TEKTRONIX®

8002
μPROCESSOR LAB

SUPPLEMENT TO
SYSTEM USER'S MANUAL

This manual supports Tektronix Disc Operating System (TEKDOS) Version 2.

Tektronix, inc.
P.O. Box 500
Beaverton, Oregon 97077
070-2463-00

Serial Number _____

First Printing JAN 1978

WARRANTY

The 8001/8002 μProcessor Lab System (including options) is warranted against defective materials and workmanship under normal use and service for a period of 90 days from date of initial shipment. CRTs found to be defective within 12 months from the date of shipment will be exchanged at no charge (this does not include installation).

On site warranty repairs is provided during normal working hours (for the 90-day period). Travel to the site is confined to those areas in which Tektronix states it has service facilities available for this product.

Tektronix shall be under no obligation to furnish warranty service if:

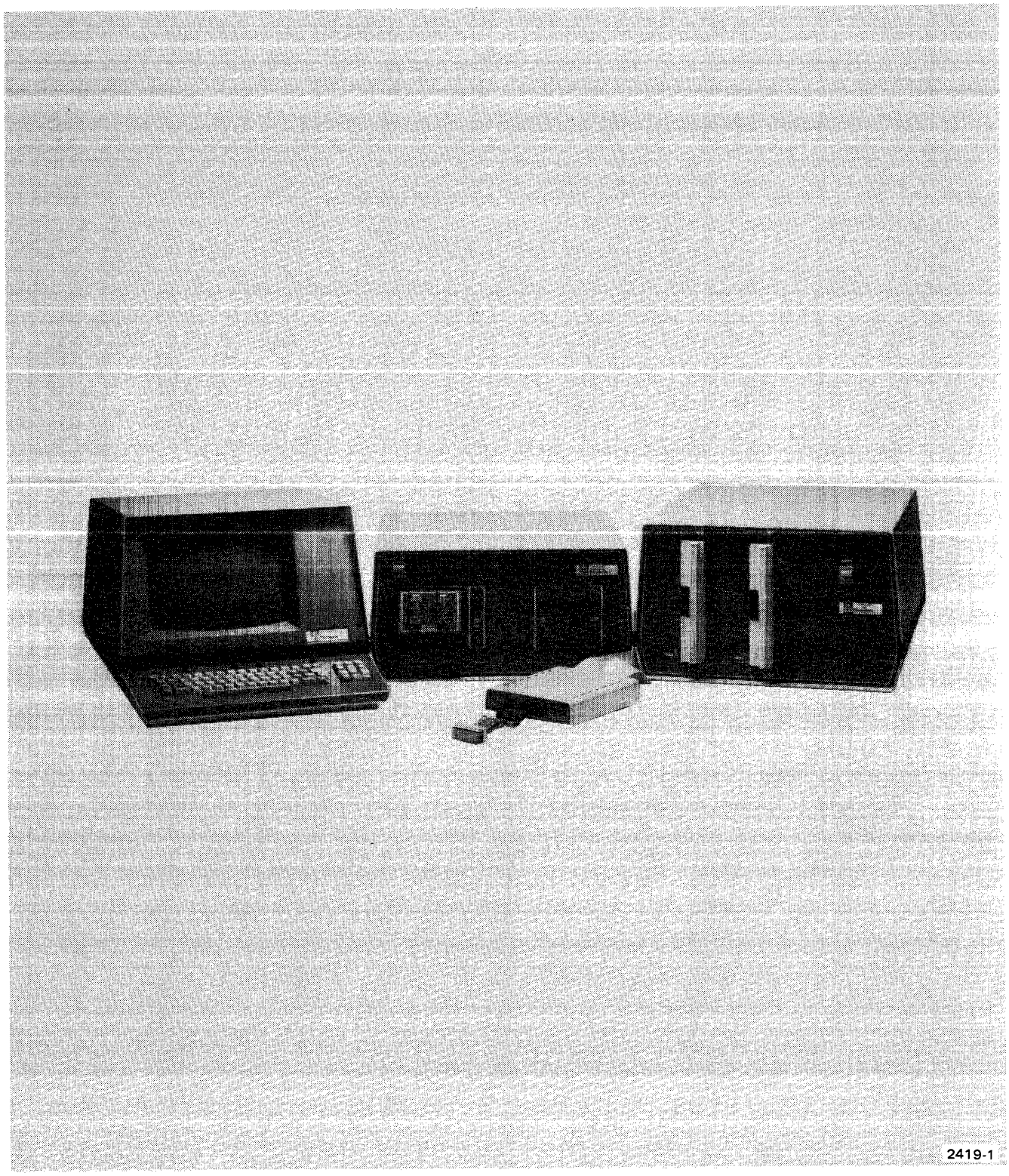
a. Attempts to install, repair, or service the equipment are made by personnel other than Tektronix service representatives.

b. Modifications are made to the hardware or software by personnel other than Tektronix service representatives.

c. Damage results from connecting the 8001/8002 μProcessor Lab System to incompatible equipment.

Specifications and price change privileges reserved.

CONTENTS


2419-1

Section 2
Change Information

BECOMING FAMILIAR WITH THE SYSTEM

Section 2 of the 8002 System User's Manual describing the macroscopic features of the 8002 Lab is not affected by the Version 2 update to the Tektronix Disc Operating System.

Section 3
Change Information

COMMAND CONVENTIONS

Section 3 of the 8002 System User's Manual describing command conventions is not affected by the Version 2 update to the Tektronix Disc Operating System.

Section 4
Change Information

TEKTRONIX DISC OPERATING SYSTEM

Section 4 of the 8002 System User's Manual describing the Tektronix Disc Operating System (TEKDOS) is not affected by the Version 2 update to TEKDOS.

Section 5
Change Information

TEXT EDITOR

Section 5 of the 8002 System User's Manual describing the Text Editor is not affected by the Version 2 update to TEKDOS.

Section 6
Change Information

ASSEMBLING AND LINKING

Section 6 of the 8002 System User's Manual describing the Assembler and Linker commands is not affected by the Version 2 update to TEKDOS.

Section 7
Change Information

EMULATOR ENVIRONMENT

This section supplements the description of the operating environment of the emulator processor contained in Section 7 of the 8002 System User's Manual. The changes and additions specific to Version 2 of TEKDOS include user program loading and storing and memory control.

TABLE OF CONTENTS

LOADING AND STORING

The commands in this section are used to move object code between program memory and flexible disc storage. The changes and additions specific to Version 2 of TEKDOS include the implementation of a second hexadecimal format; TEKDOS now supports both Intel and Tektronix hexadecimal formats, as explained in the following commands.

Tektronix Hexadecimal Format

Object code from the Assembler is stored on a flexible disc in either Intel hexadecimal format or in Tektronix Hexadecimal Format (TEKHEX). Files in Intel format are serviced by the RVHEX and WVHEX commands; Intel hexadecimal format is described in the discussion of these two commands. The RHEX and WHEX commands require TEKHEX format.

Additionally, data transferred by the TEKDOS command, SEND, between the 8002 μProcessor Lab and the 8001 μProcessor Lab is automatically formatted in Tektronix Hexadecimal Format.

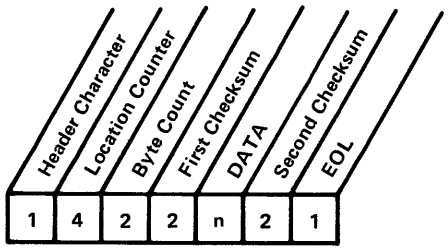
A Tektronix Hexadecimal Formatted file consists of message blocks. Types of message blocks include data blocks, terminating blocks, and abort blocks.

GENERAL FEATURES

The following features apply to all message blocks:

- Each message block has a slash (/) as a header character.
- Every message block is terminated by a carriage return.
- A message block is limited to 72 characters. As a result, a full-length data block contains 30 data bytes (that is, 60 hexadecimal characters).
- All characters must belong to the printable ASCII character set. Any characters encountered during decoding that are not in the ASCII character set are disregarded as random interline characters or line hits.
- All characters except header characters and error information must be ASCII-encoded hexadecimal integers.
- A checksum is calculated as the sum of the four-bit hexadecimal digits, modulo 256.

DATA BLOCK FORMAT DESCRIPTION



All entries in the data block are ASCII-encoded hexadecimal digits, except the **Header Character** and the **EOL**. The following list describes each entry in the data block:

The **Header Character** is always a slash (/).

The **Location Counter** consists of four hexadecimal digits and gives the starting location of the block in program memory.

The **Byte Count** is the number of data bytes in this block. The byte count is a two-digit hexadecimal number.

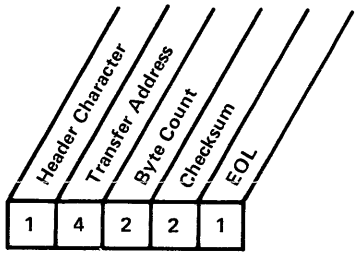
The **First Checksum** is the eight-bit sum of the four-bit hexadecimal values of the six digits that make up the location counter and the byte count. The checksum is a two-digit hexadecimal number.

DATA consists of **n** data bytes, each composed of two hexadecimal digits. The maximum number of data bytes per block is 30. Thus, the maximum number of hexadecimal digits in the data section is 60.

The **Second Checksum** is the eight-bit sum, modulo 256, of the four-bit hexadecimal values of the digits that make up the **n** bytes of data. The checksum is a two-digit hexadecimal number.

EOL is a carriage return.

TERMINATING BLOCK FORMAT DESCRIPTION



2463-6

All entries in the terminating block are ASCII-encoded hexadecimal digits, except the **Header Character** and the **EOL**. The following list describes each entry in the terminating block:

The **Header Character** is always a slash (/).

