Asynchronous Data Communication ROM Programming

Part No. 98046-90010
Microfiche No. 98046-99010

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Table of Contents

Chapter 1
Table of Contents

General Information

Chapter 1
General Information

Introduction

This manual was written to show you how to implement asynchronous data communications. It is assumed that you have access to a host computer, link, and the appropriate modem (if required).

Relatively simple datacomm programs can be written with the additional BASIC language statements provided by the Basic Data Communications ROM package. Here is an example program:

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !       ********************************************************
80   !       !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                  ! Dimension the string variables
110       CDISCONNECT 4                  ! Ensure you are disconnected
120       CCOM 1500                      ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                 ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF       ! Link is connected here
150       CCONTROL 4;ACK ON              ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit     ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive      ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE               ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                     ! Wait here for an interrupt
230   !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN  ! Check for a PROMPT before continuing
260       CREAD 4;A$                     ! Get the visual prompt character
270       PRINT A$;                      ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$         ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE            ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN  ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                     ! Read the data from the host
350       PRINT A$                       ! Print the data from the host
360  GOTO Receive                        ! Go check for more input data
370  !
380  END
```

The Basic Data Communications ROM

The following table shows the ROM(s) required for asynchronous communication. ROM installation procedures are explained in the Owner's Manual supplied with your Desktop Computer.

Desktop Computer	Option ROM(s)
System 35	98317A
System 45B	98417A

The Datacomm Package

The Basic Data Communications ROM and the HP 98046 Interface (or datacomm package for short) provide the following:

- Additional BASIC statements for data communications.
- Asynchronous line protocol management.
- Program selectable automatic modem line handling.
- Troubleshooting capability.

The BASIC Statements

The additional BASIC language statements required for data communications are described in Chapters 2, 3, 4 and 5. An example program is shown at the beginning of each of these chapters. The BASIC statements that are discussed within the chapter are highlighted so that you can see how they appear in an actual program. Optional parameters allow you to vary your datacomm program in order to match the requirements of the host computer. Defaults are provided for all parameters. Chapter 10 describes an additional statement that is provided as an enhancement for terminal emulator applications.

Techniques are shown in Chapters 6 and 7 that explain how to use the features provided by the Basic Data Communications ROM. Examples show how to use the CSTAT features to input data on a line by line (line mode) and a character by character (character mode) basis. Example statement definition and log-on procedures are shown to help you establish your "first connection" to the host computer.

Troubleshooting procedures, probable causes, and suggested remedial steps are explained in Chapters 8 and 9. The additional BASIC statements provided for troubleshooting datacomm problems are described.

Log-On

The Log-On procedure is dictated by the host computer. To obtain Log-On procedures, you should contact the system analyst responsible for the host computer.

NOTE

The programming requirements that enable you to perform datacomm operations are dictated by the host computer. If you implement Desktop-to-Desktop communications, you may specify your own requirements. (See Appendix G for an example Desktop-to-Desktop program.)

Chapter Summaries

- Chapter 1 provides a description of the asynchronous datacomm package, an introduction to data communications terminology, and an introduction to asynchronous communication.
- Chapter 2 describes the statements that allocate memory and establish the data communication link: CCOM, CMODEL ASYNC, CCONNECT. CDISCONNECT is also described.
- Chapter 3 describes the statements that move data to and from the BASIC strings contained in your datacomm program: CREAD and CWRITE.
- Chapter 4 describes the statements that provide control and interrupt capability: ON INT and CCONTROL.
- Chapter 5 describes the statement and function that enables you to monitor the status of your datacomm operation: CSTATUS and CSTAT.
- Chapter 6 explains how to use the CSTAT function to determine when to read and write data to the communications link.
- Chapter 7 explains how to establish your "first connection." Log-on procedures are described.
- Chapter 8 describes the BASIC statements provided for troubleshooting, CTRACE and CDUMP.
- Chapter 9 describes datacomm errors and suggests possible corrective action.
- Chapter 10 describes the TDISP statement. An example TDISP program is provided.
- Appendix A contains the complete ASCII table.
- Appendix B contains a listing of the RS-232/CCITT V.24 signals.
- Appendix C contains a list of modems that have been tested with the Datacomm Package.
- Appendix D provides a detailed explanation of memory allocation for data communications.
- Appendix E describes half duplex and secondary channel protocols. Example programs are shown.
- Appendix F contains a description and program showing a typical autoanswer application.
- Appendix G contains example program applications.

Terminology

The world of data communications is full of many terms that may be new to you. The following terms are used in the discussion of the various options provided by the CMODEL ASYNC statement:

Term	Meaning
Character	May consist of a number, letter, or a special symbol (e.g. 1, 4, A, b, $, #).
Code	Binary representation of a character. The code used is ASCII.
Line	Consists of one or more characters.
Terminator	A character or sequence of characters that is inserted into the data stream to indicate the end of a line.
Parity	Check bit that may be inserted at the end of a transmitted character so that the receiving device can determine if the character was correctly received.
Prompt	A predictable character or sequence of characters sent by the host to indicate that the host is ready to receive data from you.
Gap	A pre-defined time between the start of one transmitted character and the start of the next transmitted character.
Host	The remote computer that is connected to the other end of your communications link.
Character Mode	A transfer mode is which output data is transmitted and received data is processed on a character by character basis.
Line Mode	A transfer mode in which output data is transmitted and received data is processed on a line by line basis.

Other terms, such as input queue, output queue, protocol machine, etc. are used throughout this manual. Refer to the Data Communication Basics Manual (98046-90000) for a discussion of these terms.

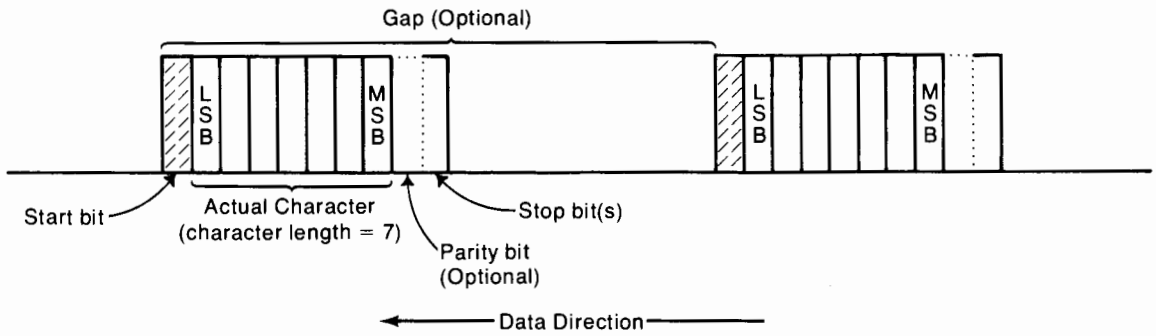
Asynchronous Communication

Asynchronous communication (as implemented by the Basic Data Communications ROM) means the following:

- Each character sent over the communications link is independent of all other characters sent over the link. Characters are not synchronized with respect to each other.
- Each bit within a character is synchronized with all other bits within that character.
- Each character is preceded by a start bit and followed by one or more stop bits. The start bit is used to synchronize the receiving device to the character being transmitted, and the stop bit(s) are used to signal the end of the character.

The following drawings show typical asynchronous characters.

Character Definition



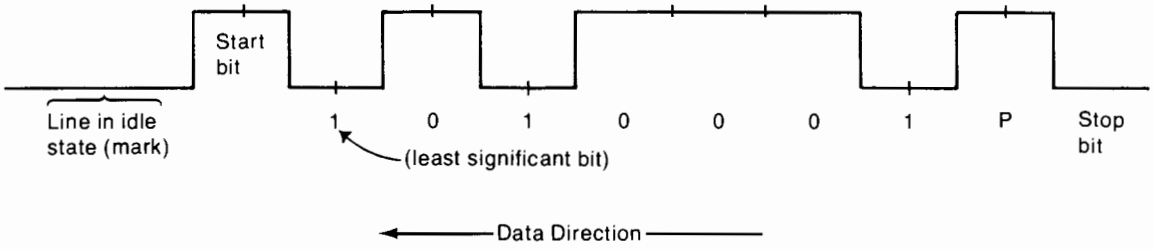
The actual data, however, is transmitted over the line using two voltage levels to represent the two possible states of a binary digit (0 and 1). The following table shows the two possible binary states and the meanings assigned to each.

Binary State:	logic 0	logic 1
Voltage Range:	+ 3V to + 25V	− 3 to − 25V
Level Name:	SPACE	MARK
Line State:	high	low (idle)

When data is not being transmitted, the line is held in the LOW state. Unlike the other methods of interfacing, the serial protocol does not use any type of handshake process. When the transmitting device has a byte of information ready to send, it merely puts the information on the data line, expecting the receiving device to be ready to take it. If the first bit of the data byte sent happens to be a logical 1 (LOW state), the receiver could not distinguish this bit from the quiet line, which is also a LOW state. Therefore, each byte of data is preceded by a start bit, which is defined to be in the HIGH state. This transition from the LOW state (idle line) to the HIGH state (start bit) lets the receiver know that a byte of data is being transmitted. As an example, let's look at how the transmitter would encode the ASCII character "E" to be sent over the data line.

The next drawing shows the state changes that take place on the data line to send the ASCII "E." The transmitting device first pulls the data line HIGH (start bit) to tell the receiver that a data byte is coming. It holds the line high for an amount of time agreed upon between the transmitter and the receiver, called a bit time. Following the start bit, the bits of the data byte itself are placed on the data line. The least significant bit (bit 0) is sent first, and each bit is held on the line by the transmitter for one bit time. When the receiver senses the leading edge of the start bit, it waits for one half of a bit time in order to synchronize itself as closely as possible to the center of that start bit. Then, each bit time interval after that, it samples the state of the data line and reads a logic 1 or 0. These time intervals at which the receiver samples the data line are marked by ticks. After the last (most significant) data bit has been sent, the transmitter may also send a parity bit (marked P) which will be discussed later.

Character Transmission



From this diagram, we see that the successful transmission of a data byte is highly dependent on precision timing. If the receiver is sampling the data line at a rate significantly faster or slower than the transmitter is setting that line, it is possible that the receiver will either miss a bit, or sample the same bit twice, resulting in erroneous data being received.

After the last data bit has been sent, the transmitter then allows the line to stay in the idle (low) state for some set minimum time interval before sending the next start bit to begin the next character transmission. This idle time is sometimes called a stop bit, although it does not actually represent a bit of real data. It merely allows the receiver time to process the data byte just received before the next one comes along. For some devices, one bit time may not be enough to process the previous character and be ready for the next one. In this case, the transmitter and receiver may agree that the transmitter will wait in the idle state for 1.5 or 2 bit times before sending the next start bit.

Protocols

Protocols must be implemented in order to ensure an orderly exchange of information between your Desktop Computer and the host. Optional protocol parameters are provided with the CMODEL ASYNC statement that enable you to match the requirements of various host computers.

The previous drawing shows how data characters are transferred asychronously. These characters, however, are of no use unless the two devices that are communicating can interpret them into meaningful information. The next section explains these various types of specifiers:

Character Length	How many bits make up a complete character?
Parity	Is parity specified? If so, what type?
Stop bits	How many stop bits are output with each character?
Gap	What is the minimum permissible spacing between characters?
Prompt	What character sequence defines a prompt?
Terminator	What character (or characters) are used to indicate line terminator sequences?

Character Length

Character length defines how many bits are used for each character. Each character is preceded by a start bit, and after the specified number of bits in the character have been passed, the correct parity bit (if specified), and the stop bit(s) are inserted.

NOTE

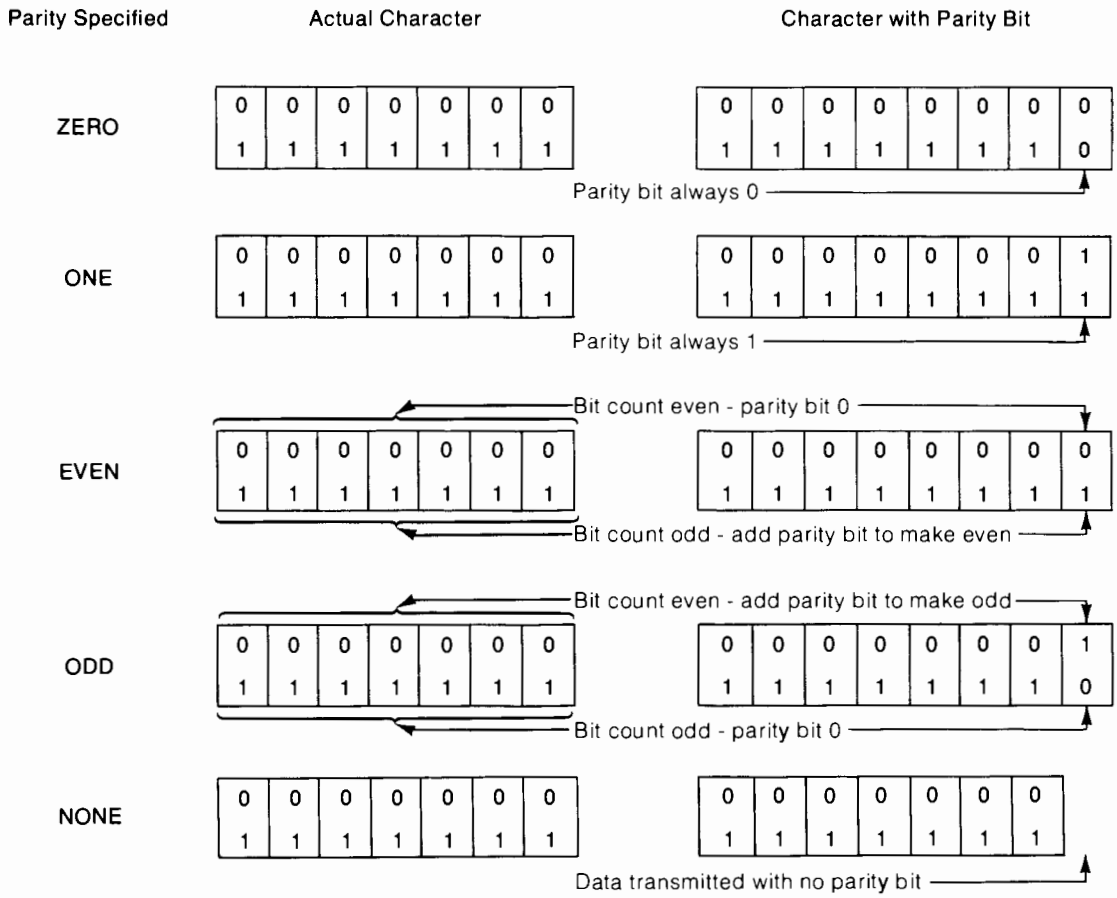
Character length does **not** include start, stop, and parity bits.

Parity

Parity is a method of error checking. Parity (when specified) checks the output data for an even or odd bit count and adds the appropriate parity bit (as needed). Input data is checked for the correct parity (as specified) and generates an error if the data pattern is incorrect and converts the character which generated the error into an underscore character (see READ ALL, Chapter 4).

The following tables show the effect of the various parity specifiers with the Start and Stop bits omitted for clarity. Character length is assumed to be 7.

Parity Tables



Stop Bits

Stop bits are used to indicate the end of a character. The stop bit(s) are added to the end of each of the characters as they are transmitted. If a parity bit has been specified, the stop bit(s) follow the parity bit.

Gap

Some devices are capable of receiving data at high transfer rates but require a minimum time between characters in order to input the data and clear their input buffer. The gap parameter is used to specify the time delay between the start of successive characters.

Prompt

A prompt is a predictable character or character sequence sent from the host to the terminal to indicate that the host is ready to receive data from the terminal. The host sends a visual prompt (e.g.,":") followed by the prompt sequence as a terminator. The HP3000 transmits a DC1 (CHR$(17)) as a prompt sequence.

Separators

When data is transferred asynchronously, separators are inserted into the message stream to indicate that a complete line has been sent. Input data is checked for the presence of input line separators (INSEP). The program is informed when INSEP is detected. Output line separators (OUTSEP) are inserted into the output message stream at the end of each line to indicate the completion of an output line.

NOTE

The requirements for predictable prompt and separator characters are explained in Chapter 2 (see the CMODEL Statement).

Datacomm Interface Select Code Guidelines

The HP 98046B Datacomm Interface requires two (2) adjacent select codes, an EVEN number, and the next higher ODD value. Select codes are determined by the setting of the rotary switch at the top rear of the interface housing, and can be changed by using a small screwdriver. The switch can be set to either the even or odd value for any given select code pair. To ensure reliable operation, observe the following rules:

- **Do not use 98046B select codes of 8 or greater** if you are using the tape drive(s) in your desktop computer. System lock-up occurs when certain datacomm activities and mass storage operations are attempted simultaneously. If you need more select codes, set other interfaces to the higher values instead of the 98046B.
- Every interface must use a unique select code or select code pair. Select code 0 is reserved for internal printer and keyboard, 15 and 16 for tape drive and CRT, and 14 and 15 are reserved for graphics and optional tape drive for the HP 9845. Don't forget that the 98046B occupies TWO adjacent select codes.
- Datacomm statements that address the HP 98046B Datacomm Interface include a select code specifier. Most programmers use the EVEN select code although you can arbitrarily use either the even or odd select code. Statements that can use either select code include CMODEL, CCONNECT, CDISCONNECT, CCONTROL, CSTAT, CSTATUS, CDUMP, and CTRACE.

  The ON INT# statement is used to provide program interrupt by datacomm when certain CSTATUS conditions are met. If CSTATUS element 1 is used to interrupt the program, the ODD select code value MUST be used. For all other CSTATUS interrupts, the ON INT# statement must specify the EVEN select code. Refer to Chapter 5 for more information.

Chapter 2
Table of Contents

Memory Allocation and Connecting

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !       ********************************************************
80   !       !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                  ! Dimension the string variables
110       CDISCONNECT 4                  ! Ensure you are disconnected
120       CCOM 1500                      ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                 ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF       ! Link is connected here
150       CCONTROL 4;ACK ON              ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit     ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive      ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE               ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                     ! Wait here for an interrupt
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN  ! Check for a PROMPT before continuing
260       CREAD 4;A$                     ! Get the visual prompt character
270       PRINT A$;                      ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$         ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE            ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN  ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                     ! Read the data from the host
350       PRINT A$                       ! Print the data from the host
360  GOTO Receive                        ! Go check for more input data
370  !
380  END
```

Chapter 2
Memory Allocation and Connecting

The following is a summary of the information provided in this chapter:

Statement or Function	Parameter or Option	Default
CCOM	memory size	
CMODEL ASYNC, select code	MEMLIMIT = expression	500
	INBUFFER = expression	260
	TBUFFER = expression	130
	INSEP = string	CR/LF
	OUTSEP = string	CR
	PROMPT = string	DC1
	ALERTN = expression	32767(off)
	CHECK = expression	3(odd)
	CHARLENGTH = expression	7(bits)
	STOPBITS = expression	1
	GAP = expression	0(milliseconds)
	HALF DUPLEX or FULL DUPLEX	FULL DUPLEX
CCONNECT select code	SPEED = expression	300(baud)
	INSPEED = expression	300(baud)
	OUTSPEED = expression	300(baud)
	EXTERNAL HANDSHAKE ON or HANDSHAKE OFF	HANDSHAKE ON
	LOST CARRIER = expression	400(milliseconds)
	NO ACTIVITY = expression	600(seconds)
CDISCONNECT select code	HOLD	

The highlighted program lines on the facing page show typical use of these statements.

The CCOM Statement

The CCOM statement specifies (in words) the amount of user memory that is allocated for your datacomm operation.

Syntax:

CCOM memory size

memory size: The memory size parameter is highly dependent on which optional parameters are implemented by the CMODEL statement. If defaults for the CMODEL ASYNC memory allocation parameters are used, then CCOM memory size MUST be greater than 1416.

More than one CMODEL statement can appear in a program. To determine the minimum memory size required by the CCOM statement, see Appendix D.

Considerations:

- CCOM memory is allocated at pre-run.
- The CCOM statement MUST be executed from a program. If a CCOM statement is contained in a subprogram, it is ignored.
- If the memory size parameter is insufficient for the datacomm operation, error 307 is generated.
- If a CCOM statement with a different memory size is executed while a datacomm channel is active, error 301 is generated.
- Larger memory allocation provides more buffering and allows more overlap between datacomm activities and program processing. Increasing CCOM memory does NOT provide additional memory space for the CMODEL unless the appropriate MEMLIMIT, INBUFFER, or TBUFFER parameter is increased (see Appendix D, Memory Management, for details).
- The CCOM statement may appear anywhere in the datacomm program but MUST precede any COM statements.
- CCOM memory remains allocated when a stop or pause is executed.
- CCOM memory is reclaimed by executing CONTROL-STOP (Reset).

Under certain conditions, CCOM memory allocation can be reclaimed by executing SCRATCH C or CCOM 0; see Appendix D, Memory Management, for details.

The CMODEL ASYNC Statement

The CMODEL ASYNC statement provides you with asynchronous terminal emulating capability. The CMODEL ASYNC statement provides optional parameters that may be specified for your particular application. Defaults for all parameters are provided. The CMODEL ASYNC statement provides an interrupt to the EVEN select code for autoanswer implementation (see Appendix F). The CCOM memory MUST be allocated prior to executing the CMODEL ASYNC statement. The CMODEL ASYNC statement canNOT be executed to a connected channel. When the CMODEL ASYNC statement is executed, the memory space required to satisfy the MEMLIMIT, INBUFFER, and TBUFFER parameters is reserved from the total CCOM allocation.

Syntax:

CMODEL ASYNC,select code[; parameter list]

select code: The valid select code (channel) for your datacomm operation.[1]

parameter list: The parameter list may contain none, any, or all of the following:

- MEMLIMIT = expression
- INBUFFER = expression
- TBUFFER = expression
- INSEP = string
- OUTSEP = string
- PROMPT = string
- ALERTN = expression
- CHECK = expression
- CHARLENGTH = expression
- STOPBITS = expression
- GAP = expression
- HALF DUPLEX

  or
- FULL DUPLEX

If more than one optional parameter is included, then each parameter must be separated by a comma as a delimiter.

The optional parameters for the CMODEL ASYNC statement may be included in the parameter list in any order. If any parameter is omitted from the parameter list, then the default for the omitted parameter is assumed.

1 For select code guidelines, see Chapter 1.

For clarity of explanation, the parameters for the CMODEL ASYNC statement have been separated into functional groups. These functional groups and their associated parameters are shown below.

Memory Allocation Group	Protocol Group
MEMLIMIT	INSEP[1]
INBUFFER	OUTSEP[2]
TBUFFER	PROMPT[1]
	ALERTN[1]
	CHECK[3]
	CHARLENGTH[3]
	STOPBITS[2]
	GAP[2]
	HALF DUPLEX[3]
	FULL DUPLEX[3]

The Memory Allocation Group

The following group of parameters enable you to specify the portion of the entire CCOM memory allocation that is used by this CMODEL ASYNC. Details about memory allocation can be found in Appendix D, Memory Management.

MEMLIMIT = expression

Specifies the maximum amount of input and output queue space that is allocated for this CMODEL ASYNC statement.

expression:

An integer value ranging from 250[4] to 32055.

Default: **MEMLIMIT = 500**

INBUFFER = expression

Specifies the amount of CCOM memory that is allocated for the input buffer for this CMODEL ASYNC statement.

expression:

An integer value ranging from 150[4] to 29060.

Default: **INBUFFER = 260**

TBUFFER = expression

The TBUFFER parameter specifies the amount of CCOM memory that is allocated for the trace buffer for this CMODEL ASYNC statement.

expression:

An integer value ranging from 10[4] to 28920.

Default: **TBUFFER = 130**

1 These specifiers affect input (receive) operations only.

2 These specifiers affect output (transmit) operations only.

3 These specifiers affect transmit and receive operations.

4 If a parameter value less than the minimum shown is specified, then the minimum for that parameter is assumed.

The input queue, output queue, INBUFFER and TBUFFER are explained in the Data Communication Basics manual, Chapter 5.

The Protocol Group

The purpose of the options contained in the protocol group is to enable you to implement the protocols required by the host computer. This section explains each of the protocol group options.

NOTE

The PROMPT and INSEP parameters are used when the host sends PREDICTABLE characters to indicate a prompt or end-of-line. The OUTSEP parameter is used when the host expects a PREDICTABLE end-of-line sequence from you. When no predictable prompt or end-of-line characters are output or expected by the host, you must implement your own techniques to determine prompt and end-of-line indication.

INSEP = string

The input line separator specification allows you to recognize the character sequence that is sent by the host to indicate the end of a line of data. This permits the program to process input data on a line by line basis rather than a character by character basis for greater program efficiency. BASIC is informed when an INSEP sequence is detected (see CSTATUS elements 1 and 3, Chapter 5).

string: Any valid string expression not exceeding 2 characters.

Default: **INSEP** = **CR**/**LF** (CHR$(13)&CHR$(10)).

OUTSEP = string

Output line terminator. This allows you to specify a terminator to an output line. The OUTSEP sequence is output when CWRITE...ENDLINE is executed.

string: Any valid string expression not exceeding 2 characters.

Default: **OUTSEP** = **CR** (CHR$(13)).

PROMPT = string

The prompt informs your Desktop Computer that the host is ready to accept data. The PROMPT specifier designates the character that is recognized as a prompt from the host. The prompt from the host is a control character that immediately follows the visual prompt that appears on your computer. The BASIC program is informed when a prompt has been detected (see CSTATUS element 2, Chapter 5).

string: Any valid string expression not exceeding 2 characters.

Default: **PROMPT** = **DC1** (CHR$(17))

ALERTN = expression

When the amount of characters specified by the expression have been input, BASIC is informed. (See CSTATUS element 1 in Chapter 5.) This provides an alternate method of defining a complete block of input data. It may be desirable for the BASIC program to process a specified number of input characters rather than a complete line. ALERTN may also be used when there is no predictable input line separator sequence (INSEP) from the host.

expression:

Any integer ranging from 1 to 32767. When the value of the expression is greater than the INBUFFER parameter, this feature is disabled (off).

