ; CAN
0019  AEHM  ==  19H ; EM
001A  ASUB  ==  1AH ; SUB
001B  AESC  ==  1BH ; ESC
001C  AFSP  ==  1CH ; FS
001D  AGS  ==  1DH ; GS
001E  ABS  ==  1EH ; RS
001F  AUS  ==  1FH ; US
0020  ASP  ==  20H ; SPACE
007F  ARUB  ==  7FH ; RUBOUT (DEL)
EQUATE - TURBODOS OPERATING SYSTEM SYMBOLIC EQUIVALENCES
Copyright (C) 1981 by Software 2000, Inc.

; WARM START ENTRYPONT
WBOOT  ==  0000H

; I/O CONFIGURATION ENTRYPOINT
IOBITE  ==  0003H

; CURRENT DEFAULT DRIVE
CURDRV  ==  0004H

; OPERATING SYSTEM ENTRYPONT
OPESTS  ==  0005H

; DEFAULT FILE CONTROL BLOCK
TFDB  ==  0005CH

; DEFAULT DISK BUFFER ADDRESS
TBUF  ==  0080H

; TRANSIENT PROGRAM AREA BASE
TPA  ==  0100H

; WORKING STORAGE RELATIVE TO 0
.LOC  0

; PD REQUEST DESCRIPTOR PACKET
PDRDP:

; PD REQUEST FUNCTION NUMBER
PDRFCN:  .BLKB  1

; PD REQUEST DRIVE NUMBER
PDRDRV:  .BLKB  1

; PD REQUEST TRACK NUMBER
PDRTRK:  .BLKW  1

; PD REQUEST SECTOR NUMBER
PDRSEC:  .BLKW  1

; PD REQUEST SECTOR COUNT
PDRSC:  .BLKW  1

; PD REQUEST DMA ADDRESS
PDRDMA:  .BLKW  1

; PD REQUEST DRIVE SPEC TABLE A
PDRT:

; PD REQUEST DESCRIPTOR PACKET LENGTH
PDRLEN  ==  -PDRDP

; DISK TYPE INFORMATION
DSKNFO:

; BLOCK SIZE
BLKSIZ:  .BLKB  1

; NUMBER OF BLOCKS
NMBLKS:  .BLKW  1

; NUMBER OF DIRECTORY BLOCKS
NMBDIR:  .BLKB  1

; PHYSICAL SECTOR SIZE (2^N*128)
SECSIZ:  .BLKB  1

; PHYSICAL SECTORS PER TRACK
SECTR:  .BLKW  1

; PHYSICAL TRACKS PER DISK
TRKDSK:  .BLKW  1

; NUMBER OF RESERVED TRACKS
RESTR:  .BLKW  1

; DISK INFO LENGTH
DNFOL  ==  -DSKNFO

; END
; COPYRIGHT (C) 1981, SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIKES
; MICHAEL D. BUSCH
; VERSION: 09/08/81
IDENT HDWNIT ;MODULE ID
;INSERT DREQUATE ;DRIVER SYMBOLIC EQUIVALENCES
;LOC ;LOCATE IN PROGRAM AREA
;WORD NITLEN+2 ;INITIALIZATION CODE LENGTH
0000' 0020
0000' 0020
0002' CD 0000:04 HDWNIT::CALL MPENIT# ;INITIALIZE MEMORY PARITY
0005' CD 0000:05 CALL SPINIT# ;INITIALIZE SERIAL/PARALLEL I/O
0008' CD 0000:06 CALL RTCNIT# ;INITIALIZE REAL TIME CLOCK
000B' CD 0000:07 CALL DSKINA# ;INITIALIZE DISK DEVICE A
000E' CD 0000:08 CALL DSKINB# ;INITIALIZE DISK DEVICE B
0011' CD 0000:09 CALL DSKINC# ;INITIALIZE DISK DEVICE C
0014' CD 0000:0A CALL DSKIND# ;INITIALIZE DISK DEVICE D
0017' CD 0000:0B CALL NETNIT# ;INITIALIZE NETWORK DRIVER
001A' 21 0002' LXI H, HDWNIT ;GET INITIALIZATION CODE ADDRESS
001D' C3 0000:0C JMP DEALOC# ;DE-ALLOCATE INITIALIZATION CODE
001E NITLEN = .-HDWNIT ;INITIALIZATION CODE LENGTH
;END
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

AUTHORS: RONALD E. RAIKES
          MICHAEL D. BUSCH

VERSION: 09/08/81

IDENT CON96

MODULE ID

INSERT DREQUATE

DRIVER SYMBOLIC EQUIVALENCES

;LOC .DATA; LOCATE IN DATA AREA

;LOC .PROG; LOCATE IN PROGRAM AREA

CON96 - TURBODOS OPERATING SYSTEM NULL CONSOLE DRIVER

COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

CONSO: CALL DMS# ;POSITION TO NEXT LINE
CONSI: CALL [ACR] [ALF]

;END
PSA Macro Assembler [C12011-0102 ]

STXON - TURBODOS OPERATING SYSTEM XON/XOFF PRINTER DRIVER

COPYRIGHT (C) 1981 BY SOFTWARE 2000, INC.

; COPYRIGHT (C) 1981 BY SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIAES
;          MICHAEL D. BUSCH
; VERSION: 09/08/81
; IDENT LSTXON ;MODULE ID
; INSERT DREQUATE ;DRIVER SYMBOLIC EQUIVALENCES

0000' .LOC .DATA# ;LOCATE IN DATA AREA

0000' 07 XONBR:: .BYTE 7 ;BAUD RATE CODE (1200 BAUD)
0001' 0C XONOFF:: .BYTE AFF ;FORM FEED CHARACTER
0002' 000000000000 INITC: .BYTE [16]0 ;INITIALIZATION COMPLETE FLAGS
0003' .LOC .PROG# ;LOCATE IN PROGRAM AREA

0000' 21 0002' LSTDRS::LXI H,INITC ;GET INITIALIZATION COMPLETE FLAGS
0003' D5 PUSH D ;SAVE FUNCTION NUMBER
0004' 58 MOV E,B ;CHANNEL NUMBER TO DE-REG
0005' 1600 MVI D,0 ;DOUBLE LENGTH
0007' 19 DAD D ;INDEX INTO FLAGS TABLE
0008' D1 POP D ;RESTORE FUNCTION NUMBER
0009' 7E MOV A,M ;GET INITIALIZATION COMPLETE FLAG
000A' B7 ORA A ;INITIALIZATION COMPLETE FLAG SET?
000B' CC 0018' CZ ..INIT ;IF NOT, INITIALIZE LIST CHANNEL
000E' 7B MOV A,E ;GET FUNCTION NUMBER
000F' FE02 CPI 2 ;FUNCTION NUMBER=2?
0011' 281A JRZ LSTOUT ;IF SO, CONTINUE
0013' FE07 CPI 7 ;FUNCTION NUMBER=7?
0015' 2810 JRZ LSTWSR ;IF SO, CONTINUE
0017' C9 RET ;ELSE, DONE
0018' 35 ..INIT: DCR M ;SET INITIALIZATION COMPLETE FLAG
0019' D5 PUSH D ;SAVE FUNCTION NUMBER
001A' C5 PUSH B ;SAVE CHANNEL NUMBER/CHARACTER
001B' 3A 0000' LDA XONBR ;GET BAUD RATE CODE
001E' 4F MOV C,A ;BAUD RATE CODE TO C-REG
001F' 1E03 MVI E,3 ;SET FUNCTION NUMBER=3
0021' CD 0000:04 CALL SERIAL# ;SET CHANNEL BAUD 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
PSA Macro Assembler [C12011-0102]

SP442 - TURBODOS OPERATING SYSTEM SERIAL/PARALLEL I/O DRIVER (IMS 442)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

; COPYRIGHT (C) 1981, SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIKES
          MICHAEL D. BUSCH
; VERSION: 09/08/81
; IDENT SP442 ;MODULE ID
; INSERT DREQUATE ;DRIVER SYMBOLIC EQUIVALENCES

0010:  IOBASE  = 10H ;SERIAL/PARALLEL I/O PORT BASE

0010:  SOCTRL  = IOBASE+00H ;SERIAL 0 CONTROL/STATUS REGISTER
0011:  SODATA  = 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:  POCTRL = IOBASE+0CH ;PARALLEL 0 PORT CONTROL REGISTER
001D:  P1DATA = IOBASE+0DH ;PARALLEL 1 DATA REGISTER
001E:  P2DATA = IOBASE+0EH ;PARALLEL 2 DATA REGISTER
001F:  PPCTRL = 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 RX 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 RX TO SEND (NOT) BIT

0036:  TOCMD = 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

FC00: BOOTPR = OFC00H ;BOOTSTRAP LOADER EPROM BASE

00000 ;LOC .DATA $ ;LOCATE IN DATA AREA
STXON - TURBODOS OPERATING SYSTEM XON/XOFF PRINTER DRIVER

COPYRIGHT (C) 1981 BY SOFTWARE 2000, INC.

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
SP442 – TURBODOS OPERATING SYSTEM SERIAL/PARALLEL I/O DRIVER (IMS 442)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0000' 0040  SERBSZ: .WORD 64 ;SERIAL BUFFER SIZE
0002'  49   INTMSK: .BYTE 1<ROMDIS I<SOXIE1 I<S1XIE ; INTERRUPT MASK
0003'   00   SOBR: .BYTE 0 ;SERIAL 0 BAUD RATE
0004'   00   S1BR: .BYTE 0 ;SERIAL 1 BAUD RATE
0005'   0000  SERBUF: .WORD 0 ;SERIAL BUFFER ADDRESS
0007'   0000  SERPTR: .WORD 0 ;SERIAL BUFFER POINTER
0009'   00   S00CHR: .BYTE 0 ;SERIAL 0 OUTPUT CHARACTER
000A'   00   S10CHR: .BYTE 0 ;SERIAL 1 OUTPUT CHARACTER

000B'   0000  SOISPH: .WORD 0 ;SERIAL 0 INPUT SEMAPHORE
000D'   0000  S0IH: .WORD 0 ;SEMAPHORE COUNT
000F'   0000  S1IH: .WORD 0 ;SEMAPHORE P/D HEAD

0011'   0000  S1ISPH: .WORD 0 ;SERIAL 1 INPUT SEMAPHORE
0013'   0013  S1IH: .WORD 0 ;SEMAPHORE P/D HEAD
0015'   0013  S1IH: .WORD 0 ;SEMAPHORE P/D HEAD

0017'   0000  S0OSPH: .WORD 0 ;SERIAL 0 OUTPUT SEMAPHORE
0019'   0019  S0OH: .WORD 0 ;SEMAPHORE COUNT
001B'   0019  S0OH: .WORD 0 ;SEMAPHORE P/D HEAD

001D'   0000  S1OSPH: .WORD 0 ;SERIAL 1 OUTPUT SEMAPHORE
001F'   001F  S1OH: .WORD 0 ;SEMAPHORE COUNT
0021'   001F  S1OH: .WORD 0 ;SEMAPHORE P/D HEAD

0000'  0038  .WORD NITLEN+2 ;INITIALIZATION CODE LENGTH

0002'  3E89  SPINIT: MVI A,PPMODE ; INITIALIZE 8255
0004'  D31F  OUT PPCTL
0006'  3EFF  MVI A,0FFH ; CLEAR PARALLEL PORTS
0008'  D31C  OUT PODATA
000A'  D31D  OUT P1DATA
000C'  3E19  MVI A,SPMODE ; INITIALIZE UARS
000E'  D310  OUT SOCTRL
0010'  D312  OUT S1CTRL
0012'  3EC3  MVI A,JMP ; SET UP SERIAL 0 INTERRUPT VECTOR
0014'  320018  STA 3*8
0017'  210107'  LXI H,SERISR
001A'  220019  SHLD (3*8)+1
001D'  3A0002'  LDA INTMSK ; GET INTERRUPT MASK
0020'  D318  OUT SINTE ; ENABLE INTERRUPT MASKS
0022'  2A0000'  LHLD SERBSZ ; GET SERIAL BUFFER SIZE
0025'   29   DAD H ; X2
0026'  CD0000:04  CALL ALLOC# ; ALLOCATE PACKET FOR SERIAL BUFFERS
0029'  220005'  SHLD SERBUF ; SAVE SERIAL BUFFER ADDRESS
002C'  220007'  SHLD SERPTR ; SET SERIAL BUFFER POINTER
002F'  CD0000:05  CALL NIT480# ; INITIALIZE IMS 480 SERIAL PORTS
0032'  210002'  LXI H,SPINIT ; GET INITIALIZATION CODE ADDRESS
0035' C3 0000:06 ; JMP DEALOG$ ;DE-ALLOCATE INITIALIZATION CODE
0036' 7B ;INITLEN = _SPINIT ;INITIALIZATION CODE LENGTH
0038' 7B ; SERIAL: MOV A,E ;GET FUNCTION NUMBER
0039' B7 ; ORA A ;FUNCTION NUMBER=0?
003A' 2817 ; JRZ SERST ;IF SO, CONTINUE
003C' 3D ; DCR A ;FUNCTION NUMBER=1?
003D' 282A ; JRZ SERIN ;IF SO, CONTINUE
003F' 3D ; DCR A ;FUNCTION NUMBER=2?
0040' 284A ; JRZ SEROUT ;IF SO, CONTINUE
0042' 3D ; DCR A ;FUNCTION NUMBER=3?
0043' CA 01A2' ; JZ SERSBR ;IF SO, CONTINUE
0046' 3D ; DCR A ;FUNCTION NUMBER=4?
0047' CA 01D5' ; JZ SERSBR ;IF SO, CONTINUE
004A' 3D ; DCR A ;FUNCTION NUMBER=5?
004B' CA 01E4' ; JZ SERSMC ;IF SO, CONTINUE
004E' 3D ; DCR A ;FUNCTION NUMBER=6?
004F' CA 0208' ; JZ SERSMC ;IF SO, CONTINUE
0052' C9 ; ELSE, DONE
0053' 2A 0005" ; SERST: LHLD SERBUF ;GET SERIAL BUFFER ADDRESS
0056' ED5B 0007" ; LDDED SERPTR ;GET SERIAL BUFFER POINTER
005A' E5 ; .STL: PUSH H ;SAVE SERIAL BUFFER ADDRESS
005B' AF ; XRA A ;CLEAR CARRY/PRESET RETURN CODE=0
005C' ED52 ; DSBC D ;END OF SERIAL BUFFER?
005E' E1 ; POP H ;RESTORE SERIAL BUFFER ADDRESS
005F' C8 ; RZ ;IF END OF SERIAL BUFFER, DONE
0060' 73 ; MOV A,B ;ELSE, GET CHANNEL NUMBER
0061' 96 ; SUB M ;NEXT CHARACTER=REQUESTED CHANNEL
0062' 23 ; INX H ;ADVANCE TO CHARACTER
0063' 4E ; MOV C,M ;GET CHARACTER FROM BUFFER
0064' 23 ; INX H ;ADVANCE TO NEXT CHANNEL NUMBER
0065' 2F ; CMA ;PRESST RETURN CODE=OFFH
0066' C8 ; RZ ;IF REQUESTED CHANNEL, DONE
0067' 18F1 ; JMPR ;.STL ;CONTINUE
0069' 78 ; SERIN: MOV A,B ;GET CHANNEL NUMBER
006A' FE02 ; CPI 2 ;CHANNEL NUMBER=0/1?
006C' 3805 ; JRC .SOI ;IF SO, CONTINUE
006E' CD 0000:07 ; CALL IN480# ;ELSE, GET IMS 480 IN SEMAPHORE
0071' 1809 ; JMPR .ICOM ;CONTINUE
0073' 21 000B" ;.SOI: LXI H,SOISPH ;GET SERIAL 0 IN SEMAPHORE
0076' B7 ; ORA A ;CHANNEL NUMBER=0?
0077' 2803 ; JRZ .ICOM ;IF SO, CONTINUE
0079' 21 0011" ; LXI H,SO1SPH ;ELSE, GET SERIAL 1 IN SEMAPHORE
007C' CD 0000:08 ; .ICOM: CALL WAIT$ ;WAIT FOR CONSOLE INPUT
007F' CD 0053' ; CALL SERST ;GET SERIAL CHANNEL STATUS
0082' B7 ; ORA A ;CHARACTER AVAILABLE?
0083' 28E4 ; JRZ SERIN ;IF NOT, CONTINUE
0085' 79 ; MOV A,C ;ELSE, GET INPUT CHARACTER
0086' F3 ; DI ;DISABLE INTERRUPTS
0087' CD 018C' ; CALL MOVBUF ;MOVE BUFFER TAIL DOWN
008A' FB ; EI ;ENABLE INTERRUPTS
008B: C9 RET ;DONE

008C: 78 SEROUT: MOV A,B ;GET CHANNEL NUMBER
008D: FE02 CPT 2 ;CHANNEL NUMBER=0/1?
008F: 3805 JRC .SO10 ;IF SO, CONTINUE
0091: CD 0000:09 CALL OUT480# ;ELSE, GET IMS 480 OUT SEMAPHORE
0094: 1817 JMPF .COM ;CONTINUE