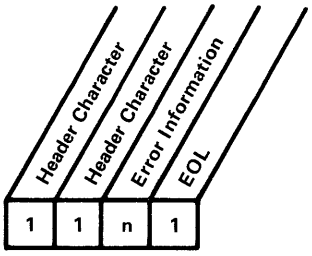
The **Transfer Address** consists of four hexadecimal digits.

The **Byte Count** is set to zero, indicating a terminating block.

The **Checksum** is the eight-bit sum of the four-bit hexadecimal values of the six digits that make up the transfer address and the byte count. The checksum is a two-digit hexadecimal number.

EOL is a carriage return.

ABORT BLOCK FORMAT DESCRIPTION



2463-7

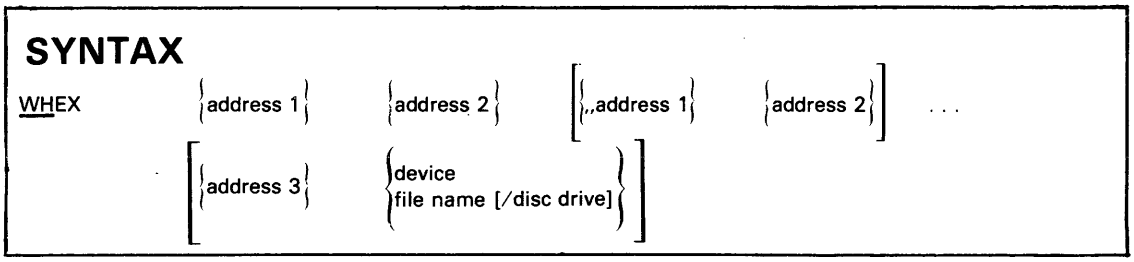
The following list describes each entry in the abort block:

The two **Header Character** entries are slashes (/ /).

Error Information is an arbitrary string of ASCII characters.

EOL is a carriage return.

SYNTAX



In addition to the description of the WHEX command in the System User's Manual, the following two changes in command line format apply:

1. No more than 13 pairs of start and stop addresses may be specified in one command line.
2. The entire command line must be entered as a single line on the system console.

SYNTAX

RVHEX [/bias amount] [device / filename [/disc drive]]

PURPOSE

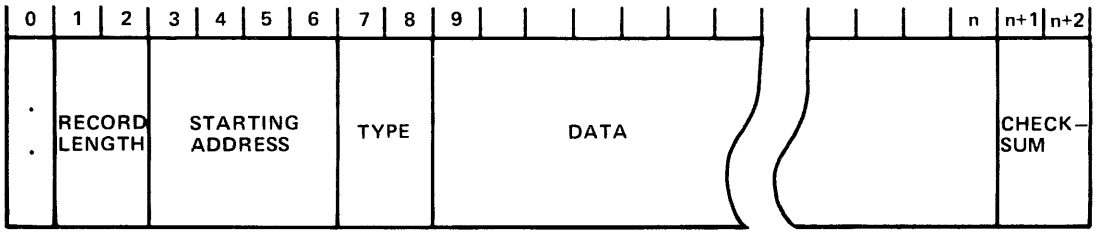
The RVHEX command reads and converts Intel format hexadecimal code from the specified device or disc file into binary code, and loads the binary code into program memory.

EXPLANATION

The **bias amount** parameter is the positive or negative hexadecimal amount by which the initial load address is to be altered. The default value for the bias amount is 0.

The default input **device** is the paper tape reader (PPTR).

The INTEL load module format is as follows:



2463-3

character 0 — The **header character**; a colon.

1 & 2 — The number of **data bytes**.

3 through 6 — The program memory **address** of the first data byte in this record.

7 & 8 — The record **type** where 00 = normal data record, and 01 = last record of a file.

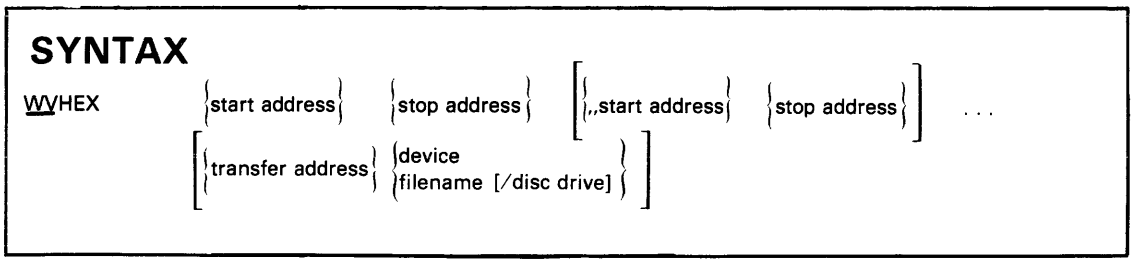
9 through n — **Data**. Each pair of hexadecimal digits represents one data byte.

(n+1) & (n+2) — The **checksum**, the 2's complement of the sum of the eight-bit bytes from columns 1 through n.

A record length of 0 indicates the last record of a file.

This is the load module format required by the system loader of previous TEKDOS versions.

SYNTAX



PURPOSE

The WVHEX command converts binary code in program memory to ASCII characters representing the binary code. It then writes these ASCII characters in INTEL format to the specified output device or file.

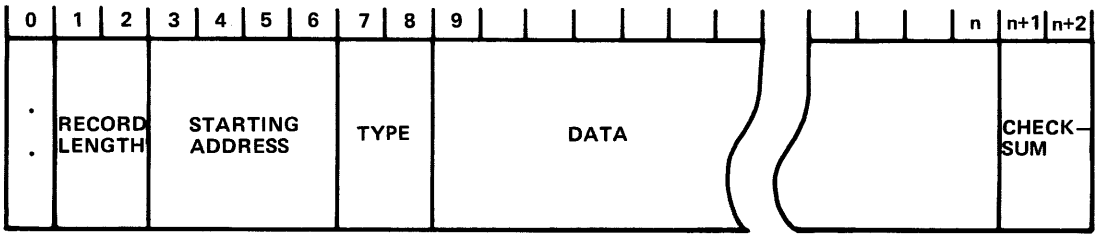
EXPLANATION

The **start and stop addresses** are hexadecimal program memory addresses; they are the upper and lower bounds of the binary code to be converted and written. There can be no more than 13 pairs of start and stop addresses. Each pair must be separated by two commas. The entire command must be entered as a single line on the system console.

The **transfer address** denotes the first executable instruction of the object program, and is included only as a reference for the user. It has no effect on the object program, the Assembler, or the WVHEX command.

The default output **device** is CONO (console output).

WVHEX outputs hexadecimal ASCII characters in INTEL format:



2463-3

character 0 — The **header character**; a colon.

1 & 2 — The number of **data bytes**.

3 through 6 — The program memory **address** of the first data byte in this record.

7 & 8 — The record **type**, where 00 = normal data record, 01 = last record of a file.

9 through n — **Data**. Each pair of hexadecimal digits represents one data byte.

(n+1) & (n+2) — The **checksum**, the 2's complement of the sum of the eight-bit bytes from columns 1 through n.

A record length of 0 indicates the last record of a file.

This is the load module format required by the system loader of previous TEKDOS versions.

MEMORY CONTROL

The commands in this section are used to manipulate system memory, program memory, and the user's prototype memory. The changes and additions specific to Version 2 of TEKDOS include the commands listed below.

SYNTAX

BIAS [W = amount 1] [X = amount 2] [Y = amount 3] [Z = amount 4]

PURPOSE

The BIAS command sets or displays the bias applied to addresses in the DUMP, EXAM, and PATCH commands.

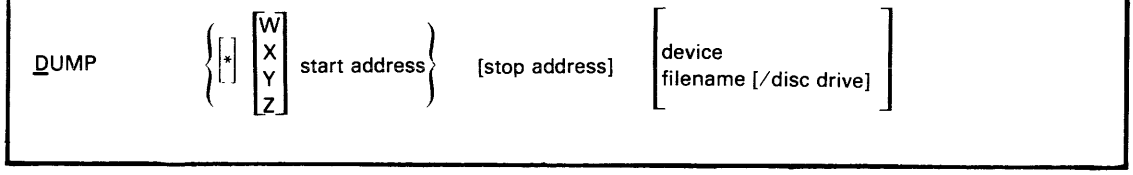
EXPLANATION

The main use of this command is to allow relative addressing when working with one or more relocatable load modules. This is the reason for the four indices: W, X, Y, and Z. The **amount parameters** are positive hexadecimal values. If no BIAS command is given, 0 bias is assumed by DUMP, EXAM, and PATCH. Bias can be removed, from Y and X, for example, by the command:

```
> BIAS Y = 0    X = 0
```

When BIAS is entered without parameters, all current index assignments are displayed on the system console.

SYNTAX



PURPOSE

The DUMP command copies the specified contents of either program or system memory to the device or file named.

EXPLANATION

The DUMP command displays the contents of memory between the **start and stop addresses** as hexadecimal digits. The display format is either 8 words or 16 bytes per line where two hexadecimal digits represent one byte and four digits represent one word. The specific format depends on the microprocessor used. Each line of the display begins with the address of the first byte or word in that line.

If the **asterisk** is present, system memory is selected; otherwise, program memory is accessed.

The start address may also include a **bias index**, W, X, Y, or Z; this index indicates the amount added to both the start and stop addresses. The bias indices are set and changed by the BIAS command. Delimiters should not be used between the asterisk, bias index, and start address. If no bias index is given, 0 bias is applied.

If the stop address is omitted, only one line of 8 words or 16 bytes is displayed. If no device or file is specified, the command defaults to CONO (console output).