Default: **ALERTN = 32767** (off)

CHECK = expression

The CHECK parameter specifies parity in accordance with the following values for the expression.

0 Specifies the parity bit that is output with the character is always 0. Input characters are NOT checked for parity.

1 Specifies the parity bit that is output with the character is always 1. Input characters are NOT checked for parity.

2 Specifies that the output characters are sent with even parity, and the input characters are checked for even parity.

3 Specifies that the output characters are sent with odd parity, and the input characters are checked for odd parity.

4 Specifies that the output characters are sent with NO parity bit, and the input characters are NOT checked for parity.

Default: **CHECK = 3**

CHARLENGTH = expression

The CHARLENGTH parameter enables you to specify how many bits make up each character. This does NOT include the parity bit if parity is specified.

expression:

The value of expression can be 5,6,7, or 8.

Default: **CHARLENGTH = 7**

NOTE

When CHARLENGTH = 8 is specified, the value of the expression for the CHECK = parameter MUST be 2, 3 or 4. A value of 0 or 1 generates Error 309.

STOPBITS = expression

The STOPBITS parameter allows you to specify how many stop bits are sent with each character.

expression:

The value of expression can be 1, 1.5, or 2.

Default: **STOPBITS = 1**

GAP = expression

The GAP parameter allows you to specify the minimum time (in milliseconds) between the start bit of one character and the start bit of the next character for your transmitted data. This is used with slow speed computers and terminals to prevent overrun.

expression:

The expression is an integer ranging in value from 0 to 511.

Default: **GAP = 0**(off)

HALF DUPLEX
or
FULL DUPLEX

The HALF DUPLEX parameter allows you to specify communications using half duplex line protocols. Half duplex and full duplex protocols are explained in the Data Communication Basics Manual, Chapter 3. A half duplex example is shown in Appendix E.

Default: **FULL DUPLEX**

The following table shows the parameters used to implement the asynchronous protocols that are required by different host computers.

Parameter	Host Computer	
	HP-3000	**HP-1000**[1]
INSEP =	CR/LF	CR/LF
OUTSEP =	CR	CR
PROMPT =	DC1	DC1
ALERTN =	*	*
CHECK =	3(odd)	4(none)
CHARLENGTH =	7	8
STOPBITS =	1	1
GAP =	0	0
HALF DUPLEX or FULL DUPLEX	FULL DUPLEX	FULL DUPLEX

* User defined option.

1 The HP-1000 12966-60006 interface must be used. If other HP-1000 interfaces are used, the parameters may be different than those shown here.

The following two examples are acceptable CMODEL ASYNC statements and are equivalent.

```
10  CMODEL ASYNC,4;MEMLIMIT=500,INBUFFER=260,TBUFFER=130,INSEP=CHR$(13)&CHR$(10)
,OUTSEP=CHR$(13),PROMPT=CHR$(17),CHARLENGTH=7,CHECK=3,GAP=0,FULL DUPLEX
```

```
10  CMODEL ASYNC,4
```

The CCONNECT Statement

The CCONNECT statement, when executed, specifies the transfer rate and the handshake mode and establishes the line connection. The CCOM statement and the CMODEL ASYNC statement for the specified select code MUST be executed prior to executing the CCONNECT statement. Executing more than one CCONNECT statement for the same select code generates error 301.

NOTE

Once a line connection is established, it is maintained until a CDISCONNECT is executed or the Desktop Computer is reset. Stopping or pausing the program does NOT terminate the connection.

Syntax:

CCONNECT select code [; option list]

select code: The valid select code (channel) for your datacomm operation.[1]

option list: The option list may contain:

- SPEED = expression
- INSPEED = expression
- OUTSPEED = expression
- EXTERNAL
- LOST CARRIER = expression
- NO ACTIVITY = expression
- HANDSHAKE ON

  or
- HANDSHAKE OFF

1 For select code guidelines, see Chapter 1.

The following SPEED parameters allow you to specify the various transfer rates from the internal clock.

SPEED = expression	The SPEED parameter allows one specification for both transmit and receive data rates.
expression:	The values for the expression are found in the next table.
INSPEED = expression	The INSPEED parameter enables you to specify the data rate of the input (receive) operation.
expression:	The values for the expression are found in the next table.
OUTSPEED = expression	The OUTSPEED parameter enables you to specify the data rate for your output (transmit) operation.
expression:	The values for the expression are found in the next table.

The INSPEED and OUTSPEED parameters are provided for split speed (different transmit and receive data rates) operation. Acceptable values for all SPEED parameters are:

0 (EXTERNAL)
50
75
110
135
150
200
300
600
1200
1800
2400
4800
9600

Default: **SPEED = 300**

EXTERNAL — The EXTERNAL parameter, when specified, allows the data transfer rate to be controlled by an external source, e.g., the host computer or a modem. The SPEED = 0 parameter is provided as a numeric specifier for EXTERNAL.

NOTE

Asynchronous communication seldom uses the EXTERNAL or SPEED = 0 parameter.

The ROM is designed to avoid redundant SPEED specifiers. It is important to note that the SPEED, INSPEED, OUTSPEED, and EXTERNAL parameters are only valid in the following combinations:

First Parameter Specified	Other Parameter Allowed
INSPEED	OUTSPEED
OUTSPEED	INSPEED
SPEED	None
EXTERNAL	None

LOST CARRIER=expression

This option is valid only when HANDSHAKE ON is specified. When HANDSHAKE OFF is specified, LOST CARRIER is ignored. The LOST CARRIER parameter provides an automatic disconnect from the link as follows:

- Half duplex operation — The Data Set Ready[1] modem line becomes inactive (low) for the time specified.
- Full duplex operation — The Data Set Ready[1] or Data Carrier Detect line becomes inactive (low) for the time specified.

When the LOST CARRIER disconnect is executed, all modem lines are set low.

expression:

The expression is an integer value ranging from 0 to 510. The expression value specifies the period of time (in milliseconds) that the LOST CARRIER parameter waits before executing a disconnect. When a value of 0 is specified the automatic disconnect for LOST CARRIER is disabled.

Default: **LOST CARRIER = 400**

NOACTIVITY = expression

The NOACTIVITY parameter allows you to specify an automatic disconnect if there is no transmitted or received data detected on the line for the amount of time specified. This option is used to prevent line tie-up with resultant large telephone charges if the link is unintentionally left connected.

expression:

The expression is an integer value ranging from 0 to 5100 and represents the period of time (in seconds) that the program senses NOACTIVITY before executing an automatic disconnect. When a value of 0 is specified the automatic disconnect for NOACTIVITY is disabled.

Default: **NO ACTIVITY = 600** (10 minutes)

1 Modem signals are described in the next section (see HANDSHAKE ON).

HANDSHAKE OFF

The HANDSHAKE OFF option is provided for hardwired[1] (no modem) 3 wire applications. When HANDSHAKE OFF is specified, modem signals are still controlled. The following modem signals are monitored but do not affect operation:

- Clear To Send
- Data Set Ready
- Data Carrier Detect

Default: **HANDSHAKE ON**

HANDSHAKE ON

The HANDSHAKE ON option is provided for datacomm operations that involve modems. When HANDSHAKE ON is specified, the ROM and interface automatically monitor and control the appropriate modem signals. The following modem signals can be monitored with CSTATUS[2] elements 11, 12, 13 and 14:

- Data Set Ready — CSTATUS element 11
- Secondary Carrier Detect — CSTATUS element 12
- Carrier Detect — CSTATUS element 13
- Clear To Send — CSTATUS element 14

Modem signals and handshakes are explained next.

1 See Appendix B for commonly used 3 wire connection.

2 See the CSTATUS statement Chapter 5.

Modem Handshake

Modem handshake signals are used to allow conversation between two modems, the host and the modem, the terminal and the modem, and the terminal and the host. Typical types of information exchanged with handshake signals are:

• Request-to-Send	Terminal or host to modem. Requests control of the line in order to transmit information.
• Clear-to-Send	Modem to terminal or host. You have control of the line, you may transmit.
• Data-Terminal-Ready	Terminal or host to modem. Tells the modem that the terminal equipment is ready to connect to the line.
• Data-Set-Ready	Modem to terminal or host. Informs the terminal equipment that the modem is ready for data transfer to begin.
• Data-Carrier-Detect	This signal is used to inform the receiving terminal or host that incoming data is valid.

Using Modem Handshake (HANDSHAKE ON)

The following handshake signals are required:

- The Clear-to-Send signal must be active (high) before you can transmit.
- The Data-Carrier-Detect signal must be active (high) before you can receive.

The handshake sequence that occurs when the CCONNECT statement is executed is explained next.

The 25 Second Timeout

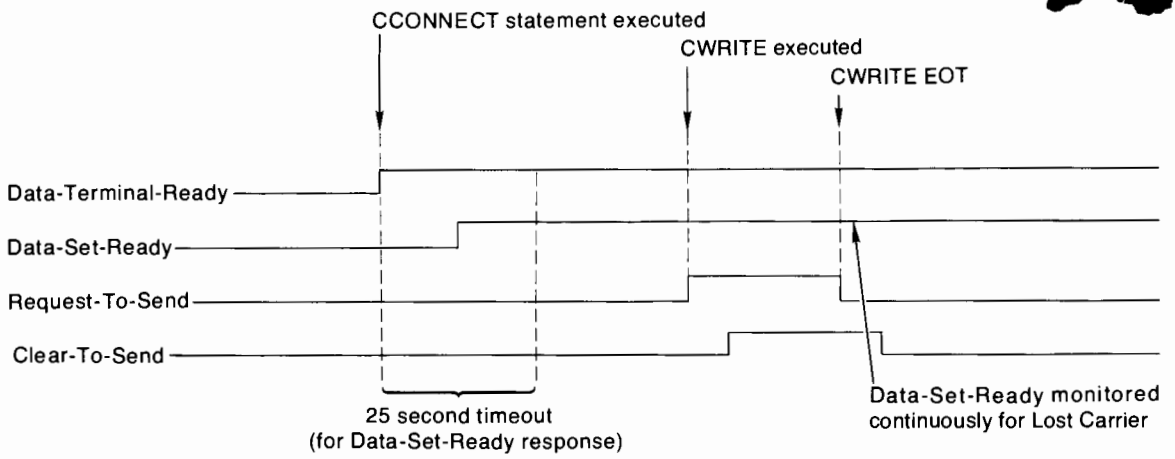
For HALF DUPLEX operation, when the CCONNECT statement is executed, the Data-Terminal-Ready signal is set high (active) and the Data-Set-Ready line from the modem is monitored. Data-Set-Ready must respond (become active) within 25 seconds or error 10,[1] Timeout before connection, is generated. When a CWRITE is executed, Request-to-Send is activated and Clear-to-Send is monitored. When Clear-to-Send is detected, your data transmission can begin. Data-Set-Ready is continually monitored and if it drops for longer than 400 milliseconds (LOST CARRIER timeout), error 104[1] is generated.

For FULL DUPLEX operation, Request-to-Send and Data-Terminal-Ready are activated at the same time. Data-Set-Ready and Data-Carrier-Detect are monitored for the 25 second timeout period and error 10[1] is generated if both signals do not respond within 25 seconds. Data-Set-Ready and Data-Carrier-Detect are then monitored continually and if either signal drops for more than 400 milliseconds (LOST CARRIER timeout), error 104[1] is generated.

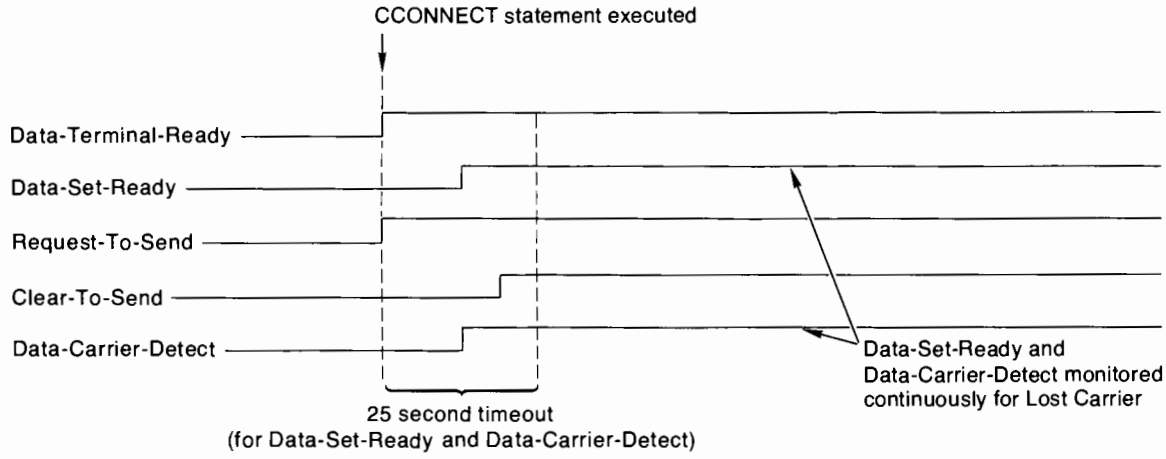
1 Error 10 and 104 are datacomm errors (see CSTATUS element 0, Chapter 5).

The following diagrams show typical modem handshake sequences.

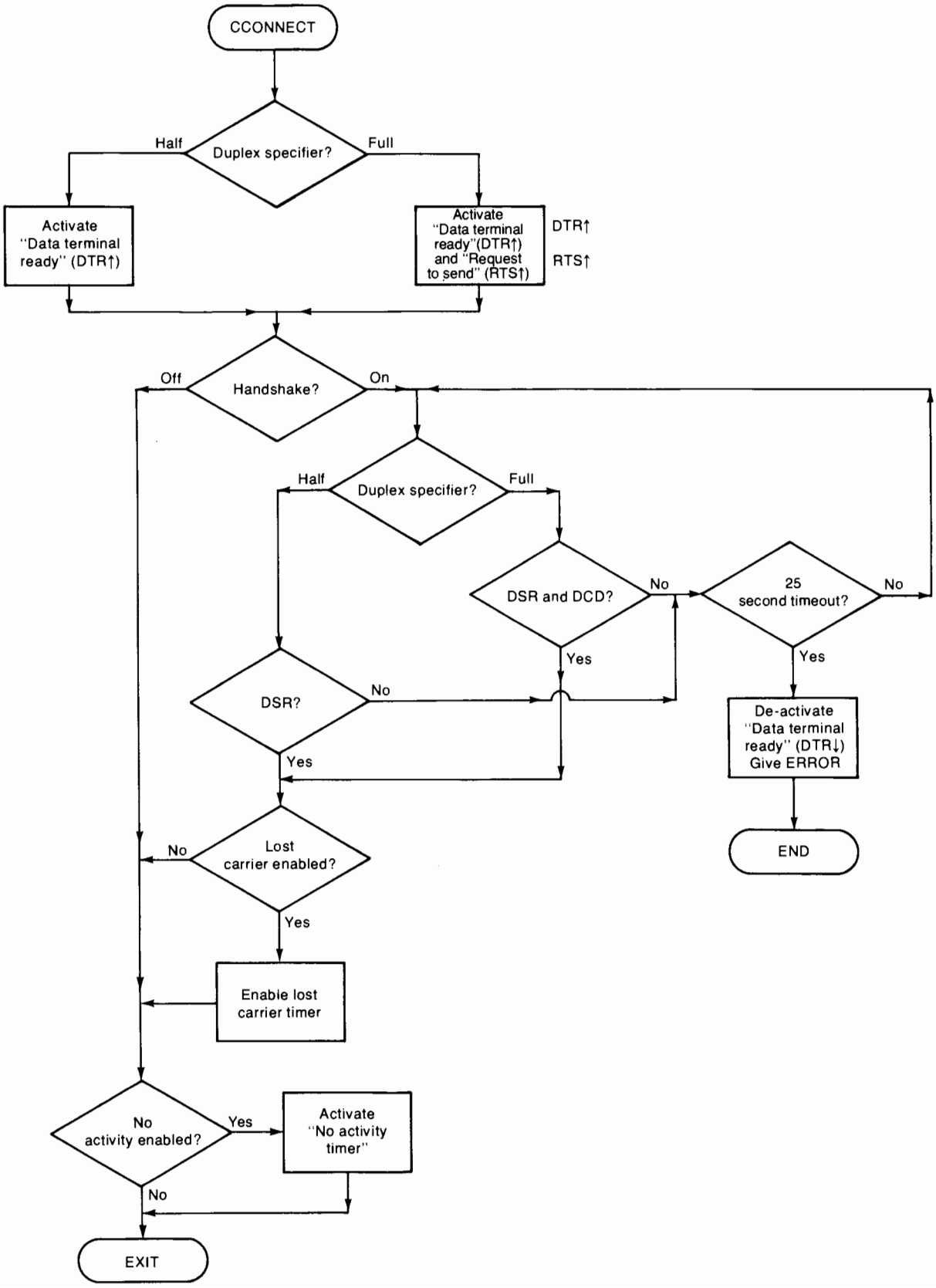
Establishing a half duplex connection.



Establishing a full duplex connection.



CCONNECT Flowchart



The CDISCONNECT Statement

The CDISCONNECT statement terminates all datacomm activity for the specified select code. If the optional HOLD parameter is not specified, all modem control signals are set low (0), and an automatic hang-up on a switched line is executed. CDISCONNECT can be executed to a channel that is not connected. A CDISCONNECT statement is normally placed before the CMODEL statement to prevent errors caused when attempting to execute a CMODEL statement for an active channel.

Syntax:

CDISCONNECT select code [; HOLD]

select code: The valid select code (channel) for your datacomm operation.

HOLD: The optional HOLD parameter prevents a true disconnect from the link. The datacomm channel is deactivated and communication is disabled. The modem signals remain in the active state to prevent automatic hang-up. The HOLD parameter is normally used to allow you to change the CMODEL statement options without disconnecting from the link.

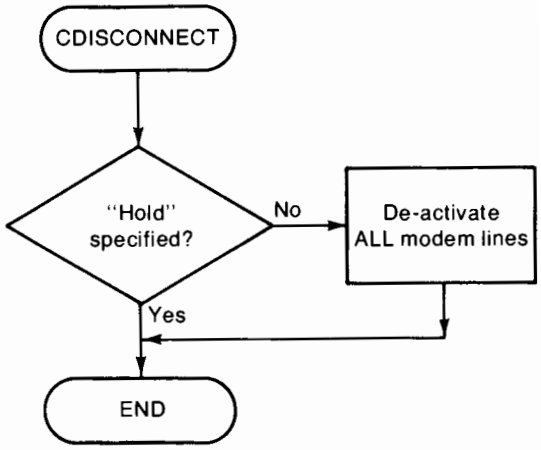
NOTE

In order to reconnect to a datacomm channel following a CDISCONNECT, you MUST execute a CMODEL and CCONNECT for the disconnected channel.

NOTE

When CDISCONNECT.... HOLD is executed, the modem signals remain active and the link connection is maintained. If you are connected to a telephone line, you remain connected until a CDISCONNECT without a HOLD is executed. All automatic disconnects (NOACTIVITY, LOST CARRIER) are disabled.

CDISCONNECT Flowchart



Chapter 3
Table of Contents

Data Transfer

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !      ********************************************************
80   !      !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                  ! Dimension the string variables
110       CDISCONNECT 4                  ! Ensure you are disconnected
120       CCOM 1500                      ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                 ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF       ! Link is connected here
150       CCONTROL 4;ACK ON              ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit     ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive      ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE               ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                     ! Wait here for an interrupt
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN ! Check for a PROMPT before continuing
260       CREAD 4;A$                     ! Get the visual prompt character
270       PRINT A$;                      ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$         ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE            ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                     ! Read the data from the host
350       PRINT A$                       ! Print the data from the host
360  GOTO Receive                        ! Go check for more input data
370  !
380  END
```

Chapter 3
Data Transfer

The following is a summary of the information provided in this chapter:

Statement or Function	Parameter or Option
CREAD select code	input list
CWRITE select code	output list ENDLINE EOT

The highlighted program lines on the facing page show typical use of these statements.

The CREAD Statement

The CREAD statement transfers data from the input queue into BASIC strings. The CREAD statement is provided to accomplish:

- Orderly extraction of message text as related to the line and logical separators.
- Processing of messages that are longer than the input queue storage capacity.

A CREAD can be executed at any time, whether data is present or not. Executing CREAD causes message text characters to be transferred from the input queue to the input list. INSEP and PROMPT sequences are stripped from the input data stream, but their positions are noted. Data transfers are terminated when one of the following conditions occur:

- An INSEP is encountered.
- The input queue is emptied.
- The BASIC string is filled.

Syntax:

CREAD select code ; input list

select code: The valid select code (channel) for your datacomm operation.

input list: The input list consists of one or more string variables, substrings, or string array elements. In general only one string variable is used with the CREAD statement, however, if more than one string variable is specified, a comma must be used as a separator between variables.

A step by step example in Chapter 6 explains how data is entered through the INBUFFER, and input queue.

The CWRITE Statement

The CWRITE statement outputs the data from BASIC strings to the output queue for transmission over the datacomm link. The CWRITE statement is also used to insert user defined line separators and control the Request-to-Send line. When CWRITE is executed, the Request-to-Send modem line is activated (see the next flow chart).

Syntax:

CWRITE select code ; output list

select code: The valid select code (channel) for your datacomm operation.

output list: The output list consists of any or all of the following.

- One or more string expressions.
- One or more string variables.
- ENDLINE
- EOT

string expression: Any valid string expression.

string variable: Any valid string variable.

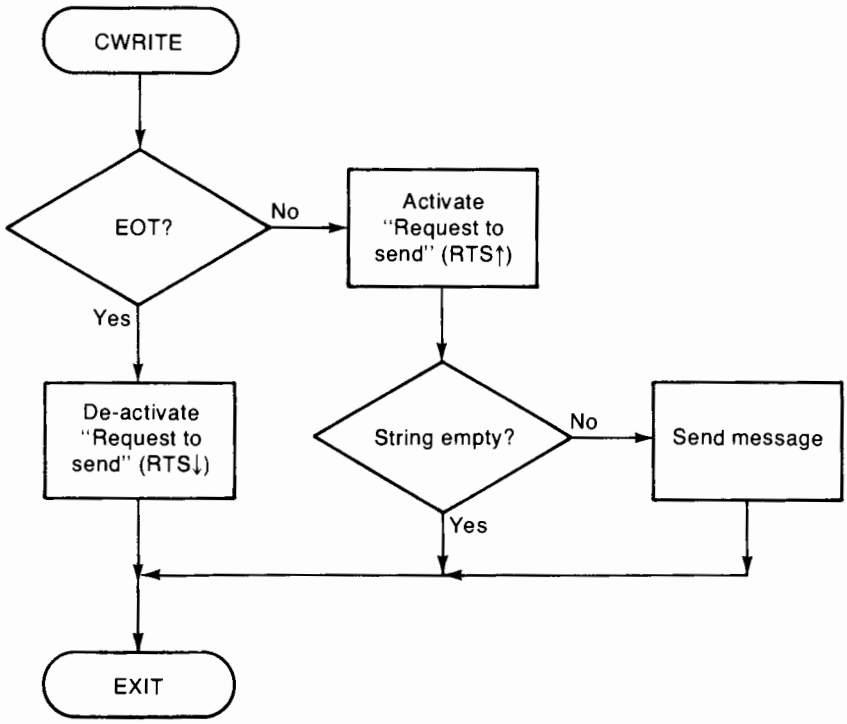
ENDLINE: The CWRITE ENDLINE statement outputs the user defined line separator (OUTSEP =) as part of the message text. The ENDLINE specifier is used to indicate the end of each line of data that is transmitted. No automatic endline sequence is output, as with a PRINT statement.

EOT: The CWRITE...EOT statement is used with half duplex operations only. The CWRITE...EOT statement, when executed, sets the RTS (Request-to-Send) modem line to the inactive (0) state (see Modem Handshake, Chapter 2).

NOTE

If the host sends a PREDICTABLE prompt, you should check for CSTATUS element 2 True (1) before executing a CWRITE to the host. The host sends a prompt sequence (which sets element 2 True) only when your data can be accepted. Failure to do so may result in a loss of data.

CWRITE Flowchart



Chapter 4
Table of Contents

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !      ************************************************************
80   !      !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                  ! Dimension the string variables
110       CDISCONNECT 4                  ! Ensure you are disconnected
120       CCOM 1500                      ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                 ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF       ! Link is connected here
150       CCONTROL 4;ACK ON              ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit     ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive      ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE               ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                     ! Wait here for an interrupt
230   !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN ! Check for a PROMPT before continuing
260       CREAD 4;A$                    ! Get the visual prompt character
270       PRINT A$;                     ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$        ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE           ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                    ! Read the data from the host
350       PRINT A$                      ! Print the data from the host
360  GOTO Receive                       ! Go check for more input data
370  !
380  END
```

Chapter 4
Interrupts and Control

The following is a summary of the information provided in this chapter.

Statement or Function	Parameter or Option	Default
ON INT# select code	Priority GOTO GOSUB CALL	
OFF INT# select code		
CCONTROL select code	DRIVER1 ON or DRIVER1 OFF	DRIVER1 OFF
	DRIVER2 ON or DRIVER2 OFF	DRIVER2 OFF
	DRIVER3 ON or DRIVER3 OFF	DRIVER3 ON
	READALL ON or READALL OFF	READALL OFF
	XON or ACK ON or ACK OFF	ACK OFF
	RESET	
	SUSPEND	
	BREAK	
	INTMASK = expression	6

The highlighted program lines on the facing page show typical use of these statements.

The ON INT# Statement

The ON INT# statement provides for end-of-line branching when pre-defined conditions occur. This is accomplished by the ON INT# statement which specifies the branch and the priority level for the branch. With the exception of ON INT...CALL, the ON INT statement is effective only within the current program segment. Interrupt operations are explained in detail in the I/O ROM Programming Manual supplied with your Desktop Computer.

Syntax:

ON INT# select code [, priority]branch

select code: The valid select code (channel) for your datacomm operation.

NOTE

To interrupt from input queue status (complete message or ALERTN characters present), the ODD select code for the interrupt MUST be specified. All other interrupts are generated for the EVEN select code only. See the CSTATUS statement, Chapter 5.

