

**ENCORE 200**  
Portable Network Analyzer



# ENCORE 200

## Portable Network Analyzer

### TECHNICAL MANUAL DESCRIPTION AND OPERATION



Digitech Industries, Inc.  
66 Grove Street, P.O. Box 547  
Ridgefield, CT 06877 U.S.A.  
(203) 438-3731 TELEX: 969623

Printed in U.S.A.



**PROPRIETARY INFORMATION NOTICE**

This manual may not be reproduced or used for manufacturing purposes except by written permission from DIGITECH.

**WARNING**

Be careful when working near the 115 VAC line connections. Serious injury or death may result from contact with these terminals.

Copyright 1982

DIGITECH Industries, Inc.  
66 Grove Street, P.O. Box 547  
Ridgefield, Connecticut 06877-0547  
(203) 438-3731 TELEX 969-623

Printed in U.S.A.

TABLE OF CONTENTS

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
1	INTRODUCTION . . . . .	1-1
	1. GENERAL . . . . .	1-1
	2. USING THE MANUAL . . . . .	1-3
2	UNPACKING AND INSPECTION . . . . .	2-1
	1. GENERAL . . . . .	2-1
	2. ENVIRONMENTAL CONSIDERATIONS. . . . .	2-1
	3. UNPACKING AND HANDLING . . . . .	2-1
	4. INSPECTION . . . . .	2-4
	5. SAFETY PRECAUTIONS. . . . .	2-4
	6. CLEANING. . . . .	2-4
	7. STORAGE . . . . .	2-4
	8. INSTALLATION. . . . .	2-4
	A. Placement . . . . .	2-4
	B. Power Requirements. . . . .	2-4
3	PRODUCT DESCRIPTION . . . . .	3-1
	1. GENERAL . . . . .	3-1
	2. THREE LEVELS OF OPERATION . . . . .	3-2
	3. DESCRIPTION . . . . .	3-2
	A. Primary Display. . . . .	3-2
	B. Real Time Display. . . . .	3-3
	C. Disc Drive . . . . .	3-3
	D. Memories. . . . .	3-3
	E. Keyboard. . . . .	3-3

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
3	F. Interface. . . . .	3-3
4.	DESIGN PHILOSOPHY. . . . .	3-5
5.	ARCHITECTURE . . . . .	3-5
	A. Control Signal Attributes. . . . .	3-5
	IO Control . . . . .	3-6
	DMA Control . . . . .	3-6
	INT (Interrupt) Control. . . . .	3-7
	B. Control Signal Assignment . . . . .	3-7
	Disc Controller . . . . .	3-7
	CRT Controller . . . . .	3-7
	Keyboard, External RS-232, Speaker, and Transmitter . . . . .	3-7
	Front End 2/3. . . . .	3-8
	Memory . . . . .	3-8
6.	STANDARD CONFIGURATION . . . . .	3-8
7.	OPTIONS. . . . .	3-10
8.	CONTROLS, DISPLAYS, AND CONNECTORS. . . . .	3-11
9.	CONTROL CLUSTER AND SPECIAL FUNCTION KEYS . . . . .	3-16
	A. Control Cluster Keys. . . . .	3-16
	B. Special Function Keys . . . . .	3-17
10.	T-CONNECTOR. . . . .	3-19
11.	AUDIO OUTPUT . . . . .	3-19
4	INITIAL SET-UP AND OPERATION . . . . .	4-1
	1. GENERAL . . . . .	4-1
	2. INITIAL SET-UP . . . . .	4-1

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
4	POWER-UP. . . . .	4-2
	3. TYPICAL OPERATION IN LEVEL-1 . . . . .	4-5
	4. TYPICAL OPERATION IN LEVEL-2 . . . . .	4-7
	5. TYPICAL OPERATION IN LEVEL-3 . . . . .	4-9
5	LEVEL-1 OPERATION . . . . .	5-1
	1. GENERAL . . . . .	5-1
	LEVEL-1 DIRECTORY. . . . .	5-1
	2. ASYNCHRONOUS LIBRARY . . . . .	5-3
	Monitor/Trap . . . . .	5-4
	Off-Line Data Display . . . . .	5-6
	Echo as DCE . . . . .	5-6
	Echo as DTE . . . . .	5-6
	TTY Conversational as DTE. . . . .	5-6
	TTY Conversational as DCE. . . . .	5-6
	3. SYNCHRONOUS LIBRARY. . . . .	5-7
	Monitor/Trap . . . . .	5-9
	Off-Line Data Display . . . . .	5-9
	Transparent Monitor. . . . .	5-9
	Poll/Select Remote . . . . .	5-10
	Network Analysis . . . . .	5-10
	Overall Line Performance . . . . .	5-11
	Station Performance. . . . .	5-12
	4. BIT ORIENTED LIBRARY . . . . .	5-13
	Monitor/Trap . . . . .	5-15

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
5	Off-Line Data Display . . . . .	5-15
	5. OFF-LINE DATA DISPLAY LIBRARY . . . . .	5-15
	Display Captire Buffer HDUX. . . . .	5-17
	Display Capture Buffer FDUX. . . . .	5-18
	Display Data Disc . . . . .	5-19
	Display Capture Buffer and EIA Status . . . . .	5-20
	Display Disc and EIA Status. . . . .	5-21
	Dump Capture Buffer to XIO Port . . . . .	5-22
	6. MEASUREMENT LIBRARY. . . . .	5-22
	Bert. . . . .	5-24
	Cert. . . . .	5-25
	Measure CPU Response . . . . .	5-26
	Count ACK/NAK's - Async . . . . .	5-27
	Count Polls Per Minute. . . . .	5-28
	Monitor RTS/CTS Delay . . . . .	5-29
	Measure RTS/CTS Delay . . . . .	5-30
	7. MASTER/SLAVE OPERATION . . . . .	5-30
	8. USER DEFINED LIBRARY . . . . .	5-32
	9. EIA TIMING MONITOR. . . . .	5-33
	10. MASTER DIRECTORY. . . . .	5-33
6	LEVEL-2 OPERATION AND PROGRAMMING. . . . .	6-1
	1. GENERAL . . . . .	6-1
	2. OPERATION . . . . .	6-2
	3. EXECUTE PROGRAM . . . . . MODE "E" . . . . .	6-3

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
6	Typical Execute Procedure . . . . .	6-3
	4. EDIT/ENTER PROGRAM . . . MODE "P" . . . . .	6-3
	Program Mode Commands . . . . .	6-5
	Typical Program Entry Procedure . . . . .	6-6
	Programming Hints . . . . .	6-6
	Instruction Set . . . . .	6-8
	5. SET IO PARAMETERS . . . MODE "T". . . . .	6-24
	6. DISPLAY CAPTURE BUFFER . . . MODE "D". . . . .	6-24
	Display Mode Commands . . . . .	6-26
	Typical Display Mode Operation. . . . .	6-29
	7. PRINT LEVEL-2 MENU . . . . .	6-31
	Typical Menu Print Operation. . . . .	6-32
	8. XIO . . . . .	6-33
	9. LOAD NEW MENU FROM PACER. . . . .	6-33
	Typical PACER to ENCORE Menu Load . . . . .	6-34
	10. LOAD NEW MENU FROM DISKETTE . . . . .	6-35
	Typical Menu Load from Diskette . . . . .	6-35
	11. DUMP CURRENT MENU TO PACER. . . . .	6-36
	Typical ENCORE to PACER Menu Dump . . . . .	6-36
	12. DUMP CURRENT MENU TO DISKETTE . . . . .	6-38
	Typical Menu Dump to Diskette. . . . .	6-38
7	LEVEL-3 OPERATION AND PROGRAMMING. . . . .	7-1
	1. GENERAL . . . . .	7-1
	2. LEVEL-3 COMMANDS. . . . .	7-1



TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	ASGN . . . . .	7-2
	D* . . . . .	7-3
	DXFR . . . . .	7-3
	D FORMAT. . . . .	7-3
	EDFT. . . . .	7-4
	EDFT. . . . .	7-4
	EDFT <i>program name.</i> . . . . .	7-4
	EDFT <i>program name!</i> . . . . .	7-4
	# EDFT <i>program name!</i> . . . . .	7-4
	D EDFT <i>program name.</i> . . . . .	7-4
	D EDFT <i>program name!</i> . . . . .	7-4
	# D EDFT <i>program name!</i> . . . . .	7-4
	HELP . . . . .	7-8
	IO. . . . .	7-8
	Speed . . . . .	7-8
	Sync. . . . .	7-8
	NRZI . . . . .	7-8
	Mode . . . . .	7-8
	Stops . . . . .	7-9
	Clock . . . . .	7-9
	Parity . . . . .	7-9
	Lang. . . . .	7-10
	Typical IO Entry Procedure. . . . .	7-11
	KILL. . . . .	7-11
	KILL <i>program name.</i> . . . . .	7-11

