DataGeneral

RCX70
Reference Manual
(AOS)

093-000172-00

RCX70
Reference Manual
(AOS)

093-000172-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

rdering No. 093-000172

Revision 00, June 1979

NOTICE

RCX70
Reference Manual
(AOS)
093-000172

Revision History:

Original Release - June 1979

Preface

Between the covers of this manual you will find all the reference material you will need to create and maintain the RCX70 system for your installation. This book documents programmer's and system reference material, as well as system generation procedures.

We wrote this book for you. In doing so, we encountered a slight problem; there are four RCX70 modes of operation available. Hence, there are four types of YOU out there as well. Chapter 1 provides an overview of all the RCX70 modes of operation. All of you should read Chapter 1, then use the following chapter summaries to determine which chapters apply to your particular RCX70 installation.

Chapter 1 Provides an overview of all RCX70 modes of operation.

Chapter 2 Provides a summary of some Data General DASHER™ Display Keyboard functions and commands. It also introduces a sample screen format that you'll refer to often as you read through the manual.

Chapter 3 Discusses RCX70 commands which you can use under any mode of operation.

Chapter 4 Contains information on direct IBM 3271 emulation. It discusses general synchronous-line concepts plus RCX70 commands applicable in a synchronous-line mode of operation.

Chapter 5 Contains information on how to interface RCX70 to another AOS process. It includes a brief section on Interprocess Communications (IPCs), but you should read the *AOS Programmer's Manual* for a complete discussion of them.

Chapter 6 Describes a dual-interface mode of operation which uses both 3271 emulation (synchronous line) and the AOS interprocess (IPC) interface.

Chapter 7 Describes how to bring up and run RCX70. It includes configuration and initialization techniques for all of the possible modes of operation. Chapter 7 also supplies operator commands, telling you how to assign and release terminals once you have brought RCX70 up.

Appendix A Contains all RCX70 error messages.

Appendix B Is a table of ASCII and EBCDIC address representations. Use it to generate screen formats and to determine screen addresses.

Appendix C Tells you how to configure your own ASCII and EBCDIC tables if you do not have a standard ASCII keyboard.

Suggested Manuals

In addition to this manual, you should read the following manuals:

- *AOS Programmer's Manual* (093-000120)
- *RCX70 Terminal Operator's Guide* (093-000170)
- *Data General Communications System* (014-000070)

Reader, Please Note:

We use these conventions for command formats in this manual:

COMMAND required *[optional]* ...

Where	Means
COMMAND	You must enter the command (or its accepted abbreviation) as shown.
required	You must enter some argument (such as a filename). Sometimes, we use: $\{\text{required}_1 \atop \text{required}_2\}$ which means you must enter *one* of the arguments. Don't enter the braces; they only set off the choice.
[optional]	You have the option of entering this argument. Don't enter the brackets; they only set off what's optional.
...	You may repeat the preceding entry or entries. The explanation will tell you exactly what you may repeat.

Additionally, we use certain symbols in special ways:

Symbol	Means
)	Press the NEW LINE or RETURN key on your terminal's keyboard.
□	Be sure to put a space here. (We use this only when we must; normally, you can see where to put spaces.)

All numbers are decimal unless we indicate otherwise; e.g., 35_8.

Finally, in examples we use

THIS TYPEFACE TO SHOW YOUR ENTRY)
THIS TYPEFACE FOR SYSTEM QUERIES AND RESPONSES.

) is the AOS CLI prompt.

End of Preface

Contents

Chapter 1 - Introduction

Chapter 2 - Keyboard Operations

Chapter 3 - Global Commands and Orders

Chapter 4 - Direct 3271 Emulation

Chapter 5 - RCX70 As a Terminal Handler for Other AOS Processes

Chapter 6 - Dual Interface (Distributed Processing Mode)

Chapter 7 - Printer, Configuration, Initialization, and Operator Commands

Appendix A - RCX70 Error Messages

Appendix B - EBCDIC and ASCII Address Representations

Appendix C - Creating EBCDIC-to-ASCII or ASCII-to-EBCDIC Optional Files for RCX70GEN

Illustrations

Tables

Table Caption

Chapter 1
Introduction

What is AOS RCX70?

RCX70 is a software building block that you can use to build distributed processing systems. RCX70 allows communication between any AOS equipped Data General system and an IBM 360/370 host processor via a synchronous communications line, using the IBM Binary Synchronous Communication (BCC) protocol. To the host system, RCX70 appears to be an IBM 3271 terminal cluster controller.

RCX70 runs as a user process under the Advanced Operating System (AOS). Used strictly as a 3271 controller emulator, RCX70 can control up to 16 Data General 6052 or 6053 DASHER™ Displays. You may connect the terminals directly to your ECLIPSE® system, or you may attach them via asynchronous modems and a telephone line. RCX70 also provides 3270 compatible line printer support. Idea (Interactive data entry/access) software and other AOS processes can use RCX70 to provide communication to the host computer.

Modes of Operation

There are four possible modes of operation for AOS RCX70. We will tell you how to configure and initialize each one in Chapter 7. We limit ourselves here to discussing how each mode works. Please refer to Figure 1-1 as you read the following text.

Direct 3271 Emulation (Synchronous-Line) Mode

Figure 1-1A illustrates direct 3271 emulation. Users who connect their RCX70 system to an IBM host computer via a synchronous communications line enter data on a Data General DASHER Display terminal. Once the user completes input, RCX70 will transmit the data over the synchronous line to the host computer. The host computer will process the data and send it back to RCX70. RCX70 will then route the data to the proper user terminal.

This mode is particularly suited to you if you need to emulate an IBM 3271 controller but you also wish to take advantage of local AOS processing capabilities. While RCX70 controls some of your AOS terminals, you can use all AOS capabilities from your remaining terminals. You can put terminals under RCX70's control and remove them while RCX70 is running. Chapter 4 contains more information about the direct synchronous-line mode of operation.

Terminal Handling (IPC) Mode

AOS RCX70 includes an interface to another process running in the same AOS ECLIPSE computer. This interface, illustrated in Figure 1-1B, uses the Interprocess Communication (IPC) mechanism. Once a user completes input to RCX70, RCX70 sends the data to the local application, which processes it and sends it back to RCX70. The data formats used to describe screen data are the same for the synchronous-line and IPC modes of operation.

The IPC mode provides a screen-handling facility that you can use to write applications in languages such as COBOL, PL/I, or DG/L™ . You may develop new applications by taking advantage of these screen-handling capabilities.

Distributed Processing (Dual-Interface) Mode

The distributed processing (see Figure 1-1C) mode contains aspects of both the 3271 emulator mode and the AOS terminal handler mode. It provides a true distributed processing capability.

The ECLIPSE system is connected to a host via a synchronous communications line. Both RCX70 and a user application are running under AOS. The user enters data at an RCX70 terminal. When (s)he completes the input, RCX70 sends it to the local application via the IPC mechanism. Then, the application processes the input.

In many cases, the application will have all the information it needs in the local database. Then, it will send its reply to RCX70, which will send it to the proper terminal.

If it cannot process the input, the local application may have to send the data to the host via the synchronous line. First, it sends the data to RCX70; then RCX70 transmits the data to the host. Thus RCX70 takes care of all the details of 3270 synchronous-line protocol. When the host finishes processing the data, it sends its reply down the synchronous line to RCX70. And RCX70 sends it to the local application for any further processing. Finally, when all processing is complete, the local application sends its reply to RCX70, which in turn sends it to the originating terminal.

IBM 3270 Protocol Emulator Mode

Figure 1-1D illustrates RCX70 as a 3270 protocol emulator. You can configure an RCX70 system without any terminal support. For example, you can use Idea to communicate with terminals and to provide local processing. The Idea processes use RCX70 to communicate with the host. Any number of Idea processes can use RCX70 simultaneously. And any other AOS process can make use of RCX70 in exactly the same way that Idea does. This mode provides a second way to create a distributed processing environment.

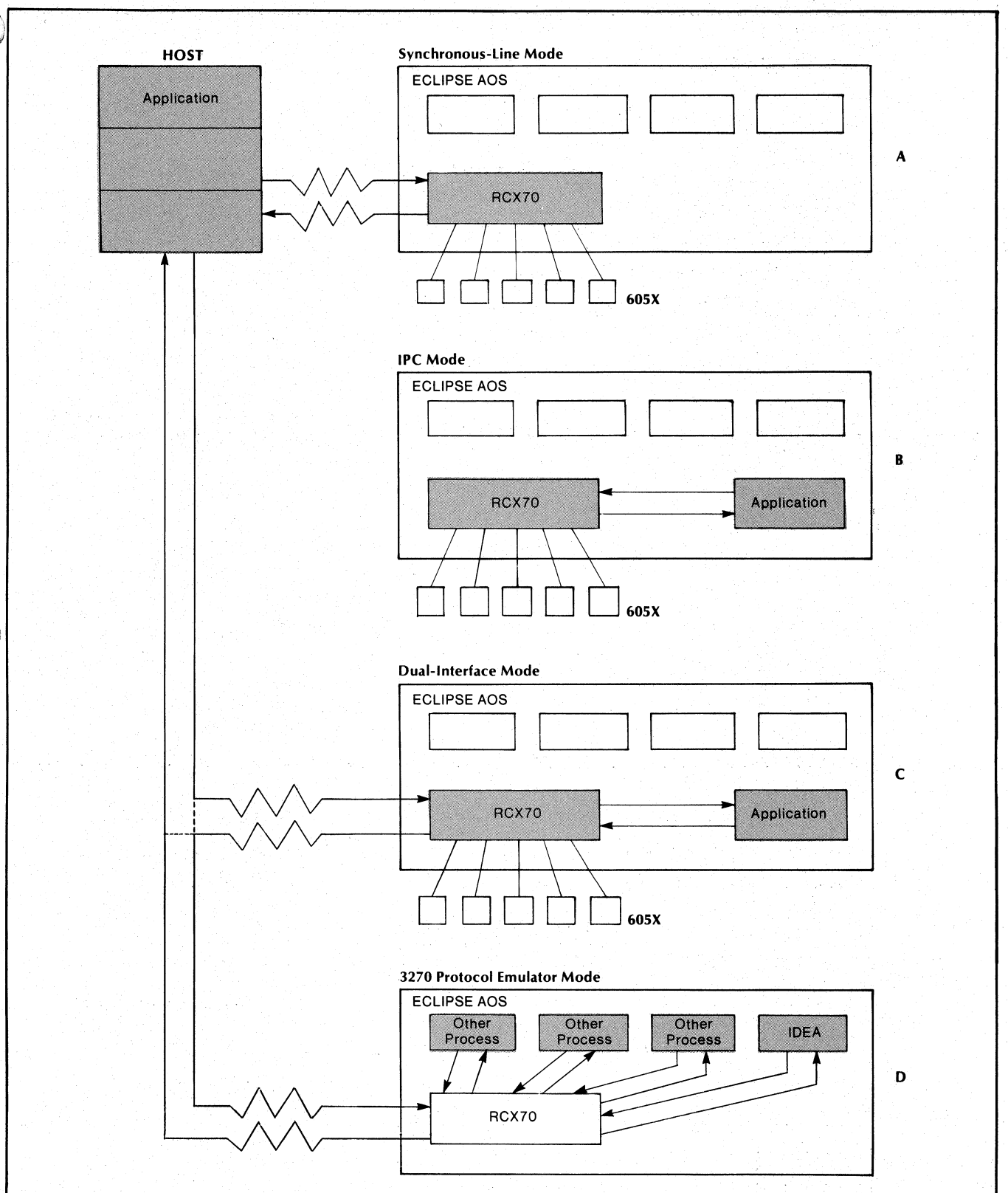


Figure 1-1. RCX70 Modes of Operation

End of Chapter

Chapter 2
Keyboard Operations

The *RCX70 Terminal Operator's Guide* contains a complete guide to keyboard functions, but we give you a brief summary here.

Two keyboards are available for your RCX70 system: Data General terminal models 6052 and 6053. Both keyboards provide the basic alphanumeric typewriter key layout.

Main Keyboard

The basic keyboard (6052) has a main keyboard with 52 keys. Each key generates a single code each time you strike it. You may alter the key code by depressing the CTRL and/or SHIFT key simultaneously with the alphanumeric. The 6052 keyboard lets you generate codes for 64 uppercase ASCII characters as well as 30 control codes.

The enhanced keyboard (6053) has a main keyboard with 56 keys. Each key generates a single code each time you strike it. You may alter the key code by depressing the CTRL and/or SHIFT key simultaneously with the alphanumeric. The 6053 keyboard lets you generate codes for 96 upper- and lowercase ASCII characters as well as 30 control codes.

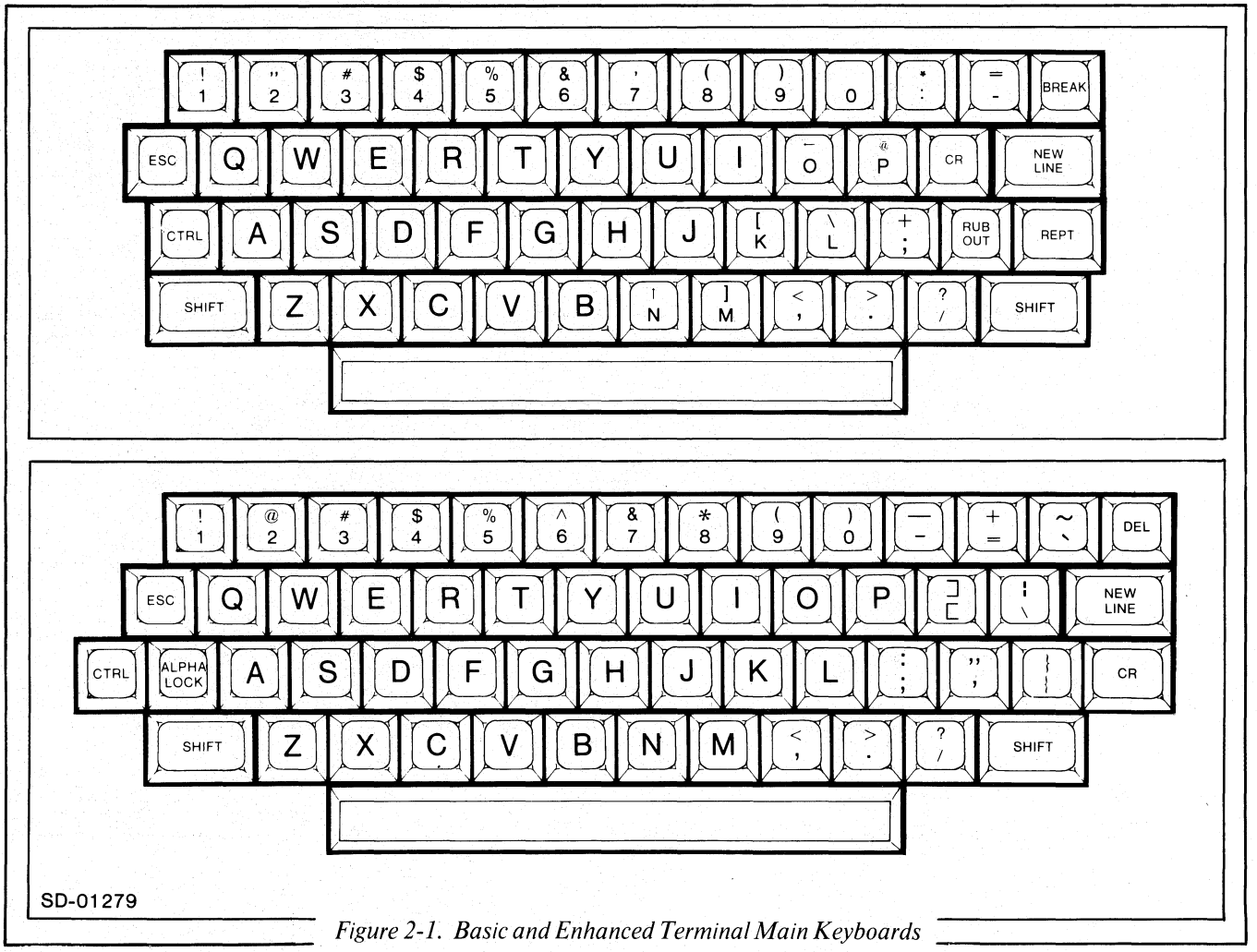


Figure 2-1. Basic and Enhanced Terminal Main Keyboards

Numeric Keyboard

The numeric keyboard consists of 11 keys in a configuration similar to that of an electronic calculator. Each key generates a single code every time you strike it. The numeric keys generate ASCII code for the ten decimal digits (0 through 9) and the period character. The CTRL and SHIFT keys have no effect on the numeric keyboard.

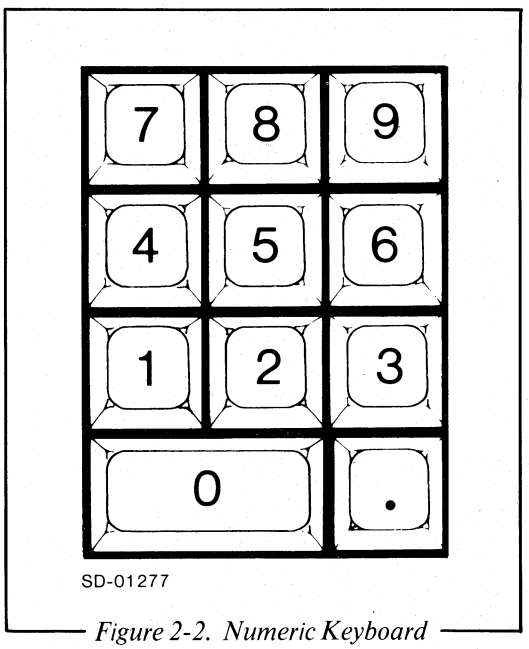


Figure 2-2. Numeric Keyboard

User Function Keys and the RCX70 Template

The basic (6052) keyboard contains 8 user function keys. The enhanced (6053) keyboard contains 11 user function keys. Each of these keys generates a sequence of two codes every time you strike it. The first code of the sequence is always the user function header code. The second code depends on the user function key that you strike and on whether you simultaneously depress the CTRL and/or SHIFT key.

We have included a keyboard template in the RCX70 documentation package. The template fits over the row of 8 user function keys and provides you with the ability to generate 24 separate codes from the 8 keys. Figure 2-4 illustrates the template, but you should read the *RCX70 Terminal Operator's Guide* to familiarize yourself with individual key functions.

Function Keypad

The basic keyboard's function keypad consists of 10 keys; the enhanced keyboard's function keypad has 12 keys. Each of these keys, except the PRINT and unlabeled keys, generates a single code each time you press it. These codes specify cursor control and screen management functions to the display.

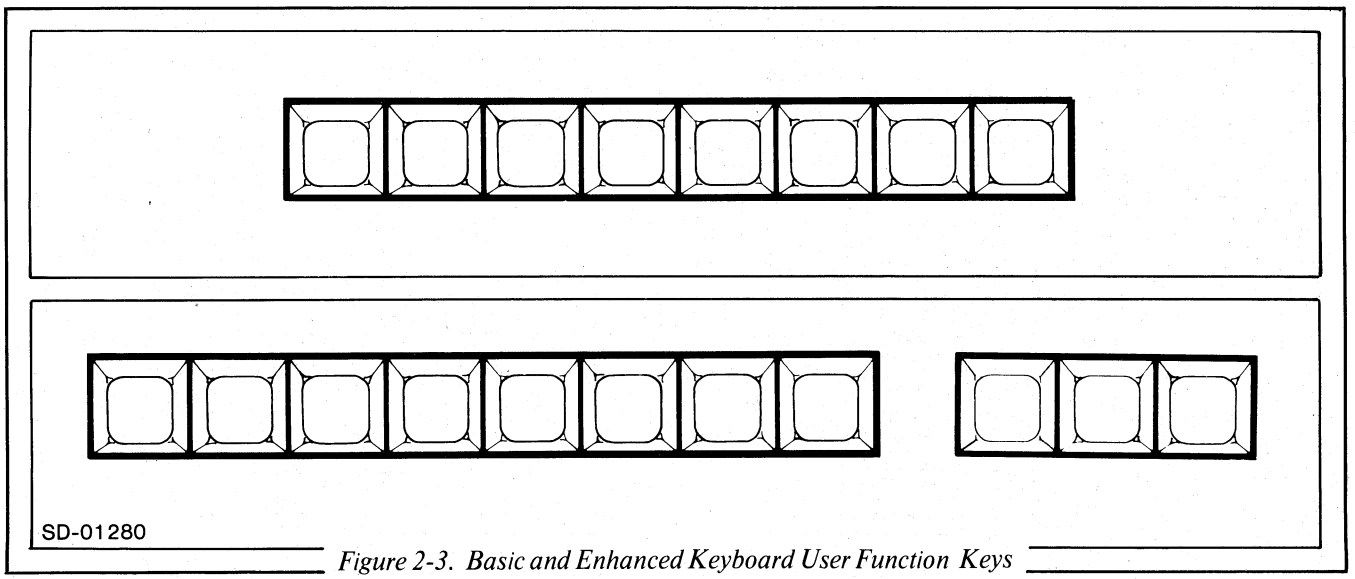
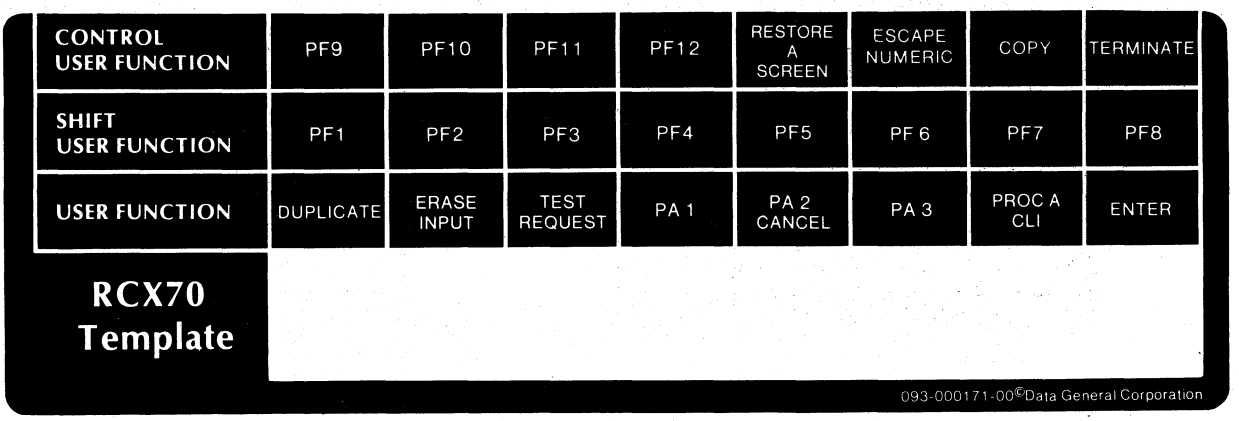
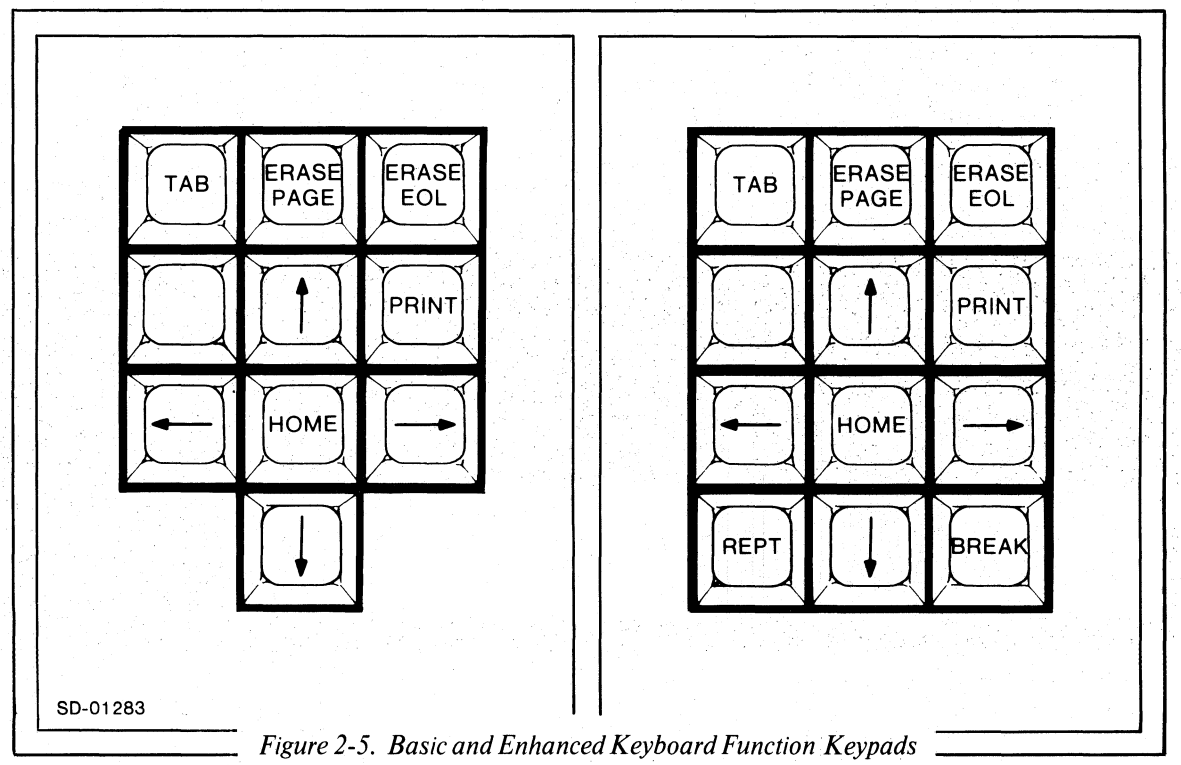


Figure 2-3. Basic and Enhanced Keyboard User Function Keys



SD-01282

Figure 2-4. RCX70 Keyboard Template



SD-01283

Figure 2-5. Basic and Enhanced Keyboard Function Keypads

Display Operations

When you type in or receive data on an RCX70 display station, it appears on a cathode ray tube (CRT) screen as either alphanumeric characters (A through Z, 0 through 9) or special symbols. The following section describes Data General's display stations and their features. The 6052 and 6053 CRTs are identical except that the 6053 has upper- and lowercase capabilities and an underscore function, which the 6052 doesn't have.

Formatted and Unformatted Screens

A screen consists of 24 80-character lines or a total of 1920 character positions. A screen can be either formatted or unformatted at any particular time. *A formatted screen* is an image that has been divided into groups of contiguous character positions called fields. In addition to the group of character positions, each field contains an attribute byte describing the legal contents of those positions. In most cases, the application program formats these fields. *An unformatted screen* has not been subdivided; you can use the screen freely.

Figure 2-6 illustrates a formatted screen display. In this example, the solid characters represent data that is displayed; triangles represent locations storing attribute bytes; shaded characters represent data you can't display. In any display image, you (the operator) will never see the attribute byte. It will appear on your screen as a blank position.

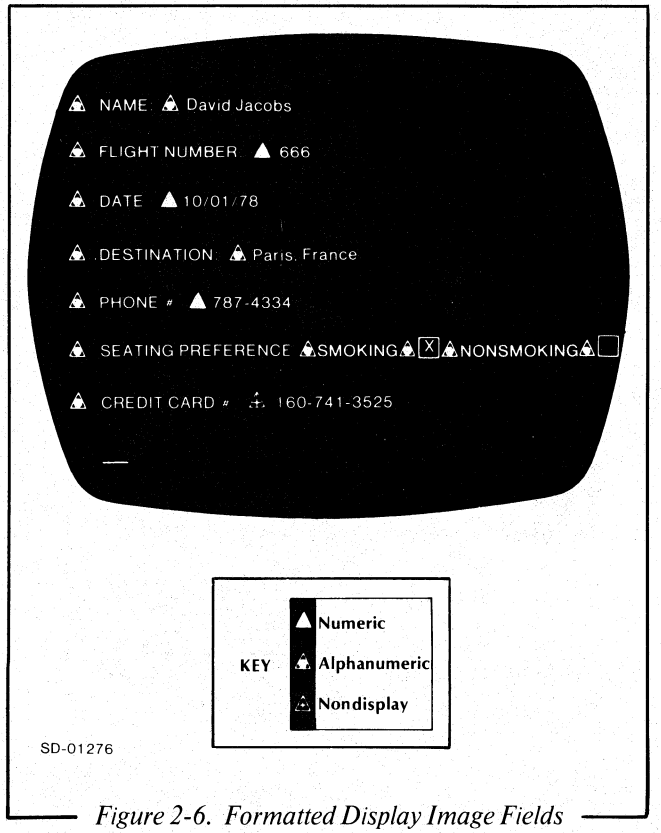


Figure 2-6. Formatted Display Image Fields

Fields

The application defines the attribute bytes (triangles in Figure 2-6) to contain the field's attributes. The attribute byte is always the first byte in a field. A field, then, is a set of character positions starting with an attribute byte and ending with the last character position before the next attribute byte. You cannot alter the attributes of a field from the keyboard.

In addition to defining the start of a field, an attribute byte defines a particular field's characteristics:

Protected Fields — A user cannot modify a protected field. You might use protected fields for labels, headings or prompts. In Figure 2-6, the protected fields contain the words NAME, FLIGHT NUMBER, DATE, DESTINATION, PHONE #, SEATING PREFERENCE, SMOKING, NONSMOKING, and CREDIT CARD #

Unprotected Fields — The application will designate input fields as unprotected fields; that is, you can type data into and modify unprotected fields. In Figure 2-6 the unprotected fields contain the words David Jacobs, 666, 10/01/78, Paris, France, 787-4334, X, and 160-741-3525.

Alphanumeric Fields — In an alphanumeric, unprotected field, you can enter alphabetic, numeric, and all other characters.

Numeric Fields — A numeric field will accept only the numeric characters 0 through 9, the period (decimal point), and the minus sign. In Figure 2-6 the expected field contents for the responses to FLIGHT NUMBER and PHONE # are unprotected numerics.

Character Display — Either nondisplay or display. If the attribute byte specifies a nondisplay field, you will not see any of the input keyed into that field. In Figure 2-6 the field contents for the response to CREDIT CARD # are unprotected, nondisplay alphanumerics.

Attribute bytes occupy one character location in the buffer. They will never appear on a display or printout and, except for the MDT bit discussed later, they are always protected from operator modification.

Key Functions

This section will give you a general overview of some basic RCX70 key functions. For a more detailed description of key functions you should read the *RCX70 Terminal Operator's Guide.*

Tables 2-1 and 2-2 illustrate the EBCDIC and ASCII character sets. If you are using a 6053 terminal, you can enter alphabetic characters into the display buffer in upper- or lowercase code. If you are using a 6052 terminal, you can enter only uppercase alphabetic characters.

When you enter an alphanumeric character in an alphanumeric, unprotected data field, it will appear above the cursor and the cursor will advance to the next character location in the unprotected field. RCX70 blocks any attempt to enter data into a protected field or an attribute byte.

Cursor

RCX70's cursor (displayed on your screen as an underscore) indicates the position where you can currently enter or modify text. For example, if the cursor is positioned under one character in a line of characters, you can modify that character by pressing the appropriate key. If the cursor is positioned under a blank space on the display screen, you can insert a character in that position. If the cursor is positioned beneath a protected field or attribute byte, you cannot modify that position (even if it is blank).

When you start RCX70, it will automatically display the cursor beneath the first location on the screen. From that point on, either you or the application can reposition the cursor to any place on the screen. You can place the cursor in any field (including protected and nondisplay fields) but, again, you cannot modify certain fields.

RCX70 provides an automatic-skip function for the cursor. After you enter a character into the last position of an unprotected data field, RCX70 repositions the cursor according to the attribute byte that defines the next field. If the attribute byte defines the next field as numeric and unprotected or as alphanumeric and either protected or unprotected, RCX70 positions the cursor after the attribute byte and under the first character location in that field. If the attribute byte defines the next field as numeric and protected, the cursor skips that field entirely; RCX70 positions the cursor under the first character location in the next unprotected field.

Cursor-Move Keys

You use four keys on the function keypad to move the cursor: ↑ (up), ← (left), → (right), and ↓ (down). These keys move the cursor into adjoining character locations one position at a time. The Backspace key, located on the main keyboard and also labeled ←, performs the same function as the cursor-move left key. You can use the cursor-move keys to position the cursor to any character location, regardless of the location's attribute status.

The cursor-move up and down keys cause the cursor to wrap. Vertical cursor wrap involves no horizontal movement; the cursor remains in the same character column. If you press the cursor-move up key when the cursor is on the top line, RCX70 will reposition the cursor to the bottom line. If you press the cursor-move down key when the cursor is at the bottom of the screen, RCX70 will reposition the cursor to the top of the screen.

Field-Move Keys

You can use three keys to position the cursor to the first position in a field.

Backtab

The Backtab key is the blank key on your function keypad. If you strike Backtab, the cursor will move to the first character position of the preceding unprotected field. If you strike Backtab when all fields are protected, RCX70 will position the cursor at screen location 0.

NEW LINE

The NEW LINE key is located on your main keyboard. It moves the cursor to the first character position of the first unprotected field on the following line. If there are no unprotected fields on the following line, the cursor searches through the remaining lines until it finds an unprotected field. If there are no unprotected fields, RCX70 positions the cursor to screen location 0.

TAB and Skip

The TAB and Skip key is the top, left key on your function keypad. It moves the cursor to the first character location in the next unprotected field. If the screen has no unprotected fields, RCX70 positions the cursor to screen location 0.

Delete (6053) or RUBOUT (6052)

The Delete (DEL) key is on your main keyboard. If you are using the enhanced keyboard (6053), the DEL key is located at the top, right corner. If you are working from the basic keyboard (6052), the Delete key is labeled RUBOUT and located on the right side of the third line.

Table 2-1. EBCDIC Character Set

Bits 4567	0,1 / 2,3 / Hex 0 1	00 00	00 01	00 10	00 11	01 00	01 01	01 10	01 11	10 00	10 01	10 10	10 11	11 00	11 01	11 10	11 11
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE			SP	&	-									0
0001	1	SOH	SBA					/		a	j			A	J		1
0010	2	STX	EUA		SYN					b	k	s		B	K	S	2
0011	3	ETX	IC							c	l	t		C	L	T	3
0100	4									d	m	u		D	M	U	4
0101	5	PT								e	n	v		E	N	V	5
0110	6	EA		ETB						f	o	w		F	O	W	6
0111	7			ESC	EOT					g	p	x		G	P	X	7
1000	8									h	q	y		H	Q	Y	8
1001	9		EM							i	r	z		I	R	Z	9
1010	A					¢	!	¦	:								
1011	B					.	$	,	#								
1100	C	FF	DUP		RA	<	*	%	@								
1101	D		SF	ENQ	NAK	(	)	-	;								
1110	E		FM			+	;	>	=								
1111	F		ITB		SUB	¦	¬	?	"								

SD-01307

Table 2-2. ASCII Character Set

Bits 4	3	2	1	7 6 5 / Hex 0 1	0 0 0 / 0	0 0 1 / 1	0 1 0 / 2	0 1 1 / 3	1 0 0 / 4	1 0 1 / 5	1 1 0 / 6	1 1 1 / 7
0	0	0	0	0	NUL	DLE	SP	0	@	P		p
0	0	0	1	1	SOH	SBA	|!	1	A	Q	a	q
0	0	1	0	2	STX	EUA	"	2	B	R	b	r
0	0	1	1	3	ETX	IC	#	3	C	S	c	s
0	1	0	0	4	EOT	RA	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	EA	SYN	&	6	F	V	f	v
0	1	1	1	7		ETB		7	G	W	g	w
1	0	0	0	8			(	8	H	X	h	x
1	0	0	1	9	PT	EM	)	9	I	Y	i	y
1	0	1	0	A	NL	SUB	*	:	J	Z	j	z
1	0	1	1	B		ESC	+	;	K	[	k	
1	1	0	0	C	FF	DUP	,	<	L	\	l	
1	1	0	1	D		SF	-	=	M	]	m	
1	1	1	0	E		FM	.	>	N	¬ ^	n	
1	1	1	1	F		ITB	/	?	O	_	0	

SD-01308

You use DEL or RUBOUT to correct typing errors. When you strike this key, RCX70 deletes the character to the left of the cursor. It also shifts all characters which are to the right of the cursor, on the same line, and in the current unprotected field one character to the left. RCX70 will also insert nulls into vacant character locations at the end of the line. If the field occupies more than one line, RCX70 will shift-left only those characters which are in the line identified by the cursor.

In order to use the Delete key successfully, the character location to the left of the cursor must be in an unprotected field; otherwise, the keyboard will stop accepting input and ring its bell. Note that when you press DEL or RUBOUT, the cursor moves one character to the left.

Duplicate

The Duplicate key is the user function 1 key. The row of user function keys is located at the top of the keyboard. Figure 2-4 illustrates the RCX70 keyboard template, which will help you use the user function keys.

The Duplicate key inserts a special duplicate character into the display buffer and directs RCX70 to perform a standard Tab command. When you press the Duplicate key, a right bracket will appear on the screen and RCX70 will send the application an 034 (EBCDIC and ASCII). The Duplicate character informs the application that a particular field can be duplicated. The application determines what happens at this point; that is, a duplication will not immediately or necessarily occur.

ERASE EOF (Erase to End of Field)

The Erase to End of Field key is the ERASE EOL key on your function keypad (see Figure 2-5). This key clears the cursor location and all locations to the right of the cursor up to the next attribute byte or the end of that line or screen, whichever comes first. Then RCX70 inserts nulls into all the cleared positions. The cursor does not move and RCX70 notes that you have modified the field.

If you strike this key while in a protected field or on an attribute byte, RCX70 does nothing. ERASE EOF is particularly useful if you want to replace the current contents of a field with new data; e.g., replace spaces with information.

Erase Input

The Erase Input key is in the row of user function keys at the top of your terminal (see Figure 2-3). To direct RCX70 to perform the erase input function, you must hold down the user function 2 key.

This key inserts nulls in all unprotected fields and repositions the cursor to the first character location in the screen's first unprotected field. If the screen contains no unprotected fields, RCX70 repositions the cursor to screen location 0. It inserts no null characters and marks all fields as not yet modified.

If the entire screen is unprotected, RCX70 inserts null characters into all buffer locations and positions the cursor at screen location 0.

Field Mark

The Field Mark key is the left square bracket on your main keyboard. On an unformatted screen, you can use this key to tell the application that it has reached the end of a field. You can also use it in an unprotected field to denote a variable number of subfields.

When you strike the Field Mark key, RCX70 receives a 133 from the keyboard, inserts an FM code (Field Mark, 036_8 in both EBCDIC and ASCII) into the buffer and displays a left bracket on your screen.

If you try to insert a Field Mark in a protected field or on an attribute byte, the keyboard will stop accepting input and echo a bell. Then you must strike ESC to continue typing.

Insert Mode

You enter Insert mode by simultaneously pressing the CTRL and E keys. Then when you strike any other character, RCX70 will display that character and insert it into the screen buffer, providing one of the following conditions is true:

1. There is a null character directly above the cursor.
2. There is a null character in the field somewhere to the right of the cursor.

If neither condition is true, the keyboard will stop accepting input and echo a bell. Then you must strike ESC to continue in Insert mode.

When you insert a character into an unprotected field, RCX70 shifts all following characters in that field one position to the right.

RCX70 will automatically exit from Insert mode when you leave the current field. To exit from Insert mode at any other time, you must strike CTRL-E again.

Reset

The Reset key is the ESC key on your main keyboard. It allows you to recover from a data entry error after the keyboard has stopped accepting input. It also allows you to begin typing after an application has written to the screen.

Escape Numeric Mode

You enter Escape Numeric mode by simultaneously pressing the CTRL key and the user function 6 key. If you use this key while in a numeric field (and without Data General's extended attribute byte set), RCX70 allows you to enter a sequence of nonnumeric characters.

Once you enter Escape Numeric mode, the field is in effect a nonnumeric field, so you can enter any character. However, the attribute byte does not change.

If the numeric bit is not set, RCX70 will report an error. If you try to enter nonnumeric data without first striking the Escape Numeric mode keys, the keyboard will continue to accept input. However, when you fill the field or hit a delimiter, RCX70 will move the cursor to the first illegal character and ring the bell. You must then hit the Reset key to continue typing.

Program Attention Keys

The program attention keys are Clear (ERASE PAGE), CANCEL, ENTER, TEST REQUEST, the program access (PA) keys, and all program function (PF) keys. All of the program attention keys are application dependent; that is, the keys solicit application action.

End of Chapter

Chapter 3
Global Commands and Orders

An RCX70 command is a directive to RCX70 from either a host application sending via the synchronous line or from a local application sending via the AOS IPC mechanism. Keyboard users do not issue actual RCX70 commands from their keyboards.

Two RCX70 commands, Write and Erase/Write, allow you to mix orders with your data. Orders perform several functions within a Write or Erase/Write command, including defining fields and updating the current buffer position.

You can use the commands and orders discussed in this chapter under any RCX70 mode of operation. When a command or order is the same for all modes, we provide all the information you will need to implement it. When the syntax and command codes differ from synchronous-line modes to IPC modes, we provide a command overview with a reference to the appropriate section of another chapter.

RCX70 executes two general types of commands to control application operations:

Read commands — Transfer RCX70 buffer data, key data, and status information from remote configurations to the host application. RCX70 can execute total or partial reads.

Write commands — Transfer orders and total or partial data from the host application to the RCX70 system. You can include two types of orders in Write data streams. RCX70 uses buffer orders to position and format all data transferred to the buffer, to erase unprotected buffer data, and to reposition the cursor. RCX70 executes immediately executable buffer orders as soon as it receives the data from the application. The second type of order, the print order, determines printer formats. RCX70 stores print orders in the buffer as data and executes them only during a print operation.

Table 3-1 is a multipurpose table which you'll often refer to in Chapters 3, 4, 5, and 6. You use this table to determine the hexadecimal (hex) codes that RCX70 will transmit for attribute characters, AID characters, and cursor addresses. RCX70 encodes information in only the six, low-order bits (bits 2-7) of any character. Then, prior to transmission, it assigns the appropriate EBCDIC or ASCII code.

For example, if you want to determine the hex code transmitted for an attribute byte, you first determine the values of bits 2 to 7. Then find the proper bit configuration under Bits 2-7 in Table 3-1. The hex code that the control unit will transmit (EBCDIC or ASCII) appears to the right of that bit configuration.

Table 3-1. Graphic Representation of 6-bit Codes

Bits 2-7	Graphic	EBCDIC	ASCII	Bits 2-7	Graphic	EBCDIC	ASCII
00 0000	□	40	20	10 0000	-	60	2D
00 0001	A	C1	41	10 0001	/	61	2F
00 0010	B	C2	42	10 0010	S	E2	53
00 0011	C	C3	43	10 0011	T	E3	54
00 0100	D	C4	44	10 0100	U	E4	55
00 0101	E	C5	45	10 0101	V	E5	56
00 0110	F	C6	46	10 0110	W	E6	57
00 0111	G	C7	47	10 0111	X	E7	58
00 1000	H	C8	48	10 1000	Y	E8	59
00 1001	I	C9	49	10 1001	Z	E9	5A
00 1010	¢ , [1	4A	5B	10 1010	¦	6A	7C
00 1011	.	4B	2E	10 1011	,	6B	26
00 1100	<	4C	3C	10 1100	%	6C	25
00 1101	(	4D	28	10 1101	_	6D	5F
00 1110	+	4E	2B	10 1110	>	6E	3E
00 1111	|!1	4F	21	10 1111	?	6F	3F
01 0000	&	50	26	11 0000	0	F0	30
01 0001	J	D1	4A	11 0001	1	F1	31
01 0010	K	D2	4B	11 0010	2	F2	32
01 0011	L	D3	4C	11 0011	3	F3	33
01 0100	M	D4	4D	11 0100	4	F4	34
01 0101	N	D5	4E	11 0101	5	F5	35
01 0110	O	D6	4F	11 0110	6	F6	36
01 0111	P	D7	50	11 0111	7	F7	37
01 1000	Q	D8	51	11 1000	8	F8	38
01 1001	R	D9	52	11 1001	9	F9	39
01 1010	!,]1	5A	5D	11 1010	:	7A	3A
01 1011	$	5B	24	11 1011	#	7B	23
01 1100	*	5C	2A	11 1100	@	7C	40
01 1101	)	5D	29	11 1101	'	7D	27
01 1110	;	5E	3B	11 1110	=	7E	3D
01 1111	, ^1	5F	5E	11 1111	"	7F	22

[1] EBCDIC graphic on left, ASCII graphic on right.

Table 3-2 illustrates the Read, Write, and Control commands and the corresponding codes that RCX70 executes.

Table 3-2. RCX70 Command Codes

Command	EBCDIC Hex	ASCII	Graphic
	Synchronous-line and IPC commands		
Read Buffer	F2	32	2
Read Modified	F6	36	6
Write	F1	31	1
Erase/Write	F5	35	5
Copy	F7	37	7
Erase All Unprotected	6F	3F	?
	IPC Only Commands		
Init Terminal	-	38	8
Release Terminal	-	39	9
IPC Off	-	3A	:
Disconnect	-	3B	;

We describe synchronous-line and IPC commands in this chapter. We describe IPC only commands in Chapter 5.

Read Commands

As Table 3-2 shows, there are two types of Read commands: Read Buffer and Read Modified. When RCX70 executes a Read Buffer command, the entire contents of the specified screen are sent to the host application. You should use the Read Buffer command primarily for diagnostic purposes. The results of a Read Modified command are contingent upon specific instructions from a running application or from operator commands. The data read from a Read Modified command can include:

- Fields of keyboard-modified data,
- Program function key codes, and/or
- Program access key codes.

In remote configurations, RCX70 will normally execute a Read through a general or specific poll sequence. (See Chapter 4, "Direct 3271 Emulation," for a discussion of general and specific polls.) When an operator hits an ENTER, PA, or PF key in a program running under a local (IPC) configuration, RCX70 executes a Read Modified command.

An application does not need to issue Read commands. It is more efficient to wait for the keyboard user to hit an ENTER, PA, or PF key. Then the data will be picked up by the next poll or sent immediately by the IPC. If your application does issue a Read command, it may receive the data before the keyboard user was ready to send it.

Read Buffer Command

NOTE: If you have an RCX70 system configured with a synchronous line, see Chapter 4 for Read Buffer command syntax and code. If your RCX70 system is configured for IPCs, see Chapter 5 for syntax and code.

You can use the Read Buffer command to double check the results of a previous Write command. A Read Buffer transfers all data including nulls to the host application. The data it transfers begins at a specified address in a buffer and continues to the end of that buffer. RCX70 determines the starting address as follows:

- If the Read Buffer command follows a Select sequence, a Copy command, an Erase All Unprotected command, or another Read command, (without another intervening command) the starting address of the read is location 0.

- If the Read Buffer command follows a Write or Erase/Write command, the read starts at the current buffer address and ends when RCX70 transfers the contents of the entire buffer. The read always starts at location 0 for IPC configurations.

The data stream transferred to the host application is in the format in Figure 3-1.

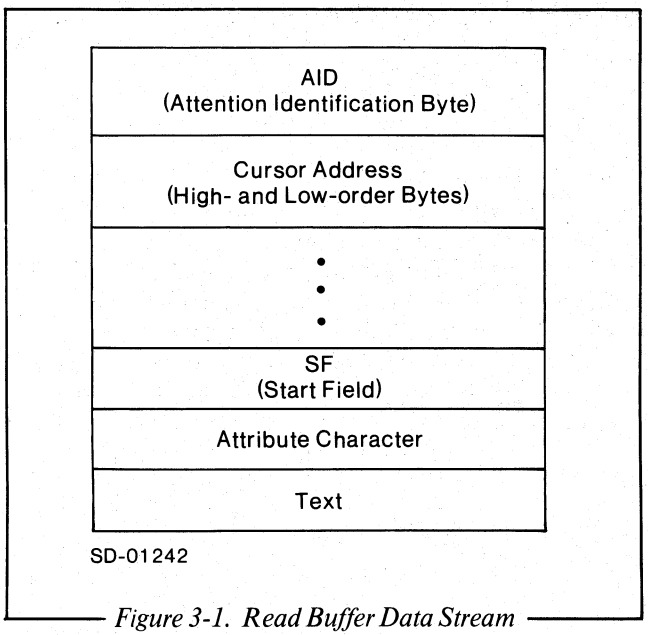


Figure 3-1. Read Buffer Data Stream

The AID byte describes the program attention key (program function or program access) that the user last hit. Table 3-3 describes all the possible AID byte values. The AID byte is the first character of a three-character Read header.

The high- and low-order bytes that make up the cursor address complete the Read Header sequence. The cursor address indicates the position of the cursor when RCX70 executes the command. It is not necessarily the same as the starting buffer address.

Execution of the Read Buffer command has no effect on the cursor location. Cursor addresses start with position 0 at the top, left corner of your screen. Position 79 is at the top, right corner and position 80 is at the left edge of the second row. See Appendix B for a list of cursor address byte configurations.

When transferring buffer data, RCX70 inserts Start Field (SF) order codes before each attribute character. The SF byte identifies the beginning of each field. RCX70 inserts the SF after the Read header only if there is an attribute character in the first transferred location. If there is no attribute character, the stream will start with the actual text. If there is an attribute character, the actual buffer text will follow the SF and attribute bytes.

Read Modified Function

The Read Modified sequence is the primary way that a host receives data from a screen. RCX70 executes a Read Modified sequence when it receives the Read Modified command from the synchronous line or IPC interface, or when a user strikes an ENTER, PA, or PF key. If the sequence was initiated by a user hitting an ENTER, PA or PF key, and if RCX70 is communicating with the synchronous line, RCX70 initiates the sequence on the next poll. If RCX70 is communicating with the IPC interface, RCX70 initiates the sequence immediately. Since RCX70 initiates a Read Modified sequence when the user hits an ENTER, PA or PF key, the application does not usually have to issue the command.

The Read Modified sequence transfers all fields which the user modified from the screen buffer to the application. Unlike Read Buffer, Read Modified does not transfer nulls or unmodified fields. In short, Read Modified provides a way for the application to determine what the keyboard user has typed. The application can ignore extraneous screen data and the nulls that fill the field area where the user has not typed.

The Read Modified sequence directs RCX70 to execute one of three Read sequences, depending upon which program attention key a user strikes:

1. Full Read Modified sequence
2. Short Read sequence
3. Test Request Read sequence

If the user has not hit a key and the sequence is initiated by a Read Modified command, RCX70 executes the Full Read Modified Sequence.

Table 3-3 and the following text list all the program attention keys that a user might strike and the resulting Read Modified command sequence that RCX70 initiates.

