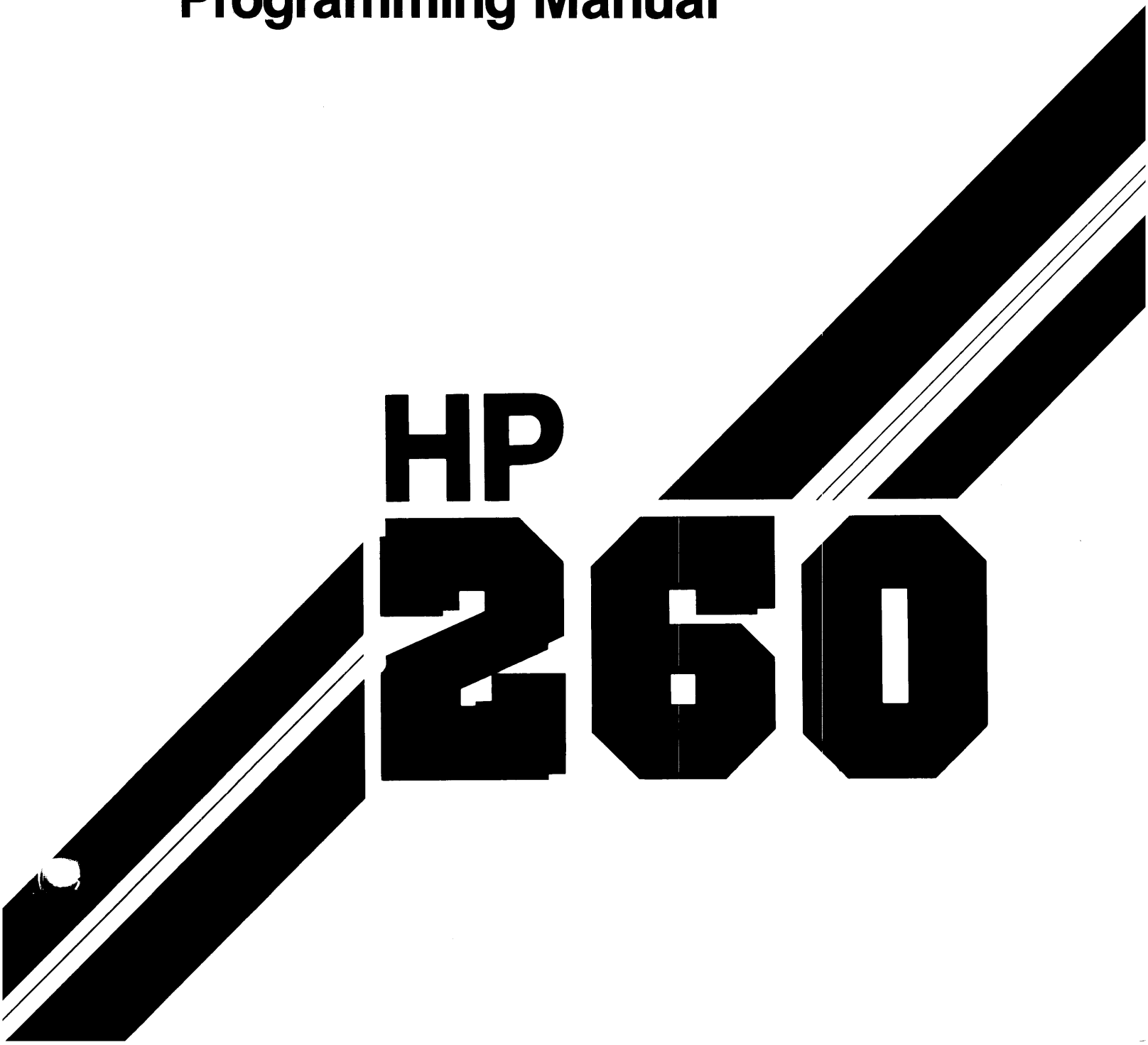
HEWLETT-PACKARD

TIO/260
Programming Manual



HP 260 Computer Systems

TIO

Programming Manual

HEWLETT PACKARD
HERRENBERGER STRASSE 130, D-7030 BOEBLINGEN

Part No. 45120-90006
E0986

Printed in Federal Republic of Germany 09/86

FEDERAL COMMUNICATION COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT
(for U.S.A. only)

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

CONTENTS

CONTENTS (continued)

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Feb 1985	B.07.00
Second Edition	Sep 1986	B.08.00

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the most recent version of each page in the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Effective Pages	Date
all	Sep 1986

INTRODUCTION

SECTION 1

The TIO DROM (Disc Resident Optional Module) and the Asynchronous Serial Interface (ASI) provide a means to connect up to ten RS-232 asynchronous devices to your computer system. The devices may be terminals, printers or HP 3000 computers.

NOTE

The TIO DROM is supplied with the operating systems of the following computer systems:

- all HP 250 Small Business Computer Systems
- all HP 260 Small Business Computer Systems

This manual describes operation of the TIO DROM for the most recent version operating system of the previously mentioned computer systems. (At this edition of the TIO Programming Manual, B.08 is the most recent version of the operating system for these computer systems.)

NOTE

Throughout this manual, the name "HP 250/260" is used to refer to the HP 250 and HP 260 Small Business Computer Systems; other computer systems are referenced by name or as simply "the remote system".

PREREQUISITE READING

This manual assumes that you are familiar with your computer system and that you have read and understood your system's BASIC Programming Manual. It is also assumed that you have read the manuals which accompany the remote printers and terminals, and that you understand the devices. Use of the HP 3000 requires a thorough understanding of that computer's operating system.

INTRODUCTION TO TIO

Using the CONFIG utility, each asynchronous port of your computer can be configured as any of the following:

- "None" - the port is not used.

- "Workstation" port - a port that is used for both input and output; the computer monitors the port for input (reading data from the ASI input buffer) and sends data to the workstation display (writing data to the ASI output buffer).

- "Printer" port - a port that is used only only for output; when the ASI output buffer for the port is available, a PRINT or PRINT USING statement is used to send output data.

- "Terminal" port - a port that is used for both input and output. The program reads data from the port's ASI input buffer. Then the remote device sends more data. The program can send output when the port's output buffer is empty. A terminal operates in half-duplex fashion; it is ready for either input or output at any one time. The program determines which one.

- "Computer" port - a port that is always enabled for input; an HP 3000 may transmit data at any time to your computer. Before your computer can output to a "Computer" port, the HP 3000 must first send your computer a signal enabling the port for output; the program on your computer detects the signal sent from the HP 3000 and then outputs the data to the "Computer" port.

- "General" port - see the TIO-II manual for the use of a general port with terminal input/output controlled applications.

Your computer's installation manual describes the hardware configuration necessary to connect ports to the various devices. (If your system is an HP 250, this information is located in the ASI Installation Note, part number 45120-90065.) Configuring the TIO software into the system is described in the section titled "System Configuration" in your computer's Utilities Manual.

An application program can be written to send output to and receive input from asynchronous devices. TIO statements allow the application program to determine the state of each asynchronous port. (These statements are described in the section of this manual titled "TIO Statements".) The program uses PRINTER IS, PRINT ALL IS or SYSTEM PRINTER IS (coupled with the device address) to direct output to the remote device. The device address is calculated from the port number to which the device is connected; for example, the device connected to port 1 has a device address of 11, the device connected to port 2 has a device address of 12, etc.

The TIO DROM keeps status information on each port. (For example, "Is the device connected or disconnected?" and "Is the input/output buffer full or empty?".) This information can be transferred to the application program through interrupts, by including TIO statements in the program.

Additionally, TIO provides statements for sending and receiving data to/from an HP 3000. Still other TIO statements control "modes" of communication.

WHAT'S IN THIS MANUAL

The next section of this manual describes the syntax and function of each statement provided by the TIO DROM; this section is titled "TIO Statements".

The section titled "Programming with TIO" will help you structure your application program; additionally, it provides many tips for programming with TIO as well as descriptions of programming approaches.

This manual's appendices contain helpful reference information, such as ASCII character codes, TIO error codes, and a condensed syntax reference. Additionally, the appendix titled "The LK3000 Utility" teaches you how to use the LK3000 Utility for connecting your computer to an HP 3000 and transfering data files between these computers.

TIO PORT FUNCTIONALITY

There are a maximum of 12 asynchronous ports on your HP 260.

- 5 ports on each of the two ASI boards
- 2 integrated serial ports on the processor board of the HP 260 Series 30 and Series 40 computers.

With the current revision of the Operating System, the integrated serial ports **are not** supported by the TIO DROM. Therefore, the term "asynchronous port", in this manual, refers to any of the ports on the ASI board(s) of your HP 250/260.

TIO STATEMENTS

SECTION 2

SYNTAX CONVENTIONS

The statements in this manual use the same syntax conventions as in the BASIC Programming Manual.

SHADING	All shaded keywords and characters must appear as shown.
italics	All parameters (user selectable values) are shown in italics
. . .	An ellipsis indicates that the previous parameter can be repeated.
[ ]	All parameters in brackets are optional. If there are brackets within brackets, the parameter within the inner bracket may only be specified if the parameter in the outer bracket is specified. Parameters may also be stacked in brackets. For example A or B or neither may be selected when the following is shown: [A / B]
{ }	One parameter must be selected from those stacked within braces. For example A or B or C must be selected when the following is shown: { A / B / C }

THE ON-CONDITION STATEMENTS

The TIO statements operate similar to the ON KEY # statement* , in that the program continues execution until a preset condition occurs. When the condition occurs, the program branches to a specified routine.

The syntax of the "ON" statements is as follows:

ON *condition* **#** *device address [,priority] branching statement*

The condition can be one of the following:

BREAK	When a break is received, the interrupt occurs.
CONNECT	When the remote device is connected, the interrupt occurs.

*ON KEY # is described in the Branching and Subroutines section of the BASIC Programming Manual.