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	D KILL <i>program name</i> . . . . .	7-11
	MENU . . . . .	7-12
	MENU . . . . .	7-12
	D MENU . . . . .	7-13
	RUN. . . . .	7-13
	RUN. . . . .	7-13
	RUN! . . . . .	7-13
	<i>Program name</i> . . . . .	7-14
	<i>Program name!</i> . . . . .	7-14
	# <i>program name!</i> . . . . .	7-14
	D <i>program name</i> . . . . .	7-14
	D <i>program name!</i> . . . . .	7-14
	# D <i>program name!</i> . . . . .	7-14
	SAVE . . . . .	7-15
	SAVE . . . . .	7-15
	SAVE <i>program name</i> . . . . .	7-15
	SAVE <i>program name!</i> . . . . .	7-15
	D SAVE . . . . .	7-15
	D SAVE <i>program name</i> . . . . .	7-16
	D SAVE <i>program name!</i> . . . . .	7-16
	TIME . . . . .	7-16
	STATUS/MEMORY ALLOCATION/ <span style="border: 1px solid black; padding: 2px;">VE STAT</span> . . . . .	7-17
	XIO . . . . .	7-18
3.	COMBASIC PROGRAMMING . . . . .	7-20
	PROGRAMMING RULES AND AIDS . . . . .	7-20

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	RULES. . . . .	7-20
	PROGRAMMING AIDS. . . . .	7-21
	OPERATORS. . . . .	7-21
	AND. . . . .	7-21
	OR . . . . .	7-22
	RTL. . . . .	7-23
	RTR. . . . .	7-23
	XOR. . . . .	7-24
	VARIABLES . . . . .	7-25
	A. Integer Variables . . . . .	7-25
	B. Floating Point Variables . . . . .	7-28
	C. String Variables. . . . .	7-29
	D. Byte Variables . . . . .	7-32
	E. Reserved Variables . . . . .	7-36
	F. Other Variables . . . . .	7-37
	REAL TIME CLOCK. . . . .	7-37
	TIMERS/TRIGGERS . . . . .	7-38
	IO PARAMETERS IN COMBASIC . . . . .	7-40
	ATTRIBUTE AND STATUS . . . . .	7-40
	A. Attribute Byte . . . . .	7-41
	B. Status Byte. . . . .	7-43
	CAPTURE DATA CONTROL . . . . .	7-43
	BLOCK CHECK CALCULATIONS. . . . .	7-46
	FRONT ENDS. . . . .	7-48
	EXECUTION TIMES . . . . .	7-50

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	Timing Chart . . . . .	7-50
4.	EXAMPLE PROGRAMS . . . . .	7-51
	SCREEN CONTROL . . . . .	7-51
	CALCULATE. . . . .	7-54
	DISPLAY DISC . . . . .	7-57
5.	COMBASIC INSTRUCTION SET . . . . .	7-61
	ABORT . . . . .	7-61
	ABORT ALL . . . . .	7-61
	ABORT FULL. . . . .	7-61
	ABORT KBD . . . . .	7-61
	ABORT PIN . . . . .	7-61
	ABORT P2/P3. . . . .	7-61
	ABORT RTC . . . . .	7-61
	ABORT TIMER . . . . .	7-61
	ABORT XIOIN . . . . .	7-61
	BOOT . . . . .	7-62
	CALL . . . . .	7-63
	CHAIN. . . . .	7-64
	CLEAR . . . . .	7-65
	CLEAR CBUF. . . . .	7-65
	CLEAR CHAIN . . . . .	7-65
	CLEAR MSTM . . . . .	7-65
	CLEAR RETURN . . . . .	7-65
	DATA/READ/RESTORE . . . . .	7-65
	DIM. . . . .	7-66

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	DISP. . . . .	7-66
	DISP B. . . . .	7-67
	DISP C. . . . .	7-67
	DISP D. . . . .	7-67
	DISP E. . . . .	7-68
	DISP F. . . . .	7-68
	DISP G. . . . .	7-68
	DISP H. . . . .	7-68
	DISP L. . . . .	7-69
	DISP P. . . . .	7-69
	DISP R. . . . .	7-69
	DISP SA\$-SZ\$. . . . .	7-69
	DISP SLA\$-SIZ\$. . . . .	7-70
	DISP dBG . . . . .	7-70
	DISP dBS. . . . .	7-70
	DISP dE . . . . .	7-70
	DISP dR . . . . .	7-71
	DISP T1 . . . . .	7-71
	DISP T2 . . . . .	7-71
	DISP T4 . . . . .	7-71
	DISP T8 . . . . .	7-71
	DLC. . . . .	7-72
	DLC A\$-Z\$. . . . .	7-72
	FESYN. . . . .	7-73
	FOR/NEXT. . . . .	7-73

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	GOSUB/RETURN . . . . .	7-73
	GOTO <i>line number</i> . . . . .	7-74
	IF. . . . .	7-74
	IF A-Z. . . . .	7-74
	IF FA-FZ. . . . .	7-75
	IF a-z . . . . .	7-75
	IF A\$-Z\$. . . . .	7-75
	IF "program name". . . . .	7-75
	INPUT. . . . .	7-76
	INPUT A-Z. . . . .	7-76
	INPUT FA-FZ. . . . .	7-76
	INPUT a-z . . . . .	7-76
	INPUT A\$-Z\$. . . . .	7-77
	LET. . . . .	7-77
	LET A-Z. . . . .	7-77
	LET A\$-Z\$. . . . .	7-78
	LET A\$-Z\$=CBUFB . . . . .	7-78
	LET A\$-Z\$=CBUFE . . . . .	7-79
	LET A\$-Z\$=RTC . . . . .	7-79
	LET FA-FZ. . . . .	7-79
	LET FA-FZ=MSTM. . . . .	7-80
	LET a-z . . . . .	7-80
	LET a-z=a-z LPAR . . . . .	7-80
	LET a-z=DSTAT. . . . .	7-81
	LET a-z=a-z HEX . . . . .	7-81

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	LET a-z=XIOP or LET XIOP=a-z . . . . .	7-82
	LET a-z=0B " <i>binary number</i> " . . . . .	7-83
	LET a-z=0H " <i>HEX pair</i> " . . . . .	7-83
	LET a-z=0O " <i>octal number</i> ". . . . .	7-83
	LET BCS. . . . .	7-83
	LET CLOCK . . . . .	7-84
	LET LANG . . . . .	7-84
	LET MODE. . . . .	7-85
	LET NRZI . . . . .	7-85
	LET PARITY . . . . .	7-85
	LET SPEED. . . . .	7-85
	LET STOPS. . . . .	7-85
	LET SYNC . . . . .	7-86
	MICRO. . . . .	7-86
	ON . . . . .	7-86
	PRINT. . . . .	7-87
	PRINT A-Z. . . . .	7-88
	PRINT A\$-Z\$. . . . .	7-88
	PRINT FA-FZ. . . . .	7-88
	PRINT a-z . . . . .	7-88
	PRINT "[ <i>HEX number</i> ]". . . . .	7-89
	PRINT " <i>string constant</i> ". . . . .	7-89
	PRINT "( <i>string constant</i> )". . . . .	7-89
	PRINT %. . . . .	7-89
	PRINT #0 . . . . .	7-89

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	PRINT #1 . . . . .	7-89
	PRINT #2 . . . . .	7-89
	PRINT #3 . . . . .	7-90
	PRINT #4 . . . . .	7-90
	PRINT #5 . . . . .	7-90
	READ . . . . .	7-90
	REM . . . . .	7-90
	RESTORE . . . . .	7-90
	RETURN. . . . .	7-90
	SCREEN . . . . .	7-90
	SCREEN B . . . . .	7-90
	SCREEN C . . . . .	7-91
	SCREEN G a-z . . . . .	7-91
	SCREEN H . . . . .	7-91
	SCREEN I . . . . .	7-91
	SCREEN L . . . . .	7-92
	SCREEN N . . . . .	7-92
	SCREEN P = ____ . . . . .	7-92
	SCREEN R . . . . .	7-92
	SCREEN S . . . . .	7-92
	SCREEN SA-SY. . . . .	7-93
	SCREEN "string constant" . . . . .	7-93
	SCREEN SZ . . . . .	7-93
	SCREEN U . . . . .	7-93
	SCREEN Z . . . . .	7-93



TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	STATE. . . . .	7-94
	STATE BRG . . . . .	7-94
	STATE CRC = ____ . . . . .	7-94
	STATE DISCON/DISCOFF . . . . .	7-95
	STATE FAST/NFAST. . . . .	7-95
	STATE FCS = ____ . . . . .	7-95
	STATE FDUX. . . . .	7-96
	STATE FLTR/NFLTR . . . . .	7-96
	STATE HDUX. . . . .	7-96
	STATE LEAD=/NLEAD. . . . .	7-96
	STATE NLEAD8. . . . .	7-96
	STATE LOCK/NLOCK . . . . .	7-97
	STATE MODEM. . . . .	7-97
	STATE P3ONLY/NP3ONLY. . . . .	7-98
	STATE PIPE . . . . .	7-98
	STATE STAT/NSTAT. . . . .	7-98
	STATE SYNC/NSYNC . . . . .	7-99
	STATE TERM. . . . .	7-99
	STATE UPC/NUPC . . . . .	7-99
	STATE XIDL . . . . .	7-100
	STOP . . . . .	7-100
	TON/TOFF. . . . .	7-100
	TONP2. . . . .	7-100
	TONP2B . . . . .	7-101
	TONP2D. . . . .	7-101

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	TONP3. . . . .	7-101
	TONP3B . . . . .	7-101
	TONP3D . . . . .	7-101
	TONP4. . . . .	7-101
	TONP5. . . . .	7-101
	TONP6. . . . .	7-101
	TONP8. . . . .	7-101
	TONP11 . . . . .	7-101
	TONP18 . . . . .	7-102
	TONP20 . . . . .	7-102
	TONP22 . . . . .	7-102
	TONP24 . . . . .	7-102
	WAIT . . . . .	7-102
	WAIT A-Z . . . . .	7-102
	WAIT XDONE. . . . .	7-102
	WHEN . . . . .	7-103
	WHEN ERROR . . . . .	7-103
	WHEN FULL . . . . .	7-103
	WHEN KBD a-z . . . . .	7-103
	WHEN P ___ = 0/1. . . . .	7-103
	WHEN P2/P3 . . . . .	7-104
	WHEN P2/P3 a-z . . . . .	7-104
	WHEN P2/P3=A\$-Z\$. . . . .	7-104
	WHEN P2/P3=BCB. . . . .	7-105
	WHEN P2/P3=BCC. . . . .	7-105

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
7	WHEN P2/P3=BCG. . . . .	7-105
	WHEN P2/P3=FLAG. . . . .	7-105
	WHEN P2/P3="[HEX pair]" . . . . .	7-106
	WHEN P2/P3=PE . . . . .	7-106
	WHEN P2/P3="string constant" . . . . .	7-106
	WHEN RTC. . . . .	7-106
	WHEN TIMER. . . . .	7-107
	WHEN XIOIN . . . . .	7-107
	Front End Programming . . . . .	7-107
	CLR. . . . .	7-107
	CMA . . . . .	7-107
	DNZ. . . . .	7-107
	DSZ. . . . .	7-107
	FEM. . . . .	7-108
	HEX. . . . .	7-108
	INP . . . . .	7-108
	INU . . . . .	7-108
	IXO . . . . .	7-108
	IXU . . . . .	7-108
	JMP. . . . .	7-108
	MOV. . . . .	7-108
	MSK. . . . .	7-109
	NPE. . . . .	7-109
	PRE. . . . .	7-109
	QPE. . . . .	7-109

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
<b>7</b>	<b>RSN. . . . .</b>	7-109
	<b>SET . . . . .</b>	7-109
	<b>TAS. . . . .</b>	7-109
	<b>TAS A. . . . .</b>	7-110
	<b>TDS. . . . .</b>	7-110
	<b>TDS A. . . . .</b>	7-110
	<b>UPD. . . . .</b>	7-110
	<b>XBC. . . . .</b>	7-110
	<b>XMT. . . . .</b>	7-110
<b>8</b>	<b>X25/X75 OPERATION AND PROGRAMMING. . . . .</b>	8-1
	<b>1. GENERAL . . . . .</b>	8-1
	<b>PURPOSE . . . . .</b>	8-1
	<b>2. BACKGROUND. . . . .</b>	8-1
	<b>3. LEVEL-1, PHYSICAL INTERFACE. . . . .</b>	8-1
	<b>4. LEVEL-2, FRAME LEVEL. . . . .</b>	8-1
	<b>CHARACTER ORIENTED FRAMING (BISYNC, BSC). . . . .</b>	8-2
	<b>Start Flag . . . . .</b>	8-3
	<b>End Flag . . . . .</b>	8-3
	<b>Transparency. . . . .</b>	8-3
	<b>Block Check Sequence . . . . .</b>	8-3
	<b>BIT ORIENTED FRAMING (HDLC/SDLC). . . . .</b>	8-4
	<b>Flag Sequence Field . . . . .</b>	8-4
	<b>Address Field. . . . .</b>	8-4
	<b>Control Field. . . . .</b>	8-4

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
8	Information Field . . . . .	8-7
	Frame Check Sequence (FCS). . . . .	8-7
5.	LEVEL-3, PACKET LEVEL . . . . .	8-8
	General Format Identifier . . . . .	8-8
	Logical Channel Group Number . . . . .	8-9
	Logical Channel Number . . . . .	8-9
	Packet Type Identifier . . . . .	8-9
	Called DTE Address Length. . . . .	8-9
	Calling DTE Address Length . . . . .	8-9
	Called DTE Address . . . . .	8-10
	Calling DTE Address. . . . .	8-10
	Facility Field Length. . . . .	8-10
	User Data . . . . .	8-10
	CALL SET-UP AND CLEARING PACKETS . . . . .	8-11
	CAL? . . . . .	8-11
	CAL! . . . . .	8-12
	CLR? . . . . .	8-12
	CLR! . . . . .	8-13
	DATA AND INTERRUPT. . . . .	8-13
	Data Packet . . . . .	8-13
	INT?. . . . .	8-14
	INT!. . . . .	8-14
	FLOW CONTROL AND RESET . . . . .	8-15
	RR . . . . .	8-15
	RNR. . . . .	8-15

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
8	REJ . . . . .	8-16
	RSR? . . . . .	8-16
	RSR! . . . . .	8-17
	RESTART . . . . .	8-17
	RST? . . . . .	8-17
	RST!. . . . .	8-18
6.	TRANSMISSION PROCEDURES . . . . .	8-18
	LINK DOWN STATE . . . . .	8-18
	LINK SET-UP (LAPB) . . . . .	8-19
	LINK CLEAR-DOWN (LAPB) . . . . .	8-20
	LINK SET-UP (LAP) . . . . .	8-20
	LINK CLEAR-DOWN (LAP) . . . . .	8-21
	INFORMATION TRANSFER (LAP and LAPB) . . . . .	8-21
	A. Receiving Acknowledgement . . . . .	8-21
	B. Receiving Reject . . . . .	8-21
	C. Reset (LAPB) . . . . .	8-21
	D. Reset (LAP) . . . . .	8-21
7.	X25/X75 OPERATING SYSTEM . . . . .	8-23
8.	INITIAL OPERATION (Power-Up) . . . . .	8-23
9.	X25/X75 MENU . . . . .	8-27
10.	REAL TIME MONITOR AND DISPLAY . . . . .	8-29
	MONITOR PARAMETERS . . . . .	8-30
	Audible Alarm . . . . .	8-30
	Capture . . . . .	8-31
	Stop-on-Error. . . . .	8-31

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
8	Frame Delay . . . . .	8-31
	REAL TIME MONITOR. . . . .	8-32
	SPLIT SCREEN FORMAT. . . . .	8-35
	A. Frame Level Header. . . . .	8-35
	FLAGS. . . . .	8-35
	A . . . . .	8-35
	TYPE . . . . .	8-36
	NR . . . . .	8-36
	NS . . . . .	8-36
	PF. . . . .	8-36
	FCS. . . . .	8-36
	B. Packet Level Header. . . . .	8-36
	Q . . . . .	8-36
	D . . . . .	8-36
	LG . . . . .	8-36
	LCN. . . . .	8-36
	ID. . . . .	8-36
	PR . . . . .	8-37
	PS. . . . .	8-37
	M . . . . .	8-37
	OTC. . . . .	8-37
	Errors . . . . .	8-37
	X25 MENU . . . . .	8-37
	IO MODE. . . . .	8-37
	Speed . . . . .	8-37