Full Read Modified Sequence

When a keyboard user strikes the ENTER key or any of the program function keys (PF 1 through PF 12), RCX70 initiates a Full Read Modified sequence. If RCX70 receives a Read Modified command but a user has not struck a program attention key, RCX70 will still initiate the Full Read Modified sequence. This is the first case listed in Table 3-3; RCX70 will generate a 60 EBCDIC or a 2D ASCII on unsolicited Reads from the host application.

Table 3-3. AID Byte Configurations

AID	Hex Character (EBCDIC)	Hex Character (ASCII)	Graphic Character	Commands Executed by Read Modified in a General Poll	Data Transfer from RCX70
No AID generated (display or display station)	60	2D	-	Rd Modified	Remote Poll- No Read operation occurs; otherwise RCX70 transfers field addresses and all modified fields.
No AID generated (printer)	E8	59	Y	Rd Modified	
ENTER key	7D	27	'	Rd Modified	RCX70 transfers the AID code and cursor address, followed by an SBA order, an attribute address plus 1, and the text in each modified field. RCX70 suppresses nulls.
PF 1 key	F1	31	1	Rd Modified	
PF 2 key	F2	32	2	Rd Modified	
PF 3 key	F3	33	3	Rd Modified	
PF 4 key	F4	34	4	Rd Modified	
PF 5 key	F5	35	5	Rd Modified	
PF 6 key	F6	36	6	Rd Modified	
PF 7 key	F7	37	7	Rd Modified	
PF 8 key	F8	38	8	Rd Modified	
PF 9 key	F9	39	9	Rd Modified	
PF 10 key	7A	3A	:	Rd Modified	
PF 11 key	7B	23	#	Rd Modified	
PF 12 key	7C	40	@	Rd Modified	
PA 1 key	6C	25	%	Short Rd	AID code only.
PA 2 (CNCL) key	6E	3E	>	Short Rd	
PA 3 key	6B	2C	,	Short Rd	
CLEAR key	6D	5F	-	Short Rd	
TEST REQ key	F0	30	0	Tst Req Rd	A test request message. AID tranferred on Read Buffer only.

In a Full Read Modified sequence, RCX70 transfers all keyboard-modified fields from the screen buffer to the host application. RCX70 determines whether a field has been modified by checking the Modified Data Tag (MDT) bit. The MDT is located in the attribute byte of every field. If the MDT bit is set to 0, the field has not been modified. If the MDT bit is set to 1, either the field has been modified or the MDT was initially set by the application.

When RCX70 performs a Full Read Modified sequence, it examines each attribute byte. If the MDT is set to 0, RCX70 skips that field and jumps to the next attribute byte. If the MDT is set to 1, RCX70 transfers the data in that field (suppressing nulls), then jumps to the next attribute byte.

Data Stream Transfer - In a Read Modified sequence, the data stream returned to the host application is in the format in Figure 3-2.

As with the Read Buffer command, the AID byte and the high- and low-order cursor address bytes make up the Read header. These bytes are always the first three characters of the Read Modified data stream. Again, Table 3-3 lists the AID byte values and Appendix B lists cursor address byte configurations. The modified field data follows the Read header. A Set Buffer Address (SBA) order (generated by RCX70) followed by the address of the field's first data byte (attribute address + 1) precedes the actual alphanumeric data. RCX70 does not return the attribute byte itself to the application. The application can determine what the attribute byte must be because it wrote the attribute byte to begin with. The keyboard user cannot change any attribute bits except the MDT.

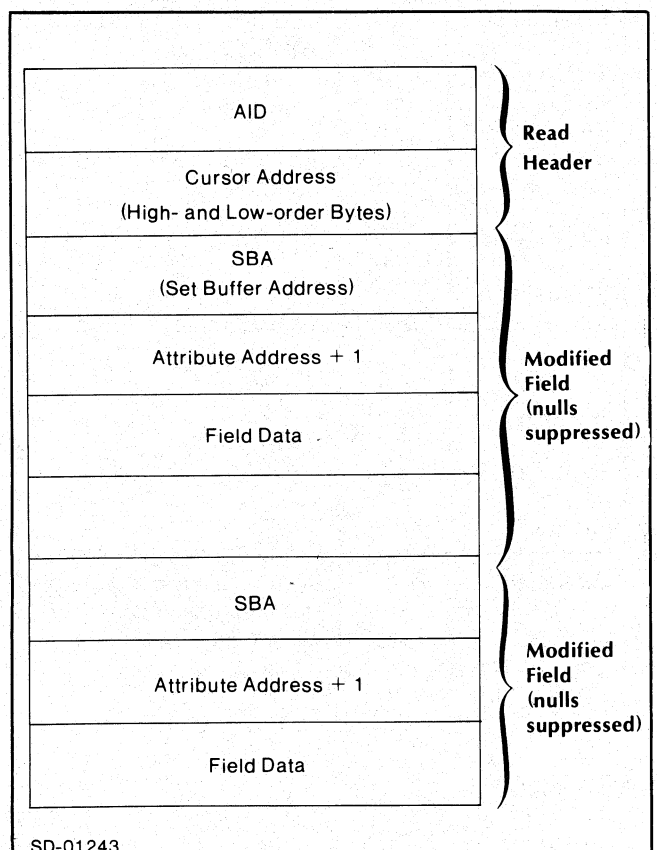


Figure 3-2. Full Read Modified Sequence Data Stream

Command Start Address - RCX70 defines the command start address as follows:

1. If a Read Modified command follows a Select sequence, a Read Buffer, or a Copy, (without another intervening command) the search starts at location 0.

2. If the keyboard user hits an ENTER, PA, or PF key to initiate the command, the search starts at location 0.

3. If a Read Modified command follows a Write, an Erase/Write, or another Read Modified, (without another intervening command) the search starts at the current buffer address.

Command Termination - RCX70 stops searching for modified fields after it checks the last buffer location. If the last modified field in the screen buffer wraps around to the beginning of the buffer, RCX70 stops searching for modified fields before the first attribute byte following the wrapped field. RCX70 then sets the buffer address to the attribute byte of the next field. Figure 3-3 illustrates command termination in a wrapped field.

If the buffer field is not wrapped, the Read Modified sequence stops when RCX70 transfers the last modified field. RCX70 then sets the buffer address to 0.

Formatted and Unformatted Buffers - As stated in Chapter 2, a formatted buffer is one that contains fields; an unformatted buffer has no fields. If a formatted buffer has not been modified (all MDT bits are 0), a Read Modified sends only the 3-character Read header (AID plus high- and low-order cursor address). For an unformatted buffer, RCX70 sends a data stream containing the Read header and all the data in the buffer (except nulls). RCX70 sends the data even if it hasn't been modified. Since the buffer is unformatted, there are no attribute bytes, no SBA orders, and no buffer addresses sent with the data stream.

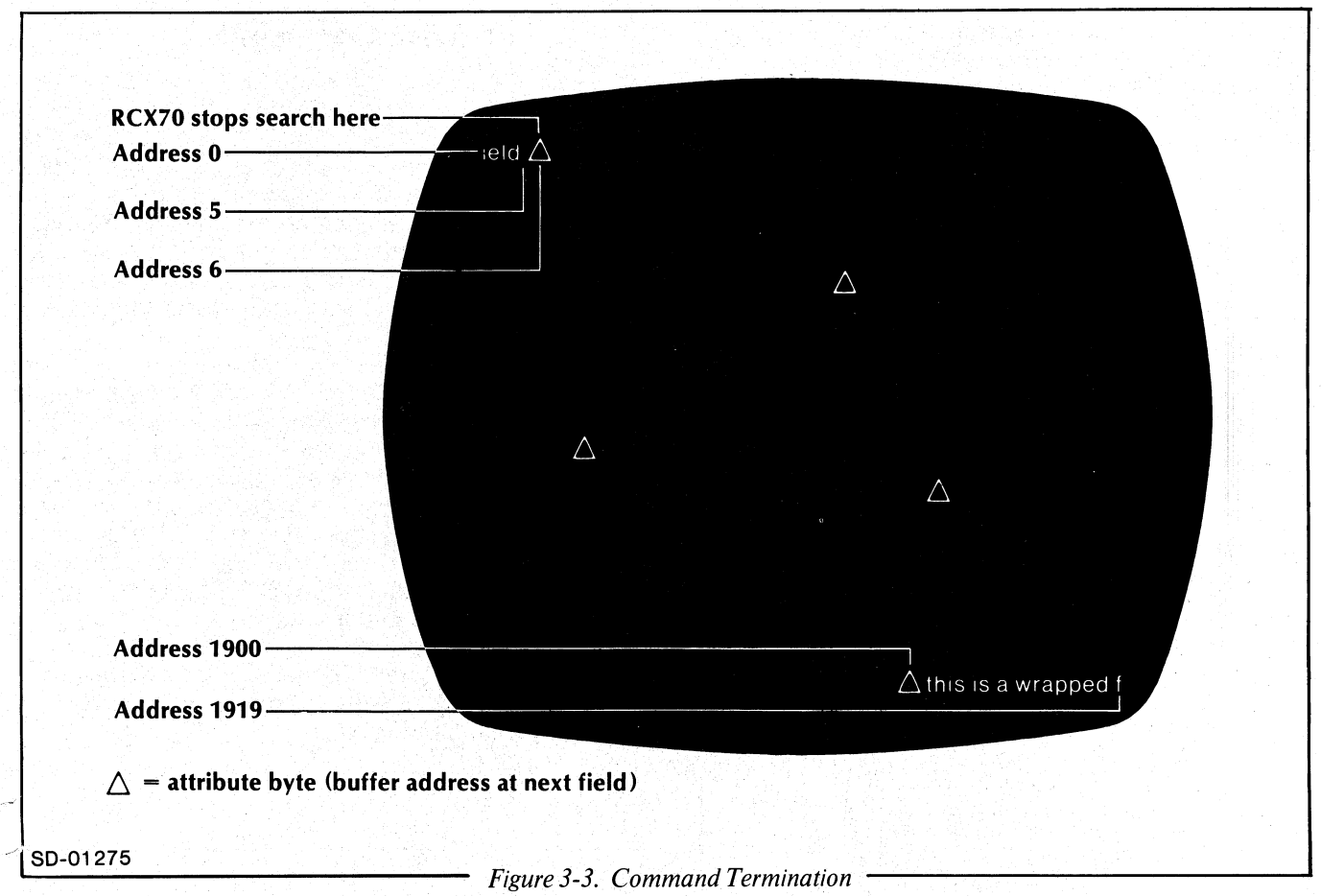


Figure 3-3. Command Termination

NOTE: If a buffer contains no attribute bytes, RCX70 will create an unprotected, alphanumeric display and place its attribute byte in the lower, right corner of the screen. However, RCX70 will respond to Read commands as if there were no attribute byte.

If the data was part of an unformatted buffer, you cannot determine exactly where a piece of that data came from or which pieces of it were modified. Transfer of an unformatted buffer in a Read Modified sequence always starts at location 0 and RCX70 always transfers the entire buffer. After command completion, RCX70 resets the buffer address to 0.

Short Read Sequence

When a keyboard user strikes PA1, CANCEL (PA2), PA3, or Clear (ERASE PAGE), RCX70 initiates a Short Read sequence. In a Short Read sequence, RCX70 returns only the AID byte to the host application; it returns no cursor address information or screen data. In other words, RCX70 only returns information about the key the user hit.

Test Request Read Sequence

If a keyboard operator presses the TEST REQUEST key before a Read Modified sequence, RCX70 executes a Test Request Read sequence. A Test Request Read returns the Test Request Read header (shown in Figure 3-4) followed by all the screen data. The screen data here is exactly the same as the screen data returned for a Read Modified sequence except that the first three bytes (AID and cursor address) are not returned.

In a formatted buffer, RCX70 examines all MDT bits. If all MDT bits are set to 0, RCX70 sends only the Test Request Read header. If RCX70 finds any fields with MDT bits set to 1, it sends all data in those fields (except nulls) to the host application. RCX70

determines where MDT searches begin and where the transfer ends in the same fashion as in the Read Modified sequence. In an unformatted buffer (no attribute bytes), RCX70 starts at address 0 and sends all data except nulls to the application.

NOTE: A Read Modified command does not reset the keyboard. That is, if another Read Modified is sent without an intervening Write, RCX70 will return the same data and the same AID byte. You can avoid this predicament by following the Read Modified command with a Write command.

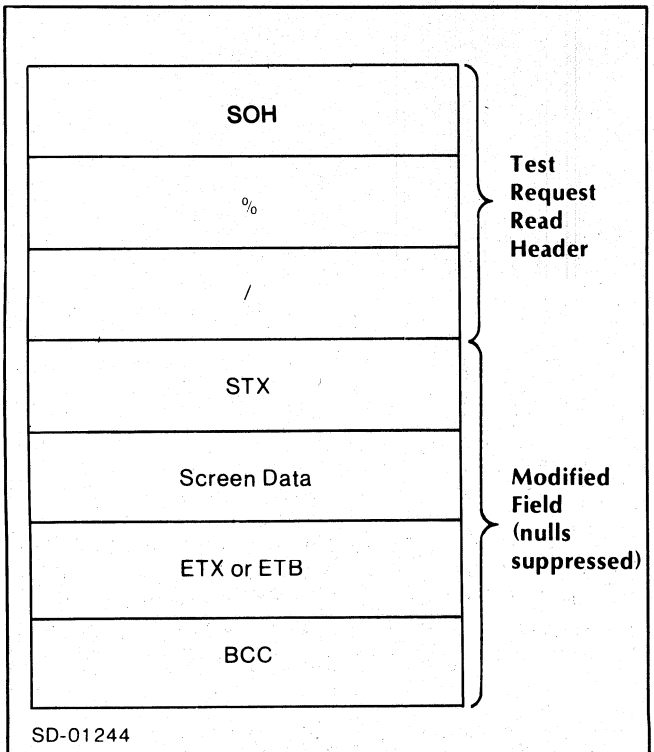


Figure 3-4. Test Request Read Data Stream

Write Commands

As Table 3-2 shows, there are two types of Write commands: Write and Erase/Write. A Write command tells the host application to write data to a selected terminal or printer as directed by specific orders. Erase/Write is identical to Write except that RCX70 erases the entire screen before any writing begins. When RCX70 erases the screen, it inserts nulls into all buffer locations in the device buffer, repositions the cursor to character location 0, and resets the buffer address to 0.

You can include buffer orders and print orders in Write data streams so that the application can write particular parts of the screen without rewriting the entire screen. The Write and Erase/Write data blocks differ depending on whether your RCX70 system is configured for a synchronous line or IPCs. If you are operating in direct 3271 emulation mode, see Chapter 4 for more details. If you are in terminal handling mode, see Chapter 5 for more information.

Figure 3-5 illustrates the Write data block that the host sends to RCX70. Shaded areas represent data sent from remote configurations only.

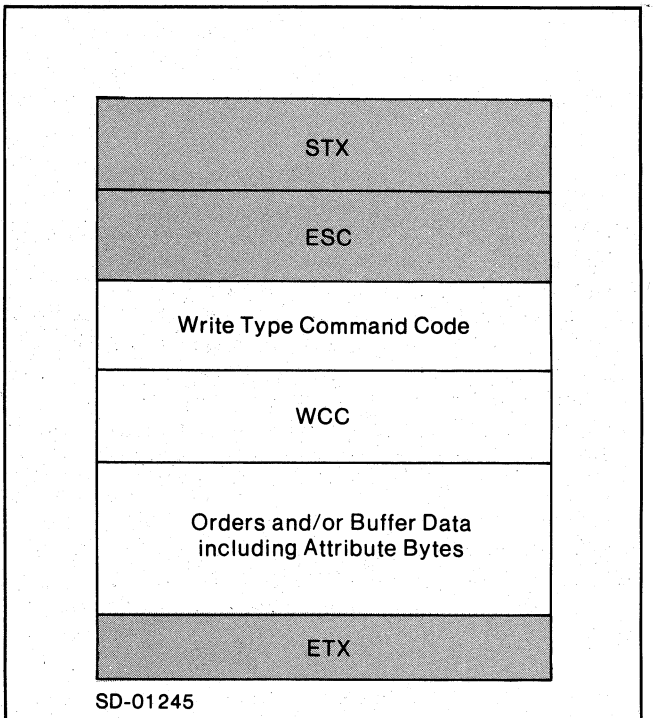


Figure 3-5. Write, Erase/Write Data Block

The data block format which a Write command sends to RCX70 consists of one of the Write codes in Table 3-2, a Write Control Character (WCC), and the specific orders and/or buffer data that RCX70 will need to modify an existing buffer. In synchronous-line modes, the data block format also includes Start of Text (STX), Escape (ESC) and End of Text (ETX) characters. In IPC mode, the format always includes a Set Buffer Address (SBA) order immediately following the WCC and addressing information (two bytes following the SBA). You don't have to include an SBA sequence after the WCC in synchronous-line mode, but it is a good idea to include one. If you always include an SBA sequence, you will not accidentally start a command at the wrong address.

WCC Byte

A WCC (Write Control Character) byte should always follow the Write type command code byte. If you don't follow the Write command with text, RCX70 will assume an all zero WCC. If the WCC specifies an action that the device cannot perform, it will not signal an error. Table 3-4 illustrates the WCC byte format and documents the function of each bit. Before transmission, the host must convert the 6-bit code to a graphic character according to Table 3-1.

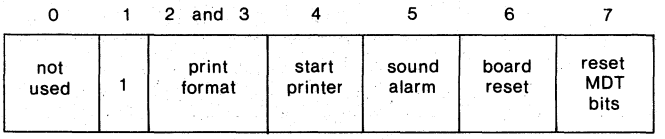
Orders and Buffer Data

Orders and buffer data are intermixed in the bytes of the data block following the WCC. *Buffer orders* describe where to put data and occasionally perform control functions. RCX70 executes buffer orders as soon as it receives them from the application. *Print orders* act as controls for the printer. RCX70 stores print orders in the buffer and executes them only during a print operation. An application can send a format with print orders to a terminal, but the orders will have no effect and will appear as blanks. Table 3-5 lists all buffer and print orders and their corresponding codes.

Buffer Orders

Start Field Order - The SF order informs RCX70 that the next byte in the data block is a field attribute byte. If you include this order, RCX70 stores the attribute byte at the current buffer position. Then, it sets a control bit at that address. The control bit marks that location as an attribute byte for future operations. The SF order does not take up a position in the buffer, but the attribute byte does. For more information on attribute byte formats, see "Screen Attribute Bytes" in this chapter.

Table 3-4. WCC Byte Format and Contents



0	1	2 and 3	4	5	6	7
not used	1	print format	start printer	sound alarm	board reset	reset MDT bits

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2, 3	Define Printout Format. (Only applicable if sent to printer.) **If** **Then** 00 New Lines in data stream set line length. RCX70 automatically generates a New Line to wrap a line around from right edge of printer. 01 40-character print line. 10 64-character print line. 11 80-character print line.
4	Start Printer Bit. If bit set to 1, RCX70 starts printer after transferring all block data. If bit not set, RCX70 ignores printout format from bits 2 and 3.
5	Sound Alarm Bit. If device has an alarm and if bit is set to 1, RCX70 sounds alarm on specified device.
6	Keyboard Reset Bit. If bit set to 1, unlocks keyboard so user can continue typing. Clears AID byte.
7	Reset MDT Bit. If bit set to 1, clears all Modified Data Tag bits back to 0.

Set Buffer Address Order - The SBA order tells RCX70 that the next two bytes are a new buffer address. RCX70 then updates the current buffer address and starts or continues all Write operations from the new address. If an SBA order precedes another buffer order (PT, RA, or EUA), RCX70 will start that order from the updated address. If the address following an SBA order is invalid, RCX70 aborts the command. See Appendix B for address codes.

Table 3-5. Buffer and Print Orders

Buffer Order	Byte 0 Order Code		Byte 1	Byte 2	Byte 3
	EBCDIC	ASCII			
Start Field (SF)	1D	1D	attribute character[1]		
Set Buffer Address (SBA)	11	11	1st address byte[2]	2nd address byte[2]	
Insert Cursor (IC)	13	13			
Program Tab (PT)	05	09			
Repeat to Address (RA)	3C	14	1st address byte[2]	2nd address byte[2]	Repeat character
Erase Unprotected to Address (EUA)	12	12	1st address byte[2]	2nd address byte[2]	
Extended Attribute (EA)	06	06	1st extended attribute byte[3]	2nd extended attribute byte[4]	
Print Order					
New Line (NL)	15	0A			
End Message (EM)	19	19			
Form Feed (FF)	0C	0C			

Footnotes:
1. See Table 3-6.
2. See Appendix B.
3. See Table 3-7.
4. See Table 3-8.

Insert Cursor Order - The IC order moves the cursor to the current buffer address. The buffer address itself does not change. Figure 3-6 illustrates the IC order.

Program Tab Order - The PT order advances the current buffer position to the first byte of the next unprotected field. If RCX70 receives a PT order when the current buffer position contains an unprotected attribute byte, RCX70 advances the buffer address one location. If the PT order does not follow a WCC or another order, RCX70 inserts nulls into all buffer locations from the current buffer address to the final buffer location. RCX70 will insert the nulls even if the field is protected. If the initial PT order was clearing fields, a subsequent PT order will also clear fields. If RCX70 does not find any unprotected fields after searching the remainder of the buffer (beginning at the current buffer address) it sets the current buffer address to position 0. If you wish to restart the search from location 0, you must issue another PT order.

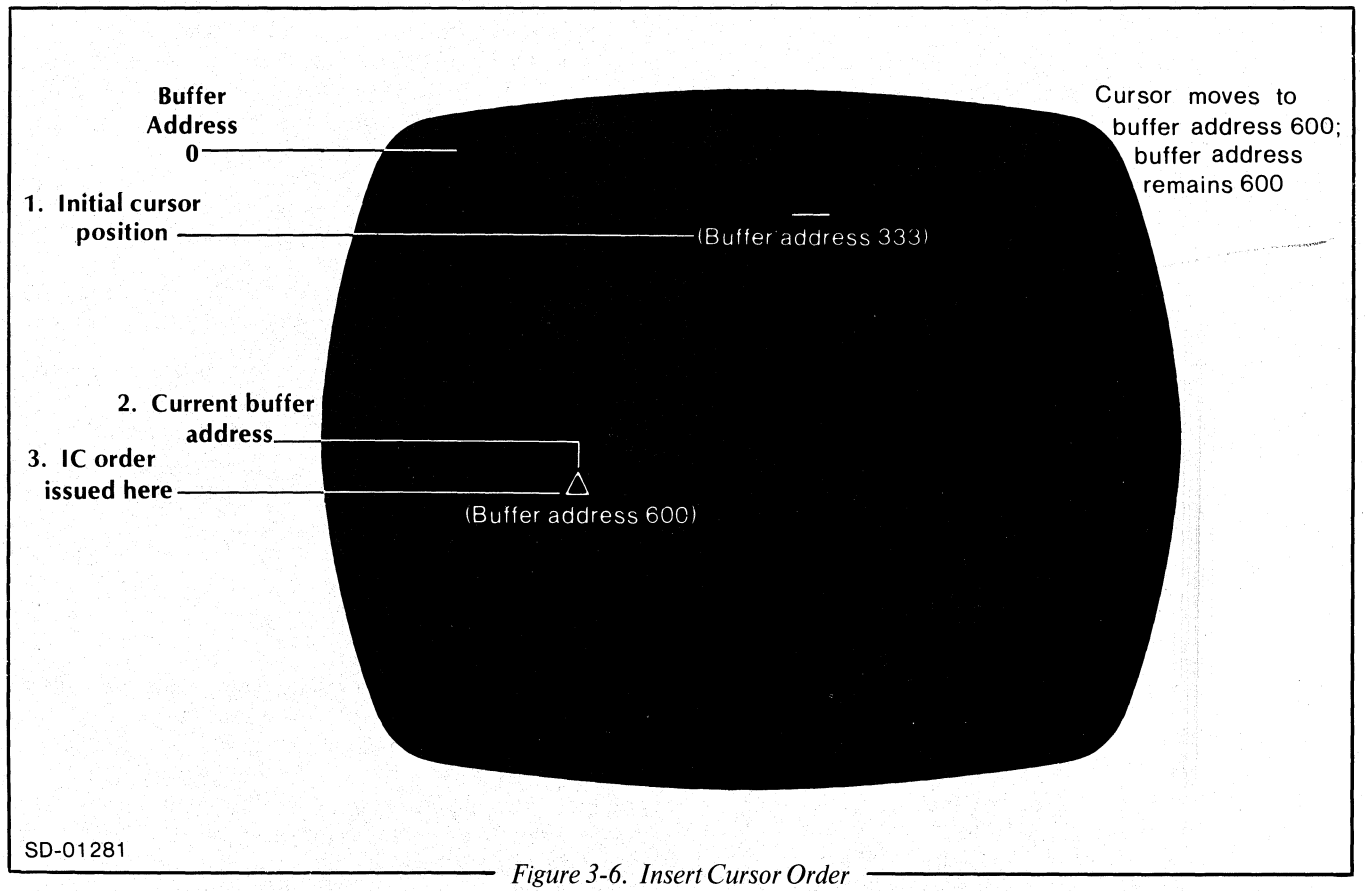


Figure 3-6. Insert Cursor Order

Repeat To Address Order - When RCX70 receives an RA order it stores a character in every buffer position up to, but not including, a specified stop address. The stop address follows the RA order in the next two bytes of the Write data block. See Appendix B to determine the code for the stop address you desire.

The character which RCX70 will repeat follows the 2-byte stop address. It can be any alphanumeric or null character. See Tables 2-1 and 2-2 for the ASCII and EBCDIC codes for the specified repeat address.

If you enter an invalid stop address, RCX70 terminates the Write operation, does not store the repeat character, and generates an error message. If you specify a stop address which is lower than the current address, the RA order wraps around the screen. If the current address and the stop address are equal, RCX70 fills the entire screen buffer with the repeat character. If the repeat character overwrites a location which previously contained an attribute byte, RCX70 will merge the current and previous fields into one large field.

Erase Unprotected To Address Order - An EUA order inserts nulls into all unprotected fields from the current buffer address up to, but not including, a specified stop address. RCX70 does not insert nulls into attribute bytes. The stop address follows the EUA order in the next two bytes of the Write data block. See Appendix B to determine the code for the stop address you desire.

If you specify an invalid stop address, RCX70 terminates the Write operation and generates an error condition. If you specify a stop address lower than the current buffer address, the EUA order wraps around the screen. If the current address and the stop address are equal, RCX70 will erase all unprotected fields in the buffer.

Extended Attribute Order - Data General provides the Extended Attribute order to define a field (for RCX70) in more detail than is possible using the IBM attribute byte. The EA order must follow an IBM-type attribute byte and the extended attribute bytes must follow the EA order. Hence, the data block sent to RCX70 would be ordered as in Figure 3-7.

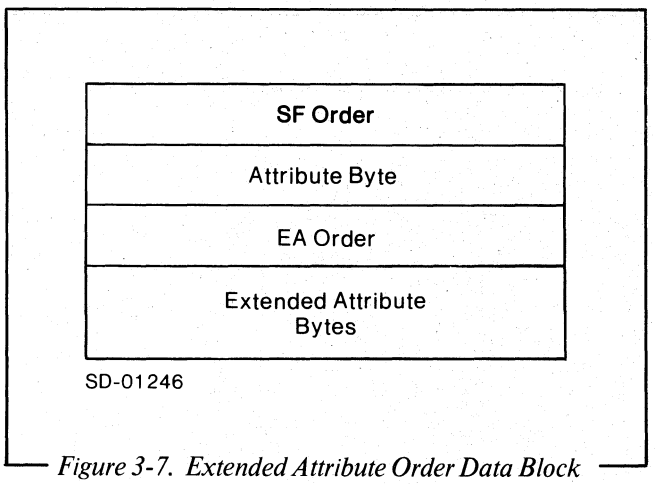


Figure 3-7. Extended Attribute Order Data Block

See "Extended Attributes" in this chapter for more detail.

Print Orders

New Line Order - RCX70 executes the NL print order only if NL is part of a print field with no specified line length. (You can set line lengths in the WCC.) RCX70 does not execute the NL order if it appears in a field that will not be printed. (See "Formatted and Unformatted Screens" in Chapter 2 for information on attribute bytes.) Instead RCX70 prints the NL as a space. If a line length is set, RCX70 will not execute the order; instead, it will print the order as a graphic 5.

End Message Order - RCX70 executes EM orders only if EM is part of a print field with no specified line length. If a line length is set, RCX70 will not execute the order; instead, it will print the order as a graphic 9. Once RCX70 executes an EM order, the printer will stop printing. Like the NL order, RCX70 will ignore EM orders in a field that will not be printed.

Form Feed Order - When RCX70 executes a FF order, the printer will feed a page of paper. It will also print a space in the first position of the line corresponding to the buffer position which the FF occupies. RCX70 executes a FF order in a print field with or without a prespecified line length. You can include any number of FF orders in a data buffer.

There are certain restrictions on where you may place a FF order in the print buffer. The FF order can appear in any field whether it will be printed or not. A valid FF order must be in the first character position of a line, which can be one of the following locations:

- The first location in the buffer,
- The first position after a valid NL, or
- The first position of any print line (i.e., the 41st character where lines are 40 characters long).

If the FF is located in a valid position, the printer advances the paper to the next form.

The FF order is invalid in any position other than those listed above. An invalid FF order will appear on the printed page as a left angle bracket (<).

Buffer Data Addressing

RCX70 stores data characters in successive buffer locations until it encounters an order in the data stream which changes the buffer address, or until it has stored all the other data. During a Write operation, RCX70 advances the buffer address one location each time it stores a character. When communicating with the IPC interface, RCX70 must receive a Set Buffer Address (SBA) as the first bytes of text. It uses the SBA to determine the starting address for IPC data entry. When it receives a command from the synchronous-line interface, RCX70 determines the starting address for data entry by the following criterion:

1. A Set Buffer Address (SBA) order following the WCC may have defined an address.
2. If the Write command follows a Select command or if it follows a Control command (Copy, Erase All Unprotected), the starting address will be the cursor location.
3. If the Write command is chained from another Write command or from a Read command, the starting buffer address is the address at the end of the previous command.

Reader, Please Note

1. RCX70 processes the block one character at a time. It does not check the entire block for validity before processing. Therefore, RCX70 may process and transfer certain data or orders in the block while rejecting later data or orders. If this happens, RCX70 may leave the current buffer address at a point previous to where the user expected. Consequently, any following commands would write to an unintended part of the screen. To avoid this, the application should issue an SBA at the beginning of each block.

2. If an application sends a series of commands to a printer, it must include the Write or Erase/Write command with the Start Print WCC bit set as the last command in the chain. If the application fails to do this, RCX70 prints only the data sent to it thus far.

3. RCX70 carries out orders in a very literal way. If you enter an absurd set of commands or orders, e.g., write to a screen and immediately erase the screen, RCX70 will follow the orders without question or error message.

Screen Attribute Bytes

A field is a set of character positions starting at an attribute byte and ending at the last character position before the next attribute byte. Table 3-6 describes the format and functions of the screen attribute byte. Before transmission, the host must convert the 6-bit code to a graphic character according to Table 3-1.

Extended Attribute Bytes

Data General has provided RCX70 with extended attribute bytes which work in conjunction with the IBM-defined attribute byte. They provide an application using RCX70 with a more detailed way of specifying the legal contents of a field. Extended attribute bytes always follow an Extended Attribute order (see "Orders and Buffer Data" in this chapter).

Extended attribute bytes consist of two bytes which, unlike IBM-type bytes, do not occupy a screen position. The two bytes contain three bit positions to specify the legal contents of a field: alpha, numeric, or special character. You can set one or any combination of these three bits. Tables 3-7 and 3-8 illustrate the extended attribute byte formats and document the function of each bit.

Before transmission, the host must convert the 6-bit code to a graphic character according to Table 3-1.

Table 3-6. Screen Attribute Byte

0	1	2	3	4 and 5	6	7
not used	1	protect/ unpro	alphanum/ numeric	dis/nondis	reserved	MDT

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2	Protected or Unprotected. If bit set to 1, the field is protected. If the user tries to type anything into a protected field, the keyboard locks.
3	Alphanumeric or Numeric. If bit set to 1, the user can enter only numeric characters (0 through 9, minus, period). The user must strike a function key to escape numeric mode. At generation time, you can direct RCX70 to accept a comma instead of a period.
4 and 5	Display/Nondisplay. These bits will tell RCX70 whether the field will be display or nondisplay and whether the display will be underlined. **If** **Then** 00 Display the field. 01 Display the field. 10 Display the field with underlines. 11 Do not display or print the field.
6	Reserved.
7	Modified Data Tag. Set to 1 if user enters any data into the field.

Table 3-7. First Extended Attribute Byte

0	1	2	3	4	5	6	7
not used	1	alpha permitted	numeric permitted	special characters	decimal position	required	full

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2	Alpha permitted. If bit set to 1, all alphabetical characters are legal in this field.[1]
3	Numeric permitted. If bit set to 1, all numerics are legal in this field.[1]
4	Special character permitted. If bit set to 1, all special characters are permitted in this field.
5	Decimal position set. If bit set to 1, the maximum number of positions to the right of the decimal point is specified in the second extended attribute byte and a maximum of one decimal point may occur in the field.
6	Required. If bit set to 1, the field is required. The user must enter at least one character into the field before entering the screen.
7	Full. If bit set to 1, this field must contain exactly the maximum number of characters allowed, or must be completely empty.

[1] Normal alphabetic characters are A through Z (upper- and lowercase); normal numerics are 0 through 9, minus and period; normal special characters are anything other than alphas and numerics. At configuration time, you can specify any character to be other than its normal type. For example, you might substitute a comma for a period in a numeric field. You can also add additional characters; i.e., if you add an E to the numeric field, FORTRAN-type scientific notation would be legal. RCX70 encounters no problem in defining the E as both alpha and numeric.

Table 3-8. Second Extended Attribute Byte

0	1	2	3 through 7
not used	1	blink	maximum number of digits permitted to right of the decimal

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2	Blink. If bit set to 1, the field will blink on the screen. Data typed into a blink field will not blink on the screen, but data displayed as part of a format with the blink extended attribute will blink.
3 through 7	Maximum number of digits permitted to right of decimal. Any number less than or equal to the maximum number of characters allowed for the field is legal. The maximum number of digits permitted to the right of the decimal is 32.

Copy Command

A RCX70 Copy command copies part or all of a device buffer to a selected device. By using the Copy command, an application can print screen data on a line printer or refer the data to another screen. The line printer and receiving screen are *destination* devices. The screen from which the data is copied is the *sending* device. In remote configurations, a previous Select command sequence determines the destination device (see Chapter 4). In local configurations, the destination device is a user-supplied part of the Copy data stream code (see Chapter 5). Figure 3-8 illustrates the Copy data blocks for remote and local configurations.

The Sending Device Address

The sending device address consists of a single character.

CCC Byte

The Copy command data block consists of the Copy command code from Table 3-2, a Copy Control Character (CCC), and a sending device address. You must follow the command code with the CCC and the address byte. Otherwise, RCX70 will abort the command and generate an error message.

The CCC describes the type of data RCX70 will copy, starts print operations at the receiving device, defines the print format for the print operation, and sounds the alarm if the receiving device is a terminal. Table 3-9 illustrates the CCC byte format and documents the function of each bit. Before transmission, the host will convert the 6-bit code to a graphic character according to Table 3-1.

RCX70 copies the data defined in CCC bits 6 and 7 from the sending device to the destination device. RCX70 also fills any buffer positions that it doesn't copy with nulls. Upon completion of the Copy command, RCX70 positions the cursor at the position in the destination device where it was originally in the sending device. For example, assume that RCX70 executes a Copy command when the cursor is located in position 66 of the sending device. Then when it finishes the Copy, RCX70 will position the cursor in location 66 of the destination device.

You can protect the sending device from Copy by inserting a protected, alphanumeric attribute byte in position 0 of the buffer. If you try to copy from a protected device, RCX70 will clear the destination device but will copy nothing.

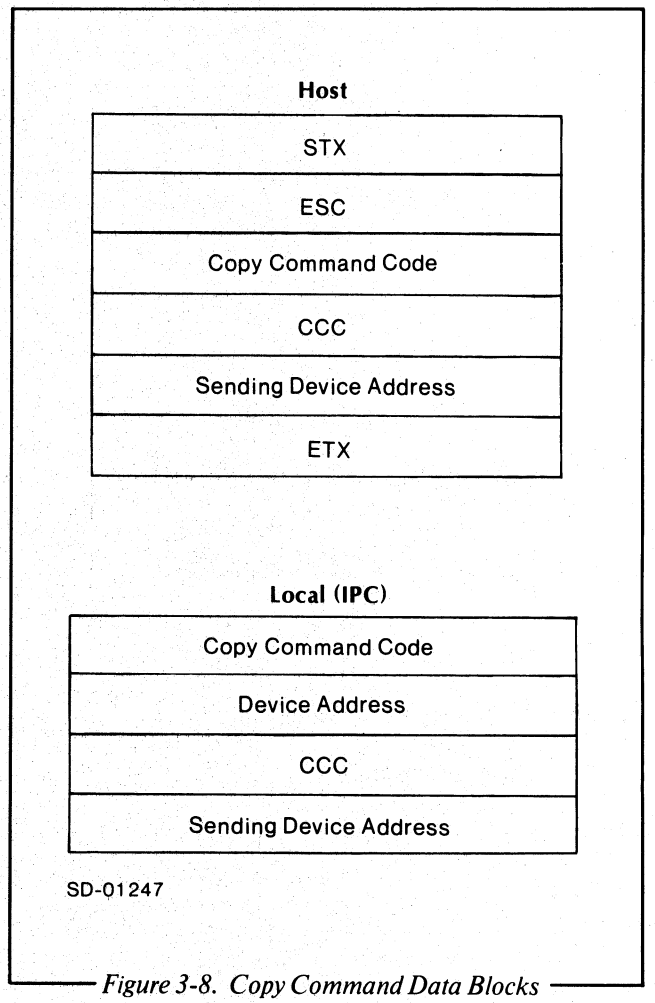


Figure 3-8. Copy Command Data Blocks

Table 3-9. CCC Byte Format and Contents

0	1	2 and 3	4	5	6 and 7
not used	1	print format	start printer	sound alarm	define data to be copied

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2 and 3	Define Print Format. (Used only if destination is a printer.) **If** **Then** 00 New Lines in data stream set line length. RCX70 automatically generates a New Line to wrap a line around from the right edge of the printer. 01 40-character print line. 10 64-character print line. 11 80-character print line.
4	Start Printer Bit. If bit set to 1, RCX70 directs the destination device to start the print operation after it receives all the data. If bit not set, RCX70 ignores the print format from bits 2 and 3.
5	Sound alarm bit. If bit set to 1, RCX70 sounds the alarm on the destination terminal.
6 and 7	Define the data to be copied. **If** **Then** 00 RCX70 copies only attribute bytes. 01 RCX70 copies attribute bytes and unprotected alphanumeric fields (including nulls). If RCX70 encounters a protected field, it inserts nulls into all protected, alphanumeric locations. 10 RCX70 copies attribute bytes and protected alphanumeric fields (including nulls). If RCX70 encounters an unprotected field, it inserts nulls into all unprotected, alphanumeric locations. 11 RCX70 copies the entire contents of the buffer (including nulls).

You can define the sending and destination devices as the same device. In doing so, RCX70 provides a way for an application to execute selective device buffer erase operations. Bits 6 and 7 of the CCC byte determine which part of the buffer RCX70 will erase. RCX70 will fill all fields outside of those specified in the CCC with nulls. For example, the screen in Figure 3-9 consists of six fields: NAME, ADDRESS, SALARY and the responses to each of these three fields.

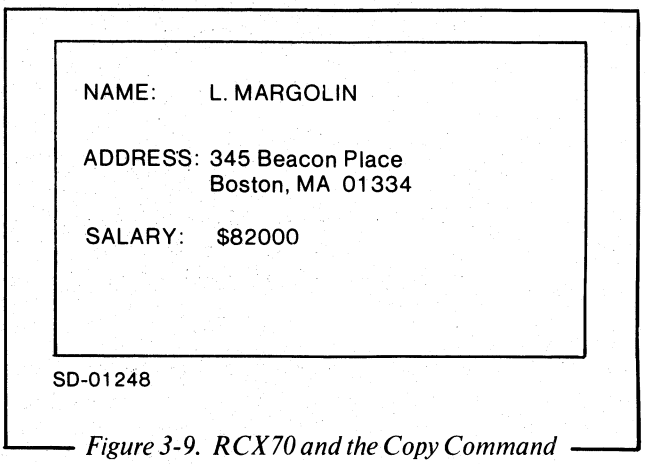


Figure 3-9. RCX70 and the Copy Command

The NAME, ADDRESS, and SALARY fields are protected alphanumerics. The responses to NAME and ADDRESS are unprotected alphanumerics and the response to SALARY is an unprotected numeric. Bits 6 and 7 of the CCC byte are set to 10 (copy attribute bytes and protected alphanumerics).

Let's assume that, as the manager of a small company, you want to access the names, addresses, and salaries of all employees, but you do not want this information to be bandied about all over the screen. Hence, you perform a selective erase operation with your terminal as the sending and destination device. The Copy command transfers the buffer's contents to RCX70. RCX70 inserts nulls into all the unprotected fields, then transfers the screen back to your terminal. (You might also have used the Erased All Unprotected command, described next, to perform the same function.)

Reader, Please Note

If the destination address is invalid RCX70 aborts the command. In remote configurations, RCX70 turns on bit 7 of sense/status byte 1.

Erase All Unprotected Command

You use the Erase All Unprotected command to clear all the unprotected buffer data on the screen. In remote configurations, the Erase All Unprotected command appears in a block which the host system sends after a Select Sequence (see Chapter 4). In local configurations, the IPC interface defines a device address in the data block.

This command performs five tasks at the selected device:

1. Clears all unprotected buffer characters to nulls.
2. Resets the MDT bit to 0 for every unprotected field.
3. Unlocks the keyboard.
4. Resets the AID byte to 60 EBCDIC or 2D ASCII. This clears the cause of the last program attention.
5. Repositions the cursor to the first character location in the buffer's first unprotected field. If there are no unprotected fields, this command positions the cursor to character location 0.

If the entire buffer is protected, RCX70 clears nothing and does not reset MDT bits. However, RCX70 still unlocks the keyboard, resets the AID byte, and repositions the cursor to character location 0.

Figure 3-10 illustrates the Erase All Unprotected data block.

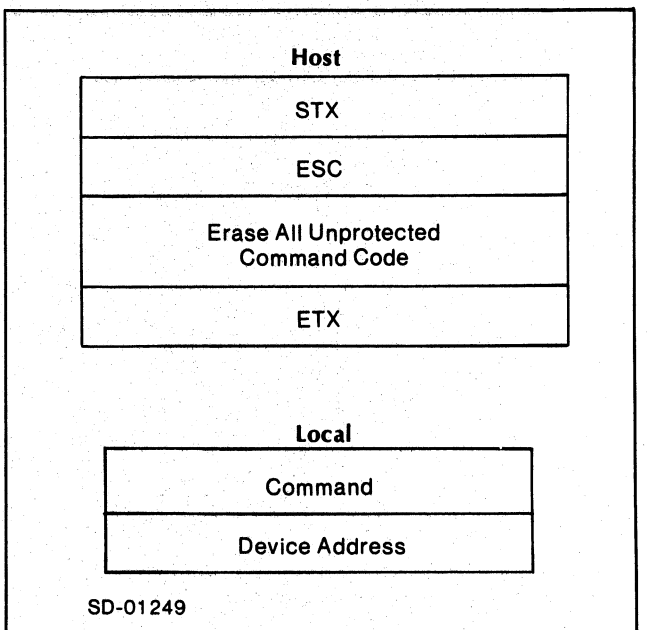


Figure 3-10. Erase All Unprotected Data Block

End of Chapter

Chapter 4
Direct 3271 Emulation

This chapter provides all the information you will need if your RCX70 system is configured for a synchronous-line interface. We discuss basic synchronous-line concepts here, but you should read the appropriate sections of the following manuals for specific BSC (Binary Synchronous Communication) protocol:

- *AOS Programmer's Manual* (093-000120)
- *Data General Communications System* (014-000070)

Synchronous Line Overview

RCX70 collects data from each attached terminal by responding to a general or specific poll. When RCX70 receives a general poll, it checks each of the terminals to see if there is any data to send to the remote system. If a keyboard user has hit any of the program attention keys (e.g., ENTER), RCX70 flags the screen data as ready to be sent. When RCX70 receives a specific poll it performs the same functions as for the general poll, but it only performs them on *one,* addressed terminal.

An application issues a Select sequence in order to send data to a screen. It sends a data block containing commands (Write, Erase/Write, Read, etc.) and screen data.

An application can communicate with all the devices attached to RCX70 by issuing a Broadcast command. Broadcast can send a message to all terminal users without selecting each terminal. In addition to those commands listed above, RCX70 can execute all the commands documented in Chapter 3 under a synchronous-line mode of operation.

Protocol

RCX70 uses BSC protocol and AOS synchronous-line system calls for all communications with the remote system. Table 4-1 lists all the data-link control characters. See Chapter 10 of the *AOS Programmer's Manual* for a complete discussion of bisynchronous protocol.

Table 4-1. Data-link Control Characters

ACK0	Even acknowledgement
ACK1	Odd acknowledgement
CRC	Cyclic Redundancy Check
DLE	Data Link Escape
ENQ	Enquiry
EOT	End of Transmission
ESC	Escape
ETB	End of Transmission Block
ETX	End of Text
ITB	End of Intermediate Transmission Block
NAK	Negative Acknowledgement
RVI	Reverse Interrupt
SOH	Start of Heading
STX	Start of Text
SYN	Synchronous Idle
TTD	Temporary Text Delay
WACK	Wait Before Transmit

Data Blocking

RCX70 sends data in blocks containing a maximum of 256 characters each. Each block has an STX as its first character and an ETB or ETX (depending on whether or not it is the last block of the series) as its last character. In other words, RCX70 sends the contents of a terminal 256 characters at a time with ETBs between blocks and an ETX as the last character of the last block.

NOTE: A Set Buffer Address order in a Write command consists of a 3-byte sequence. RCX70 will never split the sequence placing part of it in one block and part in the next block.

Buffer Address

RCX70 maintains a number called the current buffer address for each terminal and printer. Every time a terminal or printer receives a data byte, RCX70 inserts the byte at the current buffer address and advances the address one position. RCX70 can also alter the buffer address according to directives from specific commands and orders.

Code Structures

In remote configurations, RCX70 can send or receive code in either EBCDIC or ASCII. The choice of code depends on the application but the application must choose the same code for all units on a particular communications line. Tables 2-1 and 2-2 list the EBCDIC and ASCII codes.

Command Chaining

You can chain commands in a remote RCX70 configuration. Once an application selects or polls an RCX70 system, that RCX70 will remain selected until the remote system or RCX70 writes an EOT. The application can send many command blocks to the same selected device without reselecting that device. Likewise, the application can reply to a block of screen data which RCX70 sent in response to a general or specific poll. The reply will always be a data block which contains a command. RCX70 applies the command to the specifically polled terminal or to the device which last sent data in response to a poll sequence.

General Poll Command

When RCX70 receives a General Poll command from the host application, it checks each attached device. If a device has the device-end status pending, RCX70 sends that status to the remote system. If a device has one of the other status bits or no status, RCX70 executes a Read command and sends the screen data over the synchronous line to the remote host. The type of Read RCX70 executes depends on which key the user hit to prepare the screen data for transfer. RCX70 will send a Test Request Read if the user hit a TEST REQUEST key. RCX70 will execute a Read Modified command in response to the ENTER key or one of the program function keys (PF1-PF12). RCX70 will execute a Short Read if the user struck PA1, PA2 (CANCEL), PA3, or CLEAR. RCX70 won't send anything if the user has not hit a program attention key since the last poll.

RCX70 will continue to send data from each terminal (that has data to send) until it either runs out of data or is stopped by the application. The application can stop RCX70 by responding with a command block instead of an ACK. In this case, RCX70 will automatically direct the command to the device that sent the last block to the application. The application can also stop RCX70 by sending an RVI or an EOT.

If no data is available from a specified device, RCX70 will go on to check the next terminal. It does not matter in which order RCX70 checks the terminals. It will check all terminals for each general poll it receives.

RCX70 receives a General Poll command from the host application in the byte sequence documented in Figure 4-1.

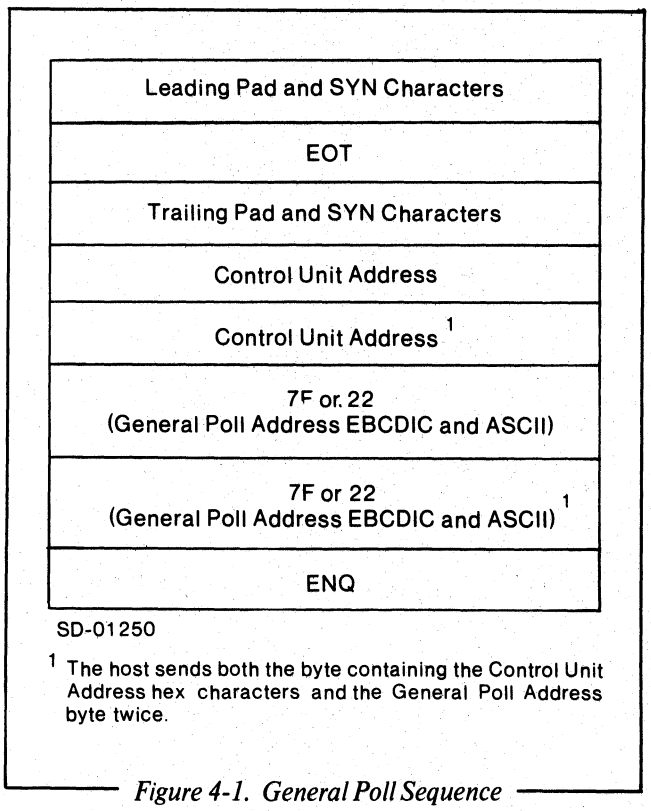
For a discussion of the data block format which RCX70 returns for each of the possible Read commands, see "Data Block Formats Returned By RCX70" in this chapter. For a detailed description of each Read command, see Chapter 3.

Control Unit Address Byte

Table 4-2 lists the control units and their corresponding EBCDIC and ASCII addresses.

General Poll Address Byte

When using EBCDIC, the General Poll address is always 7F; when using ASCII, it is always 22.



[1] The host sends both the byte containing the Control Unit Address hex characters and the General Poll Address byte twice.