---

priority: The optional priority parameter specifies the priority level (1-15) assigned to end-of-line branches from the specified select code. If a GO SUB or CALL is specified, the system priority level is set to the priority level specified by the ON INT statement, then restored to the previous system priority level upon return from the subroutine or subprogram. End-of-line branches of a higher priority are granted end-of-line service within the interface service subroutine or subprogram, but lower priority branches are not serviced until the system priority level drops lower than the priority level of the requesting interface. In datacomm programs, the ODD select code should normally have a higher priority than the EVEN select code. This prevents transmitting before you have processed all input data from the host. The default system priority is 0.

branch: Allowable branch parameters are:

- GO TO line ID
- GOSUB line ID
- CALL name

line ID: The line ID parameter specifies the line identifier of the service routine or subroutine.

name: The name parameter specifies the name of the service subprogram.

GO TO : An ON INT statement specifying a GO TO is active only within the program segment in which it is executed. Transfer to a different program segment temporarily disables the ON INT...GO TO...conditions. When a program segment containing the ON INT...GO TO... statement is re-entered, the ON INT conditions are automatically reinstated. An end-of-line branch associated with an ON INT...GO TO...statement does not alter the system priority level or exit the current program segment.

GO SUB : An ON INT statement specifying a GO SUB is active only within the program segment in which it is executed. Transfer to a different program segment temporarily disables the ON INT...GO SUB...conditions. When the program segment containing the ON INT...GO SUB...statement is re-entered, the ON INT conditions are automatically reinstated. An end-of-line branch associated with an ON INT...GO SUB...statement sets the system priority level to the statement's priority level (the priority parameter) but does not exit the current segment.

CALL : An ON INT statement specifying a CALL is active regardless of the current program segment. An end-of-line branch associated with an ON INT...CALL...statement remains active until it is redefined by another ON INT statement for the same select code or until it is cancelled by an OFF INT statement for the same select code.

Two system statements, ENABLE and DISABLE affect ON INT end-of-line branches as well as ON KEY end-of-line branches. The DISABLE statement inhibits end-of-line branching until an ENABLE statement is executed. If an interrupt is logged in while end-of-line branches are inhibited, the end-of-line branch for that interrupt is taken when the ENABLE statement is executed.

NOTE

When implementing datacomm operations under interrupts, the ON KEY[1] statement should have the HIGHEST priority or you may never be able to send a BREAK.

---

1 See the ON KEY statement in your Desktop Computer Operating and Programming Manual.

The OFF INT Statement

The OFF INT statement enables you to disable the current ON INT condition for the specified select code. The OFF INT statement is only in effect within its program segment. Interrupts are not logged while OFF INT is in effect, they are lost. Interrupts logged while a disable is in effect are processed when a subsequent ENABLE is executed.

Syntax:

OFF INT # select code

select code: The valid select code (channel) for your datacomm operation.

The CCONTROL Statement

The CCONTROL statement provides control features for the datacomm operation. CCONTROL also provides the INTMASK specification so that you can specify the CSTATUS conditions that generate interrupts. The CCONTOL statement can be executed from a program or from the keyboard in the live keyboard mode.

Syntax:

CCONTROL select code ; command(s)

select code: The valid select code (channel) for your datacomm operation.

command(s): The command list may consist of one or more of the following:

- DRIVER 1 ON or OFF
- DRIVER 2 ON or OFF
- DRIVER 3 ON or OFF
- READALL ON or OFF
- XON or ACK ON or ACK OFF
- RESET
- SUSPEND
- BREAK
- INTMASK = expression

The following DRIVER commands provide direct control of three RS232 hardware drivers on the 98046 interface which enable you to control secondary[1] channel modem signals, and select the alternate modem speed (two speed modems only). The modem lines that are controlled depend on the option cable supplied with the 98046 interface card. The control lines and connector pin numbers are shown in the following table. It is possible to connect these hardware drivers to other RS 232 lines for special applications by modifying the 98046 cable. See the 98046 interface Installation and Service Manual (P/N 98046-90030).

1 An example secondary channel program is provided in Appendix E.

Command	Cable Option		Default
	Standard	Option1	
DRIVER1 ON or DRIVER1 OFF	Not Used	Not Used	**DRIVER1 OFF**
DRIVER2 ON or DRIVER2 OFF	Secondary Carrier Detect(Pin 12)	Secondary Request to Send(Pin 19)	**DRIVER2 OFF**
DRIVER3 ON or DRIVER3 OFF	Not Used	Data Rate Select(Pin 23)	**DRIVER3 ON**

You should check the status of the appropriate modem lines (see CSTATUS elements 11, 12, 13 and 14) before and after executing a CCONTROL...DRIVER ON/OFF statement in order to ensure that the modem line responses are as expected. Failure to ensure correct modem line status prior to CWRITE operations can produce unexpected loss of data. These lines are always reset to default condition by the CCONNECT statement.

READALL ON

The READALL ON command causes all characters, including control characters, to be transferred to the BASIC strings by the CREAD statement. Parity and framing errors are NOT changed to the underscore character.

READALL OFF

Normally (READALL OFF) the line separators, prompts, and the ENQ, NULL, and DEL characters are not transferred to the BASIC string variable. Characters with parity or framing errors are converted to underscores.

Default: = **READALL OFF**

ACK ON

ACK ON and XON are FULL DUPLEX options. ACK ON enables the ACKnowledge (CHR$(6)) response to an ENQuiry (CHR$(5)) from the host computer. The ACK response means that the Desktop Computer can accept at least 140 characters from the host. This is the common protocol for HP1000 and HP3000 computers.

X ON

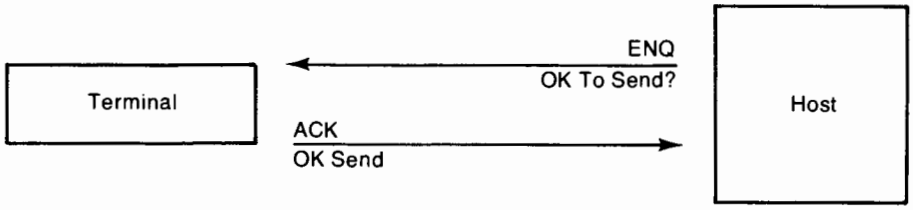
XON allows the Desktop Computer to send a "busy" indication in order to prevent further data transmission by the host. When there are less than 140 empty character spaces remaining in the input buffer, the interface automatically transmits a DC3 (CHR$(19)) character to suspend further transmission from the host. When the input buffer is completely empty, the interface automatically transmits a DC1 (CHR$(17)) to the host to resume transmission.

ACK OFF

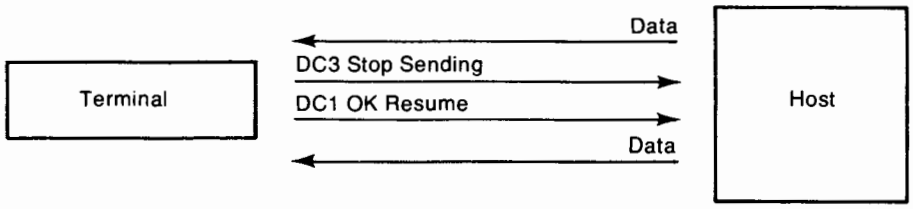
ACK OFF disables ACK ON and XON. No response or "busy" indication is sent to the host. The following drawings show typical ENQ-ACK and DC1-DC3 interactions.

Default: = **ACK OFF**

ENQ/ACK Handshake



XON/OFF Handshake



RESET

CCONTROL RESET is normally used during error recovery (see Chapter 8). When CCONTROL...RESET is executed, the specified channel is returned to the inactive state, all pending datacomm activity is aborted, but all modem signals remain active. The following are reinitialized:

- Input Queue.
- Output Queue.
- INBUFFER.
- TBUFFER.
- Interface Card.

BREAK

CCONTROL BREAK causes the modem to output the break sequence to the remote device. The break sequence consists of a 500 millisecond "mark" followed by a 500 millisecond "space."[1] The "break" signal is normally interpreted by the host as a signal to halt current datacomm activity.

1 Mark and Space are explained in Chapter 1.

SUSPEND

CCONTROL...SUSPEND prevents further datacomm activity for the specified channel (select code). All modem signals remain active. Subsequent CREAD statements can be executed to clear the input queue but further data communication activity is not allowed until a CCONTROL...RESET has been executed. The trace buffer is NOT initialized. The NO ACTIVITY remains active and a NO ACTIVITY timeout is executed when the NO ACTIVITY parameter is exceeded. CCONTROL...SUSPEND may be executed prior to dumping line trace information to ensure that the channel is suspended.

INTMASK = expression

The INTMASK parameter enables you to specify the CSTATUS conditions that generate an interrupt. The ON INT statement must be executed before an interrupt is generated. All interrupts are generated for the EVEN select code except element 1 (input queue status), which generates an interrupt for the ODD select code.

expression:

The expression is the integer value of the summed total of the CSTATUS bits that you specify. See following table:

Array Element	Bit Value	Meaning
A (0)	1	Internal datacomm error.
A (1)	2	Input queue status.
A (2)	4	Prompt detection status.
A (3)		Cannot be specified to generate an interrupt.
A (4)		Cannot be specified to generate an interrupt.
A (5)	32	Datacomm suspended.
A (6)	64	Parity or framing error
A (7)		Not used.
A (8)	256	Trace buffer full.
A (9)	512	Break received.
A (10)		Not used.
A (11)	2048	Receiver 1 — Option cable dependent (see CSTATUS, Chapter 5).
A (12)	4096	Receiver 2 — Option cable dependent (see CSTATUS, Chapter 5).
A (13)	8192	Receiver 3 — Option cable dependent (see CSTATUS, Chapter 5).
A (14)	16 384	Receiver 4 — Option cable dependent (see CSTATUS Chapter 5).

CSTATUS is described in Chapter 5.

For example:

You want to specify an interrupt for select code 6 if there is an internal datacomm error (A(0)) or a parity or framing error (A(6)). The value for INTMASK is 65, 1+64.

```
CCONTROL 6; INTMASK = 65
```

Select Code 6

Set INTMASK to 65
1(for A(0)) + 64(for A(6))

Default: **INTMASK** = **6**(element 1 and element 2)

Chapter 5
Table of Contents

Status

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !      ************************************************************
80   !      !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                    ! Dimension the string variables
110       CDISCONNECT 4                    ! Ensure you are disconnected
120       CCOM 1500                        ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                   ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF         ! Link is connected here
150       CCONTROL 4;ACK ON                ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit       ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive        ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE                 ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                       ! Wait here for an interrupt
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN    ! Check for a PROMPT before continuing
260       CREAD 4;A$                       ! Get the visual prompt character
270       PRINT A$;                        ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$           ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE              ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN    ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                       ! Read the data from the host
350       PRINT A$                         ! Print the data from the host
360  GOTO Receive                          ! Go check for more input data
370  !
380  END
```

Chapter 5
Status

The following is a summary of the information provided in this chapter:

Statement or Function	Parameter or Option
CSTATUS select code	integer array
CSTAT(select code,integer array element)	

The highlighted program lines on the facing page show typical use of these statements.

Introduction

The CSTATUS statement and the CSTAT function enable you to monitor the various states of the protocol machine[1], memory allocation, and modem signals.

1 The protocol machine is explained in Chapter 5, Data Communication Basics Manual (98046-90005).

The CSTATUS Statement

The CSTATUS statement returns status information relating to the entire datacomm operation. This status information is entered into a sixteen element integer array variable specified by the CSTATUS statement. CSTATUS cannot be executed unless CMODEL ASYNC and CCONNECT have been executed.

Much of the information provided by the CSTATUS statement is intended for use in troubleshooting datacomm problems. Details about troubleshooting are found in Chapter 9.

Syntax:

```
CSTATUS select code ; integer array
```

select code: The valid select code (channel) for your datacomm operation.

integer array: The integer array MUST conform to the following specifications:

- MUST be an INTEGER array.
- The FIRST element MUST be element 0.
- The array MUST be ONE dimensional.
- The array MUST contain a minimum of 16 elements (0 through 15).

The following statement shows how to set up the integer array.

```
INTEGER A(0:15)
```

The following statement enters the status information into the CSTATUS array.


```
CSTATUS 4;A(*)
```

The Array Elements

The following section explains the information that is returned into the integer array elements when the CSTATUS statement is executed.

The value required by the CCONTROL...INTMASK statement to generate an interrupt for the various CSTATUS elements is also shown. To specify two or more elements to generate an interrupt, simply add the corresponding intmask values together and enter the sum into the CCONTROL...INTMASK statement. CCONTROL...INTMASK is explained in Chapter 4.

The explanations of the various CSTATUS elements assume that the appropriate interrupt conditions (ON INT, INTMASK) have been specified in order to generate the interrupts as described.

Element 0 Intmask Value = 1

Element 0 provides datacomm errors. The value returned indicates the error that has occurred. Only the first error that occurs is entered into element 0. No subsequent errors are entered into element 0 until the first error is cleared. The occurrence of any of these errors suspends all datacomm activity for this channel. If bit 0 is not specified to generate an interrupt, the computer beeps and displays the appropriate error message. These errors are trapped when bit 0 is specified to generate an interrupt for the even select code and an error trapping routine is used (see CSTATUS Error Trapping, Chapter 9). Element 0 is NOT reset to 0 when CSTATUS is executed. Errors are cleared by executing CCONTROL...RESET.

Error Number	Meaning
10	Timeout before connection
11	Clear to Send line false or missing clock
100	Channel MEMLIMIT overflow
102	Input buffer overflow
103	Internal buffer overflow
104	Autodisconnect forced
106	NOACTIVITY timeout
200	98046 buffer overflow

These errors are explained in detail in Chapter 9.

NOTE

Error 10 (Timeout before connection) means the CCONNECT statement has been executed, but the actual channel connection has not been established (see The 25 Second Timeout, Chapter 2). The routine for trapping error 10 is explained in Chapter 9.

Element 1 Intmask Value = 2

Element 1 provides the status of the input queue. When a complete input message (message terminator detected (INSEP)), or ALERTN characters have been input without a message terminator, the value of element 1 is set true (1) and an interrupt is generated for the ODD select code. Element 1 is reset to 0 when CREAD is executed and there are no message terminators or fewer than ALERTN characters remaining in the input queue.

NOTE

This is the only CSTATUS element that generates an interrupt for the ODD select code.

Value	Meaning
0	There is no complete line (no INSEP), or there are fewer than ALERTN characters in the input queue.
1	There is a complete line (INSEP detected), or there are ALERTN characters in the input queue.

Element 2 Intmask Value = 4

Element 2 provides prompt sequence detection status. When a prompt sequence (PROMPT) is detected, the value of element 2 is set true (1) and an interrupt is generated for the EVEN select code. Reception of a prompt indicates that the host is ready to receive data. Element 2 is reset to 0 when CWRITE is executed.

Value	Meaning
0	No prompt has been received, or no prompt has been received since last CWRITE statement.
1	Prompt character (PROMPT) has been detected.

Element 3 Does not provide interrupt.

Element 3 indicates whether the most recent CREAD operation terminated with an input line separator (INSEP) sequence. Element 3 is reset when a subsequent CREAD is executed.

Value	Meaning
0	The most recent CREAD was not terminated with an INSEP sequence.
1	The most recent CREAD operation was terminated by an INSEP sequence.

Element 4 Does not provide interrupt.

Element 4 provides memory allocation information. Two types of information are provided, depending upon when element 4 is accessed. After the CMODEL statement has been executed and prior to CCONNECT execution, the value returned by element 4 indicates the amount of CCOM memory (in words) that is required for the CMODEL. After CCONNECT has been executed, the value indicates the approximate remaining queue capacity (in words) of the channel.

Element 5 Intmask Value 32

Element 5 provides link status information. In cases where the value of element 5 is 1 or 2, no interrupt is generated. When the value of element 5 is 3 (datacomm operation suspended), the interrupt is generated for the EVEN select code.

Value	Meaning
1	CMODEL statement executed, but no CCONNECT
2	The line is idle (initial state after CCONNECT is executed).
3	Datacomm operation is suspended.

Element 6 Intmask Value 64

Element 6 provides a count of the errors detected in the received data since the most recent CCONNECT, CSTATUS, or CCONTROL RESET statement was executed. When a parity error or a framing error is detected, the value of element 6 is incremented and an interrupt is generated for the EVEN select code. The character that caused the error is converted to the underscore character when it is moved from the INBUFFER to the input queue. Element 6 is reset to zero when CSTATUS is executed.

Element 7

Element 7 is not used by CMODEL ASYNC.

Element 8 Intmask Value 256

Element 8 provides trace buffer status information. Element 8 monitors the trace buffer status in the STOP FULL or STOP ERROR modes[1] only. The value of element 8 is set to 1 when there is no room remaining in the trace buffer for data, at which time tracing is stopped. A value of 0 indicates that the trace buffer is not full. When the value is set to 1, an interrupt is generated for the EVEN select code. Element 8 is reset to 0 when CTRACE...STOPFULL[1] or CTRACE...WRAP[1] is executed.

Value	Meaning
0	There is room in the trace buffer for trace data.
1	The trace buffer is full or a parity or framing error has been detected in the input data.

1 See the CTRACE statement, Chapter 8.

Element 9 Intmask Value 512

Element 9 provides break detection information. When a break is detected during a receive operation, element 9 is set to 1 and an interrupt is generated for the EVEN select code. Element 9 is reset to 0 when CSTATUS is executed. When you detect a break, you may detect a parity error if a subsequent line turn around is detected.

Value	Meaning
0	No break detected.
1	Break detected in receive operation.

Element 10

Element 10 is not used by CMODEL ASYNC.

Elements 11, 12, 13 and 14

Elements 11, 12, 13 and 14 enable you to monitor the status of specific modem signals. The specific modem signal that is monitored depends on the cable option supplied with the 98046 interface card. The following table shows the modem lines that are monitored by these CSTATUS elements and their associated intmask values.

CSTATUS ELEMENT	Modem Line (Standard Cable)	Modem Line (Option 1 Cable)	INTMASK Value
11	DATA TERMINAL READY	DATA SET READY	2048
12	SECONDARY REQUEST TO SEND	SECONDARY DATA CARRIER DETECT	4096
13	REQUEST TO SEND	DATA CARRIER DETECT	8192
14	DATA CARRIER DETECT	CLEAR TO SEND	16384

A value of 1 for the CSTATUS element indicates that the associated modem line is active (true) and a value of 0 indicates that the associated modem line is inactive (false). When a modem line changes state (false to true or true to false), an interrupt is generated for the EVEN select code if interrupts are enabled and the intmask value for that modem line has been included in the CCONTROL...INTMASK statement.

Element 15

Element 15 is not used by CMODEL ASYNC.

The CSTAT Function

The CSTAT function is provided to enable you to quickly select a specific CSTATUS array element and return the value contained in that element. The CSTAT does not update or reset the CSTATUS elements. CSTAT may be executed prior to executing CCONNECT. Executing a CSTAT statement prior to a CMODEL statement returns a value of − 1 for ALL elements.

Syntax:

```
CSTAT(select code, integer array element)
```

select code:	The valid select code (channel) for your datacomm operation.
integer array element:	The element is an integer (0 through 15) that corresponds to the CSTATUS element from which the data is obtained.

For example, to check for a prompt received on select code 6 execute:

```
CSTAT (6,2)
```

If the value returned is 1, then a prompt sequence has been detected.

If the value returned is 0, then no prompt sequence has been detected.

Chapter 6
Table of Contents

Transfer Control

Chapter 6
Transfer Control

Using CSTAT for Program Control

The CSTAT function is used to provide all of the information necessary for orderly transfer of information between your Desktop Computer and the host.

CSTAT(select code,1) is true (1) when:

- An INSEP sequence is in the input queue.
- The number of characters specified by the ALERTN= option are in the input queue.

CSTAT (select code,2) is true (1) when:

- The prompt sequence specified by the PROMPT = option is received from the host.

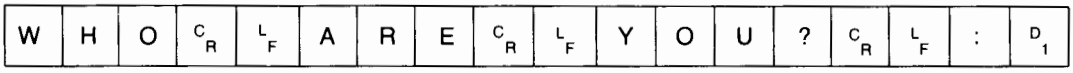
Data Input

The following section explains how data is moved from the interface through the input buffer, the input queue and into the BASIC strings.

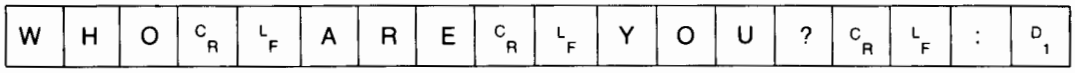
Here is the sequence:

- Data is moved from the interface to the input buffer by the drivers.
- Data is moved from the input buffer to the input queue by the protocol machine.
- Data is moved from the input queue to the BASIC strings by the CREAD statement.

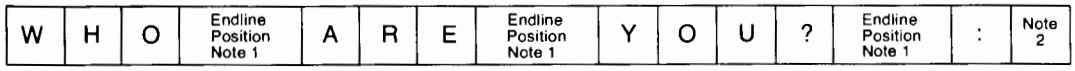
Assume that the following data is present in the interface buffer:



The data is moved to the input buffer in the following form:



Assume that there is sufficient space in the input queue for all of the data. The p-machine then moves the data into the input queue in the following form:



Data is checked for errors, prompts, input line separators, etc. as it is entered into the input queue and the appropriate CSTATUS elements are updated at this time.

Note 1 The actual endline (CR/LF) sequences are not entered into the input queue and no queue space is occupied, but their positions are remembered. When an endline sequence is detected CSTAT (4,1) is set true. CSTAT (4,1) is not reset until a CREAD is executed and all endline sequences are accounted for.

Note 2 The prompt (D1) sequence is not entered into the input queue. When a prompt sequence is detected CSTAT (4,2) is set true. CSTAT (4,2) is reset when a CWRITE is executed.

Data is moved from the input queue into BASIC strings by employing one of the following methods:

- Line mode input.
- Character mode input.

Line Mode Input

The following explanation shows how data is entered into the BASIC strings on a line-by-line basis.

The following parameters are assumed:

- DIM A$ [80]
- INSEP = CHR$ (13) & CHR$(10)
- OUTSEP = CHR$ (13)
- PROMPT = CHR$ (17)

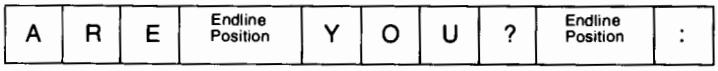
Execute the following statements:

```
100  CREAD 4;A$                          ! Read the input up to the INSEP
110  PRINT A$                            ! Print the data that was input
120  PRINT "CSTAT(4,1)=";CSTAT(4,1)      ! Print the value of CSTAT(4,1)
130  PRINT "CSTAT(4,2)=";CSTAT(4,2)      ! Print the value of CSTAT(4,2)
```

Here is the printout:

```
WHO
CSTAT(4,1)= 1
CSTAT(4,2)= 1
```

The data remaining in the input queue is:

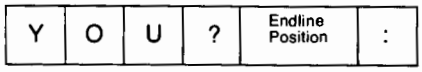


Execute the previous four program statements.

Here is the printout:

```
ARE
CSTAT(4,1)= 1
CSTAT(4,2)= 1
```

The data remaining in the input queue is:



Execute the previous four program lines.

Here is the printout

```
YOU?
CSTAT(4,1)= 0
CSTAT(4,2)= 1
```

CSTAT (4,1) is reset to 0 when the CREAD statement is executed because there are no subsequent endline positions remaining in the input queue.

The data remaining in the input queue is:

Execute the previous four program lines.

Here is the printout:

```
:
CSTAT(4,1)= 0
CSTAT(4,2)= 1
```

the input queue is now empty.

Execute the following program lines.