DISCONNECT	When the remote device is disconnected, the interrupt occurs.
INPUT	When an input terminator (CR) is received, the interrupt occurs.
OUTPUT	When the output buffer is empty, the interrupt occurs.
TRIGGER	When a DC1 is received from the HP 3000, the interrupt occurs.

The device address is the port number plus 10; e.g., the device attached to port 3 has a device address of 13.

Before an ON-condition statement is executed, the program must have exclusive use of the device. This is done with the REQUEST statement which is described later in this section.

The priority parameter is used by the system to determine which interrupt to service first. The priorities range from 1 - the lowest, thru 15 - the highest.

A priority of 1 is assumed when the parameter is not used. The first interrupt condition to occur with a priority greater than the current execution priority will be processed.

The branching statement can be a GOTO, GOSUB or CALL statement. If the branch is a GOTO statement, the execution priority remains the same as before the interrupt occurred. If the branch is a GOSUB or CALL statement, the execution priority changes to the priority specified in the ON statement until the end of the subroutine or subprogram. It is then restored to the execution priority in effect before the interrupt occured.

GOTO	When an interrupt occurs with the sufficient priority, the program branches to the specified line and continues execution. The program cannot return to the line where the interrupt occurred. The GOTO statement uses less system overhead than the GOSUB or CALL statements use.
GOSUB	When an interrupt occurs, the program branches to the subroutine. After the subroutine has been executed, the program returns to the line where the interrupt occurred.
CALL	When an interrupt occurs, the program branches to the subprogram. After the subprogram has been executed, the program returns to the line where the interrupt occurred. Parameters cannot be passed.

Subprograms create a new environment during their execution. If an interrupt condition occurs during a subprogram and the action is another CALL with priority higher than the current execution priority, then the interrupt is serviced. If the action is a GOTO or GOSUB, the interrupt is not serviced until the subprogram exits with a SUBEXIT or SUBEND.

An interrupt generally occurs after execution of the current line. However, if an interrupt occurs during execution of a WAIT or INPUT statement, execution of the statement is suspended while the interrupt is serviced.

To prevent interrupts during critical portions of a program, use DISABLE statements to enclose the lines. Once the DISABLE statement is executed, no interrupts occur until the ENABLE statement is executed.

THE ON BREAK # STATEMENT

The terminals supported on the HP 250/260 have a key labeled BREAK. If this key is pressed, the ON BREAK # statement will detect it.

ON BREAK # *device address* (, *priority*) *branching statement*

If the break occurs when the port is in the output state, the remainder of the current output buffer is discarded, the break interrupt is activated and the port remains in the output state. The interrupt routine specified in the ON BREAK # statement should output a prompt to the remote terminal user to indicate the break has been serviced. The interrupt routine would typically enable input from the terminal through the ON INPUT # statement in order to determine the operation desired by the user.

If the break occurs when the port is in the input state, then the input state is cancelled and the port is set in the output state. The input buffer is discarded. The interrupt routine may then output a prompt as outlined above.

The ON BREAK # condition is cancelled by executing an OFF BREAK # statement.

OFF BREAK # device address

If the program has not established a break interrupt routine, or has cancelled the condition by execution of an OFF BREAK # statement, pressing the BREAK key on the remote terminal while the port is in output mode has no effect. Pressing the BREAK key while in the input mode, however, causes the current input line to be rejected (as if the remote terminal user pressed CONTROL X) and a new input line is begun.

Examples:

```
ON BREAK # Terminal,5 GOTO Break
OFF BREAK # Terminal
ON BREAK # 14 CALL Break
ON BREAK # Terminal+10,Priority GOSUB Break
```

THE ON CONNECT/DISCONNECT # STATEMENTS

The program can detect if a device has been connected or disconnected by use of the ON CONNECT # or ON DISCONNECT # statements.

ON CONNECT # device address (, priority) branching statement
ON DISCONNECT # device address (, priority) branching statement

Only one of these statements may be in effect at any time. The ON CONNECT # statement cancels execution of the ON DISCONNECT # statement. The ON DISCONNECT # statement cancels the ON CONNECT # statement. The ON DISCONNECT # statement should be executed at the beginning of program execution, or when the device is first addressed. If the port is actually disconnected, then the interrupt condition is satisfied immediately. The ON CONNECT # statement should then be executed in response to the disconnect interrupt. If the port is connected, then the interrupt condition is satisfied and the device can be used.

NOTE

These interrupts are serviced according to the setting of the ports on the ASI board. Refer to the appendix titled, "Setting ASI Port Functionality" for a description of how to select the functionality of the ports on your computer's ASI board.

The program may terminate detection of the connect/disconnect interrupt condition by use of the corresponding OFF CONNECT # or OFF DISCONNECT # statement:

```
OFF CONNECT # device address
OFF DISCONNECT # device address
```

Examples:

```
FOR Device=1 TO 5
        ON DISCONNECT #Device+10 CALL Disc
NEXT Device

ON CONNECT #12 GOTO Int_12

ON CONNECT #14,Priority GOSUB Connect

OFF CONNECT #Device

OFF DISCONNECT #14
```

THE ON INPUT # STATEMENT

The ON INPUT # statement explicitly enables input from a terminal and informs the system of the desired action to be taken when a complete input line is received.

ON INPUT # *device address* [,*priority*][*branching statement*]

The ON INPUT # statement remains in effect until input is received, an ON OUTPUT # statement addressing the same port is executed or an OFF INPUT # statement is executed.

If the branching statement is omitted, TIO assumes that an ON INPUT # statement specifying a branch was previously executed.

For example, assume the subroutine Txy is called whenever input is available. The first ON INPUT # statement would have a branching statement of GOSUB Txy. When input is received from the terminal an interrupt occurs and program execution continues in the Txy subroutine. Before the subroutine is ended with a RETURN, the ON INPUT # statement with no branch is executed. When input is received from the terminal, an interrupt occurs and the action statement in the previous ON INPUT # statement, GOSUB Txy, is executed.

As the terminal user enters a line of information, the ASI echoes each character as it is accepted and stores it in the input buffer associated with the port. The input buffer holds up to 255 characters. The user can backspace and retype one or more characters by means of the ASCII backspace character, CONTROL H, or the backspace key (not the editing keys) the user can also cancel the entire line and retype it from the beginning by means of the cancel character CONTROL X . The ASI updates the port input buffer accordingly so the HP 250/260 is not concerned with the editing operations.

The remote terminal user indicates the end of the entry by a carriage return. The ASI echoes the carriage return , terminates the input state of the remote terminal and notifies the HP 250/260 of the available input. TIO informs the program of the available input data through a software interrupt. The branching statement is then executed and the application program accepts the input line for processing.

The AREAD$ Function

The string function AREAD$ transfers data from the ASI input buffer to a string variable.

*variable$* **=AREAD$** (*device address*)

If the AREAD$ function is attempted when no terminal input line is present in the ASI input buffer, execution error 315, "NO INPUT AVAILABLE", results. The carriage return character is not transferred to the string variable.

When the program has completed processing the input, the program explicitly re-enables further terminal input by means of the ON INPUT # statement.

Examples:

```
        REQUEST 12,Wait
        IF Wait=1 THEN Queue
        ON INPUT #12,3 GOSUB In_12
        •
        •
In_12:  Next_line$=AREAD$(12)
        •
        •
        RELEASE 12
        RETURN
Queue:  !

        REQUEST 12
        ON INPUT #12 GOSUB Input
        •
        •
Input:  Next_line$=AREAD$(12)
        IF Next_line$="EXIT" THEN 550
        •
        •
530     ON INPUT #12
540     RETURN
550     RELEASE 12
560     RETURN
```

Cancelling Input

Unusual conditions may arise after a program has enabled input from a remote terminal by means of an ON INPUT # statement, but before the terminal user has supplied a carriage return. For example, the remote user needs to be informed of some pending event (such as the computer is about to be turned off). These conditions require the capability for the program to terminate the input state in order to perform output.

OFF INPUT #*device address*

Once the OFF INPUT # statement is executed, any input characters already received by the ASI will be discarded. Even if the terminal user has supplied a carriage return, if the input available interrupt is being held pending by the operating system due to execution priority considerations, the entire input line is discarded. The execution of an OFF INPUT # statement addressing a port not currently in the input state results in no operation and is not an error.

Examples:

```
OFF INPUT #12

OFF INPUT #Device
```

Echoing Input

As each character is entered from a terminal, the ASI places it in the port's input buffer and then echoes the character back to the terminal. Some operations , such as data entry from a CRT form or data cartridge, require suppression of input character echoing.

ECHO OFF #*device address*

ECHO ON #*device address*

The default mode, established by successful execution of a REQUEST statement, is input echoing on.

If input echoing is disabled, then the ASI does not perform the normal character and line editing functions (backspace and cancel line). Any CONTROL H or CONTROL X characters received with echo disabled are stored in the input buffer.

Examples:

```
IF Flag=1 THEN ECHO OFF #Terminal
ECHO ON #13
```

THE ON OUTPUT # STATEMENT

The program can direct output to a terminal or remote printer using the PRINTER IS, SYSTEM PRINTER IS and PRINT ALL IS statements.

When the program is communicating interactively with more than one remote device, the programmer must be concerned with output buffer overflow. The ASI output buffer holds up to 255 data bytes, including carriage return and line feeds. If the output buffer fills, then the task is blocked until the buffer becomes empty. Unless the program logic considers this possibility, the generation of output for one device may temporarily prevent processing of input and generation of output to all other devices.

The ON OUTPUT # statement detects when the output buffer is empty.

ON OUTPUT #device address[,priority][branching statement]

The ON OUTPUT # statement remains in effect until an ON INPUT # statement is executed, another ON OUTPUT # statement is executed or an OFF OUTPUT # statement is executed.

The action routine would normally contain one of the print statements which directs output to a particular device. Then any statement which causes output (PRINT, PRINT USING, CAT, etc.) can be used. If the branching statement is omitted, TIO assumes that an ON OUTPUT # statement specifying a branch was executed previously.