The DUMP command reduces the value of the start address to the next lower multiple of 10_{16}, while the stop address is increased to one less than the next higher multiple of 10_{16}; that is, the right-most hexadecimal digit of the **start address** is set to 0, and the right-most digit of the **stop address** is set to F. The adjusted stop address must be greater than the adjusted start address.

```
> DUMP *2456 2455 LPT1
> DUMP *245D 245D LPT1
> DUMP *245F 245Ø LPT1
```

These three commands will all output system memory between 2450 and 245F to the line printer.

```
> DUMP 6AØØ 69FF CONO
```

This command will result in an error message, since the adjusted start address exceeds the adjusted stop address.

SYNTAX

EXAM {[*] [W X Y Z] address}

PURPOSE

The EXAM command displays the contents of the specified hexadecimal address of either program or system memory, and permits those contents to be altered.

EXPLANATION

The EXAM command displays the specified **address** in memory as hexadecimal digits on the system console. The contents of the address may be a two-digit byte or a four-digit word, depending on the processor. The display format is: the specified address, followed by an "=" sign, and then by the contents of that address. When the current address is a multiple of 10_{16}, the display moves to the beginning of the next line on the console.

If the **asterisk** is present, system memory is accessed; otherwise, program memory is selected.

Bias can be applied to the address by using a **bias index**, W, X, Y, or Z, in the address. Bias values are set and changed by the BIAS command. Delimiters should not be used between the asterisk, bias index, and address. If no bias index is given, 0 bias is applied.

Once EXAM is entered, further display or alterations may be made using the following keys:

- SPACE BAR — displays the contents of the next address.
- LINE FEED or RUB OUT — advances to the next line, and displays the current address and its contents.
- RETURN or ESC — terminates the EXAM command.
- Entering a hexadecimal digit pair (for 8-bit processor) or digit quadruple (for 16-bit processor) replaces the contents of the current address with the entered value, and displays the contents of the next address.

Memory contents that have been altered with the EXAM command remain altered after the ESC key is used to terminate EXAM.

SYNTAX

PATCH { [*] [W X Y Z] address } { hexadecimal string }

PURPOSE

The PATCH command alters either program or system memory with the specified string of hexadecimal constants.

EXPLANATION

The PATCH command is used to replace the contents of memory at the specified hexadecimal **address** with the string of hexadecimal digits listed. The **string** replaces the contents of the specified address and subsequent addresses, continuing until the string is exhausted. The string may be from one to 29 bytes long. Word-addressable microprocessors require that the string length be a multiple of four digits; byte-addressable microprocessors require that the string length be a multiple of two digits.

If the asterisk appears in the address, system memory is selected; if not, program memory is accessed.

Bias may be applied to the address by including a bias index, W, X, Y, or Z, in the address. Bias values are set and changed by the BIAS command. If no bias index is given, 0 bias is applied. Delimiters should not be used between the asterisk, bias index, and address.

Section 8
Change Information

DEBUG SYSTEM

This section supplements the information pertaining to the debug system described in Section 8 of the 8002 System User's Manual. The changes and additions specific to Version 2 of TEKDOS are listed below:

TABLE OF CONTENTS

SYNTAX

TRACE {ALL / JMP} [STEP] [start address] [stop address]

or

TRACE OFF

PURPOSE

The TRACE command allows you to monitor program execution through the debug system.

EXPLANATION

The TRACE ALL command monitors all instructions executed by the emulator processor; the TRACE JMP command monitors only jump instructions. Enter the TEKDOS command GO after TRACE ALL or TRACE JMP to begin program execution. The TRACE OFF command stops all monitor activity.

If STEP is specified in the command line, the GO command must be entered after each trace line to continue program execution. If STEP is not specified, two commas must be inserted in the command line. The **start and stop addresses** are the hexadecimal program memory addresses denoting the beginning and the end of the program that is being traced. The stop address must be greater than the start address. The default value for the stop address is FFFF.

The TRACE command generates the display of a trace line on the system console for every program instruction monitored. For the 8080 Emulator Processor, the trace follows the format below:

ADDR INST MNEM OPER SP RF RA RB RC RD RE RH RL

The trace line elements, all of which are in hexadecimal digits, are described:

ADDR	— The address of the last instruction executed.
INST	— The hexadecimal representation of the instruction.
MNEM	— The instruction mnemonic.
OPER	— The value or address of the operand.
SP	— The contents of the stack pointer.
RF	— The contents of the flag register.
RA	— The contents of register A.
RB	— The contents of register B.
RC	— The contents of register C.
RD	— The contents of register D.
RE	— The contents of register E.
RH	— The contents of register H.
RL	— The contents of register L.

Pressing the ESC key while in TRACE command mode suspends program execution; entering a GO command resumes execution and monitoring from the point in the program at which execution was suspended.

DEBUG AND REAL TIME

Because of the large volume of information made available by the TRACE and BKPT commands, the debug system is not able to keep pace with the emulator processor when either of these commands has been invoked through the debug system. To avoid erroneous displays as a result of the incompatible activity rates, the emulator processor's operation is slowed whenever a TRACE or BKPT command is current. If neither a debug TRACE or BKPT is in effect, the emulator processor runs at its normal rate.

Section 9
Change Information

PROM PROGRAMMER

Section 9 of the 8002 System User's Manual describing the PROM Programmer is not affected by the Version 2 update to TEKDOS.

Section 1
Change Information

8002 μPROCESSOR LAB SYSTEM INTRODUCTION

This manual supplements the information contained in the 8002 μProcessor Lab System User's Manual. The material that follows is presented in the same format used in the 8002 System User's Manual; each section in this supplement corresponds to its parallel section in the User's manual. Note that this is a supplement to the 8002 System User's Manual, rather than its replacement.

This manual describes all Version 2 changes and additions to the Tektronix Disc Operating System (TEKDOS). The changed and additional Version 2 information includes such features as hexadecimal data formats, the emulator environment, the debug system, and inter-system communications.

Section 10
Change Information

SERVICE CALLS

Section 10 of the 8002 System User's Manual describing service calls is not affected by the Version 2 update to TEKDOS.

Section 11
Change Information

REAL-TIME PROTOTYPE ANALYZER

Section 11 of the 8002 System User's Manual describing the Real-Time Prototype Analyzer is not affected by the Version 2 update to TEKDOS.

Section 12

Change Information
INTER-SYSTEM COMMUNICATION

By virtue of its inter-system communications capability, the 8002 μProcessor Lab may communicate with another computer system (referred to here as the external computer), or with various peripheral devices.

In this section you will find:

- information about the four RS-232-C ports used for communication with peripheral equipment and modems.
- descriptions of the commands and methods used by the 8002 μProcessor Lab to communicate with another computer system.
- descriptions of the commands and methods used by the 8002 μProcessor Lab in downloading object code from another computer system into the 8002 μProcessor Lab program memory.

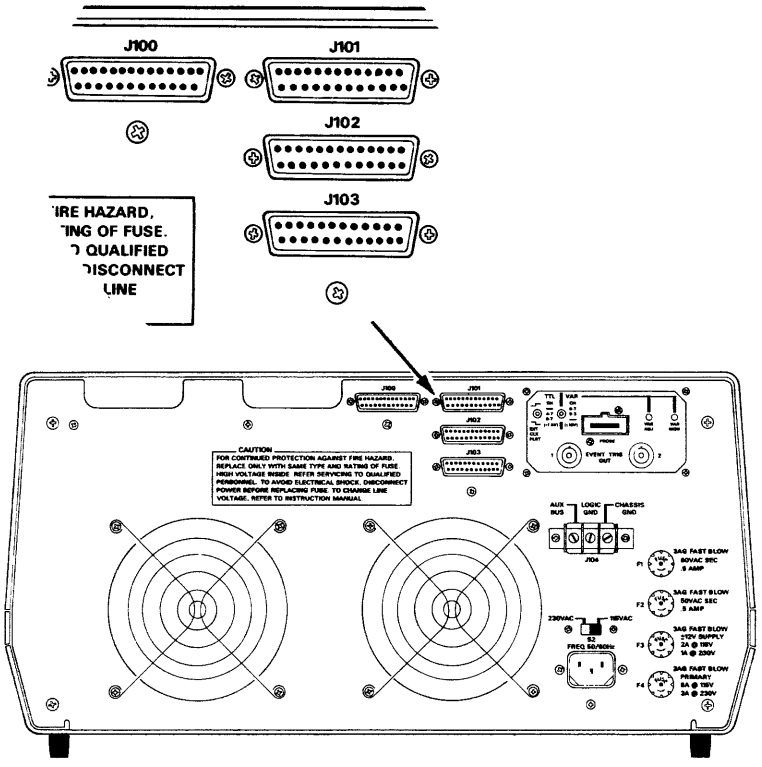
CONTENTS

SYSTEM I/O PORTS

The 8002 μProcessor Lab has four RS-232-C I/O ports. These ports are used for transferring data between the 8002 μProcessor Lab and peripheral equipment. Each of these ports is associated with specific device names in the software system. Electrical specifications for these ports conform to the RS-232-C standards and can be found in the 8002 μProcessor Lab System Service Manual.

The ports are set up for asynchronous serial data transmission. The byte structure is: one start bit, eight data bits and one stop bit. More stop bits are ignored when received. The eighth data bit, the parity bit, is not used.

The jacks for the four I/O ports are located on the rear panel of the main chassis as shown in the drawing below:



2393-2

JACK ASSIGNMENTS

Each I/O port has a specific software device assignment. Each I/O port may connect to a peripheral device by a specific jack on the rear panel of the 8002 μProcessor Lab. Thus, each jack is assigned to a specific software device.

The system console connects to jack J1ØØ. Any terminal-like device may be used as the system console, so long as it has a keyboard, a display and an RS-232-C communications port. Two Tektronix terminals are available as options: the CT8100 CRT Terminal and the CT8101 Printing Terminal.