0096: 21 0009# .SO10: LXI H,SOOCHR ;GET SERIAL 0 OUTPUT CHARACTER
0099: 67 ORA A ;CHANNEL NUMBER=0?
009A: 2B01 JRZ .SO1C ;IF SO, CONTINUE
009C: 23 INX H ;GET SERIAL 1 OUTPUT CHARACTER
009D: 71 .SO1C: MOV M,C ;SAVE OUTPUT CHARACTER
009E: 21 0017# LXI H,SOOSPH ;GET SERIAL 0 OUT SEMAPHORE
00A1: 11 00BF# LXI D,SOOPOL ;GET SERIAL 0 OUT POLL ROUTINE
00A4: 67 ORA A ;CHANNEL NUMBER=0?
00A5: 2B06 JRZ .COM ;IF SO, CONTINUE
00A7: 21 001D# LXI H,SOOSPH ;GET SERIAL 1 OUT SEMAPHORE
00AA: 11 00E3# LXI D,SOOPOL ;GET SERIAL 1 OUT POLL ROUTINE
00AD: E5 .COM: PUSH H ;SAVE SEMAPHORE ADDRESS
00AE: D5 PUSH D ;SAVE POLL ROUTINE ADDRESS
00AF: CD 0000:0A CALL LNKPOL# ;CREATE POLL ROUTINE
00B2: 21 00BB# LXI H,...RET ;GET RETURN ADDRESS
00B5: E3 XTHL ;SIMULATE CALL/GET POLL ROUTINE
00B6: 23 INX H ;ADVANCE PAST LINK POINTERS
00B7: 23 INX H
00B8: 23 INX H
00B9: 23 INX H
00BA: E9 PCHL ;EXECUTE POLL ROUTINE
00BB: E1 ...RET: POP H ;RESTORE SEMAPHORE ADDRESS
00BC: C3 0000:08 JMP WAIT# ;DISPATCH IF NECESSARY

00BF: SOOPOL: ;SERIAL 0 OUTPUT POLL ROUTINE
00BF: 0000 .WORD 0 ;SUCCESSOR LINK POINTER
00C1: 0000 .WORD 0 ;PREDECESSOR LINK POINTER
00C3: DB10 IN SOCTRL ;GET SERIAL 0 STATUS
00C5: CB4F BIT TBE,A ;TRANSMIT BUFFER EMPTY?
00C7: CB8 RZ ;IF NOT, DONE
00C8: 21 0003# LXI H,SOBR ;ELSE, GET SERIAL 0 BAUD RATE CODE
00CB: CB76 BIT 6,M ;CTS HANDSHAKING REQUESTED?
00CD: 2803 JRZ .NCTS ;IF NOT, CONTINUE
00CF: CB7F BIT CTSN,A ;CHECK CLEAR TO SEND (NOT) STATUS
00D1: CO RNZ ;IF CLEAR TO SEND FALSE, DONE
00D2: 3A 0009# ...NCTS: LDA SOOCHR ;GET SERIAL 0 OUTPUT CHARACTER
00D5: D311 OUT SODATA ;OUTPUT CHARACTER
00D7: 21 00BF# LXI H,SOOPOL ;GET SERIAL 0 OUT POLL ROUTINE
00DA: CD 0000:0B CALL UNLINK# ;UNLINK POLL ROUTINE
00DD: 21 0017# LXI H,SOOSPH ;GET SERIAL 0 OUT SEMAPHORE
00E0: C3 0000:0C JMP SIGNAL# ;SIGNAL PROCESS AS READY

00E3: NOPOP: ;SERIAL 1 OUTPUT POLL ROUTINE
00E3: 0000 .WORD 0 ;SUCCESSOR LINK POINTER
00E5: 0000 .WORD 0 ;PREDECESSOR LINK POINTER
00E7 DB12 IN S1CTRL ;GET SERIAL 1 STATUS
00E9 CB4F BIT TBE,A ;TRANSMIT BUFFER EMPTY?
00EB C8 HZ ;IF NOT, DONE
00EC 21 0004" LXI H,S1BR ;ELSE, GET SERIAL 1 BAUD RATE CODE
00EF CB76 BIT 6,M ;CTS HANDSHAKING REQUESTED?
00F1 2803 JRZ ..NCTS ;IF NOT, CONTINUE
00F3 CB7F BIT CTSN,A ;CHECK CLEAR TO SEND (NOT) STATUS
00F5 C0 RMZ ;IF CLEAR TO SEND FALSE, DONE
00F6 3A 000A" ..NCTS: LDA S10CHR ;GET SERIAL 1 OUTPUT CHARACTER
00F9 D313 OUT S1DATA ;OUTPUT CHARACTER
00FB 21 00E3' LXI H,S1POL ;GET SERIAL 1 OUT POLL ROUTINE
00FE CD 0000:0B CALL UNLINK# ;UNLINK POLL ROUTINE
0101 21 001D" LXI H,S10SPH ;GET SERIAL 1 OUT SEMAPHORE
0104 C3 0000:0C JMP SIGNAL# ;SIGNAL PROCESS AS READY
0107 ED73 0000:0D ŠERISR: SSDP INTSP# ;SAVE STACK POINTER
010B 31 0000:0E LXI SP,INTSTK# ;SET UP AUX STACK POINTER
010E F5 PUSH PSW ;SAVE REGISTERS
010F C5 PUSH B
0110 D5 PUSH D
0111 E5 PUSH H
0112 CD 0126' CALL ..SOI ;CHECK FOR SERIAL 0 INPUT
0115 CD 013D' CALL ..S1I ;CHECK FOR SERIAL 1 INPUT
011B ET POP H ;RESTORE REGISTERS
011C D1 POP D
011D C1 POP B
011E F1 POP PSW
011F ED7B 0000:0D LSPD INTSP# ;RESTORE STACK POINTER
0123 C9 0000:10 JMP ISRXT# ;CONTINUE
0126 DB10 ..SOI: IN SOC0X ;GET SERIAL 0 STATUS
0128 CB47 BIT RDA,A ;CHARACTER AVAILABLE
012A C8 HZ ;IF NOT, DONE
012B 21 000B" LXI H,SOISPH ;GET SERIAL 0 INPUT SEMAPHORE
012E E5 PUSH H ;SAVE SERIAL 0 INPUT SEMAPHORE
012F CD 0000:0C CALL SIGNAL# ;SIGNAL PROCESS AS READY
0131 D1 POP D ;RESTORE SERIAL 0 INPUT SEMAPHORE
0132 DB11 IN SQDATA ;GET SERIAL 0 DATA CHARACTER
0135 4F MOV C,A ;SERIAL 0 DATA CHARACTER TO C-REG
0136 0600 MV1 B,0 ;SET CHANNEL NUMBER=0
0138 21 0003" LXI H,SOBR ;GET SERIAL 0 BAUD RATE
013B 1815 JMPR SERISC ;CONTINUE
013D DB12 ..S1I: IN S1CTRL ;GET SERIAL 1 STATUS
013F CB47 BIT RDA,A ;CHARACTER AVAILABLE
0141 C8 HZ ;IF NOT, DONE
0142 21 0011" LXI H,S1ISPH ;GET SERIAL 1 INPUT SEMAPHORE
0145 E5 PUSH H ;SAVE SERIAL 1 INPUT SEMAPHORE
0146 CD 0000:0C CALL SIGNAL# ;SIGNAL PROCESS AS READY
0149 D1 POP D ;RESTORE SERIAL 1 INPUT SEMAPHORE
014A DB13 IN S1DATA ;GET SERIAL 1 DATA CHARACTER
014C 4F MOV C,A ;SERIAL 1 DATA CHARACTER TO C-REG
014D 0601 MV1 B,1 ;SET CHANNEL NUMBER=1
014F 21 0004" LXI H,S1BR ;GET SERIAL 1 BAUD RATE
PSA Macro Assembler [C12011-0102].

SP442 - TURBODOS OPERATING SYSTEM SERIAL/PARALLEL I/O DRIVER (IMS 442)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0152'  CBE     SERISC:  BIT 7, H ;SIGN BIT ON BAUD RATE CODE?
0154'  281C    JRZ     MCCC ;IF NOT, CONTINUE
0156'  CBB9    RES     7, C ;ELSE, STRIP SIGN BIT ON CHARACTER
0158'  CD 0000:11 CALL    SLVRES# ;CHECK FOR SLAVE RESET
015B'  3A 0000:12 LDA     ATNCHR# ;GET ATTENTION CHARACTER
015E'  91      SUB     C ;CHARACTER=ATTENTION CHARACTER?
0160'  2011    JRNZ    MCCC ;IF NOT, CONTINUE
0161'  3C      INR     A ;ELSE, GET SEMAPHORE COUNT=1
0162'  12      STA     D ;SET SEMAPHORE COUNT=1
0163'  C5      --FBL:  PUSH    B ;SAVE CHARACTER/CHANNEL NUMBER
0164'  CD 0053' CALL    SERST ;GET SERIAL CHANNEL STATUS
0167'  C1      POP     B ;RESTORE CHARACTER/CHANNEL NUMBER
0168'  87      ORA     A ;CHARACTER AVAILABLE?
0169'  2807    JRZ     MCCC ;IF NOT, CONTINUE
016B'  C5      PUSH    B ;SAVE CHARACTER/CHANNEL NUMBER
016C'  CD 018C' CALL    MOVBUF ;MOVE BUFFER TAIL DOWN
016F'  C1      POP     B ;RESTORE CHARACTER/CHANNEL NUMBER
0170'  18F1    JMPR    ..FBL ;CONTINUE
0172'  2A 0000W  ..NCC: LHLH     SERBSZ ;GET SERIAL BUFFER SIZE
0175'  29      DAD    H ;X2
0176'  ED5B 0005W LDED     SERBUF ;GET SERIAL BUFFER ADDRESS
017A'  19      DAD    D ;CALC END OF SERIAL BUFFER ADDRESS
017B'  ED5B 0007W LDED     SERPTR ;GET SERIAL BUFFER POINTER
017F'  87      ORA     A ;CLEAR CARRY FLAG
0180'  ED52    DSBC    D ;SERIAL BUFFER FULL?
0182'  C8      RZ      IF SO, DONE
0183'  EB      XCHG    ;SERIAL BUFFER POINTER TO HL-REG
0184'  70      MOV     M,S ;STORE CHANNEL NUMBER IN BUFFER
0185'  23      INX    H
0186'  71      MOV     M,C ;STORE INPUT CHARACTER IN BUFFER
0187'  23      INX    H
0188'  22 0007W SHLD     SERPTR ;UPDATE SERIAL BUFFER POINTER
018B'  C9      RET     ;DONE
018C'  EB      MOVBUF: XCHG    ;SOURCE ADDRESS TO DE-REG
018D'  2A 0007W LHLH     SERPTR ;GET SERIAL BUFFER POINTER
0190'  ED52    DSBC    D ;CALC LENGTH OF TAIL TO MOVE DOWN
0192'  4D      MOV     C,L ;LENGTH OF TAIL TO BC-REG
0193'  44      MOV     B,H
0194'  EB      XCHG    ;SOURCE ADDRESS TO HL-REG
0195'  5D      MOV     E,L ;COPY SOURCE ADDRESS INTO DE-REG
0196'  54      MOV     D,H ;CALC DESTINATION ADDRESS
0197'  1B      DCX    D ;IF LENGTH OF TAIL=0, CONTINUE
0198'  1B      DCX    D ;ELSE, MOVE TAIL DOWN
0199'  2802    JRZ     ..X ;IF LENGTH OF TAIL=0, CONTINUE
019B'  EDB0    LDIR    SERPTR ;UPDATE SERIAL BUFFER POINTER
019D'  ED53 0007W  ..X: SDED     RET     ;DONE
01A1'  C9      RET     ;DONE

01A2'  78      SERSBR: MOV     A,B ;GET CHANNEL NUMBER
01A3'  FED2    CPI     2 ;CHANNEL NUMBER=0/1?
01A5'  D2 0000:13 JNC     SBBR480# ;IF NOT, CONTINUE
01A8'  21 0003W LXI     H,SBR ;ELSE, GET SERIAL 0 BAUD RATE
01AB'  B7      ORA     A ;CHANNEL NUMBER=0?
01AC'  2801  JRZ  ..COM1  ;IF SO, CONTINUE
01AE'  23   INX  H  ;ELSE, GET SERIAL 1 BAUD RATE
01AF'  71   ..COM1: MOV  M,C  ;SAVE BAUD RATE CODE
01B0'  CD 01C6' CALL GETBTV  ;GET BAUD RATE TIMER VALUE
01B3'  73   MOV  A,B  ;GET CHANNEL NUMBER
01B4'  87   ORA  A  ;CHANNEL NUMBER=0?
01B5'  3E36  MVI  A,T0CMD  ;GET TIMER 0 COMMAND
01B7'  0E14  MVI  C,TIMO  ;GET TIMER 0 DATA REGISTER
01B9'  2804  JRZ  ..COM2  ;IF CHANNEL NUMBER=0, CONTINUE
01BB'  3E76  MVI  A,T1CMD  ;ELSE, GET TIMER 1 COMMAND
01BD'  0E15  MVI  C,TIM1  ;GET TIMER 1 DATA REGISTER
01BF'  D317  ..COM2: OUT  TIMCTL  ;SELECT TIMER
01C1'  ED59  OUTP  E  ;OUTPUT LSB OF TIMER VALUE
01C3'  ED51  OUTP  D  ;OUTPUT MSB OF TIMER VALUE
01C5'  C9  RET  ;DONE

01C6'  79  GETBTV: MOV  A,C  ;GET REQUESTED BAUD RATE CODE
01C7'  E60F  ANI  0FH  ;EXTRACT RELEVANT BITS
01C9'  87  ADD  A  ;X2
01CA'  5F  MOV  E,A  ;TO E-REG
01CB'  1600  MVI  D,0  ;MAKE IT DOUBLE LENGTH
01CD'  21 0000:14  LXI  H,BRTBL#  ;GET BAUD RATE TABLE
01D0'  19  DAD  D  ;INDEX INTO TABLE
01D1'  5E  MOV  E,M  ;GET TIMER VALUE
01D3'  23  INX  H
01D4'  C9  RET  ;DONE

01D5'  78  SERRBR: MOV  A,B  ;GET CHANNEL NUMBER
01D6'  FE02  CPI  2  ;CHANNEL NUMBER=0/1?
01D8'  D2 0000:15  JNC  RBR480#  ;IF NOT, CONTINUE
01DB'  21 0003"  LXI  H,SOBR  ;ELSE, GET SERIAL 0 BAUD RATE
01DE'  87  ORA  A  ;CHANNEL NUMBER=0?
01DF'  2801  JRZ  ..COM  ;IF SO, CONTINUE
01E1'  23  INX  H  ;ELSE, GET SERIAL 1 BAUD RATE
01E2'  7E  ..COM: MOV  A,M  ;GET CURRENT BAUD RATE CODE
01E3'  C9  RET  ;DONE

01E4'  78  SERSMC: MOV  A,B  ;GET CHANNEL NUMBER
01E5'  FE02  CPI  2  ;CHANNEL NUMBER=0/1?
01E7'  D2 0000:16  JNC  SMC480#  ;IF NOT, CONTINUE
01EA'  87  ORA  A  ;CHANNEL NUMBER=0?
01EB'  3A 0002"  LDA  INTMSK  ;GET INT MASK
01EE'  200A  JRNZ  ..CH1  ;IF CHANNEL NUMBER NOT=0, CONTINUE
01F0'  CBBF  RES  SORTSN,A  ;CLEAR SERIAL 0 REQ TO SEND (NOT)
01F2'  CB79  BIT  7,C  ;SERIAL 0 REQ TO SEND TO BE ON?
01F4'  200C  JRNZ  ..COM  ;IF SO, CONTINUE
01F6'  CBFF  SET  SORTSN,A  ;ELSE, SET SERIAL 0 RTS (NOT)
01F8'  1808  JMPR  ..COM  ;CONTINUE
01FA'  CBA7  ..CH1: RES  S1RTSN,A  ;CLEAR SERIAL 1 REQ TO SEND (NOT)
01FC'  CB79  BIT  7,C  ;SERIAL 1 REQ TO SEND TO BE ON?
01FE'  2002  JRNZ  ..COM  ;IF SO, CONTINUE
0200'  CB77  SET  S1RTSN,A  ;ELSE, SET SERIAL 1 RTS (NOT)
0202'  32 0002"  ..COM: STA  INTMSK  ;UPDATE INT MASK

---

"PSA Macro Assembler [C12011-0102]" Page 7
"©442 - TURBODOS OPERATING SYSTEM SERIAL/PARALLEL I/O DRIVER (IMS 442)" RIGHT (C) 1981, SOFTWARE 2000, INC."
PSA Macro Assembler [C12011-0102]

SP442 - TURBOS Operating System Serial/Parallel I/O Driver (IMS 442)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0205' D318 OUT SINTE ; SET SERIAL 1 REQUEST TO SEND
0207' C9 RET ; DONE

0208' 78 SERRMC: MOV A,B ; GET CHANNEL NUMBER
0209' FE02 CPI 2 ; CHANNEL NUMBER=0/1?
0208' D2 0000:17 JNC RMC480# ; IF NOT, CONTINUE
020E' B7 ORA A ; CHANNEL NUMBER=0?
020F' DB12 IN S1CTRL ; GET SERIAL 0 STATUS
0211' 2B02 JRZ .COM ; IF CHANNEL NUMBER=0, CONTINUE
0213' DB12 IN S1CTRL ; ELSE, GET SERIAL 1 STATUS
0215' EE80 .COM: ANI 1<CTSN ; EXTRACT CLEAR TO SEND (NOT)
0217' EE80 XRI 1<CTSN ; COMPLIMENT IT
0219' C9 RET ; DONE