Figure 4-1. General Poll Sequence

Table 4-2. Control Unit Addressing for General Poll

Control Unit Number	EBCDIC I/O Character	EBCDIC Hex	ASCII I/● Character	ASCII Hex
0	□	40	□	20
1	A	C1	A	41
2	B	C2	B	42
3	C	C3	C	43
4	D	C4	D	44
5	E	C5	E	45
6	F	C6	F	46
7	G	C7	G	47
8	H	C8	H	48
9	I	C9	I	49
10	¢	4A	[	5B
11	.	4B	.	2E
12	<	4C	<	3C
13	(	4D	(	28
14	+	4E	+	2B
15	|	4F	|or !	21
16	&	50	&	26
17	J	D1	J	4A
18	K	D2	K	4B
19	L	D3	L	4C
20	M	D4	M	4D
21	N	D5	N	4E
22	O	D6	O	4F
23	P	D7	P	50
24	Q	D8	Q	51
25	R	D9	R	52
26	!	5A	]	5D
27	$	5B	$	24
28	*	5C	*	2A
29	)	5D	)	29
30	;	5E	;	3B
31	¬	5F	¬ or ^	5E

Data Block Formats Returned By RCX70

Status Message - When RCX70 detects a terminal with sense/status pending, it sends the Status message shown in Figure 4-2 to the host. Note that the Status message contains a 2-byte block consisting of sense/status information. Tables 4-3 and 4-4 describe the bit functions for each sense/status byte. Before transmission, RCX70 converts the sense/status byte to a graphic character according to Table 3-1.

Test Request Message - If RCX70 does not detect sense/status pending, it will perform a Read command. If the user hit a TEST REQUEST key, RCX70 will send a Test Request message. See Chapter 3 (Figure 3-4) for an illustration of the Test Request message. RCX70 will also send an ETX or ETB as described under "Data Blocking" in this chapter.

Read Modified or Short Read Sequence - If a terminal has no sense/status pending and if the user hit any program attention key other than TEST REQUEST since the last poll, RCX70 will execute a Read Modified or Short Read command. RCX70 executes Read Modified in response to the ENTER key or any of the program function keys. It executes a Short Read in response to any of the program access keys. A Short Read returns only the information about which key the user hit. Figure 4-3 shows the format for the data returned from a Read command after a general poll.

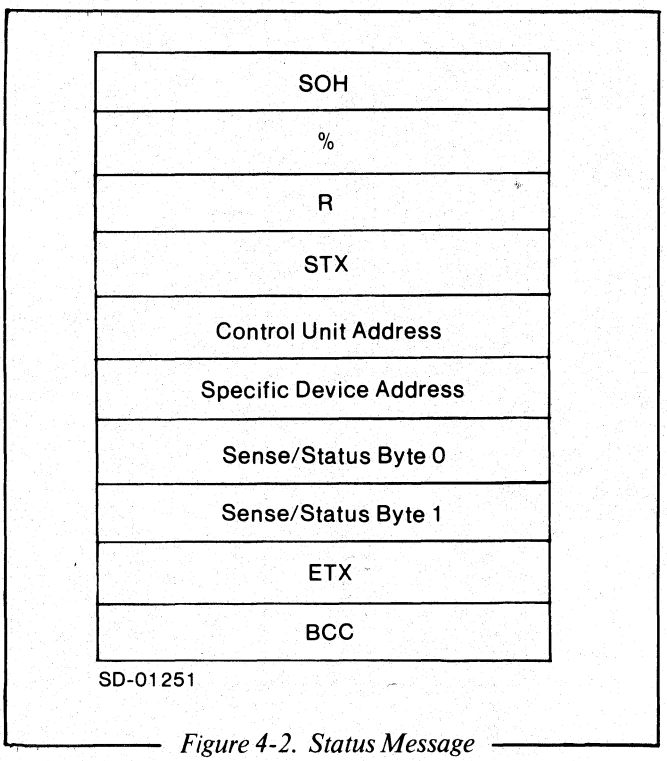
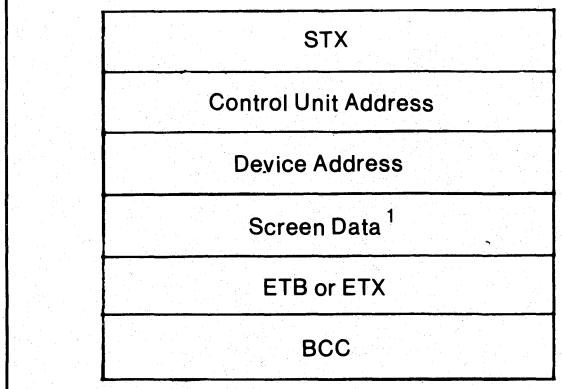


Figure 4-2. Status Message

Table 4-3. Bit Configuration for First Sense/Status Byte

0	1	2	3	4	5	6	7
not used	1	reserved	reserved	IPC file filled	device error	available	not used

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2	Reserved.
3	Reserved.
4	Set to 1 if device's IPC file is filled.
5	Set to 1 as a result of a device error.
6	Set to 1 if device has become available.
7	Not used by RCX70.



[1] See Chapter 3 for the layout of the Read command screen data (Figure 3-2) and a complete description of Read commands.

Figure 4-3. Read Modified or Short Read Command Data Block

Table 4-4. Bit Configuration for Second Sense/Status Byte

0	1	2	3	4	5	6	7
not used	1	command invalid	screen or printer unavailable	not used	not used	not used	illegal Write command

Bit	Function
0	Not used by RCX70.
1	Always a 1 for EBCDIC lines.
2	Set to 1 if RCX70 has received an invalid command.
3	If bit set to 1, RCX70 has received a command for a screen or printer which is unavailable.
4	Not used by RCX70.
5	Not used by RCX70.
6	Not used by RCX70.
7	If bit set to 1, at least one of the following conditions applies: • On a Write command, RCX70 received an illegal buffer address or incomplete order. • No CCC or sending address on Copy command. • Invalid command sequence.

General Poll -- Summing Up

Figure 4-4 shows how RCX70 receives and responds to a general poll in a multidrop synchronous-line mode of operation. In this example we find an ECLIPSE with an RCX70 system connected via a synchronous-line to a host. Other RCX70s or IBM 3270s can be connected to the multidrop line. When the host conducts a general poll, it includes one of the addressing codes from Table 4-2 in the sequence. All the devices on the multidrop line monitor the line for their address. When RCX70 recognizes its own Control Unit Address it responds by checking each of its attached terminals for data to return to the host.

Reader, Please Note

1. RCX70 will not respond to a poll:
   - if RCX70 is not up. (AOS will not respond to a poll if RCX70 is not up.)
   - if the Poll sequence is incorrect; e.g., including characters out of order or missing characters.
   - if RCX70 or another control unit is still selected; that is, no EOT has been received.
2. If the host detects a BCC error, it will respond by sending a NAK to AOS. AOS will retry the transmission up to eight times. If the BCC is still incorrect, the host system will respond with an EOT. RCX70 will try to send the same data the next time it receives a poll.
3. AOS handles all data retransmission.
4. The host system sends an RVI if it wants the control unit to stop sending data. The control unit responds with an EOT, then stops transmitting data.

093-000172-00

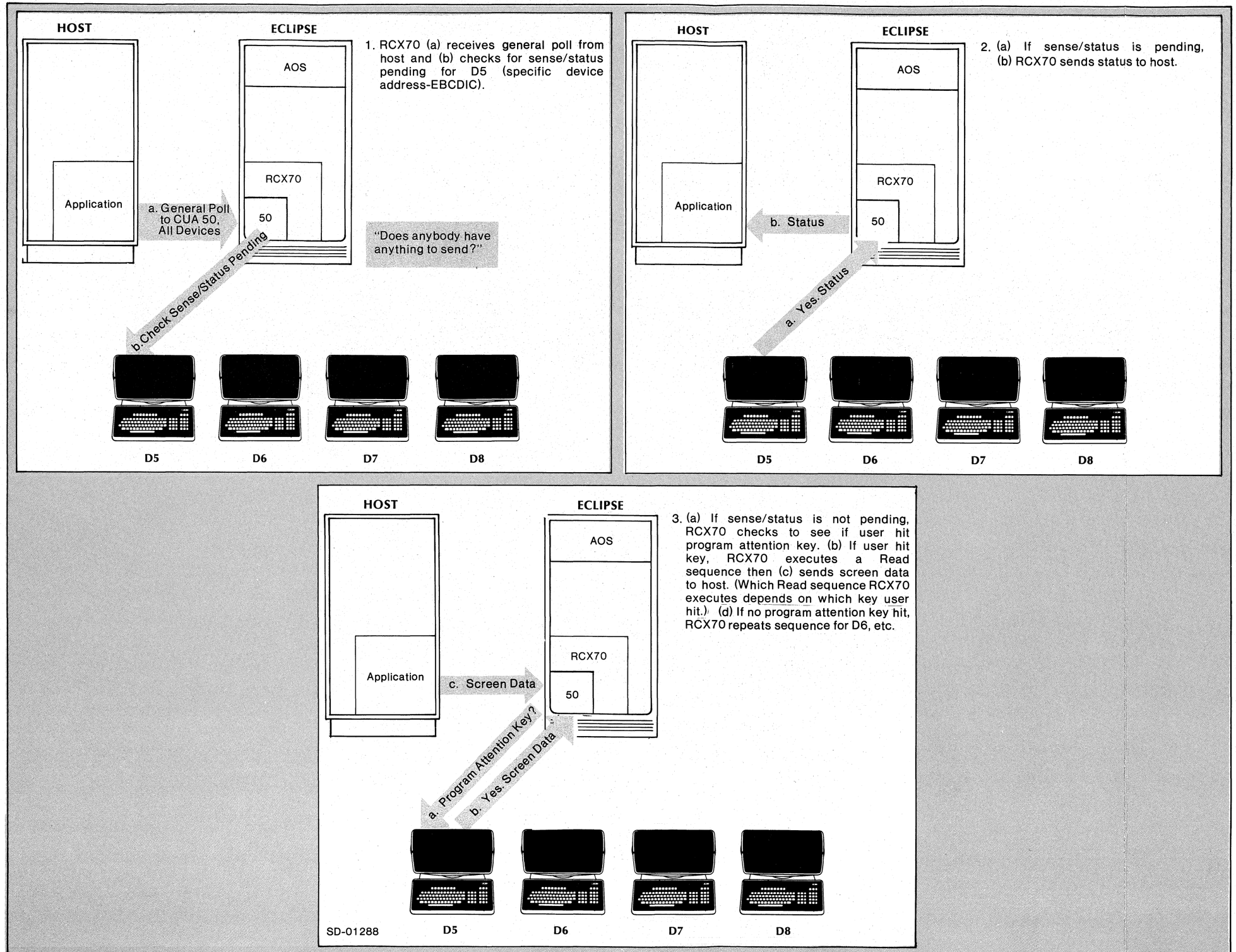


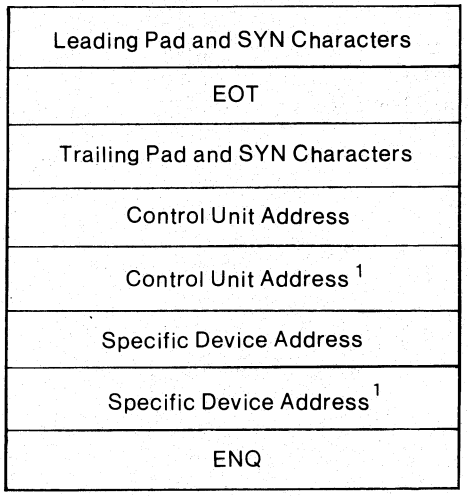
Figure 4-4. RCX70 and the General Poll

Specific Poll Command

The Specific Poll command is identical to the General Poll command except that the Poll sequence includes a specific device address instead of a general device address. By conducting a specific poll, an application can immediately obtain data from a specific terminal or printer. With a general poll, RCX70 might have to service several other terminals before reaching the desired one.

In a specific poll, when RCX70 recognizes its own Control Unit Address it checks the status of the terminal or printer corresponding to the specific device address. If the device has pending status (device unavailable, device end, or error status) RCX70 sends that status to the remote system. If status is not pending and if a user struck one of the program attention keys since the last general or specific poll, RCX70 executes a Read command and sends the screen data to the remote host. RCX70 will send a Test Request Read if the user hit a TEST REQUEST key. RCX70 will execute a Read Modified command in response to ENTER or one of the program function keys (PF1-PF12). RCX70 executes a Short Read if the user struck PA1, PA2 (CANCEL), PA3, or CLEAR. If the user did not strike a program attention key on the specified device, RCX70 responds with an EOT.

RCX70 receives a specific poll from the host in the byte sequence illustrated in Figure 4-5.



[1] The host sends both the byte containing the Control Unit Address characters and the specific device address byte twice.

Figure 4-5. Specific Poll Sequence

Control Unit Address Byte

The Control Unit Address for a Specific Poll command is the same as for a General Poll command. See Table 4-2 to determine control unit addressing.

Specific Device Address Byte

Table 4-5 lists the device numbers and the corresponding EBCDIC and ASCII device address for each unit.

Data Block Formats Returned By RCX70

The Status message, sense/status bytes, Test Request message, and Read command messages are the same for the Specific Poll command as those described for the General Poll command. See the previous section of this chapter for all the information you need.

Specific Poll -- Summing Up

Figure 4-6 illustrates how RCX70 receives and responds to a specific poll. When the host issues a specific poll, it includes a Control Unit Address (CUA) and a specific device address in the sequence. When RCX70 recognizes its own CUA, it checks the status of the device specified in the sequence and executes the appropriate command.

Reader, Please Note

1. RCX70 will not respond to a poll:
   - if RCX70 is not up.
   - if the Poll sequence is incorrect; e.g., including characters out of order or missing characters.
   - if RCX70 or another control unit is still selected; that is, no EOT has been received.
2. If the host detects a BCC error, it will respond by sending a NAK to AOS. AOS will retry the transmission up to eight times. If the BCC is still incorrect, the remote system will respond with an EOT. RCX70 will try to send the same data the next time it receives a poll.
3. AOS handles all data retransmission.
4. The host system sends an RVI if it wants the control unit to stop sending data. The control unit responds with an EOT and stops transmitting data.

Table 4-5. Device Addressing for Specific Poll

Device Number	EBCDIC I/O Character	EBCDIC Hex	ASCII I/O Character	ASCII Hex
0	□	40	□	20
1	A	C1	A	41
2	B	C2	B	42
3	C	C3	C	43
4	D	C4	D	44
5	E	C5	E	45
6	F	C6	F	46
7	G	C7	G	47
8	H	C8	H	48
9	I	C9	I	49
10	¢	4A	[	5B
11	.	4B	.	2E
12	<	4C	<	3C
13	(	4D	(	28
14	+	4E	+	2B
15	|	4F	| or !	21
16	&	50	&	26
17	J	D1	J	4A
18	K	D2	K	4B
19	L	D3	L	4C
20	M	D4	M	4D
21	N	D5	N	4E
22	O	D6	O	4F
23	P	D7	P	50
24	Q	D8	Q	51
25	R	D9	R	52
26	!	5A	]	5D
27	$	5B	$	24
28	*	5C	*	2A
29	)	5D	)	29
30	;	5E	;	3B
31	¬	5F	¬ or ^	5E
32	a	81	a	61

Device Number	EBCDIC I/O Character	EBCDIC Hex	ASCII I/O Character	ASCII Hex
33	b	82	b	62
34	c	83	c	63
35	d	84	d	64
36	e	85	e	65
37	f	86	f	66
38	g	87	g	67
39	h	88	h	68
40	i	89	i	69
41	j	91	j	6A
42	k	92	k	6B
43	l	93	l	6C
44	m	94	m	6D
45	n	95	n	6E
46	o	96	o	6F
47	p	97	p	70
48	q	98	q	71
49	r	99	r	72
50	s	A2	s	73
51	t	A3	t	74
52	u	A4	u	75
53	v	A5	v	76
54	w	A6	w	77
55	x	A7	x	78
56	y	A8	y	79
57	z	A9	z	7A
58	S	E2	S	53
59	T	E3	T	54
60	U	E4	U	55
61	V	E5	V	56
62	W	E6	W	57
63	X	E7	X	58
IPC Interface	,	6B	,	2C
Sync Interface	%	6C	%	25

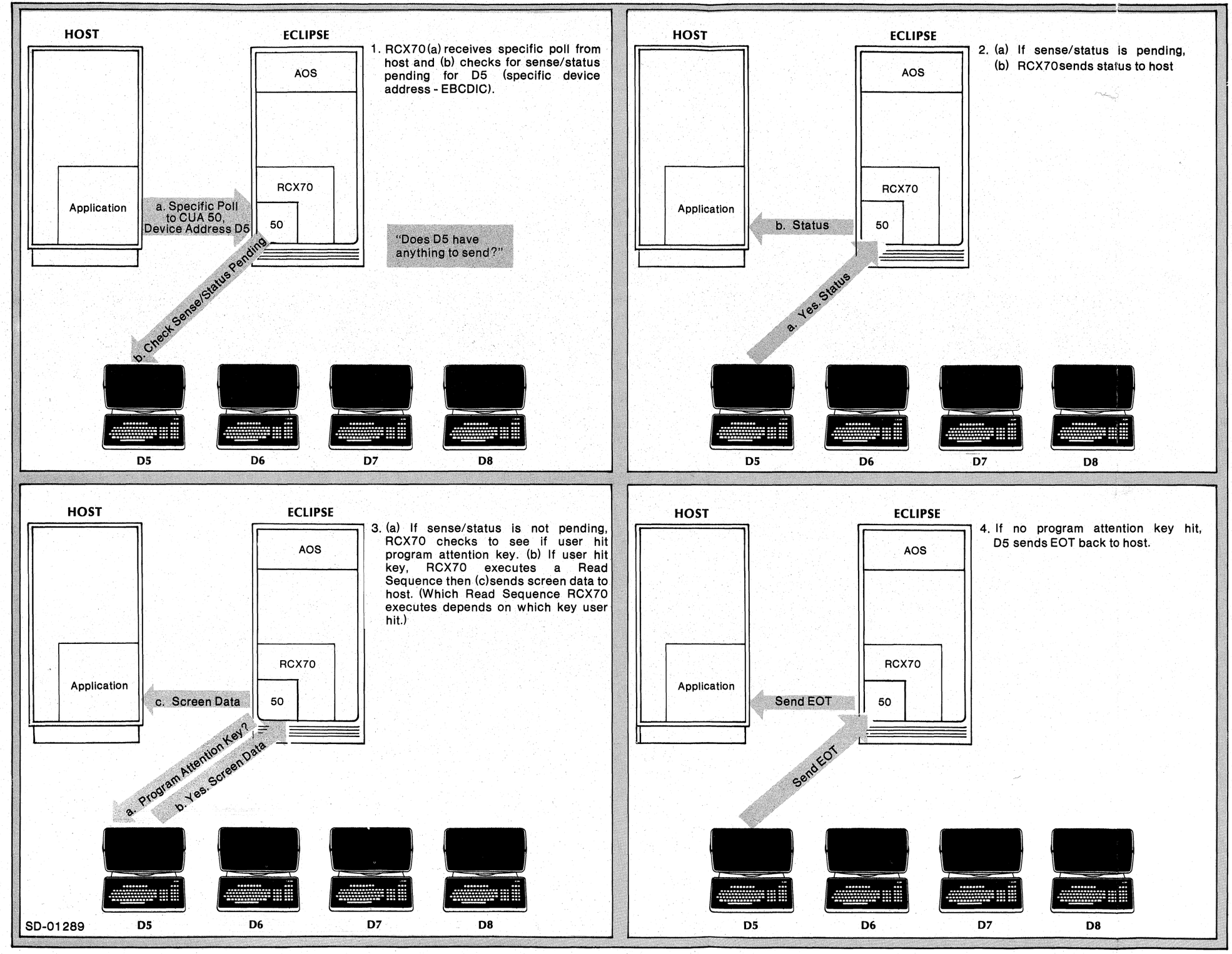


Figure 4-6. RCX70 and the Specific Poll

Select Command

Before sending any commands or data, the host application must select a device. Once it selects a device, the application sends all subsequent commands to that device until it sends an EOT. The Select command, then, provides a way for the host application to write to or read from a terminal or printer, or to issue Control commands.

The Select command is similar to the General and Specific Poll commands in that the host sends a Control Unit Address, followed by a device address, but the control unit for Select has a different CUA code. Device addresses are the same. The Select addressing sequence prepares RCX70 and a specific device for incoming Write, Read, or Control commands. We illustrate the Select data block in Figure 4-7.

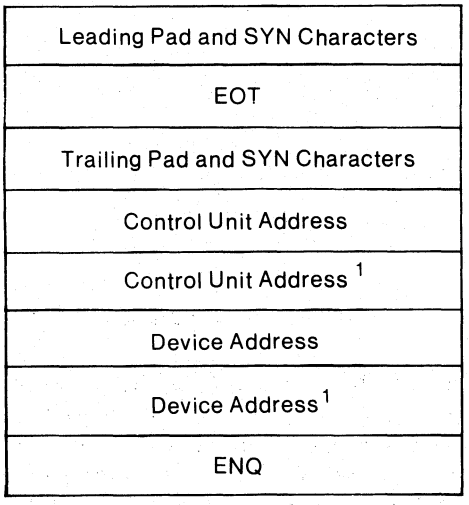


Figure 4-7. Select Command Data Block

Control Unit Address

Table 4-6 lists the control units and the corresponding EBCDIC and ASCII address for each unit.

Table 4-6. RCX70 Control Unit Selection Addresses

Control Unit Number	EBCDIC I/O Character	EBCDIC Hex	ASCII I/O Character	ASCII Hex
0	-	60	-	2D
1	/	61	/	2F
2	S	E2	S	53
3	T	E3	T	54
4	U	E4	U	55
5	V	E5	V	56
6	W	E6	W	57
7	X	E7	X	58
8	Y	E8	Y	59
9	Z	E9	Z	5A
10	¦	6A	¦	7C
11	,	6B	,	2C
12	%	6C	%	25
13	_	6D	_	5F
14	>	6E	>	3E
15	?	6F	?	3F
16	0	F0	0	30
17	1	F1	1	31
18	2	F2	2	32
19	3	F3	3	33
20	4	F4	4	34
21	5	F5	5	35
22	6	F6	6	36
23	7	F7	7	37
24	8	F8	8	38
25	9	F9	9	39
26	:	7A	:	3A
27	#	7B	#	23
28	@	7C	@	40
29	'	7D	'	27
30	=	7E	=	3D
31	"	7F	"	22

Device Address

The device address for this command is the same as for a Specific Poll command. See Table 4-5 to determine device addressing.

After the Select Command -- Summing Up

After receiving a successful Select, RCX70 acknowledges the Select and receives the next block from the host. That block will contain a command and may contain data directed to a particular screen. The screen data will appear only on a Write or Erase/Write.

Figure 4-8 describes how RCX70 receives and responds to a Select command. The host sends a Select command sequence to a Control Unit Address (different from a general poll CUA) and a device address. If RCX70 recognizes its own CUA, it acknowledges response and carries out a Read, Write, or Control command depending on which the block following the Select command specifies. Once selected, RCX70 will remain so until it receives an EOT.

Reader, Please Note

1. RCX70 will not respond to a Select:
   - if RCX70 is not up.
   - if the Select sequence is incorrect; e.g., including characters out of order or missing characters.
   - if RCX70 or another control unit is still selected; that is, no EOT has been received.
2. If the host detects a BCC error, it will respond by sending a NAK to AOS. AOS will retry transmission up to eight times. If the BCC is still incorrect, the host system will respond with an EOT. RCX70 will try to send the same data the next time it receives a Select.
3. AOS handles all data retransmission.
4. The host system sends an RVI if it wants the control unit to stop sending data on a Read (Modified or Buffer) command. The control unit responds with an EOT and stops transmitting data.
5. There is a bisynchronous rule prohibiting two consecutive, conversational replies. To conform to this rule, the application should never send a Read and then respond to the first block of data with a command block.

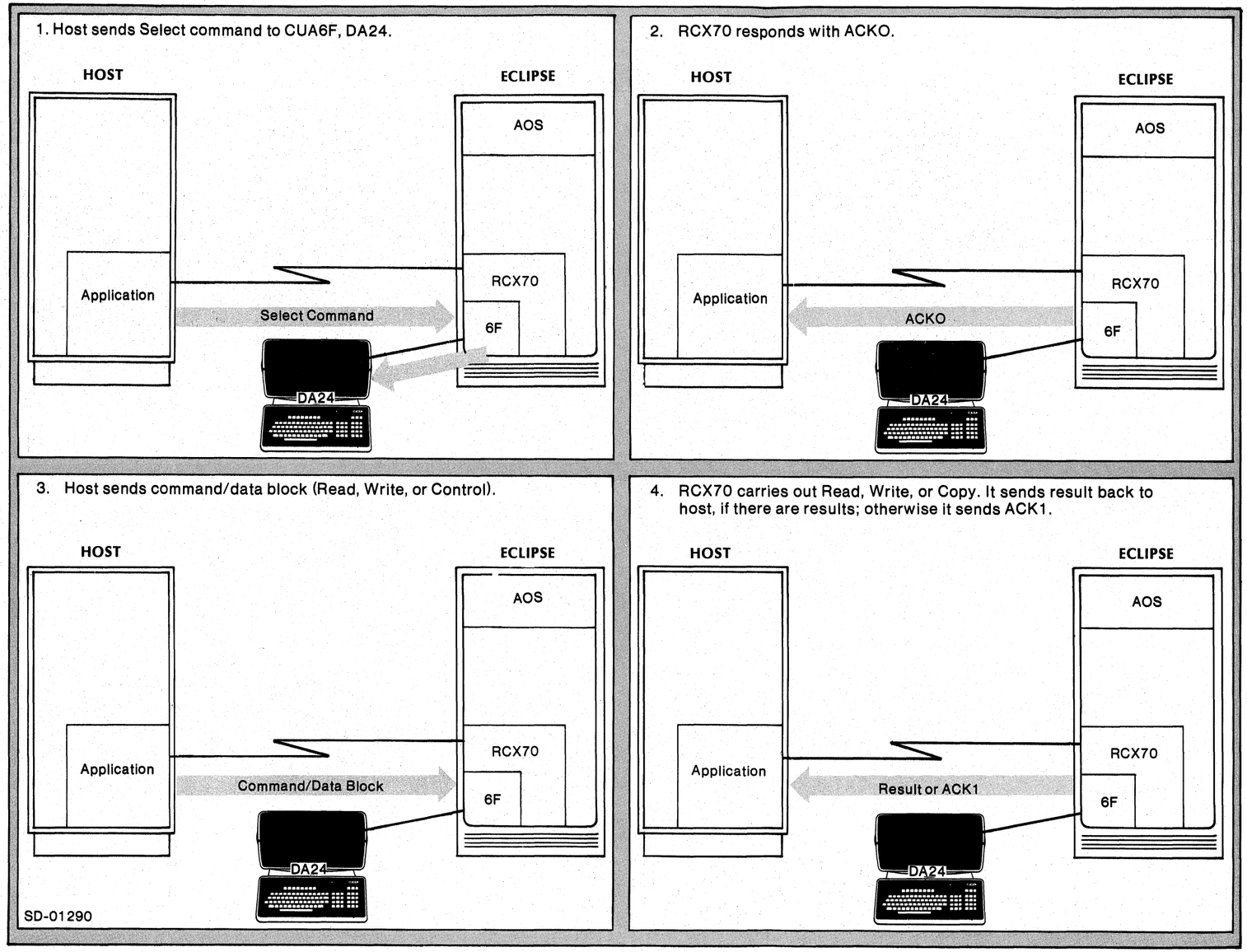
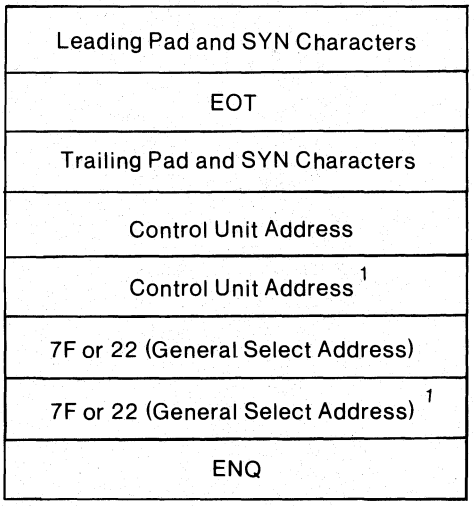


Figure 4-8. RCX70 and the Select Command

Broadcast Command

An application can use the Broadcast command to notify all RCX70 users at a particular CUA of an impending event. The application does not have to select each device in order to send a message.

When RCX70 receives a Broadcast command, it reacts exactly as it would for a Select command except that it performs the same sequence of commands for each device. See Figure 4-9 for the format of the Broadcast data block. Note that the only difference from the Select data block is the General Select Address instead of a device address. When using EBCDIC, the General Select Address is always 7F; when using ASCII, it is always 22.



[1] The host sends both the byte containing the CUA and the General Select Address byte twice.

Figure 4-9. Broadcast Command Data Block

Control Unit Address

Control unit addressing for this command is the same as for the Select command. See Table 4-6 for the control units and the corresponding EBCDIC and ASCII address for each unit.

General Select Address

In EBCDIC, the General Select Address is always 7F; in ASCII, it is always 22.

After the Broadcast Command -- Summing Up

After receiving a successful Broadcast block, RCX70 acknowledges the Broadcast and receives the next block from the remote system. That block should contain a Write or Erase/Write command. RCX70 will transmit the message in the Write or Erase/Write to all terminals communicating with the synchronous-line.

Figure 4-10 illustrates RCX70's reaction to a Broadcast command. The host sends an emergency broadcast to RCX70's CUA and a General Select Address. RCX70 responds with an ACK0 and sends the message contained in the next data block to all attached terminals.

Reader, Please Note

1. When RCX70 receives a Broadcast message from the synchronous-line, it will transmit the message only to those terminals already communicating with the synchronous line. A Broadcast message from an IPC port will go only to those terminals currently communicating with the IPC port. Neither message will go to a printer.

2. RCX70 will not respond to a Broadcast:

   - if RCX70 is not up.
   - if the Broadcast sequence is incorrect; e.g., including characters out of order or missing characters.
   - if RCX70 or another control unit is still selected; that is, no EOT has been received.

3. If the host detects a BCC error, it will respond by sending a NAK to AOS. AOS will retry the transmission up to eight times. If the BCC is still incorrect, the host system will respond with an EOT. RCX70 will try to send the same data the next time it receives a Broadcast.

4. AOS handles all data retransmission.

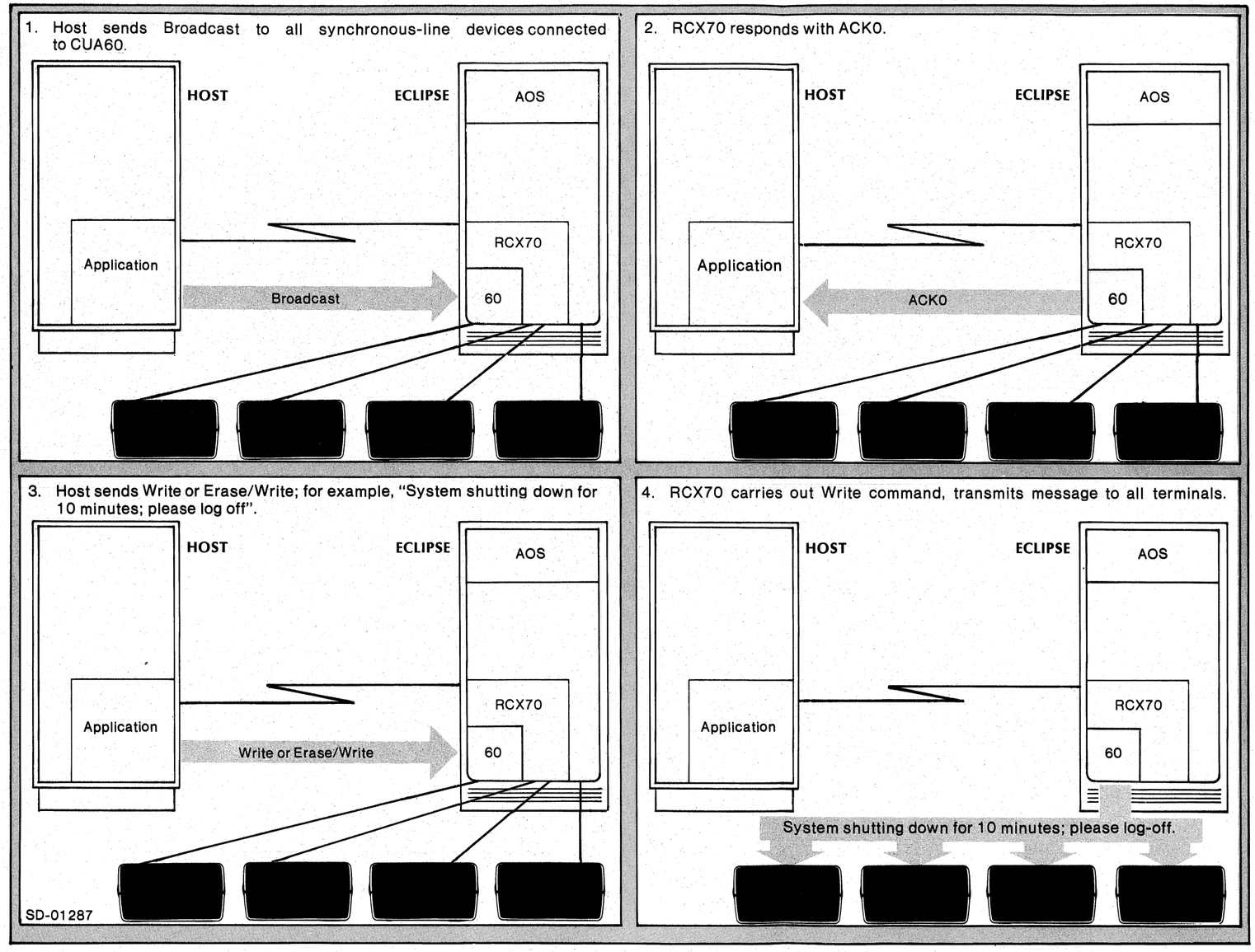


Figure 4-10. RCX70 and the Broadcast Command

Write and Erase/Write Commands

The Write and Erase/Write commands appear in a block that the host system sends after a Select. RCX70 will apply the Write command to the device selected in the Select command. We discuss Write and Erase/Write in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

Figure 4-11 illustrates the Write and Erase/Write data block sent to RCX70 after a Select Command.

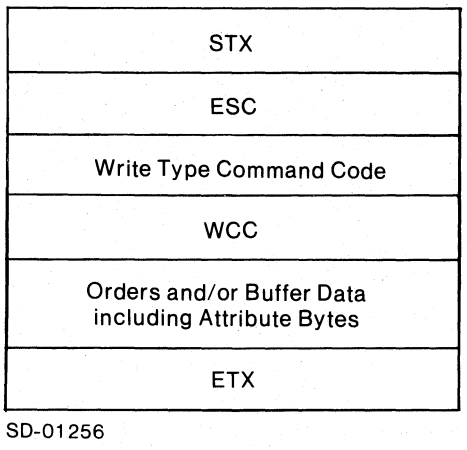


Figure 4-11. Write and Erase/Write Data Block

Command Codes

Command	EBCDIC	ASCII
Write	F1	31
Erase/Write	F5	35

WCC

For a complete discussion of the WCC (Write Control Character), see Chapter 3.

Orders and Buffer Data

We describe Write orders and buffer data in Chapter 3. See Table 3-5 and the text following it.

Reader, Please Note

1. The host system may send a sequence of blocks containing different Write orders or buffer data. AOS acknowledges each block. RCX70 processes each block, one character at a time, as it arrives. It does not check the entire block for validity before processing. Therefore, RCX70 may process and transfer certain data or orders in the block while rejecting later data or orders. This can result in RCX70 writing to an unintended part of the screen. To avoid this, the application should issue an SBA at the beginning of each block.

2. Incorrect data in the block results in the following:

   - If an ESC character does not follow the STX, RCX70 ignores the block and does not execute the command.
   - If the command code is invalid, RCX70 ignores the block and sets an error in the status bit (bit 2 of the sense/status byte).
   - If an ETX follows the command code, RCX70 assumes an all zero WCC byte, which means that nothing will happen as a result of the command. An all zero WCC in an Erase/Write command means that RCX70 will not erase the screen.

3. If your application is chaining commands to a printer, it must include the Write or Erase/Write command with the Start Print WCC bit set as the last command in the chain. If the application does not include this bit, RCX70 prints only the data sent to it thus far.

4. RCX70 carries out orders in a very literal way. If you enter an absurd set of commands or orders, e.g., write to a screen and immediately erase the screen, RCX70 will follow the instructions without question or error message.

Copy Command

The Copy command appears in a block that the host system sends after a Select command block. RCX70 will apply the Copy command to the device selected in the Select command. We discuss the Copy command in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

Figure 4-12 illustrates the Copy command data block sent to RCX70 after a Select.

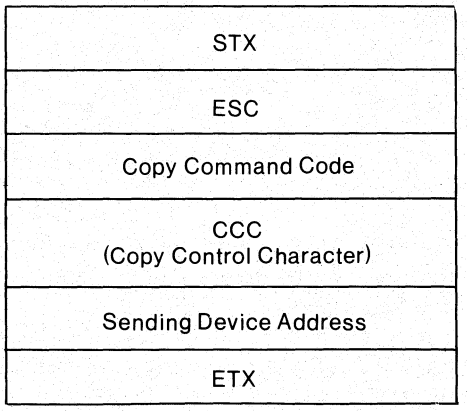


Figure 4-12. Copy Command Data Block

Command Codes

Command	EBCDIC	ASCII
Copy	F7	37

CCC

For a complete discussion of the CCC (Copy Control Character), see Chapter 3.

Sending Device Address

The Sending Device Address consists of a graphic character, which defines the device address of the device from which data is being copied.

Reader, Please Note

1. The host system may send a sequence of commands including one or more Copy commands.

2. Incorrect data in the block results in the following:

   - If an ESC character does not follow the STX, RCX70 ignores the block and does not execute the command.

   - If the command code is invalid, RCX70 ignores the block and sets an error in the status bit (bit 2 of the sense/status byte 1).

   - If RCX70 does not receive the sending address or if the address is invalid, RCX70 aborts the command and sets bit 7 of the sense/status byte 2.

Read Modified Function

The host system generates a Read Modified sequence in one of two ways:

- A general poll causes RCX70 to execute a Read Modified sequence on each RCX70 device where a user has hit an attention key. A specific poll causes RCX70 to execute a Read Modified sequence on a specified device.
- The Read Modified sequence can appear in a block that the host system sends after a Select command block. Normally, the host won't send Read Modified sequences since it is easier to poll. If the host does issue a Read Modified sequence it may receive data that the keyboard user did not want to send, since RCX70 will execute the sequence even if the user did not strike a program attention key.

Figure 4-13 illustrates the Read Modified sequence data block sent to RCX70 after a Select. For detailed information on Read Modified sequences, see Chapter 3.

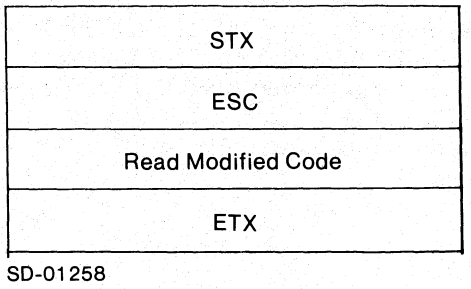


Figure 4-13. Read Modified Sequence Data Block

Read Modified Code

Command	EBCDIC	ASCII
Read Modified	F6	36

Concepts

The Read Modified sequence is the primary way that the host receives data from a screen. The Read Modified sequence transfers all fields which the user modified from the screen buffer to the host system. The Read Modified function directs RCX70 to carry out one of three sequences, depending on which program attention key the user strikes:

- Full Read Modified sequence
- Short Read sequence
- Test Request Read sequence

Table 3-2 and the text following it describe all Read sequences in detail.

Reader, Please Note

1. The AID byte values are listed in Table 3-1.
2. If RCX70 executes the Read Modified sequence as the result of a general or specific poll, the application will always see a user-generated program attention key. In remote configurations, if RCX70 receives a poll and a user has not hit a program attention key, RCX70 responds with an EOT.
3. If the host system sends a Select and then a Read Modified, and if the user has not hit an attention key, RCX70 will still execute a Read Modified.

Read Buffer Command

The Read Buffer command appears in a block that the host system sends after a Select command block. RCX70 will apply the Read Buffer command to the device selected in the Select command. You use Read Buffer primarily to check the results of a Write command.

We discuss Read Buffer in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

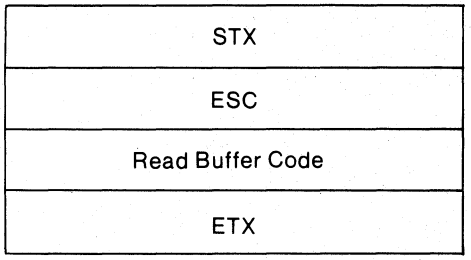


Figure 4-14. Read Buffer Data Block

Figure 4-14 illustrates the Read Buffer data block sent to RCX70 after a Select command.

Command Codes

Command	EBCDIC	ASCII
Read Buffer	F2	32

Concepts

You use the Read Buffer command to double check the results of a previous Write command. When RCX70 executes a Read Buffer, it transfers all data (including nulls) to the host application, beginning from a specified address in a screen buffer and continuing through the end of that buffer.

Reader, Please Note

The host application should not send arbitrary Read Buffer commands. If it does, RCX70 may send incomplete information to the remote system. RCX70 responds to an unsolicited Read Buffer by sending all the data the user has typed in until the time that RCX70 received the Read Buffer block.

Erase All Unprotected Command

The Erase All Unprotected command appears in a block that the host system sends after a Select command block. RCX70 will apply the Erase All Unprotected command to the device selected in the Select command.

We discuss Erase All Unprotected in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

Figure 4-15 illustrates the Erase All Unprotected data block sent to RCX70 after a Select.

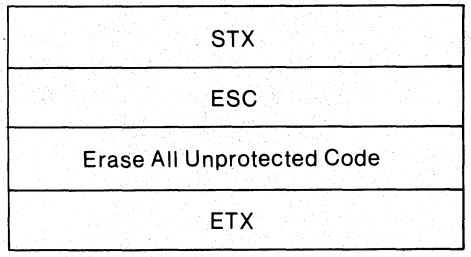


Figure 4-15. Erase All Unprotected Data Block

Command Codes

Command	EBCDIC	ASCII
Erase All Unprotected	6F	3F

Concepts

Erase All Unprotected performs five tasks at the selected device:

1. Clears all unprotected buffer characters to nulls.
2. Resets the MDT bit to 0 for every unprotected field.
3. Unlocks the keyboard.
4. Resets the AID byte to 60 EBCDIC or 2D ASCII. This clears the cause of the last program attention.
5. Repositions the cursor to the first character location in the buffer's first unprotected field. If there are no unprotected fields, this command positions the cursor to character location 0.

Reader, Please Note

1. If the entire buffer is protected, RCX70 clears nothing and does not reset MDT bits. However, RCX70 still unlocks the keyboard, resets the AID byte, and repositions the cursor to character location 0.
2. The Erase All Unprotected command may occur in a series with other commands. The entire series would apply to the selected device.

Direct 3271 Emulation Example

Please refer to Figure 4-16 as you read the following text.

Footloose Travel Agency in Boston has a Data General ECLIPSE Computer with a strictly remote RCX70 mode of operation. Footloose communicates over a multidrop synchronous line with an IBM 370 owned by Fancy Free Aviation at O'Hare Airport in Chicago.

It is a busy morning at Footloose Travel. Fancy Free just started an intensive advertising campaign for a special package that allows passengers to fly round trip from New York to London for the astoundingly low price of $127.00. At 7:00 AM, two hours before opening, the lines start forming in front of Footloose.

Joyce James, manager of Footloose Travel, pushes her way through the crowd and opens the agency. She smiles and breathes a sigh of relief. Joyce knows that she can service the large crowd quickly and efficiently now that Footloose has an RCX70 system.

Following the instructions in the *AOS System Managers Guide,* Joyce powers up her ECLIPSE, loads the disk pack, turns on the disk drive, and starts up AOS. Since Footloose has run RCX70 before, she does not have to configure a new RCX70 system. Instead, she disables CON2 from the operator's console, reserving it for RCX70, and bootstraps RCX70 by typing

```
CONTROL @EXEC DISABLE @CON2
XEQ RCX70 @CON2
```

from the AOS system console (typically CON0).

RCX70 enables CON2 as an RCX70 terminal. Since Joyce is the only employee at Footloose this morning, she specifies only one terminal for RCX70. If she wanted to enable more than one terminal, Joyce could have included more arguments in the RCX70 command; for example,

```
XEQ RCX70 @CON3 @CON4 @CON5 ...
```

RCX70 now exclusively owns CON2. The cursor appears in the upper, left corner of the screen.

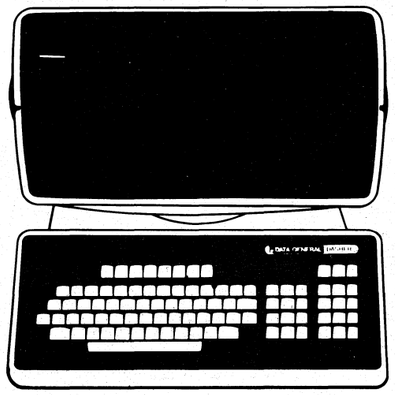
First, Joyce must determine the number of seats available on the New York to London flights. She knows she can obtain this information by pressing PF (program function) key 5, which was defined by the remote application. RCX70 receives a general poll and performs a Read Modified. The AID byte indicates PF5. RCX70 then recognizes a Select sequence on the line for CON2, followed by a Write command. And RCX70 directs the Write to CON2.

Once Joyce obtains the number of available seats, she presses PF6 to write a standard reservation screen to her terminal. Again, the PF key was defined by the host application. When the reservation form appears on the terminal, Joyce asks each customer about his/her flight plans and keys in the information on the form. When she fills a screen, Joyce strikes the ENTER key, then another PF6 for another reservation form.

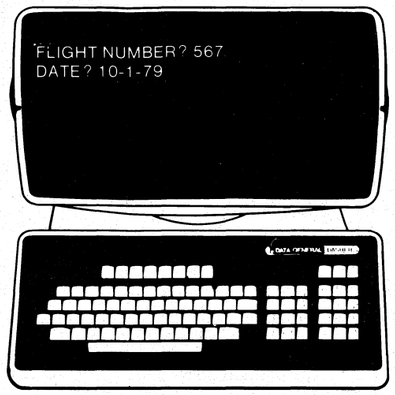
The host 370 conducts a general poll of Joyce's RCX70 and transfers all data back to Chicago. Once Fancy Free Aviation receives the data, it updates its Seats Available file. Joyce continues to fill out reservation forms and sends all data to the host at Fancy Free. She repeats this process until she books all the customers on their desired flights.

1. Joyce boots RCX70 from the system console.

2. CON2 now enabled as RCX70 terminal.

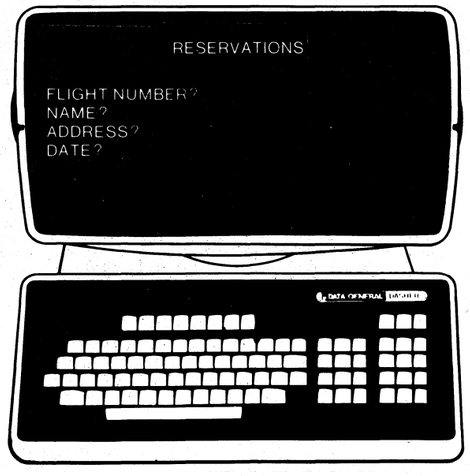


3. Joyce presses PF5. On the next general poll, RCX70 performs Read Modified and sends information to host.

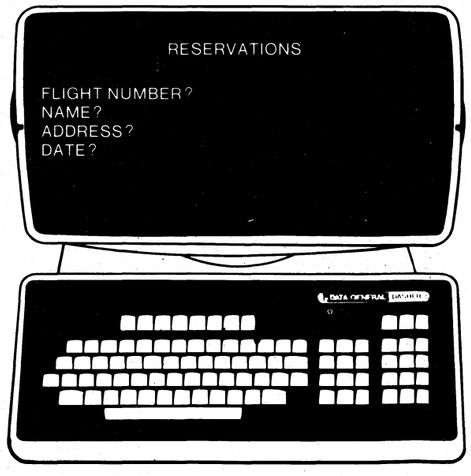


4. Host processes data and sends Select and Erase/Write commands to CON2. Joyce supplies data and hits ENTER.

5. Joyce presses PF6 to obtain reservation screen. RCX70 receives Erase/Write. Reservation screen appears on CON2.



6. Joyce fills in screen and presses ENTER.



7. Joyce presses PF6 again to obtain another reservation screen. RCX70 receives Erase/Write. Reservation screen appears on CON2.

SD-01284

Figure 4-16. Direct 3271 Emulation Mode Example

End of Chapter

Chapter 5
RCX70 As a Terminal Handler for Other AOS Processes

You can use RCX70 as a terminal handler for other AOS processes running in the same ECLIPSE system. This local interface uses the Interprocess Communication (IPC) mechanism.

This chapter provides all the information you will need if your RCX70 system is configured for a strictly local (IPC) interface. It contains an IPC overview, but you should read the appropriate sections of the following manuals if you are unfamiliar with AOS and/or IPCs.

- *AOS Programmer's Manual* (093-000120)
- *Data General Communications System* (014-000070)

A local application sends all commands to RCX70 via the IPC interface. Commands documented in this chapter include

- Write and Erase/Write
- Copy
- Erase All Unprotected
- Read Modified and Read Buffer
- Init Terminal
- Release Terminal

We discuss Write, Erase/Write, Copy, Erase All Unprotected, Read Modified and Read Buffer in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

IPC Overview

AOS provides a facility for processes to communicate with one another. This facility, called Interprocess Communications (IPC), allows you to send free format messages of arbitrary length between the RCX70 process and a local application.

IPCs are sent between ports. A *port* is a full duplex communications path to a process. Each process can have up to 128 ports. AOS assigns each port a port number. AOS also allows you to name ports by using the ?CREATE system call described in Chapter 5 of the *AOS Programmer's Manual.*

IPC Mode Initial Conditions

In a strictly local mode of operation, an RCX70 terminal communicates with a local application. All data input at the terminal goes to the application via the IPC interface and all the data seen on the screen comes from the same interface.

When RCX70 is brought up in IPC mode, it must establish contact with the application on the other end of the IPC interface. You can bring up either RCX70 or the local application first.

Figure 5-1 illustrates RCX70's course of action during initialization. First, RCX70 creates an IPC file in the working directory of the RCX70 process to identify its own port number. The name of the file is always RCX70.IPC. Next, RCX70 executes an ?ILKUP system call on the local application's IPC file. If the local application does not exist, RCX70 determines that it is not up yet. RCX70 will wait awhile and keep trying ?ILKUPs until the application comes up. If the application is not up after five minutes, RCX70 will terminate itself.