Jack J1Ø1 is usually connected to a communications modem. This allows the 8002 μProcessor Lab to serve as a remote terminal in communication with another computer system.

Jack J1Ø2 is used to connect the line printer to the 8002 μProcessor Lab. The TEKTRONIX Line Printer LP8200 is available as an option.

Jack J1Ø3 is the paper tape reader/punch connection. The Remex RAR 8050 is recommended for use with the system.

The software device names and their jack assignments are shown in Table 12-1.

Table 12-1

JACK AND DEVICE ASSIGNMENTS

Jack Number	Input Device	Output Device	Device Description
J1ØØ*	CONI	CONO	Console Terminal
J1Ø1	REMI	REMO	Communications Module
J1Ø2	- - - -	LPT1	Line Printer
J1Ø3	PPTR	PPTP	Paper Tape Reader/Punch

*Device TTYR (Teletypewriter reader) may also be used for input to J1ØØ.

The software does not accept any input from J1Ø2. This jack only provides output service to a line printer type of device.

CONTROL CHARACTER HANDLING

When TEKDOS is performing a binary read or write operation, all data transferred is treated as binary data. However, when TEKDOS is performing an ASCII read or write operation, the control characters are handled uniquely for each device.

ASCII Write

For an ASCII write, TEKDOS handles the carriage return character uniquely for each output device. The other control characters are not affected on an ASCII write.

For example, in an ASCII write operation to LPT1, the line printer, carriage return characters are transformed into linefeed characters.

Table 12-2

CARRIAGE RETURN TRANSFORMATION ON AN ASCIII WRITE

Device Name	Transformation from a CR
CONO	CR, LF, XOF, NULL, NULL, NULL
REMO	CR
LPT1	LF
PPTP	CR, LF

CR—carriage return
LF—linefeed
XOF—transmitter off
NULL—null character

Table 12-2 shows how a carriage return character is handled for each of the output devices. The other control characters are not affected on an ASCII write.

ASCII Read

When TEKDOS is performing as ASCII read operation from a device into the 8002 μProcessor Lab, the control characters are transformed. In some cases these transformations are echoed back to the device. In addition, the internal operating status of TEKDOS is affected. Table 12-3 shows the device, each control character, its transformation, and the effect on the status.

Table 12-3

CONTROL CHARACTER TRANSFORMATION ON AN ASCII READ

Device	Control Character	Transformations	Status Effect
CONI	BS (backspace)	Echo BS and delete last character entered.	
	CR (carriage return)	CR LF XOF NULL NULL NULL.	EOL (end of line).
	CTRL-Z	CR.	EOL EOF (end of line, then end of file).
	DEL	Echo and delete last character.	
	ESC	CR LF XOF NULL NULL NULL.	Break, then EOL.
	LF	Ignored.	
	NULL	Ignored.	
	XOF	Echo XON.	
TTYR	Treats all control characters in the same manner as CONI. In addition, an ASCII read from TTYR: 1) enables the TTY reader bit in the I/O port that starts the reader on a mini-computer-modified ASR-33, and 2) sends XON to CONO.		
PPTR and REMI	CR	CR.	EOL.
	CTRL-Z	No transformation.	Kills line, EOF.
	DEL	Ignored.	
	LF	Ignored.	
	NULL	Ignored.	

SYNTAX

COMM [E = {L|R}] [L = {I|O}] [P = prompt sequence] [T = delay time] [M = parity]

PURPOSE

The TEKDOS command COMM, allows the 8002 μProcessor Lab to communicate with an external computer system.

EXPLANATION

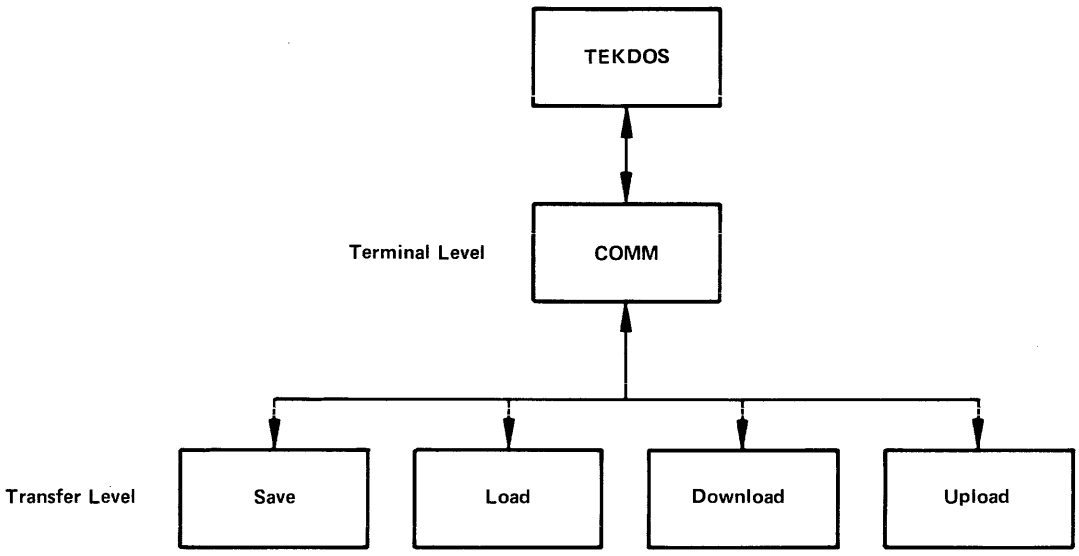
To communicate with an external computer, connect the communications modem to J1Ø1 on the rear panel of the 8002 μProcessor Lab. The logical devices REMI and REMO are assigned to this port.

The COMM command invokes the communications support package which allows operation of the 8002 μProcessor Lab either as a terminal or in a transfer level mode with an external computer.

The external computer software must provide for transferring files. In the case of the load mode or the save mode, the external computer software must be capable of transmitting or receiving object code files prepared in Tektronix Hexadecimal Format. In the unformatted modes, the external computer software must be able to transmit or receive any file, regardless of format or content. A discussion of Tektronix Hexadecimal Format appears beginning on page 7-3 of this supplement.

OPERATIONAL LEVELS

The TEKDOS communication command, COMM, supports two operational levels: the terminal level and the transfer level. In addition, the transfer level can be entered in any one of four modes: load mode, save mode, or one of two unformatted modes.



Whenever the COMM command is invoked, the terminal level is entered. One of the operational modes in the transfer level can then be entered. Each operational mode can be entered only from the terminal level. After exiting from the transfer level, control is passed back to the terminal level. Control can then be returned to TEKDOS from the terminal level.

Terminal Level

When the COMM command is invoked and the terminal level is entered, the 8002 μProcessor Lab acts as a terminal to the external computer. With the exception of the first null character typed, every character typed on the keyboard is sent to the external computer. To send a null to the external computer, two null characters must be entered from the keyboard.

A null character is entered from most terminals by holding down the CTRL and SHIFT keys while striking the "P" key. However, with the TEKTRONIX CT8101 Printing Terminal, the null character is entered by holding down the CTRL key while striking the semicolon key.

Control is returned to TEKDOS from the terminal level by entering NULL ESC.

Transfer Level

Each operational mode at the transfer level is entered from the terminal level. The method of entry, conditions for exit, and uses of each mode are described in the following text.

Load Mode

The load mode is used to download Tektronix Hexadecimal Formatted files from an external computer system into the program memory of the 8002 μProcessor Lab. The load mode is entered from the terminal level by typing a null character followed by a carriage return.

When the external computer has sent a message block and its prompt sequence, if any, and the delay time has elapsed, the 8002 μProcessor Lab sends an acknowledgment back to the external computer. Receipt of the message block is also indicated on the console by an asterisk. If the message block is correctly received and a positive acknowledgment has been sent, the load address and the byte count are displayed on the console.

Exit from the load mode to the terminal level may be caused by an of the following coniditions:

- Entering a BREAK from the keyboard.
- Receiving a terminating message from the external computer.
- Receiving an abort message from the external computer.
- Attempting to load beyond location FFFF in program memory.
- Memory write error in the 8002 μProcessor Lab.
- Device read error in the 8002 μProcessor Lab.

Save Mode

The save mode is used to transfer object code in TEKHEX format from the program memory of the 8002 μProcessor Lab to an external computer. The save mode is entered from the terminal level by typing the following command sequence:

$$\{\text{NULL}\}\left\{\begin{matrix}\\ \text{startaddr}\end{matrix}\right\}\{\text{endaddr}\}\left[\{\text{startaddr}\}\{\text{endaddr}\}\right]\ldots\left[\text{transaddr}\right]$$

The **NULL** character is entered by holding down the CTRL and the SHIFT keys while striking the "P" key (with the TEKTRONIX CT8101 Printing Terminal, by holding down the CTRL key while striking the semicolon key). The start address (**startaddr**) and the end address (**endaddr**) are the addresses of the lower and upper bounds, respectively, of the object code in program memory. The transfer address (**transaddr**), is an optional address for the location to begin execution. These addresses are entered as hexadecimal values in a single command line. Control is returned to the terminal level as soon as the specified data blocks have been transferred.

The save mode waits for the prompt sequence (if one was specified) from the external computer before sending each message block. Each message block consists of 72 or fewer characters, and is terminated by a carriage return. If the external computer sends a negative acknowledgment, the current message block is sent again. If the external computer sends a positive acknowledgment, the value of the current location and the byte count are displayed on the console. Then the next message block is transferred to the external computer.

Exit from the save mode to the terminal level may be caused by any one of the following conditions:

- Entering an ESC from the keyboard.
- Receiving an abort message from the external computer.
- Device read or write error in the 8002 μProcessor Lab.

Unformatted Modes

The unformatted modes are used to transfer files between the 8002 μProcessor Lab and an external computer. The files to be transferred may be in any ASCII format and are transferred directly as they appear, with no error checking.