021A' D31C P0OUT: OUT P0DATA ; OUTPUT BYTE TO PARALLEL 0
021C' C9 RET ; DONE

021D' D31D P1OUT: OUT P1DATA ; OUTPUT BYTE TO PARALLEL 1
021F' C9 RET ; DONE

0220' DB1E P2IN: IN P2DATA ; INPUT BYTE FROM PARALLEL 2
0222' C9 RET ; DONE

0223' 21 0002" EBPROM: LXI H,INTMSK ; GET INTERRUPT MASK
0226' CB86 RES ROMDIS,M ; RESET ROM DISABLE BIT
0228' 7E MOV A,M ; GET INTERRUPT MASK
0229' D318 OUT SINTE ; TURN ON BOOTSTRAP ROM
022B' 21 FC00 LXI H,BOOTPR ; RETURN BOOT PROM ADDR
022E' C9 RET ; DONE

022F' 21 0002" DBPROM: LXI H,INTMSK ; GET INTERRUPT MASK
0232' CBC6 SET ROMDIS,M ; SET ROM DISABLE BIT
0234' 7E MOV A,M ; GET INTERRUPT MASK
0235' D318 OUT SINTE ; TURN OFF BOOTSTRAP ROM
0237' C9 RET ; DONE

; END
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

AUTHORS: RONALD E. RAIKES
MICHAEL D. BUSCH

VERSION: 09/08/81

IDENT SER480

I/O PORT BASE

I/OBASE = 0E0H

S2DATA = I/OBASE+00H
00E0
S2IER = I/OBASE+01H
00E1
S2IIIDR = I/OBASE+02H
00E2
S2LCSR = I/OBASE+03H
00E3
S2MCR = I/OBASE+04H
00E4
S2LSR = I/OBASE+05H
00E5
S2MSR = I/OBASE+06H
00E6
S3DATA = I/OBASE+08H
00E8
S3IER = I/OBASE+09H
00E9
S3IIDR = I/OBASE+0AH
00EA
S3LCSR = I/OBASE+0BH
00EB
S3MCR = I/OBASE+0CH
00EC
S3LSR = I/OBASE+ODH
00ED
S3MSR = I/OBASE+0EH
00EE
S4DATA = I/OBASE+10H
00F0
S4IER = I/OBASE+11H
00F1
S4IIDR = I/OBASE+12H
00F2
S4LCSR = I/OBASE+13H
00F3
S4MCR = I/OBASE+14H
00F4
S4LSR = I/OBASE+15H
00F5
S4MSR = I/OBASE+16H
00F6
S5DATA = I/OBASE+18H
00F8
S5IER = I/OBASE+19H
00F9
S5IIDR = I/OBASE+1AH
00FA
S5LCSR = I/OBASE+1BH
00FB
S5MCR = I/OBASE+1CH
00FC
S5LSR = I/OBASE+1DH
00FD
S5MSR = I/OBASE+1EH
00FE

IERTCW = 01H
0001
LCRCW = 03H
0003
MRCRW = 03H
0003

RDA = 0
0000
TBE = 5
0005
CTS = 4
0004

RECEIVED DATA AVAILABLE BIT
TRANSMIT BUFFER EMPTY BIT
CLEAR TO SEND BIT
; LOCATE IN DATA AREA

0001W 00  S2BR: .BYTE 0 ;SERIAL 2 BAUD RATE
0002W 00  S3BR: .BYTE 0 ;SERIAL 3 BAUD RATE
0003W 00  S4BR: .BYTE 0 ;SERIAL 4 BAUD RATE
0004W 00  S5BR: .BYTE 0 ;SERIAL 5 BAUD RATE
0005W 00  S2OCHR: .BYTE 0 ;SERIAL 2 OUTPUT CHARACTER
0006W 00  S3OCHR: .BYTE 0 ;SERIAL 3 OUTPUT CHARACTER
0007W 00  S4OCHR: .BYTE 0 ;SERIAL 4 OUTPUT CHARACTER
0008W 00  S5OCHR: .BYTE 0 ;SERIAL 5 OUTPUT CHARACTER

0009W 00  S2ISPH: .WORD 0 ;SERIAL 2 INPUT SEMAPHORE
000AW 0000  ;SEMAPHORE COUNT
000CW 0000  ;SEMAPHORE P/D HEAD

000EW 0000  S3ISPH: .WORD 0 ;SEMAPHORE COUNT
0010W 0010  ;SEMAPHORE P/D HEAD
0012W 0010  ;SEMAPHORE P/D HEAD

0014W 0000  S4ISPH: .WORD 0 ;SEMAPHORE COUNT
0016W 0016  ;SEMAPHORE P/D HEAD
0018W 0016  ;SEMAPHORE P/D HEAD

001AW 0000  S5ISPH: .WORD 0 ;SEMAPHORE COUNT
001CW 001C  ;SEMAPHORE P/D HEAD
001EW 001C  ;SEMAPHORE P/D HEAD

0020W 0000  S2OSP: .WORD 0 ;SEMAPHORE COUNT
0022W 0022  ;SEMAPHORE P/D HEAD
0024W 0022  ;SEMAPHORE P/D HEAD

0026W 0000  S3OSP: .WORD 0 ;SEMAPHORE COUNT
0028W 0028  ;SEMAPHORE P/D HEAD
002AW 0028  ;SEMAPHORE P/D HEAD

002CW 0000  S4OSP: .WORD 0 ;SEMAPHORE COUNT
002EW 002E  ;SEMAPHORE P/D HEAD
0030W 002E  ;SEMAPHORE P/D HEAD

0032W 0000  S5OSP: .WORD 0 ;SEMAPHORE COUNT
0034W 0034  ;SEMAPHORE P/D HEAD
0036W 0034  ;SEMAPHORE P/D HEAD

0000* .LOC  .PROG.* ;LOCATE IN PROGRAM AREA
ER480 – TURBODOS OPERATING SYSTEM SERIAL DRIVER (IMS 480)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0000  0026  .WORD  NITLEN+2  ;INITIALIZATION CODE LENGTH
0002  3E03  NIT480::MVI  A, LCRCW  ;GET LINE CONTROL REGISTER VALUE
0004  D3E3  OUT  S2LCR  ;SET LINE CONTROL REGISTER 0
0006  D3ED  OUT  S3LCR  ;SET LINE CONTROL REGISTER 1
0008  D3F3  OUT  S4LCR  ;SET LINE CONTROL REGISTER 2
000A  D3FF  OUT  S5LCR  ;SET LINE CONTROL REGISTER 3
000C  3E03  MVI  A, MCRCW  ;GET MODEM CONTROL REGISTER VALUE
000E  D3E4  OUT  S2MCR  ;SET MODEM CONTROL REGISTER 0
0010  D3EC  OUT  S3MCR  ;SET MODEM CONTROL REGISTER 1
0012  D3F4  OUT  S4MCR  ;SET MODEM CONTROL REGISTER 2
0014  D3F8  OUT  S5MCR  ;SET MODEM CONTROL REGISTER 3
0016  3E01  MVI  A, IERCW  ;GET INT ENABLE REGISTER VALUE
0018  D3E1  OUT  S2IER  ;SET INT ENABLE REGISTER 0
001A  D3E9  OUT  S3IER  ;SET INT ENABLE REGISTER 1
001C  D3F1  OUT  S4IER  ;SET INT ENABLE REGISTER 2
001E  D3F9  OUT  S5IER  ;SET INT ENABLE REGISTER 3
0020  21 0002  ;LXI  H, NIT480  ;GET INITIALIZATION CODE ADDRESS
0023  C3 0000:04  ;JMP  DEALOC#  ;DE-ALLOCATE INITIALIZATION CODE

0024  NITLEN = -.NIT480  ;INITIALIZATION CODE LENGTH

0026  21 0008  ;LXI  H, S2ISP  ;GET SERIAL 2 IN SEMAPHORE
0029  D602  SUI  2  ;REMOVE CHANNEL NUMBER BIAS
002B  C8  ;IF CHANNEL NUMBER=2, DONE
002C  21 000E  ;LXI  H, S3ISP  ;ELSE, GET SERIAL 3 IN SEMAPHORE
002F  3D  DCR  A  ;CHANNEL NUMBER=3?
0030  C8  ;IF SO, DONE
0031  21 0014  ;LXI  H, S4ISP  ;ELSE, GET SERIAL 4 IN SEMAPHORE
0034  3D  DCR  A  ;CHANNEL NUMBER=4?
0035  C8  ;IF SO, DONE
0036  21 001A  ;LXI  H, S5ISP  ;ELSE, GET SERIAL 5 IN SEMAPHORE
0039  C9  ;DONE

003A  21 0004  ;LXI  H, S20CHR  ;GET SERIAL 2 OUTPUT CHARACTER
003D  CD 01B7  ;CALL  CHHNBC  ;DO COMMON SETUP
0040  71  ;MOV  H, C  ;SAVE OUTPUT CHARACTER
0044  21 0020  ;LXI  H, S2OSP  ;GET SERIAL 2 OUT SEMAPHORE
0047  11 005F  ;LXI  D, S2OPOL  ;GET SERIAL 2 OUT POLL ROUTINE
004F  C8  ;IF CHANNEL NUMBER=2, DONE
0050  21 0026  ;LXI  H, S3OSP  ;ELSE, GET SERIAL 3 OUT SEMAPHORE
0053  11 0085  ;LXI  D, S3OPOL  ;GET SERIAL 3 OUT POLL ROUTINE
0056  3D  DCR  A  ;CHANNEL NUMBER=3?
0057  C8  ;IF SO, DONE
005A  21 002C  ;LXI  H, S4OSP  ;ELSE, GET SERIAL 4 OUT SEMAPHORE
005D  11 00AB  ;LXI  D, S4OPOL  ;GET SERIAL 4 OUT POLL ROUTINE
005F  3D  DCR  A  ;CHANNEL NUMBER=4?
0062  C8  ;IF SO, DONE
0065  21 0032  ;LXI  H, S5OSP  ;ELSE, GET SERIAL 5 OUT SEMAPHORE
0068  11 00D1  ;LXI  D, S5OPOL  ;GET SERIAL 5 OUT POLL ROUTINE
006B  C9  ;DONE

005F  0000  ;WORD 0  ;SUCCESSOR LINK POINTER

005F  S2OPOL:  .WORD 0  ;SERIAL 2 OUTPUT POLL ROUTINE
SER480 - TURBODOS OPERATING SYSTEM SERIAL DRIVER (IMS 480)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0061' 0000  .WORD 0 ;PREDECESSOR LINK POINTER
0063' DBE5  IN  S2LSR ;GET SERIAL 2 LINE STATUS REGISTER
0065' CB6F  BIT  TBE,A ;TRANSMIT BUFFER EMPTY?
0067' C8  RZ ;IF NOT, DONE
0068' 21 0000* LXI  H,S2BR ;ELSE, GET SERIAL 2 BAUD RATE
006B' CB76  BIT  6,M ;CLEAR TO SEND HANDSHAKING REQUESTER
006D' 2805  JBRZ  ..NCTS ;IF NOT, CONTINUE
006F' DBE6  IN  S2MSR ;GET SERIAL 2 MODEM STATUS REGISTER
0071' CB67  BIT  CTS,A ;CLEAR TO SEND STATUS TRUE?
0073' C8  RZ ;IF NOT, DONE
0074 3A 0004* ..NCTS: LDA  S2OCHR ;GET SERIAL 2 OUTPUT CHARACTER
0077' D3ED  OUT  S2DATA ;OUTPUT CHARACTER
0079' 21 005F* LXI  H,S2OPOL ;GET SERIAL 2 OUT POLL ROUTINE
007C' CD 0000:05 CALL  UNLINK# ;UNLINK POLL ROUTINE
007F' 21 0020* LXI  H,S2OSP ;GET SERIAL 2 OUT SEMAPHORE
0082' C3 0000:06 JMP  SIGNAL# ;SIGNAL PROCESS AS READY
0085' S3POL: ;SERIAL 3 OUTPUT POLL ROUTINE
0085' 0000  .WORD 0 ;SUCCESSOR LINK POINTER
0087' 0000  .WORD 0 ;PREDECESSOR LINK POINTER
0089' DBED  IN  S3LSR ;GET SERIAL 3 LINE STATUS REGISTER
008B' CB6F  BIT  TBE,A ;TRANSMIT BUFFER EMPTY?
008D' C8  RZ ;IF NOT, DONE
008E' 21 0001* LXI  H,S3BR ;ELSE, GET SERIAL 3 BAUD RATE
0091' CB76  BIT  6,M ;CLEAR TO SEND HANDSHAKING REQUESTER
0093' 2805  JBRZ  ..NCTS ;IF NOT, CONTINUE
0095' DBEE  IN  S3MSR ;GET SERIAL 3 MODEM STATUS REGISTER
0097' CB67  BIT  CTS,A ;CLEAR TO SEND STATUS TRUE?
0099' C8  RZ ;IF NOT, DONE
009A 3A 0005* ..NCTS: LDA  S3OCHR ;GET SERIAL 3 OUTPUT CHARACTER
009D' D3ED  OUT  S3DATA ;OUTPUT CHARACTER
009F' 21 0085' LXI  H,S3OPOL ;GET SERIAL 3 OUT POLL ROUTINE
00AD' CD 0000:05 CALL  UNLINK# ;UNLINK POLL ROUTINE
00A5' 21 0026* LXI  H,S3OSP ;GET SERIAL 3 OUT SEMAPHORE
00A8' C3 0000:06 JMP  SIGNAL# ;SIGNAL PROCESS AS READY
00AB' S4POL: ;SERIAL 4 OUTPUT POLL ROUTINE
00AB' 0000  .WORD 0 ;SUCCESSOR LINK POINTER
00AD' 0000  .WORD 0 ;PREDECESSOR LINK POINTER
00AF' DBF5  IN  S4LSR ;GET SERIAL 4 LINE STATUS REGISTER
00B1' CB6F  BIT  TBE,A ;TRANSMIT BUFFER EMPTY?
00B3' C8  RZ ;IF NOT, DONE
00B4' 21 0002* LXI  H,S4BR ;ELSE, GET SERIAL 4 BAUD RATE
00B7' CB76  BIT  6,M ;CLEAR TO SEND HANDSHAKING REQUESTER
00BB' 2805  JBRZ  ..NCTS ;IF NOT, CONTINUE
00B5' DBF6  IN  S4MSR ;GET SERIAL 4 MODEM STATUS REGISTER
00B7' CB67  BIT  CTS,A ;CLEAR TO SEND STATUS TRUE?
00BF' C8  RZ ;IF NOT, DONE
```
PSA Macro Assembler [C12011-0102 ]

©1981, SOFTWARE 2000, INC.

00C0' 3A 0006" "NCTS: LDA S4OCHR ;GET SERIAL 4 OUTPUT CHARACTER
00C3' D3F0 OUT S4DATA ;OUTPUT CHARACTER
00C5' 21 00AB' LXT H,S4OPOL ;GET SERIAL 4 OUT POLL ROUTINE
00C8' CD 0000:05 CALL UNLINK;UNLINK POLL ROUTINE
00CB' 21 002C" LXI H,S4OSP ;GET SERIAL 4 OUT SEMAPHORE
00CE' C3 0000:06 JMP SIGNAL# ;SIGNAL PROCESS AS READY

00D1' S50POL: ;SERIAL 5 OUTPUT POLL ROUTINE
00D3' 0000 .WORD 0 ;SUCCESSOR LINK POINTER
00D5' DBFD IN S5LSR ;GET SERIAL 5 LINE STATUS REGISTER
00D7' CB6F BIT TBE,A ;TRANSMIT BUFFER EMPTY?
00D9' C8 RZ ;IF NOT, DONE
00DA' 21 0003" LXT H,S5BR ;ELSE, GET SERIAL 5 BAUD RATE
00DD' CB76 BIT 6, H ;CLEAR TO SEND HANDSHAKING REQUESTER

00DF' 2805 JRZ .NCTS ;IF NOT, CONTINUE
00E1' DBFE IN S5MSR ;GET SERIAL 5 MODEM STATUS REGISTER
00E3' CB67 BIT CTS,A ;CLEAR TO SEND STATUS TRUE?
00E5' C8 RZ ;IF NOT, DONE
00E6' 3A 0007" "NCTS: LDA S5OCHR ;GET SERIAL 5 OUTPUT CHARACTER
00E9' D3F8 OUT S5DATA ;OUTPUT CHARACTER
00EB' 21 00D1' LXT H,S5OPOL ;GET SERIAL 5 OUT POLL ROUTINE
00EE' CD 0000:05 CALL UNLINK;UNLINK POLL ROUTINE
00F1' 21 0032" LXT H,S5OSP ;GET SERIAL 5 OUT SEMAPHORE
00F4' CB76 .WORD 0 ;PREDECESSOR LINK POINTER

00FA' CD 0116' CALL .S3I ;CHECK FOR SERIAL 3 INPUT
00FD' CD 0128' CALL .S4I ;CHECK FOR SERIAL 4 INPUT
0100' CD 013A' CALL .S5I ;CHECK FOR SERIAL 5 INPUT
0103' C9 RET ;DONE

0104' DBE5 .S2I: IN S2LSR ;GET SERIAL 2 STATUS
0106' CB47 BIT RDA,A ;CHARACTER AVAILABLE
0108' C8 RZ ;IF NOT, DONE
0109' E5 21 0008" LXI H,S2ISP ;GET SERIAL 2 INPUT SEMAPHORE
010C' E5 PUSH H ;SAVE SERIAL 2 INPUT SEMAPHORE
010D' CD 0000:06 CALL SIGNAL# ;SIGNAL PROCESS AS READY
0110' DBE0 IN S2DATA ;GET SERIAL 2 DATA CHARACTER
0112' 0602 MVI B, 2 ;SET CHANNEL NUMBER=2
0114' 1834 JMPR .SIC ;CONTINUE
0116' DBED .S3I: IN S3LSR ;GET SERIAL 3 STATUS
0118' CB47 BIT RDA,A ;CHARACTER AVAILABLE
011A' C8 RZ ;IF NOT, DONE
011B' 21 000E" LXI H,S3ISP ;GET SERIAL 3 INPUT SEMAPHORE
011E' E5 PUSH H ;SAVE SERIAL 3 INPUT SEMAPHORE
011F' CD 0000:06 CALL SIGNAL# ;SIGNAL PROCESS AS READY
0122' DBE8 IN S3DATA ;GET SERIAL 3 DATA CHARACTER
0124' 0603 MVI B, 3 ;SET CHANNEL NUMBER=3
0126' 1822 JMPR .SIC ;CONTINUE
0128' DBF5 .S4I: IN S4LSR ;GET SERIAL 4 STATUS
012A' CB47 BIT RDA,A ;CHARACTER AVAILABLE
```
PSA Macro Assembler [C12011-0102 ]