```
CWRITE 4;ENDLINE
PRINT "CSTAT(4,1)=";CSTAT(4,1)
PRINT "CSTAT(4,2)=";CSTAT(4,2)
```

Here is the printout:

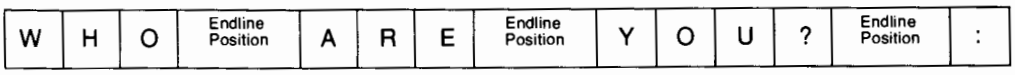
```
CSTAT(4,1)= 0
CSTAT(4,2)= 0
```

CSTAT (4,2) is reset to 0 when the CWRITE statement is executed. CSTAT (4,1) remains 0 because no data has been sent to the input queue at this time.

Character Mode Input

The next section explains how to input data into the BASIC strings one character at a time.

Here is the same input queue data that was used for the line mode explanation.



The following parameters are changed:

- DIM A$[1]
- ALERTN = 1

Execute the following statements:

```
100  CREAD 4;A$                          ! Read the input
110  PRINT A$                            ! Print the data that was input
120  PRINT "CSTAT(4,1)=";CSTAT(4,1)      ! Print the value of CSTAT(4,1)
130  PRINT "CSTAT(4,2)=";CSTAT(4,2)      ! Print the value of CSTAT(4,2)
140  PRINT "CSTAT(4,3)=";CSTAT(4,3)      ! Print the value of CSTAT(4,3)
```

Here is the printout:

```
W
CSTAT(4,1)= 1
CSTAT(4,2)= 1
CSTAT(4,3)=0
```

CSTAT (4,1) is true because:

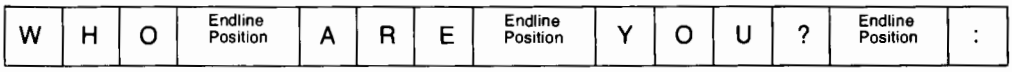
- Endline sequences remain in the queue.
- There are ≥ ALERTN characters remaining in the queue (in this case ALERTN = 1).

CSTAT (4,1) remains true until both of the above statements are false, i.e., there are NO endline sequences remaining in the queue AND there are fewer than ALERTN characters remaining in the queue.

CSTAT (4,2) Identical to the line mode explanation for CSTAT (4,2).

CSTAT (4,3) is used to indicate that an endline sequence has been passed by the most recent CREAD. CSTAT (4,3) is set when a CREAD is executed and an endline sequence is encountered. CSTAT (4,3) is reset when a subsequent CREAD statement is executed and no endline sequence is encountered.

Assume the following is contained in the input queue.



Run the following program statements:

```
300  CREAD 4;A$                          ! Read 1 character into A$
310  IF CSTAT(4,3) THEN 340              ! Check for INSEP indication
320  PRINT A$;                           ! Print character without CR/LF
330  GOTO 300                            ! Get another character
340  PRINT A$                            ! Print character with CR/LF
350  PRINT "CSTAT(4,3)=";CSTAT(4,3)      ! Print CSTAT(4,3) value
360  IF CSTAT(4,1) THEN 300              ! If more data, return and get it
```

Here is the printout:

```
WHO
CSTAT(4,3)= 1
ARE
CSTAT(4,3)= 1
YOU?
CSTAT(4,3)= 1
```

Line 300 — Enters a character from the queue.
Line 310 — Checks CSTAT (4,3) condition:
- If the true-jump to endline sequence. (Line 340)
- If false-continue

Line 320 — Prints character without endline sequence.
Line 330 — Go back and get another character.
Line 340 — Print last character in the line, endline sequence.
Line 350 — Print CSTAT (4,3) value on a new line.
Line 360 — Checks
- If true — there is more data in the queue — jump back to the beginning (Line 300).
- If false — continue.

When this program sequence is complete, the prompt could then be serviced in the same manner as described for the line mode input.

Chapter 7
Table of Contents

The First Connection

Chapter 7
The First Connection

Introduction

This chapter explains how to connect to a host computer. Examples show how to define each datacomm statement that is needed to establish a connection. A simple example program is also given. Actual log-on procedures are explained and typical log-on problems are explained.

Memory Allocation

The first question that must be answered is: "How much memory must be allocated for the datacomm program?" For a "first connection," it is suggested that the defaults for the CMODEL ASYNC memory allocation parameters be assumed. These defaults are:

- MEMLIMIT = 500
- INBUFFER = 260
- TBUFFER = 130

If these defaults are assumed, the memory allocation parameters can be omitted from the CMODEL ASYNC statement and the following CCOM statement can be used:

```
CCOM 1500
```

Protocol Parameters

The next set of questions help you determine which CMODEL ASYNC and CCONNECT protocol parameters to specify. You may omit each parameter if it requires a value that is the same as the default provided. These parameters can be divided into three groups:

- What the host expects from you (transmit parameters).
- What the host sends to you (receive parameters).
- Both (transmit and receive parameters).

What the Host Expects from You

- STOPBITS — How many stop bits are required?
- OUTSEP — What line terminator does the host expect?
- GAP — What spacing between characters is required?

What the Host Sends to You

- PROMPT — What predictable prompt sequence (if any) does the host send?
- INSEP — What predictable line terminator sequence (if any) does the host send?

Both

- CHARLENGTH — How many bits make up a complete character? Remember, CHARLENGTH does NOT include start, stop, or parity bits.
- CHECK — What type parity checking is used?
- Are HALF or FULL DUPLEX transfers implemented? You must first determine if a modem is to be used; and if used, whether it is a half or full duplex modem (see the next page, "Connection").

An additional parameter, ALERTN, is provided to allow you to process input data on single character or group of characters basis, rather than a line-by-line basis. If the host sends no predictable prompt or endline sequences, the ALERTN specifier should be used.

Default
ALERTN = 32767

Connection

The next group of questions determine which CCONNECT options are used.

- SPEED — What is the data transfer rate?
- HANDSHAKE — Are the modems being used?

Two additional parameters, LOST CARRIER and NO ACTIVITY allow you to specify an automatic disconnect from the link if the Data Set Ready (and Data Carrier Detect)[1] modem signal is lost or no data transfer activity occurs for a specified period of time.

Statement	Parameter	Default	Your Value
CMODEL ASYNC	MEMLIMIT=	500	
	INBUFFER=	260	
	TBUFFER=	130	
	INSEP=	CHR$(13)& CHR$(10)	
	OUTSEP=	CHR$(13)	
	PROMPT=	CHR$(17)	
	ALERTN=	32767	
	CHECK=	3	
	CHARLENGTH=	7	
	STOPBITS=	1	
	GAP=	0	
	HALF DUPLEX or FULL DUPLEX	FULL DUPLEX	
CCONNECT	SPEED=	300	
	INSPEED=	Note 1	
	OUTSPEED=	Note 1	
	EXTERNAL	Note 1	
	LOST CARRIER=	400	
	NO ACTIVITY=	600	
	HANDSHAKE ON or HANDSHAKE OFF	HANDSHAKE ON	

Note 1. The default SPEED=300 parameter disables INSPEED, OUTSPEED, and EXTERNAL.

1 Data Set Ready and Data Carrier Detect are both monitored in FULL DUPLEX operation.

Here's how to select the appropriate CMODEL ASYNC statement. Assume the following answers:

• 1 stop bit required.	Default
• CR line terminator required.	Default
• No spacing between characters required.	Default
• DC1 prompt sent by the host.	Default
• CR/LF line terminator sent by the host.	Default
• 7 bit character length required.	Default
• Odd parity checking implemented.	Default
• Full duplex transfers are implemented.	Default
• Default ALERTN used.	Default

If the defaults for the memory allocation parameters are assumed, then the following statement can be used:

```
CMODEL ASYNC, select code
```

Here's how to select the appropriate CCONNECT statement.
Assume the following answers:

• Data transfer rate is 300.	Default
• Direct connection is used (no modem)	HANDSHAKE OFF

If the default NO ACTIVITY is assumed (HANDSHAKE OFF disables LOST CARRIER) then the following statement can be used:

```
CCONNECT select code; HANDSHAKE OFF
```

Host Computer Handshake

The answer to the next question determines whether a CCONTROL statement is needed and, if needed, which options to specify. The answer to this question is obtained from the systems analyst responsible for the host computer.

- Does the host computer require ENQ/ACK or XON/XOFF handshakes?

ENQ/ACK and XON/XOFF handshakes are described in Chapter 4.

If ENQ-ACK handshake is required by the host computer (HP3000 and HP1000 computers use ENQ-ACK handshake) the following statement must follow the CCONNECT statement.

```
CCONTROL select code; ACK ON
```

Log-On

Before you attempt a log-on procedure, you must also obtain the actual log-on procedure from the system analyst. The log-on procedure should contain the following:

- Actual syntax required.
- Account information (name and/or number).
- Password (if used).

The following example assumes that you are connected to an HP 3000 computer.

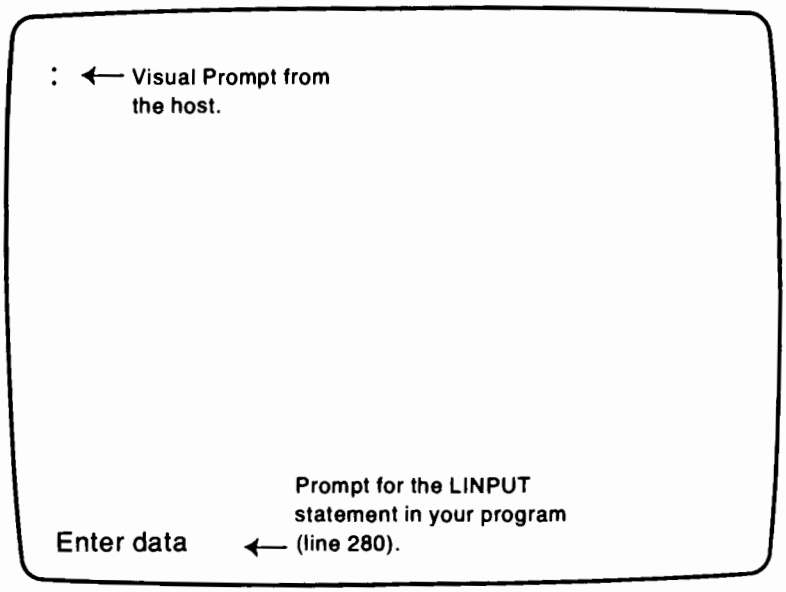
Key the following program into your Desktop Computer.

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !      ********************************************************
80   !      !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                 ! Dimension the string variables
110       CDISCONNECT 4                 ! Ensure you are disconnected
120       CCOM 1500                     ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF      ! Link is connected here
150       CCONTROL 4;ACK ON             ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit    ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive     ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE              ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                    ! Wait here for an interrupt
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN ! Check for a PROMPT before continuing
260       CREAD 4;A$                    ! Get the visual prompt character
270       PRINT A$;                     ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$        ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE           ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                    ! Read the data from the host
350       PRINT A$                      ! Print the data from the host
360  GOTO Receive                       ! Go check for more input data
370  !
380  END
```

If you are using a modem to connect to a host computer, you must first establish the modem connection[1] to the host prior to running the datacomm program on your Desktop Computer.

1 Contact your systems analyst for appropriate modem dial-up and connection procedures.

Press **RUN** on the Desktop Computer. The following should appear on your display:



Press **STOP** on the Desktop Computer.

Key the following statement into your Desktop Computer.

```
CDUMP 4;FORCE[1]
```

Press **EXECUTE**

The following should appear on your display.

```
DUMP FORCED
LINE TRACE    04
IN      |I35|R0 |    |R0 |    |   |IC0|I08|E0 |IC0|    |   |IC0|CR |IC0|LF |IC0| :
TX      |   |   |    |   |    |CR |   |   |   |   |    |AK |   |   |   |   |   |

IN  IC0|D1 |IC0
TX     |   |
```

1 ENDLINE sent to the host (Line 200)
2 Host sends ENQ
3 ACK sent to the host (Line 150)
4 Host sends visual prompt
5 Host sends prompt sequence (DC1)

1 The CDUMP statement is presented here to show you how to determine whether the proper PROMPT, ENQ/ACK and endline sequences are actually being transferred. The CDUMP statement and the associated control codes that appear on your display are explained in Chapter 8.

The previous display shows how a normal connection sequence appears on the CRT when the CDUMP statement is executed.

```
DUMP FORCED
LINE TRACE    04
IN      |I35|RO |    |RO |    |
TX      |   |   |    |   |  • |CR
                                1
```

1 Send ENDLINE to the host (Line 200)
Host does not answer

This display shows how a connection sequence that failed appears on the CRT when the CDUMP statement is executed. Check the following parameters in your program:

- SPEED
- CHARLENGTH
- PARITY
- OUTSEP
- HALF DUPLEX or FULL DUPLEX
- HANDSHAKE ON or OFF

When the appropriate visual prompts appear on the CRT, key your log-on message[1] into the desktop computer.

The following should appear on your display:

```
                1
   ┌──────────────────────────┐
   :HELLO BILLD.MANUALS

2 ┌CPU=1. CONNECT=1. FRI, JAN 11, 1980,  9:28 AM
  └
   HP3000 / MPE III B.01.00.  FRI, JAN 11, 1980,  9:28 AM
```

1 Echo of log-on message
2 Computer I.D.

If your log-on echo looks like the following, you have parity or overrun errors.

```
:H_____B___D.MANUA_S
  1    1       1
```

1 The characters that are in error are converted to the underscore

1 The log-on syntax must be obtained from your systems analyst. The examples contained in the manual use a hypothetical account. The echo and computer identification shown in this manual are from that account.

You have established communications with the host computer. You can now access the internal subsystems provided by the host (e.g. EDITOR, BASIC, etc.). When you are ready to disconnect from the host, key the following into the Desktop Computer in response to the ":" prompt:

```
BYE
```

Press **CONTINUE**

The following should appear on your display:

```
1 { BYE
2 { CPU=1. CONNECT=1. FRI, JAN 11, 1980,  9:35 AM
```

1 Sign off message echo
2 Computer sign off message

Press **STOP**

Key the following into the Desktop Computer:

```
CDISCONNECT 4
```

Press **EXECUTE**

Your datacomm link is now disconnected.

If you do not execute a CDISCONNECT statement, an automatic disconnect is executed by the NO ACTIVITY timer after 10 minutes have elapsed. When the NO ACTIVITY timeout is executed, the Desktop Computer beeps and the following error message appears on the display:

```
*04 NOACTIVITY TIMEOUT
```

Chapter 8
Table of Contents

CTRACE and CDUMP

Chapter 8
CTRACE and CDUMP

Introduction

This section of the manual explains the CTRACE and CDUMP statements. These statements are provided by the Basic Data Communications ROM as troubleshooting aids. Examples at the end of this chapter show how the various CTRACE and CDUMP specifiers affect the printout. A table of the various control codes is also provided.

CTRACE

The CTRACE statement allows you to control the contents and operating mode of the trace buffer (TBUFFER). Contents options are provided by the CTRACE statement to enable you to select the data that is entered into the trace buffer. Operating mode options allow you to specify continuous tracing or to direct the trace buffer to stop tracing and generate an interrupt when pre-defined conditions occur. The CTRACE statement can be executed from a program or from the keyboard in the live keyboard mode.

Syntax:

CTRACE select code ; option[, option]

select code: The valid select code (channel) for your datacomm operation.

option: The option consists of one or more of the following:

- IN ON or IN OFF
- OUT ON or OUT OFF
- TX ON or TX OFF
- WRAP or STOP FULL or STOP ERROR

Default: **IN ON, OUT ON, TX ON, WRAP**

The control codes (CTL information) sent by the P-machine to control the tracing are always recorded by the trace buffer in Hexidecimal code. Input, output, and transmitted data can be recorded by the trace buffer as follows:

IN ON or IN OFF — This is the data that is actually received from the modem (if used) or from the 3 wire link (when no modem is used). This is the information sent from the interface to the Desktop Computer. This option enables (ON) or disables (OFF) the input data and control codes from being recorded by the trace buffer. INSEP sequences, prompt sequences, and ENQ characters are included as part of the input data.

OUT ON or OUT OFF — This is the information that is sent from the Desktop Computer to the interface card. This option enables (ON) or disables (OFF) the output data and control codes from being recorded by the trace buffer. Endline sequences (OUTSEP) are included as part of the output data.

TX ON or TX OFF — This is the data that is actually sent to the modem (if used) or to the 3 wire link (when no modem is used). This option enables (ON) or disables (OFF) the transmitted data from being recorded by the trace buffer. The transmitted data is obtained from a loop back connection in the interface cable.

The operating mode of the trace buffer is controlled as follows:

WRAP — This is the "continuous mode" and is the DEFAULT mode. The STOP ERROR and STOP FULL modes are disabled. All data entered into the trace buffer is retained until the trace buffer is full. As more data is entered into the buffer, the oldest (first entered) data is then lost.

STOP FULL — The STOP ERROR and WRAP modes are disabled. When the trace buffer is full, all tracing stops, CSTATUS element 8 is set to a 1, and an interrupt is generated for the even select code.

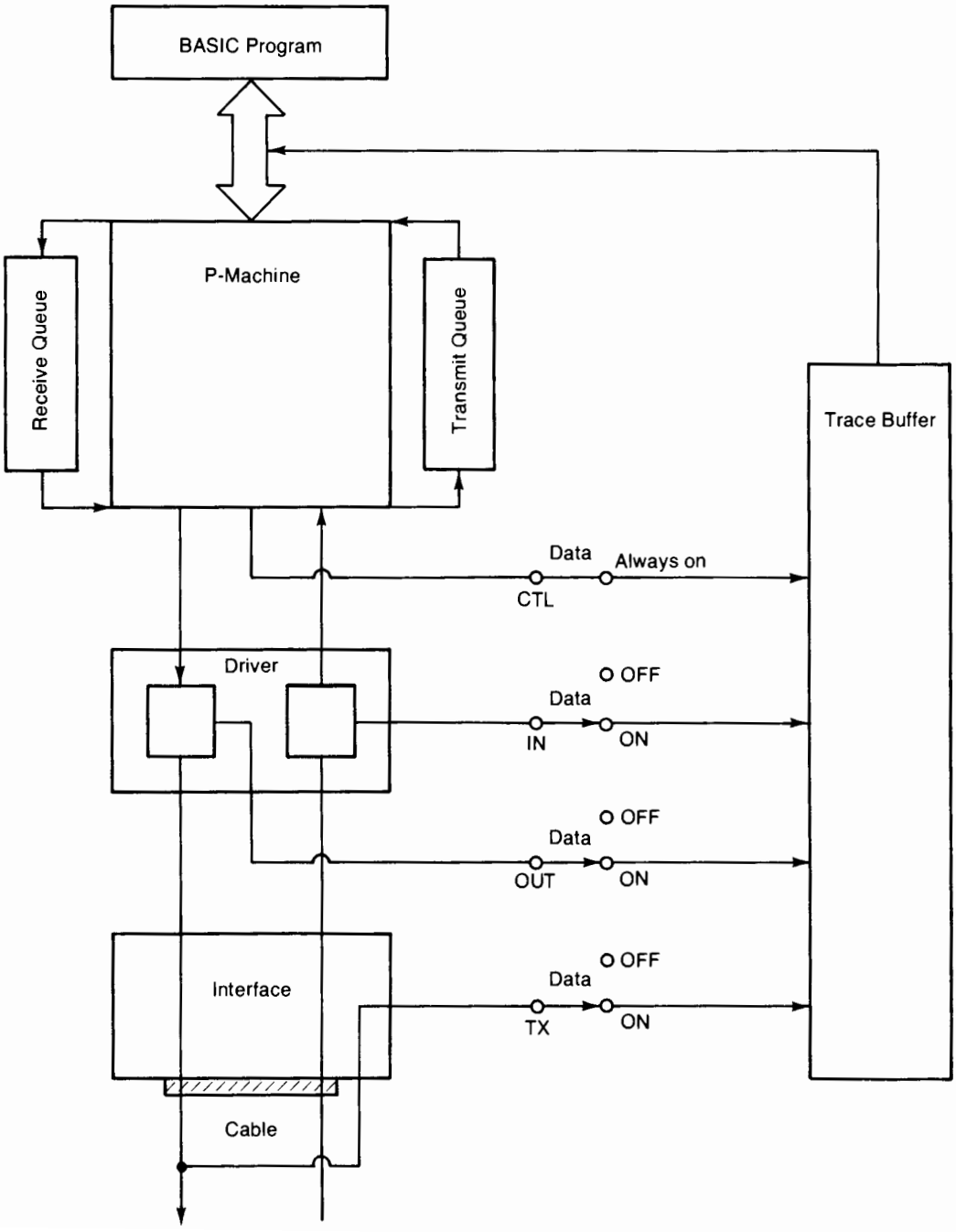
STOP ERROR — The STOP FULL mode is disabled. The trace buffer operates in the WRAP mode until either a framing error or a parity error is detected. When an error is detected, tracing is halted approximately 10 to 20 characters after the error, CSTATUS element 8 is set to a 1, and an interrupt is generated for the even select code.

NOTE

When the WRAP or STOP ERROR mode is specified, any previous data entered into the trace buffer is retained until subsequent data replaces it (see WRAP).

This diagram shows how IN ON, OUT ON, and TX ON data are entered into the trace buffer. The default condition (all ON) is shown.

CTRACE Diagram



CDUMP

The CDUMP statement allows you to select any or all of the trace buffer contents to be output to the PRINTER IS device. Data is output in ASCII code. Optional parameters enable you to specify the printout of the IN, OUT, and TX data in hexidecimal or octal code. The trace buffer data can also be entered into a string variable as ASCII data. The CDUMP statement can be executed in a program or from the keyboard in the live keyboard mode. Examples are provided after this discussion that show the various CDUMP options.

Syntax:

CDUMP select code[; dump option[, dump option]]

select code: The valid select code (channel) for your datacomm operation.

dump option: The following dump options may be specified:

- FORCE
- LTRACE ALL
- LTRACE IN
- LTRACE OUT
- LTRACE CTL
- LTRACE TX
- STRING = string variable
- CODE HEX
- CODE OCT

FORCE

In normal operation, a CDUMP cannot be executed when the channel is active. A CCONTROL...SUSPEND must be executed prior to the CDUMP statement. The FORCE option enables you to override this requirement and execute a CDUMP statement while the channel is active. Some trace information may be lost if the trace buffer is operating in the wrap mode. Data may appear out of order if transfers occur during CDUMP execution. The message "DUMP FORCED" is printed at the beginning of the printout.

LTRACE ALL

The LTRACE ALL option, when specified, causes the IN, OUT, CTL, and TX data in the trace buffer to be output to the PRINTER IS device or to the string variable specified by the STRING=option.

LTRACE IN

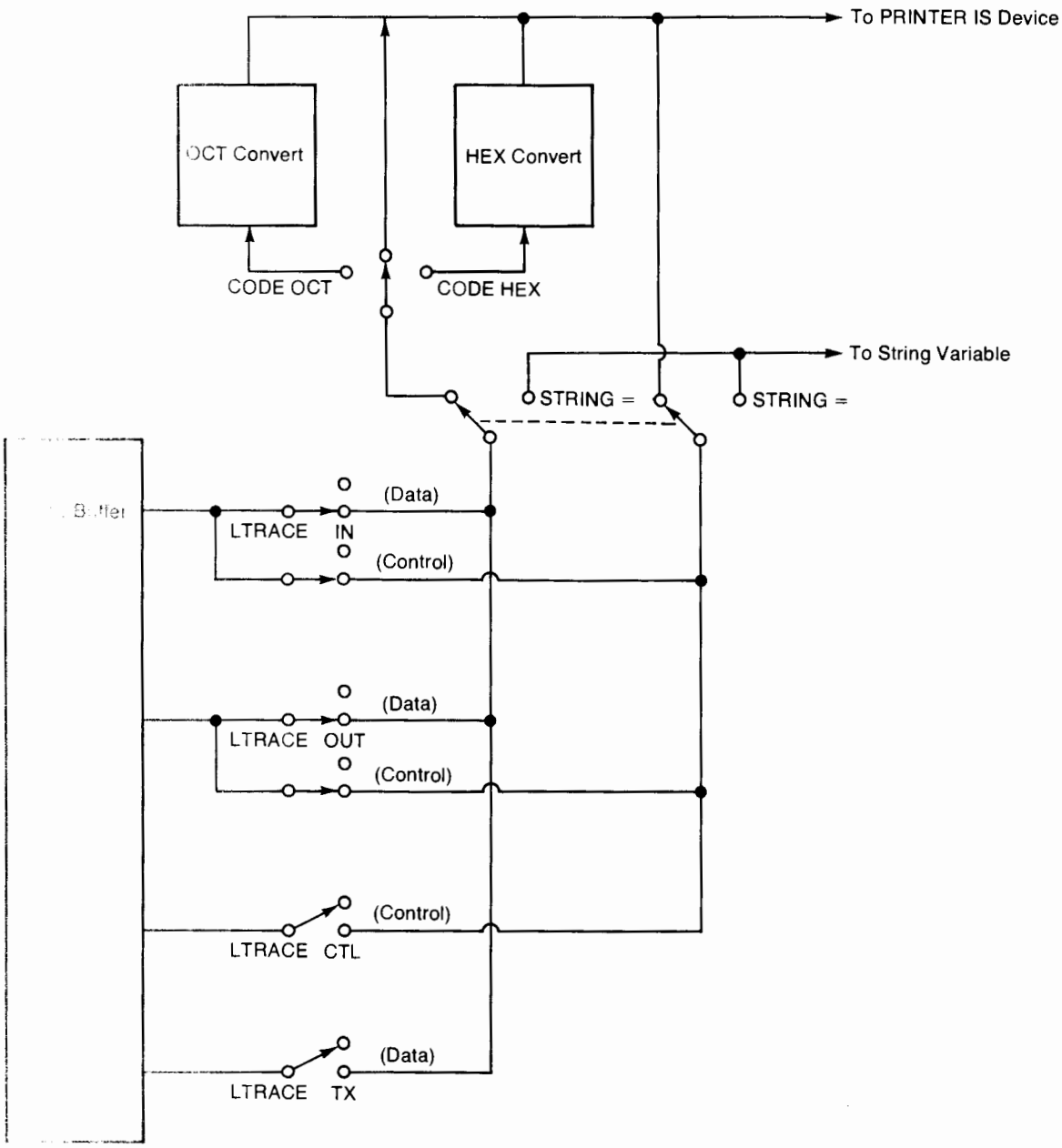
The LTRACE IN option, when specified, causes the IN (input) data in the trace buffer to be output to the PRINTER IS device or to the string variable specified by the STRING=option.

LTRACE OUT	The LTRACE OUT option, when specified, causes the OUT (output) data in the trace buffer to be output to the PRINTER IS device or to the string variable specified by the STRING=option.
LTRACE CTL	The LTRACE CTL option, when specified, causes the CTL (control) data in the trace buffer to be output to the PRINTER IS device or to the string variable specified by the STRING=option.
LTRACE TX	The LTRACE TX option, when specified, causes the TX (transmit) data in the trace buffer to be output to the PRINTER IS device or to the string variable specified by the STRING=option.
STRING=string variable	The STRING=option allows you to specify the destination of the CDUMP output to be a string variable instead of the PRINTER IS device. The data is entered into the string variable until the dimension limit has been reached. The data is entered in ASCII code ONLY. This option can be used to record the dump data onto an external device (e.g., tape) for later use.
CODE HEX	The CODE HEX option, when specified, causes the IN, OUT, and TX data to be output to the PRINTER IS device in hexidecimal code. This option cannot be specified if the STRING=option is used.
CODE OCT	The CODE OCT option, when specified, causes the IN, OUT, and TX data to be output to the PRINTER IS device in octal code. This option cannot be specified if the STRING=option is used.

The CDUMP statements presented in this section are for explanation of the various options. The default CDUMP statement, as presented in Chapter 7, is intended for normal use.

Default: **CDUMP...LTRACE IN, LTRACE OUT**

CDUMP Diagram



s shown.

ALL specifies LTRACE IN, LTRACE OUT,
LTRACE CTL, and LTRACE TX.

ll control codes are output in HEX code only. All data is output in ASCII code
AL or HEX conversion is specified.

This section shows the results that are obtained when various CDUMP options are specified. The log-on[1] message that appears on these examples was obtained from an account that was created on an HP3000 computer. The actual CDUMP results that you obtain will show the log-on[1] and account information that you enter into the Desktop Computer.