For example, assume the main program executes an ON OUTPUT # statement with a branch of GOSUB Txy. When the ASI output buffer for the port is empty, an interrupt occurs and program execution continues in the Txy subroutine. Before the subroutine is ended with a RETURN, the ON OUTPUT # statement with no action statement is executed. When the ASI output buffer is empty, an interrupt occurs and the branch in the previous ON OUTPUT # statement, GOSUB Txy, is executed.

Here are two more examples:

```
    ON OUTPUT #Printer ,4 CALL Out_print
    •
    •
SUB Out_print
    Printer=11+(CURKEY-25)/3
    Wait=1
    REQUEST Printer ,Wait
    IF Wait=1 THEN GOSUB Queve
    PRINTER IS Printer
    PRINT "Next output"
    •
    •
SUBEND

    ON OUTPUT #2 GOTO Output
    •
    •
Output  PRINTER IS 12
    •
    •
    ON OUTPUT #2
    RETURN
```

HP 3000 INPUT / OUTPUT

TIO allows a program to communicate with an HP 3000 computer system. A small utility program , called LK3000, which emulates an interactive terminal to the HP 3000 exists on the system disc. An HP 3000 to HP 250/260 file transfer utility also exists. Refer to the appendix titled, "The LK 3000 Utility" for details.

In this section, "input" refers to data received by the HP 250/260 and "output" refers to data sent by the HP 250/260.

Input

The program does not explicitly enable input since the HP 3000 may transmit data at any time. The ON INPUT # statement establishes an interrupt routine which is activated whenever an input line from the HP 3000 is available in the ASI input buffer. The interrupt routine reads the input line by means of the AREAD$ function.

The ASI input buffer for an HP 3000 port is capable of stacking input lines, since the HP 3000 is always enabled for input and the program may not be able to accept and process one input line before the next one arrives. An ENQ / ACK protocol is used to prevent the ASI input buffer from overflowing. (The HP 3000 sends an ENQ [enquiry] after every 80 characters. The HP 250/260 sends back an ACK [acknowledgement] if there is room for at least an additional 80 characters.)

The convention of ENQ / ACK protocol limits the maximum useful length of each input line to 165 bytes. By disabling the ENQ / ACK protocol (possibly by logging on the HP 3000 with TERM = 9 for example), the full 255 bytes are available (with some risk of input buffer overflow).

Since HP 3000 input is always enabled, the ON INPUT # statement need not be executed following processing of each line.

The HP 3000 terminates input with either a carriage return or ASCII DC1 (device control one). The carriage return (if any) is included as the last character of the string returned by the AREAD$ function. The DC1 terminator is not included in the input string.

Ready for Output

The HP 3000 must explicitly enable the port for output from the HP 250/260. To do this, the HP 3000 terminates its data transmission with a DC1. The ON TRIGGER # statement detects the ready state.

ON TRIGGER #*device address*[,*priority*]*branching statement*

Executing the ON TRIGGER # statement informs the system of the action to be taken upon receipt of the DC1 from the HP 3000. The interrupt is activated only after the program has accepted all the data in the ASI input buffer which was received from the HP 3000 prior to the DC1.

TIO sets the state of the port to output when the interrupt is activated. The ON TRIGGER # branching statement is then executed. The ON TRIGGER # statement remains in effect and does not need to be re-executed after each interrupt.

The interrupt routine directs output to the HP 3000 port that was earlier specified by the PRINTER IS statement. Only one line of output may be sent to the HP 3000. Output is sent using a PRINT or PRINT USING statement. (ON OUTPUT # is not used.) Once a carriage return has been transmitted, TIO terminates the output state of the port.

Only the output state of the port is affected by the ON TRIGGER # and an interrupt routine. The port is always enabled for input.

Example:

```
1000    ON TRIGGER #12 GOTO Out12
1001 !
2940 Out12:  ON ERROR GOTO Out13
2950    FOR I=1 TO LEN(C$)   !Send command string to computer
2960     PRINT C$[I;1]       !character-by-character.
2970     WAIT 24000/Baud(Baud) !Delay to allow character turnaround
2980    NEXT I
2990    PRINT Cr$;           !terminate the command string
3000    OFF ERROR
3010    IF Debug AND (State<>1) THEN LDISP CHR$ (137)&"COMMAND"&
FNState$&CHR$(128)&" "&C$
3020    Command=LEN(C$)      !Flag command sent---the command
3030    ENABLE               !string will be echoed and must be
3040    WAIT                 !ignored when received.
3050 !
```

Block Mode

Certain applications, such as file transfers from the HP 250/260, require the link from the HP 250/260 to the HP 3000 to use the full output data rate. TIO provides a means to simulate the Block Mode Transfer feature of Hewlett-Packard terminals. This mode ensures reliability of data transfers at the full data rate.

BLOCK MODE ON #*device address*

BLOCK MODE OFF #*device address*

After execution of the BLOCK MODE ON # statement, when TIO receives a DC1 from the HP 3000, it sends a DC2 back. When the HP 3000 is ready to receive data in the Block mode, it again sends a DC1. TIO then activates the ON TRIGGER # interrupt.

When Block mode is enabled, the output data is not echoed back to the HP 250/260.

Successful execution of a REQUEST statement (to the HP 3000) sets Block mode off.

Example:

```
REQUEST HP3000
BLOCK MODE ON #HP3000
ON TRIGGER #HP3000,5 GOSUB File_trans
•
•
BLOCK MODE OFF #HP3000
```

One-Character Output

The HP 250/260 can send certain one-character output transmissions to the HP 3000 when the computer ready condition has not been established. For example, many HP 3000 utility programs respond to CONTROL Y by terminating the current operation. The SEND statement sends the transmission.

SEND #*device address* , *character code*

The character code is a numeric expression which represents the character. ASCII character codes are listed in Appendix A of this manual. The character code for CONTROL Y is 25.

If the state of the port has been enabled for output, the output state is terminated after the character code is sent.

Break Output

The program can break data transfer to the HP 3000 with the SEND BREAK # statement.

SEND BREAK #*device address*

If the state of the port was enabled for output, the output state is terminated.

Examples:

```
SEND BREAK #HP3000

IF Flag=1 THEN SEND BREAK #15
```

BASIC STATEMENTS USED WITH TIO

Four BASIC statements are used frequently with TIO. The statements are described fully in the BASIC Programming Manual. They are briefly described here for your convenience.

REQUEST / RELEASE

Before any ON condition statement is executed, the program must have exclusive access to the required device.

Successful execution of the REQUEST statement gives the program exclusive use of a device. When the program is finished with the device, the RELEASE statement is used.

Successful execution of a REQUEST statement addressing a terminal causes TIO to implicitly execute the OFF INPUT # and ECHO ON # conditioning statements. For an HP 3000, TIO executes a BLOCK MODE OFF #.

REQUEST *device address*[, *wait option*]

If the wait option is omitted and the device is already reserved by another program, an error 131 results.

If the wait option is present, its initial value specifies whether or not the program is to be delayed if the device is already reserved. The wait option must be a variable.

Wait = 0 the requesting program may be delayed.
Wait ≠ 0 the requesting program may not be delayed.

If the initial value is 0, then TIO resets the value to indicate the results of the request attempt.

Wait = 0 exclusive access granted.
Wait = 1 device already reserved.

Exclusive access is relinquished by the RELEASE statement.

RELEASE *device address*

DISABLE / ENABLE

The DISABLE statement inhibits all interrupts (including ON KEY # interrupts); interrupts are still recorded. When the ENABLE statement is given, interrupts are serviced according to their priority. If two interrupts have the same priority, they are serviced according to their port number (port 15, then 14, 13, etc.).

DISABLE

ENABLE

The CURKEY Function

CURKEY is a function which returns a number indicating the source of an ON condition interrupt.

CURKEY *numeric variable*

The values CURKEY returns are shown in the following table:

Value	Condition
0	No interrupts have occurred
1-24	Softkeys 1-24
25-27	Port 1, device address 11
28-30	Port 2, device address 12
31-33	Port 3, device address 13
34-36	Port 4, device address 14
37-39	Port 5, device address 15
40-42	Port 6, device address 16
43-45	Port 7, device address 17
46-48	Port 8, device address 18
49-51	Port 9, device address 19
52-54	Port 10, device address 20

Three values are allocated for each port. An ON INPUT # or ON OUTPUT # interrupt returns the first value (25 for port 1, 28 for port 2, etc.). The second value is returned for an ON BREAK # interrupt. ON CONNECT # and ON DISCONNECT # interrupts cause the third value to be returned.

PROGRAMMING WITH TIO

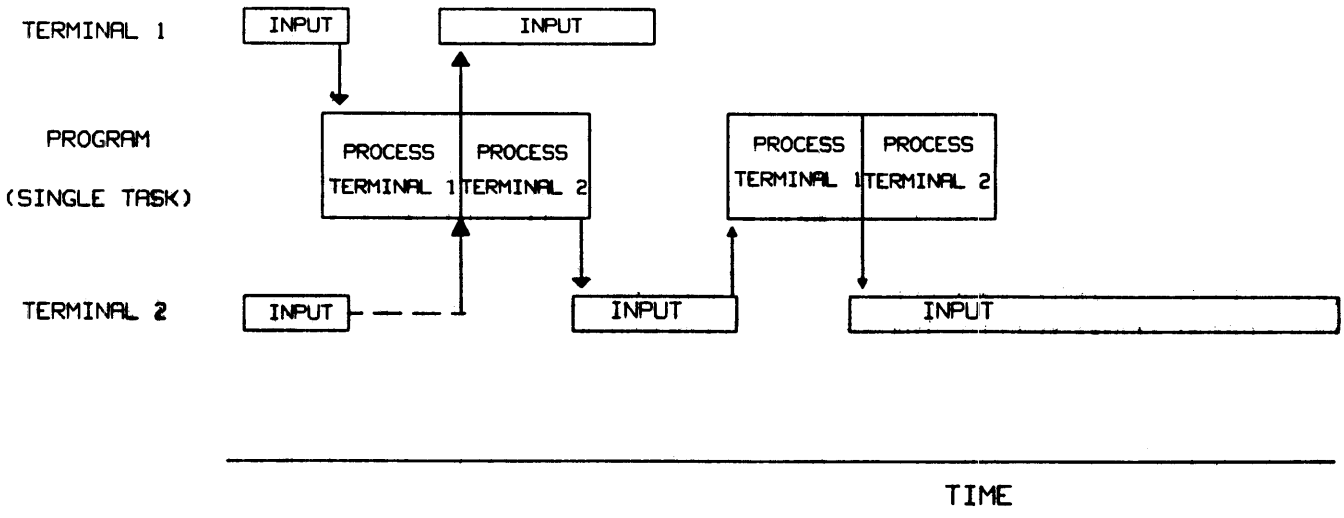
SECTION 3

PROGRAMMING OVERVIEW

TIO application programs can control remote terminals which are dedicated to an application. Since the terminal is not used as a workstation, the user is not confused by system messages or error codes. The application program can be tailored to the terminal user. Passwords and security features embedded in the programs can control access to sensitive information.