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
8	Sync. . . . .	8-39
	NRZI . . . . .	8-39
	Mode . . . . .	8-39
	Stops . . . . .	8-39
	Clock . . . . .	8-39
	Parity . . . . .	8-39
	Lang. . . . .	8-39
	TYPICAL IO OPERATING PROCEDURE . . . . .	8-39
	PRINTER IO MODE . . . . .	8-41
	DISPLAY CAPTURED DATA . . . . .	8-41
	ERASE CAPTURE BUFFER. . . . .	8-43
	TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE. . . . .	8-43
11.	FAST CAPTURE AND DISPLAY. . . . .	8-48
	SET MONITOR PARAMETERS . . . . .	8-49
	Monitor . . . . .	8-49
	Capture . . . . .	8-49
	Display. . . . .	8-49
	Screen. . . . .	8-50
	START MONITOR. . . . .	8-50
	DISPLAY CAPTURED DATA . . . . .	8-50
	X25 MENU . . . . .	8-50
	SET LINE IO . . . . .	8-50
	SET PRINTER IO (XIO). . . . .	8-50
	LEVEL-3 COMMAND MODE . . . . .	8-51



TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
8	TYPICAL FAST CAPTURE AND DISPLAY OPERATING PROCEDURE. . . . .	8-51
12.	PACKET-LEVEL ACTIVE. . . . .	8-53
	X25 MENU. . . . .	8-54
	IO or XIO SET-UP. . . . .	8-54
	PRINT SCENARIO-BUFFER . . . . .	8-54
	SIMULATE DTE. . . . .	8-54
	SIMULATE DCE. . . . .	8-54
	SIMULATE DTE (LINK MONITOR). . . . .	8-54
	SIMULATE DCE (LINK MONITOR). . . . .	8-54
	SCENARIO-DRIVEN MONITOR . . . . .	8-54
	LINE-ACTIVITY-BUFFER DISPLAY . . . . .	8-55
	HOW TO WRITE A SCENARIO . . . . .	8-55
	TYPICAL SCENARIO ENTRY PROCEDURE . . . . .	8-57
	A. Interactive Displays . . . . .	8-59
	B. Example Flowchart . . . . .	8-61
	C. Example Scenario . . . . .	8-62
	D. Scenario Instruction Set . . . . .	8-66
13.	PACKET SPECIFICATIONS. . . . .	8-73
	Packet Type Name. . . . .	8-75
	HEX ID . . . . .	8-75
	Response. . . . .	8-75
	Class . . . . .	8-75
	Minimum Octets . . . . .	8-75
	Maximum Octets . . . . .	8-75

## TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
8	Addresses . . . . .	8-75
	Variable Length Addresses . . . . .	8-75
	Calling Address Length. . . . .	8-76
	Called Address Length. . . . .	8-76
	Number Facility Fields. . . . .	8-76
	Auxiliary Information . . . . .	8-76
	TYPICAL PACKET SPECIFICATION OPERATING PROCEDURE. . . . .	8-76
14.	LINK-TESTER . . . . .	8-79
15.	IO MODE. . . . .	8-81
16.	SET DATE AND TIME . . . . .	8-81
	TYPICAL DATE AND TIME OPERATING PROCEDURE . . . . .	8-82
17.	PRINTER IO MODE . . . . .	8-83
18.	LEVEL-3 COMMAND MODE . . . . .	8-83
9	OPERATOR MAINTENANCE . . . . .	9-1
	1. GENERAL . . . . .	9-1
	2. THE DISKETTE . . . . .	9-1
	DISK HANDLING . . . . .	9-2
	CLEANING. . . . .	9-2
	3. ERROR MESSAGES . . . . .	9-3
	4. TROUBLESHOOTING TIPS . . . . .	9-6
	5. APPLICATIONS ASSISTANCE. . . . .	9-7
	6. TRAINING . . . . .	9-7
	PACER-103 TRAINING . . . . .	9-7
	ENCORE INTRODUCTION . . . . .	9-7
	ENCORE ADVANCED PROGRAMMING . . . . .	9-8

TABLE OF CONTENTS (Cont'd)

<u>Chapter</u>	<u>Paragraph</u>	<u>Page</u>
	APPENDIX A. SPECIFICATION SUMMARY	
	APPENDIX B. DICOL	
	APPENDIX C. COMBASIC INSTRUCTION SUMMARY AND EXECUTION TIMES	
	APPENDIX D. CONTROL GRAPHICS AND CODE CHARTS	
	APPENDIX E. X25/X75 SCENARIO AND LINK TEST INSTRUCTIONS	
	APPENDIX F. FORMS AND RECORDS	
	INDEX	

LIST OF ILLUSTRATIONS

<u>Chapter</u>	<u>Figure</u>	<u>Page</u>
1	1-1 ENCORE 200 . . . . .	1-0
2	2-1 ENCORE External Equipment Connections . . . . .	2-2
	2-2 Installation Data . . . . .	2-3
3	3-1 ENCORE 200 . . . . .	3-1
	3-2 ENCORE Block Diagram . . . . .	3-5
	3-3 Standard Configuration. . . . .	3-9
	3-4 ENCORE 200 Front View. . . . .	3-13
	3-5 ENCORE 200 Rear View . . . . .	3-15
4	4-1 Typical Operating Configuration. . . . .	4-1
	4-2 Power-Up Flowchart. . . . .	4-2
	4-3 Diskette Insertion. . . . .	4-3

## LIST OF ILLUSTRATIONS (Cont'd)

<u>Chapter</u>	<u>Figure</u>	<u>Page</u>
4	4-4 Master Directory . . . . .	4-4
5	5-1 Level-1 Directory . . . . .	5-1
	5-2 Keystroke Flowchart, Level-1 Directory . . . . .	5-2
	5-3 Asynchronous Library . . . . .	5-3
	5-4 Keystroke Flowchart, Asynchronous Library. . . . .	5-3
	5-5 Keystroke Flowchart, Monitor/Trap . . . . .	5-5
	5-6 Synchronous Library . . . . .	5-7
	5-7 Keystroke Flowchart, Synchronous Library . . . . .	5-8
	5-8 Typical Monitor Display . . . . .	5-11
	5-9 Line Totals. . . . .	5-12
	5-10 Station Totals. . . . .	5-12
	5-11 Bit Oriented Library. . . . .	5-13
	5-12 Keystroke Flowchart, Bit Oriented Library . . . . .	5-14
	5-13 Off-Line Data Display Library. . . . .	5-16
	5-14 Keystroke Flowchart, Off-Line Data Display Library. . . . .	5-16
	5-15 Measurement Library . . . . .	5-23
	5-16 Keystroke Flowchart, Measurement Library. . . . .	5-23
	5-17 BERT Measurement Results. . . . .	5-24
	5-18 CERT Measurement Results. . . . .	5-25
	5-19 CPU Response Time . . . . .	5-26
	5-20 ACK/NAK Analysis . . . . .	5-27
	5-21 Poll Analysis . . . . .	5-28
	5-22 Monitor RTS/CTS Delay . . . . .	5-29
	5-23 Measure RTS/CTS Delay as DTE. . . . .	5-30

LIST OF ILLUSTRATIONS (Cont'd)

<u>Chapter</u>	<u>Figure</u>	<u>Page</u>
5	5-24 XIO DCE/DTE Cable. . . . .	5-31
	5-25 EIA Timing Monitor . . . . .	5-33
6	6-1 Level-2 Directory . . . . .	6-2
	6-2 Program Mode Display, Menu Not Loaded. . . . .	6-4
	6-3 Program Mode Display, Menu Loaded. . . . .	6-4
	6-4 Typical Execute Display . . . . .	6-7
	6-5 Typical Captured Data Display . . . . .	6-25
	6-6 Typical Full and Half Duplex Lines. . . . .	6-25
	6-7 Level-2 Menu Print . . . . .	6-32
	6-8 PACER to ENCORE Menu Load. . . . .	6-34
	6-9 Menu Load from Diskette. . . . .	6-35
	6-10 ENCORE to PACER Menu Dump. . . . .	6-37
	6-11 Menu Dump to Diskette . . . . .	6-38
7	7-1 ASGN Mode Format . . . . .	7-3
	7-2 Power-Up Default IO. . . . .	7-10
	7-3 Typical Storage Menu . . . . .	7-12
	7-4 Typical Disc Menu. . . . .	7-13
	7-5 Time Entry Mode . . . . .	7-16
	7-6 Status Mode . . . . .	7-17
	7-7 Variable Entry Mode. . . . .	7-17
	7-8 XIO Mode Display . . . . .	7-19
	7-9 Attribute Byte . . . . .	7-41
	7-10 Status Byte. . . . .	7-43