Key the following program into the Desktop Computer:

```
10   ! ************************************************************************
20   !              Note the CCOM and CMODEL parameter changes.
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !      ********************************************************
80   !       !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                  ! Dimension the string variables
110       CDISCONNECT 4                  ! Ensure you are disconnected
120       CCOM 2500                      ! Specify the CCOM memory as 2500 words
130       CMODEL ASYNC,4;TBUFFER=1000    ! Define the CMODEL (increase TBUFFER)
140       CCONNECT 4;HANDSHAKE OFF       ! Link is connected here
150       CCONTROL 4;ACK ON              ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit     ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive      ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE               ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                     ! Wait here for an interrupt
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN ! Check for a PROMPT before continuing
260       CREAD 4;A$                     ! Get the visual prompt character
270       PRINT A$;                      ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$         ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE            ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                     ! Read the data from the host
350       PRINT A$                       ! Print the data from the host
360  GOTO Receive                        ! Go check for more input data
370  !
380  END
```

Run the program and enter the appropriate log-on[1] message. Remember to press the "**CONTINUE**" key to transmit the data to the host.

Your log-on data should appear on the display.

1 Log-on procedures are explained in Chapter 7.

Stop ("**STOP**" key) the program and execute the following statement:

```
CDUMP select code; LTRACE ALL
```

The computer beeps and displays:

```
ERROR 303
```

The error is generated because a CDUMP statement cannot be executed while datacomm is active (not suspended) unless the FORCE option is specified. Stopping the program does NOT suspend datacomm.

Execute the following statements:

```
CCONTROL select code; SUSPEND
CDUMP select code; LTRACE ALL
```

The following dump should appear on the display:

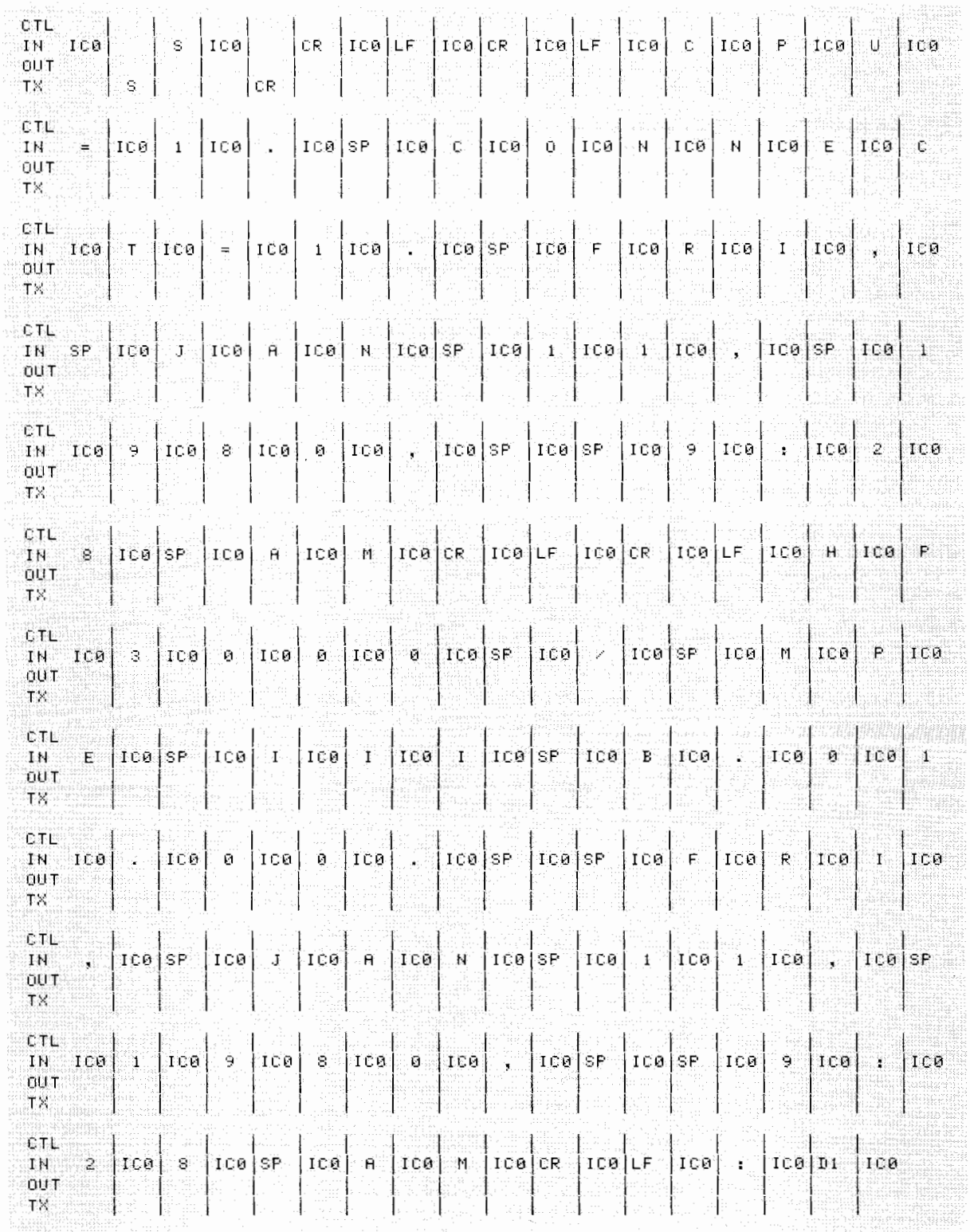
```
LINE TRACE    04
CTL C09 C07
IN                                                  I35 R0
OUT       002 0E2  ?  0B5 EX  035      0E3 RS  0F0 RS  0E6  d  0A2 SY  0A3
TX

CTL
IN
OUT SY  0B7  J. 0E5 SX  0E7 NU  0E8 NU  0E2  ?  0E4 R0  0F3 NU  0A8  E  0A9  E
TX

CTL
IN                                              R0  I01         I08 CR  IC0 LF  IC0  :
OUT 0AA  *. 0A7  A  0A6  A  035 088 001             CR
TX                                                  CR

CTL
IN  IC0 D1  IC0
OUT          H  E  L  L  O  SP  B  I  L  L  D  .  M  A  N  U
TX

CTL
IN
OUT  A  L  S       H  IC0        E  IC0     L  IC0     L  IC0     O  IC0
TX          H        CR    E        L        L        L        O
CTL
IN      SP  IC0       B  IC0     I  IC0     L  IC0     L  IC0     D  IC0
OUT
TX  SP          B        I        L        L        D        .
CTL
IN   .  IC0       M  IC0     A  IC0     N  IC0     U  IC0     A  IC0     L
OUT
TX         M        A        N        U        A        L
```


CTL IN OUT TX
IC0 S IC0 CR IC0 LF IC0 CR IC0 LF IC0 C IC0 P IC0 U IC0
S CR
CTL IN OUT TX
= IC0 1 IC0 . IC0 SP IC0 C IC0 O IC0 N IC0 N IC0 E IC0 C
CTL IN OUT TX
IC0 T IC0 = IC0 1 IC0 . IC0 SP IC0 F IC0 R IC0 I IC0 , IC0
CTL IN OUT TX
SP IC0 J IC0 A IC0 N IC0 SP IC0 1 IC0 1 IC0 , IC0 SP IC0 1
CTL IN OUT TX
IC0 9 IC0 8 IC0 0 IC0 , IC0 SP IC0 SP IC0 9 IC0 : IC0 2 IC0
CTL IN OUT TX
8 IC0 SP IC0 A IC0 M IC0 CR IC0 LF IC0 CR IC0 LF IC0 H IC0 P
CTL IN OUT TX
IC0 3 IC0 0 IC0 0 IC0 0 IC0 SP IC0 / IC0 SP IC0 M IC0 P IC0
CTL IN OUT TX
E IC0 SP IC0 I IC0 I IC0 I IC0 SP IC0 B IC0 . IC0 0 IC0 1
CTL IN OUT TX
IC0 . IC0 0 IC0 0 IC0 . IC0 SP IC0 SP IC0 F IC0 R IC0 I IC0
CTL IN OUT TX
, IC0 SP IC0 J IC0 A IC0 N IC0 SP IC0 1 IC0 1 IC0 , IC0 SP
CTL IN OUT TX
IC0 1 IC0 9 IC0 8 IC0 0 IC0 , IC0 SP IC0 SP IC0 9 IC0 : IC0
CTL IN OUT TX
2 IC0 8 IC0 SP IC0 A IC0 M IC0 CR IC0 LF IC0 : IC0 D1 IC0

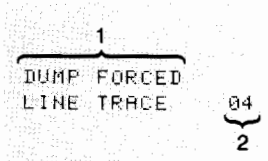
"What," you may ask, "is this?" The answer, quite simply, is that this is a complete copy of the transactions that occurred during the conversation between the Desktop Computer and the host. This is the information that is copied into the trace buffer. The next section shows what all the codes mean and how to interpret them.

NOTE

If tracing for any of the data has been disabled (OFF) then the line containing that data is empty. For example, if IN OFF has been specified for the CTRACE statement, then no data will appear in the IN portion of the dump.

The Header

The header information appears at the beginning of each dump that is displayed. Here is the description of the information contained in the header.

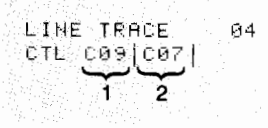


1 If the CDUMP statement included the FORCE option, this message will appear as part of the header
2 Indicates the select code of the channel (in this case, 4)

The remaining data provided by the dump is divided into four lines. Each group of four lines is separated by a blank line. The four lines of information are preceded by identifiers which indicate the type of data in the corresonding line (see the next diagram).

CTL

The CTL section of the trace contains trace buffer operating mode information. Here is an explanation of the CTL information.



1 Indicates that the trace buffer is operating in the "WRAP" mode (default)
2 Indicates that all data is being copied into the trace buffer (default)

Control codes always appear as three character mnemonics. Control codes from the CTL trace always begin with the letter "C" followed by a two character hexidecimal code.

The following control codes can appear as CTL information:

Code	Meaning
C01	TX trace flag on (TX ON).
C02	OUT trace flag ON (OUT ON).
C04	IN trace flag ON (IN ON).
C07	All data copied into trace buffer (default).
C08	Trace buffer in fill mode (STOP FULL).
C09	Trace buffer in WRAP mode (default).
C13	IN trace flag off (IN OFF).
C15	OUT trace flag off (OUT OFF).
C16	TX trace flag off (TX OFF).

IN

The data contained in the IN portion of the dump includes:

- Modem status information.
- Error indication.
- Timeout indication.
- Break detection.
- Actual input data.

Execute the following statement:

```
CDUMP select code;LTRACE IN
```

A dump similar to the following should appear on the display:

```
LINE TRACE    04
IN     |I35|RO |   |RO |I01|   |I08|CR |IC0|LF |IC0| : |IC0|D1 |IC0|   | H |IC0
IN      | E |IC0|   | L |IC0|   | L |IC0|   | O |IC0|   |SP |IC0|   | B |IC0|
IN   I |IC0|   | L |IC0|   | L |IC0|   | D |IC0|   | . |IC0|   | M |IC0|   | A
IN  IC0|   | N |IC0|   | U |IC0|   | A |IC0|   | L |IC0|   | S |IC0|   |CR |IC0
IN  LF |IC0|CR |IC0|LF |IC0| C |IC0| P |IC0| U |IC0| = |IC0| 1 |IC0| . |IC0|SP
IN  IC0| C |IC0| O |IC0| N |IC0| N |IC0| E |IC0| C |IC0| T |IC0| = |IC0| 1 |IC0
IN   . |IC0|SP |IC0| F |IC0| R |IC0| I |IC0| , |IC0|SP |IC0| J |IC0| A |IC0| N
IN  IC0|SP |IC0| 1 |IC0| 1 |IC0| , |IC0|SP |IC0| 1 |IC0| 9 |IC0| 8 |IC0| 0 |IC0
IN   , |IC0|SP |IC0|SP |IC0| 9 |IC0| : |IC0| 2 |IC0| 8 |IC0|SP |IC0| A |IC0| M
IN  IC0|CR |IC0|LF |IC0|CR |IC0|LF |IC0| H |IC0| P |IC0| 3 |IC0| 0 |IC0| 0 |IC0
IN   0 |IC0|SP |IC0| / |IC0|SP |IC0| M |IC0| P |IC0| E |IC0|SP |IC0| I |IC0| I
IN  IC0| I |IC0|SP |IC0| B |IC0| . |IC0| 0 |IC0| 1 |IC0| . |IC0| 0 |IC0| 0 |IC0
IN   . |IC0|SP |IC0|SP |IC0| F |IC0| R |IC0| I |IC0| , |IC0|SP |IC0| J |IC0| A
IN  IC0| N |IC0|SP |IC0| 1 |IC0| 1 |IC0| , |IC0|SP |IC0| 1 |IC0| 9 |IC0| 8 |IC0
IN   0 |IC0| , |IC0|SP |IC0|SP |IC0| 9 |IC0| : |IC0| 2 |IC0| 8 |IC0|SP |IC0| A
IN  IC0| M |IC0|CR |IC0|LF |IC0| : |IC0|D1 |IC0
```

This data is the same data that appeared in the IN portion of the LTRACE ALL dump. All control codes and data bytes are present. Control codes always appear as three character mnemonics. IN trace control codes always begin with the letter "I" followed by a two character hexidecimal code. IN data bytes appear as ASCII characters or octal or hexidecimal codes (see CODE OCT and CODE HEX).

The following control codes can appear in the IN trace.

Read Control Bytes

Code	Meaning
I01	Normal Operations begun.
I02	Suspend service to the S.I.O. (see 98046-90030).
I08	The following data bytes are received data.
I22	Parity or framing error detected (see Read Register 1[1]).
I35	Modem signals changed or modem signal status (see Modem Status Register[1]). The next byte is the modem status byte.

Write Control Bytes

Code	Meaning
I8C	Desktop Computer timeout.
I8D	All data has been transmitted (see OUT-08D).
IC0	Receiver timeout, moves data from the interface to the mainframe
IC1	Transmit timeout (CSTATUS (0) error 11).
IC3	Timeout for NO ACTIVITY (CSTATUS (0) error 106).
IC5	Break received from the remote device.
IC7	Auto disconnect forced (CSTATUS (0) error 104).

Considerations:

- The Write Control Bytes (I00—I7F) NEVER have associated data bytes.
- The Read Control Bytes (I80—IFF) may or may not have associated data bytes.
- The data bytes are ALWAYS associated with the preceding Read Control Byte.

1 Fold out at the end of this chapter.

Here's how to interpret the trace:

```
           1               2
   ┌───────┴───────┐      ┌┴┐          3          4
   LINE TRACE              04        ┌─┴─┐      ┌─┴─┐
5{ IN        |I35|RO  |      |RO  |I01|      |I08|CR  |IC0|LF  |IC0| :  |IC0|D1  |IC0|     | H  |IC0
              └─┬─┘└─┬─┘
                6    7
   IN        | E  |IC0|     | L  |IC0|     | L  |IC0|     | O  |IC0|     |SP  |IC0|     | B  |IC0|
   IN    I  |IC0|     | L  |IC0|     | L  |IC0|     | D  |IC0|     | .  |IC0|     | M  |IC0|     | A
   IN   IC0|     | N  |IC0|     | U  |IC0|     | A  |IC0|     | L  |IC0|     | S  |IC0|     |CR  |IC0
   IN   LF  |IC0|CR  |IC0|LF  |IC0| C  |IC0| P  |IC0| U  |IC0| =  |IC0| 1  |IC0| .  |IC0|SP
   IN   IC0| C  |IC0| O  |IC0| N  |IC0| N  |IC0| E  |IC0| C  |IC0| T  |IC0| =  |IC0| 1  |IC0
   IN    .  |IC0|SP  |IC0| F  |IC0| R  |IC0| I  |IC0| ,  |IC0|SP  |IC0| J  |IC0| A  |IC0| N
   IN   IC0|SP  |IC0| 1  |IC0| 1  |IC0| ,  |IC0|SP  |IC0| 1  |IC0| 9  |IC0| 8  |IC0| 0  |IC0
   IN    ,  |IC0|SP  |IC0|SP  |IC0| 9  |IC0| :  |IC0| 2  |IC0| 8  |IC0|SP  |IC0| A  |IC0| M
   IN   IC0|CR  |IC0|LF  |IC0|CR  |IC0|LF  |IC0| H  |IC0| P  |IC0| 3  |IC0| 0  |IC0| 0  |IC0
   IN    0  |IC0|SP  |IC0| /  |IC0|SP  |IC0| M  |IC0| P  |IC0| E  |IC0|SP  |IC0| I  |IC0| I
   IN   IC0| I  |IC0|SP  |IC0| B  |IC0| .  |IC0| 0  |IC0| 1  |IC0| .  |IC0| 0  |IC0| 0  |IC0
   IN    .  |IC0|SP  |IC0|SP  |IC0| F  |IC0| R  |IC0| I  |IC0| ,  |IC0|SP  |IC0| J  |IC0| A
   IN   IC0| N  |IC0|SP  |IC0| 1  |IC0| 1  |IC0| ,  |IC0|SP  |IC0| 1  |IC0| 9  |IC0| 8  |IC0
   IN    0  |IC0| ,  |IC0|SP  |IC0|SP  |IC0| 9  |IC0| :  |IC0| 2  |IC0| 8  |IC0|SP  |IC0| A
   IN   IC0| M  |IC0|CR  |IC0|LF  |IC0| :  |IC0|D1  |IC0
```

1 Header
2 Channel (select code)
3 Read Control Byte (normal operations begun)
4 Read Control Byte (the following data is received data)
All subsequent data bytes are received data until another Read Control Byte is encountered.
5 IN data identifier
6 Read Control Byte (modem status)
7 Modem status
RO indicates an ASCII DEL character. The bit pattern of the DEL character is 1111111 (see Appendix A). (Bit 0 ↙ points to the last 1)
▭ Actual received data (including echoed data).

See the modem status register (foldout at the end of this chapter) for your interface cable. The DEL bit pattern (all 1s) means that no modem lines are active (this is a direct connection).

OUT

The data contained in the OUT portion of the dump includes:

- Modem control signals.
- CMODEL ASYNC parameter set up (CHARLENTH, GAP, CHECK).
- Set and start various timeouts (e.g., NO ACTIVITY, Auto disconnect).
- Set CCONNECT parameters (SPEED, HALF/FULL DUPLEX, etc.).
- Actual ouput data.

Execute the following statement:

```
CDUMP select code; LTRACE OUT
```

A dump similar to the following should appear on the display:

```
LINE TRACE    04
OUT    |002|0E2| ? |0B5|EX |035|   |0E3|RS |0F0|RS |0E6| d |0A2|SY |0A3|SY |0B7

OUT  J.|0E5|SX |0E7|NU |0E8|NU |0E2| ? |0E4|RO |0F3|NU |0A8| E |0A9| E |0AA| *.

OUT 0A7| A |0A6| A |035|088|001|   |CR |   | H | E | L | L | O |SP | B | I | L

OUT  L | D | . | M | A | N | U | A | L | S |   |CR |
```

This data is the same data that appeared in the OUT portion of the LTRACE ALL dump. All control codes and data bytes are present. Control codes always appear as three character mnemonics. OUT trace control codes always begin with the letter "O" followed by a two character hexidecimal code. OUT data bytes appear as ASCII characters or octal or hexidecimal code (see CODE OCT and CODE HEX).

Read Control Bytes

Code	Meaning
O01	Begin normal operation.
O02	Begin communication with the Desktop Computer.
O35	Read modem status register (see I35).

Write Control Bytes

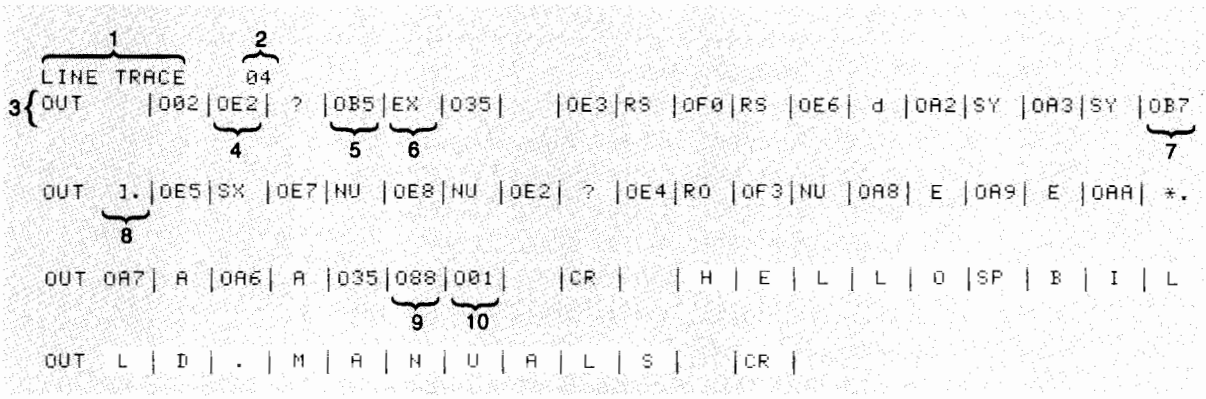
Code	Meaning
O88	Transmit the following data.
O8D	Wait for data to be output before continuing (see I8D).
OA2	Write to SIO A register 1 (See Write Register 1[1]).
OA3	Write to SIO B register 1 (See Write Register 1[1]).
OA6	Write to SIO A register 3 (See Write Register 3[1]).
OA7	Write to SIO B register 3 (See Write Register 3[1]).
OA8	Write to SIO A register 4 (See Write Register 4[1]).
OA9	Write to SIO B register 4 (See Write Register 4[1]).
OAA	Write to SIO A register 5 (See Write Register 5[1]).
OB5	Set modem lines (see Modem Control Register[1]).
OB7	Set transfer rates (see Data Rate Table).
OE2	Set modem mask
OE3	Set no activity timer value
OE4	Set character mask (#bits/character)
OE5	Set receive timeout value
OE6	Set transmit timeout value
OE7	Set GAP timeout value
OE8	Set ERROR mask (CTRACE STOP ERROR)
OED	Set transmit timer
OEE	Start mainframe timer (e.g. 25sec connect timeout)
OF0	Start no activity timer
OF1	Set autodisconnect timer value
OF2	Set autodisconnect mask(half/full duplex and handshake on/off
OF3	Set "OR" mask for transmitting (check=1)

Considerations:

- The Read Control Bytes (O00—O7F) NEVER have associated data bytes.
- The Write Control Bytes (O80—OFF) may or may not have associated data bytes.
- The data bytes are ALWAYS associated with the preceding Write Control Byte.

1 Fold out at the end of this chapter.

Here's how to interpret the trace:



1 Header
2 Select code (channel)
3 OUT data identifier
4 Write Control Byte (set modem mask)
5 Write Control Byte (set modem lines)
6 Value for modem lines
7 Write Control Byte (set transfer rates)
8 Value for transfer rates
9 Write Control Byte (transmit the following data)
10 Read Control Byte (begin normal operation)
☐ Actual transmitted data

The EX indicates an ASCII ETX character. The bit pattern of an ETX is 0[1]0000011 (see Appendix A). See the modem control register (fold out at the end of this chapter) for your interface option cable. For the 98046 Option 1 cable, the 00000011 bit pattern indicates the following:

Value	Bit	Modem Line	State
1	0	DRIVER1	low (OFF) } These are default states for the CCONTROL...DRIVER statement.
1	1	DRIVER2	low (OFF)
0	2	DRIVER3	high (ON)
0	3	Not used	high
0	4	Request to Send	high (ON)
0	5	Data Terminal Ready	high (ON)
0	6	Not used	high
0	7	Not used	high

1 All ASCII characters consist of 7 bits. The read, write, and modem registers require 8 bits. The eighth bit value of 1 is indicated by the presence of a decimal point (".") immediately following the ASCII character (see the next explanation).

The "]" indicates an ASCII] character. The bit pattern of a] character is 1011101 (see Appendix A). The decimal point (".") indicates that the eighth bit is set (1). The bit pattern is 11011101. For transfer rate specification, this bit pattern is divided into two four bit fields, 1101 and 1101. These fields specify the transmit and receive transfer rates (baud) as follows:

Data Rate Table

	Receive Rate				Transmit Rate			
Bit	7	6	5	4	3	2	1	0
Ext	0	0	0	0	0	0	0	0
Ext	0	0	0	1	0	0	0	1
50	0	0	1	0	0	0	1	0
75	0	0	1	1	0	0	1	1
135	0	1	0	0	0	1	0	0
200	0	1	0	1	0	1	0	1
600	0	1	1	0	0	1	1	0
2400	0	1	1	1	0	1	1	1
9600	1	0	0	0	1	0	0	0
4800	1	0	0	1	1	0	0	1
1800	1	0	1	0	1	0	1	0
1200	1	0	1	1	1	0	1	1
2400	1	1	0	0	1	1	0	0
1 300	1	1	0	1	1	1	0	1
150	1	1	1	0	1	1	1	0
110	1	1	1	1	1	1	1	1

2

1 The bit pattern "1101 1101" indicates Receive and Transmit rates are both 300.
2 This bit is indicated by the decimal point.

CODE OCT

Execute the following statement:

```
CDUMP select code;LTRACE OUT,CODE OCT
```

A dump similar to the following should appear on the display:

```
      1        2
   LINE TRACE  04
3{OUT    |002|0E2|077|0B5|003|035|   |0E3|036|0F0|036|0E6|144|0A2|026|0A3|026|0B7
 OUT 335|0E5|002|0E7|000|0E8|000|0E2|077|0E4|177|0F3|000|0A8|105|0A9|105|0AA|252
 OUT 0A7|101|0A6|101|035|088|001|   |015|   |110|105|114|114|117|040|102|111|114
 OUT 114|104|056|115|101|116|125|101|114|123|   |015|
```

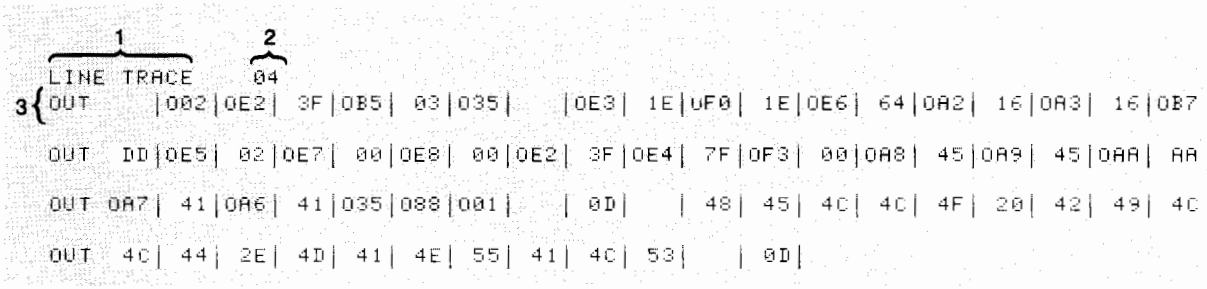
1 Header
2 Select code (channel)
3 OUT data identifier
☐ Note that all ASCII data bytes have been converted to octal code.