The ON-condition statements overlap I/O and processing. For example, instead of waiting for a terminal to respond to a prompt, the program does other processing until a carriage return is received from the terminal. Depending upon relative priorities, the program can either accept the input immediately or finish the current processing before accepting the input. If the input is not received before processing is finished, the program uses the WAIT statement to explicitly wait for the input.

Overlapping I/O and processing of other tasks is particularly useful in applications where several terminals are serviced by a single program. For example:



PROGRAMMING TIPS

Input / Output States

The three types of I/O ports: terminal, computer, and printer, can be set in either an input or an output state. The printer is obviously an output device, thus the port is always in the output state. The computer port is always ready for input from the remote computer. When the remote computer sends a DC1 to the HP 250/260, the the port will accept output, but it is still ready to accept input. Similarly when receiving input, one-character output (such as CONTROL-Y), or a break can be output.

A terminal port can be in only one state at a time. The program specifies which one. The state of a terminal is output at power-up or after CONTROL HALT is pressed. To enable input from a terminal, the ON INPUT # statement must be executed.

The change from output state to input state is never automatic. To re-enable output, successful execution of an OFF INPUT # or REQUEST statement is required. The output state is automatically enabled after an AREAD$ function and after a break is detected.

Branching Statements

The action and consequences of each of the branching statements should be understood well.

GOTO - When the interrupt condition is satisfied, execution branches to the line specified in the GOTO statement.

The execution priority is not changed.

```
10      REQUEST 11          Execution priority = 0
20      PRINTER IS 11
30      PRINT "ENTER YOUR NAME"
40      ON INPUT #11 GOTO 60
50      WAIT
60      NAME$=AREAD$(11)      Execution priority = 0
70      .
80      .
```

The GOTO statement is not recognized in other (subprogram) environments.

```
10      REQUEST 11
22      ON INPUT #11, 5 GOTO 40
30      CALL X
40      A$=AREAD$(11)
 •
 •
100     SUB X      }
110     WAIT       }   The interrupt can never occur in this
120     SUBEXIT    }   environment (X) because the branching
                       statement is not defined.
```

There is no "return" with a GOTO statement.

```
10      REQUEST 11
20      ON INPUT #11 ,2 GOTO X
 •
 •
400     WAIT
410  X:     A$=AREAD$(11)
```

If the interrupt occurs between line 20 and line 400, the program will branch out of some processing. It is impossible to return to complete the processing because the system does not remember where it was when the interrupt occurred.

GOSUB - Program execution branches to the subroutine specified when the interrupt condition is satisfied. When a RETURN is encountered, execution continues at the point the interrupt occurred.

The execution priority is raised to the value given in the ON-condition statement.

```
10      REQUEST 11                Execution priority = 0
20      ON INPUT #11 ,3 GOSUB X
30      FOR I=1 TO 1E99
40         DISP  I ;
50      NEXT  I
60      END
100  X:    A$=AREAD$(11)          Execution priority = 3
 •
 •
150     RETURN
```

The GOSUB statement is not recognized in other (subprogram) environments. This is the same as the GOTO statement.

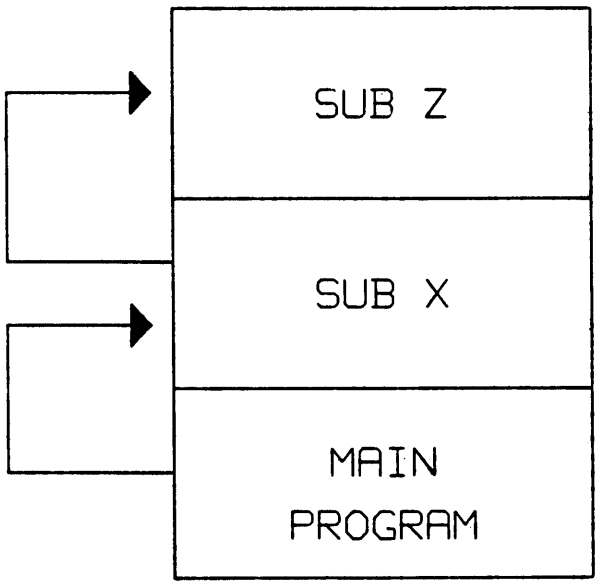
Execution resumes at the point of interruption after the subroutine has completed execution. In the example program above, if the interrupt occured when line 40 was executing, execution would resume at line 50 after the subroutine was done.

CALL - Program execution in the current environment is suspended when the interrupt condition is satisfied. A new environment is created and remains in effect until a SUBEXIT or SUBEND is encountered or another CALL statement is executed.

The execution priority is raised to the value given in the ON-condition statement. This is the same as the GOSUB statement.

The CALL statement is recognized in all successive environments including the one containing the statement.

```
10      REQUEST 11
20      CALL  X
 :
100     SUB  X
110     ON INPUT #11 CALL  Y
120     CALL  Z
 :
200     SUB  Z
 :
```



This program is a good example of how NOT to program with ON-condition statements. When subprogram X is called, the ON INPUT # statement is executed. The interrupt is defined in subprogram X and in subprogram Z. When both X and Z are exited, however, the interrupt is no longer defined.

NO BRANCHING STATEMENT - Informs the system of the programmer's intent to re-establish an interrupt condition in a previous environment. This serves to turn on the input or output interrupt. It is useful when switching environments or when changing a port's input/output state.

```
10      REQUEST 12
20      ON INPUT #12 CALL  X
30      WAIT

100     SUB X
110     A$=AREAD$(12)

170     ON INPUT #12
180     SUBEXIT
```

Using ON OUTPUT #, OFF OUTPUT # or OFF INPUT # clears the branching statement of the ON INPUT # statement which was executed in the same environment. Therefore, an ON INPUT # with no branching statement will subsequently be ineffective.

```
10      REQUEST 12
20      ON INPUT # 12 CALL X
30      CALL Y

100     SUB X
110     OFF INPUT #12
120     PRINTER IS 12
130     PRINT "..."
180     ON INPUT #12
190     SUBEXIT
```

The OFF INPUT #12 and ON INPUT #12 statements are executed in a different environment than the initial ON INPUT #12 statement.

PROGRAMMING APPROACHES

Once TIO statements and the concepts are understood, you are ready to begin programming. The most useful program approaches are introduced next.

Straight Line Approach

The following program communicates with one terminal. It demonstrates the ease of programming for one remote device. Later, the same program will be expanded for multiple terminals.

```
5       OPTION BASE 1
10      DIM A$[254] ,B$[254]
20      Port=11
30      REQUEST Port
40      PRINTER IS Port ,WIDTH(-1)
50      PRINT "Please enter your name:";
60      ON INPUT #Port GOTO Ini
70      WAIT
80    Ini:  A$=AREAD$(Port)
90      PRINT "What's your street address"&A$&"?"
100     ON INPUT #Port GOTO In2
110     WAIT
120    In2: B$=AREAD$(Port)
```

Input data from the remote terminal; equivalent to LINPUT A$ directed to the main console.

Modular Approach

The modular approach to programming is useful when input from the terminal will determine which task is to be done.

In the following example, the program accepts a command from the terminal and the FNInterp function determines the task (X1, X2, ...) to be done.

```
5       OPTION BASE 1
10      DIM Commd$[254]
20      Port=11
30      REQUEST Port
40      PRINTER IS Port ,WIDTH(-1)   !This is the default width
for all TIO devices
50      PRINT "Please enter a command";LIN(1);":";
60      ON INPUT #Port GOSUB Service
70      WAIT
80      END
100   Service: Commd$=AREAD$(Port)
110     ON FNInterp(Commd$)+1 GOTO 120;140;160
120     PRINT "ERROR: COMMAND NOT RECOGNIZED."
130     GOTO 200
140     CALL X1(Commd$)
150     GOTO 200
160     CALL X2(Commd$)
170     GOTO 200
 •
 •
200     PRINT ";";
210     ON INPUT #Port
220     RETURN
```

Array Addressing Mode

Expanding a program from accepting input from one terminal to accepting input from several terminals can be accomplished with little difficulty if the initial program was designed properly.

In the following example, the initial program is shown in the straight line approach example:

```
10      OPTION BASE 1
20      DIM A$(5)[254], B$(5)[254]
30      DISABLE
40      FOR Port=11 TO 20
50         REQUEST Port
60         PRINTER IS Port ,WIDTH(-1)
70         PRINT "Please enter your name:";
80         ON INPUT #Port GOTO Ini
90      NEXT Port
100     ENABLE
110     WAIT
200   Ini:  DISABLE
210     Port=(CURKEY-25)/3+11   !CALCULATE PORT NUMBER
220     A$(Port-10)=AREAD$(Port)
230     PRINTER IS Port,WIDTH(-1)
240     PRINT "What's your street address "&A$"?"
250     ON INPUT #Port GOTO In2
260     ENABLE
270     WAIT
300   In2: DISABLE
310     Port=(CURKEY-25)/3+11
320     B$(Port-10)=AREAD$(Port)
```

The DISABLE and ENABLE statements are used to protect critical sections of code from GOTO interrupts.