SER480 - TURBODOS OPERATING SYSTEM SERIAL DRIVER (IMS 480)
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

012C: C3 RZ ;IF NOT, DONE
012D: 21 0014 H, S4ISP H ;GET SERIAL 4 INPUT SEMAPHORE
0130: 25 PUSH H ;SAVE SERIAL 4 INPUT SEMAPHORE
0131: CD 0000:06 CALL SIGNAL# ;SIGNAL PROCESS AS READY
0134: DBFO IN S4DATA ;GET SERIAL 4 DATA CHARACTER
0135: 0604 MVI B, 4 ;SET CHANNEL NUMBER=4
0138: 1810 JMPR ..SIC ;CONTINUE
0139: DBFD ..S5I: IN S5LSR ;GET SERIAL 5 STATUS
013C: CB47 BIT RDA,A ;CHARACTER AVAILABLE
013E: C3 RZ ;IF NOT, DONE
013F: 21 001A H, S5ISP H ;GET SERIAL 5 INPUT SEMAPHORE
0142: E5 PUSH H ;SAVE SERIAL 5 INPUT SEMAPHORE
0143: CD 0000:06 CALL SIGNAL# ;SIGNAL PROCESS AS READY
0146: DBF8 IN S5DATA ;GET SERIAL 5 DATA CHARACTER
0148: 0605 MVI B, 5 ;SET CHANNEL NUMBER=5
014A: 4F ..SIC: MOV C, A ;SERIAL DATA CHARACTER TO C-REG
014B: 78 MOV A, B ;GET CHANNEL NUMBER
014C: 21 0000 H, S2BR ;GET SERIAL 2 BAUD RATE
014F: CD 01B7 CALL CHNDBC ;DO COMMON SETUP
0152: D1 POP D ;RESTORE SERIAL INPUT SEMAPHORE
0153: C3 0000:07 JMP SERISC# ;CONTINUE
0156: 21 0000 8B480: LXI H, S2BR ;GET SERIAL 2 BAUD RATE
0159: CD 01B7 CALL CHNDBC ;DO COMMON SETUP
015C: F5 PUSH PSW ;SAVE CHANNEL NUMBER
015D: 71 MOV M, C ;SAVE BAUD RATE CODE
015E: CD 0000:08 CALL GETBTV# ;GET BAUD RATE TIMER VALUE
0161: F1 POP PSW ;RESTORE CHANNEL NUMBER
0162: 87 ADD A ;X2
0163: 87 ADD A ;X2=X4
0164: 87 ADD A ;X2=X8
0165: F6E3 ORI IOBASE+3 ;CALC LINE CONTROL REGISTER
0167: 4F MOV C, A ;LINE CONTROL REGISTER TO C-REG
016B: 3E83 MVI A, LRCW180H ;GET DIVISOR LATCH ACCESS BIT
016A: ED79 OUTP A ;SELECT DIVISOR LATCH
016C: 0D DCR C ;CALC DATA REGISTER
016D: 0D DCR C
016E: 0D DCR C
016F: ED59 OUTP E ;OUTPUT LSB OF BAUD RATE TIMER VALUE
0171: 0C INR C ;CALC DATA REGISTER+1
0172: ED51 OUTP D ;OUTPUT MSB OF BAUD RATE TIMER VALUE
0174: 0C INR C ;CALC LINE CONTROL REGISTER
0175: 0C INR C
0176: 3E03 MVI A, LRCW ;GET LINE CONTROL REGISTER VALUE
0178: ED79 OUTP A ;DE-SELECT DIVISOR LATCH
017A: C9 RET ;DONE
017B: 21 0000 8BR480: LXI H, S2BR ;GET SERIAL 2 BAUD RATE
017E: CD 01B7 CALL CHNDBC ;DO COMMON SETUP
0181: 7E MOV A, M ;GET CURRENT BAUD RATE
0182: C9 RET ;DONE
SMC480: MOV A,C ;GET REQUESTED MODEM CONTROLS
         MOV RRC RRC ;SHIFT MODEM CONTROLS INTO BITS 6/7
         MOV D,A ;MODEM CONTROL REGISTER VALUE TO D-F
SMC480: MOV A,B ;GET CHANNEL NUMBER
         SUI 2 ;REMOVE CHANNEL NUMBER BIAS
         ADD A ;X2
         ADD A ;X2=X4
         ADD A ;X2=X8
         ORI IOBASE+4 ;CALC MODEM CONTROL REGISTER
SMC480: MOV C,A ;MODEM CONTROL REGISTER TO C-REG
SMC480: RET D ;OUTPUT MODEM CONTROLS
         ;DONE
SMC480: MOV A,B ;GET CHANNEL NUMBER
         SUI 2 ;REMOVE CHANNEL NUMBER BIAS
         ADD A ;X2
         ADD A ;X2=X4
         ADD A ;X2=X8
         ORI IOBASE+6 ;CALC MODEM STATUS REGISTER
SMC480: MOV C,A ;MODEM STATUS REGISTER TO C-REG
SMC480: INP D ;GET MODEM STATUS REGISTER
         XRA A ;SET RETURN CODE=0
         BIT 4,D ;CLEAR TO SEND BIT SET?
         JRZ ..NCTS ;IF NOT, CONTINUE
         MOV E,A ;CHANNEL NUMBER TO DE-REG
         DAD D INDEX INTO CHARACTER SAVE AREA
         RET ..NCTS
         ;DONE
         ;DONE
         ;DONE
         ;DONE
         ;DONE
         ;DONE
         ;DONE
BRT442 — TURBODOS OPERATING SYSTEM IMS SERIAL PORT BAUD RATE TABLE (OPTIONAL)
COPYRIGHT (C) 1980 BY SOFTWARE 2000, INC.

; COPYRIGHT (C) 1980 BY SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIKES
          MICHAEL D. BUSCH
; VERSION: 09/08/81
.
. IDENT  BRT442 ; MODULE ID
. INSERT DEQUATE ; 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
0016'  0015  .WORD  BR3600 ;3600 BAUD TIMER VALUE
0018'  0010  .WORD  BR4800 ;4800 BAUD TIMER VALUE
001A'  000B  .WORD  BR7200 ;7200 BAUD TIMER VALUE
001C'  0008  .WORD  BR9600 ;9600 BAUD TIMER VALUE
001E'  0004  .WORD  BR192K ;19200 BAUD TIMER VALUE
.
.END
PSA Macro Assembler [C12011-0102]

3K401 - TURBODOS OPERATING SYSTEM IMS FLOPPY DISK DRIVER
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

; COPYRIGHT (C) 1981, SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIKES
; MICHAEL D. BUSCH
; VERSION: 09/10/81

; IDENT DSK401
; MODULE ID

; 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 (83ARD)
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
FDCSFT = 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 COMMAND
000A
FDCRID = 0AH ; FDC READ ID COMMAND
000D
FDCFMT = 0DH ; FDC FORMAT TRACK COMMAND
000F
FDCSK = 0FH ; FDC SEEK COMMAND
0005
FDCVR = 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
FDCDFM = 6 ; FDC DOUBLE-DENSITY BIT
0007
FDCNT = 7 ; FDC MULTI-TRACK BIT

0004
FDCBSTY = 4 ; FDC BUSY STATUS
0005
FDCSE = 5 ; FDC SEEK END
0006
FDCOUT = 6 ; FDC OUTPUT MODE
0007
FDCRBY = 7 ; FDC READY FOR DATA

00D0
SRTSR = (16-3)<4 ; 8 INCH FDD STEP RATE (3 MS-REMEX)
00A0
SRT8S = (16-6)<4 ; 8 INCH FDD STEP RATE (6 MS-SHUG)

0024
HDLT = 18*2 ; FDD HEAD LOAD TIME (36 MS)
0001
H Dut = 1 ; FDD HEAD UNLOAD TIME (16 MS)

0003
STONR = 3 ; STATUS REGISTER 0 NOT READY
0004
STOEC = 4 ; STATUS REGISTER 0 EQUIP CHECK
PSA Macro Assembler [C12011-0102]

DSK401 — TURBODOC OPERATING SYSTEM IMS FLOPPY DISK DRIVER
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

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

0000  SLOWSR = 0 ;SLOW STEP RATE (FLAGS)

0002  TSB = 2 ;TWO-SIDED DISK BIT (TYPE CODE)
0003  DDD = 3 ;DOUBLE DENSITY DISK BIT (TYPE CODE)

0004  MINI = 4 ;MINI—FLOPPY DISK BIT (TYPE CODE)

0000'  .LOC ;LOCATE IN PROGRAM AREA

0000'  21 0000" DSKDR%::LXI H,DMXSPH ;GET MUTUAL EXCLUSION SEMAPHORE
0003'  CD 0000:04 CALL WAIT# ;DISPATCH IF NECESSARY
0006'  CD 0012' CALL ..DD ;CALL DISK DRIVER
0009'  F5  PUSH PSW ;SAVE RETURN CODE
000A'  21 0000" LXI H,DMXSPH ;GET MUTUAL EXCLUSION SEMAPHORE
000D'  CD 0000:05 CALL SIGNAL# ;SIGNAL PROCESS AS READY
0010'  F1  POP PSW ;RESTORE RETURN CODE
0011'  C9  RET ;DONE

0012'  ED73 0012" ..DD: SSPPD RETSP ;SAVE ERROR RETURN STACK POINTER
0016'  DD7EOO MOV A,PDRFCN(X) ;GET PD REQ FUNCTION NUMBER
0019'  B7  ORA A ;PD REQ FUNCTION NUMBER=0?
001A'  283D JBRZ RDSSK ;IF SO, CONTINUE
001C'  3D  DCR A ;PD REQ FUNCTION NUMBER=1?
001D'  284C JBRZ WDDSK ;IF SO, CONTINUE
001F'  3D  DCR A ;PD REQ FUNCTION NUMBER=2?
0020'  CA 028A' JZ RETDST ;IF SO, CONTINUE
0023'  3D  DCR A ;PD REQ FUNCTION NUMBER=3?
0024'  CA 0303' JZ RETRKY ;IF SO, CONTINUE
0027'  3D  DCR A ;PD REQ FUNCTION NUMBER=4?
0028'  285C JBRZ FMTDSK ;IF SO, CONTINUE
002A'  C9  RET ;ELSE, DONE

002B'  002E ;.WORD NITLEN+2 ;INITIALIZATION CODE LENGTH

002D'  DB8E DSKIN%::IN FDCST ;GET FDC STATUS
002F'  3C  INR A ;CONTROLLER PRESENT?
0030'  2821 JBRZ ..X ;IF NOT, CONTINUE
0032'  3EC3 MVI A,JMP ;ELSE, INITIALIZE INTERRUPT VECTOR
0034'  32 0028 STA 5#8 ;(VECTORED INTERRUPT-5)
0037 21 0387' LXI H,DSKISR
003A 22 0029 SHL (5*8)+1
003D AF XRA A
003E D3E8 OUT DMACTL ;DISABLE DMA CONTROLLER
0040 3E03 MVI A,FDCSFY ;GET FDC SPECIFY COMMAND
0042 CD 0412' CALL CMDRDY ;OUTPUT COMMAND TO FDC
0045 3ED1 MVI A,SR8B1HDUT ;GET REMEX STEP RT/HEAD UNLD
0047 CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
004A 3E24 MVI A,HDLT ;GET HEAD LOAD TIME/NON-DMA BIT
004C CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
004F 3E01 MVI A,1<DSKENDI
0051 D38C OUT DSCCTL ;ENABLE CONTROLLER INTERRUPTS
0053 21 002D' ..X: LXI H,DSKIN; ;GET INITIALIZATION CODE ADDRESS
0056 C3 0000:06 JMP DEALOC; ;DE-ALLOCATE INITIALIZATION CODE

0059 3ED8 MVI A,MAXTRY ;GET MAX TRY COUNT
005B 32 000C' STA TRYCNT ;SET TRY COUNT
005E 3ED6 ..RD: MVI A,FDCRD ;GET FDC READ COMMAND
0060 0E40 MVI C,DMARD ;GET DMA READ COMMAND
0062 CD 00F2' CALL DSKCOM ;CALL COMMON CODE
0065 C8 RJNZ ;NO ERRORS, RET A=0
0066 CD 0149' CALL RETRY ;ERRORS, RECALIBRATE
0069 18F3 JMPR ..RD ;TRY AGAIN

006B 3ED8 MVI A,MAXTRY ;GET MAX TRY COUNT
006D 32 000C' STA TRYCNT ;SET TRY COUNT
0070 3E05 ..WR: MVI A,FDCWR ;GET FDC WRITE COMMAND
0072 0E80 MVI C,DMAWR ;GET DMA WRITE COMMAND
0074 CD 00F2' CALL DSKCOM ;CALL COMMON CODE
0077 2008 JRNZ ..RT ;IF ERRORS, RETRY
0079 3E06 MVI A,FDCRD ;ELSE, GET FDC READ COMMAND
007B 0E00 MVI C,DMAVFY ;GET DMA VERIFY COMMAND
007D CD 00F2' CALL DSKCOM ;CALL COMMON CODE
0080 C8 RJNZ ;NO ERRORS, RET A=0
0081 CD 0149' ..RT: CALL RETRY ;ERRORS, RECALIBRATE
0084 18EA JMPR ..WR ;TRY AGAIN

0086 DD7E02 FMTDSK: MOV A,PDRTRK(X) ;GET PD REQ TRACK NUMBER
0089 87 ORA A ;PD REQUEST TRACK NUMBER=0?
008A 2006 JRNZ ..NTR0 ;IF NOT, CONTINUE
008C CD 0350' CALL SELCUR ;ELSE, SELECT I/O DISK
008F CD 024F' CALL RECAL ;RECALIBRATE DRIVE
0092 3E0A ..NTR0: MVI A,MAXTRY ;GET MAX TRY COUNT
0094 32 000C' STA TRYCNT ;SET TRY COUNT
0097 CD 01D4' ..FMT: CALL SEEK ;SELECT DISK AND SEEK
0099 D3E8 MVI A,DMANR ;GET DMA WRITE COMMAND
009C 32 0011' STA IOMAC ;SET DMA COMMAND
009F DD6E08 MOV L,PDRCTC(X) ;GET PD REQ TRANSFER COUNT
00A2 DD6609 MOV H,PDRCTC+1(X)
00A5 DD5E0A MOV E,PDRDMA(X) ;GET PD REQUEST DMA ADDRESS
00A8 DD560B MOV D,PDRDMA+1(X)
00AB CD 016B' CALL DMANIT ;INITIALIZE DMA CONTROLLER
00AE' 3EO0 MVI A, FDCFMT ; GET FORMAT TRACK COMMAND
00B0' DDCB047E BIT 7, PDRSEC(X) ; DOUBLE DENSITY FLAG SET?
00B4' 2B02 JRZ ...SD ; IF NOT, CONTINUE
00B6' CBH7 SET FDCFM, A ; ELSE, SET DOUBLE DENSITY BIT
00B8' CD 0412' ...SD: CALL CMDRDY ; SEND FORMAT COMMAND TO FDC
00B8' DD7EQ01 MOV A, PDRDRV(X) ; GET PD REQUEST DRIVE NUMBER
00BE' DDCB057E BIT 7, PDRSEC+1(X) ; HEAD NUMBER ONE FLAG SET?
00CC' 2B02 JRZ ...HDO ; IF NOT, CONTINUE
00C4' CBD7 SET 2, A ; ELSE, SET HEAD ONE BIT
00C6' CD 0416' ...HDO: CALL DATOUT ; OUTPUT UNIT NUMBER TO FDC
00C9' DD7EQ04 MOV A, PDRSEC(X) ; GET PD REQUEST SECTOR (LSB)
00CC' E603 ANI 3 ; EXTRACT FORMAT SECTOR SIZE
00CE' CD 0418' CALL DATOUT ; OUTPUT FORMAT SECTOR SIZE TO FDC
00D1' DD7EQ06 MOV A, PDRSC(X) ; GET PD REQUEST SECTOR COUNT
00D4' CD 0418' CALL DATOUT ; OUTPUT SECTORS/TRACK TO FDC
00D7' DD7EQ05' MOV A, PDRSEC+1(X) ; GET PD REQUEST SECTOR (MSB)
00DA' E67F ANI 7FH ; EXTRACT FORMAT GAP LENGTH
00DC' CD 0418' CALL DATOUT ; OUTPUT FORMAT GAP LENGTH TO FDC
00DF' 3EE5 MVI A, 0E5H ; GET FORMAT FILLER BYTE
00E1' CD 0418' CALL DATOUT ; OUTPUT FORMAT FILLER BYTE TO FDC
00E4' CD 0380' CALL WTINT ; WAIT FOR INTERRUPT
00E7' 3A 0021' LDA STO ; GET STATUS REGISTER 0
00EA' E5CO ANI OC0H ; ANY ERRORS?
00EC' C8 RZ ; NO ERRORS, RET A=0
00ED' CD 0149' CALL RETRY ; ERRORS, RECALIBRATE
00F0' 18A5 JMPR ...FNT ; TRY AGAIN