CODE HEX

Execute the following statement:

```
CDUMP select code;LTRACE OUT,CODE HEX
```

A dump similar to the following should appear on the display:



1 Header
2 Select code
3 OUT data identifier
☐ Note that all data bytes have been converted to hexidecimal code.

Bit patterns for associated control byte interpretation can be simplified if you specify octal or hexidecimal code.

For example:

The octal code for the ASCII]. is 335. The bit pattern for 335 octal is $\underbrace{11}_{3}\underbrace{011}_{3}\underbrace{101}_{5}$.

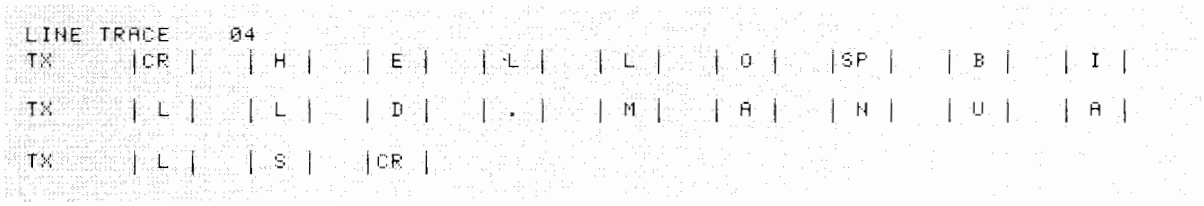
TX

The data contained in the TX portion of the dump is the actual data that was transmitted over the line to the remote.

Execute the following statement:

```
CDUMP select code;LTRACE TX
```

A dump similar to the following should appear on the display:



This is the same data that appeared in the TX portion of the LTRACE ALL dump. This is the data that was actually transmitted from the interface card to the link. Control codes do not appear in the TX dump. Data bytes appear as ASCII characters or octal or hexidecimal code (see CODE OCT and CODE HEX).

Read and write control bytes that appear in the IN or OUT dump are interpreted as follows. The control byte table shows the register that is associated with each control byte. The bit pattern of the data byte that follows is then mapped into that register to set the appropriate bits.

STRING

Execute the following statements:

```
CDUMP select code;LTRACE TX,STRING=A$
A$
```

The following should appear on the display:

1 Header
2 Select code (channel)
3 Identifies the following data byte as a transmitted data byte (from TX data)
☐ Actual transmitted data

The identifiers are shown next.

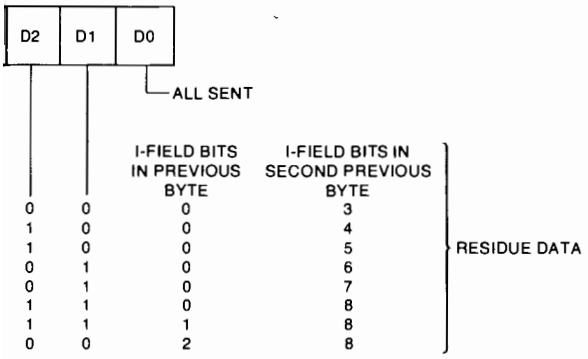
t — next data byte is TX data.
i — next data byte is IN data.
o — next data byte is OUT data.

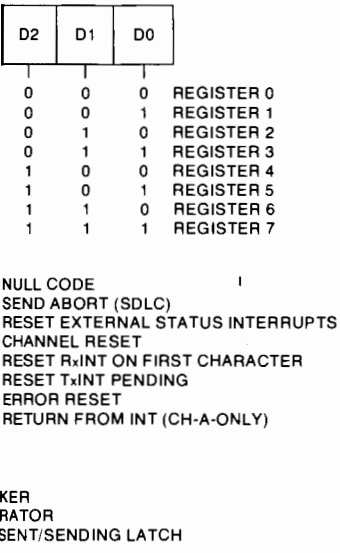
Control Bytes are identified as follows:

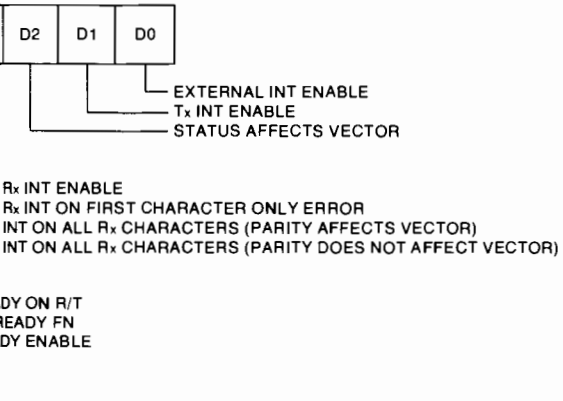
I — next data byte is an IN control byte.
O — next data byte is an OUT control byte.
C — next data byte is a CTL control byte.

1 Fold out at the end of this chapter.

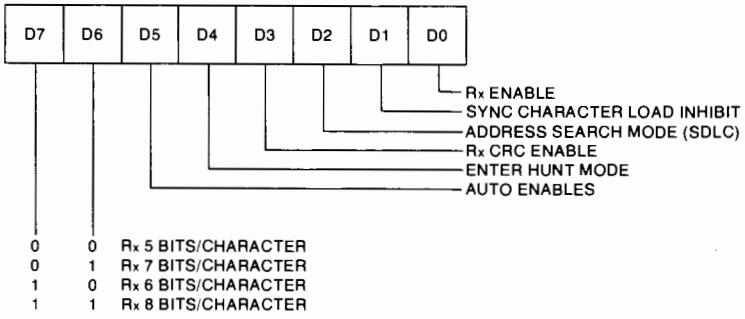
Read/Write Registers



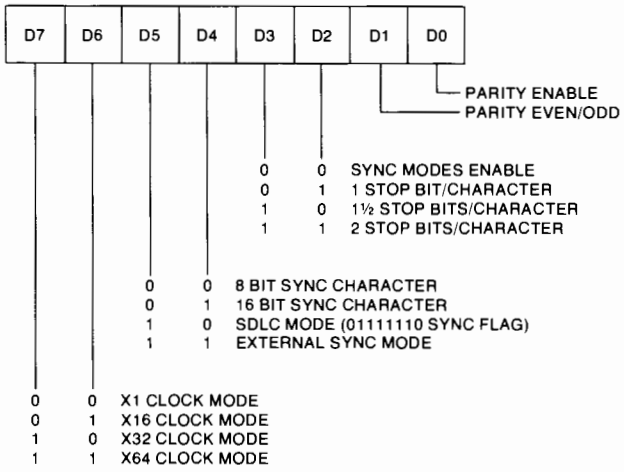




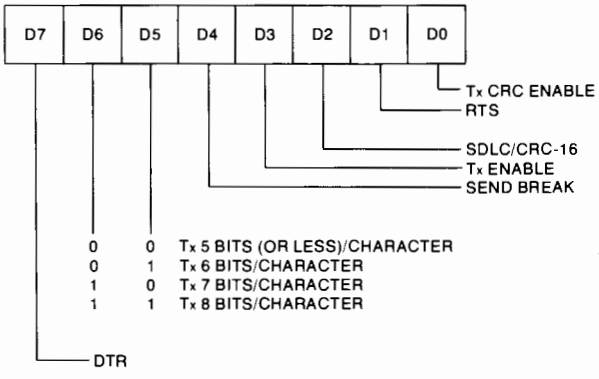
Write Register 3



Write Register 4

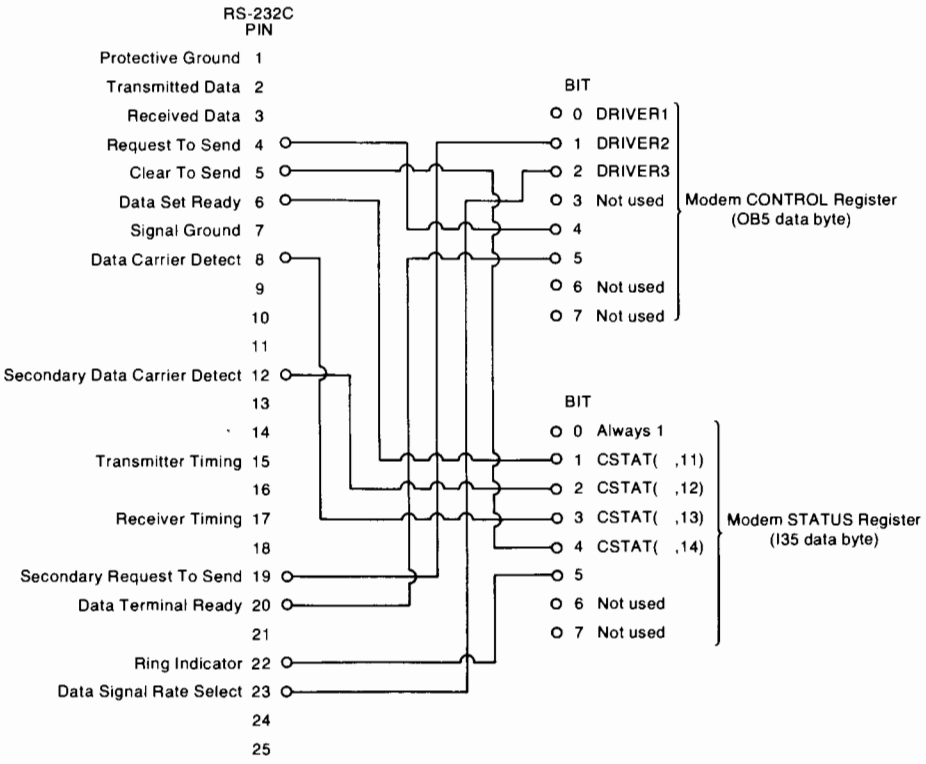


Write Register 5



Registers

98046 Option 001 Cable

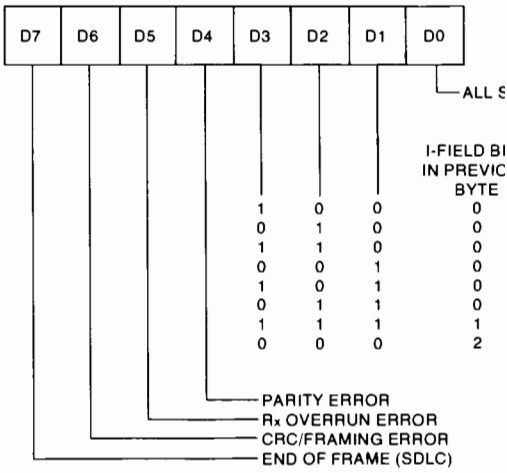


Interpretation

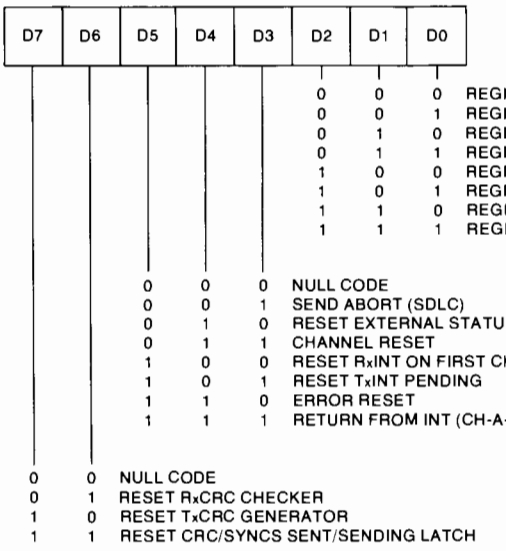
Bit value=0 — The line is active (high).

Bit value=1 — The line is inactive(low).

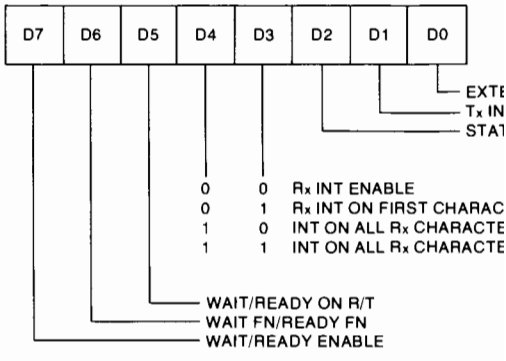
Read Register 1



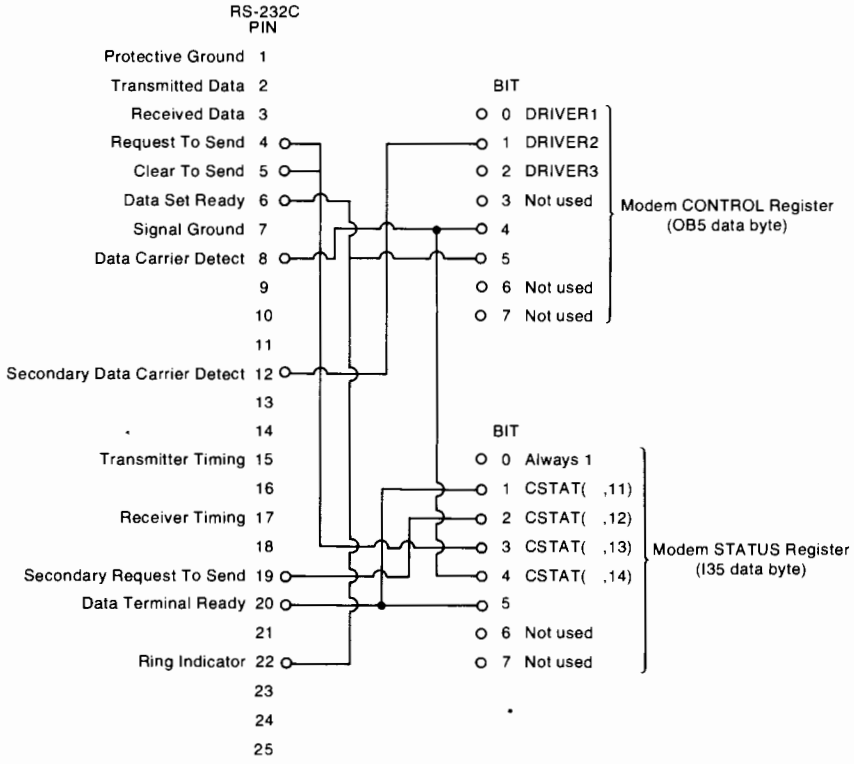
Write Register 0



Write Register 1



98046 Standard Cable



Interpretation

Bit value = 0 — The line is active (high).
Bit value = 1 — The line is inactive (low).

Chapter 9
Table of Contents

Troubleshooting

Chapter 9
Troubleshooting

Introduction

This section of the manual explains how to isolate and correct data communication problems that may arise. The Basic Data Communications ROM provides the following tools as troubleshooting aids:

- Data Communications ROM errors.
- CSTATUS 0 errors.
- The CTRACE statement.[1]
- The CDUMP statement.[1]

When you are having problems you should always:

- Ensure that all of the equipment has been switched on.
- Wait at least 25 seconds to make sure a connection has been established.
- Observe the symptoms, then check the probable cause utilizing the features provided by the CTRACE and CDUMP statements.

1 CTRACE and CDUMP are explained in Chapter 8.

The datacomm ROM errors are serial errors. When a serial datacomm error occurs, the Desktop Computer beeps, displays the error and the program line number where the error occurred, and terminates the program. Serial datacomm errors can be trapped using the ON ERROR statement (see your Operating and Programming Manual. An ON ERROR example is shown following the serial error table).

The following table shows the serial error number, explains what the error means, and lists probable causes for the error.

Serial Errors

Error (ERRN)	Meaning	Probable Cause
300	CCOM area not allocated	CCOM statement missing from the program.
301	Not allowed when channel is active	CCONNECT or CMODEL attempted to connected channel.
302	CMODEL statement required	CMODEL statement missing or CCONNECT executed prior to CMODEL.
303	Not allowed when trace is active	Attempted to execute a CDUMP statement while a datacomm operation is active. Use CCONTROL...SUSPEND or CDUMP...FORCE.
304	Too many characters in CWRITE	Note 1
305	New CCOM size not allowed when channel is active	Attempted to execute a CCOM statement of a different size without disconnecting the channel.
306	98046 card failure	More than 1 interface set to the same select code, card not properly seated, or card failure. If you suspect a card failure, see the hardware test procedures in the 98046 Interface Installation and Service Manual (98046-90030).
307	Insufficient CCOM allocation	CMODEL statement(s) requires more memory allocation than specified by the CCOM statement. Refer to Appendix D, Memory Allocation.

Error (ERRN)	Meaning	Probable Cause
308	Illegal character in CWRITE of non-TRANSPARENT data	Note 1
309	Not allowed for this CMODEL	Illegal CMODEL parameter (e.g. Synchronous parameter specified for CMODEL ASYNC), or CHARLENGTH=8 specified with CHECK=0 or CHECK=1.
310	CCONNECT statement required	CREAD, CWRITE, CCONTROL, or CSTATUS statement executed to a disconnected channel.
311	Not allowed while datacomm is suspended	CWRITE or CTRACE attempted to a suspended channel.
312	Improper CSTATUS array	CSTATUS array must be a one dimensional 16 element INTEGER array.

Note 1: Errors 304 and 308 are synchronous errors only. These errors are not valid for asynchronous communication.

Serial Error Trapping

Here is how to trap a serial datacomm error. Key the following program lines into your Desktop Computer:

```
10   ! ********************************************************************************
20   !
30   !  The following program shows how to trap SERIAL errors and how to correct
40   !  the problem without terminating the program.  When the error message
50   !  appears, key in the following statement, press EXECUTE, and CONTINUE.
60   !                              CMODEL ASYNC, 4
70   !                              --------------
80   ! ********************************************************************************
90   !
100  ON ERROR GOTO Errorcheck                  ! This statement traps the error
110  !
120  DIM A$[80],B$[160]                        ! Dimension the string variables
130       CDISCONNECT 4                        ! Ensure you are disconnected
140       CCOM 1500                            ! Specify memory allocation
150 Connect:!
160       CCONNECT 4;HANDSHAKE OFF             ! Link is connected here
170       CCONTROL 4;ACK ON                    ! Specify ENQ-ACK handshake
180       CWRITE 4;ENDLINE                     ! Send C/R to the host (I am here)
190  !
200  !
210 Promptcheck:  IF NOT CSTAT(4,2) OR CSTAT(4,1) THEN Readcheck
220  !
230  !
```

```
240       CREAD 4;A$                          ! Get the PROMPT
250       PRINT A$;                           ! Print the PROMPT
260       LINPUT B$                           ! Enter your data here
270       CWRITE 4;B$,ENDLINE                 ! Send data and C/R
280  !
290 Readcheck:  IF NOT CSTAT(4,1) THEN Promptcheck
300  !
310  !
320       CREAD 4;A$                          ! Read data from the host
330       PRINT A$                            ! Print data from the host
340  GOTO Promptcheck
350  !
360  !
370 Errorcheck: !
380  !
390  IF ERRN<>302 THEN Errwrong              ! Check for correct error number
400  PRINT "You have detected the following error - ";ERRM$
410  PRINT "Key  CMODEL ASYNC,4  into the Desktop, press  EXECUTE  and CONTINUE"
420  !
430  PAUSE                                   ! Wait for corrections
440  GOTO Connect                            ! Go try again
450  !
460 Errwrong: !
470  PRINT "THE ERROR DETECTED WAS";ERRM$
480  PRINT "THIS PROGRAM ONLY CORRECTS ERROR 302"
490  END
```

Press **RUN**

The following error message appears on your display:

```
You have detected the following error - ERROR 302 IN LINE 160
```

Follow the directions that appear on the display.

The link connection should now be established.

CSTATUS 0 ERRORS

The following table shows the datacomm error values that are entered into CSTATUS element 0 when they occur. The table shows the error number, what the error means, and lists probable causes for the error.

CSTATUS 0 Errors

Value	Error	Meaning	Probable Cause
10	Timeout before connection	Data Set Ready (and Data Carrier Detect if full duplex) did not become active within the 25 second timeout period.	1. HANDSHAKE OFF not specified for direct (no modem) connection. 2. Modem not switched on. 3. Remote and local modems not compatible.
11	Clear to Send line false or missing clock.	Unable to transmit for 10 seconds.	1. Incorrect speed parameters. 2. Clear to Send signal missing, check CSTATUS element 14.
100	Channel MEMLIMIT overflow	Input or output queues have overflowed.	1. Incorrect INSEP specified. 2. CREAD priority set lower than CWRITE priority. 3. Insufficient MEMLIMIT specified. 4. Incorrect speed parameters. 5. CREAD not executed when CSTATUS element 1 is true.
102	Input buffer overflow		1. Need ENQ/ACK or XON handshake. 2. Need larger INBUFFER specified. 3. Too many simultaneous I/O operations with datacomm. 4. Modem signals floating.
103	Internal buffer overflow		1. Modem signals noisy. 2. Non-standard modem being used. 3. RS-232 connection improperly wired. 4. Receiving repeated "break" signals from the host.

CSTATUS 0 Errors (cont.)

Value	Error	Meaning	Probable Cause
104	Autodisconnect forced	Data Set Ready (or Data Carrier Detect if full duplex) inactive for LOST CARRIER timeout.	1. Incorrect HALF DUPLEX/ FULL DUPLEX specified. 2. LOST CARRIER specifier too small or not needed.
106	NO ACTIVITY timeout	No data has been transmitted or received for the NO ACTIVITY timeout.	1. NO ACTIVITY parameter too small. 2. Datacomm suspended. 3. Neglected to disconnect from the link.
200	98046 buffer overflow		1. Too much simultaneous I/O operations with datacomm. 2. Remove FHS transfers.

CSTATUS element 0 errors can be trapped by specifying an ON INT statement to generate an interrupt for the even select code and specifying the INTMASK value to include the bit value of 1 in the CCONTROL...INTMASK = statement. Since the CCONTROL...INTMASK = statement canNOT be executed prior to the CCONNECT statement, the "Timeout before connection" (value 10) error is handled as follows:

- Prior to executing a CCONNECT statement, CSTATUS element 0 error 10 (Timeout before connection) generates an interrupt for the even select code if an ON INT statement has been specified for that select code.
- After the CCONNECT statement has been executed, the "Timeout before connection" error canNOT occur, and the INTMASK value defaults to a value of 6 (see CCONTROL...INTMASK, Chapter 4).

CSTATUS Error Trapping

Here's how to trap CSTATUS element 0 errors:

```
10   ! ************************************************************************
20   !
30   !  The following program is an interrupt driven line mode emulator.  This
40   !  program is for a direct (NO MODEM) connection to an H.P. 3000.  The
50   !  CONNECT statement (line 170) species "HANDSHAKE ON" which is incorrect.
60   !  The program will wait for 25 seconds, and then generate a CSTAT 0 error.
70   !  When the error message appears, press "CONTINUE", and the program will
80   !  do a CDISCONNECT, a CMODEL, and a new CCONNECT with HANDSHAKE OFF to
90   !  correct the problem for you.  You will have to correct any other errors.
100  ! ************************************************************************
110  !
120  DIM A$[80],B$[160]                    ! Dimension the string variables
130       ON INT #4 GOTO Intcheck          ! Even select code interrupt
140       CDISCONNECT 4                    ! Ensure you are disconnected
150       CCOM 2000                        ! Specify memory allocation
160       CMODEL ASYNC,4                   ! Define the cMODEL (ALL DEFAULTS)
170       CCONNECT 4;HANDSHAKE ON          ! Specify HANDSHAKE ON (this is wrong)
180 Waiting:   !
190       IF CSTAT(4,5)<>2 THEN Waiting ! Wait here until connected
200       CCONTROL 4;ACK ON                ! Host requires ENQ-ACK
210       ON INT #4,1 GOSUB Transmit       ! Even select code interrupt
220       ON INT #5,2 GOSUB Receive        ! Odd select code interrupt
230       CWRITE 4;ENDLINE                 ! Send CR to the host( I am here)
240 Spin: !
250       GOTO Spin                        ! Wait here for interrupts
260  !
270 Transmit: !
280       IF NOT CSTAT(4,2) THEN RETURN ! Check for PROMPT detected
290       CREAD 4;A$                       ! Get the PROMPT character
300       PRINT A$;                        ! Print the PROMPT character
310       LINPUT "Enter data",B$           ! Enter your data here
320       CWRITE 4;B$,ENDLINE              ! Send data and C/R
330       RETURN
340  !
350 Receive: !
360       IF NOT CSTAT(4,1) THEN RETURN ! Check for data from host
370       CREAD 4;A$                       ! Read data from the host
380       PRINT A$                         ! Print data from the host
390  GOTO Receive
400  !
410 Intcheck: !
420       IF CSTAT(4,0)<>10 THEN Errout ! Check for CSTAT(0) error
430  !
440  PRINT "THE ERROR IS CSTATUS 0, #";CSTAT(4,0);"- Timeout before connection"
450  PRINT "THE PROGRAM CORRECTS THE PROBLEM WHEN YOU PRESS THE CONTINUE KEY."
460  PAUSE
470  !
480       CDISCONNECT 4                    ! Disconnect from the link
490       CMODEL ASYNC,4                   ! Execute new CMODEL statement
500       CCONNECT 4;HANDSHAKE OFF         ! New CCONNECT statement (HANDSHAKE OFF)
510       GOTO Waiting                     ! Go back to Waiting
520  !
530 Errout:                                ! Error other than expected
540  PRINT "THERE IS A DIFFERENT PROBLEM, YOU WILL HAVE TO CORRECT THAT ONE"
550  END
```

Other Errors

Some problems may arise that do not generate error messages. The following table lists some of the common problems that may occur, their cause, and suggested corrective active.