This program can communicate with five terminals because the system always knows where to go when it gets an input line from a terminal. Having the system keep track of program state flow in this manner is called an **Implicit State Machine**. This is further discussed later in this chapter.

Executive Mode

In the following example the Service subroutine is used as an input/output traffic manager. It is referred to as an **Executive Routine**.

```
5       OPTION BASE 1
10      DIM A$[254], B$[254], Input$[254]
20      DIM Buff$(5)[750]
25      State=1
30         DISABLE
40         FOR Port=11 TO 20
50           PACK USING P1;Buff$(Port-10)
60           REQUEST Port
70           PRINTER IS Port
75           PRINT "Please enter your name:";
80           ON INPUT #Port GOSUB Service
90         NEXT Port
100        ENABLE
110        WAIT
120  P1:     PACKFMT State, A$,B$
130   !
200  Service:     Port=(CURKEY-25)/3+11
210        Input$=AREAD$(Port)
220        UNPACK USING P1 ;Buff$(Port-10)
230        PRINTER IS Port
240        GOSUB X
250        PACK USING P1;Buff$(Port-10)
260        ON INPUT #Port
270        RETURN
280   !
300  X:     ON State GOTO S1,S2,S3
310  S1:     A$=Input$
320          PRINT "What's your street address "&A$&"?"
330          State=2
340          RETURN
350   !
360  S2:     B$=Input$
```

This program demonstrates how PACK and UNPACK can be used to swap variables into and out of a common area. This program also illustrates how to use the **Explicit State Machine** approach.

STRUCTURED PROGRAMMING

When the application program addresses one remote device with one task, you may only need to add a few statements to the current non-TIO program to drive that device. For example, assume a report is written at the end of the working day. Instead of outputting the report to the local (default) printer, the report is now to go to a remote printer. If this is the only remote I/O function the program performs, the only

modifications needed are changing the device address in the PRINTER IS statements in the appropriate places.

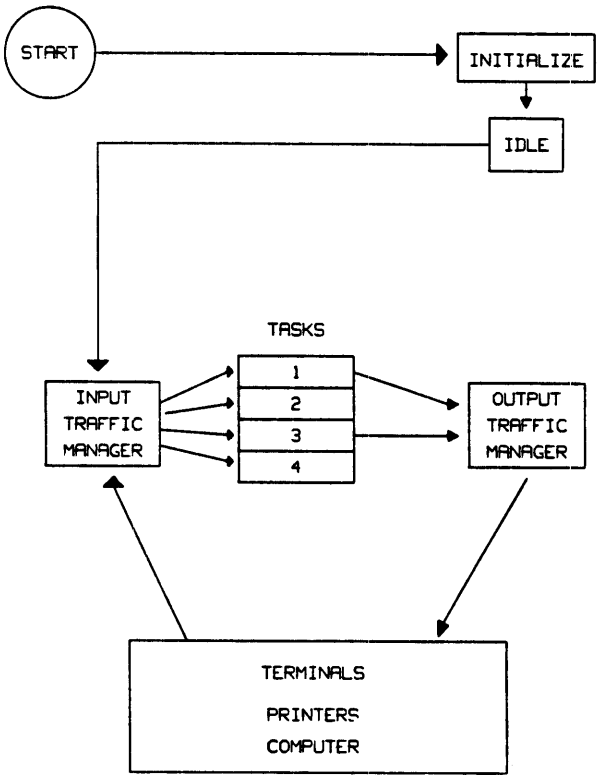
The creation of a more complex TIO application program, however, usually consists of many tasks which may be executing concurrently. In the TIO environment, each remote device may be associated with a task which controls the processing and input/output associated with the others. If interaction is desired among tasks, however, you must ensure that it happens efficiently.

The multi-tasking environment is more complex than that of sequential programing for one task. The problems demand a highly organized and structured approach. Without such an approach, you may have a program containing persistent (but unrepeatable, under debugging conditions) interference problems among the tasks.

Structured programming is concerned with improving the programming process through better program organization. Structured programming techniques, such as constructive use of subroutines and subprograms, ensure that a program is understandable, easily modified and documented, and easier to debug.

Basic Structural Flow

The following diagram can be used for many TIO applications.



Structured Programming Diagram

Notes on the above Diagram

Initialize - the logic flow begins with the devices being initialized. This can include logging on to the HP 3000, testing whether a printer is connected, or sending prompts to the terminals.

Wait - once the initialization is complete, the program waits for input from one or more of the devices.

Input Traffic Manager - input goes to the Input Traffic Manager routine which initiates a task.

Tasks - each task is processed according to its priority.

Output Traffic Manager - if output is required, the Output Traffic Manager initiates and completes it to the appropriate device.

Terminals, Printers, Computer - the program waits in an idle state until the next input from the devices or until one of the traffic managers begins the next task.

Example Program

The program used to demonstrate the Executive Mode of programming was designed according to the preceding structural flow diagram.

```
5     OPTION BASE 1
10    DIM A$[254], B$[254], Input$[254]
20    DIM Buff$(5)[750]
25    State=1
30       DISABLE
40       FOR Port=11 TO 20
50         PACK USING P1;Buff$(Port-10)
60         REQUEST Port
70         PRINTER IS Port
75         PRINT "Please enter your name:";
80         ON INPUT #Port GOSUB Service
90       NEXT Port
100      ENABLE
110      WAIT
120   P1:    PACKFMT State,A$,B$
130    !
200   Service:   Port=(CURKEY-25)/3+11
210      Input$=AREAD$(Port)
220      UNPACK USING P1;Buff$(Port-10)
230      PRINTER IS Port
240      GOSUB X
250      PACK USING P1;Buff$(Port-10)
260      ON INPUT #Port
270      RETURN
280    !
300   X:    ON State GOTO S1,S2,S3
310   S1:    A$=Input$
```

```
320          PRINT "What's your street address "&A$&"?"
330          State=2
340          RETURN
350   !
360  S2:     B$=Input$
•
•
```

TRANSACTION DRIVEN APPLICATIONS

The design techniques described here are suitable for the creation of application programs driven by external events, such as remote terminals controlled by users. The application program and each user exchange data in an interactive fashion. The program usually supplies prompts to facilitate communication. The user might supply commands consisting of keywords and perhaps parameters to direct the program into specific operating modes and input data when requested by the current operating mode. In response to user commands (or perhaps by default), the program may display CRT forms to facilitate data entry and may generate and transmit reports either directly to the user's remote terminal or to some other output device. Finally, in response to user commands which cannot be fulfilled, the program generates messages which give the user the proper course of action.

A transaction consists of a logically complete interchange of prompts, commands, processing, input data and output reports. A transaction may be as simple as typing in a single command, in which case the transaction is just the action performed by that command. Transactions should be kept as simple as possible, otherwise the operational requirements (user training, program reliability, etc.) become extremely demanding.

Transactions may be categorized as follows:

- **Security and overhead operations** such as user sign on and sign off
- **Data retrieval operations** accessing a data base or a normal file in read-only mode. The objective of the data retrieval may be either quick "on-line" access to information or a printed report.
- **Batch data-entry operations** resulting in an intermediate or transaction file which is not the final end product of the application. The transaction file is later used as input in batch mode to a program which creates or updates the final end product (usually a data base).
- **Interactive data-entry operations** in which the data base is updated immediately, making the updated information available to other users as soon as the transaction is complete. If this technique is chosen, the parallel maintenance of a transaction file, as in batch data entry, should be strongly considered to allow backup and recovery from errors.

Generally, an application program offering interactive data entry transactions must ensure that multiple users and their associated tasks are protected from one another. In the worst case, the designer may have to prevent any other access to the data base while an update transaction is in progress. This includes data retrieval access in read-only mode since reporting of partially updated data may be unacceptable to the application.