00F2' 32 0010* DSKCOM: STA IORWC ; SET FDC READ/WRITE COMMAND
00F5' 79 MOV A, C ; GET DMA COMMAND
00F6' 32 0011* STA IODMAC ; SET DMA COMMAND
00F9' DD7EQ04 MOV A, PDRSEC(X) ; GET PD REQ SECTOR NUMBER
00FC' 32 0015* STA CURSEC ; SET CURRENT SECTOR
00FF' DD6EOA MOV L, PDRDMA(X) ; GET PD REQUEST DMA ADDRESS
0102' DD6EOB MOV H, PDRDMA+1(X)
0105' 22 0016* SHLD CURADR ; SET CURRENT DMA ADDRESS
0108' DD7EQ06 MOV A, PDRSC(X) ; GET PD REQ SECTOR COUNT
010B' 32 0018* STA CURSC ; SET CURRENT SECTOR COUNT
010E' CD 01D4' CALL SEEK ; SELECT DISK AND SEEK
0111' AF XRA A
0112' 32 0019* STA IOERR ; CLEAR I/O ERROR STATUS
0115' CD 0183' ...RWL: CALL SETID ; SET UP SECTOR ID INFO
0118' CD 0159' CALL SETUP ; SETUP READ/WRITE DMA
011B' CD 03E1' CALL CMDOUT ; SEND SECTOR ID INFO TO FDC
011E' CD 0380' CALL WTINT ; WAIT FOR INTERRUPT
0121' 21 0019* LXI H, IOERR ; GET I/O ERROR STATUS
0124' 3A 0021' LDA STO ; GET STATUS REGISTER 0
0127' B6 ORA M ; ADD NEW STATUS
012B' 77 MOV M, A ; UPDATE I/O ERROR STATUS
012C' CD 0453' CALL GETXLT ; GET TRANSLATION TABLE ADDRESS
012C' 2815 JRZ ...NI ; IF TRANSLATION NOT REQUIRED, CONT.
012E' 21 0015* LXI H, CURSEC ; ELSE, GET CURRENT SECTOR NUMBER
0131' 34 INR M ; INCREMENT CURRENT SECTOR
0132' CD 0448' CALL CALCSS ; CALC SECTOR SIZE
0135' EB XCHG ; SECTOR SIZE TO DE-REG
PSA Macro Assembler [C12011-0102 ]

7SK401 - TURBODOX OPERATING SYSTEM IMS FLOPPY DISK DRIVER

COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0136' 2A 0016" LHLH CURADR ;GET CURRENT DMA ADDRESS
0139' 19 DAD D ;CALC NEXT DMA ADDRESS
013A' 22 0016" SHLD CURADR ;UPDATE CURRENT DMA ADDRESS
013D' 21 0018" LXI H,CURSC ;GET CURRENT SECTOR COUNT
0140' 35 DCR M ;DECREMENT CURRENT SECTOR COUNT
0141' 20D2 JRNZ ..RWL ;IF TRANSFER NOT COMPLETE, CONTINUE
0143' 3A 0019" ..NI: LDA IOCERR ;GET I/O ERROR STATUS
0146' E6CO ANI OCOH ;EXTRACT COMPLETION STATUS
0148' C9 RET ;DONE

0149' 0E07 RETRY: MV C,ABEL ;GET BELL CHARACTER
0148' CD 0000:07 CALL CONOUT# ;OUTPUT TO CONSOLE
014E' CD 024F' CALL RECAL ;RECALIBRATE DRIVE
'0151' 21 000C" LXI H,TRYCNT ;GET RETRY COUNT
0154' 35 DCR M ;DECREMENT RETRY COUNT
0155' CO RNZ ;IF COUNT NOT EXHAUSTED, TRY AGAIN
0156' C3 0470' JMP FATAL ;CONTINUE

0159' CD 0453' SETUP: CALL GETXLT ;GET TRANSLATION TABLE ADDRESS
015C' DD6E08 MOV L,PDTRC(X) ;GET PD REQ TRANSFER COUNT
015F' DD6609 MOV H,PDTRC+1(X)
0162' 2803 JRZ ..NI ;IF NO TRANSLATION RQRD, CONTINUE
0164' CD 0448' CALL CALCSS ;ELSE, CALC SECTOR SIZE
0167' ED5B 0016" ..NI: LDED CURADR ;GET CURRENT DMA ADDRESS

016B' AF DMANIT: XRA A
016C' D388 OUT DMACTL ;RESET DMA CONTROLLER
016E' 2B DCX H ;TERMINAL COUNT-1 FOR 8257
016F' 7D MOV A,L ;GET LSB OF TERMINAL COUNT
0170' D383 OUT CH1TC ;SEND LSB OF TERMINAL COUNT
0172' 3A 0011" LDA IODMAC ;GET I/O DMA COMMAND
0175' B4 ORA H ;ADD TO MSB OF TERMINAL COUNT
0176' D383 OUT CH1TC ;SEND MSB OF TERMINAL COUNT
0178' 7B MOV A,E ;GET LSB
0179' D382 OUT CH1DMA ;OUTPUT IT TO DMA CONTROLLER
017B' 7A MOV A,D ;GET MSB
017C' D382 OUT CH1DMA ;OUTPUT IT TO DMA CONTROLLER
017E' 3E42 WI A,CH1ENA ;GET CHANNEL 1 ENABLE COMMAND
0180' D388 OUT DMACTL ;ENABLE DMA CONTROLLER
0182' C9 RET ;DONE

0183' DD7E02 SETID: MOV A,PDTRK(X) ;GET PD REQ TRACK NUMBER
0186' 32 001A" STA CYL ;SET CYLINDER
0189' 3A 0015" LDA CURSEC ;GET CURRENT SECTOR
018C' 4F MOV C,A ;SECTOR NUMBER TO C-REG
018D' CD 0453' CALL GETXLT ;GET TRANSLATION TABLE ADDRESS
0190' 2804 JRZ ..NI ;IF TRANSLATION NOT REQUIRED, CONT
0192' 0600 VI B,O ;ELSE, MAKE SECTOR DOUBLE LENGTH
0194' 09 DAD B ;INDEX INTO TRANSLATION TABLE
0195' 4F MOV C,M ;GET TRANSLATED SECTOR NUMBER
0196' 0C ..NI: INR C ;CONVERT SECTOR TO BASE 1
0197' DD4613 MOV B,SECTR(X) ;GET NUMBER OF SECTORS/TRACK
019A' CD 0461' CALL GETTCA ;GET DISK TYPE CODE ADDRESS
019D' CB56 BIT TSD,N ;TWO SIDED DISK?
PSA Macro Assembler [C12011-0T02 ]

DSK401 - TURBODOS OPERATING SYSTEM IMS FLOPPY DISK DRIVER
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

019F' 2802 JRZ ..SSD ; IF NOT, CONTINUE
01A3' CB38 SRLR B ; ELSE, CALC NUMBER OF SECTORS/SIDE
01A3' 78 . ..SSD: MOV A,B ; GET NUMBER OF SECTORS/SIDE
01A4' 32 001E' STA EOT ; SET END OF TRACK SECTOR NUMBER
01A7' B9 CMP C ; FRONT SIDE OF DISK?
01A8' 3E00 MVI A,0 ; PRESET FOR FRONT SIDE
01AA' 3005 JRNC ..FS ; IF FRONT SIDE, CONTINUE
01AC' 79 MOV A,C ; GET SECTOR NUMBER
01AD' 90 SUB B ; SUBTRACT ONE SIDES WORTH
01AE' 4F MOV C,A ; TO C-REG
01AF' 3E01 MVI A,1 ; GET HEAD #1
01B1' 32 001B' ..FS: STA HEAD ; SET HEAD NUMBER
01B4' 79 MOV A,C ; GET SECTOR NUMBER
01B5' 32 001C' STA REC ; SET RECORD NUMBER
01B8' DDE12 MOV A,SECSIZ(X) ; GET SECTOR SIZE
01BB' 32 001D' STA SIZE ; SET RECORD SIZE
01BE' 57 ORA A ; N=0?
01BF' 3E50 MVI A,128 ; PRESET DTL=128
01C1' 2802 JRZ ..NO ; IF N=0, CONTINUE
01C3' 3EFE MVI A,OFFH ; ELSE, DTL=OFFH
01C5' 32 0020' ..NO: STA DTL ; SET DATA LENGTH
01C8' CD 0469' CALL GETDST ; GET DST ADDRESS
01CB' 11 0000:08 LXI D,GAPLEN# ; GET OFFSET TO GAP LENGTH
01CE' 19 DAD D ; CALC GAP LENGTH ADDRESS
01CF' 7E MOV A,M ; GET GAP LENGTH
01D0' 32 001F' STA GPL ; SET GAP LENGTH
01D3' C9 RET ; DONE

01D4' CD 0350' SEEK: CALL SELCUR ; SELECT I/O DISK
01D7' DDE01 MOV A,PDRDRV(X) ; GET PD REQ DISK NUMBER
01DA' 3C INR A ; INCREMENT IT
01DB' 47 MOV B,A ; TO B-REG
01DC' 37 STC ; SET CARRY FLAG
01DD' 21 0000 LXI H,O ; INITIALIZE MASK
01EO' ED6A ..SL: DADC H ; GET DRIVE MASK
01E2' 10FC DJNZ ..SL CALTBL ; GET DRIVE CALIBRATED TABLE
01E4' ED5B 000D# LDED CALTBL ; GET DRIVE CALIBRATED TABLE
01E8' 2C INR L
01E9' 2D DCR L ; DRIVE 0-7?
01EA' 2006 JRNZ ..DO7 ; IF SO, CONTINUE
01EC' 7A MOV A,D ; GET CALIBRATED MAP
01ED' B4 ORA H ; SET CALIBRATED BIT
01EE' BA CMP D ; WAS IT CALORIZED?
01EF' 57 MOV D,A ; UPDATE MAP
01F0' 1804 JMPR ..UM
01F2' 7B ..DO7: MOV A,E ; GET CALIBRATED MAP
01F3' B5 ORA L ; SET CALIBRATED BIT
01F4' BB CHP E ; WAS IT CALORIZED?
01F5' 5F MOV E,A ; UPDATE MAP
01F6' ED53 000D# ..UM: SDEK CALTBL ; UPDATE TABLE
01FA' 2844 JRZ ..NRR ; IF DRIVE CALIBRATED, CONTINUE
01FC' 3A 000F# LDA FLAGS ; ELSE, GET FLAGS
01FF' CB47 BIT SLOWSR,A ; SLOW STEP RATE SET?
0201' 203A JRNZ ..RD ; IF SO, CONTINUE
CPU: 2001-12-01
PSA Macro Assembler [C12011-0102 ] 

DSK401 - TURBODOS OPERATING SYSTEM IMS FLOPPY DISK DRIVER
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0203' DDT7EO1
0205' CD 0259' MOV A,PDRDRV(X) ;GET PD REQ DISK NUMBER
0209' 201D CALL RECCMD ;SEND RECALIBRATE COMMAND
020B' DDT7EO1 mov B,A ;CONTROLLER DISK TO B-REG
020E' 47 MOV C,76 ;CYLINDER 76 TO C-REG
020F' 0E4C MVI
0211' CD 026E' CALL SEKCMD ;SEND SEEK COMMAND
0214' 2012 JRR% •• SSSI jIF ERROR, CONTINUE
0216' CD 028A' CALL RETDST ;ELSE, READ DISK ID
0219' 3A 0024* LDA RCYL ;GET PRESENT CYLINDER NUMBER
021C' FE4C MOV C,16 ;CYLINDER 76 TO C-REG
021E' 2008 CALL R!CCMJ) ;SEND R!CALIBRATE COMMAND
0221' CD 02412' CALL CMRDY ;OUTPUT COMMAND TO FDC
0224' 3E01 MVI A,SRT8S1HDUT iGET SHUGHART STEP RATE/HEAD 1
0227' CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
0232' 3E24 MVI A,HDLT ;GET HEAD LOAD TIME/NON-DMA BIT
0234' CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
0237' FB CALL ET ;ENABLE INTERRUPTS
023B' 21 000F' LXI H,FLAGS ;GET FLAGS
023B' CBC6 CALL SLOWSR,H ;SET SLOW STEP RATE BIT
023D' CD 024F' ..SRD: CALL REC ;RE-CALIBRATE DRIVE
0240' DDT7EO1 ..SRR: MOV A,PDRDRV(X) ;GET PD REQ DISK NUMBER
0243' 47 MOV B,A ;CONTROLLER DISK TO B-REG
0244' DDT7EO2 MOV A,PDRTRK(X) ;GET PD REQ TRACK NUMBER
0247' 4F MOV C,76 ;CYLINDER 76 TO C-REG
0248' CD 026E' CALL SEKCMD ;SEND SEEK COMMAND
024B' C8 CALL RECAL ;RE-CALIBRATE DRIVE
024C' C3 0470' JMP FATAL ;CONTINUE
024F' DDT7EO1 RECAL: MOV A,PDRDRV(X) ;GET PD REQ DISK NUMBER
0252' CD 0259' CALL RECCMD ;SEND RECALIBRATE COMMAND
0255' C8 CALL CMRDY ;OUTPUT COMMAND TO FDC
0258' C3 0470' JMP FATAL ;CONTINUE
0259' F5 CALL PSW ;SAVE CONTROLLER DISK
025A' 3E07 MVI A,FDCRCL ;GET FDC RECALIBRATE COMMAND
025C' CD 0412' CALL CMRDY ;OUTPUT COMMAND TO FDC
025F' F1 POP PSW ;RESTORE CONTROLLER DISK
0260' CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
0263' CD 0380' CALL WTINT ;WAIT FOR INTERRUPT
0266' 3A 0021" LDA STO ;GET STATUS REGISTER 0
0269' EEED CALL OCORH1<FDCE ;EXTRACT COMPLETION STATUS
026B' FE20 CPI 1<FDCE ;ANY ERRORS?
026C' C9 RET ;DONE
026E' C5 SEKCMD: PUSH B ;SAVE DISK/TRACK
026F' 3E0F MVI A,FDCSK ;GET FDC SEEK COMMAND
0271' CD 0412' CALL CMRDY ;OUTPUT COMMAND TO FDC
0274' C1 POP B ;RESTORE DISK/TRACK

DMK401 - TURBODOS OPERATING SYSTEM IMS FLOPPY DISK DRIVER
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0275' C5  PUSH  B  ;SAVE DISK/TRACK
0276' 78  MOV  A,B  ;GET CONTROLLER DISK
0277' CD 0418'  CALL  DATOUT  ;OUTPUT IT TO FDC
027A' C1  POP  B  ;RESTORE DISK/TRACK
027B' 79  MOV  A,C  ;GET CYLINDER NUMBER
027C' CD 0418'  CALL  DATOUT  ;OUTPUT IT TO FDC
027F' CD 0380'  CALL  WINT  ;WAIT FOR INTERRUPT
0282' 3A 0021'  LDA  STO  ;GET STATUS REGISTER 0
0285' E6ED  ANI  OCOH;FDCSE ;EXTRACT COMPLETION STATUS
0287' FE20  CPI  1<FDCSE ;ANY ERRORS?
0289' C9  RET  ;DONE