## LIST OF ILLUSTRATIONS (Cont'd)

<u>Chapter</u>	<u>Figure</u>	<u>Page</u>
8	8-1 Character Oriented (BSC) Frame Structure . . . . .	8-2
	8-2 Bit Oriented (HDLC/SDLC) Frame Structure . . . . .	8-4
	8-3 Information Transfer Frame. . . . .	8-5
	8-4 Supervisory Command/Response Frames . . . . .	8-5
	8-5 Unnumbered Command/Response Frames. . . . .	8-6
	8-6 Call Request or Incoming Call Packet . . . . .	8-11
	8-7 Call Accepted or Call Connected Packet . . . . .	8-12
	8-8 Clear Request or Clear Indication Packet. . . . .	8-12
	8-9 Clear Confirmation Packet . . . . .	8-13
	8-10 Data Packet . . . . .	8-13
	8-11 Interrupt Packet. . . . .	8-14
	8-12 Interrupt Confirmation Packet. . . . .	8-14
	8-13 Receive Ready Packet . . . . .	8-15
	8-14 Receive Not Ready Packet . . . . .	8-15
	8-15 Reject Packet. . . . .	8-16
	8-16 Reset Request Packet . . . . .	8-16
	8-17 Reset Confirmation Packet. . . . .	8-17
	8-18 Restart Request Packet . . . . .	8-17
	8-19 Restart Confirmation Packet . . . . .	8-18
	8-20 LAPB Set-Up Diagram and Typical Display . . . . .	8-19
	8-21 LAP Set-Up Diagram and Typical Display. . . . .	8-20
	8-22 X25/X75 Power-Up Flowchart. . . . .	8-23
	8-23 Diskette Insertion . . . . .	8-24
	8-24 X25/X75 Menu (HDLC/SDLC). . . . .	8-25
	8-25 X25/X75 Menu (BISYNC). . . . .	8-26

LIST OF ILLUSTRATIONS (Cont'd)

<u>Chapter</u>	<u>Figure</u>	<u>Page</u>
8	8-26 X25/X75 Menu Keystrokes (HDLC/SDLC). . . . .	8-27
	8-27 X25/X75 Menu Keystrokes (BISYNC). . . . .	8-28
	8-28 Real Time Monitor and Display Submenu . . . . .	8-29
	8-29 Real Time Monitor Flowchart . . . . .	8-29
	8-30 Monitor Parameters Prompts . . . . .	8-30
	8-31 Typical Display Mode Format . . . . .	8-32
	8-32 Typical Monitor Display Frame and Packet . . . . .	8-34
	8-33 Typical Monitor Display Frame Only . . . . .	8-34
	8-34 Typical Monitor Display Packet Only. . . . .	8-35
	8-35 Power-Up Default IO. . . . .	8-40
	8-36 Real Time Monitor and Display Submenu . . . . .	8-43
	8-37 Monitor Parameters Prompts . . . . .	8-44
	8-38 Typical Monitor Format . . . . .	8-45
	8-39 Passive Monitor Menu . . . . .	8-48
	8-40 Passive Monitor Menu Flowchart. . . . .	8-48
	8-41 BOP Monitor Parameters Set-Up. . . . .	8-49
	8-42 X25 Active Menu . . . . .	8-53
	8-43 X25 Active Menu Flowchart. . . . .	8-53
	8-44 Scenario Entry Display. . . . .	8-57
	8-45 Scenario Display Mode . . . . .	8-60
	8-46 Line Activity Display . . . . .	8-60
	8-47 Scenario Flowchart . . . . .	8-61
	8-48 Finite Packet Definition . . . . .	8-76
	8-49 Formatted Packet Definition . . . . .	8-77
	8-50 Link-Tester Display . . . . .	8-79

## LIST OF ILLUSTRATIONS (Cont'd)

<u>Chapter</u>	<u>Figure</u>	<u>Page</u>
8	8-51 Time Entry Format . . . . .	8-82
9	9-1 Typical Diskette. . . . .	9-1

## LIST OF TABLES

<u>Chapter</u>	<u>Table</u>	<u>Page</u>
2	2-1 PACKING DATA . . . . .	2-1
3	3-1 RS-232 INTERFACE PIN ASSIGNMENT. . . . .	3-4
	3-2 EQUIPMENT SUPPLIED . . . . .	3-8
	3-3 OPTIONS. . . . .	3-10
	3-4 CONTROLS, DISPLAYS, AND CONNECTORS. . . . .	3-11
	3-5 CONTROLS, DISPLAYS, AND CONNECTORS. . . . .	3-14
	3-6 USE OF CONTROL CLUSTER AND SPECIAL FUNCTION KEYS. . . . .	3-18
5	5-1 MONITOR/TRAP COMMANDS. . . . .	5-3
	5-2 DISPLAY ATTRIBUTES. . . . .	5-6
	5-3 HDUX BUFFER DISPLAY COMMANDS. . . . .	5-8
	5-4 HDUX DISC DISPLAY COMMANDS . . . . .	5-10
	5-5 CAPTURE EIA STATUS . . . . .	5-11
	5-6 STATUS AND DATA BUFFER DISPLAY COMMANDS . . . . .	5-11
	5-7 STATUS AND DATA DISC DISPLAY COMMANDS. . . . .	5-12
	5-8 TRANSPARENT MONITOR COMMANDS. . . . .	5-16
	5-9 MASTER/SLAVE SPECIAL FUNCTION KEYS . . . . .	5-31



LIST OF TABLES (Cont'd)

<u>Chapter</u>	<u>Table</u>	<u>Page</u>
6	6-1 PACER/ENCORE PROGRAM MODE COMMANDS. . . . .	6-5
	6-2 DICOL/LEVEL-2 INSTRUCTION SET. . . . .	6-8
	6-3 DISPLAY MODE COMMANDS. . . . .	6-26
7	7-1 LEVEL-3 COMMANDS. . . . .	7-2
	7-2 EDIT MODE COMMANDS AND KEYSTROKES. . . . .	7-5
	7-3 LANGUAGES WITH PARITY . . . . .	7-9
	7-4 XIO BIT AND PARITY SELECTION . . . . .	7-19
	7-5 XIO SPEEDS . . . . .	7-19
	7-6 INSTRUCTION OPERATORS . . . . .	7-25
	7-7 COMBASIC STATEMENTS USING INTEGER VARIABLES. . . . .	7-27
	7-8 COMBASIC STATEMENTS USING FLOATING POINT VARIABLES . . . . .	7-29
	7-9 COMBASIC STATEMENTS USING STRING VARIABLES. . . . .	7-31
	7-10 COMBASIC STATEMENTS USING BYTE VARIABLES. . . . .	7-33
	7-11 RESERVED VARIABLES . . . . .	7-36
	7-12 DISPLAY ATTRIBUTES . . . . .	7-42
	7-13 COMBASIC STATEMENTS FOR CONTROL OF THE CAPTURE BUFFER . . . . .	7-44
	7-14 COMBASIC STATEMENTS FOR DISC CONTROL . . . . .	7-45
	7-15 BLOCK CHECK SCHEMES AND POLYNOMIALS . . . . .	7-46
	7-16 COMBASIC STATEMENTS TO USE FOR BLOCK CHECK CALCULATIONS . . . . .	7-47
	7-17 BOOT CONTROL ENTRIES. . . . .	7-62
8	8-1 GENERAL FORMAT IDENTIFIER . . . . .	8-8
	8-2 PACKET TYPE IDENTIFIER . . . . .	8-9