When the ?ILKUP call succeeds, it means that both RCX70 and the application have defined global ports, so RCX70 can send or receive messages with the ?ISEND and ?IREC calls. See "Block Formats" in this chapter for a description of the Send and Receive headers and RCX70's course of action after an ?ISEND or ?IREC.

Data Blocking

In an IPC mode of operation, RCX70 sends and receives message blocks of 256 words or 512 bytes. RCX70 will send a sequence of messages if a single message exceeds the 256-word limit. It will also fill out a short message so that it equals 256 words. The IPC header contains the real length of the data.

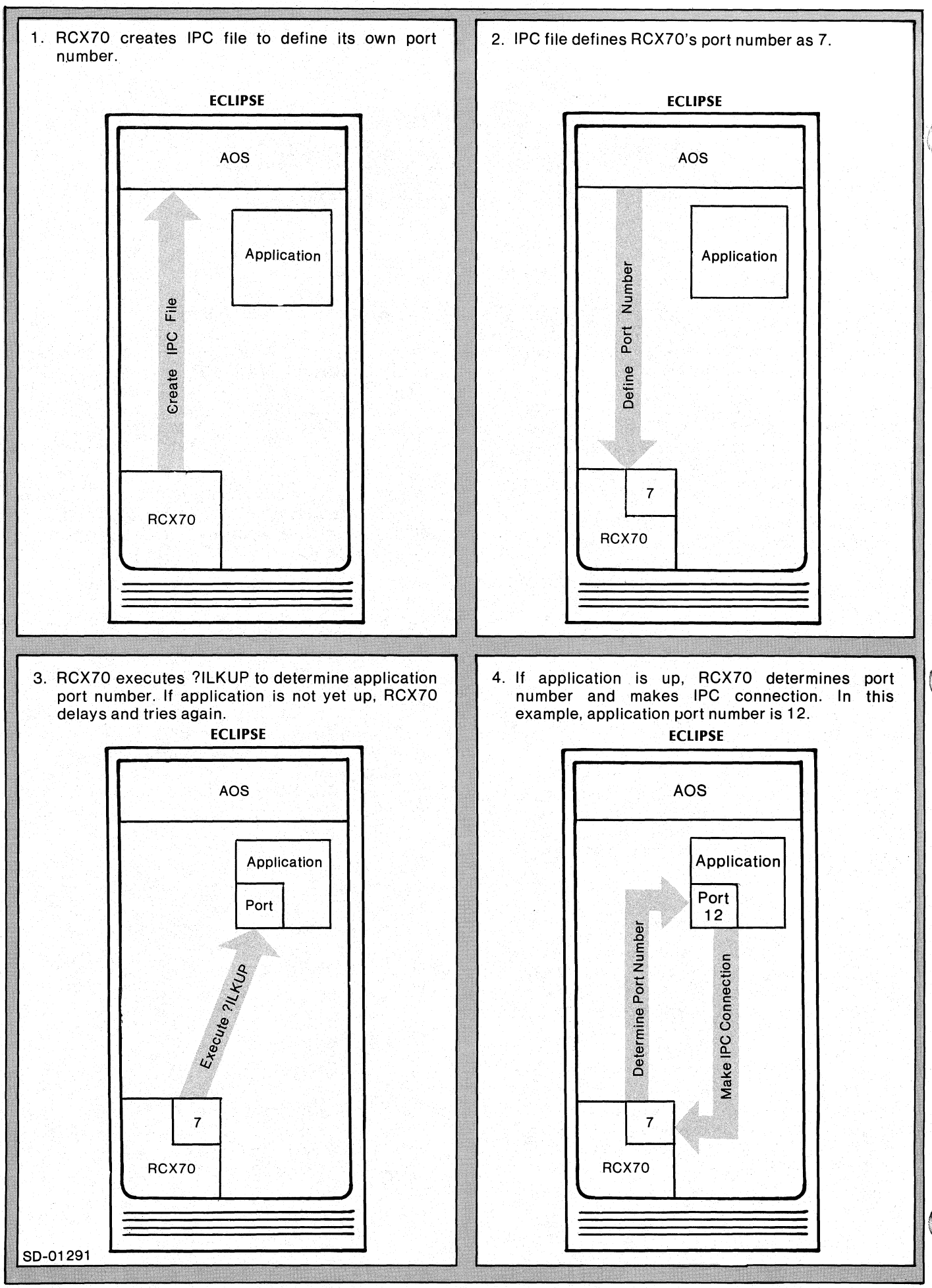


Figure 5-1. RCX70 Initialization -- First Things First

Buffer Address

RCX70 maintains a number called the current buffer address for each terminal and printer. Every time a terminal or printer receives a data byte, RCX70 inserts the byte at the current buffer address and advances that address one position. RCX70 will also alter the buffer address according to directives from specific commands and orders.

Character Set and Number Base

All IPC messages are in ASCII. All numbers in this chapter are in octal, except when noted otherwise.

Block Formats

Figure 5-2 charts the flow of operation for any command that RCX70 might send or receive in an IPC mode of operation. We explain basic IPC concepts here, but you should read Chapter 4 of the *AOS Programmer's Manual* for a detailed explanation.

Once RCX70 and the local application have defined global ports, they can send or receive messages by issuing ?ISEND or ?IREC calls. If either process sends an ?ISEND call and the other process has issued an ?IREC, AOS will send the message. If the receiving process does not have an ?IREC up, AOS will spool the message and wait for the ?IREC.

IPC Header Structures

A sending process issues a send request to the system by transmitting an ?ISEND to another process via the IPC. The sending process specifies the location of an IPC header, a parameter block which specifies information about the transfer. The IPC headers contain the following information:

Send Header

Word	Contents
0	System Flags
1	User Flags
2,3	Destination Port Number
4	Origin Port Number
5	Length of Message in Words (256)
6	Address of Message

Receive Header

Word	Contents
0	System Flags
1	User Flags
2,3	Origin Port Number
4	Destination Port Number
5	Buffer Length in Words
6	Buffer Address

If the ?ISEND is successful, the system copies words 1 through 5 of the Send header to the corresponding words in the Receive header. Word 6 of the Receive header is a pointer to a buffer that will hold the actual RCX70 command message.

IPC Header User Flags Word Format

Word 1 of both the Send and Receive headers is the user flags word. Table 5-1 shows the user flags word format in an RCX70-to-IPC message. Table 5-2 shows the user flags word format in an IPC-to-RCX70 message.

Table 5-1. IPC Header User Flags Word Format -- RCX70 to IPC

Bit	Function
0	If bit set, this is the first block of a message.
1	If bit set, this is the last block of a message.
2	If bit set, the message will be returned to the local application without processing because the command is invalid.
3	If bit set, the message will be returned without processing because the device addressed is not in use or is not configured.
4	If bit set, the message will be returned without processing because the message contains an invalid order or the buffer address is out of range.
5-6	Not used.
7-15	Data byte count.

Table 5-2. IPC Header User Flags Word Format -- IPC to RCX70

Bit	Function
0	If bit set, this is the first block of a message.
1	If bit set, this is the last block of a message.
2-6	Not used.
7-15	Data byte count.

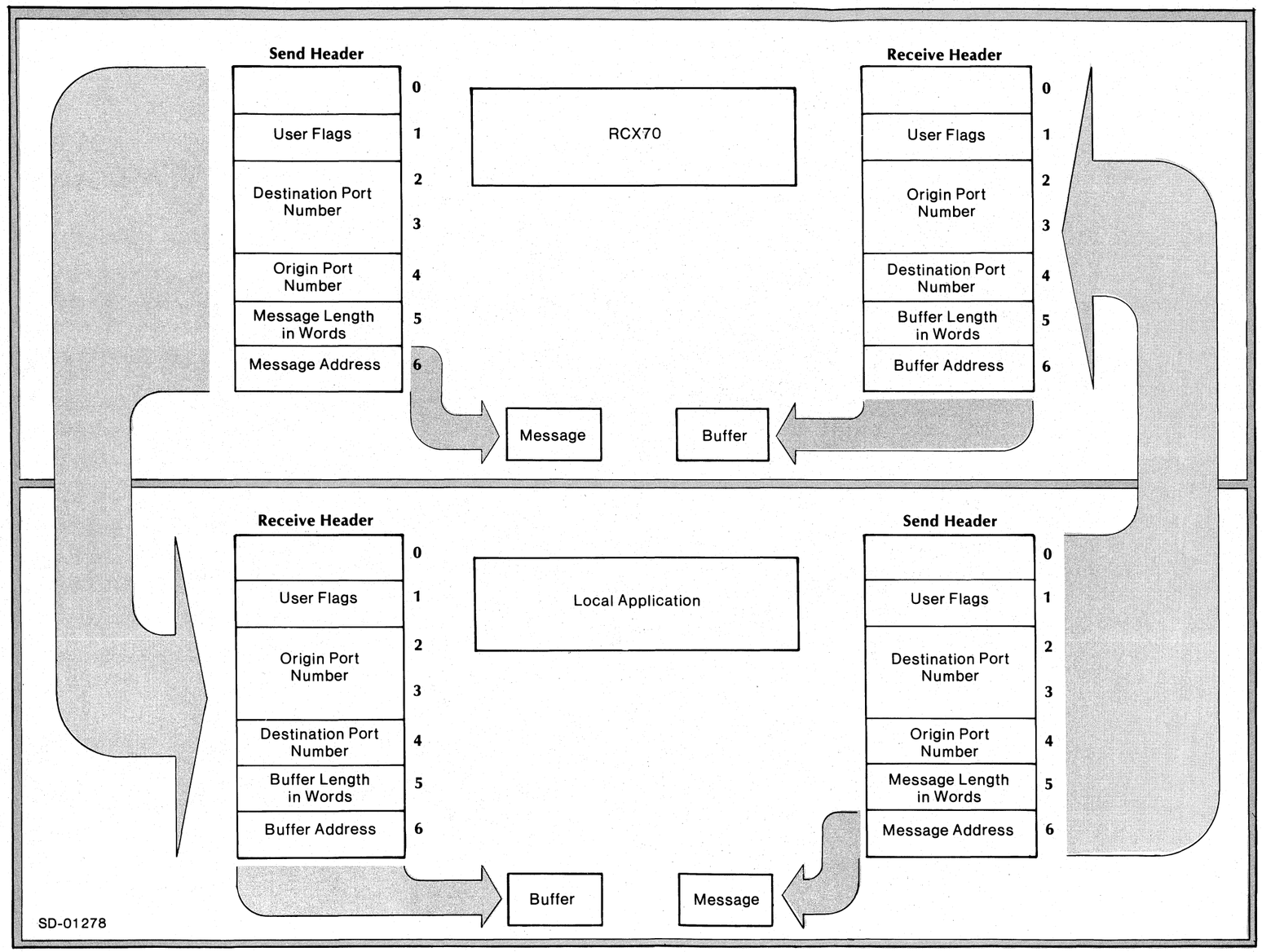


Figure 5-2. Block Formatting in RCX70 Terminal Handling Mode

RCX70-to-IPC Message Block Format

Figure 5-3 illustrates the RCX70-to-IPC message block format. RCX70 always sends messages in 256-word blocks because 256 is the most efficient size for the IPC mechanism. If the entire message cannot fit into 256 words, RCX70 writes successor blocks. RCX70 sends one or more blocks when the user hits a program attention key on a terminal communicating with the local application.

The 256-word block includes a fixed portion. RCX70 defines the actual number of bytes containing the message in the user flags word of the IPC header. This byte count (bits 7-15) does not include the fixed portion; it starts at word one. The first three bytes in the first block are always the AID byte and the two cursor address bytes.

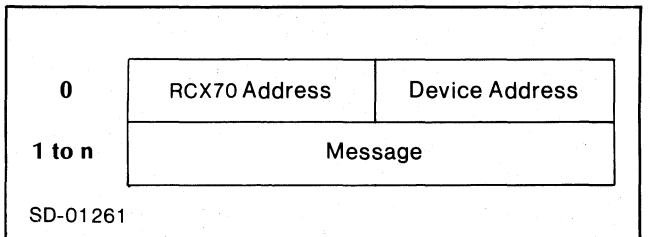


Figure 5-3. RCX70-to-IPC Message Block Format

Again, the fixed portion consists of two bytes. The user flags word of the IPC header contains the actual message length.

IPC-to-RCX70 Command Block Format

Figures 5-4 and 5-5 illustrate the IPC-to-RCX70 command block formats. The application must always send command blocks of 256 words. The IPC defines the actual number of bytes containing the commands in the user flags word of the IPC header. This byte count (bits 7-15) does not include the first two bytes (command and device address). The SBA and address bytes are part of the data and, therefore, are part of the byte count.

Since the byte count in the user flags word does not include the first two bytes, the count is zero for all commands listed in Figure 5-5 except Copy. The byte count for a Copy command is two (CCC and Sending Device Address).

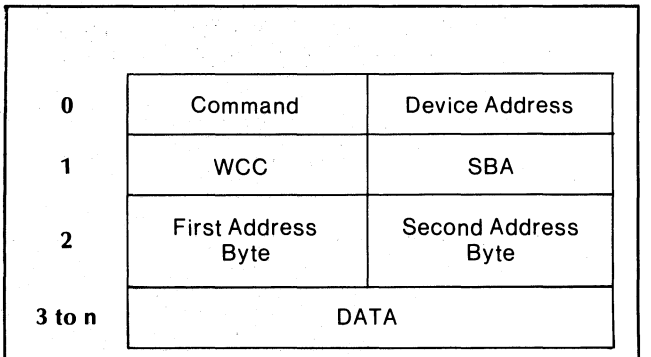


Figure 5-4. Format of Commands that Carry Data (Write and Erase/Write)

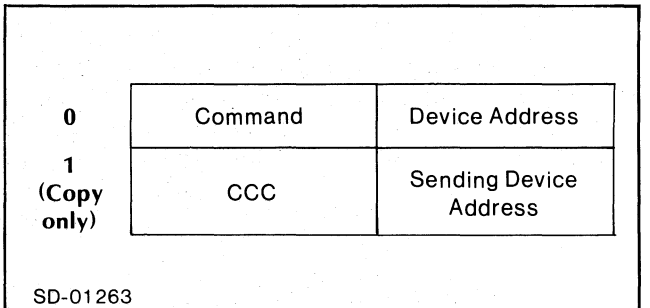


Figure 5-5. Format of Commands that Do Not Carry Data (Copy, Erase All Unprotected, Read Modified, Read Buffer, Init Terminal, Release Terminal, IPC Off)

Write and Erase/Write Commands

The IPC interface sends RCX70 the Write and Erase/Write commands. We discuss Write and Erase/Write in detail in Chapter 3. We include information specific to IPCs here, but you should read the appropriate sections of Chapter 3 carefully.

When RCX70 receives a Write command, it writes data to a terminal or printer addressed by specific Write orders. The Erase/Write command differs only in that RCX70 erases the screen before it begins writing.

Figure 5-6 shows the format of the Write and Erase/Write commands.

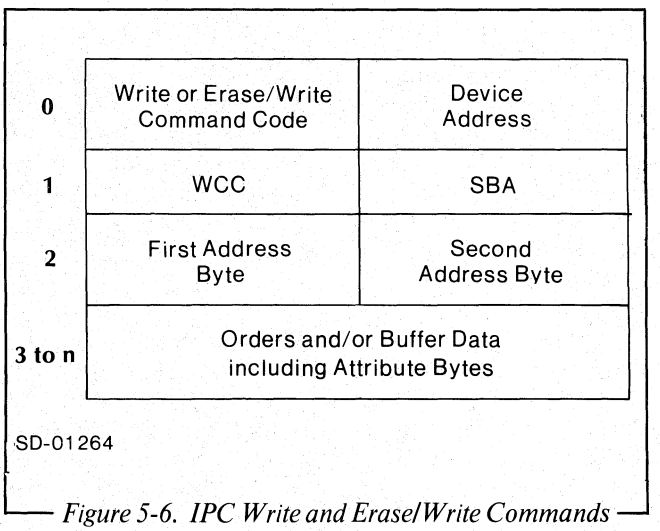


Figure 5-6. IPC Write and Erase/Write Commands

Command Codes

The numbers here are in octal. The command codes are the same as on an ASCII synchronous line.

Command	ASCII Code
Write	061
Erase/Write	065

Device Address

Table 5-3 lists the device numbers and corresponding ASCII addresses for each unit.

Table 5-3. IPC Device Addressing

Device Number	EBCDIC I/O Character	ASCII Hex	Device Number	EBCDIC I/O Character	ASCII Hex
0	□	20	33	b	62
1	A	41	34	c	63
2	B	42	35	d	64
3	C	43	36	e	65
4	D	44	37	f	66
5	E	45	38	g	67
6	F	46	39	h	68
7	G	47	40	i	69
8	H	48	41	j	6A
9	I	49	42	k	6B
10	[	5B	43	l	6C
11	.	2E	44	m	6D
12	<	3C	45	n	6E
13	(	28	46	o	6F
14	+	2B	47	p	70
15	!	21	48	q	71
16	&	26	49	r	72
17	J	4A	50	s	73
18	K	4B	51	t	74
19	L	4C	52	u	75
20	M	4D	53	v	76
21	N	4E	54	w	77
22	O	4F	55	x	78
23	P	50	56	y	79
24	Q	51	57	z	7A
25	R	52	58	S	53
26	]	5D	59	T	54
27	$	24	60	U	55
28	*	2A	61	V	56
29	)	29	62	W	57
30	;	3B	63	X	58
31	^	5E	IPC Interface	,	2C
32	a	61	Sync Interface	%	25

You can perform a Broadcast command over the IPC interface by using the general address code (22, 42 octal). A Broadcast message will go only to those terminals that are communicating over the IPC interface. For details on the Broadcast command, see Chapter 4.

WCC

For a complete discussion of the WCC (Write Control Character), see Chapter 3.

SBA

RCX70 specifies that the Write and Erase/Write IPC commands must start with an SBA (Set Buffer Address) order. The two bytes following the SBA set the new buffer address.

Orders and Buffer Data

We describe Write orders and buffer data in Chapter 3, see Table 3-5 and the text following it.

Reader, Please Note

1. There is no command chaining to determine start addresses under an IPC mode of operation. Therefore, Write and Erase/Write must start with a SBA.

2. RCX70 must receive a WCC even if the application sends nothing else. If it does not receive one, an Erase/Write will not erase the screen.

3. If the WCC specifies an action that the device cannot perform, RCX70 does not generate an error message.

4. If the WCC Start Print bit is not set, RCX70 ignores the print format specification. Print format specifications only affect printers.

5. RCX70 carries out orders in a very literal way. If you enter an absurd set of commands or orders, e.g., write to a screen and immediately erase the screen, RCX70 will follow the instructions without question or error message.

6. If an error occurs, RCX70 will write back the entire message with one or more of the error bits set in the IPC user flags word. Table 5-1 shows the format of the user flags word.

Copy Command

An application sends RCX70 the Copy command via the IPC interface. We discuss Copy in detail in Chapter 3. We include information specific to IPCs here, but you should read the appropriate sections of Chapter 3 carefully.

When RCX70 receives a Copy command it copies all or part of a device buffer to another addressed device. Copy is particularly useful for making hard-copies of screen data. Figure 5-7 shows the format of the IPC Copy command.

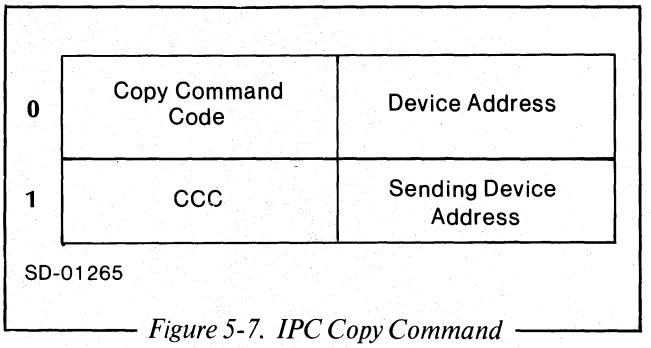


Figure 5-7. IPC Copy Command

Command Code

Command	ASCII
Copy	67

Device Address

This is the destination device address. See Table 5-3 to determine the address.

CCC

For a complete discussion of the CCC (Copy Control Character), see Chapter 3.

Sending Device Address

This is the device from which the data will be copied. See Table 5-3 to determine the address.

Reader, Please Note

1. A Copy command will always fit into one 256-word block. Therefore, you will never need successor blocks.

2. If the destination device is unavailable, RCX70 will copy the entire message with one or more error bits set in the IPC user flags word. Table 5-1 illustrates the bit format of the user flags word.

Erase All Unprotected Command

A process sends RCX70 the Erase All Unprotected command via the IPC interface. We discuss Erase All Unprotected in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

The Erase All Unprotected command performs five tasks at the addressed device:

1. Clears all unprotected buffer characters to nulls.
2. Resets the MDT bit to zero for every unprotected field.
3. Resets the AID byte to zero for every unprotected field; this clears the cause of the last program attention.
4. Unlocks the keyboard.
5. Positions the cursor to the last character location in the buffer's first unprotected field. If there are no unprotected fields, the command positions the cursor to location 0.

Figure 5-8 illustrates the format of the IPC Erase All Unprotected command.

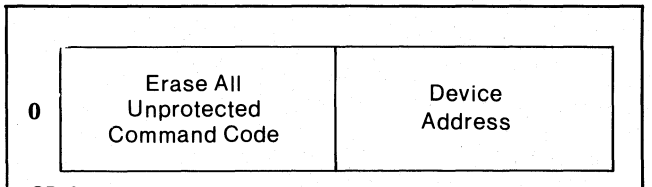


Figure 5-8. IPC Erase All Unprotected Command

Command Code

Command	ASCII Code
Erase All Unprotected	077

Device Address

See Table 5-3 to determine the desired device address.

Reader, Please Note

1. If the entire buffer is protected, RCX70 clears nothing and does not reset MDT bits. However, RCX70 still unlocks the keyboard, resets the AID byte, and repositions the cursor to location 0.
2. Erase All Unprotected will always fit into one 256-word block. Therefore, you will never need a successor block.

Read Modified and Read Buffer Commands

An application might send RCX70 the Read Modified or Read Buffer command over the IPC interface. This will not usually happen since the IPC-configured RCX70 sends modified data automatically when a keyboard user strikes a program attention key.

We discuss the Read Modified and Read Buffer commands in detail in Chapter 3. We include specific IPC information here, but you should read the appropriate sections of Chapter 3 carefully.

Figure 5-9 illustrates the format of the Read Modified and Read Buffer commands.

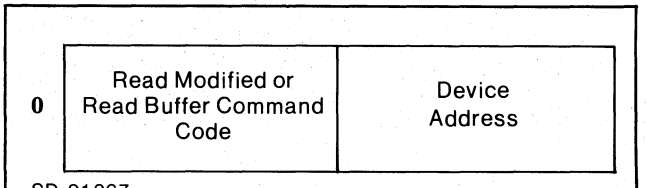


Figure 5-9. IPC Read Modified and Read Buffer Command Format

Command Codes

Command	ASCII Code
Read Modified	66
Read Buffer	62

Concepts

An application uses the Read Modified command to obtain all modified data from a screen. Under an IPC mode of operation, the system sends modified data automatically. Normally, this is all that's required since the process which initially placed the screen will already know about unmodified data.

You use the Read Buffer command to double check the results of a previous Write command. When RCX70 executes a Read Buffer, it transfers all data (including nulls) to the host application, beginning from a specified address in a screen buffer and continuing through the end of that buffer.

Device Address

See Table 5-3 to determine the desired device address.

Reader, Please Note

The only difference from the corresponding synchronous-line commands is that RCX70 always assumes the starting address of the IPC commands is 0.

Init Terminal Command

RCX70 receives the Init Terminal command from a process that wants to use RCX70 for terminal support. RCX70 gets the message from the IPC interface and issues the Assign command to the AOS PMGR. This designates the addressed terminal as a RCX70 terminal.

The Init Terminal command consists solely of the two bytes shown in Figure 5-10.

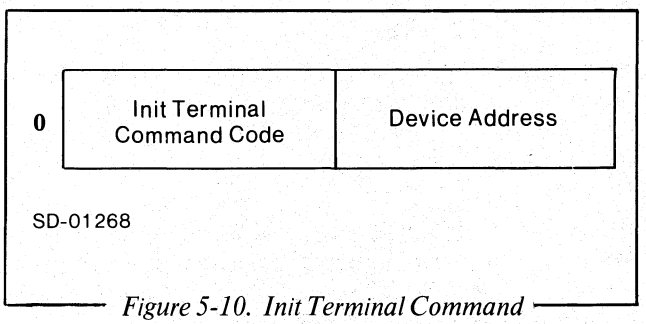


Figure 5-10. Init Terminal Command

Command Code

Command	ASCII Code
Init Terminal	070

Device Address

The device address is the address of the terminal the process wants to assign. See Table 5-3 for all device addresses.

Reader, Please Note

If RCX70 cannot assign the device, it writes a command message, INIT TERMINAL FAILURE, to the process. The ?IPTR word of the IPC header will contain the device address of the terminal. There are three reasons why the assign might fail: the terminal was not configured into RCX70, the terminal is currently assigned to another process, or the terminal does not exist on the AOS system.

Release Terminal Command

RCX70 receives the Release Terminal command from a process that wants to cut off terminal support from RCX70. RCX70 receives the message from the IPC interface, then issues a Deassign command to the AOS PMGR. This takes terminal support away from RCX70 so that the process can use the terminal for something else.

The Release Terminal command consists solely of the two bytes shown in Figure 5-11.

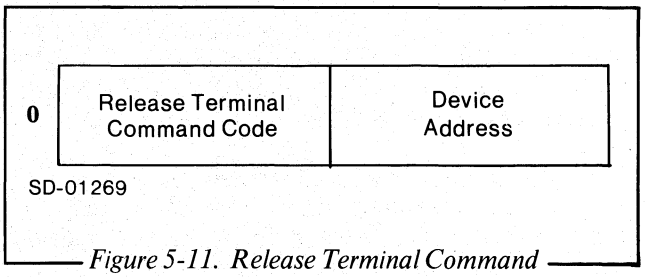


Figure 5-11. Release Terminal Command

Command Code

Command	ASCII Code
Release Terminal	071

Device Address

The device address is the address of the terminal the process wants to deassign. See Table 5-3 for all device addresses.

Reader, Please Note

If RCX70 cannot deassign the device, it writes a RELEASE TERMINAL FAILED command message to the process. The ?IPTR word of the IPC header contains the device address of the terminal that RCX70 could not deassign.

Terminal Handling Mode Example

Please refer to Figure 5-12 as you read the following text.

The O'Mally Memorial Library is a large one with an immense inventory. The conventional card catalog system was slow, inefficient, and dangerous since inconsiderate users would often remove the cards from the drawer and either replace them improperly or not replace them at all. As the library grew, the trustees realized that they would have to modernize their file system. They chose a Data General ECLIPSE and RCX70. Since all the necessary information could be accessed locally, they configured their RCX70 system for a strictly IPC interface. All the book titles in the library were entered on disk and the programmer used a database-oriented file management system (DG's INFOS) so that the books could be cross-referenced under the categories Subject, Title, and Author.

Marion Farion, the librarian, is responsible for conducting daily library searches at O'Mally. She remembers the old system all too well. A student would approach her with a bare minimum of information about a particular book. She would go to the card catalog, painstakingly search through a multitude of cards, and if she was lucky, find the card in its proper place. Marion would then copy the call number, refer to the shelf layout in front of her desk, and direct the student to the correct section of the library. By the end of the day, Marion's feet hurt from walking, her legs and back hurt from stooping over the catalog, and her vision was rapidly deteriorating.

When RCX70 was installed, Marion learned to breeze through her job. She comes into the library at 8:00 AM and disables CON2, reserving it for RCX70. She bootstraps RCX70 by typing

```
XEQ RCX70 @CON2
```

from the AOS CLI at the operator's console. If she wanted to, Marion could designate as many terminals as she wished for RCX70 by typing

```
XEQ RCX70 @CON2 @CON3 @CON4 ...
```

CON2 is now exclusively owned by RCX70. The cursor appears in the upper, left corner of the screen. Marion sits back, drinks a morning cup of coffee, and waits for her first researcher.

A young man shuffles over to Marion's desk and states that he desperately needs a book to complete the research for his final exam. But his exam is that afternoon and he is not sure about the book's title.

"It's Distributed something or other," he sheepishly states. "And I believe the author's name is T.S. Whitman."

Marion knows that there is a book search application entitled, appropriately enough, BOOKSEARCH. She strikes PF3 which she knows selects the BOOKSEARCH function.

BOOKSEARCH writes an Erase/Write back to CON2, asking which specific function Marion wishes to access. Her choices are SUBJECT, TITLE, and AUTHOR. She keys in AUTHOR and again hits ENTER.

The application sends another Erase/Write, this time asking for the name of the author. Marion types the name, T.S. Whitman, hits ENTER, and sips her coffee while the application does all the work. Within a matter of seconds, the application searches the Author disk file, obtains the titles of all books in the library by Mr. Whitman, determines where they are located, and writes all of this information back to CON2 in an Erase/Write command. One of the titles is "Distributed Internal Architecture in Data Operations." Marion asks the student about the title to be sure that this is indeed the book he was searching for and sends him directly to the proper shelf.

For the remainder of the day, Marion searches for hundreds of books with RCX70. As new books come into the library, Marion uses another local application, called UPDATE, to enter them on disk. Students are serviced quickly and efficiently, and Marion goes home feeling fresh and relaxed.

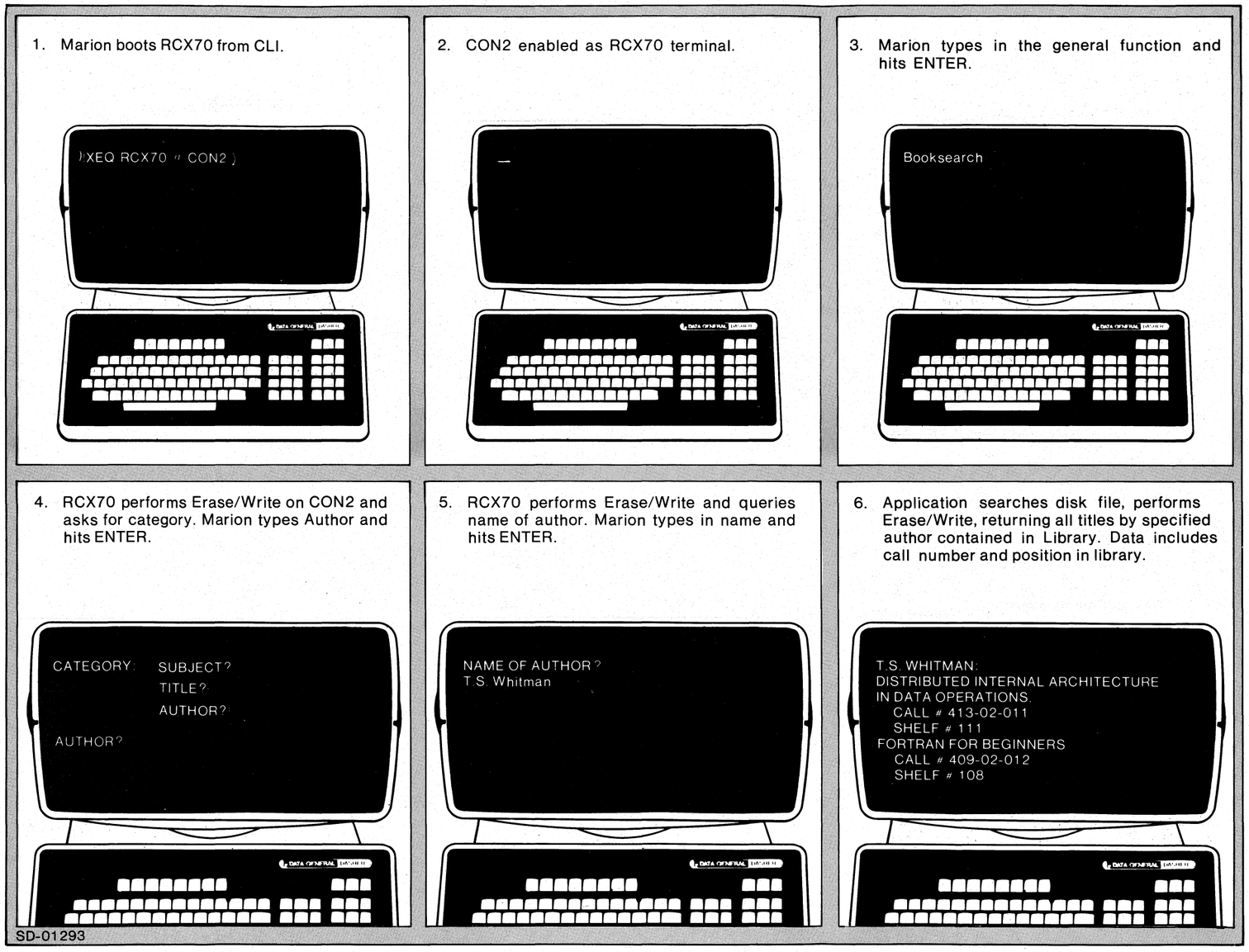


Figure 5-12. RCX70 IPC Example

End of Chapter

Chapter 6
Dual Interface
(Distributed Processing Mode)

Chapter 6 supplies information pertaining to an RCX70 system configured for a dual (synchronous-line and IPC) interface. If you are not familiar with basic synchronous-line or IPC concepts, you should read the overviews in Chapters 4 and 5 as well as the appropriate sections of the following manuals:

- *AOS Programmer's Manual* (093-000120)
- *Data General Communications System* (014-000070)

Dual Interface Overview

Under a dual-interface mode of operation, processes might want to communicate with RCX70 for four different purposes.

1. *Distributed processing.* A local application does some preliminary processing on data entered at the keyboard. Then RCX70 sends the data over the synchronous line to a remote application.

2. *Screen Handling.* One (and only one) process can communicate with RCX70 terminals so that a local application can use RCX70's screen handling facilities. There will be no need to communicate over the synchronous line. All the data which appears on the screen comes from the local application via the IPC interface, and RCX70 routes all data entered at the keyboard back to that application using the same interface. At the same time, other RCX70 terminals can communicate with a host application via a synchronous line.

3. *RCX70 as a synchronous-line handler.* You can configure an RCX70 system with a synchronous-line and IPC interface, but with no terminals. In this case, RCX70 provides applications with easy access to the synchronous line. You send data to and receive it from RCX70 via IPCs while RCX70 takes care of all the complexities of the synchronous-line protocol. Data General's IDEA can use RCX70 this way.

4. You can build an RCX70 system in which one process (a local application) communicates with terminals or with terminals and the synchronous line. In this system, any number of other AOS processes can also communicate with the synchronous line. This is a combination of numbers 1, 2, and 3.

A local application and any process using RCX70 as a synchronous-line handler send all commands to RCX70 via the IPC interface. Commands documented in this chapter include

- Write and Erase/Write
- Init Terminal
- Release Terminal
- IPC Off
- Disconnect

We discuss Write and Erase/Write in detail in Chapter 3. We repeat certain information here for your convenience, but you should read the appropriate sections of Chapter 3 carefully.

The dual-interface mode of operation differs from pure synchronous-line and pure IPC modes of operation. In a dual-interface mode, one program can communicate with another program; whereas in a pure synchronous-line or pure IPC mode, programs communicate only with a terminal.

If you attempt to execute a Read Modified, Read Buffer, Copy, or Erase All Unprotected command addressed to another program (host or local), RCX70 will stop you and report an error. A host application cannot issue these commands to a local application, or to a process using RCX70 as a synchronous-line handler. Similarly, a local application or a process using RCX70 as a synchronous-line handler cannot issue these commands to the synchronous line because RCX70 cannot pass the command down the synchronous line to the host application.

You must not issue Read commands to another program because RCX70 cannot force the other program to supply any data. When you need data from another program, you must wait for it to send the data to you. If yours is the host application, it must poll until it gets the data it wants. If yours is the local application or a process using RCX70 as a synchronous-line handler, it must issue an ?IREC and wait for the data to be sent.

You should not issue Copy or Erase All Unprotected commands to another program since they only have meaning when addressed to terminals.

Dual-Interface Mode of Operation Initial Conditions

In a dual-interface mode of operation, RCX70 terminals can communicate with a local application or with the synchronous line. Both interfaces have specific device addresses (see Table 6-1). When RCX70 comes up, all terminals are initially set to communicate with the local application.

If you bring RCX70 up in IPC mode, it must establish contact with the application on the other end of the IPC interface. It does not matter if you bring up RCX70 or the local application first.

Figure 6-1 illustrates RCX70's course of action during initialization. First, RCX70 creates an IPC file to identify its own global port number. Next, RCX70 executes an ?ILKUP system call on the local application's IPC file. If the local application does not exist, RCX70 determines that it is not up yet, delays for a bit, and keeps trying ?ILKUPs until the application comes up. If the application is not up after five minutes, RCX70 will terminate itself.

When the ?ILKUP call succeeds, it means that both RCX70 and the application have defined global ports, so RCX70 can send or receive messages with the ?ISEND and ?IREC calls. See "Block Formats" in Chapter 5 for a description of the Send and Receive headers and RCX70's course of action after an ?ISEND or ?IREC.

RCX70 will change the route of terminal output from the IPC interface to the synchronous-line interface when it receives a Disconnect command from the local application. After it receives a Disconnect, RCX70 will send all messages directly to the synchronous line.

You could set up a local application so that the first thing a user would enter would be the function (s)he is attempting to perform. The local application would determine whether the function must be done locally or remotely. For example, BILLING might be a local function while AIRLINE RESERVATION might be a remote function. (We expand on this in the example at the end of this chapter.)

If the function is purely remote, the local application sends a Disconnect to RCX70 and all keyboard input goes directly to the synchronous-line interface until the user finishes. Once (s)he is finished (s)he strikes the TERMINATE key, instructing RCX70 to route the next message to the local application.

Buffer Address

RCX70 maintains a number called the current buffer address for each terminal and printer. Every time a terminal or printer receives a data byte, RCX70 inserts that byte at the current buffer address and advances the address one position. RCX70 can also alter the buffer address according to directives from specific commands and orders.

Data Blocking

In a dual-interface mode of operation RCX70 sends and receives IPC message blocks of 256 words or 512 bytes. RCX70 will send a sequence of messages if a single message exceeds the 256-word limit. It will also fill out a short message so that it equals 256 words. The IPC header contains the real length of the data.

You should note that the 256-word block is twice as large as the block size for messages sent over the synchronous line. Messages to RCX70 which are directed to the synchronous line have a maximum synchronous-line transmission block of 256 bytes. Two of these bytes are the STX and ETX characters, so you can send only 254 message bytes at once. If you want to access the synchronous line, your process should send a maximum of 508 bytes (254 words) on each IPC message. Otherwise, you will end up with short messages. For example, if a process sends 512 bytes (256 words) in a single IPC message the result will be three synchronous line messages: two containing 254 bytes and the other containing just four bytes.

Number Base

All numbers in this chapter are in octal, unless noted otherwise.

Character Set

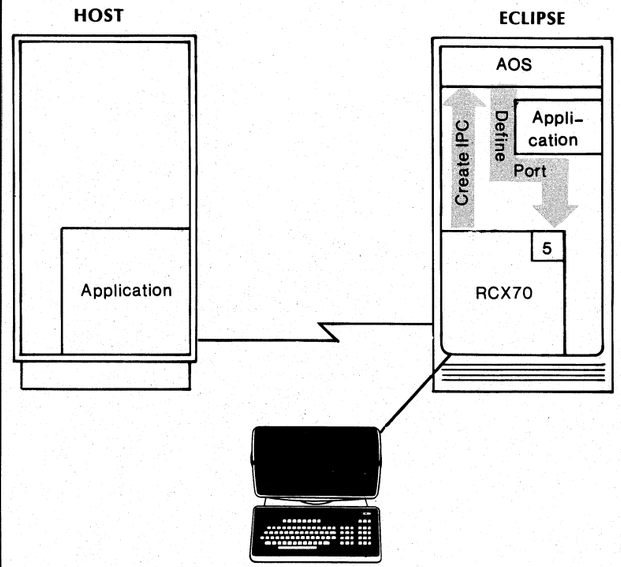
RCX70 assumes that all IPC messages are in ASCII. If the synchronous-line is operating in EBCDIC, RCX70 will translate all IPC ASCII code to EBCDIC before sending it to the synchronous line. Similarly, RCX70 will translate synchronous line EBCDIC code to ASCII before sending it to the IPC interface. All commands in this chapter are shown in ASCII only.

Block Formats

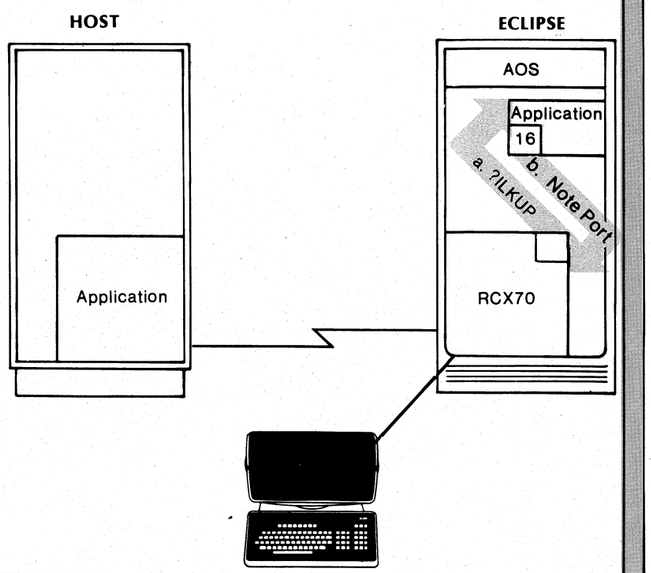
We explain IPC block formats in detail in Chapter 5. When sending to a synchronous line, RCX70 looks only at the first two bytes of the header; it does not check the validity of the WCC, SBA, etc. You can start the third byte with any character or sequence of characters, although you must organize your message in a format that the receiving program will recognize.

093-000172-00

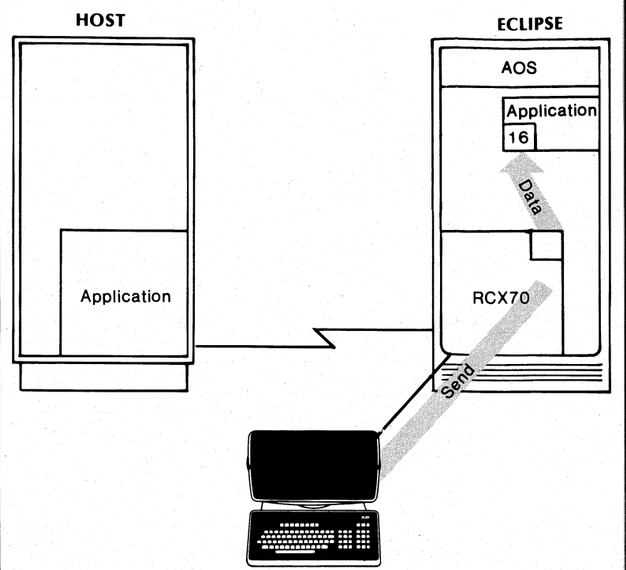
1. RCX70 creates IPC file to define its own port number. IPC file defines RCX70's global port as 5.



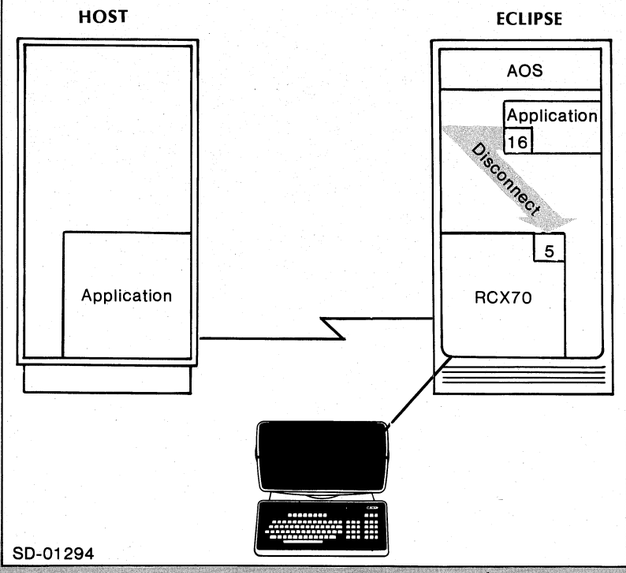
2. (a) RCX70 executes ?ILKUP to determine application's global port number. If application isn't up yet, RCX70 delays and tries again. (b) If application is up, RCX70 notes the application's global port number(in this example,16).



3. User keys in a purely remote function and sends data to local application.



4. Application determines that function is remote, issues Disconnect command.



SD-01294

5. All data goes to synchronous line until user strikes terminate key. Host application processes data and sends it back to RCX70.

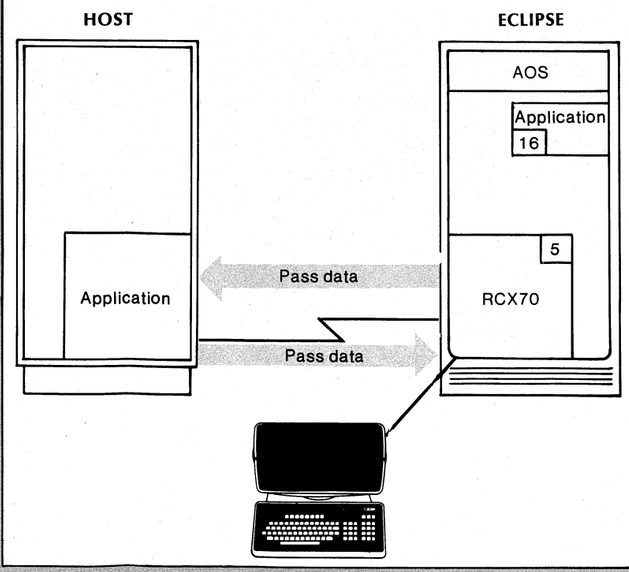


Figure 6-1. RCX70 During Initialization in a Dual-Interface Mode of Operation.

Distributed Processing

A local application may do some preliminary processing on keyboard data before sending it over the synchronous line to a remote application. Figure 6-2 illustrates distributed processing under RCX70. It shows a message moving from a terminal to a local application, then to the remote system, back to the local application, and finally back to the terminal.

A user keys in data and hits ENTER. After (s)he strikes ENTER, the keyboard will lock until it receives an unlock (WCC) sequence. RCX70 routes the data from the keyboard, over the IPC interface, to the local application. The local application processes whatever data it can and writes back to RCX70, specifying the synchronous-line interface address. If necessary, RCX70 translates the code to EBCDIC and sends the data over the synchronous line.

RCX70 handles communication between the synchronous-line and IPC interfaces by assigning a device address to each process communicating with the IPC interface. Since host-bound messages carry the device address of the IPC interface (the sending device), all messages sent from the local application to the host application look to the host like they came from the same device.

Using RCX70 as a 3270 Protocol Emulator

In Chapters 4 and 5, we discussed purely remote and purely local RCX70 configurations. This chapter covered distributed processing which uses both remote and local interfaces. In all of these cases, processes communicate with RCX70 terminals. However, you can allow other AOS processes to use RCX70 to send and receive over the synchronous line. These processes will not communicate with RCX70 terminals. In these cases, RCX70 defines the device address for the synchronous line. It then ensures that responses from the synchronous line go to the right process. Figure 6-3 illustrates RCX70 as a synchronous line handler.

Any number of processes can simultaneously use RCX70 as a synchronous line handler. The only limit is that you must configure enough device tables so that every process can have one. RCX70GEN allows you to configure the desired number of tables. In order to use RCX70 as a synchronous line handler, a process must follow this procedure:

1. The process does an ?ILKUP on RCX70.IPC; this defines the global port numbers.

2. The process issues an ?ISEND to the global port defined by the ?ILKUP. The command must be a Write or Erase/Write and the device address is %, the synchronous line.

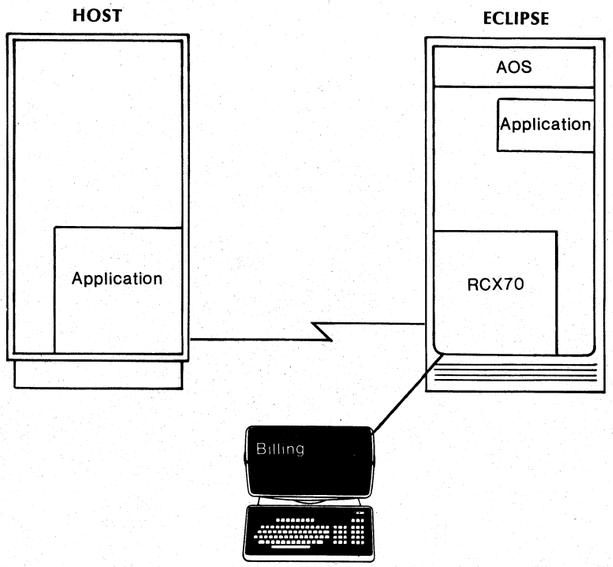
3. RCX70 receives the IPC message and scans its device tables to see if it recognizes the sending process's global port number. If this is the first message, RCX70 does not recognize the port. Therefore, RCX70 finds a free device table and assigns it to the process. Each device table has a device address in it, so the device address is now associated with the process. RCX70 will send the message down the synchronous line when it receives the next poll.

4. The process should now put up an ?IREC to wait for a response. The host application selects RCX70 and sends a reply. The device address associated with the process was sent with the initial message. So RCX70 now uses the device address to send the response to a process.

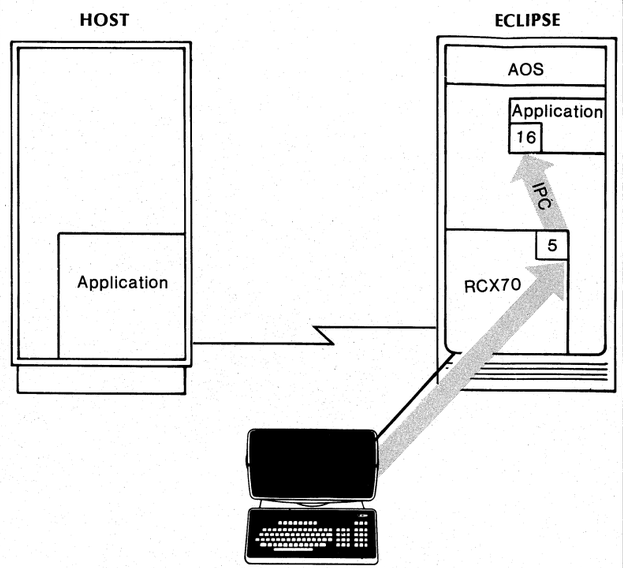
5. The process can continue to send messages. RCX70 will find the same device table each time. Therefore, RCX70 will use the same device address each time. When the process is finished, it sends an IPC Off command to free the device table.