The unformatted modes are entered from the terminal level by typing one of the following command sequences:

{name / command} {NULL} > {file name} Unformatted down-transfer mode

{name / command} {NULL} < {file name} Unformatted up-transfer mode

- The first element is the **name** or **command** that starts execution of the transfer program on the external computer.

NOTE

Since the 8002 is acting as a terminal to the external computer, the user may enter any number of instructions valid to the external computer operating system; the 8002 μProcessor Lab is not invoked until a NULL character is entered. The last command sent to the external computer before this NULL character is entered should be the name of the transfer command or program on the external computer.

- Then enter the NULL character. The NULL character is sent from most terminals by holding down the CTRL and SHIFT keys while striking the "P" key. On the TEKTRONIX CT8101 Printing Terminal, the NULL character is sent by holding down the CTRL key while striking the semicolon key.

- The next character is either a greater-than or a less-than.

  When a file residing on the external computer is to be transferred to the 8002 μProcessor Lab, enter the greater-than character >.

  When a file residing on the 8002 μProcessor Lab is to be transferred to the external computer, enter the less-than character <.

- Next enter the **file name** of the file to be transferred.

- Finally, enter a carriage return.

Exit from the unformatted down mode to the terminal level may be caused by striking the BREAK key when the file tranfer from the external computer is finished. This may be signified on the console by an asterisk. Exit from the unformatted up mode to the terminal level may be caused by entering NULL ESC. Exit from either unformatted mode may be caused by a device read or write error.

PARAMETERS

The parameters **E, L, P, M,** and **T** are optional; each defaults to a preset value if not included in the command line. The order of entry in the command line is not important.

Commas or spaces may be used as delimiters between the parameters.

Echo Parameter, E=

The echo parameter specifies local or remote echoing of any character entered on the console keyboard. The two variables for the echo parameter are **L** and **R**. **E=L** (local echo) directs the 8002 μProcessor Lab to echo any character entered. **E=R** (remote echo) directs the 8002 μProcessor Lab not to echo characters, indicating that the external computer is expected to echo any character entered. **R** is the default variable.

If the external computer does not echo the entered character, the character is not displayed unless the 8002 μProcessor Lab is set for local echo.

If the 8002 μProcessor Lab is set for local echo and the computer also echoes the character, the character is displayed twice.

Linefeed Parameter, L=

The **linefeed** parameter is used to echo a line feed by the 8002 μProcessor Lab when a carriage return is entered at the keyboard. Two variables, **I** and **O**. **I** (included) causes the 8002 to echo the linefeed. **O** (omitted) causes the carriage return alone to be echoed without linefeed. **O** is the default variable.

With the echo parameter set to local (**E=L**), the 8002 μProcessor Lab echoes only those characters entered from the keyboard. When local echo is specified, you must enter **L=I** (linefeed included), to echo a linefeed with the carriage return; otherwise you get a carriage return without linefeed.

If the external computer is echoing characters entered from the keyboard and in addition is returning a linefeed with the carriage return, set the linefeed parameter to **L=O** (linefeed omitted). If the external computer is echoing a linefeed and you specify **L=I** you receive two linefeeds with every carriage return.

If the external computer is not echoing a linefeed with the carriage return, set **L=I** to get the linefeed echoed with the carriage return.

Prompt Sequence Parameter, P=

This parameter is used to specify the **prompt character sequence** of the external computer communicating with the 8002 μProcessor Lab at transfer level. The prompt character sequence parameter defaults to the equivalent of no prompt sequence from the external computer.

The prompt sequence consists of up to six ASCII characters, represented by up to twelve hexadecimal digits. Refer to Appendix B of this supplement, "ASCII Code Conversion Table".

The prompt parameter is used in both unformatted modes to indicate an end of data transfer. In an unformatted upload, the prompt sequence may be sent by the external computer after each line of text is received and verified. In an unformatted download, the prompt character indicates an end of transmission. If no prompt sequence is specified, the BREAK key is used to end transmission.

The following examples show several prompt sequences and the parameter entry for specifying each.

Prompt Sequence	Parameter Entry
WHAT ?	P=57484154ØØ3F
(space) *	P=ØØ2A
>	P=3E

Delay Parameter, T=

The **delay** parameter indicates the time required by the external computer to prepare itself to receive a message after sending a message to the 8002 μProcessor Lab at transfer level. The unformatted modes do not require the **T** parameter. The variable is given in 100 millisecond units, and is entered as a two digit hexadecimal number. For example, a 300 ms delay is entered as **T**=03. The maximum turnaround time is 25.5 seconds, entered as **T**=FF. The default value for this parameter is 0 (no delay).

Parity Parameter, M=

The following table shows the values which may be assigned to **M**.

M=	Character Length	Parity	Stop Bits
0	7	even	2
1	7	odd	2
2	7	even	1
3	7	odd	1
4	8	none	2
5	8	none	1
6	8	even	1
7	8	odd	1

Parameter Selection

Selecting the parameters and variables needed for communicating with your external computer involves some experimentation on your part. You may be able to get some information from your computer center. However, the final set of parameters and variables you will need will have to be determined by experimentation. Select those parameters that work.

The best order to use in trying to select your parameters and variables is:

1. E—echo
2. L—linefeed
3. P—prompt sequence
4. T—delay time
5. M—parity

This whole series of experiments will require that the 8002 μProcessor Lab be connected to your external computer. Each experiment may require invoking the COMM command and logging on the external computer a number of times.

1. E—**echo** parameter: This experiment is used to determine if the external computer is echoing each character received from the 8002 μProcessor Lab.

   First, invoke the COMM command without any parameters. Next, log on to your computer.

   Now strike a character key on the console keyboard. Two conditions may occur; either the character entered appears on the console display, or nothing new appears on the display.

   A. If the character entered does not appear on the console display, your external computer is not echoing each character. Try several other characters to double check. If the characters entered are still not being echoed to the console by the external computer, you should use the **local echo** parameter, **E=L**, when invoking the COMM command.

   B. If the entered character does appear on the console display, your external computer is echoing each character it receives. Try several other characters to double check. If these characters also appear, you do not need to enter an echo parameter, or you may use the **remote echo** parameter, **E=R**.

2. L—**linefeed** parameter: This experiment is used to determine if a carriage return-linefeed is echoed by the external computer when it receives a carriage return.

   Enter a NULL ESC, returning control to TEKDOS. (NULL is entered on most terminals by holding down the CTRL and SHIFT keys while striking the P key.)

Now invoke the COMM command, entering the echo parameter determined in experiment one. Example: enter one of the following commands:

COMM **E=L** or COMM **E=R**

A. If a linefeed is echoed with the carriage return (i.e., the console display cursor or print head returns to the start of the next line), your external computer has echoed the carriage return as a carriage return-linefeed. Try the carriage return several more times. If a linefeed is still echoed with the carriage return, you do not need to specify any linefeed parameter.

B. If the linefeed is not echoed with the carriage return (i.e., the console display cursor or print head returns to the beginning of the **current** line), your external computer is not echoing a linefeed with the carriage return. Confirm this with several more tries.

   If the line feed is not being echoed with the carriage return by the external computer, you should use the **linefeed included** parameter, **L=I**, when invoking the COMM command.

3. P—**prompt sequence** parameter: The purpose of this experiment is to determine the characters, if any, that make up the prompt sequence from your external computer. By now you may have observed several prompt sequences from your external computer. The external computer may use different prompts for specific situations. A simple prompt like an asterisk or an exclamation mark may be used when the monitor is prompting. A response to unintelligible input may be the prompt sequence, "WHAT ?", or even the comment, "SYNTAX ERROR", followed on the next line with "OK".

   The prompt sequences above are not the type sought by this experiment. Here you want to determine the prompt sequence generated by your external computer during the execution of the download program. This prompt sequence can be determined by observing the console display during execution of the download program on the external computer.

   First, enter NULL ESC to return control to TEKDOS. Next, invoke the COMM command with the previously determined echo and linefeed parameters.

Now, call up and execute your download program on the external computer. The download program should have the external computer perform the following actions:

A. Read the first message block from the data file.

B. Transmit the message block to the 8002 μProcessor Lab, where the message will be displayed on the console.

C. Transmit a prompt sequence to the 8002 μProcessor Lab, which will display the prompt sequence on the console.

The visible portion of the external computer prompt sequence is then used in the **P=** parameter.

For example, your external computer may send a space followed by a question mark as the visible portion of the prompt sequence. You would then enter the following prompt sequence parameter as part of your COMM command:

P=2Ø3F

Your external computer may not send a visible prompt sequence. In this case, you do not need to enter a prompt sequence parameter when invoking the COMM command.

4. T—**delay** parameter: The delay time must be long enough to allow your external computer to prepare itself to receive a response from the 8002 μProcessor Lab. A good starting place for this experiment is 300 milliseconds[a]. Invoke the COMM command with the previously defined parameters, and set **T=03.** Then enter the command line needed to call up and start execution of your external computer download program. Instead of striking the carriage return, enter a NULL character and then the carriage return. This puts COMM into load mode while sending the command line to your external computer.

   After the first message block is received, the 8002 μProcessor Lab displays an asterisk, followed by the load address and byte count.

   A. If the delay time is long enough, the 8002 μProcessor Lab continues to display a new asterisk, load address and byte count for each message block. This indicates that the specified turnaround time is sufficient. It may, however, be unnecessarily long. You may want to invoke the COMM command with the turnaround time set to a lesser value, or even to zero.

   B. If the first asterisk, load address and byte count are displayed but after 5 to 10 seconds nothing else appears, then the delay time is too short and the system is very likely hung up. Strike the ESC key several times to return to terminal mode. Then enter NULL ESC to return control to TEKDOS. Now, start over with a larger turnaround time. Some timesharing systems may need 500 milliseconds or longer.