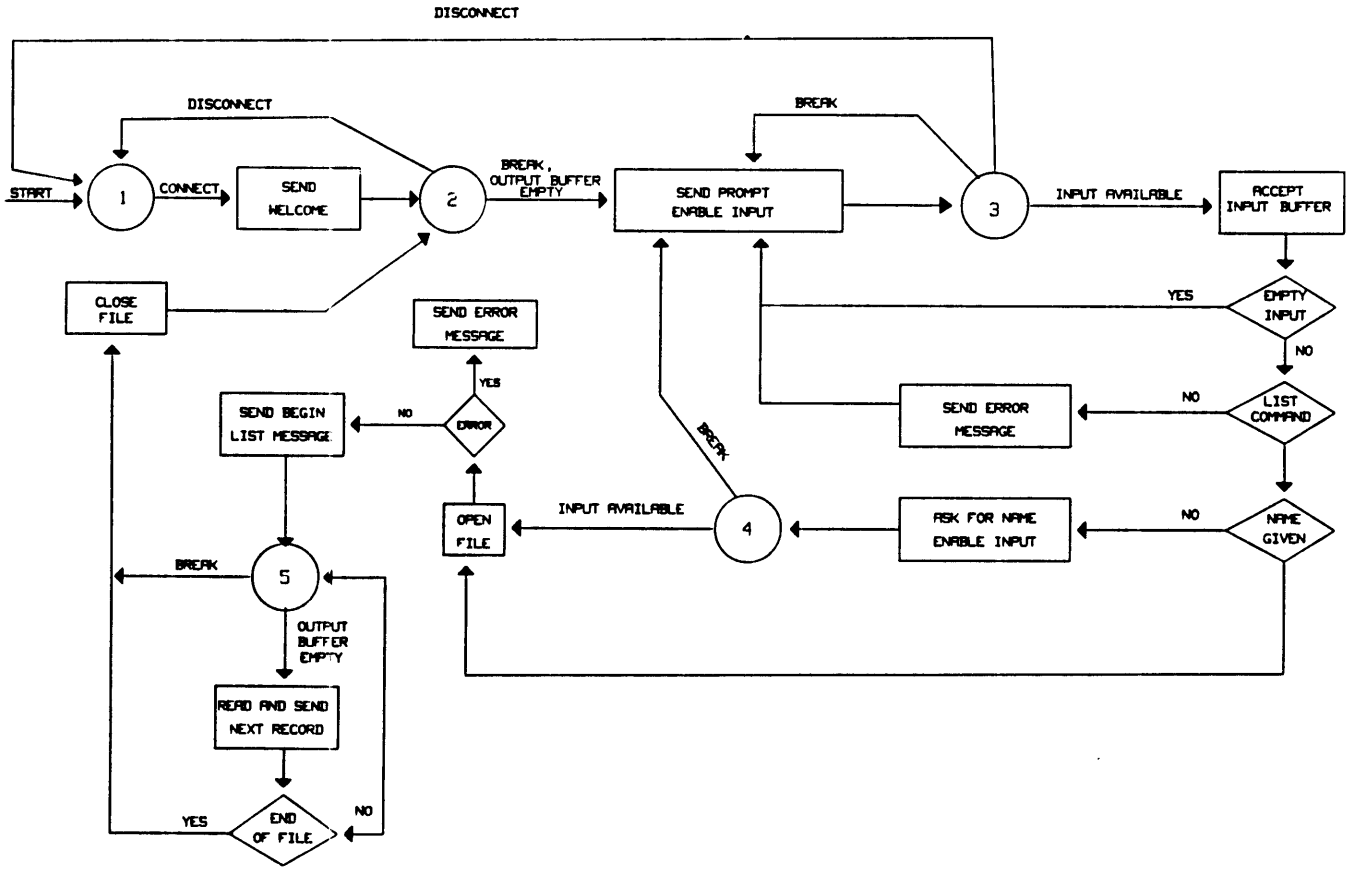
STATE MACHINE MODEL

The **State Machine Model** is a conceptual framework for designing a TIO application package. It is used to keep track of where you are in the program, and how you got there. For example, in the array addressing mode example, the variable *Port* is used to inform the system of which terminal supplied input and where the input is to be stored.

In the executive mode example, the variable *State* is given a value when a routine has completed processing. When input is received from a terminal, *State* is used to determine the next task to perform. This method is called the **Explicit State Machine Approach** because the variable is explicitly assigned a value.

STATE TRANSITION DIAGRAMS

The designer applies the state machine model to his problem by creating a state transition diagram which specifies all possible states of a remote device, all acceptable external events for each state, the transition (that is, the next state) for each external event, and a brief description of the processing taking place during each transition. An example diagram is shown next.



EXAMPLE: FILE LIST UTILITY

This program outputs a file listing to the requesting terminal. It uses the example state transition diagram shown on the preceding page.

The program is initiated at a workstation and accepts a list of port numbers; the program attempts to attach each specified port. The program sends each connected terminal an identifying welcome message and the prompt character ">". The program accepts one command from the remote terminal : "LIST" (typed in either upper or lower case); the name of the desired file may follow the command name (separated by at least one blank). If the file name is not given in the "LIST" command, the remote terminal user is asked to supply the file name. The program then opens the file (assumed to consist of ASCII string data) and outputs the file to the user's terminal. The user may terminate the file listing at any time by pressing the terminal's BREAK key. When the end of the file is detected, the program informs the user, sends the prompt character, and awaits the next command. The single-line function, FNT, defined in line 60, returns the port number (11 thru 20) of the currently interrupting terminal. The function is recommended for all TIO programs.

```
10    !  HP 260 FILE LIST UTILITY FOR REMOTE TERMINALS
20    !
60    DEF FNT=INT((CURKEY-25)/3)+11          State=1
70    INTEGER Port(1;10),I,W,Port
80    DIM C$[255],Bell$[1],Esc$[1],Home$[2],Clear$[2]
90    Bell$=CHR$(7)
100   Esc$=CHR$(27)
110   Home$=Esc$&"H"
120   Clear$=Esc$&"J"
130       DISP "HP 260 FILE LIST UTILITY FOR REMOTE TERMINALS"
140       DISP "Enter port numbers (each 1..10): ; ; ; ; ;"
150       CURSOR (XPOS-10)
160       INPUT " ";Port(*)
170       DISABLE
180       FOR I=1 TO 10
190           IF Port(I) THEN GOSUB Startport
200       NEXT I
210       ON HALT GOTO Exit
220       ENABLE
230       WAIT
240   !
250   Startport:   Port=Port(I)+10
260       IF  (Port>10) AND (port<21) THEN 300
270          DISP "PORT";Port(I);"IS OUT OF RANGE"
280          Port(I)=0
290          RETURN
300       W=0
310       REQUEST Port,W
320       IF NOT W THEN 360
330          DISP "CAN'T GET EXCLUSIVE ACCESS TO PORT";Port(I)
340          Port(I)=0
350       RETURN
360       ON CONNECT #Port GOTO C1
370       RETURN
```

```
380   !
390  Exit:    DISABLE
400       FOR I=0 TO 10
410          IF NOT Port(I) THEN RELEASE Port(I)+10
420       NEXT I
430       ENABLE
440       DISP "END OF PROGRAM"
450       STOP
460   !
470  C1:    DISABLE
480       PRINTER IS FNT
490       PRINT Home$&Clear$&"HP 260 FILE LISTER UTILITY"
500   !
510  C2:    ON DISCONNECT #FNT GOTO D1
520       ON OUTPUT #FNT GOTO O1
530       ON BREAK #FNT GOTO O1
540       ENABLE
550       WAIT
560   !
570  D1:    DISABLE
580       OFF OUTPUT #FNT
590       OFF BREAK #FNT
600       ON CONNECT #FNT GOTO C1
610       ENABLE
620       WAIT
630   !
640  O1:    DISABLE
650       PRINTER IS FNT
660   !
670  O2:    PRINT ">";
680       ON INPUT #FNT GOTO I1
690       ENABLE
700       WAIT
710   !
720  I1:    DISABLE
730       PRINTER IS FNT
740       C$=TRIM$(AREAD$(FNT))
750       IF NOT LEN(C$) THEN O2
760       IF POS (UPC$(C$)&" ","LIST ")=1 THEN List1
770       PRINT "Sorry, command not recognized."
780       GOTO O2
790   !
800  List1:   DISABLE
805       C$=TRIM$(C$[POS(UPC$(C$)&" ","LIST ")+5])
 810       IF LEN(C$) THEN List3
820       PRINT  "Enter name of file to list: ";
830       ON INPUT #FNT GOTO List2
840       ENABLE
850       WAIT
860   !
870  List2:    DISABLE
880       PRINTER IS FNT
890       C$=TRIM$(AREAD$(FNT))
900       IF NOT LEN(C$) THEN O2
```

```
910   !
920  List3:    ON ERROR GOTO List7
930       ASSIGN #FNT-10 TO C$
940       BUFFER #FNT-10
950       ON END #FNT-10 GOTO List5
960       OFF ERROR
970       PRINT "Begin listing of file: ";C$;LIN(1)
980       ON OUTPUT #FNT GOTO List4
990       ON BREAK #FNT GOTO List6
1000      ON DISCONNECT #FNT GOTO List8
1010      ENABLE
1020      WAIT
1030  !
1040  List4:    DISABLE
1050      PRINTER IS FNT
1060      ON ERROR GOTO List7
1070      READ #FNT-10;C$
1080      OFF ERROR
1090      PRINT C$
1100      ENABLE
1110      WAIT
1120  !
1130  List5:    ON ERROR GOTO 1150
1140      ASSIGN #FNT-10 TO *
1150      OFF ERROR
1160      PRINT LIN(1);"END OF FILE"
1170      GOTO C2
1180  !
1190  List6:    DISABLE
1200      PRINTER IS FNT
1210      ON ERROR GOTO 1230
1220      ASSIGN #FNT-10 TO *
1230      OFF ERROR
1240      PRINT LIN(1);Bell$&"FILE LIST TERMINATED BY BREAK"
1250      GOTO C2
1260  !
1270  List7:   ON ERROR GOTO 1290
1280      ASSIGN #FNT-10 TO *
1290      OFF ERROR
1300      PRINT Bell$&"ERROR IN ACCESSING FILE "&C$
1310      GOTO C2
1320  !
1330  List8:    DISABLE
1340      ON ERROR GOTO 1360
1350      ASSIGN #FNT-10 TO *
1360      OFF ERROR
1370      GOTO D1
```

This program implements one transaction type (the listing of a named file to the terminal). This is data retrieval transaction. The program supports five remote terminals as readily as one, with no interference among the terminals (other than possible delays waiting for access to the same disc device). Except for initialization and termination, the program utilizes no variables other than the string buffer C$. Even this buffer is shared by all attached terminals. This is an extreme example of the implicit state marker

technique; all state variable information associated with each terminal is maintained by the operating system and is invisible to the program. The program requires only five states to implement the specifications.

CONTROLLING YOUR APPLICATION

The definition and priorities of the softkeys on the workstation you use to control the terminal(s) used in your application can have a strong influence on the performance of the application.

Therefore you must be aware of the relative priorities of the softkeys and the operations in your application. With this knowledge you will avoid unexpected results when running your application.

For example, if you decide that pressing a softkey on the workstation should never be allowed to interrupt your application, you should set the priority of all of the softkeys to a value lower than the priorities specified in your application program. This is done using the ON KEY# statement (refer to the BASIC Programming Manual for the details of this statement).