Keep in mind that the process never has to bother about choosing a device address. RCX70 assigns the device address and routes messages to the proper place.

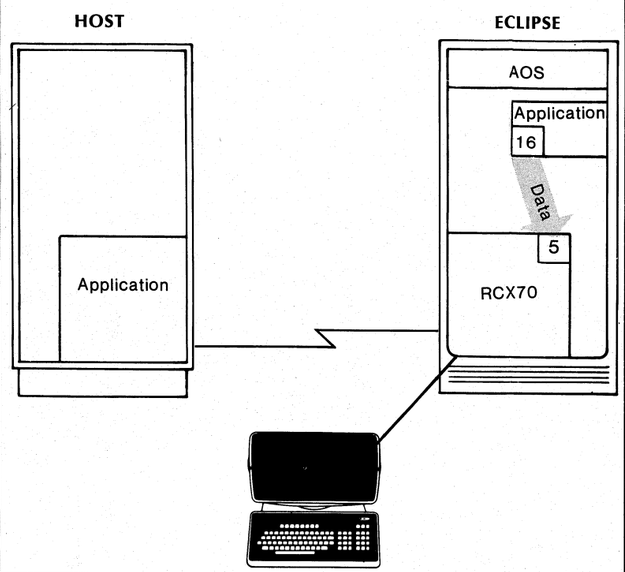
1. User hits ENTER; keyboard locks.



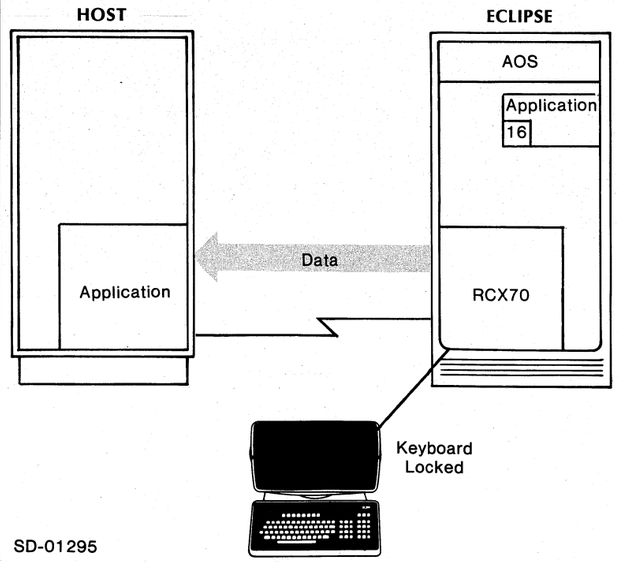
2. RCX70 routes data from keyboard to IPC interface. Local application processes all the data it can.



3. After processing, local application writes data back to RCX70 with synchronous line address.

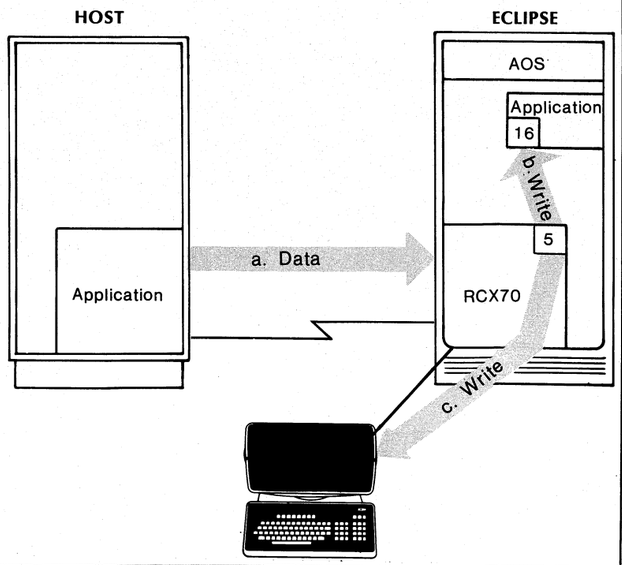


4. RCX70 translates code to EBCDIC and sends data over synchronous line to host application. Host processes data.



SD-01295

5. Host (a) sends data back to RCX70 which translates code to ASCII and (b) writes data back to local application or (c) directly to terminal.



6. If data went to local application, (a) it finishes processing data and writes back to terminal via RCX70 with WCC included. (b)WCC unlocks keyboard.

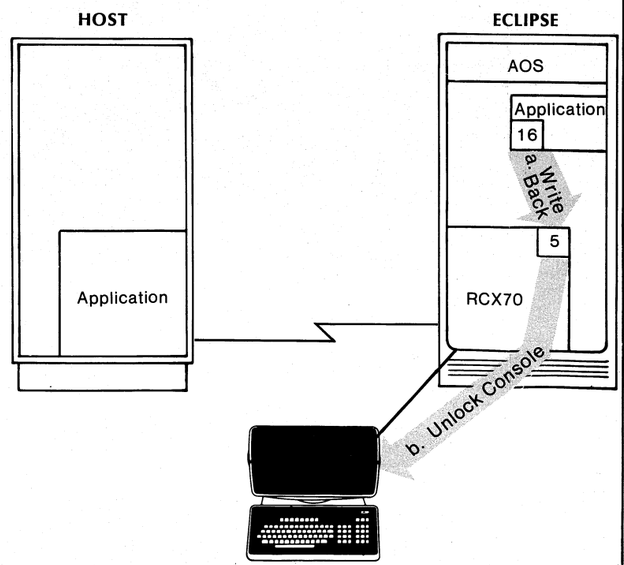
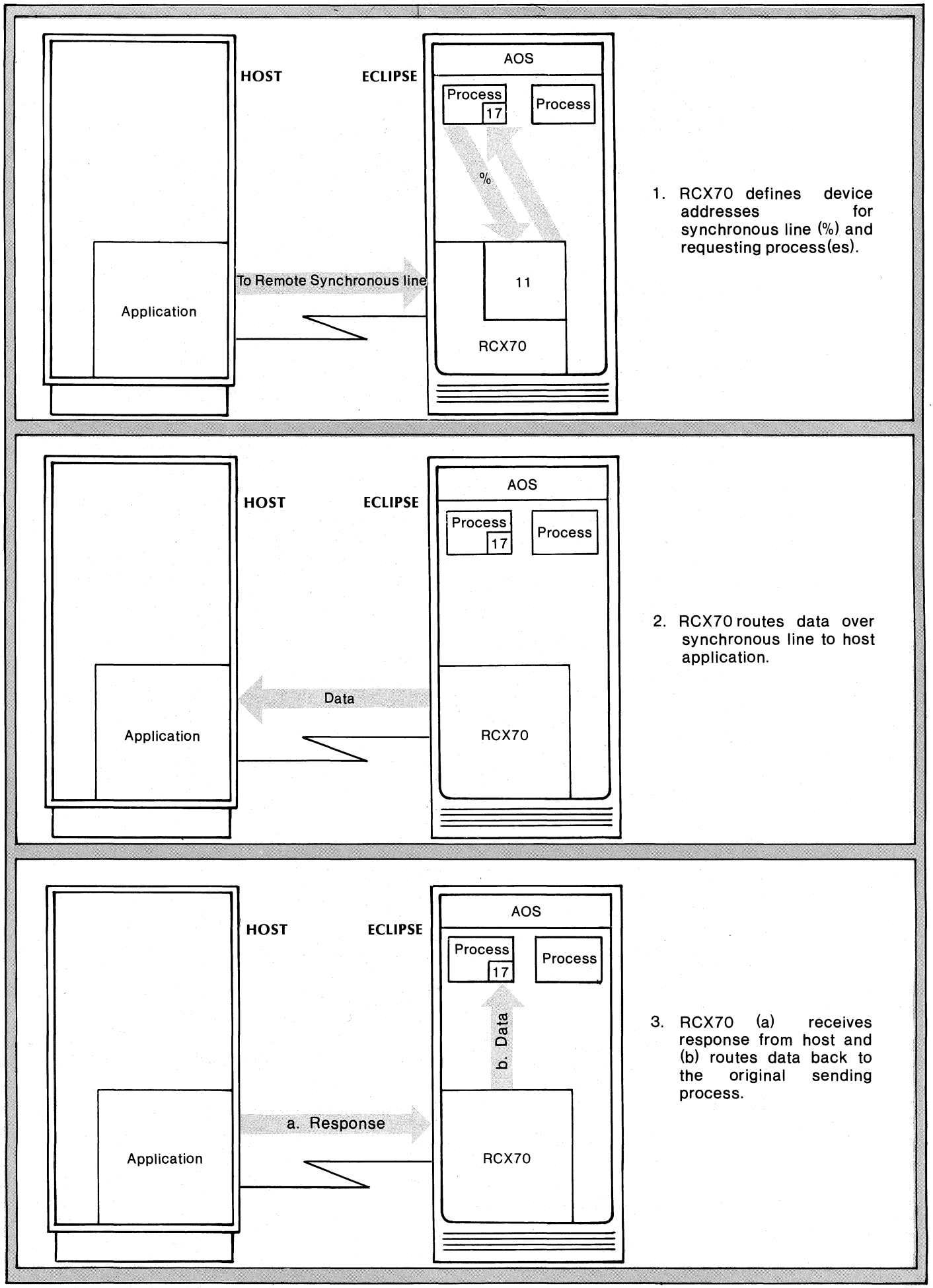


Figure 6-2. Distributed Processing



SD-01296

Figure 6-3. RCX70 as a Synchronous Line Handler

Write and Erase/Write Commands

The IPC or synchronous-line interface sends RCX70 the Write and Erase/Write commands. We discuss Write and Erase/Write in detail in Chapter 3. We include information specific to the dual interface here, but you should read the appropriate sections of Chapter 3 carefully.

When RCX70 receives a Write command, it writes data to a terminal or printer, or to a process via the IPC mechanism, as addressed by specific Write orders. The Erase/Write command differs only in that RCX70 erases the screen before it begins writing.

Figure 6-4 shows the format of the Write and Erase/Write commands.

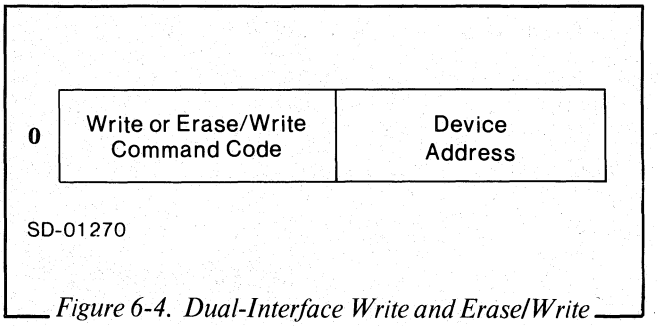


Figure 6-4. Dual-Interface Write and Erase/Write Commands

Command Codes

Command	ASCII Code
Write	061
Erase/Write	065

Device Address

Table 6-1 lists the device numbers and corresponding ASCII addresses for each unit.

You can perform a Broadcast command over the IPC interface by using the general address code (22). A Broadcast message will go only to those terminals communicating over the IPC interface. For details on the Broadcast command, see Chapter 4.

Reader, Please Note

If an error occurs, RCX70 will write back the entire message with one or more of the error bits set in the IPC user flags word. Table 5-1 shows the format of the user flags word.

Table 6-1. Device Addressing

Device Number	EBCDIC I/O Character	ASCII Hex	Device Number	EBCDIC I/O Character	ASCII Hex
0	□	20	33	b	62
1	A	41	34	c	63
2	B	42	35	d	64
3	C	43	36	e	65
4	D	44	37	f	66
5	E	45	38	g	67
6	F	46	39	h	68
7	G	47	40	i	69
8	H	48	41	j	6A
9	I	49	42	k	6B
10	[	5B	43	l	6C
11	.	2E	44	m	6D
12	<	3C	45	n	6E
13	(	28	46	o	6F
14	+	2B	47	p	70
15	!	21	48	q	71
16	&	26	49	r	72
17	J	4A	50	s	73
18	K	4B	51	t	74
19	L	4C	52	u	75
20	M	4D	53	v	76
21	N	4E	54	w	77
22	O	4F	55	x	78
23	P	50	56	y	79
24	Q	51	57	z	7A
25	R	52	58	S	53
26	]	5D	59	T	54
27	$	24	60	U	55
28	*	2A	61	V	56
29	)	29	62	W	57
30	;	3B	63	X	58
31	^	5E	IPC Interface	,	2C
32	a	61	Sync Interface	%	25

Init Terminal Command

RCX70 receives the Init Terminal command from a process that wants to use RCX70 for terminal support. RCX70 gets the message from the IPC interface and issues the Assign command to the AOS PMGR. This designates the addressed terminal as a RCX70 terminal.

The Init Terminal command consists solely of the two bytes shown in Figure 6-5.

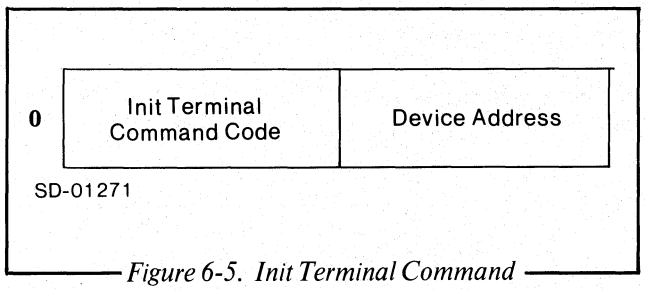


Figure 6-5. Init Terminal Command

Command Code

Command	ASCII Code
Init Terminal	070

Device Address

The device address is the address of the terminal the process wants to assign. See Table 6-1 for all device addresses.

Reader, Please Note

1. If RCX70 cannot assign the device, it writes a command message, INIT TERMINAL FAILURE, to the process. The ?IPTR word of the IPC header will contain the device address of the terminal. There are three reasons why the assign might fail: the terminal was not configured into RCX70, the terminal is currently assigned to another process, or the terminal does not exist on the AOS system.

2. The Init Terminal command never requires a successor block.

Release Terminal Command

RCX70 receives the Release Terminal command from the process that uses RCX70 to control terminals (the local application). RCX70 receives the messages from the IPC interface and issues a Deassign command to the AOS PMGR. This takes terminal support away from RCX70 so that another process can use the terminal.

The Release Terminal command consists solely of the two bytes shown in Figure 6-6.

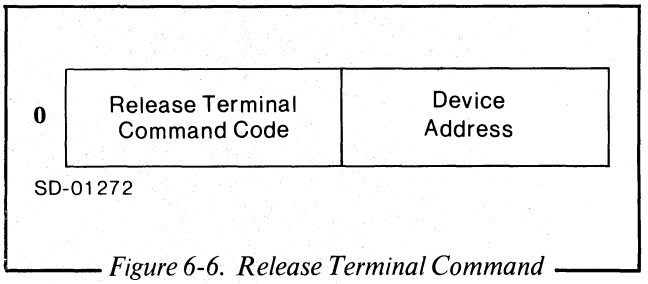


Figure 6-6. Release Terminal Command

Command Code

Command	ASCII Code
Release Terminal	071

Device Address

The device address is the address of the terminal the process wants to deassign. See Table 6-1 for all device addresses.

Reader, Please Note

1. If RCX70 cannot deassign the device, it writes a command message, RELEASE TERMINAL FAILED, to the process. The ?IPTR word of the IPC header contains the device address of the terminal that RCX70 could not deassign.

2. The Release Terminal command never requires a successor block.

IPC Off Command

When a process wants to terminate communication with RCX70, the process issues an IPC Off command. RCX70 maintains a set of tables containing the global port number of each process connected to it via the IPC. The number of tables, established at configuration time, is limited to what was configured at RCX70GEN.

When RCX70 receives an IPC Off command, it clears the table containing the global port number of the process which sent the command. Once the table is cleared, RCX70 can use it for another process that wants to communicate with the IPC. Later, when RCX70 receives a message, the IPC header contains the global port number. RCX70 scans its list of known global ports and adds this one if it does not already exist.

Since each global port requires a table and the tables are limited, your process should issue an IPC Off when it no longer needs RCX70.

The IPC Off command consists solely of the two bytes shown in Figure 6-7.

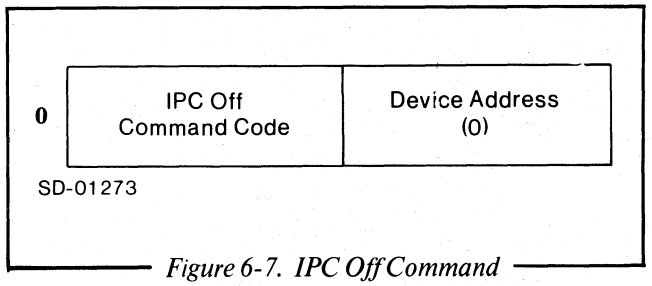


Figure 6-7. IPC Off Command

Command Code

Command	ASCII Code
IPC Off	072

Device Address

IPC Off does not use the device address byte. Therefore, you should put a zero in this byte.

Reader, Please Note

The IPC Off command never requires a successor block.

Disconnect Command

When RCX70 comes up, all data will initially go from keyboards to the local application via the IPC interface. The local application then determines whether further data should go directly to the synchronous line. If the data is addressed to the synchronous line, the application will send RCX70 a Disconnect command over the IPC interface.

When RCX70 receives a Disconnect, it sends all further data from the addressed terminal directly to the synchronous-line interface, instead of the IPC interface. This gives the local application (and you, the programmer) complete control over which application will receive keyboard data.

The Disconnect command consists solely of the two bytes shown in Figure 6-8.

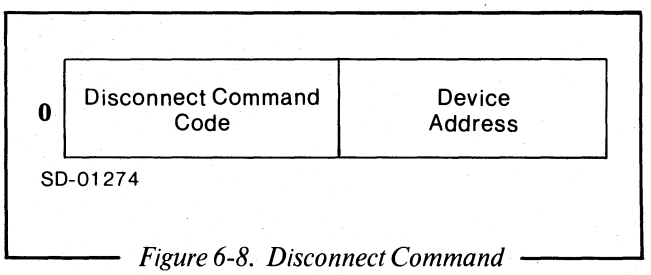


Figure 6-8. Disconnect Command

Command Code

Command	ASCII
Disconnect	073

Device Address

The device address is the address of the terminal the application wants to disconnect. Once the terminal is disconnected, all its input will go directly to the synchronous line.

Reader, Please Note

1. If the application issues a Disconnect to an already disconnected device, RCX70 will ignore it.
2. The Disconnect command never requires a successor block.
3. When it receives a Disconnect, RCX70 changes a flag in its per terminal data base to redirect data to the synchronous line.

RCX70 Dual-Interface Examples

Example 1

Please refer to Figure 6-9 as you read the following text.

Joyce James, who we first met in the Direct 3271 Emulation Mode example, decided to enhance the Footloose Travel Agency's RCX70 system. Instead of running under a purely remote mode of operation, the agency's RCX70 now runs under a dual-interface mode of operation.

Joyce wrote a program that processes all her customer's bills locally. She must still communicate with the synchronous line to obtain information about flights, hotels, and the like, but she no longer needs to go through the tedious paper work involved in updating bills or keeping track of late-paying customers. Joyce found it a simple matter to write the program. Once her local program switched control to RCX70, she did not have to worry about complex synchronous-line protocol. RCX70 did it for her.

Joyce still bootstraps RCX70 in the same fashion as in the purely remote configuration. (Joyce did have to reconfigure RCX70 to run under a dual-interface mode of operation.) Once AOS is running, she disables CON2 with the CLI CONTROL @ EXEC DISABLE command and types

```
PROC/DEF RCX70/TIPC=JOYCE_APPL.IPC @CON2
```

JOYCE_APPL.IPC is the IPC entry of the application that communicates with RCX70.

RCX70 enables CON2 as a RCX70 terminal. Again, if she wants more than one RCX70 terminal, Joyce must disable the desired terminals from the EXEC and include more arguments in the RCX70 command; e.g.,

```
PROC/DEF RCX70/TIPC=JOYCE_APPL.IPC @CON2
@CON3 @CON4 ...
```

Next, Joyce brings up her local application by typing:

```
PROC JOYCE_APPL
```

Her application creates an IPC file called JOYCE_APPL.IPC.

CON2 is now exclusively opened by RCX70. The cursor appears in the upper, left corner of the screen.

The first thing Joyce decides to do this morning is get some old business out of the way. She wants to contact all the agency's customers who have overdue bills. Joyce knows that she can get the information she needs by calling for the general billing function. She types BILLING and strikes the ENTER key.

RCX70 sends the data over the IPC interface to the local application. The application responds by sending an Erase/Write to CON1 asking Joyce which billing function she wants to access: LOCAL UPDATE, REMOTE UPDATE, or OVERDUE. Joyce types OVERDUE and hits ENTER. The application determines that overdue bills are a local function, searches the disk file, then writes the names, phone numbers, and bill amounts of Footloose Travel customers who are tardy in their payments. Joyce calls each customer and presses them for payment.

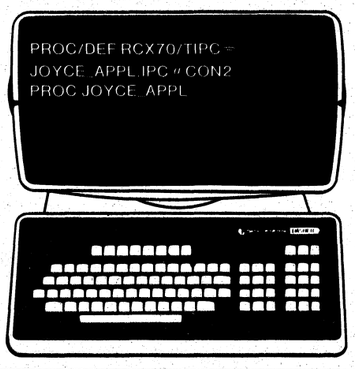
Joyce finishes calling her late-paying accounts just as the first customer enters the agency. He is a regular who states that he "heard something about a crazy kind of round trip fare from New York to London." He requests a departure date of October 1 and Joyce goes back to her terminal. She keys in AIRLINE RESERVATION and strikes ENTER. The data goes to the application, but this data cannot be processed locally.

The local application issues a Disconnect command, which ensures that further input from Joyce's terminal will go directly to the synchronous line. The local application also sends a message to the synchronous line via RCX70, telling the remote application that Joyce is working on an airline reservation. The remote application sends an Erase/Write back to CON2, asking Joyce for the flight number, departure date, and customer's name. Joyce keys in the information and presses ENTER. When the remote application receives the data, it checks its files, determines that a seat is available on the desired flight, and sends the news (via an Erase/Write) back to RCX70 and CON2.

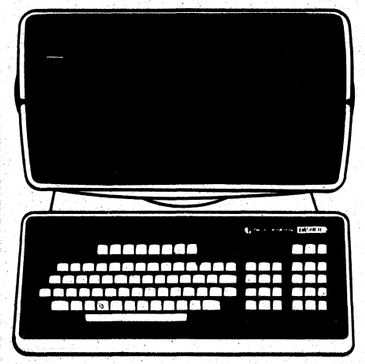
Once the reservation is confirmed, Joyce decides to update the bill. She hits the TERMINATE key so that all further input will go to the local application. Joyce types BILLING again, but this time she asks for a remote update. The data goes first to the local application which asks Joyce the customer's name and flight number. Once it receives that information, the application sends a flight price query over the synchronous line.

The remote application processes the data and sends the price information over the synchronous line to RCX70. RCX70 routes the data back to the local application where the bill is updated. Then the local application performs an Erase/Write to CON2 with the new billing information.

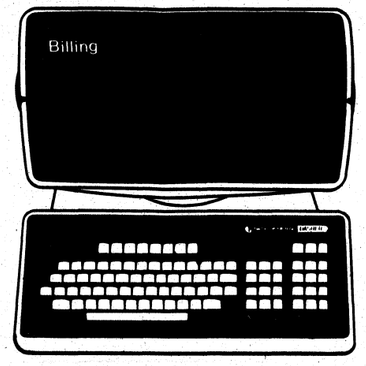
1. Joyce boots RCX70 from CLI.



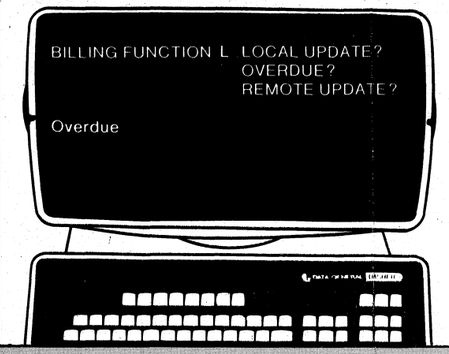
2. CON2 enabled as RCX70 terminal.



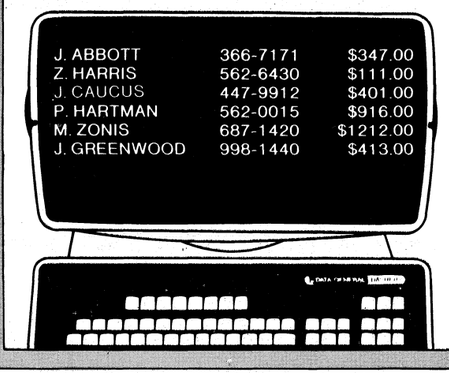
3. Joyce types in general function (Billing) and hits ENTER.



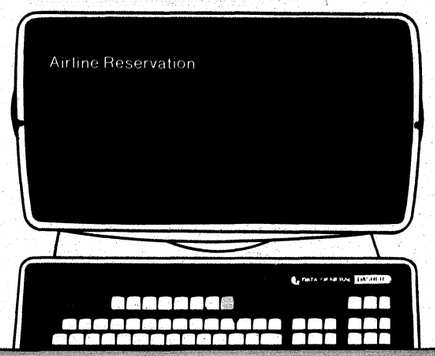
4. RCX70 performs ERASE/WRITE on CON2 as per directions from local application asking for function. Joyce types Overdue and hits ENTER.



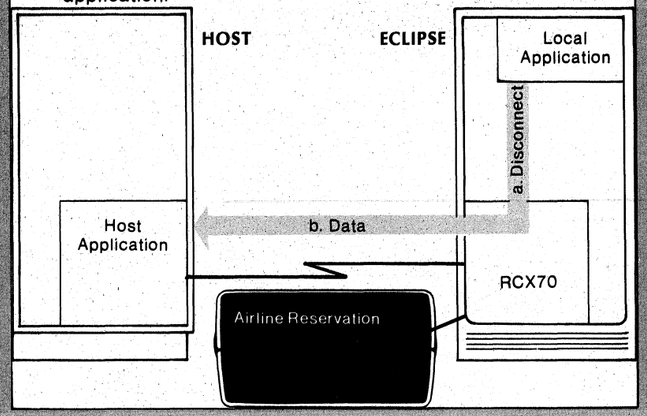
5. Overdue is a local function. Local application sends Erase/Write for all customers with overdue bills.



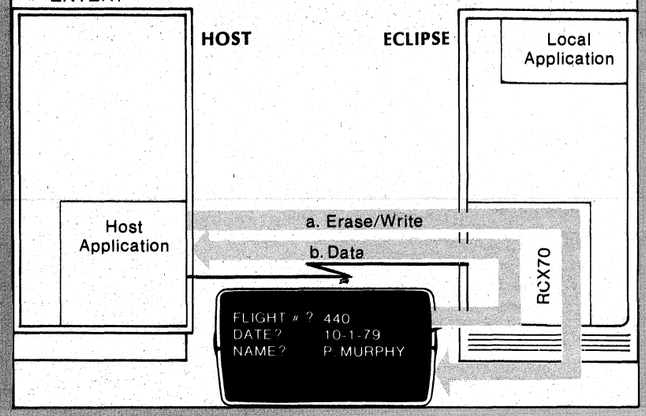
6. Joyce types in Airline Reservation to determine flight availability and hits ENTER.



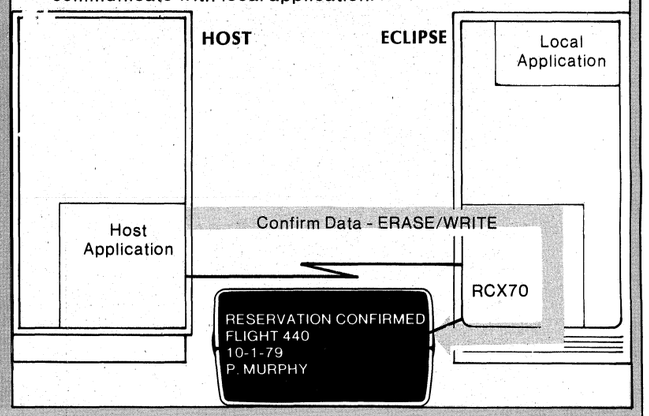
7. (a) Local application cannot do Airline. It issues disconnect and (b) sends all data to synchronous line and host application.



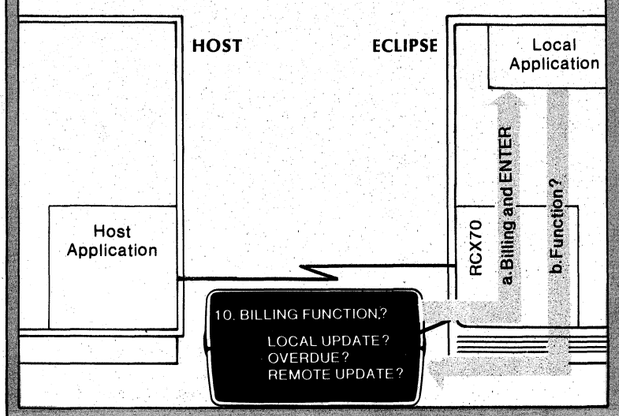
8. (a) Host application sends Erase/Write back to CON2 asking for more data. (b) Joyce keys in requested data and hits ENTER.



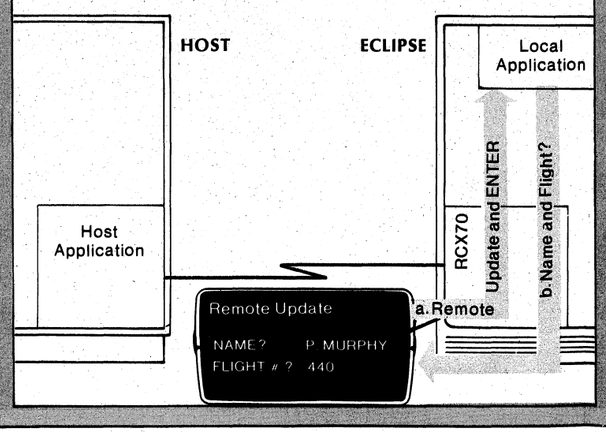
9. Host application confirms reservation and sends data back to RCX70 and CON2. Joyce presses the TERMINATE key to communicate with local application.



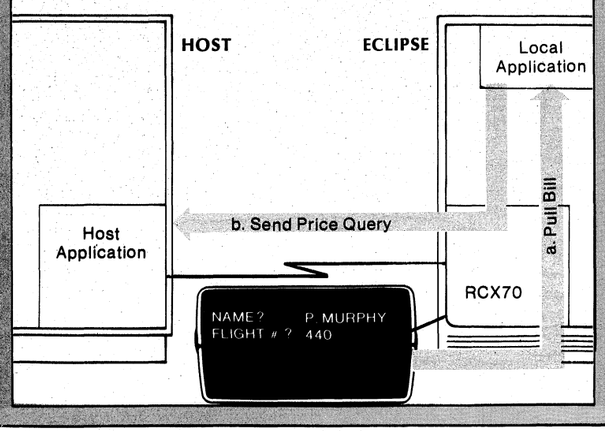
10. (a) Joyce types Billing and hits ENTER. (b) Local application asks for function.



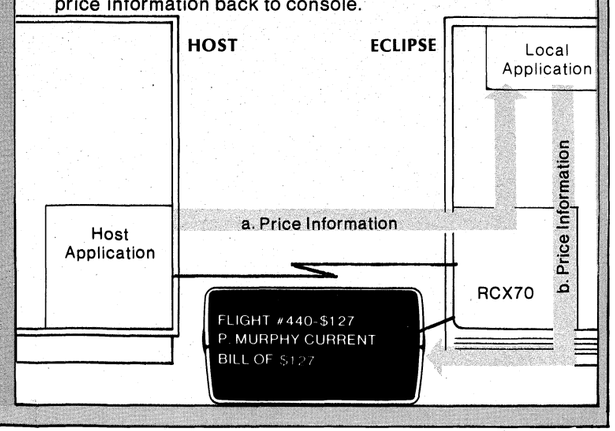
11. (a) Joyce types Remote Update and hits ENTER. (b) Local application asks for Name and Flight.



12. Local application (a) pulls current bill, then (b) sends price query for Flight #440 over synchronous line to host.



13. (a) Host sends price information over synchronous-line to local application. (b) Local updates bill and sends price information back to console.



SD-01285

Example 2

In order to present the details of sending a message from a terminal to a local application to a host application and back, we'll follow the data as it moves along that route.

In this example, we generated RCX70 with a Control Unit Address (CUA) of V. We gave the terminal in this example the device address Q. The terminal has a simple format; it has just one field. The field is unprotected and starts at the upper, left corner of the screen. The terminal is still in its initial condition; i.e., it is set to send to the local application.

The terminal user types ABC, then strikes ENTER. RCX70 determines that the data should go to the local application, so it sends an IPC message there. The user flags word sent with the message is

140011

Bit 0 (the first 1) tells the local application that this is the first block of the message. Bit 1 (the 4) tells the application that this is also the last block of the message. The last two bits (11) contain the actual length of the message byte (eleven octal message bytes).

Table 6-2 describes the format of the message.

Table 6-2. IPC Message Format - Local Application

Header	Bytes	Definition
1	V	RCX70 Select Address.
2	Q	Terminal's device address.

Message	Bytes	Definition
1	‘	The AID byte, ENTER, is a single quote.
2	□	High-order cursor address. Blank is 0.
3	C	Low-order cursor address. C is 3.
4	21	Octal 21 is an SBA order.
5	□	High-order part of attribute address plus one.
6	A	Low-order part of attribute address plus one.
7	A	First character of typed data.
10	B	Second character of typed data.
11	C	Third character of typed data.

Let's make this example even simpler. Assume that the local application doesn't do anything to the data. It sends an IPC to RCX70 with the message to relay over the synchronous line. The user flags word remains 140011.

When the application sends the message to the synchronous line, it changes the message's header byte. Table 6-3 illustrates how it now reads. The message byte remains the same. RCX70 does not look at anything after the %, so the local application could insert anything it wanted after the synchronous-line address.

RCX70 recognizes the % as the synchronous-line device code and sends the message in Table 6-4 in response to the next poll.

Table 6-3. Synchronous-Line Header Format.

Header	Bytes	Definition
0	1	Character 1 is a Write command.
1	%	Device address of synchronous line.

Table 6-4. RCX70's Response to Poll - Synchronous Line

Byte	Character	Meaning
0	STX	Start of Text character.
1	E	Poll address for controller 5, corresponds to V.
2	,	Comma is the device address corresponding to the local application.
3	‘	AID
4 5	□ C	Cursor address.
6	21	SBA
7 10	□ A	Address of attribute plus one.
11 12 13	A B C	Typed characters.
14	ETX	End of Text Character.

The receiving application sees , (comma) as the originating device. After doing whatever computation is required, it responds with a Select. Table 6-5 details the Select.

RCX70 recognizes that device , (comma) is the local application. It sends the message (via the IPC) to the local application process. The user flags word is

140007

Again, the first two bits tell the local application that this is the first and last block of the message. The last two bits contain the actual message length (seven octal bytes).

Table 6-6 shows the format of the message.

Table 6-5. Select Format

First it selects	
Character	**Meaning**
EOT	The first character in a select is always an EOT.
V	Select address of this RCX70
V	(always sent twice).
,	Selected device
,	(always sent twice).
ENQ	
Then it sends	
Character	**Meaning**
STX	
ESC	
1	Character 1 is the Write command.
C	Write Control Character (WCC). It will restore keyboard and clear the MDT bit.
21	An SBA order.
□	High-order address equals 0.
D	Low-order address equals 4.
D	First response character.
E	Second response character.
F	Last response character.

Let's assume, again, that the local application simply sends the message to the terminal (via an IPC to RCX70). The user flags word remains 140007. Table 6-7 contains the format of the message to the terminal.

The resulting message on the first line of the user's terminal will be

□ *ABCDEF*

Table 6-6. Format of Message From Host to RCX70 to Local

Header Bytes		Meaning
1	V	RCX70 Select address.
2	%	Synchronous-line device address.
Message Bytes		**Meaning**
1	C	WCC byte - restore keyboard and clear MDT bits.
2	21	SBA
3	□	High-order cursor address equals 0.
4	D	Low-order cursor address equals 4.
5	D	First character of message.
6	E	Second character of message.
7	F	Last character of message.

Table 6-7. Message Format - Local Application to Terminal

Header Bytes		Meaning
1	1	The character 1 is a Write command.
2	Q	Device address of terminal.
Message Bytes		**Meaning**
1	C	WCC byte - restores keyboard and clears MDT bits.
2	21	SBA order.
3	□	
4	D	
5	D	
6	E	
7	F	

Reader, Please Note

This example is a simple one; the local application just passes on the information that the user typed. Remember that RCX70 does not require a process to include AID bytes, cursor addresses, etc. The message can be completely free form as long as it does not contain characters that might be interpreted as protocol characters (EOT, ETX, STX, etc.).

When you design message formats, remember that even though RCX70 will send whatever you want, the receiving software may enforce arbitrary rules about what it can receive. This will vary from installation to installation.

If the local application had inserted the real originating terminal address (Q in this example) somewhere in the message, the remote application could have selected the terminal and replied directly to it. In this example, the local application had to remember that the terminal was device Q in order to route the response back.

End of Chapter

Chapter 7
Printer, Configuration, Initialization, and Operator Commands

RCX70 Printer Information

RCX70 uses the AOS EXEC print-spooling features to provide hardcopy output. Therefore RCX70 can share printers with other AOS users; you do not have to dedicate a printer to RCX70. Another advantage of using AOS spooling is that a host system can transmit messages to the RCX70 printer(s) at synchronous line speed. RCX70 will never have to reject a select sequence and tell the host system that the printer is not available. The printer may actually be busy (or even out of paper), but RCX70 will continue to accept and spool output for it.

RCX70GEN allows you to configure one or more print queues for your RCX70 system. However, it does not require that you identify each print queue you configure with an actual physical line printer. You must later set up the correspondence with EXEC commands. Therefore, you can change the correspondence without regenerating RCX70. The first print queue you configure will always correspond to a print queue called @RCX70_PRINT00, the second to @RCX70_PRINT01, etc. For example, device number 6 always corresponds to a device address of F whether the device is a print queue or a terminal. If you assign device number 6 to the first print queue you configure, the system will queue any messages addressed to device address F to print queue RCX70_PRINT00. If you assign device number 3 to the second print queue you configure, the system will queue any messages addressed to device address C to RCX70_PRINT01.

Creating Print Queues

After you run RCX70GEN, you must create and prepare the print queues. These commands are exactly like the ones your system manager issued to create and prepare the batch and LPT queues. Note that the OP process must issue these commands.

First type

```
CONTROL @EXEC CREATE PRINT RCX70_PRINT00
```

and (optionally)

```
CONTROL @EXEC CREATE PRINT RCX70_PRINT01
```

Then

```
CONTROL @EXEC OPEN RCX70_PRINT00
```

and (optionally)

```
CONTROL @EXEC OPEN RCX70_PRINT01
```

Next, you must decide the physical printer which will receive specific output. For example, suppose you have two printers: one data channel printer and one programmed I/O printer. These printers are @LPB and @LPA, respectively. You might decide that messages to device address F will go to the data channel printer and those to device address C should go to the PIO printer.

To start the queues on the proper devices, you would type

```
CONTROL @EXEC START RCX70_PRINT00 @LPB
```

and

```
CONTROL @EXEC START RCX70_PRINT01 @LPA
```

Note that you will have to start the queues each time you bring up AOS. Therefore, it is a good idea to put the commands in your UP macro.

If you should remove the PIO printer, you won't have to regenerate RCX70. You just stop the queue and restart it on the @LPB, as follows. (You don't have to bring down RCX70 to do this.)

```
CONTROL @EXEC STOP RCX70_PRINT01
CONTROL @EXEC START RCX70_PRINT01 @LPB
```

RCX70 will spool any messages received while the queue is stopped. Then RCX70 will print them on the @LPB, This means that both queues will print on the same printer. If a second data channel printer is added, you can direct the second queue to it as follows:

```
CONTROL @EXEC STOP RCX70_PRINT01
CONTROL @EXEC START RCX70_PRINT01 @LPB1
```

Note here that you will have to regenerate AOS for the added printer. You don't have to regenerate RCX70.

If you have enough printer output to justify two printers, but don't care which printer a given message is printed on, you can start a single queue on two printers. You just generate one print queue in RCX70. This print queue will correspond to RCX70_PRINT00. Then you issue the following commands:

```
CONTROL @EXEC CREATE PRINT RCX70_PRINT00
CONTROL @EXEC OPEN RCX70_PRINT00
CONTROL @EXEC START RCX70_PRINT00 @LPB
CONTROL @EXEC START RCX70_PRINT00 @LPB1
```

You should note that EXEC will support a maximum of 32 queues, including all of the batch queues and the LPT queue. This limits the number of print queues that you can generate for RCX70.

Spooling Print Output to Output Devices

EXEC allows you to spool print output to any type of output device; you are not limited to line printers. You could spool to a terminal by starting the queue with the following command:

```
CONTROL @EXEC START RCX70_PRINT00 @CONx
```

This also means that you can use a hardcopy DASHER for RCX70 output. To use an LP2 printer, start the queue with

```
CONTROL @EXEC START RCX70_PRINT00 @LPC
```

For more information on queue management, see the *AOS System Manager's Guide.*

Configuring Your RCX70 Display System

Data General provides your RCX70 installation with a configurator program called RCX70GEN. This program will produce a configuration file to meet all your requirements, whether you are running under a synchronous line, IPC, or dual-interface configuration, with or without terminals. You will define the character set, as well as which characters will be legal for any given field.

For your convenience, we have numbered each RCX70GEN query and listed your response options. Keep in mind that the actual RCX70GEN questions are not numbered.

To invoke the generation program you must first type

```
XEQ RCX70GEN
```

from the AOS CLI.

RCX70GEN creates an output file called RCX70.CONFIG and deletes the previous RCX70.CONFIG file from your working directory, if it exists. If you want to save the previous configuration file, you must rename it with the CLI Rename command. Your responses to all of the following questions will be incorporated in RCX70.CONFIG.

RCX70GEN displays

1. *NUMBER OF DEVICES CONNECTED TO THIS STATION?*

   You must supply a number in the range 2 to 66. Devices include all terminals, printers, IPC interfaces, and synchronous-line interfaces. Table 4-5 defines device addresses. The terminal IPC interface is always device address , (comma) and the synchronous-line device address is always % (per cent). You can assign terminals, printers, and IPC interfaces that won't send to terminals to any of the sixty-four device addresses. RCX70GEN allows you to pick a device address for each of these devices. All you need to do is specify a device number (from the left-most column of Table 4-5). *You can configure a maximum of 16 terminals.* The maximum total for all devices including terminals, printers, and IPC interfaces that don't send to terminals, the synchronous line, and the terminal IPC interface is sixty-six. If you specify a number outside the legal range, the message

   NUMBER OF DEVICES MUST BE BETWEEN 2 AND 66. TRY AGAIN.

   appears on your screen and RCX70GEN repeats question 1.

2. *DO YOU WANT A SYNCHRONOUS LINE CONFIGURED?*

   Answer Yes or No.

3. *IS THERE AN IPC INTERFACE THAT WILL SEND TO TERMINALS?*

   Answer Yes (see Chapter 5) or No.

4. *HIGHEST DEVICE NUMBER?*

   You must supply a number in the range 0 to 63. You do not include numbers here for the synchronous-line or IPC interface; RCX70GEN does it for you. You do specify numbers for every terminal, printer, and nonterminal IPC that you want configured into the system. If you specify a number that is outside the legal range, the message

   DEVICE RANGE IS 0 TO 63! TRY AGAIN

   appears on your screen and RCX70GEN repeats question 4.

5. *LOWEST DEVICE NUMBER?*

   Again, the range is 0 to 63. If you specify a low device number which is higher than the highest device number, RCX70GEN repeats question 4. Don't include numbers here for the synchronous-line or IPC interface; terminals, printers, and nonterminal IPCs only, please.

6. RCX70 asks this question only if you requested a synchronous-line configuration in question 2.

   ARE YOU TRANSMITTING IN EBCDIC?

   Answer Yes if your synchronous line will transmit in EBCDIC or No if it will transmit in ASCII.

RCX70 asks questions 7 through 12 only if you requested a synchronous-line configuration in question 2.

7. *IS THE SYNCHRONOUS LINE ODD PARITY?*

   Answer Yes or No.
   If you answer Yes, go to question 9.
   If you answer No, RCX70 asks

8. *IS THE SYNCHRONOUS LINE EVEN PARITY?*

   Answer Yes or No.
   If you answer No to questions 7 and 8, RCX70 sends the message:

   USING NO PARITY ON SYNCHRONOUS LINE.

9. *IS THE BCC CHARACTER CRC16?*

   You would normally use CRC16 with EBCDIC lines.
   Answer Yes or No.
   If you answer Yes, go to question 12.
   If you answer No, RCX70GEN asks

10. *IS THE BCC CHARACTER CCITT16?*

    CCITT16 is the European equivalent of CRC16.
    Answer Yes or No.
    If you answer Yes, go to question 12.
    If you answer No, RCX70GEN asks

11. *IS THE BCC CHARACTER LRC?*

    You would normally use LRC with ASCII lines.
    Answer Yes or No.
    If you answer No to questions 9, 10, and 11 RCX70GEN returns

    USING NO BCC CHARACTERS.

12. *DO YOU HAVE MULTIPLE PRINTERS ATTACHED TO THIS STATION?*

    Answer Yes or No.

13. *USE COMMA INSTEAD OF PERIOD IN NUMERIC FIELD?*

    Answer Yes or No. You should answer No unless you are using a European number scheme.

14. RCX70 asks this question only if you requested a synchronous-line configuration in question 2.

    CONTROL UNIT NUMBER?

    This is the control unit number of the actual RCX70 you are running. As long as nobody else is using the control unit number, you may choose any of the 32 device addresses from Table 4-2. The number you choose must be in the range 0 to 31.

15. RCX70 asks this question only if you requested a synchronous-line configuration in question 2.

    AOS SYNCHRONOUS LINE NUMBER?

    The number you choose must be in the range 0 to 31. When your System Manager generated AOS, (s)he defined the synchronous-line numbers. You must choose one of the defined numbers.

16. RCX70 asks this question only if you requested a synchronous-line configuration in question 2.

    MAXIMUM SIZE OF SYNCHRONOUS LINE TRANSMISSION IN BYTES?

    The number must be in the range 1 to 2048. The maximum byte size of the synchronous-line transmission should be equal to or larger than the maximum byte size of the largest message sent by the host application.

17. *USE THE DEFAULT EBCDIC TO ASCII TABLE?*

    Answer Yes or No.
    If you answer Yes, the table, which we include in the RCX70GEN package, is called E_A_TABLE.PR.
    You should answer No only if you do not have a standard ASCII keyboard..

    If you answer No, RCX70GEN displays

    TYPE THE FILENAME OF THE EBCDIC TO ASCII TABLE.

    You must have already configured the optional EBCDIC to ASCII table. Appendix C tells you how to do this. Type the name of the file.

    If RCX70GEN detects an error in the default or optional table, it returns the message

    filename IS xxx LONG PRESENTLY. THE LENGTH OF THE FILE MUST BE 256 BYTES.

    If the error is in the default table, RCX70GEN returns the message

    RCX70GEN ABORTED!

    and terminates. This will happen only if the default EBCDIC to ASCII table has been altered or destroyed. In this case, you will have to reload the default table.

18. *USE THE DEFAULT ASCII TO EBCDIC TABLE?*

    Answer Yes or No.
    If you answer Yes, the table, which we include in the RCX70GEN package is called A_E_TABLE.PR.
    You should answer No only if you do not have a standard ASCII keyboard.

    If you answer No, RCX70GEN displays

    TYPE THE FILENAME OF THE ASCII TO EBCDIC TABLE.

    You must have already configured the optional ASCII or EBCDIC table; Appendix C tells you how to configure the table. Type the name of the table.

    If RCX70GEN detects an error in the default or optional table, it returns the message

    filename IS xxx LONG PRESENTLY. THE LENGTH OF THE FILE MUST BE 256 BYTES.

19. *USE THE DEFAULT ALPHABETIC FIELD DEFINITION?*

    Answer Yes or No.
    If you answer Yes, the file is called D_ALPHABETIC_F.PR. The default file includes all alphabetic characters (A-Z, upper- and lowercase).

    If you answered Yes to question 14, go now to question 27.
    If you answered No, RCX70GEN asks

20. *DO YOU HAVE AN ALPHABETIC FIELD DEFINITION FILE?*

    If you have already created a file, answer Yes. RCX70GEN returns

    FILENAME?

    Type in the name of the alphabetic field definition file you created. RCX70GEN scans your file. If it detects an error, it returns the message

    YOUR SPECIFIED ALPHABETIC FIELD IS xxx BYTES LONG.
    THE LENGTH MUST BE 32 BYTES LONG.

    Then you must modify your file so that it is 32 bytes long.

    If you have not created your own alphabetic field definition file, answer No. RCX70GEN then asks

21. *DO YOU WANT ALL OF A-Z IN THE TABLE?*

Answer Yes or No.
If you answer Yes, go to number 23.
If you answer No, RCX70GEN asks

22. *DO YOU WANT ANY OF A-Z IN THE TABLE?*

Answer Yes or No.
If you answer No, go to number 23.
If you answer Yes, RCX70GEN asks about each alphabetic character individually. Give your desired response to each character query; for example

DO YOU WANT "A" IN THE TABLE?
DO YOU WANT "B" IN THE TABLE?
.
.
.

When RCX70GEN finishes the individual alphabetic queries, it asks

23. *DO YOU WANT ALL OF a-z IN THE TABLE?*

and repeats the procedure outlined in question 22. RCX70GEN then displays

24. *DO YOU WANT ALL OF 0-9 IN THE TABLE?*

and repeats the procedure outlined in question 22 for each of the numerals. Next, RCX70GEN displays

25. *DO YOU WANT ALL OF THE SPECIAL CHARACTERS IN THE TABLE?*

The special characters are:

(Space)	/
!	:
"	;
#	<
$	=
%	>
&	?
'	@
(	[
)	\
*	]
+	^
,	-
-	.

If you want all the special characters, answer Yes to question 25.
If you answer No, RCX70GEN will follow the procedure described in question 22, asking for each special character individually.

26. *DO YOU WANT TO OUTPUT THE TABLE TO A FILE FOR FUTURE USE?*

Answer Yes or No.
If you answer Yes, RCX70GEN will ask you for a FILENAME. Key in the filename and use this table for future RCX70GENs.

27. *USE THE DEFAULT NUMERIC FIELD DEFINITION?*

Answer Yes or No.
If you answer Yes, the file is called D_NUMERIC_F.PR. The default file includes the numbers 0 through 9, the period, and the minus sign.
If you answered Yes to question 27, go now to question 35.
If you answered No, RCX70GEN asks

28. *DO YOU HAVE A NUMERIC FIELD DEFINITION FILE?*

If you have already created a numeric file, answer Yes. RCX70GEN returns

FILENAME?