028A' CD 0303'  RETSTD:  CALL  RETRDY  ;RETURN READY STATUS
028D' B7  ORA  A  ;DRIVE READY?
028E' CB  RZ  ;IF NOT, DONE
028F' E00  MVI  C,0  ;ELSE, GET INITIAL TYPE VALUE
0291' 3A 0029'  LDA  ST3  ;GET STATUS REGISTER 3
0294' CB5F  BIT  ST3TS,A ;ONE-SIDED DISK?
0296' 2802  JRZ  ..OS  ;YES
0298' CBD1  SET  TSD,C  ;SET TWO-SIDED DISK BIT
029A' DD7E01  ..OS:  MOV  A,PDRDRV(X) ;GET PD REQ DISK NUMBER
029D' 32 0014'  STA  RIDDSK  ;SET READ ID DISK
02A0' CD 02EE'  CALL  ..FD  ;FIND DISK DENSITY
02A3' 280F  JRZ  ..DF  ;IF DENSITY FOUND, CONTINUE
02A5' C5  PUSH  B  ;ELSE, SAVE DISK TYPE CODE
02A6' DD7E01  MOV  A,PDRDRV(X) ;GET PD REQ DISK NUMBER
02A9' CD 0259'  CALL  RECCMD  ;RECALIBRATE DRIVE
02AC' C1  POP  B  ;RESTORE DISK TYPE CODE
02AD' 2032  JRNZ  ..NR  ;IF UNABLE TO RECALIBRATE, CONTINUE
02AF' CD 02EE'  CALL  ..FD  ;ELSE, ATTEMPT TO FIND DISK DENSITY
02B2' 202D  JRNZ  ..NR  ;IF DENSITY NOT FOUND, CONTINUE
02B4' B1  ..DF:  ORA  C  ;ADD SECTOR SIZE TO TYPE CODE
02B5' 4F  MOV  C,A
02B6' CB51  BIT  TSD,C  ;TWO SIDED BIT SET?
02B8' 2814  JRZ  ..FDI  ;IF NOT, CONTINUE
02BA' 21 0014'  LXI  H,RIDDSK  ;GET READ ID DISK
02BD' CD6D  SET  2,M  ;SET HEAD BIT
02BF' 3E4A  MVI  A,FDCRID1<FDCMFM  ;GET READ ID CMD (DD)
02C1' CB59  BIT  DDD,C  ;DOUBLE DENSITY BIT SET?
02C3' 2002  JRNZ  ..DD  ;IF SO, CONTINUE
02C5' CB97  RES  FDCMFM,A  ;ELSE, RESET MFM BIT
02C7' CD 02FD'  ..DD:  CALL  ..RID  ;ATTEMPT TO READ ID ON BACK SIDE
02C9' 2802  JRZ  ..FDI  ;IF READABLE, CONTINUE
02CC' CB91  RES  TSD,C  ;ELSE, RESET TWO SIDED BIT
02CE' 11 0000:09  ..FDI:  LXI  D,DSTBLK5 ;GET DISK SPEC TABLES
02DF' 79  ..SL2:  MOV  A,C  ;GET DISK TYPE CODE
02D2' 21 0000:0A  LXI  H,DTCO# ;GET OFFSET TO DISK TYPE CODE
02D5' 19  DAD  D  ;CALC DISK TYPE CODE ADDRESS
02D6' BE  CMP  M  ;DISK SPEC TABLE FOUND?
02D7' 280A  JRZ  ..DSTF  ;IF SO, CONTINUE
02D9' EB  XCHG  ;DISK SPEC TABLE ADDRESS TO HL-REG
02DA' 5E  MOV  E,M  ;GET DISK SPEC TABLE LINK POINTER
02DB' 23  INX  H
02DC' 56  MOV  D,M
02DD' 7A MOV A,D
02DE' B3 ORA E ;END OF LIST?
02DF' 20FO JRNZ SL2 ;IF NOT, CONTINUE
02E1' AF .NR: XRA A ;ELSE, SET RETURN CODE=0
02E2' C9 RET ;DONE
02E3' 13 .DSF: INX D ;ADVANCE PAST LINK POINTER
02E4' 13 INX D
02E5' D730C MOV PDRDST(X),E ;SET DISK SPEC TABLE ADDRESS
02E6' D720D MOV PDRDST+1(X),D
02E7' 3EFF MVI A,OFFH ;SET RETURN CODE=OFFH
02E8' C9 RET ;DONE
02E9' 3EOA ..FD: MVI A,FDCHID ;GET FDC READ ID COMMAND (SD)
02FA' CD 02FD' CALL .RD: ;ATTEMPT TO READ SINGLE-DENSITY
02F3' C8 RZ ;IF SINGLE-DENSITY, DONE
02F4' 3E4A MVI A,FDCHID <FDCMFM ;GET READ ID CMD (DD)
02F6' CD 02FD' CALL .RD: ;ATTEMPT TO READ DOUBLE-DENSITY
02F9' CD RNC ;IF UNABLE, DONE
02FA' C8D9 SET DDD,C ;SET DOUBLE-DENSITY DISK BIT
02FC' C9 RET ;DONE
02FD' C5 ..RDI: PUSH B ;SAVE BC
02FE' CD 0326' CALL READID ;READ DISK ID
0300' C1 POP B ;RESTORE BC
0302' C9 RET ;DONE
0303' D7601 RETRDY: MOV A,PDREDV(X) ;GET PD REQ DISK NUMBER
0306' FE04 CPI 4 ;TEST FOR VALID DRIVE NUMBER
0308' 3EOO MVI A,0 ;PRESET RETURN CODE=0
030A' DO RNC ;IF INVALID DRIVE, RETURN NOT READY
030B' DB8E IN FD CST ;GET FDC STATUS
030D' 3C INR A ;CONTROLLER PRESENT?
030E' C8 RZ ;IF NOT, DONE
030F' CD 0350' CALL SELCUR ;ELSE, SELECT REQUESTED DRIVE
0312' CD 031C' CALL .RDY: ;CHECK IF DRIVE READY
0315' C0 RNC ;IF SO, DONE
0316' 21 0001 LXI H,1 ;ELSE, DELAY ONE TICK...
0319' CD 0000:0B CALL DELAY# ;SO 765 CAN SCAN
031C' CD 036D' ..RDY: CALL SENSES ;SENSE DRIVE STATUS
031F' CB6F BIT 3RDY,A ;DRIVE READY?
0321' 3EOO MVI A,0 ;PRESET RETURN CODE=0
0323' C8 RZ ;IF DRIVE NOT READY, DONE
0324' 2F CMA ;ELSE, SET RETURN CODE=OFFH
0325' C9 RET ;DONE
0326' CD 0412' READID: CALL CMRDVY ;OUTPUT COMMAND TO FDC
0329' 3A 0014" LDA RIDDSDK ;GET READ ID DISK
032C' CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
032F' CD 0380' CALL WTINT ;WAIT FOR INTERRUPT
0332' 3A 0021" LDA STD ;GET STATUS REGISTER 0
0335' E6CO ANI OCHO ;EXTRACT COMPLETION STATUS
0337' 3A 0027" LDA RSIZE ;RETURN SECTOR SIZE
033A' C9 RET ;DONE
033B' 0000 .WORD 0 ;DELAY COMPLETE POLL ROUTINE
033B' DLPOL:
DSK401 - TURBODOS OPERATING SYSTEM IMS FLOPPY DISK DRIVER
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

033D' 0000 .WORD 0
033F' DB8C DLCPR: IN DSKCL ;GET DISK CONTROLLER STATUS
0341' CB7F BIT DSKDLCA,A ;DELAY COMPLETE (MOTORS RUNNING)?
0343' C8 RZ ;IF NOT, DONE
0344' 21 033B' LXI H,DLCPOL ;ELSE, GET POLL ROUTINE
0347' CD 0000:0C CALL UNLINK# ;UNLINK POLL ROUTINE FROM POLL LIST
034A' 21 0006* LXI H,DWTSPH ;GET DISK WAIT SEMAPHORE
034D' C3 0000:05 JMP SIGNAL# ;CONTINUE

0350' DB8C SELCUR: IN DSKCL ;GET DISK CONTROLLER STATUS
0352' 0F RRC ;EXTRACT SELECTED DRIVE
0353' E603 ANI 3
0355' 4F MOV C,A ;TO C-REG
0356' DD7E01 MOV A,PDRDRV(X) ;GET PD REQ DISK NUMBER
0359' B9 CMP C ;DRIVE ALREADY SELECTED?
035A' 2802 JRZ ...DAS ;IF SO, CONTINUE
035C' D38A OUT DSKSEL ;ELSE, SELECT CONTROLLER DISK
035E' 11 033B' ...DAS: OUT DLSKEL ;GET POLL ROUTINE
0361' CD 0000:0D CALL LNKPOL# ;CREATE POLL ROUTINE
0364' CD 033F' CALL DLCPOL ;EXECUTE POLL ROUTINE
0367' 21 0006* LXI H,DWTSPH ;GET DISK WAIT SEMAPHORE
036A' C3 0000:04 JMP WAIT# ;DISPATCH IF NECESSARY

036D' 3E04 SENSDS: MVC A,FDCSDS ;GET FDC SENSE DRIVE STATUS CMD
036F' CD 0412' CALL CMDBDT ;OUTPUT COMMAND TO FDC
0372' DD7E01 MOV A,PDRDRV(X) ;GET PD REQ DISK NUMBER
0375' CD 0418' CALL DATOUT ;OUTPUT IT TO FDC
0378' CD 041F' CALL DATALN ;GET STATUS REGISTER 3
037B' 32 0029* STA ST3 ;SAVE STATUS REGISTER 3
037E' FB EI ;ENABLE INTERRUPTS
037F' C9 RET ;DONE

0380' FB WTINT: EI ;ENABLE INTERRUPTS
0381' 21 0006* LXI H,DWTSPH ;GET DISK WAIT SEMAPHORE
0384' C3 0000:04 JMP WAIT# ;DISPATCH IF NECESSARY

0387' ED73 0000:0E DSKISR: SSPD INTSP# ;SAVE INTERRUPT STACK POINTER
038B' 31 0000:0F LXI SP,INTSTE# ;SET UP AUX STACK
038E' F5 PUSH PSW ;SAVE REGISTERS
038F' C5 PUSH B
0390' D5 PUSH D
0391' E5 PUSH H
0392' DB8E ...RQML: IN FDCST ;GET FDC STATUS
0394' CB7F BIT FDCRDY,A ;FDC READY FOR CONVERSATION?
0396' 28FA JRZ ...RQML ;IF NOT, WAIT
0398' 32 0028* STA MAINST ;SAVE MAIN STATUS REGISTER
039B' CB77 BIT FDCOUT,A ;FDC IN OUTPUT MODE?
039D' 2020 JRNZ ...RW ;IF SO, PROCESS
039F' 3E08 MVC A,FDCSIS ;GET SENSE INTERRUPT STATUS CMD
03A1' D38F OUT FDCCMD ;OUTPUT IT TO FDC DATA REGISTER
03A3' CD 041F' CALL DATAIN ;GET STATUS REGISTER 0
03A6' 4F MOV C,A ;SAVE IT IN C-REG
03A7' E6C0 ANI OCOH ;EXTRACT COMPLETION STATUS
PSA Macro Assembler [C12011-0102]

TK401 - TURBODOS OPERATING SYSTEM IMS FLOPPY DISK DRIVER

COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

03A9'  FE80   CPI  80H ; INTERRUPT STACK EMPTY?
03AB'  2829   JRZ   .X ; IF SO, DONE
03AD'  CD 041F' CALL  DATAIN ; GET PRESENT CYLINDER NUMBER
03B0'  CB69   BIT  STOE.C ; READY LINE CHANGE STATUS?
03B2'  2DE   JRZ   ...RQML ; IF SO, IGNORE
03B4'  32 0024' STA  RCTL ; ELSE, SAVE PCN
03B7'  79    MOV  A,C ; GET STATUS REGISTER 0
03B8'  32 0021' STA  STO ; SAVE IT
03BB'  3E01   MVI  A,1 ; SET INTERRUPT COMPLETION STATUS
03BD'  180F   JMPR  ...SIGC ; CONTINUE
03BF'  21 0021' ..RW:  LXI  H,RESULT ; GET RESULT TABLE
03C0'  0607   MVI  B,T ; GET LENGTH OF RESULT PHASE
03C4'  CD 041F' ..RL:  CALL  DATAIN ; GET RESULT BYTES FROM FDC
03C7'  77    MOV  M,A ; STORE IN RESULT AREA
03CB'  23    INX  H ; INCREMENT POINTER
03CF'  10F9   DJNZ  ...RL ; READ ALL SEVEN BYTES
03CB'  AF    XRA  A
03CC'  D388   OUT  DMACTL ; DISABLE DMA CONTROLLER
03CE'  21 0006' ..SIGC:  LXI  H,DWTSPH ; GET DISK WAIT SEMAPHORE
03D1'  CD 0000:05 CALL  SIGNAL# ; SIGNAL PROCESS AS READY
03D4'  18BC   JMPR  ...RQML ; FLUSH ANY REMAINING INTERRUPTS
03D6'  E1    ...X:  POP  H ; REGISTERS
03D7'  D1    POP  D
03D8'  C1    POP  B
03D9'  F1    POP  PSW
03DA'  ED7B 0000:0E LSPD  INTSP# ; RESTORE STACK POINTER
03DE'  C3 0000:10 JMP  ISRXIT# ; CONTINUE

03E1'  CD 0461'   CMDOUT: CALL  GETTCA ; GET DISK TYPE CODE ADDRESS
03E4'  3A 0010'   LDA  IORWC ; GET READ/WRITE COMMAND
03E7'  CB5E   BIT  DDD,M ; DOUBLE DENSITY DISK?
03E9'  2802   JRZ   ...SD ; IF NOT, SINGLE DENSITY
03EB'  CBF7   SET  FDCMF,M,A ; ELSE, SET DOUBLE DENSITY BIT
03ED'  CB56   ...SD:  BIT  TSD,M ; TWO-SIDED DISK?
03EF'  2802   JRZ   ...SS ; IF NOT, SINGLE SIDED
03F1'  CBFF   SET  FDCMT,M,A ; ELSE, SET MULTI-TRACK BIT
03F3'  CD 0412'   ...SS:  CALL  CMDRDY ; SEND COMMAND TO FDC
03F6'  DDTE01   MOV  A,PDRDRV(X) ; GET PD REQ DISK NUMBER
03F9'  21 001B'   LXI  H,HEAD ; GET HEAD NUMBER
03FC'  CB46   BIT  0,M ; HEAD #0?
03FF'  2802   JRZ   ...FS ; IF SO, CONTINUE
0400'  CBDD   SET  2,A ; ELSE, SET HEAD #1 BIT IN I/O DISK
0402'  CD 0418'   ...FS:  CALL  DATOUT ; OUTPUT IT TO FDC
0405'  21 001A'   LXI  H,IDINFO ; GET SECTOR ID INFO
0408'  0607   MVI  B,T ; B=LENGTH OF ID INFO
040A'  7E    ...IDL:  MOV  A,M ; GET BYTE FROM LIST
040B'  23    INX  H ; INCREMENT POINTER
040C'  CD 0418'   CALL  DATOUT ; OUTPUT BYTE TO FDC
040F'  10F9   DJNZ  ...IDL ; SEND ENTIRE LIST
0411'  C9    RET  ; DONE

0412'  CD 042A'   CMDRDY: CALL  OUTRDY ; WAIT FOR FDC READY
0415'  F3    DI   ; DISABLE INTERRUPTS
0416'  1803   JMPR  OUTCOM ; JOIN COMMON CODE
0418: CD 042A
; DATOUT: CALL OUTRDY ; WAIT FOR FDC READY

041B: 79
; OUTCOM: MOV A,C ; RESTORE OUTPUT BYTE

041C: D38F
; OUT FDCCAT ; OUTPUT BYTE TO FDC DATA REGISTER

041E: C9
; RET ; DONE

041F: DB8E
; DATAIN: IN FDCCST ; GET FDC STATUS

0421: 07
; RLC

0422: 30FB
; JRNC DATAIN ; IF NOT READY, WAIT

0424: 07
; RLC

0425: 300B
; JRNC FDCCERR ; IF WRONG DIRECTION, DIAGNOSE

0427: DB8F
; IN FDCCAT ; GET FDC DATA BYTE

0429: C9
; RET ; DONE

042A: 4F
; OUTRDY: MOV C,A ; SAVE OUTPUT BYTE

042B: DB8E
; --RW: IN FDCCST ; GET FDC STATUS

042D: 07
; RLC

042E: 30FB
; JRNC --RW ; IF NOT READY, WAIT

0430: 07
; RLC

0431: DO
; RNC ; IF DIRECTION CORRECT, DONE

0432: CD 0000:11
; FDCCERR: CALL DMS# ; SOUND BELL

0435: 87
; .ASCIS [ABEL]

0436: CD 0000:12
; CALL CONSOG ; SHIFT CONSOLE TO ERROR LINE

0439: CD 0000:11
; CALL DMS# ; DISPLAY ERROR MESSAGE

043C: 464443204572
; .ASCIS "FDC Error"

0445: C3 0445
; JMP . ; HALT

0448: 21 0080
; CALCSE: LXI H,128 ; GET 128 BYTE SECTOR LENGTH

044B: DD7E12
; MOV A,SECSEIZ(X) ; GET SECTOR SIZE

044E: 3D
; ..SL: DCR A ; DECREMENT SECTOR SIZE

044F: F8
; RM ; UNDERFLOW, DONE

0450: 29
; DAD H ; ELSE, SHIFT SECTOR SIZE LEFT

0451: 18FB
; JMP ..SL ; CONTINUE

0453: CD 0469
; GETXLT: CALL GETDST ; GET DST ADDRESS

0456: 11 0000:13
; LXI D,XLTABLE ; GET OFFSET TO TRANSLATION TABLE

0459: 19
; DAD D ; CALC TRANSLATION TABLE ADDRESS

045A: 5E
; MOV E,M ; GET TRANSLATION TABLE ADDRESS

045B: 23
; INX H

045C: 56
; MOV D,M

045D: EB
; XCHG ; TRANSLATION TABLE ADDRESS TO HL-REC

045E: 7C
; MOV A,H

045F: B5
; ORA L ; TRANSLATION REQUIRED?