## LIST OF TABLES (Cont'd)

<u>Chapter</u>	<u>Table</u>	<u>Page</u>
<b>8</b>	<b>8-3</b> DIAGNOSED ERRORS . . . . .	8-31
	<b>8-4</b> MONITOR DISPLAY COMMANDS . . . . .	8-33
	<b>8-5</b> CAPTURE DISPLAY COMMANDS . . . . .	8-42
	<b>8-6</b> INTERACTIVE SCENARIO AND DISPLAY COMMANDS .	8-55
	<b>8-7</b> LAP MOD . . . . .	8-72
	<b>8-8</b> DEFAULT PACKET DEFINITIONS. . . . .	8-74
	<b>8-9</b> LINK TEST INSTRUCTIONS . . . . .	8-80
<b>9</b>	<b>9-1</b> ERROR CODES. . . . .	9-3
	<b>9-2</b> TROUBLESHOOTING TIPS . . . . .	9-6



**TERMS AND DEFINITIONS**

<b>Abort</b>	A frame level function invoked by sending a primary or secondary station causing the recipient to discard (and ignore) all bit sequences transmitted by the sender since the preceding flag sequence.
<b>Address</b>	A character or characters used to define a specific device or cluster controller.
<b>Address Field</b>	The sequence of eight bits immediately following the opening flag of a frame identifying the secondary station sending (or designated to receive) the frame.
<b>Alphanumeric</b>	A character set containing both letters and numerals.
<b>Argument</b>	A variable or constant immediately following a COMBASIC instruction. The instruction and argument combine to form the COMBASIC statement.
<b>Attribute</b>	A characteristic of the display field, such as intensified, highlight, protected, or unprotected.
<b>Binary</b>	The number system with a base of two.
<b>Binary Code</b>	A code using two distinct characters, "0" and "1".
<b>Binary Synchronous Communications (BISYNC)</b>	A data link control procedure which uses character synchronization.
<b>Bit</b>	A binary digit.
<b>Block</b>	A set of consecutive characters handled as a unit.
<b>Bug</b>	A mistake in the design of the program.
<b>Byte</b>	A group of eight binary digits of bits normally operated upon as an entity.
<b>Call Data</b>	Data provided to the called terminal for call set-up; information such as process identification and passwords.
<b>Call Supervision Packets</b>	Packets used in the establishment and clearing phases of a call.
<b>Capture Memory</b>	That section of ENCORE memory dedicated to the storage of transmit and receive data and interface status of that data.
<b>Carriage Return</b>	A teletype operation causing the next character to be printed at the left margin.

**TERMS AND DEFINITIONS (Cont'd)**

<b>Central Processing Unit</b>	That part of a computer system that controls interpretation and execution of instructions.
<b>Character</b>	A single letter, numeral, or symbol used to represent information.
<b>Circuit Switching</b>	A method of communications where a connection between calling and called stations is established on demand for exclusive use of the circuit until the connection is released.
<b>Cluster</b>	Two or more devices sharing the same controller.
<b>Command</b>	The content of the control field of a command frame sent by the primary instructing the addressed secondary to perform some specific function.
<b>Command Frame</b>	All frames that may be transmitted by the primary station are referred to as command frames.
<b>Command Set</b>	A list of operator entered keyboard commands that provide for mode select and edit capability.
<b>Compile</b>	To create an object (binary coded) program from a program written in the source (COMBASIC) language.
<b>Constant</b>	Numeric argument used, but not changed by the program.
<b>Control Field</b>	The sequence of eight bits immediately following the address field of a frame. The content of the control field is interpreted as follows: a) by the receiving secondary, designated by the address field, as a command instructing the performance of some specific function; b) by the receiving primary, as a response from the secondary, designated by the address field, to one or more commands.
<b>Conversational Program</b>	A program interacting dynamically with on-line operators.
<b>Cyclic Redundancy Check (CRC)</b>	A block checking the method in which the numeric binary value of a block is divided by a constant divisor. The quotient is discarded and the remainder serves as a block check sequence (BCS). The transmitting station calculates and transmits the BCS; the receiving station compares the transmitted remainder to its own computer remainder and finds no error if they are equal.

**TERMS AND DEFINITIONS (Cont'd)**

<b>Data Circuit - Terminating Equipment (DCE)</b>	The equipment installed at the user's premises which provides: a) all the functions required to establish, maintain, and terminate a connection; b) the signal conversion and coding between the data terminal equipment (DTE) and the common carrier's line, e.g., data set, modem. DCE may be further interpreted as the network side of the DTE/DCE interface.
<b>Data Terminal Equipment (DTE)</b>	Any piece of equipment at which a communications path begins or ends, i.e., the complete user's installation.
<b>Debug</b>	To detect, locate, and correct program errors.
<b>Diagnostic</b>	Pertaining to the detection and isolation of a system malfunction.
<b>Execute</b>	To carry out an instruction or run a program.
<b>Flag Sequence</b>	The unique sequence of eight bits (01111110) which delimits the beginning and ending of a frame.
<b>Flowchart</b>	A graphical representation of the operation required to execute a program.
<b>Flow Control Packets</b>	Packets used to control the flow of data on a logical channel between the terminal and network.
<b>Frame</b>	The sequence of contiguous bits, bracketed by and including beginning and ending flag sequences. A valid frame contains at least 32 bits between flags and contains an address field, a control field, and a frame check sequence. A frame may or may not include an information field.
<b>Frame Check Sequence (FCS)</b>	The field immediately preceding the ending flag of a frame containing the bit sequence that provides for the detection of transmission errors by the receiver.
<b>Frame Level Logical Interface</b>	The Frame Level Logical Interface is involved in the administration of the physical link between the terminal and the network node.
<b>Full Duplex</b>	A communications channel capable of transmitting and receiving simultaneously.
<b>Half Duplex</b>	A communications channel capable of transmitting and receiving but only in one direction at a time.
<b>High-Level Data Link Control (HDLC)</b>	Bit-oriented data link control procedure presently under discussion by the International Organization for Standardization (ISO).

**TERMS AND DEFINITIONS (Cont'd)**

<b>Information Field</b>	The sequence of bits occurring between the last bit of the control field and the first bit of the frame check sequence. The information field contents are not interpreted at the frame level.
<b>Initialize</b>	To start at the beginning, setting addresses, counters, and switches to zero or some other prescribed starting point.
<b>Instruction</b>	That part of the COMBASIC statement describing the operation to be performed.
<b>Interrupt Procedure</b>	A procedure which allows a DTE to transmit signalling data on a virtual circuit without following the flow control procedures applying to data packets.
<b>Invalid Frame</b>	A sequence of bits, following the receipt of an apparent beginning flag sequence that either: a) is terminated by an abort sequence, or b) contains less than 32 bits before an apparent ending flag sequence is detected.
<b>Line Code</b>	ASCII or EBCDIC code, etc., used in communications with a computer, terminal, etc..
<b>Logical Channel</b>	Logical channels exist between the packet terminal and the network node and identify the permanent or switched virtual circuit.
<b>Mass Storage</b>	The data cartridge or disc which stores large amounts of data readily accessible to capture memory.
<b>Message</b>	A message is the basic unit of interprocess information transfer and consists of control information and process data.
<b>Object Program</b>	The binary coded program which is stored in the object buffer and output after translation from the source language.
<b>Octet</b>	A group of eight digits. Synonymous with byte.
<b>Packet</b>	A packet is the basic transmission unit on an access data link and consists of a packet header and a data field. Communication between the user and the Datapac Network takes the form of packets.
<b>Packet Level Logical Interface</b>	The Packet Level Logical Interface specifies the manner in which calls are established, maintained, and cleared and how user data and control information are structured into packets.

**TERMS AND DEFINITIONS (Cont'd)**

<b>Packet Switching</b>	The transfer of data by means of addressed packets whereby the network bandwidth is only occupied for the duration of transmission of the packet. The channel is then available for the transfer of other packets. In contrast with circuit switching, the data network may determine the routing during, rather than prior to, the transfer of a packet.
<b>Packet Terminal</b>	A packet terminal is a terminal which communicates with a network circuit over which only data, reset, interrupt, and flow control packets can flow.
<b>Physical Interface</b>	The physical connection between DTE and DCE.
<b>Poll</b>	An interrogation process to determine if a specific device or any device in a cluster has data to send.
<b>Primary</b>	That portion of the station responsible for control of the data communication link. The primary generates commands and interprets responses. Specific responsibilities assigned to the primary include: a) initialization of (data and control) information interchange; b) organization and control of data flow; c) retransmission control; and d) all recovery functions at the link level.
<b>Protocol</b>	A formal set of conventions governing the format and control of inputs and outputs.
<b>Ready State</b>	A DTE has at least one logical channel. The ready state is assumed when there is no call in existence on that channel.
<b>Reset Procedure</b>	A procedure which allows a DTE to reinitialize the flow control procedure on a virtual circuit and, in so doing, to remove all data and interrupt packets which may be in transit at the time of resetting.
<b>Response</b>	The content of the control field of a response frame advising the primary with respect to the processing by the secondary of one or more command frames.
<b>Response Frame</b>	All frames that may be transmitted by a secondary station are referred to as response frames.
<b>Restart Procedure</b>	A procedure which allows a DTE to simultaneously clear all its switched virtual circuits and reset all its permanent virtual circuits (see definition of reset procedure).
<b>Secondary</b>	That portion of a station responsible for performing link level operations, as instructed by the primary. A secondary interprets commands and generates responses.