5. M—**parity** parameter: Consult the specifications for your external computer to determine the correct parity option.

[a]Remember, delay time is specified in hundred millisecond units (one hundred milliseconds equals one tenth of a second). The variable portion of the turnaround time parameter must contain at least two digits—hence T=03.

DOWNLOADING AND UPLOADING

The terms **downloading** and **uploading** refer to the transfer of files between the 8002 μProcessor Lab and and external computer. Downloading and uploading require that the COMM command be invoked so that the 8002 μProcessor Lab can properly handle the messages and check for the receipt of valid data.

Formatted downloading requires that the external computer be programmed to transmit the object code in hexadecimally formatted message blocks. In addition, the downloading program on the external computer should respond to message acknowledgments from the 8002 μProcessor Lab.

Formatted uploading requires that the external computer be programmed to receive and store the hexadecimally formatted object code message blocks. Upon receipt of each message block, the uploading program on the external computer should return a message acknowledgment to the 8002 μProcessor Lab.

Formatted Download Message Handling (LOAD Mode)

Formatted downloading procedure involves several steps:

- The external computer sends a message block containing data to the 8002 μProcessor Lab.
- The 8002 μProcessor Lab checks the validity of the message block received.
- The 8002 μProcessor Lab sends a message acknowledgment to the external computer.

If either a prompt character sequence or a turnaround delay has been specified as a parameter in the COMM command, the specified parameters must be satisfied before the 8002 μProcessor Lab sends an acknowledgment to the external computer.

The external computer program sends one message block and then waits for an acknowledgment. If the message block is correct, the 8002 μProcessor Lab returns a positive acknowledgment to the external computer. The external computer then sends the next message block.

If the message block received from the external computer is not correct, the 8002 μProcessor Lab returns a negative acknowledgment to the external computer. Upon receipt of this acknowledgment the external computer may take some appropriate action. Usually the action is to retransmit the message block. If the 8002 μProcessor Lab continues to issue a negative acknowledgment, the external computer may then send an abort message.

A negative acknowledgment is sent back to the external computer when either checksum is wrong, or when a header character is not detected at the beginning of the message block.

The ASCII character "Ø" is a positive acknowledgment. The ASCII character "7" is a negative acknowledgment. Either acknowledgment is followed by the EOL character, a carriage return.

Comparison of Checksums

When the external computer sends a message block, the 8002 μProcessor Lab receives the entire block. The first checksum of the message block is verified. If the first checksum is bad, the 8002 μProcessor Lab returns a negative acknowledgment to the external computer and then waits for the next transmission.

If the first checksum verifies the location counter and the byte count data as being good, the data characters received are converted into bytes and are stored in the appropriate program memory. During this conversion, a second checksum is accumulated.

The second checksum transmitted by the external computer is then compared with the calculated second checksum. If these two checksums are the same, a positive acknowledgment is returned to the external computer. If these two checksums are not the same, a negative acknowledgment is sent to the external computer.

In summary, a negative acknowledgment may be sent to the external computer if either the first or the second checksum verification is bad. A positive acknowledment is not sent until the second checksums are verified as being the same. Whether the second checksum verification is positive or negative, the data block is stored in program memory. If the first checksum verification is negative, the data block is not stored in program memory.

Sample Download Program

The following program listing is an example of a downloading program written in FORTRAN. This program resides in the external computer and is used in transferring data blocks to the 8002 μProcessor Lab. The data to be downloaded must have been previously converted to Tektronix Hexadecimal Format by the external computer. (The Tektronix Hexadecimal Format is explained in Section 7.)

In this sample downloading program, the line numbers are for reference only and are not to be entered as part of the program code. A positive acknowledgment is often referred to as an **ACK,** while a negative acknowledgment is referred to as a **NAK.**

```
 1 C * THIS PROGRAM TRANSMITS A TEKTRONIX HEXADECIMAL FILE TO THE
 2 C * 8002 uPROCESSOR LAB PROGRAM MEMORY.  THIS PROGRAM IS RUN IN
 3 C * CONJUNCTION WITH THE COMMUNICATIONS PROGRAM ON THE 8002
 4 C * uPROCESSOR LAB.  EACH MESSAGE SENT TO THE 8002 uPROCESSOR
 5 C * LAB IS ACKNOWLEDGED BY EITHER A POSITIVE ACKNOWLEDGMENT,
 6 C * A ZERO, OR A NEGATIVE ACKNOWLEDGMENT, A SEVEN.  IF 5
 7 C * NEGATIVE ACKNOWLEDGMENTS TO A MESSAGE ARE RECEIVED, THIS
 8 C * PROGRAM HALTS AFTER SENDING THE 8002 uPROCESSOR LAB AN
 9 C * ABORT MESSAGE.
10 C *
11 C * LOGICAL UNIT ASSIGNMENTS
12 C *   5 - 8002 (TERMINAL) INPUT
13 C *   6 - 8002 (TERMINAL) OUTPUT
14 C *   7 - MESSAGE FILE
15 C *
16         DIMENSION MESS (80)
17         DATA IZERO/'0'/,IACK/'0'/,ISPACE/' '/,ISLASH/'/'/
18 C * INITIALIZE THE NEGATIVE ACKNOWLEDGMENT COUNTER
19 10      NAKCT = 0
20 C * READ A MESSAGE
21         READ (7,20,END = 999) MESS
22 20      FORMAT (80A1)
23         IF (MESS(1).NE.ISLASH) GO TO 10
24 C * FIND THE END OF THE MESSAGE
25         DO 30 J = 1,80
26         NUM = 81 - J
27         IF (MESS(NUM).NE.ISPACE) GO TO 40
28 30      CONTINUE
29 C * TRANSMIT THE MESSAGE
30 40      WRITE (6,20) (MESS(J),J = 1,NUM)
31         READ (5,20) IREPLY
32         IF (IREPLY.EQ.IACK) GO TO 10
33         NAKCT = NAKCT + 1
34 C * RETRANSMIT IF LESS THAN 5 CONSECUTIVE NEGATIVE ACKNOWLEDGMENTS
35         IF (NAKCT.LT.5) GO TO 40
36         GO TO 1000
37 C * IF THE LAST MESSAGE WASN'T AN END MESSAGE, SEND ONE
38 999     IF (MESS(6).EQ.ZERO.AND.MESS(7).EQ.ZERO) STOP
39         NAKCT = 0
40 50      WRITE (6,60)
41 60      FORMAT ('/00000000')
42         READ (5,20) IREPLY
43         IF (IREPLY.EQ.IACK) STOP
44         NAKCT = NAKCT + 1
45         IF (NAKCT.LT.5) GO TO 50
46 C * SEND AN ABORT MESSAGE
47 1000    WRITE (6,70)
48 70      FORMAT ('//DOWNLOAD ABORTED - 5 CONSECUTIVE NAKS RECEIVED')
49         STOP
50         END
```

The object code to be downloaded by this program must be converted to Tektronix Hexadecimal Format and stored on a message file prior to execution of this program. The download program reads each message block in turn from the message file. The current message is then sent to the 8002 μProcessor Lab which acknowledges receipt of the message.

In the program listing, lines 24 through 28 show that this program can handle a maximum of 80 characters in a message.

Lines 35 and 36 allow the current message to be transmitted five times with negative acknowledgments. After receiving the fifth NAK, the computer jumps to 1000 at line 47 and sends an **abort** message to the 8002 μProcessor Lab.

When an end-of-file condition is detected at line 21, the computer jumps to 999 at line 38. If the previous message was not an **end** message, the **end** message shown at line 41 is sent. This may also be transmitted five times with a negative acknowledgment. But after the fifth NAK, the **abort** message is sent.

The FORTRAN code at line 21, specifically **END=999**, is the method used by an IBM 370 to check for and end-of-file condition. If you are using a CDC computer, the following two lines of code are substituted for line 21:

```
READ (7,20) MESS
IF (EOF (7)) GO TO 999
```

The method used in checking for the end-of-file condition varies with the computer being used. Change this code to comply with the FORTRAN subset of your computer system.

The first character in a Tektronix Hexadecimal Formatted message block is a slash (/). If a message block is transmitted without a slash as the first character, the 8002 μProcessor Lab returns a negative acknowledgment. The fifth NAK received by the download program causes an **abort** message to be downloaded. The download program contains code at line 23 to detect and then disregard any message block without a header slash.

Your download program should check for the absence of the header slash in every message block, and should then take appropriate action when the header slash is not present.

Unformatted Download Message Handling

In this mode, the transfer program on the external computer can be any program that can copy or read from the system console. An example of such a program is the TEKDOS command, COPY. No verification or error checking is done in this mode.

The external computer sends the prompt sequence specified in the COMM command line, to indicate the end of data transfer. If no prompt sequence has been specified, the BREAK key is used to end transmission.

Formatted Upload Message Handling (SAVE Mode)

Formatted uploading follows these general steps:

- The 8002 μProcessor Lab sends a message block to the external computer.
- The external computer checks the validity of the message block received.
- The external computer sends a message acknowledgment to the 8002 μProcessor Lab.

If either a prompt character sequence or a turnaround delay has been specified as a parameter in the COMM command, the specified parameters must be satisfied before the 8002 μProcessor Lab sends a message block to the external computer. After the 8002 μProcessor Lab sends one message block, it waits for an acknowledgment from the external computer.

If the external computer receives a valid message block, the upload program on the external computer should return a positive acknowledgment. The 8002 μProcessor Lab then sends the next message block.

If the message block received by the external computer is invalid, the upload program on the external computer should return a negative acknowledgment. When the 8002 μProcessor Lab receives a negative acknowledgment, it retransmits the current message block. The upload program on the external computer should count the numbers of times it receives a negative acknowledgment for a message block. After a number of tries (any number may be used, but usually three tries is sufficient) the upload program on the external computer should take some appropriate action.