Other Errors

Problem	Cause	Suggested Corrective Action
Cannot perform tape operations.	Interface card set to select code greater than 7.	Change Interface to select code less than 8.
		Execute a CDISCONNECT before performing tape operations.
CSTAT 1 is always false.	No received data present.	Check for connection (CSTAT 5) Check for Data Carrier Detect signal (CSTAT 13)
	Input separator(INSEP) incorrect.	Correct the INSEP specifier (CMODEL)
	ALERTN too large.	Correct the ALERTN specifier (CMODEL)
	Did not prompt the remote computer.	Execute CWRITE...ENDLINE
	Wrong SPEED=specified.	Correct the SPEED=specifier (CCONNECT)
	Incorrect DUPLEX specified	Correct the DUPLEX specifier (CMODEL)
	Incorrect HANDSHAKE specified	Correct the HANDSHAKE specifier (CCONNECT)
CSTAT 2 always false	Never received a prompt from the remote computer.	
	Incorrect Prompt=specified	Correct the PROMPT=specifier (CMODEL)
		See the previous problem (CSTAT 1 always false).

Problem	Cause	Suggested Corrective Action
Data has under-scores CSTAT 6#0	Parity or framing errors	
	Incorrect CHECK=specified	Correct the CHECK=specifier (CMODEL).
	Incorrect STOPBITS=specified	Correct the STOPBITS=specifier (CMODEL).
	Incorrect CHARLENGTH=specified	Correct the CHARLENGTH=specifier (CMODEL).
	Incorrect HANDSHAKE specified	Correct the HANDSHAKE specifier (CCONNECT).
Remote computer receives no data	Incorrectly transmitting the data	
	Incorrect HANDSHAKE specified	Correct the HANDSHAKE specifier (CCONNECT).
	Incorrect SPEED=specified	Correct the SPEED=specifier (CCONNECT)
	Incorrect or no GAP=specified	Correct the GAP=specifier (CMODEL).
Remote does not recognize input when parameters are changed.	Failed to sign off before you changed parameters.	Sign off using old parameters before changing things.
Strange characters, control codes, inverse video, etc. appear on your display.	Incorrect CHARLENGTH or CHECK specified. The remote is sending 7 bits & parity but the Desktop Computer is specified for 8 bits with no parity.	Correct the CHARLENGTH= or CHECK=specifier (CMODEL).

Chapter 10
Table of Contents

Chapter 10
ON KBD and TDISP

Introduction

The programs presented in the manual thus far use the LINPUT statement for keyboard data entry. With LINPUT statements, program execution stops until the keyboard data has been entered and the CONTINUE key has been pressed. This chapter describes two methods of entering keyboard data under interrupt control. The statements provided are:

- ON KBD
- TDISP

The ON KBD Statement

The ON KBD statement provides keyboard data entry under interrupt control. The ON KBD[1] statement may be used to immediately transmit each character that is typed on the keyboard, or to enter each character into a buffer for subsequent transmission. The ON KBD statement is standard with the System 45B. For System 35 applications, the I/O ROM is required.

The TDISP Statement

The TDISP statement displays the current keyboard entry data in the keyboard entry line of the CRT. As subsequent characters are entered from the keyboard, they are appended to the current display. The characters are entered under interrupt control with the ON KBD statement. TDISP provides cursor backspace and clear-line control features. TDISP is not executable from the keyboard.

Syntax:

TDISP string expression

TDISP concatenates the value of the specified string expression to those characters appearing on the keyboard entry line of the CRT. The keyboard entry line is limited to 160 characters.

1 See the System 35 I/O ROM Programming Manual or the System 45B Operating and Programming Manual for details about the ON KBD statement.

A limited amount of cursor control is provided with TDISP. The two control characters and their respective functions follow:

`CHR$(8)` `TDISP CHR$(8)`	This character causes the cursor to appear one space to the left of the current cursor position without obliterating the character in the new position, if one exists.
`CHR$(12)` `TDISP CHR$(12)`	This character causes the entire keyboard entry line to be cleared.

Two example programs that utilize the features provided by ON KBD and TDISP are shown next.

```
10   ! ************************************************************************
20   !
30   !  This example shows how the TDISP statement affects the current keyboard
40   !  entry line of the CRT.
50   !            Press SFK 0 - Cursor backspaces and clears 1 character.
60   !            Press SFK 1 - Display line data printed, display line cleared.
70   !  The counter that is displayed gives a current count of the number of
80   !  characters contained in the Key$ variable.  This value is set to 0
90   !  when the display line is cleared.
100  !  LIMITATIONS:
110  !            You canNOT backspace when there are no characters in the line.
120  !            You canNOT enter more than 160 characters into the line.
130  ! ************************************************************************
140  DIM Key$[160]                              ! Dimension the string
150       ON KBD GOSUB Tdisp                    ! On keyboard interrupt
160 Leng_disp:    !
170       DISP LEN(Key$)                        ! Display length of string
180       GOTO Leng_disp
190 Tdisp:        !
200       Kb$=KBD$                              ! Get the keypress data
210       IF NOT LEN(Kb$) THEN RETURN           ! Ensure that key data was entered
220       IF Kb$[1,1]=CHR$(255) THEN Keycheck   ! Check for SFK pressed
230       Key$=Key$&Kb$[1,1]                    ! Add keypress to the string
240       TDISP Kb$                             ! Add keypress to the display line
250       GOTO Tdisp
260  !
270 Keycheck:     !
280       IF NUM(Kb$[2])<2 THEN Goodkey         ! Check for defined SFK
290       BEEP
300       PRINT PAGE
310       PRINT "YOU PRESSED AN UNDEFINED KEY"
320       RETURN
330  !
340 Goodkey:      !
350       ON NUM(Kb$[2])+1 GOSUB Backsp,Clr     ! Find which SFK was pressed
360       GOTO Tdisp
370  !
380 Backsp:       !
390       TDISP CHR$(8)&" "&CHR$(8)             ! Backspace and clear character
400       Key$=Key$[1,LEN(Key$)-1]              ! Remove character from string
410       RETURN
420  !
430 Clr:          !
```

```
440       PRINT PAGE
450       PRINT "THE DISPLAY LINE IS CLEARED"
460       PRINT "THE DATA YOU ENTERED IS : ";Key$
470       Key$=""                              ! Clear the string
480       TDISP CHR$(12)                       ! Clear the display line
490       RETURN
500  !
510  END
```

```
10   ! ************************************************************************
20   !
30   !  This program uses the interrupt driven LINE MODE terminal program that
40   !  has been previously shown.  The ON KBD and TDISP statements are added
50   !  to show how these features may be used.
60   !  DEFINITIONS:
70   !             SFK 0 - Backspace 1 character.
80   !             SFK 1 - Clear the display line and transmit the data.
90   !             SFK 2 - Disconnect from the link
100  !             SFK 3 - Send a "BREAK" to the remote
110  ! ************************************************************************
120  DIM A$[80],B$[160]                   ! Dimension the string variables
130    CDISCONNECT 4                      ! Ensure you are disconnected
140    CCOM 1500                          ! Specify the CCOM memory allocation
150    CMODEL ASYNC,4                     ! Define the CMODEL (ALL DEFAULTS)
160    CCONNECT 4;HANDSHAKE OFF           ! Link is connected here
170    CCONTROL 4;ACK ON                  ! Enable ENQ-ACK handshake
180  !
190    ON INT #4,1 GOSUB Transmit         ! Interrupt when a PROMPT is detected
200    ON INT #5,2 GOSUB Receive          ! Interrupt when an INSEP is detected
210    ON KBD GOSUB Outkey                ! Interrupt when a key is pressed
220  !
230    CWRITE 4;ENDLINE                   ! Send C/R (OUTSEP) to host (I am ready)
240  !
250  Spin: GOTO Spin                      ! Wait here for an interrupt
260   !
270  Transmit: !
280    IF NOT CSTAT(4,2) THEN RETURN      ! Check for a PROMPT before continuing
290    CREAD 4;A$                         ! Get the visual prompt character
300    PRINT A$;                          ! Print the visual character (e.g.,":")
310    RETURN
320  !
330  Receive: !
340    IF NOT CSTAT(4,1) THEN RETURN      ! Check for an INSEP (ALERTN not used)
350    CREAD 4;A$                         ! Read the data from the host
360    PRINT A$                           ! Print the data from the host
370    RETURN
380  !
390  Outkey:   !
400    DISP ""                            ! Clear the display line
410    Kb$=KBD$                           ! Get keypress data
420    IF NOT LEN(Kb$) THEN RETURN        ! Ensure key was entered
430    IF Kb$[1,1]=CHR$(255) THEN Keyck   ! Check for SFK press
440    IF LEN(B$)>150 THEN GOSUB Err      ! Check for 150 chars, 160 is MAX
450    B$=B$&Kb$                          ! Add key data to xmit string
460    TDISP Kb$                          ! TDISP the keyboard data
470    RETURN
480  Err:      !
490    BEEP
500    DISP "WARNING, 150 CHARACTERS ENTERED, 160 IS THE MAXIMUM ALLOWED"
```

```
510     RETURN
520 Keyck:      !
530     IF NUM(Kb$[2])<4 THEN Goodkey      ! Check for defined key
540     BEEP                               ! UNDEFINED KEY MESSAGE ROUTINE
550     DISP "YOU PRESSED AN UNDEFINED KEY"
560     RETURN
570 Goodkey:    !
580     ON NUM(Kb$[2])+1 GOSUB Backsp,Clr,Disc,Brk
590     RETURN
600 Backsp:     !
610     IF NOT LEN(B$) THEN RETURN         ! CanNOT backspace on empty line
620     TDISP CHR$(8)&" "&CHR$(8)          ! Backspace and remove unwanted char.
630     B$=B$[1,LEN(B$)-1]                 ! Remove unwanted char from data string
640     RETURN
650 Clr:        !
660     CWRITE 4;B$,ENDLINE                ! Transmit the data string and ENDLINE
670     B$=""                              ! Clear the data string variable
680     TDISP CHR$(12)                     ! Clear the display line
690     RETURN
700 Disc:       !
710     CDISCONNECT 4                      ! Disconnect from the link
720     PRINT "LINK IS DISCONNECTED"
730     END
740 Brk:        !
750     CCONTROL 4;BREAK                   ! Execute a "BREAK"
760     RETURN
770 END
```

Note

The TDISP statement is available in the HP 9845 (except HP 9845A) I/O ROM as well as in the HP 98317A and 98417A Basic (Asynchronous) Datacomm ROMs. Whenever you record a program containing the TDISP statement by executing the STORE < file specifier > command, if the HP 9845 is equipped with and I/O ROM, the recorded program uses the I/O ROM addresses from TDISP whenever the program is reloaded using the LOAD command. If the program is subsequently loaded into another HP 9845 that has no I/O ROM present, a MISSING ROM error message results. To prevent this difficulty, use the SAVE statement, or remove the I/O ROM before you STORE the program. (Power must be OFF whenever you add or remove ROMs, so be sure you have your program safely SAVEd, or remove the I/O ROM before you program the computer.) If your application requires the I/O ROM as well as datacomm, for normal program activity, you should experience no difficulty.

Appendices
Table of Contents

Appendix A

ASCII Table

ASCII Char.	EQUIVALENT FORMS			
	Binary	Oct	Hex	Dec
NULL NU	00000000	000	00	0
SOH SH	00000001	001	01	1
STX SX	00000010	002	02	2
ETX EX	00000011	003	03	3
EOT ET	00000100	004	04	4
ENQ EQ	00000101	005	05	5
ACK AK	00000110	006	06	6
BELL BL	00000111	007	07	7
BS BS	00001000	010	08	8
HT HT	00001001	011	09	9
LF LF	00001010	012	0A	10
VT VT	00001011	013	0B	11
FF FF	00001100	014	0C	12
CR CR	00001101	015	0D	13
SO SO	00001110	016	0E	14
SI SI	00001111	017	0F	15
DLE DL	00010000	020	10	16
DC1 D1	00010001	021	11	17
DC2 D2	00010010	022	12	18
DC3 D3	00010011	023	13	19
DC4 D4	00010100	024	14	20
NAK NK	00010101	025	15	21
SYNC SY	00010110	026	16	22
ETB EB	00010111	027	17	23
CAN CN	00011000	030	18	24
EM EM	00011001	031	19	25
SUB SB	00011010	032	1A	26
ESC EC	00011011	033	1B	27
FS FS	00011100	034	1C	28
GS GS	00011101	035	1D	29
RS RS	00011110	036	1E	30
US US	00011111	037	1F	31

ASCII Char.	EQUIVALENT FORMS			
	Binary	Oct	Hex	Dec
space SP	00100000	040	20	32
!	00100001	041	21	33
"	00100010	042	22	34
#	00100011	043	23	35
$	00100100	044	24	36
%	00100101	045	25	37
&	00100110	046	26	38
'	00100111	047	27	39
(	00101000	050	28	40
)	00101001	051	29	41
*	00101010	052	2A	42
+	00101011	053	2B	43
,	00101100	054	2C	44
-	00101101	055	2D	45
.	00101110	056	2E	46
/	00101111	057	2F	47
0	00110000	060	30	48
1	00110001	061	31	49
2	00110010	062	32	50
3	00110011	063	33	51
4	00110100	064	34	52
5	00110101	065	35	53
6	00110110	066	36	54
7	00110111	067	37	55
8	00111000	070	38	56
9	00111001	071	39	57
:	00111010	072	3A	58
;	00111011	073	3B	59
<	00111100	074	3C	60
=	00111101	075	3D	61
>	00111110	076	3E	62
?	00111111	077	3F	63

ASCII Char.	EQUIVALENT FORMS			
	Binary	Oct	Hex	Dec
@	01000000	100	40	64
A	01000001	101	41	65
B	01000010	102	42	66
C	01000011	103	43	67
D	01000100	104	44	68
E	01000101	105	45	69
F	01000110	106	46	70
G	01000111	107	47	71
H	01001000	110	48	72
I	01001001	111	49	73
J	01001010	112	4A	74
K	01001011	113	4B	75
L	01001100	114	4C	76
M	01001101	115	4D	77
N	01001110	116	4E	78
O	01001111	117	4F	79
P	01010000	120	50	80
Q	01010001	121	51	81
R	01010010	122	52	82
S	01010011	123	53	83
T	01010100	124	54	84
U	01010101	125	55	85
V	01010110	126	56	86
W	01010111	127	57	87
X	01011000	130	58	88
Y	01011001	131	59	89
Z	01011010	132	5A	90
[	01011011	133	5B	91
\	01011100	134	5C	92
]	01011101	135	5D	93
^	01011110	136	5E	94
_	01011111	137	5F	95

ASCII Char.	EQUIVALENT FORMS			
	Binary	Oct	Hex	Dec
`	01100000	140	60	96
a	01100001	141	61	97
b	01100010	142	62	98
c	01100011	143	63	99
d	01100100	144	64	100
e	01100101	145	65	101
f	01100110	146	66	102
g	01100111	147	67	103
h	01101000	150	68	104
i	01101001	151	69	105
j	01101010	152	6A	106
k	01101011	153	6B	107
l	01101100	154	6C	108
m	01101101	155	6D	109
n	01101110	156	6E	110
o	01101111	157	6F	111
p	01110000	160	70	112
q	01110001	161	71	113
r	01110010	162	72	114
s	01110011	163	73	115
t	01110100	164	74	116
u	01110101	165	75	117
v	01110110	166	76	118
w	01110111	167	77	119
x	01111000	170	78	120
y	01111001	171	79	121
z	01111010	172	7A	122
{	01111011	173	7B	123
|	01111100	174	7C	124
}	01111101	175	7D	125
~	01111110	176	7E	126
DEL RO	01111111	177	7F	127

Appendix B

RS-232C/CCITT V24[1]

The following table provides information about each data communications interface function. The pin assignments are also shown. Not all of the functions provided by RS-232C are implemented. The functions denoted with an * **are implemented.**

RS-232C/CCITT V24[1]

RS-232C	CCITT V24	Signal Name
*Pin 1	101	PROTECTIVE GROUND. Electrical equipment frame and AC power ground.
■	■	TRANSMITTED DATA. Data originated by the terminal to be transmitted via the sending modem.
■	■	RECEIVED DATA. Data from the receiving modem in response to analog signals transmitted from the sending modem.
*Pin 4	105	REQUEST TO SEND. Indicates to the sending modem that the terminal is ready to transmit data.
*Pin 5	106	CLEAR TO SEND. Indicates to the terminal that its modem is ready to transmit data.
*Pin 6	107	DATA SET READY. Indicates to the terminal that its modem is not in a test mode and that modem power is ON.
■	■	SIGNAL GROUND. Establishes common reference between the modem and the terminal.
*Pin 8	109	DATA CARRIER DETECT. Indicates to the terminal that its modem is receiving carrier signals from the sending modem.
Pin 9		Reserved for test.
Pin 10		Reserved for test.
Pin 11		Unassigned.
*Pin 12	122	SECONDARY DATA CARRIER DETECT. Indicates to the terminal that its modem is receiving secondary carrier signals from the sending modem.
Pin 13	121	SECONDARY CLEAR TO SEND. Indicates to the terminal that its modem is ready to transmit signals via the secondary channel.

■ These signals are commonly used for 3 wire (no modem) links.

1 International Telephone and Telegraph Consultative Committee European standard.

RS-232C/CCITT V24 (Cont'd)

RS-232C	CCITT V24	Signal Name
Pin 14	118	SECONDARY TRANSMITTED DATA. Data from the terminal to be transmitted by the sending modem's channel.
*Pin 15	114	TRANSMITTER SIGNAL ELEMENT TIMING. Signal from the modem to the transmitting terminal to provide signal element timing information.
Pin 16	119	SECONDARY RECEIVED DATA. Data from the modem's secondary channel in response to analog signals transmitted from the sending modem.
*Pin 17	115	RECEIVER SIGNAL ELEMENT TIMING. Signal to the receiving terminal to provide signal element timing information.
Pin 18		Unassigned.
*Pin 19	120	SECONDARY REQUEST TO SEND. Indicates to the modem that the sending terminal is ready to transmit data via the secondary channel.
*Pin 20	108.2	DATA TERMINAL READY. Indicates to the modem that the associated terminal is ready to receive and transmit data.
Pin 21	110	SIGNAL QUALITY DETECTOR. Signal from the modem telling whether a defined error rate in the received data has been exceeded.
*Pin 22	125	RING INDICATOR. Signal from the modem indicating that a ringing signal is being received over the line.
*Pin 23	111	DATA SIGNAL RATE SELECTOR. Selects one of two signaling rates in modems having two rates.
*Pin 24	113	TRANSMIT SIGNAL ELEMENT TIMING. Transmit clock provided by the terminal.
Pin 25		Unassigned.

Appendix C

Modems

The following list contains modems that have been tested with the HP 98046A Interface.

- Bell 103J
- Bell 103A3
- Bell 212A
- Vadic 3400 series
- UDS 103J
- Anderson Jacobson Acoustic Coupler

Appendix D

Introduction

This section is provided as a guide to enable you to obtain the most efficient use of the Desktop Computer memory for datacomm operations. Topics covered are:

- Memory Allocation
- Memory Reclamation

Memory Allocation

The following should be considered when determining memory space allocation:

- The CCOM statement defines the total memory allocation for a specific datacomm operation. The maximum value allowed for the CCOM statement is 32 718.
- There can be more than one CMODEL statement associated with one CCOM statement.
- The CCOM statement specifies the combined memory allocation for ALL associated CMODELs.
- The MEMLIMIT parameter specifies the amount of CCOM memory allocation required by this CMODEL for the input and output queues. The MEMLIMIT parameter must NOT exceed 31 500.
- The INBUFFER and TBUFFER parameters specify the amount of CCOM memory allocation required by this CMODEL for the INBUFFER and the TBUFFER. The combined allocation for INBUFFER and TBUFFER must NOT exceed 31 000.
- The total memory required by a specific CMODEL is the sum of MEMLIMIT, INBUFFER, TBUFFER, plus the overhead for that CMODEL.

NOTE

The descriptions for the CCOM, CMODEL, INBUFFER, and TBUFFER can be found in Chapter 2.

There is a simple way to determine the amount of memory required for a particular CCOM statement.

1. Create maximum CCOM size (CCOM 32 000).
2. Execute the associated CMODEL statement.
3. Execute CSTAT (select code,4) for the select code of the CMODEL statement.
4. The CSTAT function returns an integer that indicates the minimum memory allocation required for this CMODEL. Note this value.
5. Repeat steps 2 through 4 for all CMODELS that are associated with this CCOM statement.
6. The summed total of ALL of the values returned by the CSTAT statement +24 (for overhead) is the MINIMUM memory allocation required by the CCOM statement.

Memory Parameters

The following additional information is provided about the memory parameters that are associated with the CMODEL statement. The parameters described here are:

- MEMLIMIT
- INBUFFER
- TBUFFER

The MEMLIMIT Parameter

The MEMLIMIT parameter is provided so that you can specify the maximum number of words that may be used for input queue and output queue space.

NOTE

The MEMLIMIT parameter does not specify the number of text characters that may be stored due to end-of-line characters and other overhead.

Input and Output Queue Limits

Both the input queue and the output queue can occupy a MAXIMUM of two-thirds of the memory space specified by the MEMLIMIT parameter. This feature is provided to allow for one-sided communications without limiting the amount of space for one operation while the space for the other operation is not being used. See the following description.

You are receiving large amounts of data from a host computer, but you are transmitting very little information to the computer. The input queue fills up rapidly, while the output queue remains rather empty. The input queue is allowed to use as much as two-thirds of the memory space specified by the MEMLIMIT parameter while the output queue uses the remainder. The reverse situation exists if you are transmitting large amounts of data while receiving very little.

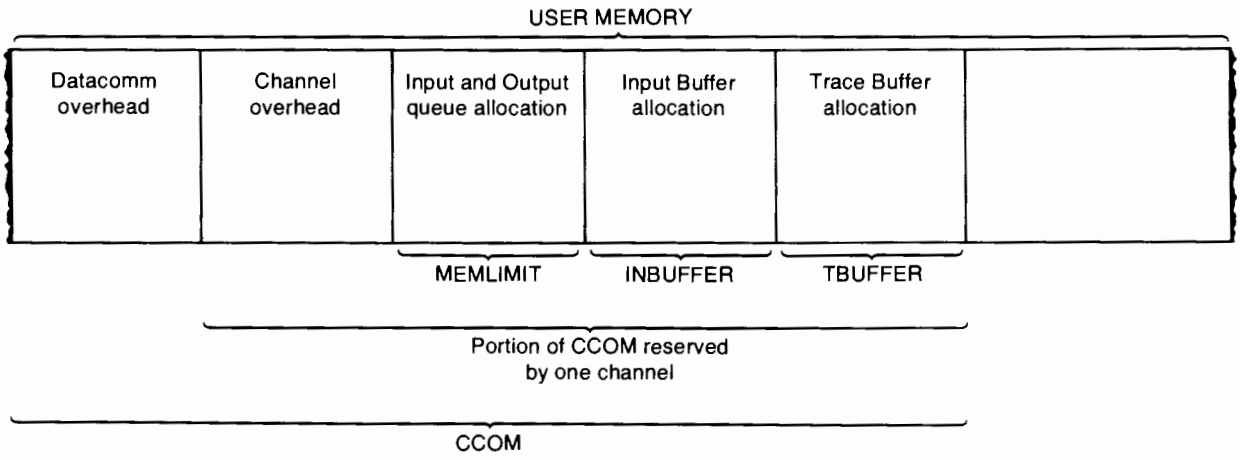
NOTE
The maximum amount of space used by the input queue and output queue combined canNOT exceed the value specified by the MEMLIMIT parameter.

INBUFFER and TBUFFER Allocation

The INBUFFER parameter provides enough space[1] for at least the amount of control and data bytes specified.

The TBUFFER parameter provides enough space[1] for at least the amount of control and data bytes specified by the expression.

Memory Allocation



Increasing the MEMLIMIT parameter increases the smoothing effect between BASIC and the real-time operations of the interface. Increasing MEMLIMIT allows more data to be held in the output queue for transmission and allows more data to be held in the input queue for entry into the BASIC strings in the program.

Increasing the INBUFFER parameter allows datacomm to accept more data from the remote, supplements the length of the receive queue, and allows more characters to be input between the "ready" (ACK or DC1) and "not ready" (no ACK or DC3) handshake signals.

Increasing the TBUFFER parameter allows more control codes and data to be copied into the trace buffer.

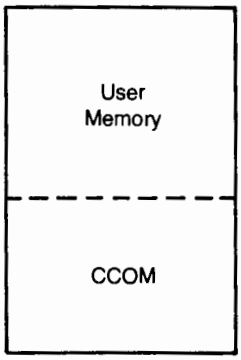
Increasing CCOM provides more memory space so that you can increase the MEMLIMIT, INBUFFER, or TBUFFER parameter. CCOM does not increase these parameters.

1 The INBUFFER and TBUFFER reouire 1 byte of overhead for each 10 characters that are entered.

Memory Reclamation

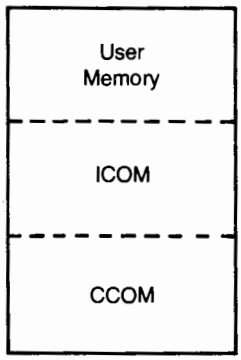
Once memory space has been allocated by a CCOM statement, that memory space remains allocated until it is specifically reclaimed. The following figures show how memory is allocated for CCOM, ICOM[1], COM and user memory.

Figure 1



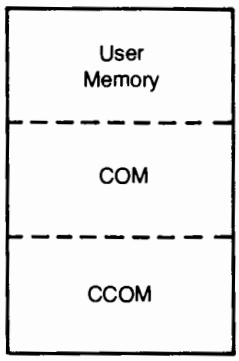
Only CCOM allocated

Figure 2



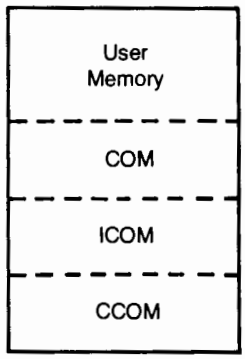
CCOM and ICOM allocated

Figure 3



CCOM and COM allocated

Figure 4



CCOM, ICOM,
and COM allocated

As shown in Figure 1, the CCOM memory allocation can be reduced and the remaining space reclaimed for user memory.