ASCII CHARACTER CODES

APPENDIX

A

ASCII Character Codes

ASCII	EQUIVALENT FORMS	
Char.	Binary	Dec
NULL	00000000	0
SOH	00000001	1
STX	00000010	2
ETX	00000011	3
EOT	00000100	4
ENQ	00000101	5
ACK	00000110	6
BELL	00000111	7
BS	00001000	8
HT	00001001	9
LF	00001010	10
VT	00001011	11
FF	00001100	12
CR	00001101	13
SO	00001110	14
SI	00001111	15
DLE	00010000	16
DC_1	00010001	17
DC_2	00010010	18
DC_3	00010011	19
DC_4	00010100	20
NAK	00010101	21
SYNC	00010110	22
ETB	00010111	23
CAN	00011000	24
EM	00011001	25
SUB	00011010	26
ESC	00011011	27
FS	00011100	28
GS	00011101	29
RS	00011110	30
US	00011111	31

ASCII	EQUIVALENT FORMS	
Char.	Binary	Dec
space	00100000	32
!	00100001	33
"	00100010	34
#	00100011	35
$	00100100	36
%	00100101	37
&	00100110	38
'	00100111	39
(	00101000	40
)	00101001	41
*	00101010	42
+	00101011	43
,	00101100	44
-	00101101	45
.	00101110	46
/	00101111	47
0	00110000	48
1	00110001	49
2	00110010	50
3	00110011	51
4	00110100	52
5	00110101	53
6	00110110	54
7	00110111	55
8	00111000	56
9	00111001	57
:	00111010	58
;	00111011	59
<	00111100	60
=	00111101	61
>	00111110	62
?	00111111	63

ASCII Character Codes

ASCII Char.	EQUIVALENT FORMS	
	Binary	Dec
@	01000000	64
A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
E	01000101	69
F	01000110	70
G	01000111	71
H	01001000	72
I	01001001	73
J	01001010	74
K	01001011	75
L	01001100	76
M	01001101	77
N	01001110	78
O	01001111	79
P	01010000	80
Q	01010001	81
R	01010010	82
S	01010011	83
T	01010100	84
U	01010101	85
V	01010110	86
W	01010111	87
X	01011000	88
Y	01011001	89
Z	01011010	90
[	01011011	91
\	01011100	92
]	01011101	93
^	01011110	94
_	01011111	95

ASCII Char.	EQUIVALENT FORMS	
	Binary	Dec
`	01100000	96
a	01100001	97
b	01100010	98
c	01100011	99
d	01100100	100
e	01100101	101
f	01100110	102
g	01100111	103
h	01101000	104
i	01101001	105
j	01101010	106
k	01101011	107
l	01101100	108
m	01101101	109
n	01101110	110
o	01101111	111
p	01110000	112
q	01110001	113
r	01110010	114
s	01110011	115
t	01110100	116
u	01110101	117
v	01110110	118
w	01110111	119
x	01111000	120
y	01111001	121
z	01111010	122
{	01111011	123
|	01111100	124
}	01111101	125
~	01111110	126
DEL	01111111	127

SYNTAX REFERENCE

APPENDIX
B

variables$ = AREAD$ (device address)
Transfers data from the ASI input buffer to the string variable.

BLOCK MODE OFF # device address
Turns off block mode data transfer to the HP 3000.

BLOCK MODE ON # device address
Turns on block mode data transfer to the HP 3000.

ECHO OFF # device address
Turns off character echoing to a remote terminal.

ECHO ON # device address
Turns on character echoing to a remote terminal. The default echo mode is echo on.

OFF BREAK # device address
Cancels the execution of the ON BREAK # statement.

OFF CONNECT # device address
Cancels the execution of the ON CONNECT # statement.

OFF DISCONNECT # device address
Cancels the execution of the ON DISCONNECT # statement.

OFF INPUT # device address
Cancels the execution of the ON INPUT # statement.

ON BREAK # device address[, priority] branching statement
Causes an interrupt when the terminal BREAK key is pressed.

ON CONNECT # device address[, priority] branching statement
Causes an interrupt when the device is connected to the port.

ON DISCONNECT # device address[, priority] branching statement
Causes an interrupt when the device is disconnected from the port.

ON INPUT # device address[, priority][branching statement]
Causes an interrupt when a carriage return is sent to the ASI input buffer from a terminal. A carriage return or DC1 sent from an HP 3000 causes the interrupt.

ON OUTPUT # device address[, priority][branching statement]
Causes an interrupt when the ASI output buffer is empty.

ON TRIGGER # device address[, priority] branching statement
Causes an interrupt when the HP 3000 sends a DC1 as the data transmission

terminator and when the ASI input buffer is empty. This signals the program that the HP 3000 is ready to accept output.

SEND # device address , character code
Sends a one-character code to the HP 3000.

SEND BREAK # device address
Sends a break to the HP 3000.

THE LK 3000 UTILITY

APPENDIX C

The LK 3000 Utility is a run-only, BASIC-language program which allows you to:

- Use the HP 250/260 as a remote terminal in an HP 3000 computer system.
- Transfer ASCII data to or from the HP 3000.
- Transfer BASIC programs to or from the HP 3000.

The utility is provided with the operating system of the HP 250/260. The utility requires that the TIO DROM is configured into the operating system and that the HP 250/260 contains an Asynchronous Serial Interface board. The HP 3000 can be connected to the HP 250/260 either directly with cables, or indirectly via modems (see the sub-section titled, "Using Modems", later in this section). The example operations in this appendix assume a direct interface to the HP 3000.

LOG-ON PROCEDURE

To load LK 3000 and log on:

1. Load the HP 250/260 operating system and execute:

   RUN "LK3000"

2. The utility first requests the port number at which the HP 3000 is connected:

   RUN "LK3000"

   HP 250/3000 INTERACTIVE LINK, for use with MPE V.
   Enter port number (1 . . 10): _

   The HP 250 has five ASI ports and the HP 260 can have either five or ten ASI ports. Type in the port number you want and press (RETURN):

   RUN "LK3000"

   HP 260/3000 INTERACTIVE LINK
   Enter port number (1 . . 10): 5

3. The HP 3000 system prompt (:) indicates that you are connected and can log-on by entering your assigned name and account. For example:

```
: HELLO JOHN.ACCOUNTS
```

To ensure using the correct protocol, append ";Term=10" to the log-on sequence when files are to be transferred. For example:

```
: HELLO JOHN.ACCOUNTS ;TERM=10
```

The standard log-on message and system prompt indicate the computer is waiting for your next command:

```
: HELLO JOHN.ACCOUNTS
HP3000 / BLOGGS COMPANY A  THU, MAR 20, 1986, 10:30 AM
:_
```

You can now execute HP 3000 commands and call any available subsystems, as described in your HP 3000 Users Manual.

LOG-OFF PROCEDURE

To end your session with the HP 3000, simply enter **BYE** in response to the system prompt:

```
: BYE

CPU=6.  CONNECT=17. THU, MAR 20, 1986, 10:47 AM
```

This closes your account and disconnects you from the HP 3000. Press (HALT) to terminate the LK 3000 utility. The following message is then displayed:

```
END OF HP 260/3000 INTERACTIVE LINK
```

NOTE

Exiting the LK 3000 utility before logging off (e.g., by pressing (HALT) (or (BREAK)) or powering off) leaves your HP 3000 account open. To return to the point where you left off, execute **RUN "LK3000"** and enter the port number.

TERMINAL OPERATION

The LK 3000 utility allows interacting with the HP 3000 using the full HP 250/260 keyboard and display control keys. Press (RETURN) to transmit each command to the HP 3000.

After you have logged onto the HP 3000, the utility defines these softkeys to aid in terminal operation:

CONTROL Y - Sends a control character, which halts operation in the current subsystem and returns the subsystem prompt.

CARRIAGE RETURN - Enters a CR character, which returns the display cursor to the start of the current line.

DATA LINK BREAK - Sends a BREAK signal, a prolonged NULL, to interrupt computer operation and returns to the system prompt.

BAUD RATE - Selects the data transmission rate. The selected rate is displayed below the softkey label. This rate must match the rate set for the HP 260's port.

TRANSFER FROM 3000 - Initiates a procedure which transfers information from a source file in your HP 3000 account to a file created on the HP 250/260.

TRANSFER TO 3000 - Initiates a procedure which transfers the contents of an existing type DATA file to a source file created in your HP 3000 account.

HARD COPY - Selects the output device to be used for terminal output operations. The address of the currently-set device is shown below the softkey label. To select another available device, press the softkey until the device address is displayed. The default printer is usually configured at device address 0.

REMOVE KEY DISP - Removes the softkey definitions and and labels, providing more display work area. Press SFK 8 again to re-define the other softkeys.

Two additional SFKs are available which do not have a definition shown on the CRT.

SFK 17 - Allows you to type in an HP 250/260 command to be executed. Such commands as CAT, PURGE, MSI are useful. After the command has executed, the LK 3000 utility resumes processing.

SFK 20 - Toggles the debug mode to the LK 3000 utility. The current contents of the display are not affected by pressing this key. In debug mode, all commands sent to the HP 3000 are displayed with an indication of current program state (input or output).

TRANSFERRING FILES

The two special procedures within the LK 3000 utility, TRANSFER TO 3000 and TRANSFER FROM 3000, provide an easy means to transmit information to or from an HP 3000 account. Whether transmitting data or programs, the information must be in ASCII- coded format. This means only HP 250/260 type DATA files and HP 3000 source files (created using EDITOR/3000 or TDP) can be used at the originating end. Each special procedure automatically creates the appropriate file at the destination. The HP 3000 uses the FCOPY utility.

Each program stored in a type PROG file can easily be duplicated into a type DATA file before using the LK 3000 utility to transfer the program to the HP 3000. For example:

```
LOAD "SALES"   (load type PROG file)
SAVE "sales"   (save in type DATA file)
```

After BASIC program lines have been transferred from the HP 3000 to a type DATA file, they can be stored into a type PROG file:

```
GET "orders"      (get program into memory)
STORE "ORDERS"    (store in type PROG file)
PURGE "orders"    (erase type DATA file)
```

HP 3000 to HP 250/260 Data Transfer

To transfer the contents of an existing HP 3000 source file to the HP 250/260:

1. If you haven't done so already, log on as explained earlier.

2. When the system prompt appears, press the TRANSFER FROM 3000 softkey:

```
:
HP 3000 TO 260 FILE TRANSFER UTILITY
HP 3000 source file name: _
```

3. Enter the name of the source file containing data or BASIC program lines to be transferred to the HP 250/260. For example:

```
HP 3000 source file name: SFORM
HP 3000 file SFORM contains 55 records of 102 bytes each.
HP 260 destination file name: _
```

NOTE

The maximum record-size that is guaranteed to be transferred from the HP 3000 to the HP 250/260 is 155 bytes. Depending on the asynchronous configuration of your HP 3000, you might find it possible to transfer files with records larger than 155 bytes. The absolute maximum record-size that can be handled in an HP 3000 to HP 260 file transfer is 240 bytes.