The action taken after sending a given number of negative acknowledgments might be to send an abort message to the 8002 μProcessor Lab. If you want to continue uploading in spite of an invalid message block, you could program the external computer to send a positive acknowledgment to the 8002 μProcessor.

The ASCII character "Ø" is a positive acknowledgment. The ASCII character "7" is a negative acknowledgment. Either acknowledgment is followed by the EOL character, a carriage return.

Sample Upload Program Flowchart

The following flowchart is for a formatted upload program that is to reside in the external computer.

The program written from this flowchart receives message blocks from the 8002 μProcessor Lab. The data to be uploaded is in Tektronix Hexadecimal Format.

In the flowchart, a positive acknowledgment is referred to as an ACK and a negative acknowledgment is referred to as a NAK.

The NAK counter is used to count the number of negative acknowledgments sent to the 8002 μProcessor Lab for each message block. A negative acknowledgment is sent each time that either checksum is determined to be invalid. When five NAKs have been sent for a message block, this flowchart shows that the external computer sends an abort message to the 8002 μProcessor Lab. Further transmission is then stopped.

The upload program checks for zeros at data items six and seven in each message block received. When data items six and seven are both zero, a terminating block has been received.

The upload program checks data items one and two in each message block for slashes. A slash in one and two indicates receipt of an abort block.

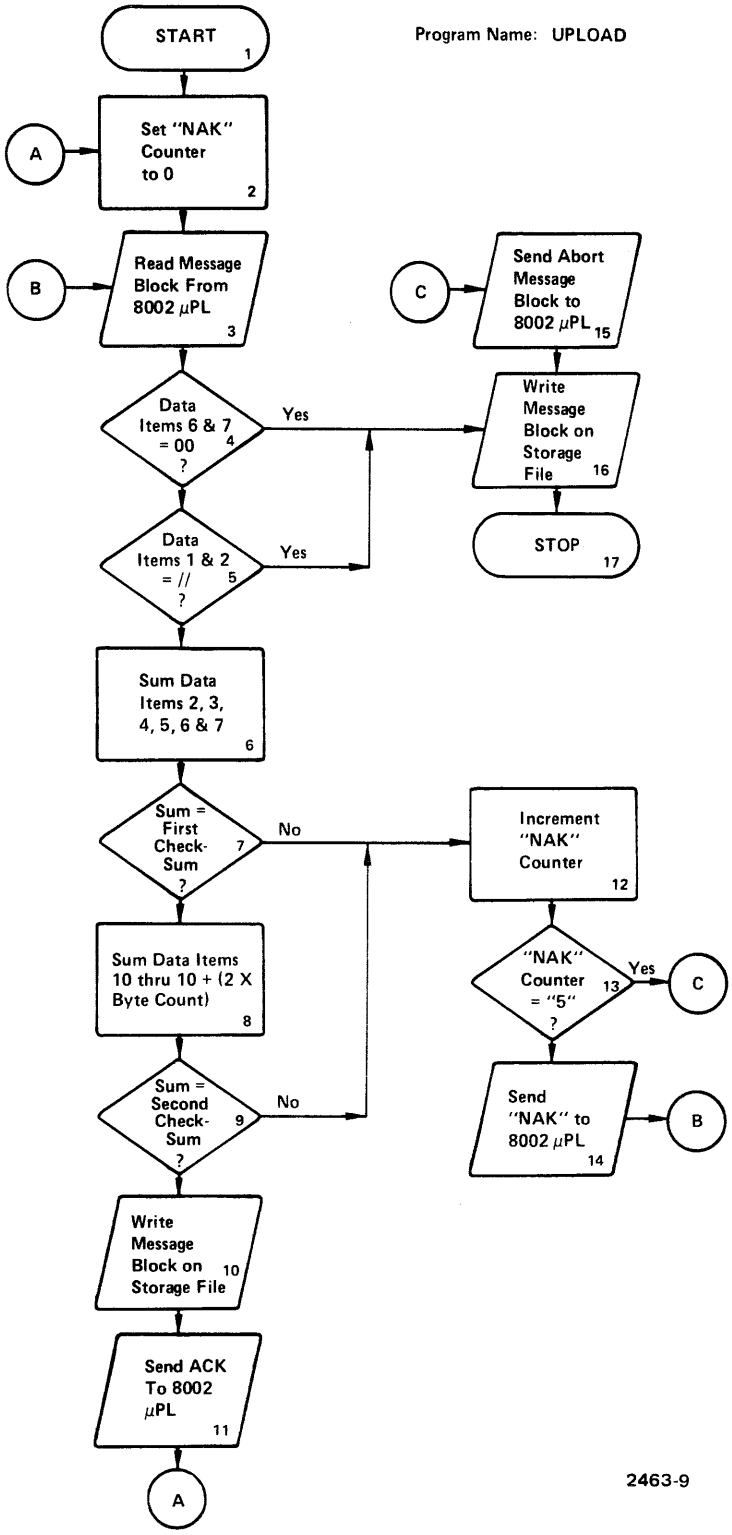
Every valid message block (this includes terminating blocks, abort blocks, and data blocks) is stored on a file in the external computer.

Every message block received, whether valid or invalid, must be acknowledged by the external computer.

Unformatted Upload Message Handling

In this mode, the transfer program on the external computer can be any program that can copy or read from the system console. An example of a transfer program is the TEKDOS command, COPY. No verification or error checking is done in this mode.

The external computer echoes the prompt sequence specified in the COMM command line after receiving each message block. This initiates transfer of the next message block.


START
1
Program Name: UPLOAD
A
Set "NAK" Counter to 0
2
B
Read Message Block From 8002 μPL
3
C
Send Abort Message Block to 8002 μPL
15
Data Items 6 & 7 = 00 ?
4
Yes
Write Message Block on Storage File
16
Data Items 1 & 2 = // ?
5
Yes
STOP
17
Sum Data Items 2, 3, 4, 5, 6 & 7
6
Sum = First Check-Sum ?
7
No
Increment "NAK" Counter
12
Sum Data Items 10 thru 10 + (2 X Byte Count)
8
"NAK" Counter = "5" ?
13
Yes
C
Sum = Second Check-Sum ?
9
No
Send "NAK" to 8002 μPL
14
B
Write Message Block on Storage File
10
Send ACK To 8002 μPL
11
A
2463-9

SYNTAX

SEND

PURPOSE

The SEND command transfers Tektronix object code files from the 8002 μProcessor Lab to the 8001 μProcessor Lab.

EXPLANATION

The SEND command invokes a computer program that allows the 8002 μProcessor Lab to function as an external computer to the 8001 μProcessor Lab. When the SEND program receives a valid filename from the 8001 μProcessor Lab, it initiates transfer of message blocks in Tektronix Hexadecimal Format. A description of Tektronix Hexadecimal (TEKHEX) Format appears in section 7 of this supplement.

Receipt of a positive acknowledgment initiates transfer of the next message block. Receipt of a negative acknowledgment causes the current message block to be retransmitted. An abort message block is transmitted if an invalid filename is received or a file read error occurs.

The SEND command returns control to TEKDOS when any one of the following conditions is met:

- Transmission of a terminating message and receipt of a positive acknowledgment.
- Receipt of two break signals from the 8001 μProcessor Lab System.
- The ESC key is struck twice on the console of the 8002 μProcessor Lab, followed by ABORT SEND (or ABORT *).

Procedure for Use of the SEND Command

The following procedure is used for effecting communication between the 8002 μProcessor Lab and the 8001 μProcessor Lab. Both labs must be operating as stand-alone units, that is the 8001 μProcessor Lab cannot be considered as the terminal for the 8002 μProcessor Lab. Each must have its own console.

The cable connecting the two labs is plugged into J101 (REMI,REMO) on the 8001 μProcessor Lab. The other end of the cable is plugged into J101 (REMI,REMO) on the 8002 μProcessor Lab. The cable should be an RS-232-C standard cable with a null modem attached, or equivalent. For further explanation of a null modem, see Appendix C of this supplement.

The transmission rates for the ports being used for this communication link should be set at 2400 baud.

Procedure Steps:

1. Start both the 8002 μProcessor Lab and the 8001 μProcessor Lab.

2. Invoke the COMM command on the 8001 μProcessor Lab by entering the following command line:

   COMM P=3F E=L L=I

3. Invoke the SEND command on the 8002 μProcessor Lab.

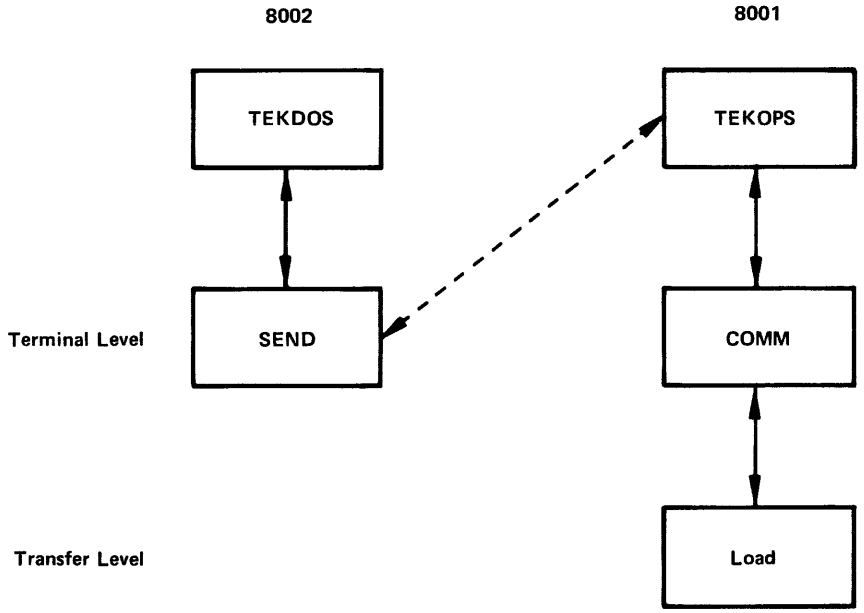
4. On the 8001 μProcessor Lab enter the following command line to invoke the load mode:

   file-name NULL carriage return

NOTE

A NULL character is entered from most terminals by holding down the CTRL and SHIFT keys while striking the "P" key. On the CT8101 Printing Terminal the NULL character is entered by holding down the CTRL key while striking the semicolon key.