As shown in Figures 2, 3, and 4, the CCOM memory allocation can be reduced but the remaining space canNot be reclaimed for user memory due to allocations for ICOM and/or COM.

When a program containing a CCOM 0 statement is executed, all memory space allocated by a previous CCOM statement is returned to user memory if:

- No ICOM or Common region exists.

  or

- SCRATCH C has been executed and no subsequent COMmon or ICOM statement has been executed.

CCOM memory allocation is always reclaimed when SCRATCH A is executed.

1 ICOM is an Assembly Language statement.

Appendix E

Half Duplex

Half duplex protocols require that the line be turned around when the terminal or host has completed its data transmission and is ready to receive data from the device at the other end of the link.

The following example programs show how to implement half duplex protocols and line turn around procedures. Note that one program is for a terminal and one program is for a controller. The terminal has initial control of the line.

```
10   ! ************************************************************************
20   !                  HALF-DUPLEX CONTROLLER PROGRAM
30   !
40   ! This program shows how to transfer data using HALF-DUPLEX protocols.
50   ! This program utilizes the MAIN datacomm channel.
60   !
70   !                  Messages are sent as follows:
80   !                              <STX>
90   !                            message
100  !                              <ETX>
110  ! 1 - Reception of the <ETX> character means it is time to transmit data.
120  ! 2 - Transmission is automatically started with CWRITE <STX>.
130  ! 3 - Data is transmitted one line at a time.
140  ! 4 - End of transmission is indicated with CWRITE <ETX>,EOT.
150  ! 5 - The remote does not echo, so this program provides its own echo.
160  !
170  ! Note that INSEP and OUTSEP characters are re-defined (CMODEL statement)
180  ! ************************************************************************
190  !
200  DIM A$[80]                          ! Dimension the string variable
210  CCOM 1500                           ! Allocate memory
220  Etx$=CHR$(3)                        ! ASCII <ETX> character
230  Stx$=CHR$(2)                        ! ASCII <STX> character
240  !
250  CMODEL ASYNC,4;HALF DUPLEX,PROMPT=Etx$,OUTSEP=CHR$(13)&CHR$(10),INSEP=CHR$(
13)
260  !
270  CCONNECT 4;HANDSHAKE ON             ! Connect to the link
280  !
290 Send: !
300  IF NOT CSTAT(4,2) OR CSTAT(4,1) OR CSTAT(4,13) THEN Receive
310  !
320  CREAD 4;A$                          ! Read the prompt message
330  PRINT A$;                           ! Print the prompt (and STX if there)
340  CWRITE 4;Stx$                       ! Send STX to get IMMEDIATE line control
350  !
360 Enterdata: !
370  !
```

```
380  LINPUT "Enter data, press CONTINUE with no data to give up the line",A$
390  IF LEN(A$) THEN Datasend                ! There is data to send
400  CWRITE 4;":",Etx$,EOT                  ! Send prompt and relinquish line
410  GOTO Receive
420  !
430 Datasend: !
440  CWRITE 4;A$,ENDLINE                     ! Send data and C/R but keep the line
450  PRINT A$                                ! Print your transmitted data (echo)
460  GOTO Enterdata                          ! Get more data to transmit
470  !
480 Receive: !
490  !
500  IF NOT CSTAT(4,1) THEN Send             ! Check for input data
510  CREAD 4;A$                              ! Get the input data
520  PRINT A$                                ! Print input data (and STX if there)
530  GOTO Receive
540  END
```

```
10   ! ************************************************************************
20   !                 HALF-DUPLEX TERMINAL PROGRAM
30   !
40   ! This program shows how to transfer data using HALF-DUPLEX protocols.
50   ! This program utilizes the MAIN datacomm channel.
60   !
70   !                 Messages are sent as follows:
80   !                              <STX>
90   !                            message
100  !                             <ETX>
110  ! 1 - Reception of the <ETX> character means it is time to transmit data.
120  ! 2 - Transmission is automatically started with CWRITE <STX>.
130  ! 3 - Data is transmitted one line at a time.
140  ! 4 - End of transmission is indicated with CWRITE <ETX>,EOT.
150  ! 5 - The remote does not echo, so this program provides its own echo.
160  !
170  ! ************************************************************************
180  !
190  DIM A$[80]                              ! Dimension the string variable
200  CCOM 1500                               ! Allocate memory
210  Etx$=CHR$(3)                            ! ASCII <ETX> character
220  Stx$=CHR$(2)                            ! ASCII <STX> character
230  !
240  CMODEL ASYNC,4;HALF DUPLEX,PROMPT=Etx$
250  !
260  CCONNECT 4;HANDSHAKE ON                 ! Connect to the link
270  CWRITE 4;ENDLINE,Etx$,ENDLINE           ! Send C/R and relinquish the line
280  !
290 Send: !
300  IF NOT CSTAT(4,2) OR CSTAT(4,1) OR CSTAT(4,13) THEN Receive
310  !
320  CREAD 4;A$                              ! Read the prompt message
330  PRINT A$;                               ! Print the prompt (and STX if there)
340  CWRITE 4;Stx$                           ! Send STX to get IMMEDIATE line control
350  !
360  LINPUT "Enter your data",A$
370  CWRITE 4;A$,ENDLINE,Etx$,EOT            ! Send data, C/R, ETX, and release line
380  PRINT A$                                ! Print your transmitted data (echo)
390  !
400 Receive: !
410  !
420  IF NOT CSTAT(4,1) THEN Send             ! Check for input data
430  CREAD 4;A$                              ! Get the input data
440  PRINT A$                                ! Print input data (and STX if there)
450  GOTO Receive
460  END
```

Secondary Channel

The following example programs show how to implement secondary channel communication. These programs utilize half duplex protocols. Note that one program is for a terminal and one program is for a controller. The terminal has initial control of the line.

```
10   ! ************************************************************************
20   !          SECONDARY CHANNEL -  HALF-DUPLEX CONTROLLER PROGRAM
30   !
40   ! This program shows how to transfer data using SECONDARY CHANNEL.
50   ! This program utilizes the HALF-DUPLEX protocols.
60   !
70   !                Messages are sent as follows:
80   !                             DRIVER2 OFF
90   !                              message
100  !                             DRIVER2 ON
110  ! 1 - DRIVER2 controls Secondary-Request-to-Send
120  ! 2 - The controller can transmit when Secondary-Carrier-Detect is true.
130  ! 3 - The controller does not have initial control of the line.
140  ! 4 - Data is received one line at a time.
150  ! 5 - The controller does not expect any prompts from the terminal.
160  ! 6 - Terminators (INSEP and OUTSEP) are reversed.
170  !
180  ! The terminal interprets our Secondary-Request-to-Send as Secondary-
190  ! Carrier-Detect.
200  ! ************************************************************************
210  !
220       DIM A$[80],B$[160]                 ! Dimension the string variable
230       CCOM 1500                          ! Allocate memory
240       CMODEL ASYNC,4;HALF DUPLEX,INSEP=CHR$(13),OUTSEP=CHR$(13)&CHR$(10)
250  !
260       CCONNECT 4;HANDSHAKE ON            ! Connect to the link
270       CCONTROL 4;DRIVER2 ON              ! Give line control to terminal
280  Write:     !
290       IF NOT CSTAT(4,12) OR CSTAT(4,1) THEN Read
300       CREAD 4;A$                         ! Enter input data
310       PRINT A$;                          ! Print the data
320       CCONTROL 4;DRIVER2 OFF             ! Take control of the line
330  Dataget:  LINPUT "ENTER DATA, PRESS 'CONTINUE' WITH NO DATA TO RELINQUISH LI
NE",B$
340       IF NOT LEN(B$) THEN Line_give      ! Check for data or line relinquish
350       CWRITE 4;B$,ENDLINE                ! Transmit data and CR
360       GOTO Dataget
370  Line_give: !
380       CWRITE 4;":",EOT                   ! Send prompt to the terminal
390       CCONTROL 4;DRIVER2 ON              ! Relinquish line control to terminal
400  Wait:      !
410       IF CSTAT(4,12) THEN Wait           ! Wait for remote to accept line control
420  Read:      !
430       IF NOT CSTAT(4,1) THEN Write       ! Check for prompt
440       CREAD 4;A$                         ! Get the input data
450       PRINT A$                           ! Print the input data
460       GOTO Read
470  END
```

```
10   ! ************************************************************************
20   !           SECONDARY CHANNEL -  HALF-DUPLEX TERMINAL PROGRAM
30   !
40   ! This program shows how to transfer data using SECONDARY CHANNEL.
50   ! This program utilizes the HALF-DUPLEX protocols.
60   !
70   !                  Messages are sent as follows:
80   !                              DRIVER2 OFF
90   !                               message
100  !                              DRIVER2 ON
110  ! 1 - DRIVER2 controls Secondary-Request-to-Send
120  ! 2 - This terminal can transmit when Secondary-Carrier-Detect is true.
130  ! 3 - This terminal uses default INSEP and OUTSEP.
140  ! 4 - This terminal has initial control of the line.
150  ! 5 - The remote does not echo, so this program provides its own echo.
160  ! 6 - The only PROMPT indication expected is a line turn-around.
170  !
180  ! The controller interprets our Secondary-Request-to-Send as Secondary-
190  ! Carrier-Detect.
200  ! ************************************************************************
210  !
220       DIM A$[80],B$[160]              ! Dimension the string variable
230       CCOM 1500                       ! Allocate memory
240       CMODEL ASYNC,4;HALF DUPLEX
250  !
260       CCONNECT 4;HANDSHAKE ON         ! Connect to the link
270  Statck:     !
280       IF NOT CSTAT(4,12) THEN Statck  ! Wait for Secondary-Carrier-Detect
290       CCONTROL 4;DRIVER2 OFF          ! Get line control
300       CWRITE 4;"READY",ENDLINE,EOT    ! Send ready message and CR
310       CCONTROL 4;DRIVER2 OFF          ! Give up line control
320       IF CSTAT(4,12) THEN Wait        ! See if remote has accepted line
330  Write:      !
340       IF NOT CSTAT(4,12) OR CSTAT(4,1) THEN Read
350       CREAD 4;A$                      ! Enter input data
360       DISP A$                         ! Display the input data
370       CCONTROL 4;DRIVER2 OFF          ! Take control of the line
380       LINPUT "ENTER DATA",B$          ! Enter your data for transmission
390       PRINT A$;B$                     ! Print input prompt, echo your data
400       CWRITE 4;B$,ENDLINE,EOT         ! Transmit data and CR
410       CCONTROL 4;DRIVER2 ON           ! Give up line control
420  Wait:       !
430       IF CSTAT(4,12) THEN Wait        ! Wait for remote to accept line control
440  Read:       !
450       IF NOT CSTAT(4,1) THEN Write    ! Check for prompt
460       CREAD 4;A$                      ! Get the input data
470       PRINT A$                        ! Print the input data
480       GOTO Read
490  END
```

Appendix F

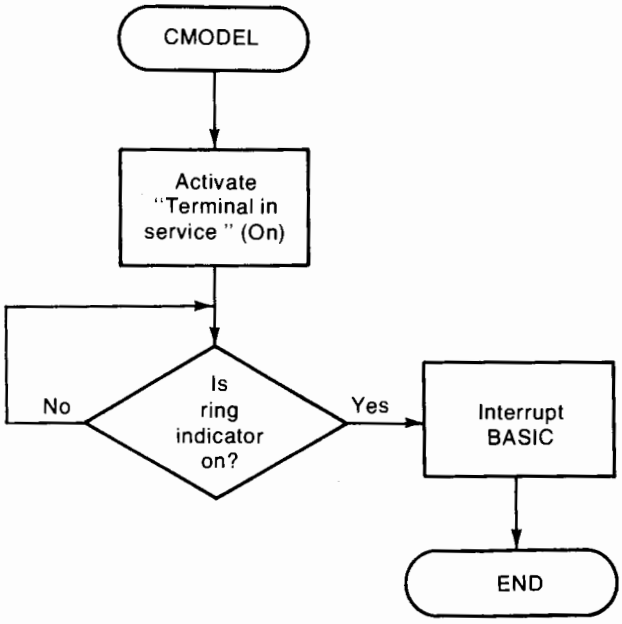
Autoanswer Implementation

The CMODEL ASYNC statement provides for automatic answering of an incoming call. When the CMODEL ASYNC statement is executed, the RING INDICATOR modem signal is monitored and an interrupt is generated for the even select code when the RING INDICATOR signal is detected. An interrupt service routine then executes the CCONNECT statement for the specified channel.

Here is an example of an autoanswer program:

```
10   ! ************************************************************************
20   !  The following program is for AUTO-ANSWER applications.  After the CMODEL
30   !  statement is executed, the program waits for an interrupt from the EVEN
40   !  select code (the interrupt indicates that a ringing signal has been
50   !  detected).  The program then branches to an interrupt service routine
60   !  which executes the CCONNECT statement and returns to normal program
70   !  execution.  No CCONTROL...INTMASK statement is required.
80   ! ************************************************************************
90   !
100  !  Line 180 - Program waits here until the ringing interrupt is detected.
110  !  Line 210 - Executes the CCONNECT statement.
120  !
130  ON INT #4 GOTO Connect                ! Enable interrupt
140  DIM A$[80],B$[160]                    ! Dimension the string variables
150       CDISCONNECT 4                    ! Ensure you are disconnected
160       CCOM 1500                        ! Specify memory allocation
170       CMODEL ASYNC,4                   ! Define the CMODEL
180 Waiting:  GOTO Waiting                 ! STAY PUT UNTIL INTERRUPTED
190  !                                       -------------------------
200 Connect:  !
210       CCONNECT 4                       ! Connect to the link (HANDSHAKE ON)
220       OFF INT #4                       ! Disable interrupts
230       CCONTROL 4;ACK ON                ! Specify ENQ-ACK handshake
240       CWRITE 4;ENDLINE                 ! Send C/R to the host
250  !
260 Promptcheck:  IF NOT CSTAT(4,2) OR CSTAT(4,1) THEN Readcheck
270  !
280       CREAD 4;A$                       ! Get the prompt
290       PRINT A$;                        ! Print the prompt
300       LINPUT B$                        ! Enter your data here
310       CWRITE 4;B$,ENDLINE              ! Send data and C/R
320  !
330 Readcheck:  IF NOT CSTAT(4,1) THEN Promptcheck
340  !
350       CREAD 4;A$                       ! Read data from the host
360       PRINT A$                         ! Print data from the host
370  GOTO Promptcheck
380  !
390  END
```

Autoanswer Flowchart



Appendix G

Programs

This section contains the following example programs:

- Line Mode
- Character Mode
- Desktop-to-Desktop

Line Mode

```
10   ! ************************************************************************
20   !
30   !  The following program is an example of an interrupt driven LINE MODE
40   !  terminal emulator.  Defaults are assumed where possible.  This is for
50   !  a direct (NO MODEM) connection to an H.P. 3000.  If a modem is used,
60   !  delete "HANDSHAKE OFF" from the CCONNECT statement (line 140).
70   !       ********************************************************
80   !       !! YOU MUST PRESS THE CONTINUE KEY TO TRANSMIT THE DATA !!
90   ! ************************************************************************
100  DIM A$[80],B$[160]                    ! Dimension the string variables
110       CDISCONNECT 4                    ! Ensure you are disconnected
120       CCOM 1500                        ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4                   ! Define the CMODEL (ALL DEFAULTS)
140       CCONNECT 4;HANDSHAKE OFF         ! Link is connected here
150       CCONTROL 4;ACK ON                ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit       ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive        ! Interrupt when an INSEP is detected
190  !
200       CWRITE 4;ENDLINE                 ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin                       ! Wait here for an interrupt
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN    ! Check for a PROMPT before continuing
260       CREAD 4;A$                       ! Get the visual prompt character
270       PRINT A$;                        ! Print the visual character (e.g.,":")
280       LINPUT "Enter data",B$           ! Enter data, press CONT key to send
290       CWRITE 4;B$,ENDLINE              ! Send data and C/R (OUTSEP)
300       RETURN
310  !
320  Receive: !
330       IF NOT CSTAT(4,1) THEN RETURN    ! Check for an INSEP (ALERTN not used)
340       CREAD 4;A$                       ! Read the data from the host
350       PRINT A$                         ! Print the data from the host
360  GOTO Receive                          ! Go check for more input data
370  !
380  END
```

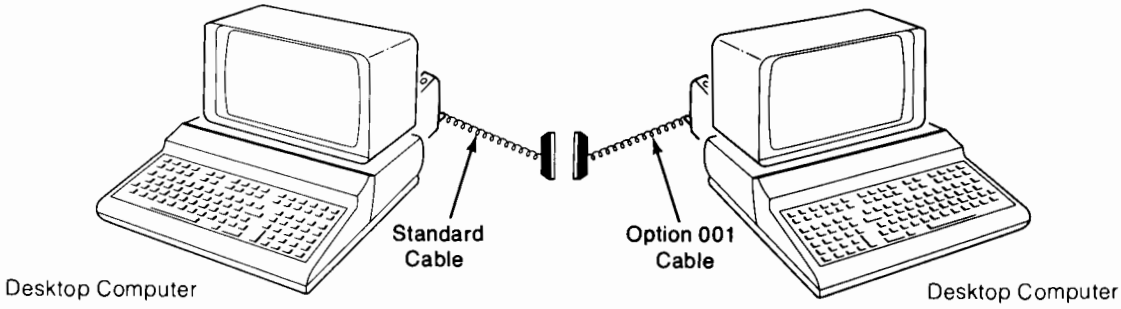
Character Mode

```
10   ! ************************************************************************
20   !
30   !  The following program is an interrupt driven CHARACTER MODE terminal
40   !  emulator.  This program uses the LINE MODE program previously shown.
50   !  The lines that have been changed or added are highlighted.  The System
60   !  35 requires an I/O ROM for this program.  This program is for a direct
70   !  (NO MODEM) connection to an H.P. 3000.  If modems are used, delete
80   !  HANDSHAKE OFF from the CCONNECT statement (line 140).
90   ! ************************************************************************
100  DIM A$[1]                              ! New dimension statement
110       CDISCONNECT 4                     ! Ensure you are disconnected
120       CCOM 1500                         ! Specify the CCOM memory allocation
130       CMODEL ASYNC,4;ALERTN=1           ! ALERTN parameter added
140       CCONNECT 4;HANDSHAKE OFF          ! Link is connected here
150       CCONTROL 4;ACK ON                 ! Enable ENQ-ACK handshake
160  !
170       ON INT #4,1 GOSUB Transmit        ! Interrupt when a PROMPT is detected
180       ON INT #5,2 GOSUB Receive         ! Interrupt (INSEP or ALERTN characters)
185       ON KBD 3 GOSUB Outkey             ! Interrupt when a key is pressed
190  !
200       CWRITE 4;ENDLINE                  ! Send C/R (OUTSEP) to host (I am ready)
210  !
220  Spin: GOTO Spin
230  !
240  Transmit: !
250       IF NOT CSTAT(4,2) THEN RETURN     ! Check for a PROMPT before continuing
260       CREAD 4;A$                        ! Get the visual prompt character
270       PRINT A$;                         ! Print the visual character (e.g.,":")
271       RETURN
272  Outkey:!
273       B$=KBD$                           ! Put key data in string for output
274       IF NOT LEN(B$) THEN RETURN        ! Ensure key was not missed
275       IF NUM(B$)<255 THEN Datasend      ! Transmit key data only if key is ASCII
276       CWRITE 4;ENDLINE                  ! Transmit a C/R (OUTSEP)
277       RETURN                            ! Go back to Spin
278  Datasend:!
279       CWRITE 4;B$                       ! Transmit the keypress data
280  !    **** LINE 280 DELETED ****
290  !    **** LINE 290 DELETED ****
300       RETURN                            ! Go back to Spin
310  !
320  Receive:!
330       IF NOT CSTAT(4,1) THEN RETURN     ! Check for INSEP or ALERTN characters
340       CREAD 4;A$                        ! Read the data from the host
345       IF CSTAT(4,3) THEN Endprint       ! Check for an INSEP
346       PRINT A$;                         ! Print input data with no C/R - L/F
347  GOTO Receive                           ! Check for more input data
348  Endprint: !
350       PRINT A$                          ! Print input data with a C/R - L/F
360  GOTO Receive                           ! Check for more input data
370  !
380  END
```

Desktop-to-Desktop

The following two programs enable 2 Desktop Computers to communicate with each other. The programs assume that the computers are connected as shown.

98046 Interface



Note that 1 interface MUST be a 98046 standard configuration while the other interface MUST be a 98046 Option 001 configuration.

The only difference in the programs is the CMODEL statement. The INSEP and OUTSEP parameters for program B must be defined so that the line terminator sequences for the two computers can be correctly interpreted.

Either Desktop Computer can initiate the conversation, and neither computer can transmit until a prompt sequence has been detected.

```
10   ! ************************************************************************
20   !
30   !              THIS PROGRAM IS USED BY "DESKTOP COMPUTER A"
40   !  This program is a line mode emulator.  Note that you must send a PROMPT
50   !  to the other Desktop Computer.  No type of handshake is specified.
60   !  Either Desktop can initiate the conversation.
70   ! ************************************************************************
80   !
90   !
100  !
110  DIM A$[80],B$[80],Prompt$[80]          ! Dimension the string variables
120  Prompt$=":"&CHR$(17)                   ! Set up the PROMPT sequence
130       CDISCONNECT 4                     ! Ensure you are disconnected
140       CCOM 1500                         ! Specify memory allocation
150       CMODEL ASYNC,4                    ! Define the CMODEL
160       CCONNECT 4;HANDSHAKE OFF          ! Link is connected here
170       CWRITE 4;Prompt$                  ! Send PROMPT to the other Desktop
180  !
190 Promptcheck:  IF NOT CSTAT(4,2) OR CSTAT(4,1) THEN Readcheck
200  !
210       CREAD 4;A$                        ! Get the PROMPT from Desktop B
220       PRINT A$;                         ! Print the PROMPT
230  !
240 Datain: LINPUT "ENTER YOUR MESSAGE(S); PRESS CONTINUE WITH NO ENTRY TO TRANS
FER CONTROL.",B$
250  !
260       IF LEN(B$)=0 THEN GOTO Prompts    ! Check for blank data (send PROMPT)
270       CWRITE 4;B$,ENDLINE               ! Send data and end-of-line
280       GOTO Datain                       ! Get more data
```

```
290  !
300 Prompts:     !
310         CWRITE 4;Prompt$                    ! Send PROMPT to Desktop B
320         GOTO Promptcheck
330  !
340 Readcheck:  IF NOT CSTAT(4,1) THEN Promptcheck
350         CREAD 4;A$                          ! Read data from Desktop B
360         PRINT A$                            ! Print data from Desktop B
370  GOTO Promptcheck
380  !
390  END
```

```
10    ! ******************************************************************************
20    !
30    !             THIS PROGRAM IS USED BY "DESKTOP COMPUTER B"
40    !  This program is a line mode emulator.  Note that you must send a PROMPT
50    !  to the other Desktop Computer.  No type of handshake is specified.
60    !  Either Desktop can initiate the conversation.  Note that the INSEP and
70    !  OUTSEP specifiers have been redefined.  This is because these specifiers
80    !  are expected by DESKTOP COMPUTER A.
90    ! ******************************************************************************
100   !
110   !
120   !
130   DIM A$[80],B$[80],Prompt$[80]           ! Dimension the string variables
140   Prompt$=":"&CHR$(17)                    ! Set up the PROMPT sequence
150         CDISCONNECT 4                     ! Ensure you are disconnected
160         CCOM 1500                         ! Specify memory allocation
170   !
180         CMODEL ASYNC,4;OUTSEP=CHR$(13)&CHR$(10),INSEP=CHR$(13)
190   !
200         CCONNECT 4;HANDSHAKE OFF          ! Link is connected here
210         CWRITE 4;Prompt$                  ! Send PROMPT to the other Desktop
220   !
230 Promptcheck:  IF NOT CSTAT(4,2) OR CSTAT(4,1) THEN Readcheck
240   !
250         CREAD 4;A$                        ! Get the PROMPT from Desktop A
260         PRINT A$;                         ! Print the PROMPT
270   !
280 Datain: LINPUT "ENTER YOUR MESSAGE(S); PRESS CONTINUE WITH NO ENTRY TO TRANS
FER CONTROL.",B$
290   !
300         IF LEN(B$)=0 THEN GOTO Prompts    ! Check for blank data (send PROMPT)
310         CWRITE 4;B$,ENDLINE               ! Send data and end-of-line
320         GOTO Datain                       ! Get more data
330   !
340 Prompts:     !
350         CWRITE 4;Prompt$                  ! Send PROMPT to Desktop A
360         GOTO Promptcheck
370   !
380 Readcheck:  IF NOT CSTAT(4,1) THEN Promptcheck
390         CREAD 4;A$                        ! Read data from Desktop A
400         PRINT A$                          ! Print data from Desktop A
410   GOTO Promptcheck
420   !
430   END
```

Subject Index

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

w

x

ROM Errors

8	Improper parameter matching
19	Improper value
300	CCOM area not allocated
301	Not allowed if channel is active
302	CMODEL statement required
303	Not allowed when trace is active
304	Synchronous error only
305	New CCOM size not allowed when channel is active
306	98046 card failure
307	Insufficient CCOM allocation
308	Synchronous error only
309	Not allowed for this CMODEL
310	CCONNECT statement required
311	Not allowed while datacomm is suspended
312	Improper CSTATUS array

CSTATUS (0) Errors

10	Timeout before connection
11	Clear to send line false, or missing clock
100	Channel MEMLIMIT overflow
102	Input buffer overflow
103	Internal buffer overflow
104	Autodisconnect forced
106	NOACTIVITY timeout
200	98046 buffer overflow

Part No. 98046-90010
Microfiche No. 98046-99010

Printed in U.S.A.
March 1, 1980