Type in the name of the numeric field definition file that you created. RCX70GEN scans your file. If it detects an error, it returns the message

YOUR SPECIFIED NUMERIC FIELD IS xxx BYTES LONG.
THE LENGTH MUST BE 32 BYTES LONG.

Then you must modify your file so that it is 32 bytes long.

If you have not created your own numeric field definition file, answer No. RCX70GEN then asks

29. *DO YOU WANT ALL OF A-Z IN THE TABLE?*

Answer Yes or No.
If you answer Yes, go to question 31.
If you answer No, RCX70GEN asks

30. *DO YOU WANT ANY OF A-Z IN THE TABLE?*

Answer Yes or No.
If you answer No, go to question 31.
If you answer Yes, RCX70GEN asks about each alphabetic character individually. Give your desired response to each character query (see question 22 for an example).

31. *DO YOU WANT ALL OF a-z IN THE TABLE?*

Answer Yes or No. RCX70GEN repeats the procedure outlined in question 30.

32. *DO YOU WANT ALL OF 0-9 IN THE TABLE?*

Answer Yes or No. Again, RCX70GEN repeats the procedure described in question 30.

33. *DO YOU WANT ALL OF THE SPECIAL CHARACTERS IN THE TABLE?*

We list all of the special characters at question 25. Answer Yes to question 33 if you want all of the special characters in the numeric file. If you answer No, RCX70GEN will follow the procedure in question 22, asking you for each special character individually.

34. *DO YOU WANT TO OUTPUT THE TABLE TO A FILE FOR FUTURE USE?*

Answer Yes or No.
If you answer Yes, RCX70GEN will ask you for a FILENAME. Key in a filename and use this table for future RCX70GENs.

35. *USE THE DEFAULT SPECIAL CHARACTER FIELD DEFINITION?*

Answer Yes or No. RCX70GEN follows the same procedures outlined in questions 19 through 26, substituting *SPECIAL CHARACTER* for *ALPHABETIC*. The name of the default special character file is D_SPECIAL_F.PR.

NOTE: Questions 36 through 44 ask you about the devices connected to your RCX70 installation. RCX70GEN will repeat this set of questions for as many devices (except synchronous-line and IPC interfaces) as you specified in Question 1. For example, if you have three terminals and a line printer attached to your system, RCX70GEN will loop through the device questions four times.

*******PLEASE CONFIGURE A DEVICE*******

36. *IS THIS DEVICE A TERMINAL?*

Answer Yes or No.
If you answer No, go to question 41.
If you answer Yes, RCX70GEN asks

37. *WHAT IS THE DEVICE NUMBER?*

Key in the device number of the terminal. If you don't already know the number, ask your system manager. The number must fall within the range you specified for the highest and lowest device numbers in questions 4 and 5. If it is outside that range, RCX70GEN will return the message

THE DEVICE NUMBER MUST BE BETWEEN < lowest > AND < highest >.
ENTER THE NUMBER AGAIN.

38. *WHAT IS THE AOS CONSOLE NUMBER?*

Enter the AOS console number. It must be between 1 and 255. If you don't know the number, ask your system manager.

NOTE: Normally, AOS console numbers begin with @CON (i.e., @CON2, @CON3, etc.). All you need to enter here is the number itself.

39. *IS THE TERMINAL A 6053?*

Answer Yes or No.
If you answer No, RCX70GEN loops back to question 36 unless you've reached the number of devices you specified in Question 1.
If you answer Yes, RCX70GEN asks

40. *DO YOU WANT LOWERCASE INPUT DISPLAYED AS UPPERCASE OUTPUT?*

Answer Yes or No.
RCX70GEN loops back to question 36 after you respond unless you've reached the number of devices you specified in Question 1.

41. RCX70 asks this question only if you responded No to Question 36.

IS THIS DEVICE A PRINTER?

Answer Yes or No.
If you answer No, go to question 44.
If you answer Yes, RCX70GEN asks

42. *WHAT IS THE PRINTER WIDTH?*

Key in the printer width. The number must be in the range 0 to 132 (characters per line). If you're not sure of the correct printer width, see your system manager.

43. *WHAT IS THE DEVICE NUMBER OF THIS PRINTER?*

    Key in the device number for the printer. If you don't know the number, check with your system manager. The number must fall within the range you specified for the highest and lowest device numbers in questions 4 and 5. If it is outside that range, RCX70GEN will return the message

    THE DEVICE NUMBER MUST BE BETWEEN < lowest > AND < highest >.
    ENTER THE NUMBER AGAIN.

    Once you key in a successful device number, RCX70GEN loops back to question 41.

44. RCX70 asks this question only if you responded No to question 41.

    IS THIS DEVICE AN IPC INTERFACE THAT WILL NEVER SEND TO TERMINALS?

    Answer Yes or No. See Chapter 5 for an explanation of the no terminal option.
    If you answer No, RCX70 loops back to question 36 unless you've reached the number of devices you specified in Question 1.
    If you answer Yes, RCX70 asks

45. *WHAT IS THE DEVICE NUMBER OF THE IPC INTERFACE?*

    Key in the device number for the IPC interface. If you don't know the number, check with your system manager. The number must fall within the range you specified for the highest and lowest device numbers in questions 4 and 5. If it is outside that range, RCX70GEN will return the message.

    THE DEVICE NUMBER MUST BE BETWEEN < lowest > AND < highest >.
    ENTER THE NUMBER AGAIN.

    RCX70GEN will repeat the device questions until it reaches the number you specified in Question 1. It will then return

46. *RCX70GEN DONE!*

    and incorporate all of your responses into RCX70.CONFIG.

You are now ready to initialize RCX70.

Initializing RCX70 from the Operator Console

Before you issue the command to start up RCX70:

- you must have the IPC privilege in your AOS user's profile and
- either you must disable terminals you will assign to RCX70
- or you must know that AOS has already disabled RCX70 terminals.

You use the AOS Disable command to disable the terminals. The format of the command is

```
CONTROL @EXEC DISABLE (@CONa @CONb @CONc
... @CONz)
```

Once you have control of your desired number of terminals, you can call RCX70 by issuing the command

```
XEQ RCX70 /TIPC=filename @CONa @CONb @CONc
... @CONz
```

from the operator's console.

If you do not want to tie up a console as the RCX70 Operator's console, use the following command to proc RCX70 from the AOS CLI.

```
PROC/DEF RCX70 /TIPC=filename @CONA @CONB
@CONC ... @CONZ
```

Command Switch

```
/TIPC=filename
```

This switch is only meaningful if an IPC interface will communicate with your RCX70 terminals. TIPC stands for Terminal IPC. If you include this switch in the command line, RCX70 will direct initial terminal output to the IPC interface instead of to the synchronous-line interface. Filename must be the IPC file on which RCX70 issues an ?ILKUP system call (see "Initial Conditions" in Chapters 5 and 6). The IPC file does not have to exist at the instant that RCX70 is brought up. However, RCX70 will not proceed out of its initialization code until the IPC file exists and the ?ILKUP succeeds.

RCX70 Operator Commands

You use the following operator commands to initialize RCX70 terminals, release RCX70 terminals, obtain statistics about synchronous line usage, and terminate RCX70.

If you used the XEQ RCX70 format, you must issue all commands to RCX70 from the operator console; i.e., the console from which you originally brought up RCX70. You cannot use the RCX70 operator console for anything but RCX70 commands.

If you used the PROC RCX70 format, other terminals can send messages to RCX70, but all responses will go to the terminal from which you issued the PROC command.

Init Terminal Command

Allow an additional terminal to communicate with RCX70.

Format

If you called RCX70 with the XEQ command:

```
INIT @CONn
```

If you called RCX70 with the PROC command:

```
CONTROL RCX70 INIT @CONn
```

Description

When you initialize RCX70, you specify terminals you want connected to the system. The Init Terminal command allows you to add terminals without bringing RCX70 down. It causes a terminal that was not previously communicating with RCX70 to start communicating with it. @CONn is the name of the terminal you want to add.

Reader, Please Note

1. When configuring your RCX70 system, you must specify all the possible terminals that you will ever try to initialize. RCX70 maintains an internal table with a list of the terminals.

2. The AOS EXEC normally assigns all terminals. Before issuing an Init Terminal command, you must issue the EXEC Deassign command from the AOS system operator's console.

3. The terminal you add must not be assigned by another process. If it is, the initialization will fail and RCX70 will return an error message.

Example

```
INIT @CON4

***DEVICE INITIALIZED @ CON4
```

Release Terminal Command

Release a terminal from communicating with RCX70.

Format

If you called RCX70 with the XEQ command:

```
REL @CONn
```

If you called RCX70 with the PROC command:

```
CONTROL RCX70 REL @CONn
```

Description

The Release Terminal command causes a terminal communicating with RCX70 to stop communicating with it. Although you can issue this command at any time, you should be careful not to issue it when a terminal is in the middle of carrying out a previous command.

If the terminal is not communicating with RCX70 when you try to release it, RCX70 will return an error message, but will take no further action.

Example

```
REL @CON7
***RELEASED DEVICE @ CON7
```

Statistics Command

Obtain statistics about synchronous-line usage.

Format

If you called RCX70 with the XEQ command:

```
STAT
```

If you called RCX70 with the PROC command:

```
CONTROL RCX70 STAT
```

Description

Use this command to obtain statistics about synchronous-line usage. The information returned to your screen includes

Total NAKS Received
Total Timeouts
Total Block Check Errors

RCX70 gets the information from the AOS synchronous-line statistics call.

If you issue the Statistics command when your RCX70 system is not configured for a synchronous-line, RCX70 will return an error message.

Example

```
STAT

TOTAL NAKS RECEIVED
     0
TOTAL TIMEOUTS
     0
TOTAL BLOCK CHECK ERRORS
     0
```

BYE Command

Terminate the RCX70 process.

Format

If you called RCX70 with the XEQ command:

```
BYE
```

If you called RCX70 with the PROC command:

```
CONTROL RCX70 BYE
```

Description

You use the Bye command to make an orderly exit from RCX70. RCX70 responds with

DO YOU WANT TO BRING RCX70 DOWN?

If you are using the first format and you really want to bring RCX70 down, type YES.
If you are using the second format and you really want to bring RCX70 down, you must type:

```
CONTROL RCX70 YES
```

You should be careful not to type Bye while RCX70 is carrying out a previous command or order. If you do, the command will bring RCX70 down immediately.

Example

```
BYE
DO YOU WANT TO BRING RCX70 DOWN?  YES
)
```

The right parenthesis is the AOS CLI prompt.

End of Chapter

Appendix A
RCX70 Error Messages

ERROR: ABBREVIATION NOT UNIQUE

The RCX70 operator console accepts minimum unique abbreviations for commands. Your command abbreviation did not reach the minimum number; hence the error message. If you tried to execute two or more commands in one command line, RCX70 prints the error message with the offending command on the following line. RCX70 continues processing after displaying the message.

ERROR: AOS ERROR # x AT DEVICE # n

RCX70 received AOS error number *x* when issuing a system call for console number *n*.

ERROR: CAN'T FIND IPC PORT FOR IPC INTERFACE; TERMINATING RCX70

When you start RCX70 with the /TIPC=filename switch, filename must either already exist or it must be created within five minutes. If filename does not exist after five minutes, RCX70 will print this message and terminate itself.

ERROR: CONSOLE NOT INITIALIZED

The console you are trying to access was not initialized by AOS. RCX70 first prints the AOS error, then this error, and finally, the erroneous console name. RCX70 continues processing after displaying the message.

ERROR: DEVICE ADDRESS x NOT AVAILABLE

You tried to send a message to a device address *(x)* that has not been initialized.

ERROR: DEVICE ALREADY IN USE

You tried to initialize a device that another process is using.

ERROR: DEVICE IS NOT A PRINTER OR CONSOLE

You issued a command to a synchronous line or to an IPC interface that you should only issue to a console or printer.

ERROR: DEVICE IS UNAVAILABLE

RCX70 is unable to communicate with the terminal.

ERROR: DEVICE NOT IN USE

Addressed device has not been initialized.

ERROR IN TERMINATING CLI (PID # x); NO DEVICE TABLE FOUND

If this message appears, the cause might be anything from RCX70 to a hardware problem. Contact your Data General representative.

ERROR IN INITIALIZE COMMAND

You misspelled a console name in the Initialize command line. RCX70 prints the offending console name on the next line and continues processing.

ERROR IN PROCING A CLI AT CONSOLE

An AOS error occurred while you were trying to proc a CLI at an RCX70 terminal. The message appears on the operator console, followed by the console number of the console in error, and, finally, the AOS error message. RCX70 continues processing. Note that the error message will *not* appear on the originating RCX70 console. This error often occurs because RCX70 has not been allocated enough son processes.

ERROR IN RELEASE COMMAND

You misspelled a console name in the Release command line. RCX70 prints the word in error followed by the offending console name and continues processing.

ERROR IN RELEASING A CONSOLE

An AOS error occurred while you were trying to release a console. RCX70 prints the AOS error, the RCX70 error, and the console number of the console in error. RCX70 continues processing.

ERROR ON ?GCHR/?SCHR CALL DURING INITIALIZATION

On initializing a terminal, RCX70 could not set the characteristics. RCX70 goes on to the next terminal.

ERROR ON ?ILKUP CALL DURING INITIALIZATION

You tried to initialize a terminal that was not initialized during AOSGEN. RCX70 goes on to the next terminal.

ERROR RESETTING CHARACTERISTICS AFTER TERMINATING CLI

RCX70 could not reset the characteristics after the CLI changed them. The characteristics remain the same as the CLI's.

ILLEGAL DEVICE TYPE RETURNED FROM ?ILKUP DURING INITIALIZATION

You tried to initialize a terminal that was not initialized during AOSGEN. RCX70 goes on to the next terminal.

INSUFFICIENT DEVICE TABLES CONFIGURED FOR IPC

You forgot to configure your nonterminal IPC interfaces during RCX70GEN. RCX70 prints this message and terminates.

NO DEVICE TABLE FOUND

You didn't configure this device during RCX70GEN. RCX70 prints this message and terminates.

NO SUCH COMMAND

This message appears on the operator's console if you try to input a nonexistent command. RCX70 prints the message with the offending command on the following line and continues processing.

NO SYNCHRONOUS LINE ACTIVE

This message appears if you try to execute the Statistics command without having configured a synchronous line.

PRINTER NOT CONFIGURED

You tried to issue a Copy command from the keyboard to a nonexistent printer. Nothing gets copied and RCX70 continues processing.

UNABLE TO OPEN DEVICE

RCX70 cannot initialize a device. AOS sends the error message on an OPEN call.

UNKNOWN COMMAND FROM IPC INTERFACE

When you are using an IPC interface that talks to terminals and the interface sends an unknown command, RCX70 prints this message. RCX70 then sends the offending message back over the IPC interface with error bits set.

UNKNOWN DEVICE ADDRESS FROM IPC INTERFACE

When the IPC interface that writes to consoles tries to address a device that has not been configured, RCX70 displays this message. RCX70 then sends the offending block back over the IPC interface.

WRITE ERROR ON @ CONn

An AOS write error occurred on the named console.

End of Appendix

Appendix C
Creating EBCDIC-to-ASCII or ASCII-to-EBCDIC Optional Files for RCX70GEN

Step One -- Insert the Numbers

If you do not choose the default EBCDIC-to-ASCII or ASCII-to-EBCDIC tables in questions 12 and 13 of RCX70GEN, you will have to create separate files to generate the tables. To create an EBCDIC-to-ASCII or ASCII-to-EBCDIC file for RCX70, you must first have hard-copy tables of the EBCDIC and ASCII codes that you will need. (We assume that if you are configuring your own table, you do not have a standard ASCII keyboard. Therefore, the EBCDIC and ASCII codes that you will need will be foreign codes.)

The EBCDIC-to-ASCII table is 128 words long, packed two characters per word:

Word 0	0	1
Word 1	.	3
Word 2	4	5

To fill in the bytes, you use one of the text editors to create a MASM source file containing the table. Then, you must take each EBCDIC character and find the corresponding character in the ASCII table. Find the ASCII octal or hexadecimal character(s) and insert that number into the proper position in the EBCDIC table. In the following example, Tables 2-1 and 2-2 supply the hexadecimal numbers. Again, the table you use will have different octal and hex codes.

Looking at Table 2-1, we see that the EBCDIC hexadecimal code for the Program Tab (PT) is 5. The corresponding ASCII hex code (from Table 2-2) shows PT as 9. All we must do is insert a 9 in position 5 (the fifth byte) of the EBCDIC table.

The ASCII-to-EBCDIC table is 64 words long, packed two characters per word. To create an ASCII-to-EBCDIC table, you simply reverse the procedure for creating an EBCDIC-to-ASCII table.

Step Two -- Assemble the File

Once you have produced a file with your table in it, you must execute the AOS Macroassembler (MASM) over the filename.

```
XEQ MASM filename
```

MASM produces an object file, filename.OB.

Step Three -- Bind the File

Issue the command

```
XEQ BIND/I filename
```

AOS produces an output file, filename.PR. Your file is now complete.

You must supply the names of these files (filename.PR) in questions 12 and 13 of the RCX70GEN dialog.

End of Appendix

Appendix B
EBCDIC and ASCII
Address Representations

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
01	01	0000	0000	40	40	□	□	20	20	□	□
01	02	0001	0001	40	C1	□	A	20	41	□	A
01	03	0002	0002	40	C2	□	B	20	42	□	B
01	04	0003	0003	40	C3	□	C	20	43	□	C
01	05	0004	0004	40	C4	□	D	20	44	□	D
01	06	0005	0005	40	C5	□	E	20	45	□	E
01	07	0006	0006	40	C6	□	F	20	46	□	F
01	08	0007	0007	40	C7	□	G	20	47	□	G
01	09	0008	0008	40	C8	□	H	20	48	□	H
01	10	0009	0009	40	C9	□	I	20	49	□	I
01	11	0010	000A	40	4A	□	¢	20	5B	□	[
01	12	0011	000B	40	4B	□	.	20	2E	□	.
01	13	0012	000C	40	4C	□	<	20	3C	□	<
01	14	0013	000D	40	4D	□	(	20	28	□	(
01	15	0014	000E	40	4E	□	+	20	2B	□	+
01	16	0015	000F	40	4F	□	|	20	21	□	!
01	17	0016	0010	40	50	□	&	20	26	□	&
01	18	0017	0011	40	D1	□	J	20	4A	□	J
01	19	0018	0012	40	D2	□	K	20	4B	□	K
01	20	0019	0013	40	D3	□	L	20	4C	□	L
01	21	0020	0014	40	D4	□	M	20	4D	□	M
01	22	0021	0015	40	D5	□	N	20	4E	□	N
01	23	0022	0016	40	D6	□	O	20	4F	□	O
01	24	0023	0017	40	D7	□	P	20	50	□	P
01	25	0024	0018	40	D8	□	Q	20	51	□	Q
01	26	0025	0019	40	D9	□	R	20	52	□	R
01	27	0026	001A	40	5A	□	!	20	5D	□	]
01	28	0027	001B	40	5B	□	$	20	24	□	$
01	29	0028	001C	40	5C	□	*	20	2A	□	*
01	30	0029	001D	40	5D	□	)	20	29	□	)
01	31	0030	001E	40	5E	□	;	20	3B	□	;
01	32	0031	001F	40	5F	□	¬	20	5E	□	↑
01	33	0032	0020	40	60	□	-	20	2D	□	-
01	34	0033	0021	40	61	□	/	20	2F	□	/
01	35	0034	0022	40	E2	□	S	20	53	□	S
01	36	0035	0023	40	E3	□	T	20	54	□	T
01	37	0036	0024	40	E4	□	U	20	55	□	U
01	38	0037	0025	40	E5	□	V	20	56	□	V
01	39	0038	0026	40	E6	□	W	20	57	□	W
01	40	0039	0027	40	E7	□	X	20	58	□	X
01	41	0040	0028	40	E8	□	Y	20	59	□	Y
01	42	0041	0029	40	E9	□	Z	20	5A	□	Z
01	43	0042	002A	40	6A	□	¦	20	7C	□	¦
01	44	0043	002B	40	6B	□	,	20	2C	□	,
01	45	0044	002C	40	6C	□	%	20	25	□	%
01	46	0045	002D	40	6D	□	_	20	5F	□	_
01	47	0046	002E	40	6E	□	>	20	3E	□	>
01	48	0047	002F	40	6F	□	?	20	3F	□	?
01	49	0048	0030	40	F0	□	0	20	30	□	0
01	50	0049	0031	40	F1	□	1	20	31	□	1
01	51	0050	0032	40	F2	□	2	20	32	□	2
01	52	0051	0033	40	F3	□	3	20	33	□	3
01	53	0052	0034	40	F4	□	4	20	34	□	4
01	54	0053	0035	40	F5	□	5	20	35	□	5
01	55	0054	0036	40	F6	□	6	20	36	□	6

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
01	56	0055	0037	40	F7	□	7	20	37	□	7
01	57	0056	0038	40	F8	□	8	20	38	□	8
01	58	0057	0039	40	F9	□	9	20	39	□	9
01	59	0058	003A	40	7A	□	:	20	3A	□	:
01	60	0059	003B	40	7B	□	#	20	23	□	#
01	61	0060	003C	40	7C	□	@	20	40	□	@
01	62	0061	003D	40	7D	□	'	20	27	□	'
01	63	0062	003E	40	7E	□	=	20	3D	□	=
01	64	0063	003F	40	7F	□	"	20	22	□	"
01	65	0064	0040	C1	40	A	□	41	20	A	□
01	66	0065	0041	C1	C1	A	A	41	41	A	A
01	67	0066	0042	C1	C2	A	B	41	42	A	B
01	68	0067	0043	C1	C3	A	C	41	43	A	C
01	69	0068	0044	C1	C4	A	D	41	44	A	D
01	70	0069	0045	C1	C5	A	E	41	45	A	E
01	71	0070	0046	C1	C6	A	F	41	46	A	F
01	72	0071	0047	C1	C7	A	G	41	47	A	G
01	73	0072	0048	C1	C8	A	H	41	48	A	H
01	74	0073	0049	C1	C9	A	I	41	49	A	I
01	75	0074	004A	C1	4A	A	¢	41	5B	A	[
01	76	0075	004B	C1	4B	A	.	41	2E	A	.
01	77	0076	004C	C1	4C	A	<	41	3C	A	<
01	78	0077	004D	C1	4D	A	(	41	28	A	(
01	79	0078	004E	C1	4E	A	+	41	2B	A	+
01	80	0079	004F	C1	4F	A	|	41	21	A	!
02	01	0080	0050	C1	50	A	&	41	26	A	&
02	02	0081	0051	C1	D1	A	J	41	4A	A	J
02	03	0082	0052	C1	D2	A	K	41	4B	A	K
02	04	0083	0053	C1	D3	A	L	41	4C	A	L
02	05	0084	0054	C1	D4	A	M	41	4D	A	M
02	06	0085	0055	C1	D5	A	N	41	4E	A	N
02	07	0086	0056	C1	D6	A	O	41	4F	A	O
02	08	0087	0057	C1	D7	A	P	41	50	A	P
02	09	0088	0058	C1	D8	A	Q	41	51	A	Q
02	10	0089	0059	C1	D9	A	R	41	52	A	R
02	11	0090	005A	C1	5A	A	!	41	5D	A	]
02	12	0091	005B	C1	5B	A	$	41	24	A	$
02	13	0092	005C	C1	5C	A	*	41	2A	A	*
02	14	0093	005D	C1	5D	A	)	41	29	A	)
02	15	0094	005E	C1	5E	A	;	41	3B	A	;
02	16	0095	005F	C1	5F	A	¬	41	5E	A	↑
02	17	0096	0060	C1	60	A	-	41	2D	A	-
02	18	0097	0061	C1	61	A	/	41	2F	A	/
02	19	0098	0062	C1	E2	A	S	41	53	A	S
02	20	0099	0063	C1	E3	A	T	41	54	A	T
02	21	0100	0064	C1	E4	A	U	41	55	A	U
02	22	0101	0065	C1	E5	A	V	41	56	A	V
02	23	0102	0066	C1	E6	A	W	41	57	A	W
02	24	0103	0067	C1	E7	A	X	41	58	A	X
02	25	0104	0068	C1	E8	A	Y	41	59	A	Y
02	26	0105	0069	C1	E9	A	Z	41	5A	A	Z
02	27	0106	006A	C1	6A	A	!	41	7C	A	!
02	28	0107	006B	C1	6B	A	,	41	2C	A	,
02	29	0108	006C	C1	6C	A	%	41	25	A	%
02	30	0109	006D	C1	6D	A	_	41	5F	A	←
02	31	0110	006E	C1	6E	A	>	41	3E	A	>
02	32	0111	006F	C1	6F	A	?	41	3F	A	?
02	33	0112	0070	C1	F0	A	0	41	30	A	0
02	34	0113	0071	C1	F1	A	1	41	31	A	1
02	35	0114	0072	C1	F2	A	2	41	32	A	2
02	36	0115	0073	C1	F3	A	3	41	33	A	3
02	37	0116	0074	C1	F4	A	4	41	34	A	4

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
02	38	0117	0075	C1	F5	A	5	41	35	A	5
02	39	0118	0076	C1	F6	A	6	41	36	A	6
02	40	0119	0077	C1	F7	A	7	41	37	A	7
02	41	0120	0078	C1	F8	A	8	41	38	A	8
02	42	0121	0079	C1	F9	A	9	41	39	A	9
02	43	0122	007A	C1	7A	A	:	41	3A	A	:
02	44	0123	007B	C1	7B	A	#	41	23	A	#
02	45	0124	007C	C1	7C	A	@	41	40	A	@
02	46	0125	007D	C1	7D	A	'	41	27	A	'
02	47	0126	007E	C1	7E	A	=	41	3D	A	=
02	48	0127	007F	C1	7F	A	"	41	22	A	"
02	49	0128	0080	C2	40	B	□	42	20	B	□
02	50	0129	0081	C2	C1	B	A	42	41	B	A
02	51	0130	0082	C2	C2	B	B	42	42	B	B
02	52	0131	0083	C2	C3	B	C	42	43	B	C
02	53	0132	0084	C2	C4	B	D	42	44	B	D
02	54	0133	0085	C2	C5	B	E	42	45	B	E
02	55	0134	0086	C2	C6	B	F	42	46	B	F
02	56	0135	0087	C2	C7	B	G	42	47	B	G
02	57	0136	0088	C2	C8	B	H	42	48	B	H
02	58	0137	0089	C2	C9	B	I	42	49	B	I
02	59	0138	008A	C2	4A	B	∅	42	5B	B	[
02	60	0139	008B	C2	4B	B	.	42	2E	B	.
02	61	0140	008C	C2	4C	B	<	42	3C	B	<
02	62	0141	008D	C2	4D	B	(	42	28	B	(
02	63	0142	008E	C2	4E	B	+	42	2B	B	+
02	64	0143	008F	C2	4F	B	|	42	21	B	!
02	65	0144	0090	C2	50	B	&	42	26	B	&
02	66	0145	0091	C2	D1	B	J	42	4A	B	J
02	67	0146	0092	C2	D2	B	K	42	4B	B	K
02	68	0147	0093	C2	D3	B	L	42	4C	B	L
02	69	0148	0094	C2	D4	B	M	42	4D	B	M
02	70	0149	0095	C2	D5	B	N	42	4E	B	N
02	71	0150	0096	C2	D6	B	O	42	4F	B	O
02	72	0151	0097	C2	D7	B	P	42	50	B	P
02	73	0152	0098	C2	D8	B	Q	42	51	B	Q
02	74	0153	0099	C2	D9	B	R	42	52	B	R
02	75	0154	009A	C2	5A	B	!	42	5D	B	]
02	76	0155	009B	C2	5B	B	$	42	24	B	$
02	77	0156	009C	C2	5C	B	*	42	2A	B	*
02	78	0157	009D	C2	5D	B	)	42	29	B	)
02	79	0158	009E	C2	5E	B	;	42	3B	B	;
02	80	0159	009F	C2	5F	B	¬	42	5E	B	↑
03	01	0160	00A0	C2	60	B	-	42	2D	B	-
03	02	0161	00A1	C2	61	B	/	42	2F	B	/
03	03	0162	00A2	C2	E2	B	S	42	53	B	S
03	04	0163	00A3	C2	E3	B	T	42	54	B	T
03	05	0164	00A4	C2	E4	B	U	42	55	B	U
03	06	0165	00A5	C2	E5	B	V	42	56	B	V
03	07	0166	00A6	C2	E6	B	W	42	57	B	W
03	08	0167	00A7	C2	E7	B	X	42	58	B	X
03	09	0168	00A8	C2	E8	B	Y	42	59	B	Y
03	10	0169	00A9	C2	E9	B	Z	42	5A	B	Z
03	11	0170	00AA	C2	6A	B	|	42	7C	B	|
03	12	0171	00AB	C2	6B	B	,	42	2C	B	,
03	13	0172	00AC	C2	6C	B	%	42	25	B	%
03	14	0173	00AD	C2	6D	B	_	42	5F	B	_
03	15	0174	00AE	C2	6E	B	>	42	3E	B	>
03	16	0175	00AF	C2	6F	B	?	42	3F	B	?
03	17	0176	00B0	C2	F0	B	0	42	30	B	0
03	18	0177	00B1	C2	F1	B	1	42	31	B	1

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
03	19	0178	00B2	C2	F2	B	2	42	32	B	2
03	20	0179	00B3	C2	F3	B	3	42	33	B	3
03	21	0180	00B4	C2	F4	B	4	42	34	B	4
03	22	0181	00B5	C2	F5	B	5	42	35	B	5
03	23	0182	00B6	C2	F6	B	6	42	36	B	6
03	24	0183	00B7	C2	F7	B	7	42	37	B	7
03	25	0184	00B8	C2	F8	B	8	42	38	B	8
03	26	0185	00B9	C2	F9	B	9	42	39	B	9
03	27	0186	00BA	C2	7A	B	:	42	3A	B	:
03	28	0187	00BB	C2	7B	B	#	42	23	B	#
03	29	0188	00BC	C2	7C	B	@	42	40	B	@
03	30	0189	00BD	C2	7D	B	'	42	27	B	'
03	31	0190	00BE	C2	7E	B	=	42	3D	B	=
03	32	0191	00BF	C2	7F	B	"	42	22	B	"
03	33	0192	00C0	C3	40	C	□	43	20	C	□
03	34	0193	00C1	C3	C1	C	A	43	41	C	A
03	35	0194	00C2	C3	C2	C	B	43	42	C	B
03	36	0195	00C3	C3	C3	C	C	43	43	C	C
03	37	0196	00C4	C3	C4	C	D	43	44	C	D
03	38	0197	00C5	C3	C5	C	E	43	45	C	E
03	39	0198	00C6	C3	C6	C	F	43	46	C	F
03	40	0199	00C7	C3	C7	C	G	43	47	C	G
03	41	0200	00C8	C3	C8	C	H	43	48	C	H
03	42	0201	00C9	C3	C9	C	I	43	49	C	I
03	43	0202	00CA	C3	4A	C	¢	43	5B	C	[
03	44	0203	00CB	C3	4B	C	.	43	2E	C	.
03	45	0204	00CC	C3	4C	C	<	43	3C	C	<
03	46	0205	00CD	C3	4D	C	(	43	28	C	(
03	47	0206	00CE	C3	4E	C	+	43	2B	C	+
03	48	0207	00CF	C3	4F	C	|	43	21	C	!
03	49	0208	00D0	C3	50	C	&	43	26	C	&
03	50	0209	00D1	C3	D1	C	J	43	4A	C	J
03	51	0210	00D2	C3	D2	C	K	43	4B	C	K
03	52	0211	00D3	C3	D3	C	L	43	4C	C	L
03	53	0212	00D4	C3	D4	C	M	43	4D	C	M
03	54	0213	00D5	C3	D5	C	N	43	4E	C	N
03	55	0214	00D6	C3	D6	C	O	43	4F	C	O
03	56	0215	00D7	C3	D7	C	P	43	50	C	P
03	57	0216	00D8	C3	D8	C	Q	43	51	C	Q
03	58	0217	00D9	C3	D9	C	R	43	52	C	R
03	59	0218	00DA	C3	5A	C	!	43	5D	C	]
03	60	0219	00DB	C3	5B	C	$	43	24	C	$
03	61	0220	00DC	C3	5C	C	*	43	2A	C	*
03	62	0221	00DD	C3	5D	C	)	43	29	C	)
03	63	0222	00DE	C3	5E	C	;	43	3B	C	;
03	64	0223	00DF	C3	5F	C	¬	43	5E	C	↑
03	65	0224	00E0	C3	60	C	-	43	2D	C	-
03	66	0225	00E1	C3	61	C	/	43	2F	C	/
03	67	0226	00E2	C3	E2	C	S	43	53	C	S
03	68	0227	00E3	C3	E3	C	T	43	54	C	T
03	69	0228	00E4	C3	E4	C	U	43	55	C	U
03	70	0229	00E5	C3	E5	C	V	43	56	C	V
03	71	0230	00E6	C3	E6	C	W	43	57	C	W
03	72	0231	00E7	C3	E7	C	X	43	58	C	X
03	73	0232	00E8	C3	E8	C	Y	43	59	C	Y
03	74	0233	00E9	C3	E9	C	Z	43	5A	C	Z
03	75	0234	00EA	C3	6A	C	¦	43	7C	C	¦
03	76	0235	00EB	C3	6B	C	,	43	2C	C	,
03	77	0236	00EC	C3	6C	C	%	43	25	C	%
03	78	0237	00ED	C3	6D	C	_	43	5F	C	_
03	79	0238	00EE	C3	6E	C	>	43	3E	C	>
03	80	0239	00EF	C3	6F	C	?	43	3F	C	?

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
04	01	0240	00F0	C3	F0	C	0	43	30	C	0
04	02	0241	00F1	C3	F1	C	1	43	31	C	1
04	03	0242	00F2	C3	F2	C	2	43	32	C	2
04	04	0243	00F3	C3	F3	C	3	43	33	C	3
04	05	0244	00F4	C3	F4	C	4	43	34	C	4
04	06	0245	00F5	C3	F5	C	5	43	35	C	5
04	07	0246	00F6	C3	F6	C	6	43	36	C	6
04	08	0247	00F7	C3	F7	C	7	43	37	C	7
04	09	0248	00F8	C3	F8	C	8	43	38	C	8
04	10	0249	00F9	C3	F9	C	9	43	39	C	9
04	11	0250	00FA	C3	7A	C	:	43	3A	C	:
04	12	0251	00FB	C3	7B	C	#	43	23	C	#
04	13	0252	00FC	C3	7C	C	@	43	40	C	@
04	14	0253	00FD	C3	7D	C	'	43	27	C	'
04	15	0254	00FE	C3	7E	C	=	43	3D	C	=
04	16	0255	00FF	C3	7F	C	"	43	22	C	"
04	17	0256	0100	C4	40	D	□	44	20	D	□
04	18	0257	0101	C4	C1	D	A	44	41	D	A
04	19	0258	0102	C4	C2	D	B	44	42	D	B
04	20	0259	0103	C4	C3	D	C	44	43	D	C
04	21	0260	0104	C4	C4	D	D	44	44	D	D
04	22	0261	0105	C4	C5	D	E	44	45	D	E
04	23	0262	0106	C4	C6	D	F	44	46	D	F
04	24	0263	0107	C4	C7	D	G	44	47	D	G
04	25	0264	0108	C4	C8	D	H	44	48	D	H
04	26	0265	0109	C4	C9	D	I	44	49	D	I
04	27	0266	010A	C4	4A	D	¢	44	5B	D	[
04	28	0267	010B	C4	4B	D	.	44	2E	D	.
04	29	0268	010C	C4	4C	D	<	44	3C	D	<
04	30	0269	010D	C4	4D	D	(	44	28	D	(
04	31	0270	010E	C4	4E	D	+	44	2B	D	+
04	32	0271	010F	C4	4F	D	|	44	21	D	!
04	33	0272	0110	C4	50	D	&	44	26	D	&
04	34	0273	0111	C4	D1	D	J	44	4A	D	J
04	35	0274	0112	C4	D2	D	K	44	4B	D	K
04	36	0275	0113	C4	D3	D	L	44	4C	D	L
04	37	0276	0114	C4	D4	D	M	44	4D	D	M
04	38	0277	0115	C4	D5	D	N	44	4E	D	N
04	39	0278	0116	C4	D6	D	O	44	4F	D	O
04	40	0279	0117	C4	D7	D	P	44	50	D	P
04	41	0280	0118	C4	D8	D	Q	44	51	D	Q
04	42	0281	0119	C4	D9	D	R	44	52	D	R
04	43	0282	011A	C4	5A	D	!	44	5D	D	]
04	44	0283	011B	C4	5B	D	$	44	24	D	$
04	45	0284	011C	C4	5C	D	*	44	2A	D	*
04	46	0285	011D	C4	5D	D	)	44	29	D	)
04	47	0286	011E	C4	5E	D	;	44	3B	D	;
04	48	0287	011F	C4	5F	D	¬	44	5E	D	↑
04	49	0288	0120	C4	60	D	-	44	2D	D	-
04	50	0289	0121	C4	61	D	/	44	2F	D	/
04	51	0290	0122	C4	E2	D	S	44	53	D	S
04	52	0291	0123	C4	E3	D	T	44	54	D	T
04	53	0292	0124	C4	E4	D	U	44	55	D	U
04	54	0293	0125	C4	E5	D	V	44	56	D	V
04	55	0294	0126	C4	E6	D	W	44	57	D	W
04	56	0295	0127	C4	E7	D	X	44	58	D	X
04	57	0296	0128	C4	E8	D	Y	44	59	D	Y
04	58	0297	0129	C4	E9	D	Z	44	5A	D	Z
04	59	0298	012A	C4	6A	D	!	44	7C	D	!
04	60	0299	012B	C4	6B	D	,	44	2C	D	,
04	61	0300	012C	C4	6C	D	%	44	25	D	%
04	62	0301	012D	C4	6D	D	_	44	5F	D	←

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
04	63	0302	012E	C4	6E	D	>	44	3E	D	>
04	64	0303	012F	C4	6F	D	?	44	3F	D	?
04	65	0304	0130	C4	F0	D	0	44	30	D	0
04	66	0305	0131	C4	F1	D	1	44	31	D	1
04	67	0306	0132	C4	F2	D	2	44	32	D	2
04	68	0307	0133	C4	F3	D	3	44	33	D	3
04	69	0308	0134	C4	F4	D	4	44	34	D	4
04	70	0309	0135	C4	F5	D	5	44	35	D	5
04	71	0310	0136	C4	F6	D	6	44	36	D	6
04	72	0311	0137	C4	F7	D	7	44	37	D	7
04	73	0312	0138	C4	F8	D	8	44	38	D	8
04	74	0313	0139	C4	F9	D	9	44	39	D	9
04	75	0314	013A	C4	7A	D	:	44	3A	D	:
04	76	0315	013B	C4	7B	D	#	44	23	D	#
04	77	0316	013C	C4	7C	D	@	44	40	D	@
04	78	0317	013D	C4	7D	D	'	44	27	D	'
04	79	0318	013E	C4	7E	D	=	44	3D	D	=
04	80	0319	013F	C4	7F	D	"	44	22	D	"
05	01	0320	0140	C5	40	E	□	45	20	E	□
05	02	0321	0141	C5	C1	E	A	45	41	E	A
05	03	0322	0142	C5	C2	E	B	45	42	E	B
05	04	0323	0143	C5	C3	E	C	45	43	E	C
05	05	0324	0144	C5	C4	E	D	45	44	E	D
05	06	0325	0145	C5	C5	E	E	45	45	E	E
05	07	0326	0146	C5	C6	E	F	45	46	E	F
05	08	0327	0147	C5	C7	E	G	45	47	E	G
05	09	0328	0148	C5	C8	E	H	45	48	E	H
05	10	0329	0149	C5	C9	E	I	45	49	E	I
05	11	0330	014A	C5	4A	E	Ø	45	5B	E	[
05	12	0331	014B	C5	4B	E	.	45	2E	E	.
05	13	0332	014C	C5	4C	E	<	45	3C	E	<
05	14	0333	014D	C5	4D	E	(	45	28	E	(
05	15	0334	014E	C5	4E	E	+	45	2B	E	+
05	16	0335	014F	C5	4F	E	|	45	21	E	!
05	17	0336	0150	C5	50	E	&	45	26	E	&
05	18	0337	0151	C5	D1	E	J	45	4A	E	J
05	19	0338	0152	C5	D2	E	K	45	4B	E	K
05	20	0339	0153	C5	D3	E	L	45	4C	E	L
05	21	0340	0154	C5	D4	E	M	45	4D	E	M
05	22	0341	0155	C5	D5	E	N	45	4E	E	N
05	23	0342	0156	C5	D6	E	O	45	4F	E	O
05	24	0343	0157	C5	D7	E	P	45	50	E	P
05	25	0344	0158	C5	D8	E	Q	45	51	E	Q
05	26	0345	0159	C5	D9	E	R	45	52	E	R
05	27	0346	015A	C5	5A	E	!	45	5D	E	]
05	28	0347	015B	C5	5B	E	$	45	24	E	$
05	29	0348	015C	C5	5C	E	*	45	2A	E	*
05	30	0349	015D	C5	5D	E	)	45	29	E	)
05	31	0350	015E	C5	5E	E	;	45	3B	E	;
05	32	0351	015F	C5	5F	E	¬	45	5E	E	↑
05	33	0352	0160	C5	60	E	-	45	2D	E	-
05	34	0353	0161	C5	61	E	/	45	2F	E	/
05	35	0354	0162	C5	E2	E	S	45	53	E	S
05	36	0355	0163	C5	E3	E	T	45	54	E	T
05	37	0356	0164	C5	E4	E	U	45	55	E	U
05	38	0357	0165	C5	E5	E	V	45	56	E	V
05	39	0358	0166	C5	E6	E	W	45	57	E	W
05	40	0359	0167	C5	E7	E	X	45	58	E	X
05	41	0360	0168	C5	E8	E	Y	45	59	E	Y
05	42	0361	0169	C5	E9	E	Z	45	5A	E	Z
05	43	0362	016A	C5	6A	E	!	45	7C	E	!
05	44	0363	016B	C5	6B	E	,	45	2C	E	,

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
05	45	0364	016C	C5	6C	E	%	45	25	E	%
05	46	0365	016D	C5	6D	E	_	45	5F	E	_
05	47	0366	016E	C5	6E	E	>	45	3E	E	>
05	48	0367	016F	C5	6F	E	?	45	3F	E	?
05	49	0368	0170	C5	F0	E	0	45	30	E	0
05	50	0369	0171	C5	F1	E	1	45	31	E	1
05	51	0370	0172	C5	F2	E	2	45	32	E	2
05	52	0371	0173	C5	F3	E	3	45	33	E	3
05	53	0372	0174	C5	F4	E	4	45	34	E	4
05	54	0373	0175	C5	F5	E	5	45	35	E	5
05	55	0374	0176	C5	F6	E	6	45	36	E	6
05	56	0375	0177	C5	F7	E	7	45	37	E	7
05	57	0376	0178	C5	F8	E	8	45	38	E	8
05	58	0377	0179	C5	F9	E	9	45	39	E	9
05	59	0378	017A	C5	7A	E	:	45	3A	E	:
05	60	0379	017B	C5	7B	E	#	45	23	E	#
05	61	0380	017C	C5	7C	E	@	45	40	E	@
05	62	0381	017D	C5	7D	E	'	45	27	E	'
05	63	0382	017E	C5	7E	E	=	45	3D	E	=
05	64	0383	017F	C5	7F	E	"	45	22	E	"
05	65	0384	0180	C6	40	F	□	46	20	F	□
05	66	0385	0181	C6	C1	F	A	46	41	F	A
05	67	0386	0182	C6	C2	F	B	46	42	F	B
05	68	0387	0183	C6	C3	F	C	46	43	F	C
05	69	0388	0184	C6	C4	F	D	46	44	F	D
05	70	0389	0185	C6	C5	F	E	46	45	F	E
05	71	0390	0186	C6	C6	F	F	46	46	F	F
05	72	0391	0187	C6	C7	F	G	46	47	F	G
05	73	0392	0188	C6	C8	F	H	46	48	F	H
05	74	0393	0189	C6	C9	F	I	46	49	F	I
05	75	0394	018A	C6	4A	F	¢	46	5B	F	[
05	76	0395	018B	C6	4B	F	.	46	2E	F	.
05	77	0396	018C	C6	4C	F	<	46	3C	F	<
05	78	0397	018D	C6	4D	F	(	46	28	F	(
05	79	0398	018E	C6	4E	F	+	46	2B	F	+
05	80	0399	018F	C6	4F	F	|	46	21	F	!
06	01	0400	0190	C6	50	F	&	46	26	F	&
06	02	0401	0191	C6	D1	F	J	46	4A	F	J
06	03	0402	0192	C6	D2	F	K	46	4B	F	K
06	04	0403	0193	C6	D3	F	L	46	4C	F	L
06	05	0404	0194	C6	D4	F	M	46	4D	F	M
06	06	0405	0195	C6	D5	F	N	46	4E	F	N
06	07	0406	0196	C6	D6	F	O	46	4F	F	O
06	08	0407	0197	C6	D7	F	P	46	50	F	P
06	09	0408	0198	C6	D8	F	Q	46	51	F	Q
06	10	0409	0199	C6	D9	F	R	46	52	F	R
06	11	0410	019A	C6	5A	F	!	46	5D	F	]
06	12	0411	019B	C6	5B	F	$	46	24	F	$
06	13	0412	019C	C6	5C	F	*	46	2A	F	*
06	14	0413	019D	C6	5D	F	)	46	29	F	)
06	15	0414	019E	C6	5E	F	;	46	3B	F	;
06	16	0415	019F	C6	5F	F	¬	46	5E	F	↑
06	17	0416	01A0	C6	60	F	-	46	2D	F	-
06	18	0417	01A1	C6	61	F	/	46	2F	F	/
06	19	0418	01A2	C6	E2	F	S	46	53	F	S
06	20	0419	01A3	C6	E3	F	T	46	54	F	T
06	21	0420	01A4	C6	E4	F	U	46	55	F	U
06	22	0421	01A5	C6	E5	F	V	46	56	F	V
06	23	0422	01A6	C6	E6	F	W	46	57	F	W
06	24	0423	01A7	C6	E7	F	X	46	58	F	X
06	25	0424	01A8	C6	E8	F	Y	46	59	F	Y
06	26	0425	01A9	C6	E9	F	Z	46	5A	F	Z

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
06	27	0426	01AA	C6	6A	F	¦	46	7C	F	¦
06	28	0427	01AB	C6	6B	F	,	46	2C	F	,
06	29	0428	01AC	C6	6C	F	%	46	25	F	%
06	30	0429	01AD	C6	6D	F	_	46	5F	F	_
06	31	0430	01AE	C6	6E	F	>	46	3E	F	>
06	32	0431	01AF	C6	6F	F	?	46	3F	F	?
06	33	0432	01B0	C6	F0	F	0	46	30	F	0
06	34	0433	01B1	C6	F1	F	1	46	31	F	1
06	35	0434	01B2	C6	F2	F	2	46	32	F	2
06	36	0435	01B3	C6	F3	F	3	46	33	F	3
06	37	0436	01B4	C6	F4	F	4	46	34	F	4
06	38	0437	01B5	C6	F5	F	5	46	35	F	5
06	39	0438	01B6	C6	F6	F	6	46	36	F	6
06	40	0439	01B7	C6	F7	F	7	46	37	F	7
06	41	0440	01B8	C6	F8	F	8	46	38	F	8
06	42	0441	01B9	C6	F9	F	9	46	39	F	9
06	43	0442	01BA	C6	7A	F	:	46	3A	F	:
06	44	0443	01BB	C6	7B	F	#	46	23	F	#
06	45	0444	01BC	C6	7C	F	@	46	40	F	@
06	46	0445	01BD	C6	7D	F	'	46	27	F	'
06	47	0446	01BE	C6	7E	F	=	46	3D	F	=
06	48	0447	01BF	C6	7F	F	"	46	22	F	"
06	49	0448	01C0	C7	40	G	□	47	20	G	□
06	50	0449	01C1	C7	C1	G	A	47	41	G	A
06	51	0450	01C2	C7	C2	G	B	47	42	G	B
06	52	0451	01C3	C7	C3	G	C	47	43	G	C
06	53	0452	01C4	C7	C4	G	D	47	44	G	D
06	54	0453	01C5	C7	C5	G	E	47	45	G	E
06	55	0454	01C6	C7	C6	G	F	47	46	G	F
06	56	0455	01C7	C7	C7	G	G	47	47	G	G
06	57	0456	01C8	C7	C8	G	H	47	48	G	H
06	58	0457	01C9	C7	C9	G	I	47	49	G	I
06	59	0458	01CA	C7	4A	G	¢	47	5B	G	[
06	60	0459	01CB	C7	4B	G	.	47	2E	G	.
06	61	0460	01CC	C7	4C	G	<	47	3C	G	<
06	62	0461	01CD	C7	4D	G	(	47	28	G	(
06	63	0462	01CE	C7	4E	G	+	47	2B	G	+
06	64	0463	01CF	C7	4F	G	|	47	21	G	!
06	65	0464	01D0	C7	50	G	&	47	26	G	&
06	66	0465	01D1	C7	D1	G	J	47	4A	G	J
06	67	0466	01D2	C7	D2	G	K	47	4B	G	K
06	68	0467	01D3	C7	D3	G	L	47	4C	G	L
06	69	0468	01D4	C7	D4	G	M	47	4D	G	M
06	70	0469	01D5	C7	D5	G	N	47	4E	G	N
06	71	0470	01D6	C7	D6	G	O	47	4F	G	O
06	72	0471	01D7	C7	D7	G	P	47	50	G	P
06	73	0472	01D8	C7	D8	G	Q	47	51	G	Q
06	74	0473	01D9	C7	D9	G	R	47	52	G	R
06	75	0474	01DA	C7	5A	G	!	47	5D	G	]
06	76	0475	01DB	C7	5B	G	$	47	24	G	$
06	77	0476	01DC	C7	5C	G	*	47	2A	G	*
06	78	0477	01DD	C7	5D	G	)	47	29	G	)
06	79	0478	01DE	C7	5E	G	;	47	3B	G	;
06	80	0479	01DF	C7	5F	G	¬	47	5E	G	↑
07	01	0480	01E0	C7	60	G	-	47	2D	G	-
07	02	0481	01E1	C7	61	G	/	47	2F	G	/
07	03	0482	01E2	C7	E2	G	S	47	53	G	S
07	04	0483	01E3	C7	E3	G	T	47	54	G	T
07	05	0484	01E4	C7	E4	G	U	47	55	G	U
07	06	0485	01E5	C7	E5	G	V	47	56	G	V
07	07	0486	01E6	C7	E6	G	W	47	57	G	W
07	08	0487	01E7	C7	E7	G	X	47	58	G	X