0460: C9
; RET ; DONE

0461: CD 0469
; GETTCA: CALL GETDST ; GET DST ADDRESS

0464: 11 0000:14
; LXI D,TYPEDS ; GET OFFSET TO DISK TYPE CODE

0467: 19
; DAD D ; CALC DISK TYPE CODE ADDRESS

0468: C9
; RET ; DONE

0469: DD6E0C
; GETDST: MOV L,PRDST(X) ; GET PD REQUEST DST ADDRESS
046C: DD660D MOV H, PDRDST+1(X) ; DONE
046F: C9 RET
0470: ED7B 0012* FATAli: LSPD RETSP ; RESTORE STACK POINTER
0474: 3EFF MVI A, OFFH ; RETURN ERROR CODE
0476: C9 RET ; DONE
0000* ; LOC .DATA # ; LOCATE IN DATA AREA
0000* DMXSPH: .WORD 1 ; MUTUAL EXCLUSION SEMAPHORE
0002* 0002* .DMXH: .WORD ..DMXH ; SEMAPHORE P/D HEAD
0004* 0002* .DMXH: .WORD ..DMXH
0006* ; DWTSPH: .WORD 0 ; SEMAPHORE COUNT
0008* 0000 .DWTH: .WORD ..DWTH ; SEMAPHORE P/D HEAD
000A* 0008* .DWTH: .WORD ..DWTH
000C* 0000 TRICNT: .BYTE 0 ; TRY COUNT
000D* 0000 CALTBL: .WORD 0 ; DRIVE CALIBRATED TABLE
000F* 0000 FLAGS: .BYTE 0 ; FLAGS
0010* 0000 IORWC: .BYTE 0 ; I/O READ/WRITE COMMAND
0011* 0000 IODMAC: .BYTE 0 ; I/O DMA COMMAND
0012* 0000 RETSP: .WORD 0 ; ERROR RETURN STACK POINTER
0014* 0000 RIDDST: .BYTE 0 ; READ ID DISK
0015* 0000 CURSEC: .BYTE 0 ; CURRENT SECTOR NUMBER
0016* 0000 CURADR: .WORD 0 ; CURRENT DMA ADDRESS
0018* 0000 CURSEC: .BYTE 0 ; CURRENT SECTOR COUNT
0019* 0000 IOERR: .BYTE 0 ; I/O ERROR STATUS
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
; .END
PSA Macro Assembler [C12011-0102]

DSKFMT - TURBODOS OPERATING SYSTEM DRIVE SPECIFICATION TABLES
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

; COPYRIGHT (C) 1981, SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIKES
           MICHAEL D. BUSCH
; VERSION: 09/08/81

IDENT DSKFMT ;MODULE ID

INSERT DRQDATE ;DRIVER SYMBOLIC EQUIVALENCES

0002 TSD = 2 ;TWO-SIDED DISK BIT (TYPE CODE)
0003 DDD = 3 ;DOUBLE DENSITY DISK BIT (TYPE CODE)
0004 MINI = 4 ;MINI-FLOPPY DISK BIT (TYPE CODE)
0000' .LOC .PROG.# ;LOCATE IN PROGRAM AREA

1024 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED

0000' 0011' DSTBLKO WORD  +DSTL ;DISK SPEC TABLE LINK POINTER
0002'  04  BYTE  4 ;BLOCK SIZE
0003' 268  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'  010  WORD  16 ;PHYSICAL SECTORS PER TRACK
0009'  04D  WORD  77 ;PHYSICAL TRACKS PER DISK
000B'  000  WORD  0 ;NUMBER OF RESERVED TRACKS
000D'  000  WORD  0 ;TRANSLATION TABLE ADDRESS
000F'  0F  BYTE  1<DDD1< TSD13 ;DISK TYPE CODE
0010'  35  BYTE  35H ;GAP LENGTH

1024 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED (MINI)

0011'  0022' WORD  +DSTL ;DISK SPEC TABLE LINK POINTER
0013'  04  BYTE  4 ;BLOCK SIZE
0014'  134  WORD (40*(10*(1<3)))/(1<4) ;NUMBER OF BLOCKS
0016'  03  BYTE  2 ;NUMBER OF DIRECTORY BLOCKS
0017'  03  BYTE  3 ;PHYSICAL SECTOR SIZE (2^N*128)
0019'  100  WORD  10 ;PHYSICAL SECTORS PER TRACK
001B'  100  WORD  40 ;PHYSICAL TRACKS PER DISK
001D'  100  WORD  0 ;NUMBER OF RESERVED TRACKS
001F'  100  WORD  0 ;TRANSLATION TABLE ADDRESS
0021'  0F  BYTE  1<MINI1<DDD1< TSD13 ;DISK TYPE CODE
0022'  35  BYTE  35H ;GAP LENGTH

1024 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED

0023'  0022' WORD  +DSTL ;DISK SPEC TABLE LINK POINTER
0025'  04  BYTE  4 ;BLOCK SIZE
0027'  0134  WORD (77*(8*(1<3)))/(1<4) ;NUMBER OF BLOCKS
0029'  03  BYTE  3 ;NUMBER OF DIRECTORY BLOCKS
002B'  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' 00  .BYTE 1<DDD13 ; DISK TYPE CODE
0021' 35  .BYTE 35H  ; GAP LENGTH

1024 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED (MINI)
0022' 0033' .WORD  +DSTL  ; DISK SPEC TABLE LINK POINTER
0024' 04  .BYTE 4  ; BLOCK SIZE
0025' 0134 .WORD (40*(5*(1<3)))/(1<3)  ; NUMBER OF BLOCKS
0026' 02  .BYTE 2  ; NUMBER OF DIRECTORY BLOCKS
0027' 03  .BYTE 3  ; PHYSICAL SECTOR SIZE (2^N*128)
0028' 05  .WORD 5  ; PHYSICAL SECTORS PER TRACK
0029' 00  .WORD 0  ; RESERVED TRACKS
002A' 00  .WORD 0  ; TRANSLATION TABLE ADDRESS
002B' 00  .BYTE 1<MINI<DDD13 ; DISK TYPE CODE
002C' 1B  .BYTE 1BH  ; GAP LENGTH

512 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED
002D' 0024' .WORD  +DSTL  ; DISK SPEC TABLE LINK POINTER
002E' 04  .BYTE 4  ; BLOCK SIZE
002F' 077  .WORD (77*(30*(1<2)))/(1<4)  ; NUMBER OF BLOCKS
0030' 04  .BYTE 4  ; NUMBER OF DIRECTORY BLOCKS
0031' 02  .BYTE 2  ; PHYSICAL SECTOR SIZE (2^N*128)
0032' 030  .WORD 30  ; PHYSICAL SECTORS PER TRACK
0033' 077  .WORD 77  ; PHYSICAL TRACKS PER DISK
0034' 00  .WORD 0  ; RESERVED TRACKS
0035' 00  .WORD 0  ; TRANSLATION TABLE ADDRESS
0036' 00  .BYTE 1<DDD11<TS12 ; DISK TYPE CODE
0037' 1B  .BYTE 1BH  ; GAP LENGTH

512 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED
0038' 0022' .WORD  +DSTL  ; DISK SPEC TABLE LINK POINTER
0039' 04  .BYTE 4  ; BLOCK SIZE
003A' 077  .WORD (77*(15*(1<2)))/(1<4)  ; NUMBER OF BLOCKS
003B' 03  .BYTE 3  ; NUMBER OF DIRECTORY BLOCKS
003C' 02  .BYTE 2  ; PHYSICAL SECTOR SIZE (2^N*128)
003D' 015  .WORD 15  ; PHYSICAL SECTORS PER TRACK
003E' 077  .WORD 77  ; PHYSICAL TRACKS PER DISK
003F' 00  .WORD 0  ; RESERVED TRACKS
0040' 00  .WORD 0  ; TRANSLATION TABLE ADDRESS
0041' 00  .BYTE 1<DDD12 ; DISK TYPE CODE
0042' 1B  .BYTE 1BH  ; GAP LENGTH

512 BYTE SECTOR, SINGLE-DENSITY, TWO-SIDED
0043' 0025' .WORD  +DSTL  ; DISK SPEC TABLE LINK POINTER
0044' 04  .BYTE 4  ; BLOCK SIZE
0045' 077  .WORD (77*(16*(1<2)))/(1<4)  ; NUMBER OF BLOCKS
DSKfmt - TURBODOS OPERATING SYSTEM DRIVE SPECIFICATION TABLES
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

```
0027: 03 ;NUMBER OF DIRECTORY BLOCKS
0028: 02 ;PHYSICAL SECTOR SIZE (2^N*128)
0029: 0010 ;PHYSICAL SECTORS PER TRACK
002B: 004D ;PHYSICAL TRACKS PER DISK
002D: 0000 ;RESERVED TRACKS
002F: 0000 ;TRANSLATION TABLE ADDRESS
0031: 06 ;BYTE 1<TS12 ?DISK TYPE CODE
0032: 1B ;BYTE 1BH ?GAP LENGTH

512 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED

0033: 0044 ;DSTL ;DISK SPEC TABLE LINK POINTER
0035: 04 ;BLOCK SIZE
0036: 009A ;WORD (77*(52*(1<1)))/(1<4) ?NUMBER OF BLOCKS
0038: 02 ;NUMBER OF DIRECTORY BLOCKS
0039: 02 ;PHYSICAL SECTOR SIZE (2^N*128)
003A: 0008 ;PHYSICAL SECTORS PER TRACK
003C: 004D ;PHYSICAL TRACKS PER DISK
003E: 0000 ;RESERVED TRACKS
0040: 0000 ;TRANSLATION TABLE ADDRESS
0042: 02 ;BYTE 2 ?DISK TYPE CODE
0043: 1B ;BYTE 1BH ?GAP LENGTH

256 BYTE SECTOR, DOUBLE-DENSITY, TWO-SIDED

0044: 0044 ;DSTL ;DISK SPEC TABLE LINK POINTER
0046: 04 ;BLOCK SIZE
0048: 009A ;WORD (77*(52*(1<1)))/(1<4) ?NUMBER OF BLOCKS
004A: 02 ;NUMBER OF DIRECTORY BLOCKS
004B: 01 ;PHYSICAL SECTOR SIZE (2^N*128)
004C: 0052 ;PHYSICAL SECTORS PER TRACK
004E: 0077 ;PHYSICAL TRACKS PER DISK
0050: 0000 ;RESERVED TRACKS
0052: 0000 ;TRANSLATION TABLE ADDRESS
0054: 02 ;BYTE 1<TDS11 ?DISK TYPE CODE
0055: 00E1 ;BYTE 0EH ?GAP LENGTH

256 BYTE SECTOR, DOUBLE-DENSITY, ONE-SIDED

0057: 0044 ;DSTL ;DISK SPEC TABLE LINK POINTER
0059: 04 ;BLOCK SIZE
005B: 009A ;WORD (77*(26*(1<1)))/(1<4) ?NUMBER OF BLOCKS
005D: 02 ;NUMBER OF DIRECTORY BLOCKS
005E: 01 ;PHYSICAL SECTOR SIZE (2^N*128)
0060: 0026 ;PHYSICAL SECTORS PER TRACK
0062: 0077 ;PHYSICAL TRACKS PER DISK
0064: 0000 ;RESERVED TRACKS
0066: 0000 ;TRANSLATION TABLE ADDRESS
0068: 02 ;BYTE 1<TDS11 ?DISK TYPE CODE
0069: 00E1 ;BYTE 0EH ?GAP LENGTH

256 BYTE SECTOR, SINGLE-DENSITY, TWO-SIDED

006B: 0044 ;DSTL ;DISK SPEC TABLE LINK POINTER
006D: 04 ;BLOCK SIZE
006F: 009A ;WORD (77*(26*(1<1)))/(1<4) ?NUMBER OF BLOCKS
0071: 02 ;NUMBER OF DIRECTORY BLOCKS
0073: 01 ;PHYSICAL SECTOR SIZE (2^N*128)
0075: 0026 ;PHYSICAL SECTORS PER TRACK
0077: 0077 ;PHYSICAL TRACKS PER DISK
0079: 0000 ;RESERVED TRACKS
007B: 0000 ;TRANSLATION TABLE ADDRESS
007D: 02 ;BYTE 1<TDS11 ?DISK TYPE CODE
007E: 00E1 ;BYTE 0EH ?GAP LENGTH
```
256 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED

.WORD 4 ;BLOCK SIZE
.WORD (77*(30*(1<1))/(1<4) ;NUMBER OF BLOCKS
.BYTE 3 ;NUMBER OF DIRECTORY BLOCKS
.BYTE 1 ;PHYSICAL SECTOR SIZE (2^N*128)
.WORD 30 ;PHYSICAL SECTORS PER TRACK
.WORD 77 ;PHYSICAL TRACKS PER DISK
.WORD 0 ;RESERVED TRACKS
.WORD 0 ;TRANSLATION TABLE ADDRESS
.BYTE 1<TSD11 ;DISK TYPE CODE
.BYTE 0EH ;GAP LENGTH

128 BYTE SECTOR, SINGLE-DENSITY, TWO-SIDED (OLD)

.WORD 4 ;BLOCK SIZE
.WORD (77*(15*(1<1))/(1<4) ;NUMBER OF BLOCKS
.BYTE 2 ;NUMBER OF DIRECTORY BLOCKS
.BYTE 1 ;PHYSICAL SECTOR SIZE (2^N*128)
.WORD 15 ;PHYSICAL SECTORS PER TRACK
.WORD 77 ;PHYSICAL TRACKS PER DISK
.WORD 0 ;RESERVED TRACKS
.WORD 0 ;TRANSLATION TABLE ADDRESS
.BYTE 1 ;DISK TYPE CODE
.BYTE 0EH ;GAP LENGTH

128 BYTE SECTOR, SINGLE-DENSITY, TWO-SIDED

.WORD 4 ;BLOCK SIZE
.WORD (76*(52*(1<0))/(1<4) ;NUMBER OF BLOCKS
.BYTE 2 ;NUMBER OF DIRECTORY BLOCKS
.BYTE 0 ;PHYSICAL SECTOR SIZE (2^N*128)
.WORD 52 ;PHYSICAL SECTORS PER TRACK
.WORD 77 ;PHYSICAL TRACKS PER DISK
.WORD 1 ;RESERVED TRACKS
.WORD 0 ;TRANSLATION TABLE ADDRESS
.BYTE 1<TSD ;DISK TYPE CODE
.BYTE 7 ;GAP LENGTH

128 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED

.WORD 4 ;BLOCK SIZE
.WORD (77*(52*(1<0))/(1<4) ;NUMBER OF BLOCKS
.BYTE 2 ;NUMBER OF DIRECTORY BLOCKS
.BYTE 0 ;PHYSICAL SECTOR SIZE (2^N*128)
.WORD 52 ;PHYSICAL SECTORS PER TRACK
.WORD 77 ;PHYSICAL TRACKS PER DISK
.WORD 0 ;RESERVED TRACKS
.WORD 0 ;TRANSLATION TABLE ADDRESS
.BYTE 1<TSD ;DISK TYPE CODE
.BYTE 7 ;GAP LENGTH

128 BYTE SECTOR, SINGLE-DENSITY, ONE-SIDED
PSA Macro Assembler [C12011-0102 ]

DSKfmt - TURBODOS OPERATING SYSTEM DRIVE SPECIFICATION TABLES
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

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 XLTB1 =: --DSTB ;TRANSLATION TABLE ADDRESS OFFSET
0051' 0055' ;
005F .WORD TRTBL ;TRANSLATION TABLE ADDRESS
0000 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
005F' 02080E140107 .BYTE 2,8,14,20,1,7,13,19,25
0067' 050B11170309 .BYTE 5,11,17,23,3,9,15,21

; END
TC442 — TURBODOS OPERATING SYSTEM IMS REAL TIME CLOCK Routines

COPYRIGHT (C) 1981 BY SOFTWARE 2000, INC.

; COPYRIGHT (C) 1981 BY SOFTWARE 2000, INC.
; AUTHORS: RONALD E. RAIKES
; MICHAEL D. BUSCH
; VERSION: 09/08/81
; IDENT RTC442 ;MODULE ID
; INSERT DRIVER SYMBOlIC EQUVALENCES
; IOBASE = 10H ;SERIAL/PARALLEL I/O PORT BASE
0010
0016
0017
0018
0019
0001
00B6
0000
0000
0000
0000
0001
0002
0004
0007
000A
000D
000F
0011
0014
0015
0017
0018
001A
001D
001F
0022
0024
0027
0028
002A
002B
0031

RTCENA = 1 ;REAL TIME CLOCK ENABLE BIT
T2CMD = 0B6H ;TIMER 2 COMMAND
; LOC .DATA. ;LOCATE IN DATA AREA
T2CNT: .BYTE 0 ;TICK COUNTER
; LOC .PROG. ;LOCATE IN PROGRAM AREA
; WORD NITLEN+2 ;INITIALIZATION CODE LENGTH
0002
0004
0007
000A
000D
000F
0011
0014
0015
0017
0018
001A
001D
001F
0022
0024
0027
002A
3EC3
32 0008
21 002A
22 0009
3EB6
D317
21 0000:04
7D
D316
7C
D316
21 0000:05
CBCE
3A 0000:05
D318
21 0002
C3 0000:06
ED73 0000:07
RTCISR: SSPD INTSP# ;SAVE STACK POINTER
31 0000:08 LXI SP,INTSTK# ;SET UP AUX STACK POINTER
F5 PUSH PSW ;SAVE REGISTERS

RTCNIAT::MVI A,JMP ;INIT RTC INTERRUPT VECTOR ADDR
STA 1*8
LXI H,RTCISR
SHLD (1*8)+1
MVI A,T2CMD ;GET TIMER 2 COMMAND
OUT TIMCTL ;SELECT TIMER 2
LXI H,RTCNT# ;GET RTC COUNTER VALUE
MOV A,L ;GET LSB OF TIMER VALUE
OUT TIM2 ;OUTPUT IT TO TIMER 2 DATA REGISTER
MOV A,H ;GET MSB OF TIMER VALUE
OUT TIM2 ;OUTPUT IT TO TIMER 2 DATA REGISTER
LXI H,INTMSK# ;GET INTERRUPT MASK
SET 1,M ;SET RTC INTERRUPT ENABLE BIT
OUT INTMSK ;GET INTERRUPT MASK
OUT SINTE ;ENABLE RTC INTERRUPT MASK
LXI H,RTCNIAT ;GET INITIALIZATION CODE ADDRESS
JMP DEALOC# ;DE-ALLOCATE INITIALIZATION CODE
NITLEN = .RTCNIAT ;INITIALIZATION CODE LENGTH
RTCNIAT: :MVI A,JMP ;INIT RTC INTERRUPT VECTOR ADDR
RTCISR: SSPD INTSP# ;SAVE STACK POINTER
LXI SP,INTSTK# ;SET UP AUX STACK POINTER
PUSH PSW ;SAVE REGISTERS
PSA Macro Assembler [C12011-0102]

RTC442 - TURBODOS OPERATING SYSTEM IMS REAL TIME CLOCK ROUTINES
COPYRIGHT (C) 1981 BY SOFTWARE 2000, INC.