If you have difficulty transferring files that have record-sizes greater than 155 bytes, you should change the structure of the file on the HP 3000 so that its records are 155 bytes long (or shorter).

4. Enter the name of a destination file, a type DATA file to be created on the HP 250/260 default drive:

```
HP 260 destination file name: SFORM1
START FILE TRANSFER
```

The utility creates the destination file and then each record from the source file. If the data already exists, LK 3000 asks if the file is to be purged then resaved. The final display is:

```
FILE TRANSFER COMPLETE

END OF PROGRAM
: _
```

If the utility cannot create the destination file, or if an error is encountered during data transfer, the utility exits the procedure and displays a message. See page C-8 for details.

HP 250/260 to HP 3000 Data Transfer

To transfer the contents of an existing type DATA file HP 3000:

1. Log on as explained earlier.

2. When the system prompt (:) appears, press the TRANSFER TO 3000 softkey:

```
:
HP 260 TO HP 3000 FILE TRANSFER UTILITY
HP 260 source file name:
```

3. Enter the name of a type DATA file containing data to be transferred to the HP 3000. For example:

```
HP 260 source file name: DATA
HP 260 source file DATA contains 22 records of 256 bytes each.
Enter estimated record count to override catalog value: 139
Enter actual maximum record size to override catalog value: 160
HP 3000 destination file name: DATA
```

Once the source file has been located, its size is displayed. If the file was SAVEd, its record size is always 256 bytes and record count is just sufficient to contain the program. However, there is a maximum of 160 bytes of data in a SAVEd file, and there can be no size-dependent problem with the transfer of SAVEd files.

On the HP 250/260, strings may cross record boundaries within HP 250/260 files. This is not true on the HP 3000. Therefore, LK 3000 gives you an opportunity to supply the record size and record count of the HP 3000 destination file. The record size must be the size of the longest string in the HP 250/260 data file. The record count must be the number of strings in the file. If exact values are not known, always supply overestimates for these values. Underestimates will result in lost data. If the size and count of the HP 250/260 file is the correct size and count for the HP 3000 file, press RETURN without entering new values.

NOTE

The maximum record size that can be transferred from the HP 250/260 to the HP 3000 is 255 bytes.

4. Enter the name of the destination file, which must be a new file (the destination file cannot be a file that already exists in your HP 3000 account):

```
HP 3000 destination file name: PAYROL
START FILE TRANSFER
```

The utility creates the new source file and transfers each record from the HP 250/260 DATA file. The final display is:

```
FILE TRANSFER COMPLETE

END OF PROGRAM
: _
```

Terminating File Transfers

If you decide not to transfer a file, whenever a file name is asked for press RETURN without giving a file name. This terminates the file transfer.

If the transfer is already in progress, press [HALT] to terminate the transfer. Press the CARRIAGE RETURN softkey repeatedly until the FCOPY prompt ">" appears. Then type EXIT [RETURN] to terminate the FCOPY utility.

Data Transfer Errors

If the subprogram encounters an error while creating a file or transferring data, it automatically exits the procedure and displays a message. For example:

```
:
HP 3000 TO 260 FILE TRANSFER UTILITY
HP 3000 source file name: SFORM
HP 3000 file SFORM contains 55 records of 102 bytes each.
HP 260 destination file name: SYSTEM
ERROR IN CREATING FILE
END OF FILE TRANSFER
: _
```

If you abort the transfer operation (via power off), you must first **RUN "LK3000"**, enter the port number and abort operation in the HP 3000's FILE COPIER subsystem. For example:

```
:
HP 3000 TO 260 FILE TRANSFER UTILITY
HP 3000 source file name: SFORM
HP 3000 file SFORM contains 55 records of 102 bytes each.
HP 260 destination file name: SFORM1
RECORD 39 TRANSFERRED              HALT pressed during file transfer.

END HP 260/3000 INTERACTIVE LINK

RUN "LK3000"

HP260/3000 INTERACTIVE LINK
Enter port number (1 . . 10): 5        re-establish link

EXPECTED "YES" or "NO".   (CIWARN 990)
ABORT? YES       respond to prompt to abort FILE COPIER subsystem

PROGRAM ABORTED PER USER REQUEST.   (CIERR 989)

HP32212A.3.20 FILE COPIER (C) HEWLETT-PACKARD CO. 1984

: _              return to operating system
```

If other HP 3000 operating system errors occur while running LK 3000, use the CONTROL Y, CARRIAGE RETURN, and/or DATA LINK BREAK softkeys to recover from the error. In some cases, re-running LK 3000 and logging-on again may be required.

There is a special-case error which is not recognized as such by the LK3000 utility. This error occurs if you under-specify the record size of the new file to be created on the HP 3000. This error shows itself by the termination of the data file transfer without the display of the message :

```
END OF FILE TRANSFER
```

USING MODEMS

The LK 3000 data communication link can be used with direct connection between the HP 3000 and the HP 250/260, and also when the devices are connected indirectly using modems.

Your HP 260 will operate with the following types of modem:

```
BELL 103J
BELL 212A
European standard CCITT V.21 modems
European standard CCITT V.22 modems
```

Operating Considerations

Be sure to consider these points when using LK 3000.

- Program (PROG) files cannot be transferred from the HP 250/260 without first making them DATA files.

- IMAGE files cannot be transferred. If you wish to transfer a data base or data set, first write an HP 250/260 program to read the data set. Then, create a DATA file and write the appropriate information into the file using PACK and UNPACK statements.

- The HP 250/260 and HP 3000 do not have the same floating point capabilities. When transferring information to the HP 3000, checks should be made to ensure that the numbers do not overflow or underflow on the HP 3000.

- Binary (BIN) files cannot be transferred.

Floating Point Ranges

Limit	HP 250/260	HP 3000
Maximum	9.9E99	5.7896E76
Minimum	1E-99	1.727E-77

TIO ERROR CODES

APPENDIX D

TIO may produce the following error codes for problems detected at execution time.

310 Port ordinal out of range. This error results from the execution of any TIO statement in which the value of the expression specifying the addressed port ordinal is not in the range 11 thru 20.

311 Priority value out of range. This error results from the execution of a TIO on-interrupt statement in which the value of the expression specifying the interrupt execution priority is not in the range 1 thru 15.

312 Invalid on-interrupt statement. This error results from the execution of a TIO on-interrupt statement in which one of the following conditions is detected:

(1) The addressed port is not a terminal, printer, or computer.

(2) An ON INPUT # statement addresses a printer port, or a port whose state is not input available, output active, or output buffer empty.

(3) An ON OUTPUT # statement addresses a computer port, or a port whose state is not output active or output buffer empty.

(4) An ON BREAK # statement addresses a computer port.

(5) An ON TRIGGER # statement addresses a port which is not a computer.

(6) An ECHO ON # or ECHO OFF # statement addresses a port which is not a terminal.

(7) A BLOCK MODE # statement addresses a port which is not a computer.

314 Ownership error. This error results from the execution of any TIO statement addressing a port to which the running task has not obtained exclusive access by means of the REQUEST statement.

315 No input available. This error results from the execution of an AREAD$ function addressing a port whose state is not input available.

316 Invalid SEND statement. A SEND or SEND BREAK statement addresses a port which is not a computer.

TIO may produce the following error codes during the processing of the indicated main system statements which link to TIO. Note that reference to the PRINTER IS statement includes the variants SYSTEM PRINTER IS and PRINT ALL IS. Reference to the PRINT statement includes PRINT USING and other output generating statements such as CAT and LIST.

130 Logical device ID out of range (PRINTER IS, REQUEST, and RELEASE statements). The addressed port ordinal is not in the range 11 thru 20.

131 Resource busy (PRINT statement, REQUEST statement with wait parameter omitted). Another running task has been granted exclusive access to the addressed port.

132 Device is not printer (PRINTER IS statement). The addressed port is not a printer, terminal, or computer.

133 Printer down or disconnected. The addressed port of a PRINTER IS is disconnected. This error also results from the execution of a PRINT statement addressing a port whose state is not output active or output buffer empty.

SETTING ASI PORT FUNCTIONALITY

APPENDIX
E

This appendix describes how to set the functions of the ports on the ASI board of your computer so that the ON CONNECT and the ON DISCONNECT statements are properly serviced.

Setting the Ports on an HP 260 Series 30 or an HP 260 Series 40

The ASI ports of the HP 260 Series 30 and the HP 260 Series 40 computers are set through software. When you select the mode of communication for each ASI port (using the CONFIG utility), the functionality of the port with respect to the TIO DROM is also defined. If you define a port as a "Modem communication" port, the port will support the ON CONNECT and ON DISCONNECT statements.

Refer to the UTILITIES MANUAL that was supplied with your system for the details of how to configure the ASI ports using the CONFIG utility program.

Setting the Ports on an HP 260 (with Product No. 45261D)

Each port on the ASI board has two banks of DIP switches associated with it. The left bank of switches is used to select the mode of communication for the port (that is, RS-232, RS-422 or current loop). The right bank of switches is used to select the control line parameters, and it is this right bank that affects programming with TIO.

To enable the ON CONNECT and ON DISCONNECT statements, switch 1 in the right bank must be in the closed position and switch 2 in the right bank must be in the open position.

Setting the Ports on an HP 250

The computers with the ASI board of part number 45120-66550 are referred to in this section as the HP 250.

The ASI board (Part Number 45120-66550) has ten jumpers associated with each of its five ports.

To enable the ON CONNECT and the ON DISCONNECT statements, jumper 9 must be in the "B" position.

NOTE

It is also necessary to connect devices to the port with a MODEM cable. Direct connect cables cannot handle the ON CONNECT and ON DISCONNECT statements.

Part No. 45120-90006 E0986

Printed in Federal Republic of Germany
September 1986

HERRENBERGER STRASSE 130
D-7030 BOEBLINGEN