At this point the file named will be loaded from the 8002 μProcessor Lab into the 8001 μProcessor Lab program memory.



SEND Command Organization. 2463-2

When transmission is completed, enter a NULL ESC on the 8001 μProcessor Lab console. To abort SEND during execution and thus return both μProcessor Labs to their operating systems, enter either:

1. BREAK BREAK NULL ESC on the 8001 μProcessor Lab System or
2. ESC ESC ABORT SEND on the 8002 μProcessor Lab System, followed by NULL ESC on the 8001 μProcessor Lab System.

Appendix A
ERROR CODES

TEKDOS ERROR CODES

1 — DIRECTORY READ ERROR
2 — DIRECTORY WRITE ERROR
3 — COMMAND FILE NOT FOUND
4 — COMMAND FILE INPUT ERROR
5 — PROCEDURE BUSY
6 — DEVICE READ ERROR
7 — DEVICE WRITE ERROR OR END-OF-DEVICE
8 — DRIVE NOT SPECIFIED
9 — INVALID DRIVE
10 — COMMAND LOAD FAILURE
11 — MEMORY AREA IN USE
12 — INVALID FILE NAME
13 — INPUT FILE NOT FOUND
14 — INVALID INPUT DEVICE
15 — INVALID OUTPUT DEVICE
16 — INPUT DEVICE ASSIGN FAILURE
17 — OUTPUT DEVICE ASSIGN FAILURE
18 — DEVICE IN USE
19 — INVALID CHANNEL NUMBER
20 — CHANNEL IN USE
21 — CHANNEL ASSIGN FAILURE
22 — COMMAND LINE BUFFER OVERFLOW
23 — INVALID COMMAND
24 — JOB NOT ACTIVE
25 — JOB NOT SUSPENDED
26 — JOB ALREADY SUSPENDED
27 — JOB EXECUTING
28 — JOB UNDER DEBUG CONTROL
29 — PROM POWER FAILURE
30 — INVALID PARAMETER
31 — PARAMETER REQUIRED
32 — TOO MANY PARAMETERS
33 — BIAS PARAMETER ERROR
34 — INVALID ADDRESS
35 — INVALID START ADDRESS
36 — INVALID END ADDRESS
37 — INVALID GO ADDRESS
38 — INVALID DEBUG USER PROGRAM ADDRESS
39 — INVALID HEX CHARACTER
40 — INVALID RHEX INPUT FORMAT
41 — INVALID BREAKPOINT ACCESS MODE
42 — INVALID REGISTER PARAMETER
43 — INVALID DATA PARAMETER
44 — INVALID TRACE MODE PARAMETER
45 — INVALID EMULATOR SRB ADDRESS
46 — EMULATOR HALTED
47 — SYSTEM AREA BAD
48 — FETCH FILE NOT FOUND
49 — FETCH FILE ASSIGN FAILURE
50 — FILE NOT A FETCH MODULE
51 — INVALID FETCH REQUEST
52 — INVALID DEVICE
53 — INVALID EMULATOR PROCESSOR
54 — INVALID MODE
55 — INVALID MEMORY
56 — INVALID DEVICE ADDRESS
57 — FILE NAME IN USE
58 — DEVICE ASSIGN FAILURE
59 — MEMORY WRITE ERROR
60 — END OF MEDIA
61 — FILE IN USE
62 — DEVICE NOT OPERATIONAL
63 — DIRECTORY FULL
64 — INVALID DISC
65 — SYSTEM MEMORY PARITY ERROR
66 — PROGRAM MEMORY PARITY ERROR
67 — EMULATOR CLOCK MISSING
68 — EMULATOR FAULTED
69 — ADDRESS NOT ON WORD BOUNDARY
70 — WORD OR BYTE BOUNDARY ERROR

TEXT EDITOR ERROR MESSAGES

This section provides a list of all Editor messages and an explanation of their meaning.

WSP FULL

The buffer is full.

NOT FOUND

The given string could not be found.

DISC FULL

Output disc is full.

NUMBER

The parameter n is in error.

RANGE

The parameter N is in error, or an attempt was made to reference lines which are not in the workspace.

MODE

An attempt was made to execute a macro string from within a macro string; this is not allowed.

NEST

The nesting brackets < and > do not balance.

COMMAND?

An unknown command was encountered in the command line.

BREAK

The ESCAPE Console Key was depressed to terminate execution of a file I/O function.

PROCEDURE ERROR

Editor usage is in error.

TEKDOS STAT=XX

XX is the TEKDOS SRB status byte returned to the Editor when an unusual request or event has occurred. The meaning of the status byte can be found in Section 10 of the System User's Manual.

NO PI

For this editing session there is no PRIMARY INPUT file; the user may not do GETs without specifying an Alternate Input file.

NO PO

For this editing session there is no PRIMARY OUTPUT file; the user may not do PUTs without specifying an Alternate Output file.

READ FILE?

An attempt was made to read from a non-existent file or an illegal input device.

(INPUT)

The Editor response is in reference to an input attempt.

(OUTPUT)

The Editor response is in reference to an output attempt.

PI
PO
AI
AO

The Editor response occurred in reference to the Primary or Alternate Input or Output, as applicable.

NEW FILE

A new file was created.

(LPT1)

The Editor response occurred in reference to the line printer.

ASSIGN PROBLEM

The Editor was unable to assign a channel to a given device.

PI=NEW FILE?

An attempt was made to "EDIT INFILENAME OUTFILENAME" where INFILENAME and OUTFILENAME were not the same file and INFILENAME was non-existent.

EOF

An end-of-file was reached on input or output, or the end of workspace text was reached.

NO FILES SPECIFIED

The user initiated the Editor without specifying any primary files; for this editing session the user may not do GETs or PUTs without specifying an Alternate file.

ABORTED

A command line exceeded 128 characters and was rejected.

TRUNCATED

An INPUT line exceeded 128 characters and was truncated to the first 128 characters entered.

A SUBSTITUTE caused the line to exceed 128 characters and the line was truncated to 128 characters.

Appendix B

CONVERSION TABLES

HEXADECIMAL TO DECIMAL CONVERSION TABLE

HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0000	0	000	0	00	0	0	0
1000	4,096	100	256	10	16	1	1
2000	8,192	200	512	20	32	2	2
3000	12,288	300	768	30	48	3	3
4000	16,384	400	1,024	40	64	4	4
5000	20,480	500	1,280	50	80	5	5
6000	24,576	600	1,536	60	96	6	6
7000	28,672	700	1,792	70	112	7	7
8000	32,768	800	2,048	80	128	8	8
9000	36,864	900	2,304	90	144	9	9
A000	40,960	A00	2,560	A0	160	A	10
B000	45,056	B00	2,816	B0	176	B	11
C000	49,152	C00	3,072	C0	192	C	12
D000	53,248	D00	3,328	D0	208	D	13
E000	57,344	E00	3,584	E0	224	E	14
F000	61,440	F00	3,840	F0	240	F	15

HEX	F000 + B00 + 70 + 3 = FB73
DEC	61440 + 2816 + 112 + 3 = 64371

ASCII CODE CONVERSION TABLE

HEXADECIMAL

LEAST SIGNIFICANT CHARACTER	MOST SIGNIFICANT CHARACTER							
—	Ø	1	2	3	4	5	6	7
Ø	NUL	DLE	SP	Ø	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	¦
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	≈
F	SI	US	/	?	O	—	o	DEL

EXAMPLES

W = 57
H = 48
a = 61
t = 74
@ = 40
NUL = ØØ
DEL = 7F

Appendix C

NULL MODEM

DESCRIPTION

A null modem consists of two RS-232 standard female plugs hard-wired back-to-back with the following pin connections:

At both ends, pin 1 is grounded to the metal mainframe.

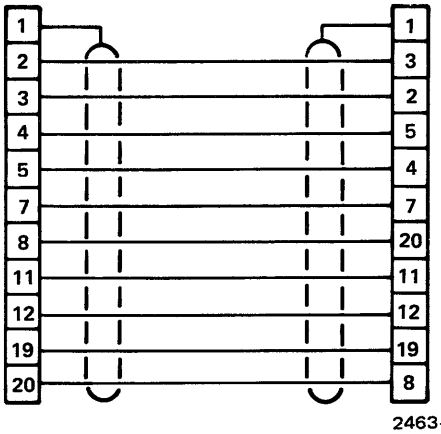
Pin 2 (data transmission path) at one end is connected to pin 3 (data reception path) at the other end.

Pin 4 (request-to-send line) at one end is connected to pin 5 (clear-to-send line) at the other end.

Pin 7 (signal ground) is connected to pin 7.

Pin 8 (external-computer-is-operational) at one end is connected to pin 20 (terminal-is-operational) at the other end.

Supervisory signal lines (pins 11, 12, and 19) from one end are connected to their corresponding pins at the other end (11 connected to 11, etc.).



Wiring Diagram for NULL Modem.

Appendix D

COMMAND INDEX

This appendix supplements the command index contained in Appendix D of the 8002 System User's Manual. Page references are to this supplement.