0032'  C5      PUSH    B
0033'  D5      PUSH    D
0034'  E5      PUSH    H
0035'  D319    OUT     T2RES ;RESET RTC INTERRUPT
0036'  21 0000   LXI     H,TICCNT ;GET TICK COUNTER
0037'  34      INR     M ;INCREMENT TICK COUNTER
0038'  7E      MOV     A,M ;GET TICK COUNT
0039'  01 0000:09 LXI     B,TICSEC# ;GET NUMBER OF TICKS PER SECOND
003A'  3B      CMP     C ;SECONDS COUNT REACHED?
003B'  3B05    JRC     ..NSEC ;IF NOT, CONTINUE
003C'  3600    MVI     M,0 ;ELSE, RESET TICK COUNTER
003D'  CD 0000:0A CALL    RTCSEC# ;SERVICE REAL TIME CLOCK MANAGER
003E'  CD 0000:0B ..NSEC: CALL    DLYTIC# ;SERVICE DISPATCHER DELAY MANAGER
0040'  E1      POP     H ;RESTORE REGISTERS
0041'  D1      POP     D
0042'  C1      POP     B
0043'  F1      POP     PSW
0044'  ED7B 0000:07 LSPD    INTSP# ;RESTORE STACK POINTER
0045'  C3 0000:0C JMP     ISRXT# ;CONTINUE

;END
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

AUTHORS: RONALD E. RAIXES
MICHAEL D. BUSCH

VERSION: 07/21/81

IDENT BPD401 MODULE ID

INSERT EQUATE O/S SYMBOLIC EQUIVALENCES

RAM =: TBUF WORKING STORAGE ADDRESS
RAMLEN = 64 WORKING STORAGE LENGTH

CH1DMA = 82H CHANNEL 1 DMA REGISTER (FDC)
CH1TC = 83H CHANNEL 1 TERMINAL COUNT (FDC)
DMACTL = 88H DMA COMMAND AND STATUS REGISTERS
DSKSEL = 8AH DISK SELECT PORT
DSKCTL = 8CH STATUS AND INT MASK (BOARD)
FDCST = 8EH DISK CONTROLLER STATUS (uPD-76S)
FDCDAT = 8FH DISK CONTROLLER DATA (uPD-76S)

CH1ENA = 42H DMA CHANNEL 1 ENABLE COMMAND
DMAVFY = 00H DMA VERIFY COMMAND
DMARD = 40H DMA READ COMMAND
DMAWR = 80H DMA WRITE COMMAND

FDCSFY = 03H FDC SPECIFY COMMAND
FDCSDS = 04H FDC SENSE DRIVE STATUS COMMAND
FDCRCL = 07H FDC RECALIBRATE COMMAND
FDCSIS = 08H FDC SENSE INTERRUPT STATUS COMMAND
FDCRID = 0AH FDC READ ID COMMAND
FDCSK = 0FH FDC SEEK COMMAND
FDCWR = 85H FDC WRITE COMMAND
FDCRD = 86H FDC READ COMMAND

DSKENI = 0 DISK CONTROLLER ENABLE INTERRUPTS
DSKDLC = 7 DISK CONTROLLER DELAY COMPLETE

FDCMF = 6 FDC DOUBLE-DENSITY BIT
FDCBSY = 4 FDC BUSY STATUS
FDCSE = 5 FDC SEEK END
FDCOUT = 6 FDC OUTPUT MODE
FDCRDY = 7 FDC READY FOR DATA

SRT5 = (16-4)<4 5 INCH FDD STEP RATE (4 MS-MINI)
SRT8S = (16-6)<4 8 INCH FDD STEP RATE (6 MS-SHUGART)
SRT8R = (16-3)<4 8 INCH FDD STEP RATE (3 MS-REMEX)
SRT8P = (16-1)<4 8 INCH FDD STEP RATE (1 MS-PERSCI)

H1T = 18*2 FDD HEAD LOAD TIME (36 MS)
BPD401 – TURBODOS OPERATING SYSTEM BOOT PROM DRIVER FOR IMS 401
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

0001 HUT = 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 ST1MW = 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 ST3TD = 4 ; STATUS REGISTER 3 TRACK 0
0005 ST3BDY = 5 ; STATUS REGISTER 3 READY
0006 ST3WP = 6 ; STATUS REGISTER 3 WRITE PROTECTED
               ;
0002 TSD = 2 ; TWO-SIDED DISK BIT (TYPE CODE)
0003 DDD = 3 ; DOUBLE DENSITY DISK BIT (TYPE CODE)
               ;
000A MAXTRY = 10 ; MAX TRY COUNT
               ;
00C0 .LOC RAM+RAMLEN ; LOCATE IN WORKING STORAGE AREA
               ;
00C0 IODSK: .BLKB 1 ; DISK NUMBER
00C1 IOTRK: .BLKW 1 ; TRACK NUMBER
00C3 IOSSEC: .BLKW 1 ; SECTOR NUMBER
00C5 IODMA: .BLKW 1 ; DMA ADDRESS
00C7 ST3BEG: .BLKB 1 ; STATUS REGISTER 3
00C8 TRYCNT: .BLKB 1 ; TRY COUNT
               ;
00C9 DSKNFO: ; DISK TYPE INFORMATION
00C9 BLKSIZE: .BLKB 1 ; BLOCK SIZE
00CA NMQLKS: .BLKW 1 ; NUMBER OF BLOCKS
00CC NMBDIR: .BLKB 1 ; NUMBER OF DIRECTORY BLOCKS
00CD SECSIZ: .BLKW 1 ; PHYSICAL SECTOR SIZE (2^N*128)
00CE SECTRK: .BLKB 1 ; PHYSICAL SECTORS PER TRACK
00D0 TRKDSK: .BLKW 1 ; PHYSICAL TRACKS PER DISK
00D2 RESTRK: .BLKW 1 ; NUMBER OF RESERVED TRACKS
00D4 XLTBL: .BLKB 1 ; TRANSLATION TABLE ADDRESS
00D6 TYPCOD: .BLKB 1 ; DISK TYPE CODE
00D7 GAPLEN: .BLKB 1 ; GAP LENGTH
000F DNFOL = -.DSKINFO ; DISK INFO LENGTH
               ;
0000 INIT: ; LOCATE IN PROGRAM AREA
0000 3E03 MVI A, FDCSFY ; GET FDC SPECIFY COMMAND
0002 01B3 CALL DATOUT ; OUTPUT FDC SPECIFIT COMMAND
0005 3EA1 MVI A, SRT8SHUT ; GET STEP RATE/HEAD UNLD TIMF
0007 CD 01B3 CALL DATOUT ; OUTPUT STEP RATE/HEAD UNLD TIMF
000A 3E24 MVI A, HLT ; GET HEAD LOAD TIME/NON-DMA BIT
000C 01B3 CALL DATOUT ; OUTPUT HEAD LOAD TIME/NON-DMA BIT
000F 21 0100 LXT H, TPA ; GET LOAD BASE ADDRESS
**PD401 - TURBODOS OPERATING SYSTEM BOOT PROM DRIVER FOR IMS 401**

*Copyright (C) 1981, SOFTWARE 2000, INC.*

```
0012 C9
0013 FE04
0015 3061
0017 32 00C0
001A 4F
001B DB8C
001D 0F
001E 6E03
0020 B9
0021 2803
0023 79
0024 D38A
0026 01 0014
0029 10FE
002B 0D
002C 20FB
002E 3E04
0030 CD 01B3
0032 3A 00C0
0036 CD 01B3
0039 CD 01A7
003C CB6F
003E 2838
0040 32 00C7
0043 CD 0165
0046 2030
0048 0E00
004A 21 00C7
004D CB5E
004F 2802
0051 CB01
0053 3E0A
0055 CD 0089
0058 2809
005A 3E4A
005C CD 0089
005F 2017
0061 CB99
0063 B1
0064 4F
0065 11 0000:04
0068 79
0069 21 0000:05
006C 19
006D BE
006E EB
006F 2809
0071 5E
0072 23
0073 56
0074 7A
0075 B3
0076 20F0
```

**PSA Macro Assembler [C12011-0102 ]**

Page 3
0078' AF ...NR: XRA A ;SET RETURN CODE=0
0079' C9 RET ;DONE
007A' 23 ...DSTF: INX H ;ADVANCE PAST LINK POINTER
007B' 23 INX H
007C' E5 PUSH H ;SAVE DST ADDRESS
007D' 1 00C9 LXT D,DSKINFO ;GET DISK INFO WORK AREA
0080' 01 000F LXT B,DFNOL ;GET DISK INFO LENGTH
0083' EDB0 LDTR ;COPY DST INTO WORK AREA
0085' E1 POP H ;RESTORE DST ADDRESS
0086' 3EFF MV1 A,OFFH ;SET RETURN CODE=OFFH
0088' C9 RET ;DONE
0089' C5 ...RID: PUSH B ;SAVE BC-REG
008A' CD 01B3' CALL DATOUT ;OUTPUT COMMAND TO FDC
008D' 3A 00C0 LDA IODSK ;GET DISK NUMBER
0090' C0 01B3' CALL DATOUT ;OUTPUT IT TO FDC
0093' CD 0170' CALL WINT ;WAIT FOR INTERRUPT
0096' 78 MOV A,B ;RETURN SECTOR SIZE
0097' C1 POP B ;RESTORE BC-REG
0098' C9 RET ;DONE
0099' ED43 00C1 READ:: SBBCD IOTRK ;SAVE TRACK NUMBER
009D' ED53 00C3 SDED IODEC ;SAVE SECTOR NUMBER
00A1' 22 00C5 SHLD IODMA ;SAVE DMA ADDRESS
00A4' 21 00C8 LXT H,TRYCNT ;GET TRY COUNT
00A7' 360A MV1 M,MAXTRY ;INITIALIZE TRY COUNT
00A9' CD 0152' ...RR: CALL SEEK ;SEEK TO REQUESTED TRACK
00AC' C2 0143' JNZ ...ERR ;IF ERRORS, CONTINUE
00AF' AF XRA A
00B0' D388 OUT DMACtl ;RESET DMA CONTROLLER
00B2' 21 0080 LXT H,128 ;GET SECTOR SIZE=0 SECTOR LENGTH
00B5' 3A 00CD LDA SECSIZ ;GET PHYSICAL SECTOR SIZE
00B8' B7 ORA A ;PHYSICAL SECTOR SIZE=0?
00BB' 2980 JNZ ...NO1 ;IF SO, CONTINUE
00BC' 3D DCR A ;SECTOR SIZE TIMES
00BD' 20FC JRNZ ...SL
00BF' 2B NO1: DCX H ;COUNT -1 FOR 8257
00C0' 7D MOV A,L ;GET LSB OF TERMINAL COUNT
00C1' D383 OUT CH1TC ;OUTPUT LSB OF TERMINAL COUNT
00C3' 7C MOV A,H ;GET MSB OF TERMINAL COUNT
00C4' F640 ORI DMARD ;ADD DMA READ COMMAND
00C6' D383 OUT CH1TC ;OUTPUT MSB OF TERMINAL COUNT
00C8' 2A 00C5 LHLDD IODMA ;GET DMA ADDRESS
00CB' 7D MOV A,L ;GET LSB OF DMA ADDRESS
00CC' D382 OUT CHI1DMA ;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 MV1 A,CH1ENA ;GET CHANNEL 1 ENABLE COMMAND
00D3' D388 OUT DMACtl ;ENABLE DMA CONTROLLER
00D5' 3E86 MV1 A,FDLCD ;GET FDC READ COMMAND
00D7' 21 00D6 LXT H,TYPECOD ;GET DISK TYPE CODE
00DA' C85E BIT DDD,M ;SINGLE DENSITY DISK?
00DC' 2B02 JRZ ...SD ;IF SO, CONTINUE
00DE' CBF7 SET FDCFMF,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  LHI LD XLTLB ;GET TRANSLATION TABLE ADDRESS
00EA'  7C  MOV A,H ;SECTOR TRANSLATION REQUIRED?
00EB'  B5  ORA L ;IF NOT, CONTINUE
00EC'  2804  JRZ ..NI ;ELSE, MAKE SECTOR DOUBLE LENGTH
00EE'  1600  MVI D,0 ;INDEX INTO TRANSLATION TABLE
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  LIX H,TTPCOD ;GET DISK TYPE CODE ADDRESS
00F9'  CB56  BIT TSD,M ;TWO SIDED DISK?
00FB'  2802  JRZ ..SSD ;IF NOT, CONTINUE
00FD'  CB3F  SRL A ;ELSE, CALC NUMBER OF SECTORS/SIDE
00FF'  57  ..SSD: MOV DB, A ;SAVE NUMBER OF SECTORS/SIDE
0100'  0600  MVI B,0 ;RESET 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'  CB07  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,C ;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
0135'  B7  ORA A ;SECTOR SIZE=0?
0136'  3EB0  MVI A,128 ;RESET DTL=128
0138'  2802  JRZ ..NO ;IF SECTOR SIZE=0, CONTINUE
013A'  3EFF  MVI A,0FFH ;ELSE, DTL=OFFH
013C'  CD 01B3'  ..NO: CALL DATOUT ;OUTPUT DTL
013F'  CD 0170'  CALL WTINT ;WAIT FOR INTERRUPT
0142'  C8  RZ ;IF NO ERRORS, DONE
0143'  CD 0165'  ..ERR: CALL RECAL ;RECALIBRATE DRIVE
0146'  2007  JRNZ ..X ;IF ERRORS, CONTINUE
0148'  21 00C8  LXI H,TRYCNT ;ELSE, GET TRY COUNT
014B' 35 DCR M
014C' C2 00 A9' JNZ ..RR
014F' 3EFF ..X: MVI A,OFFH
0151' C9 RET ;DONE

0152' 3E0F SEEK: MVI A,FDCSK ;SET FDC SEEK COMMAND
0154' CD 01B3' CALL DATOUT ;OUTPUT FDC SEEK COMMAND
0157' 3A 00C0 LDA IODSK ;GET DISK NUMBER
015A' CD 01B3' CALL DATOUT ;OUTPUT DISK NUMBER
015D' 3A 00C1 LDA IOTR ;GET TRACK NUMBER
0160' CD 01B3' CALL DATOUT ;OUTPUT TRACK NUMBER
0163' 180B JMPR WTINT ;WAIT FOR INTERRUPT

0165' 3E07 RECAL: MVI A,FDCRCL ;SET FDC RECALIBRATE COMMAND
0167' CD 01B3' CALL DATOUT ;OUTPUT FDC RECALIBRATE COMMAND
016A' 3A 00C0 LDA IODSK ;GET DISK NUMBER
016D' CD 01B3' CALL DATOUT ;OUTPUT DISK NUMBER

0170' DB8C WTINT: IN DSKCTL ;GET DISK CONTROLLER STATUS
0172' OF RRC ;TEST FOR FDC INTERRUPT
0173' 30FB JRNC WTINT ;IF NO INTERRUPT, WAIT
0175' D8BE ..RQML: IN FDST ;GET FDC STATUS
0177' 07 RLC ;FDC READY TO COMMUNICATE?
0178' 30FB JRNC ..RQML ;IF NOT, WAIT
017A' 07 RLC ;TEST FDC DIRECTION
017B' 3818 JRC ..RW ;IF FDC OUTPUT AVAILABLE, PROCESS
017D' 3E08 MVI A,FDCSIS ;GET SENSE INTERRUPT STATUS CMD
017F' D38F OUT FDCT ;OUTPUT BYTE TO FDC DATA REGISTER
0181' CD 01A7' CALL DATAIN ;GET STATUS REGISTER 0
0184' 4F MOV C,A ;SAVE IT IN C-REG
0185' E6C0 ANI 0COH ;EXTRACT COMPLETION STATUS
0187' FE80 CPI 80H ;INTERRUPT STACK EMPTY?
0189' 2818 RJC ..X: ;IF SO, DONE
018B' CD 01A7' CALL DATAIN ;GET PRESENT CYLINDER NUMBER
018E' CB69 BIT STOSE,C ;READY LINE CHANGE STATE?
0190' 28E3 JRZ ..RQML ;IF SO, IGNORE
0192' 51 MOV D,C ;GET STATUS REGISTER 0 IN D-REG
0193' 18E0 JMPR ..RQML ;FLUSH ANY REMAINING INTERRUPTS
0195' CD 01A7' ..RW: CALL DATAIN ;GET STATUS REGISTER 0
0198' 57 MOV D,A ;TO D-REG
0199' 0606 MVI B,S ;B=LENGTH OF REMAINING RESULT PHASE
019B' CD 01A7' ..RL: CALL DATAIN ;GET RESULT BYTE FROM FDC
019E' 10FB DJNZ ..RL ;READ ALL SEVEN BYTES
01A0' 47 MOV B,A ;SAVE SECTOR SIZE IN B-REG
01A1' 18D2 JMPR ..RQML ;FLUSH ANY REMAINING INTERRUPTS
01A3' 7A ..X: MOV A,D ;GET STATUS REGISTER 0
01A4' E6C0 ANI 0COH ;EXTRACT COMPLETION STATUS
01A6' C9 RET ;DONE

01A7' DB8E DATAIN: IN FDCT ;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
BPD401 - TURBODOS OPERATING SYSTEM BOOT PROM DRIVER FOR IMS 401
COPYRIGHT (C) 1981, SOFTWARE 2000, INC.

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 BYTE
01BE' D38F        OUT  FDCDAT ;OUTPUT BYTE TO FDC DATA REGISTER
01C0' C9          RET  ;DONE

01C1' AF          XFER:: XRA  A ;MAKE DEFAULT BUFFER EMPTY
01C2' 32 0080      STA  TBUF ;TRANSFER TO O/S LOADER
01C5' C3 0100      JMP  TPA ;TRANSFER TO O/S LOADER

.END