### TERMS AND DEFINITIONS (Cont'd)

<b>Selection</b>	A selection process is used to determine if a specific device can receive data.
<b>Single Access Packet Terminal</b>	Such a terminal is capable of handling only a single virtual circuit at a given time and is authorized by the network to do so.
<b>Source Program</b>	A program written in COMBASIC and stored in the source buffer.
<b>Stored Program</b>	A program written in COMBASIC and stored in nonvolatile battery supported memory (storage).
<b>Unnumbered Commands</b>	The commands that do not contain sequence numbers in the control field.
<b>Unnumbered Responses</b>	The responses that do not contain sequence numbers in the control field.
<b>Variable</b>	A symbol whose value changes during execution of a program.

### ABBREVIATIONS

<b>ACUI</b>	ASCII Communication Interface Unit
<b>ASCII</b>	American Standard Code for Information Interchange
<b>ASYN</b>	Asynchronous
<b>BASIC</b>	Beginners All-Purpose Symbolic Instruction Code
<b>BCS</b>	Block Check Sequence
<b>BERT</b>	Bit/Block Error Rate Test
<b>BSC</b>	Character Oriented
<b>CAL?</b>	Call Request/Incoming Call
<b>CAL!</b>	Call Accept/Call Connected
<b>CCITT</b>	International Telegraph and Telephone Consultive Committee
<b>CERT</b>	Character Error Rate Test
<b>CLR?</b>	Clear Request/Clear Indication

## ABBREVIATIONS (Cont'd)

<b>CLR!</b>	Clear Confirmation
<b>CMDR</b>	Command Reject
<b>COMBASIC</b>	Communications BASIC
<b>CPU</b>	Central Processing Unit
<b>CRT</b>	Cathode Ray Tube
<b>CTRL</b>	Control
<b>CTS</b>	Clear to Send
<b>DA</b>	Device Address
<b>DCD</b>	Data Carrier Detect
<b>DCE</b>	Data Communications Equipment
<b>DICOL</b>	Digitech Communications Language
<b>DISC</b>	Disconnect
<b>DLE</b>	Data Link Escape
<b>DM</b>	Disconnect Response Mode
<b>DMA</b>	Direct Memory Access
<b>DTE</b>	Data Terminal Equipment
<b>DTR</b>	Data Terminal Ready
<b>DSR</b>	Data Set Ready
<b>EBCDIC</b>	Extended Binary Coded Decimal Interchange Code
<b>EIA</b>	Electronics Industry Association
<b>F</b>	Final Bit (secondary)
<b>FCS</b>	Frame Check Sequence
<b>FDUX</b>	Full Duplex
<b>HDLC</b>	High-Level Data Link Control
<b>HDUX</b>	Half Duplex
<b>IC</b>	Incorrect Control Field (frame level)

**ABBREVIATIONS (Cont'd)**

<b>ID</b>	Identification
<b>INT?</b>	Interrupt
<b>INT!</b>	Interrupt Confirmation
<b>INFO</b>	Information Frame
<b>IO</b>	Input/Output
<b>IP</b>	Incorrect Packet Identification (packet level)
<b>LAP</b>	Link Access Procedure
<b>LAPB</b>	Link Access Procedure, Balanced
<b>LCN</b>	Logical Channel Number
<b>LCGN</b>	Logical Channel Group Number
<b>LRC</b>	Longitudinal Redundancy Check
<b>N(R)</b>	Transmitter Receive Sequence Number
<b>NRZI</b>	Non-Return-to-Zero-Inverted
<b>N(S)</b>	Transmitter Send Sequence Number
<b>NSS</b>	N(S) Sequence error (frame level)
<b>NUM</b>	Numeric
<b>OTC</b>	Octet Count
<b>P</b>	Poll Bit (primary)
<b>PAR</b>	Parity
<b>P/F</b>	Poll/Final Bit
<b>P(R)</b>	Packet Receive Sequence Number
<b>P(S)</b>	Packet Send Sequence Number
<b>RAM</b>	Random Access Memory
<b>REJ</b>	Reject
<b>RLSD</b>	Receive Line Signal Detect
<b>RNR</b>	Receiver Not Ready

**xxxx**

**ABBREVIATIONS (Cont'd)**

<b>ROM</b>	Read Only Memory
<b>RR</b>	Receiver Ready
<b>RSR?</b>	Reset Request/Reset Indication
<b>RSR!</b>	Reset Confirmation
<b>RST?</b>	Restart Request/Restart Indication
<b>RST!</b>	Restart Confirmation
<b>RTS</b>	Request to Send
<b>S</b>	Supervisory Command/Response
<b>SABM</b>	Set Asynchronous Balance Mode
<b>SARM</b>	Set Asynchronous Response Mode
<b>SDLC</b>	Synchronous Data Link Control
<b>SID</b>	Station Identification Device
<b>SPA</b>	Station Poll Address
<b>SQ</b>	Signal Quality
<b>SSA</b>	Station Select Address
<b>STX</b>	Start of Text
<b>SYNC</b>	Synchronous
<b>TFO</b>	Too Few Octets
<b>TMO</b>	Too Many Octets
<b>U</b>	Unnumbered Command/Response
<b>UA</b>	Unnumbered Acknowledge
<b>uP</b>	Microprocessor
<b>V(R)</b>	Receive Variable
<b>V(S)</b>	Send Variable
<b>XIO</b>	External Input/Output