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
07	09	0488	01E8	C7	E8	G	Y	47	59	G	Y
07	10	0489	01E9	C7	E9	G	Z	47	5A	G	Z
07	11	0490	01EA	C7	6A	G	¦	47	7C	G	¦
07	12	0491	01EB	C7	6B	G	,	47	2C	G	,
07	13	0492	01EC	C7	6C	G	%	47	25	G	%
07	14	0493	01ED	C7	6D	G	_	47	5F	G	_
07	15	0494	01EE	C7	6E	G	>	47	3E	G	>
07	16	0495	01EF	C7	6F	G	?	47	3F	G	?
07	17	0496	01F0	C7	F0	G	0	47	30	G	0
07	18	0497	01F1	C7	F1	G	1	47	31	G	1
07	19	0498	01F2	C7	F2	G	2	47	32	G	2
07	20	0499	01F3	C7	F3	G	3	47	33	G	3
07	21	0500	01F4	C7	F4	G	4	47	34	G	4
07	22	0501	01F5	C7	F5	G	5	47	35	G	5
07	23	0502	01F6	C7	F6	G	6	47	36	G	6
07	24	0503	01F7	C7	F7	G	7	47	37	G	7
07	25	0504	01F8	C7	F8	G	8	47	38	G	8
07	26	0505	01F9	C7	F9	G	9	47	39	G	9
07	27	0506	01FA	C7	7A	G	:	47	3A	G	:
07	28	0507	01FB	C7	7B	G	#	47	23	G	#
07	29	0508	01FC	C7	7C	G	@	47	40	G	@
07	30	0509	01FD	C7	7D	G	'	47	27	G	'
07	31	0510	01FE	C7	7E	G	=	47	3D	G	=
07	32	0511	01FF	C7	7F	G	"	47	22	G	"
07	33	0512	0200	C8	40	H	□	48	20	H	□
07	34	0513	0201	C8	C1	H	A	48	41	H	A
07	35	0514	0202	C8	C2	H	B	48	42	H	B
07	36	0515	0203	C8	C3	H	C	48	43	H	C
07	37	0516	0204	C8	C4	H	D	48	44	H	D
07	38	0517	0205	C8	C5	H	E	48	45	H	E
07	39	0518	0206	C8	C6	H	F	48	46	H	F
07	40	0519	0207	C8	C7	H	G	48	47	H	G
07	41	0520	0208	C8	C8	H	H	48	48	H	H
07	42	0521	0209	C8	C9	H	I	48	49	H	I
07	43	0522	020A	C8	4A	H	¢	48	5B	H	[
07	44	0523	020B	C8	4B	H	.	48	2E	H	.
07	45	0524	020C	C8	4C	H	<	48	3C	H	<
07	46	0525	020D	C8	4D	H	(	48	28	H	(
07	47	0526	020E	C8	4E	H	+	48	2B	H	+
07	48	0527	020F	C8	4F	H	|	48	21	H	!
07	49	0528	0210	C8	50	H	&	48	26	H	&
07	50	0529	0211	C8	D1	H	J	48	4A	H	J
07	51	0530	0212	C8	D2	H	K	48	4B	H	K
07	52	0531	0213	C8	D3	H	L	48	4C	H	L
07	53	0532	0214	C8	D4	H	M	48	4D	H	M
07	54	0533	0215	C8	D5	H	N	48	4E	H	N
07	55	0534	0216	C8	D6	H	O	48	4F	H	O
07	56	0535	0217	C8	D7	H	P	48	50	H	P
07	57	0536	0218	C8	D8	H	Q	48	51	H	Q
07	58	0537	0219	C8	D9	H	R	48	52	H	R
07	59	0538	021A	C8	5A	H	!	48	5D	H	]
07	60	0539	021B	C8	5B	H	$	48	24	H	$
07	61	0540	021C	C8	5C	H	*	48	2A	H	*
07	62	0541	021D	C8	5D	H	)	48	29	H	)
07	63	0542	021E	C8	5E	H	;	48	3B	H	;
07	64	0543	021F	C8	5F	H	¬	48	5E	H	↑
07	65	0544	0220	C8	60	H	-	48	2D	H	-
07	66	0545	0221	C8	61	H	/	48	2F	H	/
07	67	0546	0222	C8	E2	H	S	48	53	H	S
07	68	0547	0223	C8	E3	H	T	48	54	H	T
07	69	0548	0224	C8	E4	H	U	48	55	H	U
07	70	0549	0225	C8	E5	H	V	48	56	H	V

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
07	71	0550	0226	C8	E6	H	W	48	57	H	W
07	72	0551	0227	C8	E7	H	X	48	58	H	X
07	73	0552	0228	C8	E8	H	Y	48	59	H	Y
07	74	0553	0229	C8	E9	H	Z	48	5A	H	Z
07	75	0554	022A	C8	6A	H	¦	48	7C	H	¦
07	76	0555	022B	C8	6B	H	,	48	2C	H	,
07	77	0556	022C	C8	6C	H	%	48	25	H	%
07	78	0557	022D	C8	6D	H	_	48	5F	H	_
07	79	0558	022E	C8	6E	H	>	48	3E	H	>
07	80	0559	022F	C8	6F	H	?	48	3F	H	?
08	01	0560	0230	C8	F0	H	0	48	30	H	0
08	02	0561	0231	C8	F1	H	1	48	31	H	1
08	03	0562	0232	C8	F2	H	2	48	32	H	2
08	04	0563	0233	C8	F3	H	3	48	33	H	3
08	05	0564	0234	C8	F4	H	4	48	34	H	4
08	06	0565	0235	C8	F5	H	5	48	35	H	5
08	07	0566	0236	C8	F6	H	6	48	36	H	6
08	08	0567	0237	C8	F7	H	7	48	37	H	7
08	09	0568	0238	C8	F8	H	8	48	38	H	8
08	10	0569	0239	C8	F9	H	9	48	39	H	9
08	11	0570	023A	C8	7A	H	:	48	3A	H	:
08	12	0571	023B	C8	7B	H	#	48	23	H	#
08	13	0572	023C	C8	7C	H	@	48	40	H	@
08	14	0573	023D	C8	7D	H	'	48	27	H	'
08	15	0574	023E	C8	7E	H	=	48	3D	H	=
08	16	0575	023F	C8	7F	H	"	48	22	H	"
08	17	0576	0240	C9	40	I	□	49	20	I	□
08	18	0577	0241	C9	C1	I	A	49	41	I	A
08	19	0578	0242	C9	C2	I	B	49	42	I	B
08	20	0579	0243	C9	C3	I	C	49	43	I	C
08	21	0580	0244	C9	C4	I	D	49	44	I	D
08	22	0581	0245	C9	C5	I	E	49	45	I	E
08	23	0582	0246	C9	C6	I	F	49	46	I	F
08	24	0583	0247	C9	C7	I	G	49	47	I	G
08	25	0584	0248	C9	C8	I	H	49	48	I	H
08	26	0585	0249	C9	C9	I	I	49	49	I	I
08	27	0586	024A	C9	4A	I	∅	49	5B	I	[
08	28	0587	024B	C9	4B	I	.	49	2E	I	.
08	29	0588	024C	C9	4C	I	<	49	3C	I	<
08	30	0589	024D	C9	4D	I	(	49	28	I	(
08	31	0590	024E	C9	4E	I	+	49	2B	I	+
08	32	0591	024F	C9	4F	I	—	49	21	I	!
08	33	0592	0250	C9	50	I	&	49	26	I	&
08	34	0593	0251	C9	D1	I	J	49	4A	I	J
08	35	0594	0252	C9	D2	I	K	49	4B	I	K
08	36	0595	0253	C9	D3	I	L	49	4C	I	L
08	37	0596	0254	C9	D4	I	M	49	4D	I	M
08	38	0597	0255	C9	D5	I	N	49	4E	I	N
08	39	0598	0256	C9	D6	I	O	49	4F	I	O
08	40	0599	0257	C9	D7	I	P	49	50	I	P
08	41	0600	0258	C9	D8	I	G	49	51	I	G
08	42	0601	0259	C9	D9	I	R	49	52	I	R
08	43	0602	025A	C9	5A	I	!	49	5D	I	]
08	44	0603	025B	C9	5B	I	$	49	24	I	$
08	45	0604	025C	C9	5C	I	*	49	2A	I	*
08	46	0605	025D	C9	5D	I	)	49	29	I	)
08	47	0606	025E	C9	5E	I	;	49	3B	I	;
08	48	0607	025F	C9	5F	I	¬	49	5E	I	↑
08	49	0608	0260	C9	60	I	-	49	2D	I	-
08	50	0609	0261	C9	61	I	/	49	2F	I	/
08	51	0610	0262	C9	E2	I	S	49	53	I	S
08	52	0611	0263	C9	E3	I	T	49	54	I	T

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
08	53	0612	0264	C9	E4	I	U	49	55	I	U
08	54	0613	0265	C9	E5	I	V	49	56	I	V
08	55	0614	0266	C9	E6	I	W	49	57	I	W
08	56	0615	0267	C9	E7	I	X	49	58	I	X
08	57	0616	0268	C9	E8	I	Y	49	59	I	Y
08	58	0617	0269	C9	E9	I	Z	49	5A	I	Z
08	59	0618	026A	C9	6A	I	¦	49	7C	I	¦
08	60	0619	026B	C9	6B	I	,	49	2C	I	,
08	61	0620	026C	C9	6C	I	%	49	25	I	%
08	62	0621	026D	C9	6D	I	_	49	5F	I	_
08	63	0622	026E	C9	6E	I	>	49	3E	I	>
08	64	0623	026F	C9	6F	I	?	49	3F	I	?
08	65	0624	0270	C9	F0	I	0	49	30	I	0
08	66	0625	0271	C9	F1	I	1	49	31	I	1
08	67	0626	0272	C9	F2	I	2	49	32	I	2
08	68	0627	0273	C9	F3	I	3	49	33	I	3
08	69	0628	0274	C9	F4	I	4	49	34	I	4
08	70	0629	0275	C9	F5	I	5	49	35	I	5
08	71	0630	0276	C9	F6	I	6	49	36	I	6
08	72	0631	0277	C9	F7	I	7	49	37	I	7
08	73	0632	0278	C9	F8	I	8	49	38	I	8
08	74	0633	0279	C9	F9	I	9	49	39	I	9
08	75	0634	027A	C9	7A	I	:	49	3A	I	:
08	76	0635	027B	C9	7B	I	#	49	23	I	#
08	77	0636	027C	C9	7C	I	@	49	40	I	@
08	78	0637	027D	C9	7D	I	'	49	27	I	'
08	79	0638	027E	C9	7E	I	=	49	3D	I	=
08	80	0639	027F	C9	7F	I	"	49	22	I	"
09	01	0640	0280	4A	40	Ø	□	5B	20	[	□
09	02	0641	0281	4A	C1	Ø	A	5B	41	[	A
09	03	0642	0282	4A	C2	Ø	B	5B	42	[	B
09	04	0643	0283	4A	C3	Ø	C	5B	43	[	C
09	05	0644	0284	4A	C4	Ø	D	5B	44	[	D
09	06	0645	0285	4A	C5	Ø	E	5B	45	[	E
09	07	0646	0286	4A	C6	Ø	F	5B	46	[	F
09	08	0647	0287	4A	C7	Ø	G	5B	47	[	G
09	09	0648	0288	4A	C8	Ø	H	5B	48	[	H
09	10	0649	0289	4A	C9	Ø	I	5B	49	[	I
09	11	0650	028A	4A	4A	Ø	Ø	5B	5B	[	[
09	12	0651	028B	4A	4B	Ø	.	5B	2E	[	.
09	13	0652	028C	4A	4C	Ø	<	5B	3C	[	<
09	14	0653	028D	4A	4D	Ø	(	5B	28	[	(
09	15	0654	028E	4A	4E	Ø	+	5B	2B	[	+
09	16	0655	028F	4A	4F	Ø	—	5B	21	[	!
09	17	0656	0290	4A	50	Ø	&	5B	26	[	&
09	18	0657	0291	4A	D1	Ø	J	5B	4A	[	J
09	19	0658	0292	4A	D2	Ø	K	5B	4B	[	K
09	20	0659	0293	4A	D3	Ø	L	5B	4C	[	L
09	21	0660	0294	4A	D4	Ø	M	5B	4D	[	M
09	22	0661	0295	4A	D5	Ø	N	5B	4E	[	N
09	23	0662	0296	4A	D6	Ø	O	5B	4F	[	O
09	24	0663	0297	4A	D7	Ø	P	5B	50	[	P
09	25	0664	0298	4A	D8	Ø	Q	5B	51	[	Q
09	26	0665	0299	4A	D9	Ø	R	5B	52	[	R
09	27	0666	029A	4A	5A	Ø	!	5B	5D	[	]
09	28	0667	029B	4A	5B	Ø	$	5B	24	[	$
09	29	0668	029C	4A	5C	Ø	*	5B	2A	[	*
09	30	0669	029D	4A	5D	Ø	)	5B	29	[	)
09	31	0670	029E	4A	5E	Ø	;	5B	3B	[	;
09	32	0671	029F	4A	5F	Ø	¬	5B	5E	[	↑
09	33	0672	02A0	4A	60	Ø	-	5B	2D	[	-
09	34	0673	02A1	4A	61	Ø	/	5B	2F	[	/

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char	ASCII		Char
09	35	0674	02A2	4A	E2	S	5B	53	[S
09	36	0675	02A3	4A	E3	T	5B	54	[T
09	37	0676	02A4	4A	E4	U	5B	55	[U
09	38	0677	02A5	4A	E5	V	5B	56	[V
09	39	0678	02A6	4A	E6	W	5B	57	[W
09	40	0679	02A7	4A	E7	X	5B	58	[X
09	41	0680	02A8	4A	E8	Y	5B	59	[Y
09	42	0681	02A9	4A	E9	Z	5B	5A	[Z
09	43	0682	02AA	4A	6A	¦	5B	7C	[¦
09	44	0683	02AB	4A	6B	,	5B	2C	[,
09	45	0684	02AC	4A	6C	%	5B	25	[%
09	46	0685	02AD	4A	6D	_	5B	5F	[_
09	47	0686	02AE	4A	6E	>	5B	3E	[>
09	48	0687	02AF	4A	6F	?	5B	3F	[?
09	49	0688	02B0	4A	F0	0	5B	30	[0
09	50	0689	02B1	4A	F1	1	5B	31	[1
09	51	0690	02B2	4A	F2	2	5B	32	[2
09	52	0691	02B3	4A	F3	3	5B	33	[3
09	53	0692	02B4	4A	F4	4	5B	34	[4
09	54	0693	02B5	4A	F5	5	5B	35	[5
09	55	0694	02B6	4A	F6	6	5B	36	[6
09	56	0695	02B7	4A	F7	7	5B	37	[7
09	57	0696	02B8	4A	F8	8	5B	38	[8
09	58	0697	02B9	4A	F9	9	5B	39	[9
09	59	0698	02BA	4A	7A	:	5B	3A	[:
09	60	0699	02BB	4A	7B	#	5B	23	[#
09	61	0700	02BC	4A	7C	@	5B	40	[@
09	62	0701	02BD	4A	7D	'	5B	27	['
09	63	0702	02BE	4A	7E	=	5B	3D	[=
09	64	0703	02BF	4A	7F	"	5B	22	["
09	65	0704	02C0	4B	40	.	□	2E	20 . □
09	66	0705	02C1	4B	C1	.	A	2E	41 . A
09	67	0706	02C2	4B	C2	.	B	2E	42 . B
09	68	0707	02C3	4B	C3	.	C	2E	43 . C
09	69	0708	02C4	4B	C4	.	D	2E	44 . D
09	70	0709	02C5	4B	C5	.	E	2E	45 . E
09	71	0710	02C6	4B	C6	.	F	2E	46 . F
09	72	0711	02C7	4B	C7	.	G	2E	47 . G
09	73	0712	02C8	4B	C8	.	H	2E	48 . H
09	74	0713	02C9	4B	C9	.	I	2E	49 . I
09	75	0714	02CA	4B	4A	.	¢	2E	5B . [
09	76	0715	02CB	4B	4B	.	.	2E	2E . .
09	77	0716	02CC	4B	4C	.	<	2E	3C . <
09	78	0717	02CD	4B	4D	.	(	2E	28 . (
09	79	0718	02CE	4B	4E	.	+	2E	2B . +
09	80	0719	02CF	4B	4F	.	|	2E	21 . !
10	01	0720	02D0	4B	50	.	&	2E	26 . &
10	02	0721	02D1	4B	D1	.	J	2E	4A . J
10	03	0722	02D2	4B	D2	.	K	2E	4B . K
10	04	0723	02D3	4B	D3	.	L	2E	4C . L
10	05	0724	02D4	4B	D4	.	M	2E	4D . M
10	06	0725	02D5	4B	D5	.	N	2E	4E . N
10	07	0726	02D6	4B	D6	.	O	2E	4F . O
10	08	0727	02D7	4B	D7	.	P	2E	50 . P
10	09	0728	02D8	4B	D8	.	Q	2E	51 . Q
10	10	0729	02D9	4B	D9	.	R	2E	52 . R
10	11	0730	02DA	4B	5A	.	!	2E	5D .]
10	12	0731	02DB	4B	5B	.	$	2E	24 . $
10	13	0732	02DC	4B	5C	.	*	2E	2A . *
10	14	0733	02DD	4B	5D	.	)	2E	29 .)
10	15	0734	02DE	4B	5E	.	;	2E	3B . ;
10	16	0735	02DF	4B	5F	.	¬	2E	5E . ↑

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char	ASCII		Char
10	17	0736	02E0	4B	60	. -	2E	2D	. -
10	18	0737	02E1	4B	61	. /	2E	2F	. /
10	19	0738	02E2	4B	E2	. S	2E	53	. S
10	20	0739	02E3	4B	E3	. T	2E	54	. T
10	21	0740	02E4	4B	E4	. U	2E	55	. U
10	22	0741	02E5	4B	E5	. V	2E	56	. V
10	23	0742	02E6	4B	E6	. W	2E	57	. W
10	24	0743	02E7	4B	E7	. X	2E	58	. X
10	25	0744	02E8	4B	F8	. Y	2E	59	. Y
10	26	0745	02E9	4B	F9	. Z	2E	5A	. Z
10	27	0746	02EA	4B	6A	. ¦	2E	7C	. ¦
10	28	0747	02EB	4B	6B	. ,	2E	2C	. ,
10	29	0748	02EC	4B	6C	. %	2E	25	. %
10	30	0749	02ED	4B	6D	. _	2E	5F	. _
10	31	0750	02EE	4B	6E	. >	2E	3E	. >
10	32	0751	02EF	4B	6F	. ?	2E	3F	. ?
10	33	0752	02F0	4B	F0	. 0	2E	30	. 0
10	34	0753	02F1	4B	F1	. 1	2E	31	. 1
10	35	0754	02F2	4B	F2	. 2	2E	32	. 2
10	36	0755	02F3	4B	F3	. 3	2E	33	. 3
10	37	0756	02F4	4B	F4	. 4	2E	34	. 4
10	38	0757	02F5	4B	F5	. 5	2E	35	. 5
10	39	0758	02F6	4B	F6	. 6	2E	36	. 6
10	40	0759	02F7	4B	F7	. 7	2E	37	. 7
10	41	0760	02F8	4B	F8	. 8	2E	38	. 8
10	42	0761	02F9	4B	F9	. 9	2E	39	. 9
10	43	0762	02FA	4B	7A	. :	2E	3A	. :
10	44	0763	02FB	4B	7B	. #	2E	23	. #
10	45	0764	02FC	4B	7C	. @	2E	40	. @
10	46	0765	02FD	4B	7D	. '	2E	27	. '
10	47	0766	02FE	4B	7E	. =	2E	3D	. =
10	48	0767	02FF	4B	7F	. "	2E	22	. "
10	49	0768	0300	4C	40	< □	3C	20	< □
10	50	0769	0301	4C	C1	< A	3C	41	< A
10	51	0770	0302	4C	C2	< B	3C	42	< B
10	52	0771	0303	4C	C3	< C	3C	43	< C
10	53	0772	0304	4C	C4	< D	3C	44	< D
10	54	0773	0305	4C	C5	< E	3C	45	< E
10	55	0774	0306	4C	C6	< F	3C	46	< F
10	56	0775	0307	4C	C7	< G	3C	47	< G
10	57	0776	0308	4C	C8	< H	3C	48	< H
10	58	0777	0309	4C	C9	< I	3C	49	< I
10	59	0778	030A	4C	4A	< ¢	3C	5B	< [
10	60	0779	030B	4C	4B	< .	3C	2E	< .
10	61	0780	030C	4C	4C	< <	3C	3C	< <
10	62	0781	030D	4C	4D	< (	3C	28	< (
10	63	0782	030E	4C	4E	< +	3C	2B	< +
10	64	0783	030F	4C	4F	< ¦	3C	21	< !
10	65	0784	0310	4C	50	< &	3C	26	< &
10	66	0785	0311	4C	D1	< J	3C	4A	< J
10	67	0786	0312	4C	D2	< K	3C	4B	< K
10	68	0787	0313	4C	D3	< L	3C	4C	< L
10	69	0788	0314	4C	D4	< M	3C	4D	< M
10	70	0789	0315	4C	D5	< N	3C	4E	< N
10	71	0790	0316	4C	D6	< O	3C	4F	< O
10	72	0791	0317	4C	D7	< P	3C	50	< P
10	73	0792	0318	4C	D8	< Q	3C	51	< Q
10	74	0793	0319	4C	D9	< R	3C	52	< R
10	75	0794	031A	4C	5A	< !	3C	5D	<]
10	76	0795	031B	4C	5B	< $	3C	24	< $
10	77	0796	031C	4C	5C	< *	3C	2A	< *
10	78	0797	031D	4C	5D	<)	3C	29	<)

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
10	79	0798	031E	4C	5F	<	;	3C	3B	<	;
10	80	0799	031F	4C	5F	<	¬	3C	5E	<	↑
11	01	0800	0320	4C	60	<	-	3C	2D	<	-
11	02	0801	0321	4C	61	<	/	3C	2F	<	/
11	03	0802	0322	4C	E2	<	S	3C	53	<	S
11	04	0803	0323	4C	E3	<	T	3C	54	<	T
11	05	0804	0324	4C	E4	<	U	3C	55	<	U
11	06	0805	0325	4C	E5	<	V	3C	56	<	V
11	07	0806	0326	4C	E6	<	W	3C	57	<	W
11	08	0807	0327	4C	E7	<	X	3C	58	<	X
11	09	0808	0328	4C	E8	<	Y	3C	59	<	Y
11	10	0809	0329	4C	E9	<	Z	3C	5A	<	Z
11	11	0810	032A	4C	6A	<	¦	3C	7C	<	¦
11	12	0811	032B	4C	6B	<	,	3C	2C	<	,
11	13	0812	032C	4C	6C	<	%	3C	25	<	%
11	14	0813	032D	4C	6D	<	_	3C	5F	<	_
11	15	0814	032E	4C	6E	<	>	3C	3E	<	>
11	16	0815	032F	4C	6F	<	?	3C	3F	<	?
11	17	0816	0330	4C	F0	<	0	3C	30	<	0
11	18	0817	0331	4C	F1	<	1	3C	31	<	1
11	19	0818	0332	4C	F2	<	2	3C	32	<	2
11	20	0819	0333	4C	F3	<	3	3C	33	<	3
11	21	0820	0334	4C	F4	<	4	3C	34	<	4
11	22	0821	0335	4C	F5	<	5	3C	35	<	5
11	23	0822	0336	4C	F6	<	6	3C	36	<	6
11	24	0823	0337	4C	F7	<	7	3C	37	<	7
11	25	0824	0338	4C	F8	<	8	3C	38	<	8
11	26	0825	0339	4C	F9	<	9	3C	39	<	9
11	27	0826	033A	4C	7A	<	:	3C	3A	<	:
11	28	0827	033B	4C	7B	<	#	3C	23	<	#
11	29	0828	033C	4C	7C	<	@	3C	40	<	@
11	30	0829	033D	4C	7D	<	'	3C	27	<	'
11	31	0830	033E	4C	7E	<	=	3C	3D	<	=
11	32	0831	033F	4C	7F	<	"	3C	22	<	"
11	33	0832	0340	4D	40	(	□	28	20	(	□
11	34	0833	0341	4D	C1	(	A	28	41	(	A
11	35	0834	0342	4D	C2	(	B	28	42	(	B
11	36	0835	0343	4D	C3	(	C	28	43	(	C
11	37	0836	0344	4D	C4	(	D	28	44	(	D
11	38	0837	0345	4D	C5	(	E	28	45	(	E
11	39	0838	0346	4D	C6	(	F	28	46	(	F
11	40	0839	0347	4D	C7	(	G	28	47	(	G
11	41	0840	0348	4D	C8	(	H	28	48	(	H
11	42	0841	0349	4D	C9	(	I	28	49	(	I
11	43	0842	034A	4D	4A	(	¢	28	5B	(	[
11	44	0843	034B	4D	4B	(	.	28	2E	(	.
11	45	0844	034C	4D	4C	(	<	28	3C	(	<
11	46	0845	034D	4D	4D	(	(	28	28	(	(
11	47	0846	034E	4D	4E	(	+	28	2B	(	+
11	48	0847	034F	4D	4F	(	|	28	21	(	!
11	49	0848	0350	4D	50	(	&	28	26	(	&
11	50	0849	0351	4D	D1	(	J	28	4A	(	J
11	51	0850	0352	4D	D2	(	K	28	4B	(	K
11	52	0851	0353	4D	D3	(	L	28	4C	(	L
11	53	0852	0354	4D	D4	(	M	28	4D	(	M
11	54	0853	0355	4D	D5	(	N	28	4E	(	N
11	55	0854	0356	4D	D6	(	O	28	4F	(	O
11	56	0855	0357	4D	D7	(	P	28	50	(	P
11	57	0856	0358	4D	D8	(	Q	28	51	(	Q
11	58	0857	0359	4D	D9	(	R	28	52	(	R
11	59	0858	035A	4D	5A	(	!	28	5D	(	]
11	60	0859	035B	4D	5B	(	$	28	24	(	$

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
11	61	0860	035C	4D	5C	(	*	28	2A	(	*
11	62	0861	035D	4D	5D	(	)	28	29	(	)
11	63	0862	035E	4D	5E	(	;	28	3B	(	;
11	64	0863	035F	4D	5F	(	¬	28	5E	(	↑
11	65	0864	0360	4D	60	(	-	28	2D	(	-
11	66	0865	0361	4D	61	(	/	28	2F	(	/
11	67	0866	0362	4D	E2	(	S	28	53	(	S
11	68	0867	0363	4D	E3	(	T	28	54	(	T
11	69	0868	0364	4D	E4	(	U	28	55	(	U
11	70	0869	0365	4D	E5	(	V	28	56	(	V
11	71	0870	0366	4D	E6	(	W	28	57	(	W
11	72	0871	0367	4D	E7	(	X	28	58	(	X
11	73	0872	0368	4D	E8	(	Y	28	59	(	Y
11	74	0873	0369	4D	E9	(	Z	28	5A	(	Z
11	75	0874	036A	4D	6A	(	¦	28	7C	(	¦
11	76	0875	036B	4D	6B	(	,	28	2C	(	,
11	77	0876	036C	4D	6C	(	%	28	25	(	%
11	78	0877	036D	4D	6D	(	_	28	5F	(	_
11	79	0878	036E	4D	6E	(	>	28	3E	(	>
11	80	0879	036F	4D	6F	(	?	28	3F	(	?
12	01	0880	0370	4D	F0	(	0	28	30	(	0
12	02	0881	0371	4D	F1	(	1	28	31	(	1
12	03	0882	0372	4D	F2	(	2	28	32	(	2
12	04	0883	0373	4D	F3	(	3	28	33	(	3
12	05	0884	0374	4D	F4	(	4	28	34	(	4
12	06	0885	0375	4D	F5	(	5	28	35	(	5
12	07	0886	0376	4D	F6	(	6	28	36	(	6
12	08	0887	0377	4D	F7	(	7	28	37	(	7
12	09	0888	0378	4D	F8	(	8	28	38	(	8
12	10	0889	0379	4D	F9	(	9	28	39	(	9
12	11	0890	037A	4D	7A	(	:	28	3A	(	:
12	12	0891	037B	4D	7B	(	#	28	23	(	#
12	13	0892	037C	4D	7C	(	@	28	40	(	@
12	14	0893	037D	4D	7D	(	'	28	27	(	'
12	15	0894	037E	4D	7E	(	=	28	3D	(	=
12	16	0895	037F	4D	7F	(	"	28	22	(	"
12	17	0896	0380	4E	40	+	□	2B	20	+	□
12	18	0897	0381	4E	C1	+	A	2B	41	+	A
12	19	0898	0382	4E	C2	+	B	2B	42	+	B
12	20	0899	0383	4E	C3	+	C	2B	43	+	C
12	21	0900	0384	4E	C4	+	D	2B	44	+	D
12	22	0901	0385	4E	C5	+	E	2B	45	+	E
12	23	0902	0386	4E	C6	+	F	2B	46	+	F
12	24	0903	0387	4E	C7	+	G	2B	47	+	G
12	25	0904	0388	4E	C8	+	H	2B	48	+	H
12	26	0905	0389	4E	C9	+	I	2B	49	+	I
12	27	0906	038A	4E	4A	+	¢	2B	5B	+	[
12	28	0907	038B	4E	4B	+	.	2B	2E	+	.
12	29	0908	038C	4E	4C	+	<	2B	3C	+	<
12	30	0909	038D	4E	4D	+	(	2B	28	+	(
12	31	0910	038E	4E	4E	+	+	2B	2B	+	+
12	32	0911	038F	4E	4F	+	|	2B	21	+	!
12	33	0912	0390	4E	50	+	&	2B	26	+	&
12	34	0913	0391	4E	D1	+	J	2B	4A	+	J
12	35	0914	0392	4E	D2	+	K	2B	4B	+	K
12	36	0915	0393	4E	D3	+	L	2B	4C	+	L
12	37	0916	0394	4E	D4	+	M	2B	4D	+	M
12	38	0917	0395	4E	D5	+	N	2B	4E	+	N
12	39	0918	0396	4E	D6	+	O	2B	4F	+	O

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
12	40	0919	0397	4E	D7	+	P	2B	50	+	P
12	41	0920	0398	4E	D8	+	Q	2B	51	+	Q
12	42	0921	0399	4E	D9	+	R	2B	52	+	R
12	43	0922	039A	4E	5A	+	!	2B	5D	+	]
12	44	0923	039B	4E	5B	+	$	2B	24	+	$
12	45	0924	039C	4E	5C	+	*	2B	2A	+	*
12	46	0925	039D	4E	5D	+	)	2B	29	+	)
12	47	0926	039E	4E	5E	+	;	2B	3B	+	;
12	48	0927	039F	4E	5F	+	¬	2B	5E	+	↑
12	49	0928	03A0	4E	60	+	-	2B	2D	+	-
12	50	0929	03A1	4E	61	+	/	2B	2F	+	/
12	51	0930	03A2	4E	E2	+	S	2B	53	+	S
12	52	0931	03A3	4E	E3	+	T	2B	54	+	T
12	53	0932	03A4	4E	E4	+	U	2B	55	+	U
12	54	0933	03A5	4E	E5	+	V	2B	56	+	V
12	55	0934	03A6	4E	E6	+	W	2B	57	+	W
12	56	0935	03A7	4E	E7	+	X	2B	58	+	X
12	57	0936	03A8	4E	E8	+	Y	2B	59	+	Y
12	58	0937	03A9	4E	E9	+	Z	2B	5A	+	Z
12	59	0938	03AA	4E	6A	+	¦	2B	7C	+	¦
12	60	0939	03AB	4E	6B	+	,	2B	2C	+	,
12	61	0940	03AC	4E	6C	+	%	2B	25	+	%
12	62	0941	03AD	4E	6D	+	_	2B	5F	+	_
12	63	0942	03AE	4E	6E	+	>	2B	3E	+	>
12	64	0943	03AF	4E	6F	+	?	2B	3F	+	?
12	65	0944	03B0	4E	F0	+	0	2B	30	+	0
12	66	0945	03B1	4E	F1	+	1	2B	31	+	1
12	67	0946	03B2	4E	F2	+	2	2B	32	+	2
12	68	0947	03B3	4E	F3	+	3	2B	33	+	3
12	69	0948	03B4	4E	F4	+	4	2B	34	+	4
12	70	0949	03B5	4E	F5	+	5	2B	35	+	5
12	71	0950	03B6	4E	F6	+	6	2B	36	+	6
12	72	0951	03B7	4E	F7	+	7	2B	37	+	7
12	73	0952	03B8	4E	F8	+	8	2B	38	+	8
12	74	0953	03B9	4E	F9	+	9	2B	39	+	9
12	75	0954	03BA	4E	7A	+	:	2B	3A	+	:
12	76	0955	03BB	4E	7B	+	#	2B	23	+	#
12	77	0956	03BC	4E	7C	+	@	2B	40	+	@
12	78	0957	03BD	4E	7D	+	'	2B	27	+	'
12	79	0958	03BE	4E	7E	+	=	2B	3D	+	=
12	80	0959	03BF	4E	7F	+	"	2B	22	+	"
13	01	0960	03C0	4F	40	¦	□	21	20	!	□
13	02	0961	03C1	4F	C1	¦	A	21	41	!	A
13	03	0962	03C2	4F	C2	¦	B	21	42	!	B
13	04	0963	03C3	4F	C3	¦	C	21	43	!	C
13	05	0964	03C4	4F	C4	¦	D	21	44	!	D
13	06	0965	03C5	4F	C5	¦	E	21	45	!	E
13	07	0966	03C6	4F	C6	¦	F	21	46	!	F
13	08	0967	03C7	4F	C7	¦	G	21	47	!	G
13	09	0968	03C8	4F	C8	¦	H	21	48	!	H
13	10	0969	03C9	4F	C9	¦	I	21	49	!	I
13	11	0970	03CA	4F	4A	¦	¢	21	5B	!	[
13	12	0971	03CB	4F	4B	¦	.	21	2E	!	.
13	13	0972	03CC	4F	4C	¦	<	21	3C	!	<
13	14	0973	03CD	4F	4D	¦	(	21	28	!	(
13	15	0974	03CE	4F	4E	¦	+	21	2B	!	+
13	16	0975	03CF	4F	4F	¦	¦	21	21	!	!
13	17	0976	03D0	4F	50	¦	&	21	26	!	&
13	18	0977	03D1	4F	D1	¦	J	21	4A	!	J
13	19	0978	03D2	4F	D2	¦	K	21	4B	!	K
13	20	0979	03D3	4F	D3	¦	L	21	4C	!	L

RCX70		Dec	Hex	EBCDIC		Char	ASCII		Char
13	21	0980	03D4	4F	D4	M	21	4D	M
13	22	0981	03D5	4F	D5	N	21	4E	N
13	23	0982	03D6	4F	D6	O	21	4F	O
13	24	0983	03D7	4F	D7	P	21	50	P
13	25	0984	03D8	4F	D8	Q	21	51	Q
13	26	0985	03D9	4F	D9	R	21	52	R
13	27	0986	03DA	4F	5A	!	21	5D	]
13	28	0987	03DB	4F	5B	$	21	24	$
13	29	0988	03DC	4F	5C	*	21	2A	*
13	30	0989	03DD	4F	5D	)	21	29	)
13	31	0990	03DE	4F	5E	;	21	3B	;
13	32	0991	03DF	4F	5F	¬	21	5E	↑
13	33	0992	03E0	4F	60	-	21	2D	-
13	34	0993	03E1	4F	61	/	21	2F	/
13	35	0994	03E2	4F	E2	S	21	53	S
13	36	0995	03E3	4F	E3	T	21	54	T
13	37	0996	03E4	4F	E4	U	21	55	U
13	38	0997	03E5	4F	E5	V	21	56	V
13	39	0998	03E6	4F	E6	W	21	57	W
13	40	0999	03E7	4F	E7	X	21	58	X
13	41	1000	03E8	4F	E8	Y	21	59	Y
13	42	1001	03E9	4F	E9	Z	21	5A	Z
13	43	1002	03EA	4F	6A	¦	21	7C	¦
13	44	1003	03EB	4F	6B	,	21	2C	,
13	45	1004	03EC	4F	6C	%	21	25	%
13	46	1005	03ED	4F	6D	_	21	5F	_
13	47	1006	03EE	4F	6E	>	21	3E	>
13	48	1007	03EF	4F	6F	?	21	3F	?
13	49	1008	03F0	4F	F0	0	21	30	0
13	50	1009	03F1	4F	F1	1	21	31	1
13	51	1010	03F2	4F	F2	2	21	32	2
13	52	1011	03F3	4F	F3	3	21	33	3
13	53	1012	03F4	4F	F4	4	21	34	4
13	54	1013	03F5	4F	F5	5	21	35	5
13	55	1014	03F6	4F	F6	6	21	36	6
13	56	1015	03F7	4F	F7	7	21	37	7
13	57	1016	03F8	4F	F8	8	21	38	8
13	58	1017	03F9	4F	F9	9	21	39	9
13	59	1018	03FA	4F	7A	:	21	3A	:
13	60	1019	03FB	4F	7B	#	21	23	#
13	61	1020	03FC	4F	7C	@	21	40	@
13	62	1021	03FD	4F	7D	'	21	27	'
13	63	1022	03FE	4F	7E	=	21	3D	=
13	64	1023	03FF	4F	7F	"	21	22	"
13	65	1024	0400	50	40	&	26	20	&
13	66	1025	0401	50	C1	&	26	41	&
13	67	1026	0402	50	C2	&	26	42	&
13	68	1027	0403	50	C3	&	26	43	&
13	69	1028	0404	50	C4	&	26	44	&
13	70	1029	0405	50	C5	&	26	45	&
13	71	1030	0406	50	C6	&	26	46	&
13	72	1031	0407	50	C7	&	26	47	&
13	73	1032	0408	50	C8	&	26	48	&
13	74	1033	0409	50	C9	&	26	49	&
13	75	1034	040A	50	4A	&	26	5B	&
13	76	1035	040B	50	4B	&	26	2E	&
13	77	1036	040C	50	4C	&	26	3C	&
13	78	1037	040D	50	4D	&	26	28	&
13	79	1038	040E	50	4E	&	26	2B	&
13	80	1039	040F	50	4F	&	26	21	&
14	01	1040	0410	50	50	&	26	26	&

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
14	02	1041	0411	50	D1	&	J	26	4A	&	J
14	03	1042	0412	50	D2	&	K	26	4B	&	K
14	04	1043	0413	50	D3	&	L	26	4C	&	L
14	05	1044	0414	50	D4	&	M	26	4D	&	M
14	06	1045	0415	50	D5	&	N	26	4E	&	N
14	07	1046	0416	50	D6	&	O	26	4F	&	O
14	08	1047	0417	50	D7	&	P	26	50	&	P
14	09	1048	0418	50	D8	&	Q	26	51	&	Q
14	10	1049	0419	50	D9	&	R	26	52	&	R
14	11	1050	041A	50	5A	&	!	26	5D	&	]
14	12	1051	041B	50	5B	&	$	26	24	&	$
14	13	1052	041C	50	5C	&	*	26	2A	&	*
14	14	1053	041D	50	5D	&	)	26	29	&	)
14	15	1054	041E	50	5E	&	;	26	3B	&	;
14	16	1055	041F	50	5F	&	¬	26	5E	&	↑
14	17	1056	0420	50	60	&	-	26	2D	&	-
14	18	1057	0421	50	61	&	/	26	2F	&	/
14	19	1058	0422	50	E2	&	S	26	53	&	S
14	20	1059	0423	50	E3	&	T	26	54	&	T
14	21	1060	0424	50	E4	&	U	26	55	&	U
14	22	1061	0425	50	E5	&	V	26	56	&	V
14	23	1062	0426	50	E6	&	W	26	57	&	W
14	24	1063	0427	50	E7	&	X	26	58	&	X
14	25	1064	0428	50	E8	&	Y	26	59	&	Y
14	26	1065	0429	50	E9	&	Z	26	5A	&	Z
14	27	1066	042A	50	6A	&	I	26	7C	&	I
14	28	1067	042B	50	6B	&	,	26	2C	&	,
14	29	1068	042C	50	6C	&	%	26	25	&	%
14	30	1069	042D	50	6D	&	_	26	5F	&	_
14	31	1070	042E	50	6E	&	>	26	3E	&	>
14	32	1071	042F	50	6F	&	?	26	3F	&	?
14	33	1072	0430	50	F0	&	0	26	30	&	0
14	34	1073	0431	50	F1	&	1	26	31	&	1
14	35	1074	0432	50	F2	&	2	26	32	&	2
14	36	1075	0433	50	F3	&	3	26	33	&	3
14	37	1076	0434	50	F4	&	4	26	34	&	4
14	38	1077	0435	50	F5	&	5	26	35	&	5
14	39	1078	0436	50	F6	&	6	26	36	&	6
14	40	1079	0437	50	F7	&	7	26	37	&	7
14	41	1080	0438	50	F8	&	8	26	38	&	8
14	42	1081	0439	50	F9	&	9	26	39	&	9
14	43	1082	043A	50	7A	&	:	26	3A	&	:
14	44	1083	043B	50	7B	&	#	26	23	&	#
14	45	1084	043C	50	7C	&	@	26	40	&	@
14	46	1085	043D	50	7D	&	'	26	27	&	'
14	47	1086	043E	50	7E	&	=	26	3D	&	=
14	48	1087	043F	50	7F	&	"	26	22	&	"
14	49	1088	0440	D1	40	J	□	4A	20	J	□
14	50	1089	0441	D1	C1	J	A	4A	41	J	A
14	51	1090	0442	D1	C2	J	B	4A	42	J	B
14	52	1091	0443	D1	C3	J	C	4A	43	J	C
14	53	1092	0444	D1	C4	J	D	4A	44	J	D
14	54	1093	0445	D1	C5	J	E	4A	45	J	E
14	55	1094	0446	D1	C6	J	F	4A	46	J	F
14	56	1095	0447	D1	C7	J	G	4A	47	J	G
14	57	1096	0448	D1	C8	J	H	4A	48	J	H
14	58	1097	0449	D1	C9	J	I	4A	49	J	I
14	59	1098	044A	D1	4A	J	¢	4A	5B	J	[
14	60	1099	044B	D1	4B	J	.	4A	2E	J	.
14	61	1100	044C	D1	4C	J	<	4A	3C	J	<
14	62	1101	044D	D1	4D	J	(	4A	28	J	(

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
14	63	1102	044E	D1	4E	J	+	4A	2B	J	+
14	64	1103	044F	D1	4F	J	|	4A	21	J	!
14	65	1104	0450	D1	50	J	&	4A	26	J	&
14	66	1105	0451	D1	D1	J	J	4A	4A	J	J
14	67	1106	0452	D1	D2	J	K	4A	4B	J	K
14	68	1107	0453	D1	D3	J	L	4A	4C	J	L
14	69	1108	0454	D1	D4	J	M	4A	4D	J	M
14	70	1109	0455	D1	D5	J	N	4A	4E	J	N
14	71	1110	0456	D1	D6	J	O	4A	4F	J	O
14	72	1111	0457	D1	D7	J	P	4A	50	J	P
14	73	1112	0458	D1	D8	J	Q	4A	51	J	Q
14	74	1113	0459	D1	D9	J	R	4A	52	J	R
14	75	1114	045A	D1	5A	J	!	4A	5D	J	]
14	76	1115	045B	D1	5B	J	$	4A	24	J	$
14	77	1116	045C	D1	5C	J	*	4A	2A	J	*
14	78	1117	045D	D1	5D	J	)	4A	29	J	)
14	79	1118	045E	D1	5E	J	;	4A	3B	J	;
14	80	1119	045F	D1	5F	J	¬	4A	5E	J	↑
15	01	1120	0460	D1	60	J	-	4A	2D	J	-
15	02	1121	0461	D1	61	J	/	4A	2F	J	/
15	03	1122	0462	D1	E2	J	S	4A	53	J	S
15	04	1123	0463	D1	E3	J	T	4A	54	J	T
15	05	1124	0464	D1	E4	J	U	4A	55	J	U
15	06	1125	0465	D1	E5	J	V	4A	56	J	V
15	07	1126	0466	D1	E6	J	W	4A	57	J	W
15	08	1127	0467	D1	E7	J	X	4A	58	J	X
15	09	1128	0468	D1	E8	J	Y	4A	59	J	Y
15	10	1129	0469	D1	E9	J	Z	4A	5A	J	Z
15	11	1130	046A	D1	6A	J	|	4A	7C	J	|
15	12	1131	046B	D1	6B	J	,	4A	2C	J	,
15	13	1132	046C	D1	6C	J	%	4A	25	J	%
15	14	1133	046D	D1	6D	J	_	4A	5F	J	_
15	15	1134	046E	D1	6E	J	>	4A	3E	J	>
15	16	1135	046F	D1	6F	J	?	4A	3F	J	?
15	17	1136	0470	D1	F0	J	0	4A	30	J	0
15	18	1137	0471	D1	F1	J	1	4A	31	J	1
15	19	1138	0472	D1	F2	J	2	4A	32	J	2
15	20	1139	0473	D1	F3	J	3	4A	33	J	3
15	21	1140	0474	D1	F4	J	4	4A	34	J	4
15	22	1141	0475	D1	F5	J	5	4A	35	J	5
15	23	1142	0476	D1	F6	J	6	4A	36	J	6
15	24	1143	0477	D1	F7	J	7	4A	37	J	7
15	25	1144	0478	D1	F8	J	8	4A	38	J	8
15	26	1145	0479	D1	F9	J	9	4A	39	J	9
15	27	1146	047A	D1	7A	J	:	4A	3A	J	:
15	28	1147	047B	D1	7B	J	#	4A	23	J	#
15	29	1148	047C	D1	7C	J	@	4A	40	J	@
15	30	1149	047D	D1	7D	J	'	4A	27	J	'
15	31	1150	047E	D1	7E	J	=	4A	3D	J	=
15	32	1151	047F	D1	7F	J	"	4A	22	J	"
15	33	1152	0480	D2	40	K	□	4B	20	K	□
15	34	1153	0481	D2	C1	K	A	4B	41	K	A
15	35	1154	0482	D2	C2	K	B	4B	42	K	B
15	36	1155	0483	D2	C3	K	C	4B	43	K	C
15	37	1156	0484	D2	C4	K	D	4B	44	K	D
15	38	1157	0485	D2	C5	K	E	4B	45	K	E
15	39	1158	0486	D2	C6	K	F	4B	46	K	F
15	40	1159	0487	D2	C7	K	G	4B	47	K	G
15	41	1160	0488	D2	C8	K	H	4B	48	K	H
15	42	1161	0489	D2	C9	K	I	4B	49	K	I
15	43	1162	048A	D2	4A	K	Ø	4B	5B	K	[

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
15	44	1163	048B	D2	4B	K	.	4B	2E	K	.
15	45	1164	048C	D2	4C	K	<	4B	3C	K	<
15	46	1165	048D	D2	4D	K	(	4B	28	K	(
15	47	1166	048E	D2	4E	K	+	4B	2B	K	+
15	48	1167	048F	D2	4F	K	|	4B	21	K	!
15	49	1168	0490	D2	50	K	&	4B	26	K	&
15	50	1169	0491	D2	D1	K	J	4B	4A	K	J
15	51	1170	0492	D2	D2	K	K	4B	4B	K	K
15	52	1171	0493	D2	D3	K	L	4B	4C	K	L
15	53	1172	0494	D2	D4	K	M	4B	4D	K	M
15	54	1173	0495	D2	D5	K	N	4B	4E	K	N
15	55	1174	0496	D2	D6	K	O	4B	4F	K	O
15	56	1175	0497	D2	D7	K	P	4B	50	K	P
15	57	1176	0498	D2	D8	K	Q	4B	51	K	Q
15	58	1177	0499	D2	D9	K	R	4B	52	K	R
15	59	1178	049A	D2	5A	K	!	4B	5D	K	]
15	60	1179	049B	D2	5B	K	$	4B	24	K	$
15	61	1180	049C	D2	5C	K	*	4B	2A	K	*
15	62	1181	049D	D2	5D	K	)	4B	29	K	)
15	63	1182	049E	D2	5E	K	;	4B	3B	K	;
15	64	1183	049F	D2	5F	K	¬	4B	5E	K	↑
15	65	1184	04A0	D2	60	K	-	4B	2D	K	-
15	66	1185	04A1	D2	61	K	/	4B	2F	K	/
15	67	1186	04A2	D2	E2	K	S	4B	53	K	S
15	68	1187	04A3	D2	E3	K	T	4B	54	K	T
15	69	1188	04A4	D2	E4	K	U	4B	55	K	U
15	70	1189	04A5	D2	E5	K	V	4B	56	K	V
15	71	1190	04A6	D2	E6	K	W	4B	57	K	W
15	72	1191	04A7	D2	E7	K	X	4B	58	K	X
15	73	1192	04A8	D2	E8	K	Y	4B	59	K	Y
15	74	1193	04A9	D2	E9	K	Z	4B	5A	K	Z
15	75	1194	04AA	D2	6A	K	¦	4B	7C	K	¦
15	76	1195	04AB	D2	6B	K	,	4B	2C	K	,
15	77	1196	04AC	D2	6C	K	%	4B	25	K	%
15	78	1197	04AD	D2	6D	K	_	4B	5F	K	←
15	79	1198	04AE	D2	6E	K	>	4B	3E	K	>
15	80	1199	04AF	D2	6F	K	?	4B	3F	K	?
16	01	1200	04B0	D2	F0	K	0	4B	30	K	0
16	02	1201	04B1	D2	F1	K	1	4B	31	K	1
16	03	1202	04B2	D2	F2	K	2	4B	32	K	2
16	04	1203	04B3	D2	F3	K	3	4B	33	K	3
16	05	1204	04B4	D2	F4	K	4	4B	34	K	4
16	06	1205	04B5	D2	F5	K	5	4B	35	K	5
16	07	1206	04B6	D2	F6	K	6	4B	36	K	6
16	08	1207	04B7	D2	F7	K	7	4B	37	K	7
16	09	1208	04B8	D2	F8	K	8	4B	38	K	8
16	10	1209	04B9	D2	F9	K	9	4B	39	K	9
16	11	1210	04BA	D2	7A	K	:	4B	3A	K	:
16	12	1211	04BB	D2	7B	K	#	4B	23	K	#
16	13	1212	04BC	D2	7C	K	@	4B	40	K	@
16	14	1213	04BD	D2	7D	K	'	4B	27	K	'
16	15	1214	04BE	D2	7E	K	=	4B	3D	K	=
16	16	1215	04BF	D2	7F	K	"	4B	22	K	"
16	17	1216	04C0	D3	40	L	□	4C	20	L	□
16	18	1217	04C1	D3	C1	L	A	4C	41	L	A
16	19	1218	04C2	D3	C2	L	B	4C	42	L	B
16	20	1219	04C3	D3	C3	L	C	4C	43	L	C
16	21	1220	04C4	D3	C4	L	D	4C	44	L	D
16	22	1221	04C5	D3	C5	L	E	4C	45	L	E
16	23	1222	04C6	D3	C6	L	F	4C	46	L	F
16	24	1223	04C7	D3	C7	L	G	4C	47	L	G

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
16	25	1224	04C8	D3	C8	L	H	4C	48	L	H
16	26	1225	04C9	D3	C9	L	I	4C	49	L	I
16	27	1226	04CA	D3	4A	L	¢	4C	5B	L	[
16	28	1227	04CB	D3	4B	L	.	4C	2E	L	.
16	29	1228	04CC	D3	4C	L	<	4C	3C	L	<
16	30	1229	04CD	D3	4D	L	(	4C	28	L	(
16	31	1230	04CE	D3	4E	L	+	4C	2B	L	+
16	32	1231	04CF	D3	4F	L	|	4C	21	L	!
16	33	1232	04D0	D3	50	L	&	4C	26	L	&
16	34	1233	04D1	D3	D1	L	J	4C	4A	L	J
16	35	1234	04D2	D3	D2	L	K	4C	4B	L	K
16	36	1235	04D3	D3	D3	L	L	4C	4C	L	L
16	37	1236	04D4	D3	D4	L	M	4C	4D	L	M
16	38	1237	04D5	D3	D5	L	N	4C	4E	L	N
16	39	1238	04D6	D3	D6	L	O	4C	4F	L	O
16	40	1239	04D7	D3	D7	L	P	4C	50	L	P
16	41	1240	04D8	D3	D8	L	Q	4C	51	L	Q
16	42	1241	04D9	D3	D9	L	R	4C	52	L	R
16	43	1242	04DA	D3	5A	L	!	4C	5D	L	]
16	44	1243	04DB	D3	5B	L	$	4C	24	L	$
16	45	1244	04DC	D3	5C	L	*	4C	2A	L	*
16	46	1245	04DD	D3	5D	L	)	4C	29	L	)
16	47	1246	04DE	D3	5E	L	;	4C	3B	L	;
16	48	1247	04DF	D3	5F	L	¬	4C	5E	L	↑
16	49	1248	04E0	D3	60	L	-	4C	2D	L	-
16	50	1249	04E1	D3	61	L	/	4C	2F	L	/
16	51	1250	04E2	D3	E2	L	S	4C	53	L	S
16	52	1251	04E3	D3	E3	L	T	4C	54	L	T
16	53	1252	04E4	D3	E4	L	U	4C	55	L	U
16	54	1253	04E5	D3	E5	L	V	4C	56	L	V
16	55	1254	04E6	D3	E6	L	W	4C	57	L	W
16	56	1255	04E7	D3	E7	L	X	4C	58	L	X
16	57	1256	04E8	D3	E8	L	Y	4C	59	L	Y
16	58	1257	04E9	D3	E9	L	Z	4C	5A	L	Z
16	59	1258	04EA	D3	6A	L	|	4C	7C	L	|
16	60	1259	04EB	D3	6B	L	,	4C	2C	L	,
16	61	1260	04EC	D3	6C	L	%	4C	25	L	%
16	62	1261	04ED	D3	6D	L	_	4C	5F	L	_
16	63	1262	04EE	D3	6E	L	>	4C	3E	L	>
16	64	1263	04EF	D3	6F	L	?	4C	3F	L	?
16	65	1264	04F0	D3	F0	L	0	4C	30	L	0
16	66	1265	04F1	D3	F1	L	1	4C	31	L	1
16	67	1266	04F2	D3	F2	L	2	4C	32	L	2
16	68	1267	04F3	D3	F3	L	3	4C	33	L	3
16	69	1268	04F4	D3	F4	L	4	4C	34	L	4
16	70	1269	04F5	D3	F5	L	5	4C	35	L	5
16	71	1270	04F6	D3	F6	L	6	4C	36	L	6
16	72	1271	04F7	D3	F7	L	7	4C	37	L	7
16	73	1272	04F8	D3	F8	L	8	4C	38	L	8
16	74	1273	04F9	D3	F9	L	9	4C	39	L	9
16	75	1274	04FA	D3	7A	L	:	4C	3A	L	:
16	76	1275	04FB	D3	7B	L	#	4C	23	L	#
16	77	1276	04FC	D3	7C	L	@	4C	40	L	@
16	78	1277	04FD	D3	7D	L	'	4C	27	L	'
16	79	1278	04FE	D3	7E	L	=	4C	3D	L	=
16	80	1279	04FF	D3	7F	L	"	4C	22	L	"
17	01	1280	0500	D4	40	M	□	4D	20	M	□
17	02	1281	0501	D4	C1	M	A	4D	41	M	A
17	03	1282	0502	D4	C2	M	B	4D	42	M	B
17	04	1283	0503	D4	C3	M	C	4D	43	M	C
17	05	1284	0504	D4	C4	M	D	4D	44	M	D
17	06	1285	0505	D4	C5	M	E	4D	45	M	E

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
17	07	1286	0506	D4	C6	M	F	4D	46	M	F
17	08	1287	0507	D4	C7	M	G	4D	47	M	G
17	09	1288	0508	D4	C8	M	H	4D	48	M	H
17	10	1289	0509	D4	C9	M	I	4D	49	M	I
17	11	1290	050A	D4	4A	M	¢	4D	5B	M	[
17	12	1291	050B	D4	4B	M	.	4D	2E	M	.
17	13	1292	050C	D4	4C	M	<	4D	3C	M	<
17	14	1293	050D	D4	4D	M	(	4D	28	M	(
17	15	1294	050E	D4	4E	M	+	4D	2B	M	+
17	16	1295	050F	D4	4F	M	|	4D	21	M	!
17	17	1296	0510	D4	50	M	&	4D	26	M	&
17	18	1297	0511	D4	D1	M	J	4D	4A	M	J
17	19	1298	0512	D4	D2	M	K	4D	4B	M	K
17	20	1299	0513	D4	D3	M	L	4D	4C	M	L
17	21	1300	0514	D4	D4	M	M	4D	4D	M	M
17	22	1301	0515	D4	D5	M	N	4D	4E	M	N
17	23	1302	0516	D4	D6	M	O	4D	4F	M	O
17	24	1303	0517	D4	D7	M	P	4D	50	M	P
17	25	1304	0518	D4	D8	M	Q	4D	51	M	Q
17	26	1305	0519	D4	D9	M	R	4D	52	M	R
17	27	1306	051A	D4	5A	M	!	4D	5D	M	]
17	28	1307	051B	D4	5B	M	$	4D	24	M	$
17	29	1308	051C	D4	5C	M	*	4D	2A	M	*
17	30	1309	051D	D4	5D	M	)	4D	29	M	)
17	31	1310	051E	D4	5E	M	;	4D	3B	M	;
17	32	1311	051F	D4	5F	M	¬	4D	5E	M	↑
17	33	1312	0520	D4	60	M	-	4D	2D	M	-
17	34	1313	0521	D4	61	M	/	4D	2F	M	/
17	35	1314	0522	D4	E2	M	S	4D	53	M	S
17	36	1315	0523	D4	E3	M	T	4D	54	M	T
17	37	1316	0524	D4	E4	M	U	4D	55	M	U
17	38	1317	0525	D4	E5	M	V	4D	56	M	V
17	39	1318	0526	D4	E6	M	W	4D	57	M	W
17	40	1319	0527	D4	E7	M	X	4D	58	M	X
17	41	1320	0528	D4	E8	M	Y	4D	59	M	Y
17	42	1321	0529	D4	E9	M	Z	4D	5A	M	Z
17	43	1322	052A	D4	6A	M	¦	4D	7C	M	¦
17	44	1323	052B	D4	6B	M	,	4D	2C	M	,
17	45	1324	052C	D4	6C	M	%	4D	25	M	%
17	46	1325	052D	D4	6D	M	_	4D	5F	M	←
17	47	1326	052E	D4	6E	M	>	4D	3E	M	>
17	48	1327	052F	D4	6F	M	?	4D	3F	M	?
17	49	1328	0530	D4	F0	M	0	4D	30	M	0
17	50	1329	0531	D4	F1	M	1	4D	31	M	1
17	51	1330	0532	D4	F2	M	2	4D	32	M	2
17	52	1331	0533	D4	F3	M	3	4D	33	M	3
17	53	1332	0534	D4	F4	M	4	4D	34	M	4
17	54	1333	0535	D4	F5	M	5	4D	35	M	5
17	55	1334	0536	D4	F6	M	6	4D	36	M	6
17	56	1335	0537	D4	F7	M	7	4D	37	M	7
17	57	1336	0538	D4	F8	M	8	4D	38	M	8
17	58	1337	0539	D4	F9	M	9	4D	39	M	9
17	59	1338	053A	D4	7A	M	:	4D	3A	M	:
17	60	1339	053B	D4	7B	M	#	4D	23	M	#
17	61	1340	053C	D4	7C	M	@	4D	40	M	@
17	62	1341	053D	D4	7D	M	'	4D	27	M	'
17	63	1342	053E	D4	7E	M	=	4D	3D	M	=
17	64	1343	053F	D4	7F	M	"	4D	22	M	"
17	65	1344	0540	D5	40	N	□	4E	20	N	□
17	66	1345	0541	D5	C1	N	A	4E	41	N	A
17	67	1346	0542	D5	C2	N	B	4E	42	N	B
17	68	1347	0543	D5	C3	N	C	4E	43	N	C

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
17	69	1348	0544	D5	C4	N	D	4E	44	N	D
17	70	1349	0545	D5	C5	N	E	4E	45	N	E
17	71	1350	0546	D5	C6	N	F	4E	46	N	F
17	72	1351	0547	D5	C7	N	G	4E	47	N	G
17	73	1352	0548	D5	C8	N	H	4E	48	N	H
17	74	1353	0549	D5	C9	N	I	4E	49	N	I
17	75	1354	054A	D5	4A	N	¢	4E	5B	N	[
17	76	1355	054B	D5	4B	N	.	4E	2E	N	.
17	77	1356	054C	D5	4C	N	<	4E	3C	N	<
17	78	1357	054D	D5	4D	N	(	4E	28	N	(
17	79	1358	054E	D5	4E	N	+	4E	2B	N	+
17	80	1359	054F	D5	4F	N	|	4E	21	N	!
18	01	1360	0550	D5	50	N	&	4E	26	N	&
18	02	1361	0551	D5	D1	N	J	4E	4A	N	J
18	03	1362	0552	D5	D2	N	K	4E	4B	N	K
18	04	1363	0553	D5	D3	N	L	4E	4C	N	L
18	05	1364	0554	D5	D4	N	M	4E	4D	N	M
18	06	1365	0555	D5	D5	N	N	4E	4E	N	N
18	07	1366	0556	D5	D6	N	O	4E	4F	N	O
18	08	1367	0557	D5	D7	N	P	4E	50	N	P
18	09	1368	0558	D5	D8	N	Q	4E	51	N	Q
18	10	1369	0559	D5	D9	N	R	4E	52	N	R
18	11	1370	055A	D5	5A	N	!	4E	5D	N	]
18	12	1371	055B	D5	5B	N	$	4E	24	N	$
18	13	1372	055C	D5	5C	N	*	4E	2A	N	*
18	14	1373	055D	D5	5D	N	)	4E	29	N	)
18	15	1374	055E	D5	5E	N	;	4E	3B	N	;
18	16	1375	055F	D5	5F	N	¬	4E	5F	N	↑
18	17	1376	0560	D5	60	N	-	4E	2D	N	-
18	18	1377	0561	D5	61	N	/	4E	2F	N	/
18	19	1378	0562	D5	E2	N	S	4E	53	N	S
18	20	1379	0563	D5	E3	N	T	4E	54	N	T
18	21	1380	0564	D5	E4	N	U	4E	55	N	U
18	22	1381	0565	D5	E5	N	V	4E	56	N	V
18	23	1382	0566	D5	E6	N	W	4E	57	N	W
18	24	1383	0567	D5	E7	N	X	4E	58	N	X
18	25	1384	0568	D5	E8	N	Y	4E	59	N	Y
18	26	1385	0569	D5	E9	N	Z	4E	5A	N	Z
18	27	1386	056A	D5	6A	N	!	4E	7C	N	!
18	28	1387	056B	D5	6B	N	,	4E	2C	N	,
18	29	1388	056C	D5	6C	N	%	4E	25	N	%
18	30	1389	056D	D5	6D	N	_	4E	5F	N	_
18	31	1390	056E	D5	6E	N	>	4E	3E	N	>
18	32	1391	056F	D5	6F	N	?	4E	3F	N	?
18	33	1392	0570	D5	F0	N	0	4E	30	N	0
18	34	1393	0571	D5	F1	N	1	4E	31	N	1
18	35	1394	0572	D5	F2	N	2	4E	32	N	2
18	36	1395	0573	D5	F3	N	3	4E	33	N	3
18	37	1396	0574	D5	F4	N	4	4E	34	N	4
18	38	1397	0575	D5	F5	N	5	4E	35	N	5
18	39	1398	0576	D5	F6	N	6	4E	36	N	6
18	40	1399	0577	D5	F7	N	7	4E	37	N	7
18	41	1400	0578	D5	F8	N	8	4E	38	N	8
18	42	1401	0579	D5	F9	N	9	4E	39	N	9
18	43	1402	057A	D5	7A	N	:	4E	3A	N	:
18	44	1403	057B	D5	7B	N	#	4E	23	N	#
18	45	1404	057C	D5	7C	N	@	4E	40	N	@
18	46	1405	057D	D5	7D	N	'	4E	27	N	'
18	47	1406	057E	D5	7E	N	=	4E	3D	N	=
18	48	1407	057F	D5	7F	N	"	4E	22	N	"
18	49	1408	0580	D6	40	O	□	4F	20	O	□
18	50	1409	0581	D6	C1	O	A	4F	41	O	A

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
18	51	1410	0582	D6	C2	0	B	4F	42	0	B
18	52	1411	0583	D6	C3	0	C	4F	43	0	C
18	53	1412	0584	D6	C4	0	D	4F	44	0	D
18	54	1413	0585	D6	C5	0	E	4F	45	0	E
18	55	1414	0586	D6	C6	0	F	4F	46	0	F
18	56	1415	0587	D6	C7	0	G	4F	47	0	G
18	57	1416	0588	D6	C8	0	H	4F	48	0	H
18	58	1417	0589	D6	C9	0	I	4F	49	0	I
18	59	1418	058A	D6	4A	0	¢	4F	5B	0	[
18	60	1419	058B	D6	4B	0	.	4F	2E	0	.
18	61	1420	058C	D6	4C	0	<	4F	3C	0	<
18	62	1421	058D	D6	4D	0	(	4F	28	0	(
18	63	1422	058E	D6	4E	0	+	4F	2B	0	+
18	64	1423	058F	D6	4F	0	|	4F	21	0	!
18	65	1424	0590	D6	50	0	&	4F	26	0	&
18	66	1425	0591	D6	D1	0	J	4F	4A	0	J
18	67	1426	0592	D6	D2	0	K	4F	4B	0	K
18	68	1427	0593	D6	D3	0	L	4F	4C	0	L
18	69	1428	0594	D6	D4	0	M	4F	4D	0	M
18	70	1429	0595	D6	D5	0	N	4F	4E	0	N
18	71	1430	0596	D6	D6	0	O	4F	4F	0	O
18	72	1431	0597	D6	D7	0	P	4F	50	0	P
18	73	1432	0598	D6	D8	0	Q	4F	51	0	Q
18	74	1433	0599	D6	D9	0	R	4F	52	0	R
18	75	1434	059A	D6	5A	0	!	4F	5D	0	]
18	76	1435	059B	D6	5B	0	$	4F	24	0	$
18	77	1436	059C	D6	5C	0	*	4F	2A	0	*
18	78	1437	059D	D6	5D	0	)	4F	29	0	)
18	79	1438	059E	D6	5E	0	;	4F	3B	0	;
18	80	1439	059F	D6	5F	0	¬	4F	5E	0	↑
19	01	1440	05A0	D6	60	0	-	4F	2D	0	-
19	02	1441	05A1	D6	61	0	/	4F	2F	0	/
19	03	1442	05A2	D6	E2	0	S	4F	53	0	S
19	04	1443	05A3	D6	E3	0	T	4F	54	0	T
19	05	1444	05A4	D6	E4	0	U	4F	55	0	U
19	06	1445	05A5	D6	E5	0	V	4F	56	0	V
19	07	1446	05A6	D6	E6	0	W	4F	57	0	W
19	08	1447	05A7	D6	E7	0	X	4F	58	0	X
19	09	1448	05A8	D6	E8	0	Y	4F	59	0	Y
19	10	1449	05A9	D6	E9	0	Z	4F	5A	0	Z
19	11	1450	05AA	D6	6A	0	¦	4F	7C	0	¦
19	12	1451	05AB	D6	6B	0	,	4F	2C	0	,
19	13	1452	05AC	D6	6C	0	%	4F	25	0	%
19	14	1453	05AD	D6	6D	0	_	4F	5F	0	_
19	15	1454	05AE	D6	6E	0	>	4F	3E	0	>
19	16	1455	05AF	D6	6F	0	?	4F	3F	0	?
19	17	1456	05B0	D6	F0	0	0	4F	30	0	0
19	18	1457	05B1	D6	F1	0	1	4F	31	0	1
19	19	1458	05B2	D6	F2	0	2	4F	32	0	2
19	20	1459	05B3	D6	F3	0	3	4F	33	0	3
19	21	1460	05B4	D6	F4	0	4	4F	34	0	4
19	22	1461	05B5	D6	F5	0	5	4F	35	0	5
19	23	1462	05B6	D6	F6	0	6	4F	36	0	6
19	24	1463	05B7	D6	F7	0	7	4F	37	0	7
19	25	1464	05B8	D6	F8	0	8	4F	38	0	8
19	26	1465	05B9	D6	F9	0	9	4F	39	0	9
19	27	1466	05BA	D6	7A	0	:	4F	3A	0	:
19	28	1467	05BB	D6	7B	0	#	4F	23	0	#
19	29	1468	05BC	D6	7C	0	@	4F	40	0	@
19	30	1469	05BD	D6	7D	0	'	4F	27	0	'
19	31	1470	05BE	D6	7E	0	=	4F	3D	0	=
19	32	1471	05BF	D6	7F	0	"	4F	22	0	"

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
19	33	1472	05C0	D7	40	P	□	50	20	P	□
19	34	1473	05C1	D7	C1	P	A	50	41	P	A
19	35	1474	05C2	D7	C2	P	B	50	42	P	B
19	36	1475	05C3	D7	C3	P	C	50	43	P	C
19	37	1476	05C4	D7	C4	P	D	50	44	P	D
19	38	1477	05C5	D7	C5	P	E	50	45	P	E
19	39	1478	05C6	D7	C6	P	F	50	46	P	F
19	40	1479	05C7	D7	C7	P	G	50	47	P	G
19	41	1480	05C8	D7	C8	P	H	50	48	P	H
19	42	1481	05C9	D7	C9	P	I	50	49	P	I
19	43	1482	05CA	D7	4A	P	¢	50	5B	P	[
19	44	1483	05CB	D7	4B	P	.	50	2E	P	.
19	45	1484	05CC	D7	4C	P	<	50	3C	P	<
19	46	1485	05CD	D7	4D	P	(	50	28	P	(
19	47	1486	05CE	D7	4E	P	+	50	2B	P	+
19	48	1487	05CF	D7	4F	P	|	50	21	P	!
19	49	1488	05D0	D7	50	P	&	50	26	P	&
19	50	1489	05D1	D7	D1	P	J	50	4A	P	J
19	51	1490	05D2	D7	D2	P	K	50	4B	P	K
19	52	1491	05D3	D7	D3	P	L	50	4C	P	L
19	53	1492	05D4	D7	D4	P	M	50	4D	P	M
19	54	1493	05D5	D7	D5	P	N	50	4E	P	N
19	55	1494	05D6	D7	D6	P	O	50	4F	P	O
19	56	1495	05D7	D7	D7	P	P	50	50	P	P
19	57	1496	05D8	D7	D8	P	Q	50	51	P	Q
19	58	1497	05D9	D7	D9	P	R	50	52	P	R
19	59	1498	05DA	D7	5A	P	!	50	5D	P	]
19	60	1499	05DB	D7	5B	P	$	50	24	P	$
19	61	1500	05DC	D7	5C	P	*	50	2A	P	*
19	62	1501	05DD	D7	5D	P	)	50	29	P	)
19	63	1502	05DE	D7	5E	P	;	50	3B	P	;
19	64	1503	05DF	D7	5F	P	¬	50	5E	P	↑
19	65	1504	05E0	D7	60	P	-	50	2D	P	-
19	66	1505	05E1	D7	61	P	/	50	2F	P	/
19	67	1506	05E2	D7	E2	P	S	50	53	P	S
19	68	1507	05E3	D7	E3	P	T	50	54	P	T
19	69	1508	05E4	D7	E4	P	U	50	55	P	U
19	70	1509	05E5	D7	E5	P	V	50	56	P	V
19	71	1510	05E6	D7	E6	P	W	50	57	P	W
19	72	1511	05E7	D7	E7	P	X	50	58	P	X
19	73	1512	05E8	D7	E8	P	Y	50	59	P	Y
19	74	1513	05E9	D7	E9	P	Z	50	5A	P	Z
19	75	1514	05EA	D7	6A	P	¦	50	7C	P	¦
19	76	1515	05EB	D7	6B	P	,	50	2C	P	,
19	77	1516	05EC	D7	6C	P	%	50	25	P	%
19	78	1517	05ED	D7	6D	P	_	50	5F	P	_
19	79	1518	05EE	D7	6E	P	>	50	3E	P	>
19	80	1519	05EF	D7	6F	P	?	50	3F	P	?
20	01	1520	05F0	D7	F0	P	0	50	30	P	0
20	02	1521	05F1	D7	F1	P	1	50	31	P	1
20	03	1522	05F2	D7	F2	P	2	50	32	P	2
20	04	1523	05F3	D7	F3	P	3	50	33	P	3
20	05	1524	05F4	D7	F4	P	4	50	34	P	4
20	06	1525	05F5	D7	F5	P	5	50	35	P	5
20	07	1526	05F6	D7	F6	P	6	50	36	P	6
20	08	1527	05F7	D7	F7	P	7	50	37	P	7
20	09	1528	05F8	D7	F8	P	8	50	38	P	8
20	10	1529	05F9	D7	F9	P	9	50	39	P	9
20	11	1530	05FA	D7	7A	P	:	50	3A	P	:
20	12	1531	05FB	D7	7B	P	#	50	23	P	#
20	13	1532	05FC	D7	7C	P	@	50	40	P	@
20	14	1533	05FD	D7	7D	P	'	50	27	P	'

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
20	15	1534	05FE	D7	7E	P	=	50	3D	P	=
20	16	1535	05FF	D7	7F	P	"	50	22	P	"
20	17	1536	0600	D8	40	Q	□	51	20	Q	□
20	18	1537	0601	D8	C1	Q	A	51	41	Q	A
20	19	1538	0602	D8	C2	Q	B	51	42	Q	B
20	20	1539	0603	D8	C3	Q	C	51	43	Q	C
20	21	1540	0604	D8	C4	Q	D	51	44	Q	D
20	22	1541	0605	D8	C5	Q	E	51	45	Q	E
20	23	1542	0606	D8	C6	Q	F	51	46	Q	F
20	24	1543	0607	D8	C7	Q	G	51	47	Q	G
20	25	1544	0608	D8	C8	Q	H	51	48	Q	H
20	26	1545	0609	D8	C9	Q	I	51	49	Q	I
20	27	1546	060A	D8	4A	Q	¢	51	5B	Q	[
20	28	1547	060B	D8	4B	Q	.	51	2E	Q	.
20	29	1548	060C	D8	4C	Q	<	51	3C	Q	<
20	30	1549	060D	D8	4D	Q	(	51	28	Q	(
20	31	1550	060E	D8	4E	Q	+	51	2B	Q	+
20	32	1551	060F	D8	4F	Q	|	51	21	Q	!
20	33	1552	0610	D8	50	Q	&	51	26	Q	&
20	34	1553	0611	D8	D1	Q	J	51	4A	Q	J
20	35	1554	0612	D8	D2	Q	K	51	4B	Q	K
20	36	1555	0613	D8	D3	Q	L	51	4C	Q	L
20	37	1556	0614	D8	D4	Q	M	51	4D	Q	M
20	38	1557	0615	D8	D5	Q	N	51	4E	Q	N
20	39	1558	0616	D8	D6	Q	O	51	4F	Q	O
20	40	1559	0617	D8	D7	Q	P	51	50	Q	P
20	41	1560	0618	D8	D8	Q	Q	51	51	Q	Q
20	42	1561	0619	D8	D9	Q	R	51	52	Q	R
20	43	1562	061A	D8	5A	Q	!	51	5D	Q	]
20	44	1563	061B	D8	5B	Q	$	51	24	Q	$
20	45	1564	061C	D8	5C	Q	*	51	2A	Q	*
20	46	1565	061D	D8	5D	Q	)	51	29	Q	)
20	47	1566	061E	D8	5E	Q	;	51	3B	Q	;
20	48	1567	061F	D8	5F	Q	¬	51	5E	Q	↑
20	49	1568	0620	D8	60	Q	-	51	2D	Q	-
20	50	1569	0621	D8	61	Q	/	51	2F	Q	/
20	51	1570	0622	D8	E2	Q	S	51	53	Q	S
20	52	1571	0623	D8	E3	Q	T	51	54	Q	T
20	53	1572	0624	D8	E4	Q	U	51	55	Q	U
20	54	1573	0625	D8	E5	Q	V	51	56	Q	V
20	55	1574	0626	D8	E6	Q	W	51	57	Q	W
20	56	1575	0627	D8	E7	Q	X	51	58	Q	X
20	57	1576	0628	D8	E8	Q	Y	51	59	Q	Y
20	58	1577	0629	D8	E9	Q	Z	51	5A	Q	Z
20	59	1578	062A	D8	6A	Q	|	51	7C	Q	|
20	60	1579	062B	D8	6B	Q	,	51	2C	Q	,
20	61	1580	062C	D8	6C	Q	%	51	25	Q	%
20	62	1581	062D	D8	6D	Q	_	51	5F	Q	←
20	63	1582	062E	D8	6E	Q	>	51	3E	Q	>
20	64	1583	062F	D8	6F	Q	?	51	3F	Q	?
20	65	1584	0630	D8	F0	Q	0	51	30	Q	0
20	66	1585	0631	D8	F1	Q	1	51	31	Q	1
20	67	1586	0632	D8	F2	Q	2	51	32	Q	2
20	68	1587	0633	D8	F3	Q	3	51	33	Q	3
20	69	1588	0634	D8	F4	Q	4	51	34	Q	4
20	70	1589	0635	D8	F5	Q	5	51	35	Q	5
20	71	1590	0636	D8	F6	Q	6	51	36	Q	6
20	72	1591	0637	D8	F7	Q	7	51	37	Q	7
20	73	1592	0638	D8	F8	Q	8	51	38	Q	8
20	74	1593	0639	D8	F9	Q	9	51	39	Q	9
20	75	1594	063A	D8	7A	Q	:	51	3A	Q	:
20	76	1595	063B	D8	7B	Q	#	51	23	Q	#

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
20	77	1596	063C	D8	7C	Q	@	51	40	Q	@
20	78	1597	063D	D8	7D	Q	'	51	27	Q	'
20	79	1598	063E	D8	7E	Q	=	51	3D	Q	=
20	80	1599	063F	D8	7F	Q	"	51	22	Q	"
21	01	1600	0640	D9	40	R	□	52	20	R	□
21	02	1601	0641	D9	C1	R	A	52	41	R	A
21	03	1602	0642	D9	C2	R	B	52	42	R	B
21	04	1603	0643	D9	C3	R	C	52	43	R	C
21	05	1604	0644	D9	C4	R	D	52	44	R	D
21	06	1605	0645	D9	C5	R	E	52	45	R	E
21	07	1606	0646	D9	C6	R	F	52	46	R	F
21	08	1607	0647	D9	C7	R	G	52	47	R	G
21	09	1608	0648	D9	C8	R	H	52	48	R	H
21	10	1609	0649	D9	C9	R	I	52	49	R	I
21	11	1610	064A	D9	4A	R	¢	52	5B	R	[
21	12	1611	064B	D9	4B	R	.	52	2E	R	.
21	13	1612	064C	D9	4C	R	<	52	3C	R	<
21	14	1613	064D	D9	4D	R	(	52	28	R	(
21	15	1614	064E	D9	4E	R	+	52	2B	R	+
21	16	1615	064F	D9	4F	R	|	52	21	R	!
21	17	1616	0650	D9	50	R	&	52	26	R	&
21	18	1617	0651	D9	D1	R	J	52	4A	R	J
21	19	1618	0652	D9	D2	R	K	52	4B	R	K
21	20	1619	0653	D9	D3	R	L	52	4C	R	L
21	21	1620	0654	D9	D4	R	M	52	4D	R	M
21	22	1621	0655	D9	D5	R	N	52	4E	R	N
21	23	1622	0656	D9	D6	R	O	52	4F	R	O
21	24	1623	0657	D9	D7	R	P	52	50	R	P
21	25	1624	0658	D9	D8	R	Q	52	51	R	Q
21	26	1625	0659	D9	D9	R	R	52	52	R	R
21	27	1626	065A	D9	5A	R	!	52	5D	R	]
21	28	1627	065B	D9	5B	R	$	52	24	R	$
21	29	1628	065C	D9	5C	R	*	52	2A	R	*
21	30	1629	065D	D9	5D	R	)	52	29	R	)
21	31	1630	065E	D9	5E	R	;	52	3B	R	;
21	32	1631	065F	D9	5F	R	¬	52	5E	R	↑
21	33	1632	0660	D9	60	R	-	52	2D	R	-
21	34	1633	0661	D9	61	R	/	52	2F	R	/
21	35	1634	0662	D9	E2	R	S	52	53	R	S
21	36	1635	0663	D9	E3	R	T	52	54	R	T
21	37	1636	0664	D9	E4	R	U	52	55	R	U
21	38	1637	0665	D9	E5	R	V	52	56	R	V
21	39	1638	0666	D9	E6	R	W	52	57	R	W
21	40	1639	0667	D9	E7	R	X	52	58	R	X
21	41	1640	0668	D9	E8	R	Y	52	59	R	Y
21	42	1641	0669	D9	E9	R	Z	52	5A	R	Z
21	43	1642	066A	D9	6A	R	!	52	7C	R	!
21	44	1643	066B	D9	6B	R	,	52	2C	R	,
21	45	1644	066C	D9	6C	R	%	52	25	R	%
21	46	1645	066D	D9	6D	R	_	52	5F	R	←
21	47	1646	066E	D9	6E	R	>	52	3E	R	>
21	48	1647	066F	D9	6F	R	?	52	3F	R	?
21	49	1648	0670	D9	F0	R	0	52	30	R	0
21	50	1649	0671	D9	F1	R	1	52	31	R	1
21	51	1650	0672	D9	F2	R	2	52	32	R	2
21	52	1651	0673	D9	F3	R	3	52	33	R	3
21	53	1652	0674	D9	F4	R	4	52	34	R	4
21	54	1653	0675	D9	F5	R	5	52	35	R	5
21	55	1654	0676	D9	F6	R	6	52	36	R	6
21	56	1655	0677	D9	F7	R	7	52	37	R	7
21	57	1656	0678	D9	F8	R	8	52	38	R	8
21	58	1657	0679	D9	F9	R	9	52	39	R	9

EBCDIC and ASCII Address Representations (continued)

RCX70	Dec	Hex	EBCDIC		Char		ASCII		Char	
21 59	1658	067A	D9	7A	R	:	52	3A	R	:
21 60	1659	067B	D9	7B	R	#	52	23	R	#
21 61	1660	067C	D9	7C	R	@	52	40	R	@
21 62	1661	067D	D9	7D	R	'	52	27	R	'
21 63	1662	067E	D9	7E	R	=	52	3D	R	=
21 64	1663	067F	D9	7F	R	"	52	22	R	"
21 65	1664	0680	5A	40	!	□	5D	20	]	□
21 66	1665	0681	5A	C1	!	A	5D	41	]	A
21 67	1666	0682	5A	C2	!	B	5D	42	]	B
21 68	1667	0683	5A	C3	!	C	5D	43	]	C
21 69	1668	0684	5A	C4	!	D	5D	44	]	D
21 70	1669	0685	5A	C5	!	E	5D	45	]	E
21 71	1670	0686	5A	C6	!	F	5D	46	]	F
21 72	1671	0687	5A	C7	!	G	5D	47	]	G
21 73	1672	0688	5A	C8	!	H	5D	48	]	H
21 74	1673	0689	5A	C9	!	I	5D	49	]	I
21 75	1674	068A	5A	4A	!	¢	5D	5B	]	[
21 76	1675	068B	5A	4B	!	.	5D	2E	]	.
21 77	1676	068C	5A	4C	!	<	5D	3C	]	<
21 78	1677	068D	5A	4D	!	(	5D	28	]	(
21 79	1678	068E	5A	4E	!	+	5D	2B	]	+
21 80	1679	068F	5A	4F	!	|	5D	21	]	!
22 01	1680	0690	5A	50	!	&	5D	26	]	&
22 02	1681	0691	5A	D1	!	J	5D	4A	]	J
22 03	1682	0692	5A	D2	!	K	5D	4B	]	K
22 04	1683	0693	5A	D3	!	L	5D	4C	]	L
22 05	1684	0694	5A	D4	!	M	5D	4D	]	M
22 06	1685	0695	5A	D5	!	N	5D	4E	]	N
22 07	1686	0696	5A	D6	!	O	5D	4F	]	O
22 08	1687	0697	5A	D7	!	P	5D	50	]	P
22 09	1688	0698	5A	D8	!	Q	5D	51	]	Q
22 10	1689	0699	5A	D9	!	R	5D	52	]	R
22 11	1690	069A	5A	5A	!	!	5D	5D	]	]
22 12	1691	069B	5A	5B	!	$	5D	24	]	$
22 13	1692	069C	5A	5C	!	*	5D	2A	]	*
22 14	1693	069D	5A	5D	!	)	5D	29	]	)
22 15	1694	069E	5A	5E	!	;	5D	3B	]	;
22 16	1695	069F	5A	5F	!	¬	5D	5E	]	↑
22 17	1696	06A0	5A	60	!	-	5D	2D	]	-
22 18	1697	06A1	5A	61	!	/	5D	2F	]	/
22 19	1698	06A2	5A	E2	!	S	5D	53	]	S
22 20	1699	06A3	5A	E3	!	T	5D	54	]	T
22 21	1700	06A4	5A	E4	!	U	5D	55	]	U
22 22	1701	06A5	5A	E5	!	V	5D	56	]	V
22 23	1702	06A6	5A	E6	!	W	5D	57	]	W
22 24	1703	06A7	5A	E7	!	X	5D	58	]	X
22 25	1704	06A8	5A	E8	!	Y	5D	59	]	Y
22 26	1705	06A9	5A	E9	!	Z	5D	5A	]	Z
22 27	1706	06AA	5A	6A	!	|	5D	7C	]	|
22 28	1707	06AB	5A	6B	!	,	5D	2C	]	,
22 29	1708	06AC	5A	6C	!	%	5D	25	]	%
22 30	1709	06AD	5A	6D	!	_	5D	5F	]	←
22 31	1710	06AE	5A	6E	!	>	5D	3E	]	>
22 32	1711	06AF	5A	6F	!	?	5D	3F	]	?
22 33	1712	06B0	5A	F0	!	0	5D	30	]	0
22 34	1713	06B1	5A	F1	!	1	5D	31	]	1
22 35	1714	06B2	5A	F2	!	2	5D	32	]	2
22 36	1715	06B3	5A	F3	!	3	5D	33	]	3
22 37	1716	06B4	5A	F4	!	4	5D	34	]	4
22 38	1717	06B5	5A	F5	!	5	5D	35	]	5
22 39	1718	06B6	5A	F6	!	6	5D	36	]	6
22 40	1719	06B7	5A	F7	!	7	5D	37	]	7

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
22	41	1720	06B8	5A	F8	!	8	5D	38	]	8
22	42	1721	06B9	5A	F9	!	9	5D	39	]	9
22	43	1722	06BA	5A	7A	!	:	5D	3A	]	:
22	44	1723	06BB	5A	7B	!	#	5D	23	]	#
22	45	1724	06BC	5A	7C	!	@	5D	40	]	@
22	46	1725	06BD	5A	7D	!	'	5D	27	]	'
22	47	1726	06BE	5A	7E	!	=	5D	3D	]	=
22	48	1727	06BF	5A	7F	!	"	5D	22	]	"
22	49	1728	06C0	5B	40	$	□	24	20	$	□
22	50	1729	06C1	5B	C1	$	A	24	41	$	A
22	51	1730	06C2	5B	C2	$	B	24	42	$	B
22	52	1731	06C3	5B	C3	$	C	24	43	$	C
22	53	1732	06C4	5B	C4	$	D	24	44	$	D
22	54	1733	06C5	5B	C5	$	E	24	45	$	E
22	55	1734	06C6	5B	C6	$	F	24	46	$	F
22	56	1735	06C7	5B	C7	$	G	24	47	$	G
22	57	1736	06C8	5B	C8	$	H	24	48	$	H
22	58	1737	06C9	5B	C9	$	I	24	49	$	I
22	59	1738	06CA	5B	4A	$	¢	24	5B	$	[
22	60	1739	06CB	5B	4B	$	.	24	2E	$	.
22	61	1740	06CC	5B	4C	$	<	24	3C	$	<
22	62	1741	06CD	5B	4D	$	(	24	28	$	(
22	63	1742	06CE	5B	4E	$	+	24	2B	$	+
22	64	1743	06CF	5B	4F	$	|	24	21	$	!
22	65	1744	06D0	5B	50	$	&	24	26	$	&
22	66	1745	06D1	5B	D1	$	J	24	4A	$	J
22	67	1746	06D2	5B	D2	$	K	24	4B	$	K
22	68	1747	06D3	5B	D3	$	L	24	4C	$	L
22	69	1748	06D4	5B	D4	$	M	24	4D	$	M
22	70	1749	06D5	5B	D5	$	N	24	4E	$	N
22	71	1750	06D6	5B	D6	$	O	24	4F	$	O
22	72	1751	06D7	5B	D7	$	P	24	50	$	P
22	73	1752	06D8	5B	D8	$	Q	24	51	$	Q
22	74	1753	06D9	5B	D9	$	R	24	52	$	R
22	75	1754	06DA	5B	5A	$	!	24	5D	$	]
22	76	1755	06DB	5B	5B	$	$	24	24	$	$
22	77	1756	06DC	5B	5C	$	*	24	2A	$	*
22	78	1757	06DD	5B	5D	$	)	24	29	$	)
22	79	1758	06DE	5B	5E	$	;	24	3B	$	;
22	80	1759	06DF	5B	5F	$	¬	24	5E	$	↑
23	01	1760	06E0	5B	60	$	-	24	2D	$	-
23	02	1761	06E1	5B	61	$	/	24	2F	$	/
23	03	1762	06E2	5B	E2	$	S	24	53	$	S
23	04	1763	06E3	5B	E3	$	T	24	54	$	T
23	05	1764	06E4	5B	E4	$	U	24	55	$	U
23	06	1765	06E5	5B	F5	$	V	24	56	$	V
23	07	1766	06E6	5B	E6	$	W	24	57	$	W
23	08	1767	06E7	5B	E7	$	X	24	58	$	X
23	09	1768	06E8	5B	E8	$	Y	24	59	$	Y
23	10	1769	06E9	5B	E9	$	Z	24	5A	$	Z
23	11	1770	06EA	5B	6A	$	|	24	7C	$	|
23	12	1771	06EB	5B	6B	$	,	24	2C	$	,
23	13	1772	06EC	5B	6C	$	%	24	25	$	%
23	14	1773	06ED	5B	6D	$	_	24	5F	$	_
23	15	1774	06EE	5B	6E	$	>	24	3E	$	>
23	16	1775	06EF	5B	6F	$	?	24	3F	$	?
23	17	1776	06F0	5B	F0	$	0	24	30	$	0
23	18	1777	06F1	5B	F1	$	1	24	31	$	1
23	19	1778	06F2	5B	F2	$	2	24	32	$	2
23	20	1779	06F3	5B	F3	$	3	24	33	$	3
23	21	1780	06F4	5B	F4	$	4	24	34	$	4

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
23	22	1781	06F5	5B	F5	$	5	24	35	$	5
23	23	1782	06F6	5B	F6	$	6	24	36	$	6
23	24	1783	06F7	5B	F7	$	7	24	37	$	7
23	25	1784	06F8	5B	F8	$	8	24	38	$	8
23	26	1785	06F9	5B	F9	$	9	24	39	$	9
23	27	1786	06FA	5B	7A	$	:	24	3A	$	:
23	28	1787	06FB	5B	7B	$	#	24	23	$	#
23	29	1788	06FC	5B	7C	$	@	24	40	$	@
23	30	1789	06FD	5B	7D	$	'	24	27	$	'
23	31	1790	06FE	5B	7E	$	=	24	3D	$	=
23	32	1791	06FF	5B	7F	$	"	24	22	$	"
23	33	1792	0700	5C	40	*	□	2A	20	*	□
23	34	1793	0701	5C	C1	*	A	2A	41	*	A
23	35	1794	0702	5C	C2	*	B	2A	42	*	B
23	36	1795	0703	5C	C3	*	C	2A	43	*	C
23	37	1796	0704	5C	C4	*	D	2A	44	*	D
23	38	1797	0705	5C	C5	*	E	2A	45	*	E
23	39	1798	0706	5C	C6	*	F	2A	46	*	F
23	40	1799	0707	5C	C7	*	G	2A	47	*	G
23	41	1800	0708	5C	C8	*	H	2A	48	*	H
23	42	1801	0709	5C	C9	*	I	2A	49	*	I
23	43	1802	070A	5C	4A	*	¢	2A	5B	*	[
23	44	1803	070B	5C	4B	*	.	2A	2E	*	.
23	45	1804	070C	5C	4C	*	<	2A	3C	*	<
23	46	1805	070D	5C	4D	*	(	2A	28	*	(
23	47	1806	070E	5C	4E	*	+	2A	2B	*	+
23	48	1807	070F	5C	4F	*	|	2A	21	*	!
23	49	1808	0710	5C	50	*	&	2A	26	*	&
23	50	1809	0711	5C	D1	*	J	2A	4A	*	J
23	51	1810	0712	5C	D2	*	K	2A	4B	*	K
23	52	1811	0713	5C	D3	*	L	2A	4C	*	L
23	53	1812	0714	5C	D4	*	M	2A	4D	*	M
23	54	1813	0715	5C	D5	*	N	2A	4E	*	N
23	55	1814	0716	5C	D6	*	O	2A	4F	*	O
23	56	1815	0717	5C	D7	*	P	2A	50	*	P
23	57	1816	0718	5C	D8	*	Q	2A	51	*	Q
23	58	1817	0719	5C	D9	*	R	2A	52	*	R
23	59	1818	071A	5C	5A	*	!	2A	5D	*	]
23	60	1819	071B	5C	5B	*	$	2A	24	*	$
23	61	1820	071C	5C	5C	*	*	2A	2A	*	*
23	62	1821	071D	5C	5D	*	)	2A	29	*	)
23	63	1822	071E	5C	5E	*	;	2A	3B	*	;
23	64	1823	071F	5C	5F	*	¬	2A	5E	*	↑
23	65	1824	0720	5C	60	*	-	2A	2D	*	-
23	66	1825	0721	5C	61	*	/	2A	2F	*	/
23	67	1826	0722	5C	E2	*	S	2A	53	*	S
23	68	1827	0723	5C	E3	*	T	2A	54	*	T
23	69	1828	0724	5C	E4	*	U	2A	55	*	U
23	70	1829	0725	5C	E5	*	V	2A	56	*	V
23	71	1830	0726	5C	E6	*	W	2A	57	*	W
23	72	1831	0727	5C	E7	*	X	2A	58	*	X
23	73	1832	0728	5C	E8	*	Y	2A	59	*	Y
23	74	1833	0729	5C	E9	*	Z	2A	5A	*	Z
23	75	1834	072A	5C	6A	*	|	2A	7C	*	|
23	76	1835	072B	5C	6B	*	,	2A	2C	*	,
23	77	1836	072C	5C	6C	*	%	2A	25	*	%
23	78	1837	072D	5C	6D	*	_	2A	5F	*	_
23	79	1838	072E	5C	6E	*	>	2A	3E	*	>
23	80	1839	072F	5C	6F	*	?	2A	3F	*	?
24	01	1840	0730	5C	F0	*	0	2A	30	*	0
24	02	1841	0731	5C	F1	*	1	2A	31	*	1
24	03	1842	0732	5C	F2	*	2	2A	32	*	2

RCX70		Dec	Hex	EBCDIC		Char		ASCII		Char	
24	04	1843	0733	5C	F3	*	3	2A	33	*	3
24	05	1844	0734	5C	F4	*	4	2A	34	*	4
24	06	1845	0735	5C	F5	*	5	2A	35	*	5
24	07	1846	0736	5C	F6	*	6	2A	36	*	6
24	08	1847	0737	5C	F7	*	7	2A	37	*	7
24	09	1848	0738	5C	F8	*	8	2A	38	*	8
24	10	1849	0739	5C	F9	*	9	2A	39	*	9
24	11	1850	073A	5C	7A	*	:	2A	3A	*	:
24	12	1851	073B	5C	7B	*	#	2A	23	*	#
24	13	1852	073C	5C	7C	*	@	2A	40	*	@
24	14	1853	073D	5C	7D	*	'	2A	27	*	'
24	15	1854	073E	5C	7E	*	=	2A	3D	*	=
24	16	1855	073F	5C	7F	*	"	2A	22	*	"
24	17	1856	0740	5D	40	)	□	29	20	)	□
24	18	1857	0741	5D	C1	)	A	29	41	)	A
24	19	1858	0742	5D	C2	)	B	29	42	)	B
24	20	1859	0743	5D	C3	)	C	29	43	)	C
24	21	1860	0744	5D	C4	)	D	29	44	)	D
24	22	1861	0745	5D	C5	)	E	29	45	)	E
24	23	1862	0746	5D	C6	)	F	29	46	)	F
24	24	1863	0747	5D	C7	)	G	29	47	)	G
24	25	1864	0748	5D	C8	)	H	29	48	)	H
24	26	1865	0749	5D	C9	)	I	29	49	)	I
24	27	1866	074A	5D	4A	)	¢	29	5B	)	[
24	28	1867	074B	5D	4B	)	.	29	2E	)	.
24	29	1868	074C	5D	4C	)	<	29	3C	)	<
24	30	1869	074D	5D	4D	)	(	29	28	)	(
24	31	1870	074E	5D	4E	)	+	29	2B	)	+
24	32	1871	074F	5D	4F	)	|	29	21	)	!
24	33	1872	0750	5D	50	)	&	29	26	)	&
24	34	1873	0751	5D	D1	)	J	29	4A	)	J
24	35	1874	0752	5D	D2	)	K	29	4B	)	K
24	36	1875	0753	5D	D3	)	L	29	4C	)	L
24	37	1876	0754	5D	D4	)	M	29	4D	)	M
24	38	1877	0755	5D	D5	)	N	29	4E	)	N
24	39	1878	0756	5D	D6	)	O	29	4F	)	O
24	40	1879	0757	5D	D7	)	P	29	50	)	P
24	41	1880	0758	5D	D8	)	Q	29	51	)	Q
24	42	1881	0759	5D	D9	)	R	29	52	)	R
24	43	1882	075A	5D	5A	)	!	29	5D	)	]
24	44	1883	075B	5D	5B	)	$	29	24	)	$
24	45	1884	075C	5D	5C	)	*	29	2A	)	*
24	46	1885	075D	5D	5D	)	)	29	29	)	)
24	47	1886	075E	5D	5E	)	;	29	3B	)	;
24	48	1887	075F	5D	5F	)	¬	29	5F	)	↑
24	49	1888	0760	5D	60	)	-	29	2D	)	-
24	50	1889	0761	5D	61	)	/	29	2F	)	/
24	51	1890	0762	5D	E2	)	S	29	53	)	S
24	52	1891	0763	5D	E3	)	T	29	54	)	T
24	53	1892	0764	5D	E4	)	U	29	55	)	U
24	54	1893	0765	5D	E5	)	V	29	56	)	V
24	55	1894	0766	5D	E6	)	W	29	57	)	W
24	56	1895	0767	5D	E7	)	X	29	58	)	X
24	57	1896	0768	5D	E8	)	Y	29	59	)	Y
24	58	1897	0769	5D	E9	)	Z	29	5A	)	Z
24	59	1898	076A	5D	6A	)	¦	29	7C	)	¦
24	60	1899	076B	5D	6B	)	,	29	2C	)	,
24	61	1900	076C	5D	6C	)	%	29	25	)	%
24	62	1901	076D	5D	6D	)	_	29	5F	)	_
24	63	1902	076E	5D	6E	)	>	29	3E	)	>
24	64	1903	076F	5D	6F	)	?	29	3F	)	?
24	65	1904	0770	5D	F0	)	0	29	30	)	0

EBCDIC and ASCII Address Representations (continued)

RCX70		Dec	Hex	EBCDIC			Char	ASCII			Char
24	66	1905	0771	5D	F1	)	1	29	31	)	1
24	67	1906	0772	5D	F2	)	2	29	32	)	2
24	68	1907	0773	5D	F3	)	3	29	33	)	3
24	69	1908	0774	5D	F4	)	4	29	34	)	4
24	70	1909	0775	5D	F5	)	5	29	35	)	5
24	71	1910	0776	5D	F6	)	6	29	36	)	6
24	72	1911	0777	5D	F7	)	7	29	37	)	7
24	73	1912	0778	5D	F8	)	8	29	38	)	8
24	74	1913	0779	5D	F9	)	9	29	39	)	9
24	75	1914	077A	5D	7A	)	:	29	3A	)	:
24	76	1915	077B	5D	7B	)	#	29	23	)	#
24	77	1916	077C	5D	7C	)	@	29	40	)	@
24	78	1917	077D	5D	7D	)	'	29	27	)	'
24	79	1918	077E	5D	7E	)	=	29	3D	)	=
24	80	1919	077F	5D	7F	)	"	29	22	)	"

End of Appendix

Index

Within this index, "f" or "ff" after a page number means "and the following page" (or "pages"). In addition, primary page references for each topic are listed first. Command, calls, and acronyms are in uppercase letters (e.g., CREATE); all others are lowercase.

093-000172-00