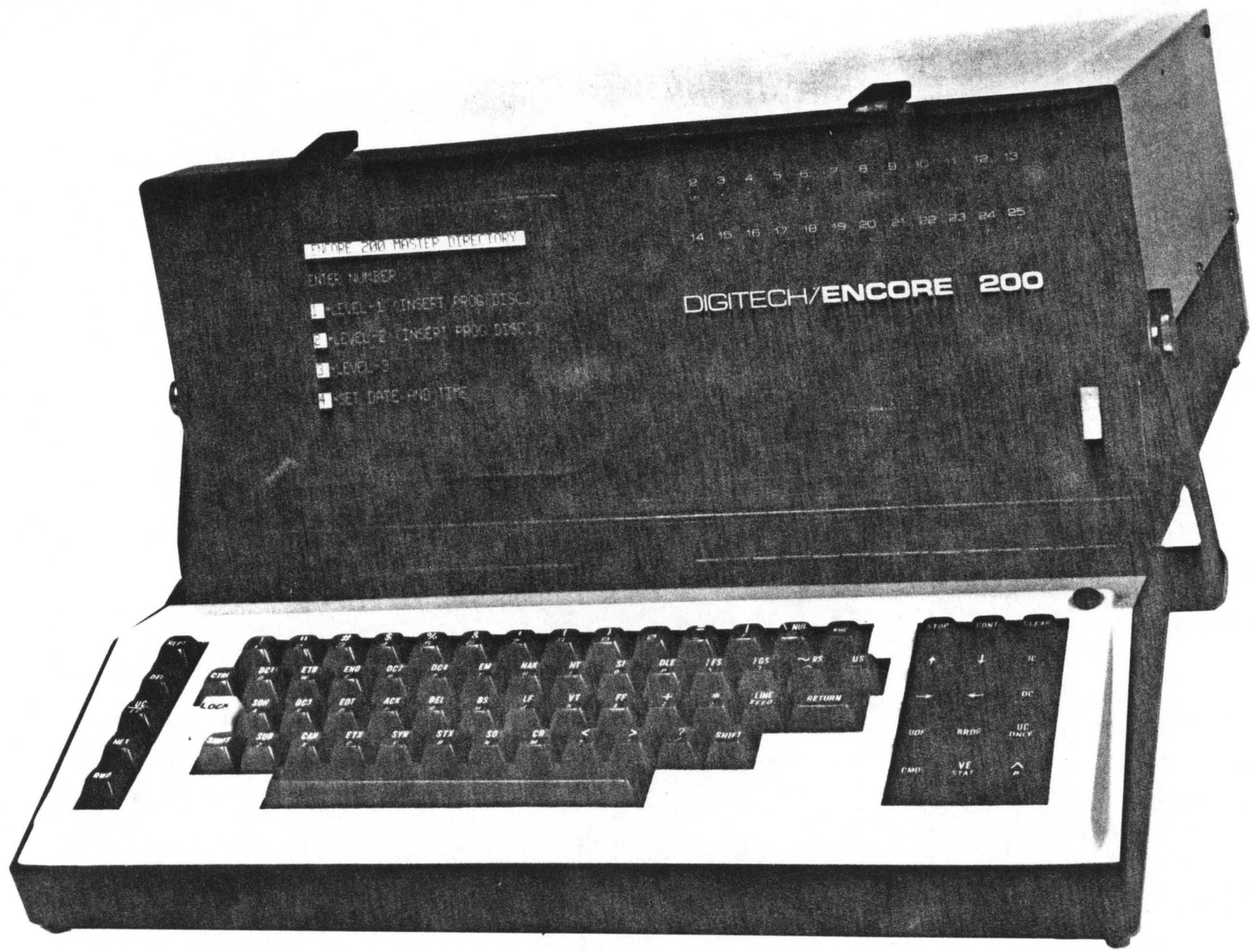


Fig. 1-1 Encore 200

## CHAPTER 1

### INTRODUCTION

#### 1. GENERAL

**1.01** Recent advancements in technology and the need to efficiently transmit and receive enormous amounts of data has created a unique environment where highly complex data networks and communications protocols have evolved. Network analysis in this environment requires the use of sophisticated diagnostic instruments which can be used for software development and network performance measurement. These instruments must be portable, easy to use, and fully programmable in order to isolate problems effectively and to minimize downtime. DIGITECH's ENCORE 200 is such an instrument. Its major features include the following:

- Three levels of operation to accommodate any user regardless of programming experience.
- Fully programmable in multiple languages.
- Complete repertoire of monitor/interactive diagnostic programs accessed by a menu driven operating system.
- Comprehensive X25/X75 diagnostic capabilities.
- Self diagnostics including power-up tests and error messages.
- Fully compatible with PACER-103 and ENCORE 100.
- Unlimited program and data storage using dual sided double density 5¼ inch diskettes.
- Authentic full ASCII terminal-style keyboard.
- Dynamic allocation to object, source, and capture memories.
- Automatic program execution on power-up.
- 26 user defined function keys.
- Remote control of slave ENCORE.
- Off-line load and dump of programs/data.
- Real time clock for event trigger and time stamp.

**1.02** To accommodate a broad range of "user expertise", the ENCORE may be operated in any one of three levels. Level-1, generally appealing to the nonprogramming field technician, is a menu driven system that retrieves and executes programs from a resident library. Level-2, for the DICOL (DIGITECH COMMUNICATIONS LANGUAGE) user, makes the PACER-103 upward compatible with the ENCORE. Level-3 allows the user to write custom programs, a technique proven to be invaluable in the development of communications system software.

**1.03** The ENCORE is capable of executing user designed programs which permit operation as an intelligent monitor or interactive simulator of network components (DTE or DCE). The integration of many functions under the control of a single program makes the ENCORE a very powerful and versatile instrument for solving complex data communications problems quickly and easily. User designed programs are written in two languages: DICOL and COMBASIC. DICOL is a high order language developed by DIGITECH for use with the PACER-103. It consists of a series of 2-digit codes representing a variety of instructions used in programming the PACER. The ability to write and execute DICOL programs in Level-2 makes PACER upward compatible with the ENCORE 200. COMBASIC (COMMunications BASIC) is an enhanced form of Dartmouth BASIC developed by DIGITECH for use in the ENCORE 100. It is the same language used in Level-3 of the ENCORE 200, making both of these units fully compatible. Both DICOL and COMBASIC programs can be edited on the ENCORE 200 and stored in nonvolatile program memory or on disc for later recall and edit or execution.

**1.04** The ENCORE is provided with a complete repertoire of nearly 100 programs. These include the programs used in Level-1, which can be modified and executed independently, and several utilities which are only accessed in Level-3. This extensive library of programs allows the ENCORE to immediately begin diagnosing network problems. Execution of selected programs permits intelligent monitor and recording of data and interface status on a network employing interval counters, timers, triggers, and alarms. Under program control, the ENCORE can simulate an interactive network component, it can be used to isolate (on-site) the device and link and subsequently test the network or device hardware and software. This interactive capability also allows Original Equipment Manufacturers to develop and test new or custom devices, systems, protocols, and networks before implementation in the user's environment. Other programs in the library can be used to optimize a network by measuring the efficiency and initialization of various links, measuring such parameters as response time, line utilization, and error performance.

**1.05** A comprehensive X25/X75 applications package can be supplied as an option to ENCORE 200. It is a disc based menu driven operating system providing both monitor and interactive capabilities. User written scenarios may be executed in the active modes while the passive modes permit real time monitor, capture, and display of both packet and frame level information.

**1.06** Self diagnostics are an important part of any microprocessor based test instrument and ENCORE 200 is no exception. During power-up, for example, the ENCORE performs a series of tests to determine the operational capability of the unit and displays a message to the user indicating that the unit has either completed or failed the test. In addition, there are more than 30 different error messages which alert the user to illegal procedures or failures within the system.

**1.07** The ENCORE 200 maintains the same disc based operating system used in the ENCORE 100, and is therefore fully compatible with the ENCORE 100 and the PACER-103. From the user's point of view, all three levels of operation are identical for both units. Both program and data discs are fully interchangeable. DICOL and COMBASIC programs written on the PACER-103 or ENCORE 100 may be transferred to and edited or executed by the ENCORE 200 and vice versa.

**1.08** A single 5¼ inch disc drive provides the ENCORE 200 with up to 240K bytes of permanent program and data storage on a single dual sided double density diskette. Additional diskettes expand the storage capabilities and permit easy upgrade for X25/X75 and SNA applications. A complete series of disc related commands and utility programs allow the user to format and copy diskettes as well as transfer programs and data from diskette to solid state memory and vice versa.

## GENERAL INFORMATION, USING THE MANUAL

**1.09** An authentic terminal-style ASCII keyboard provides the ultimate in tactile feedback and is further enhanced by audible verification of each keystroke. The large nonglare keys are labeled with both ASCII character and control legends. Additional keys provide control of the cursor and special ENCORE functions.

**1.10** Dynamic memory allocation allows the user to assign specific amounts of source, object, and capture memory manually or automatically. This permits the most efficient use of all memory and the reallocation of memory to fit the needs of individual programs as they are executed. Up to 21.75K may be allocated to the object and capture memories; up to 18K may be allocated to source.

**1.11** A unique feature of the ENCORE is its ability to execute any stored program on power-up. The power-up assignment may include the name of the program to be executed along with appropriate memory allocation or other command sequences. This assignment is executed immediately following the self test diagnostics performed whenever the unit is turned on. Execution of the power-up sequence eliminates the need for operator intervention and permits instant diagnostics of the circuit under test.

**1.12** Up to 26 user defined functions of up to 16 keystrokes each may be written to significantly reduce the total keystrokes required for repetitive entries. As an example, two keystrokes,   could perform the same function as typing EDIT  LIST . Each function may include any combination of keystrokes for command entry or program execution.

**1.13** Remote control of a slave ENCORE is accomplished in Level-1 by simple menu selection. The physical connections are made via the RS-232 compatible XIO ports. Once the master/slave is established, the slave ENCORE responds to commands entered via the master ENCORE's keyboard. The contents of the slave screen are viewed at the master ENCORE through the use of the , , and  keys which transfer the contents of the slave screen to the master, stops the transfer, and exits the Master/Slave Mode, respectively.

**1.14** Off-line program load and dump is accomplished in the Edit Mode. Using the LOAD and DUMP commands, the ENCORE is capable of transferring programs to or from another ENCORE, PACER, or other recording device. The transfer occurs at the RS-232 compatible XIO port at any one of 16 speeds up to 19.2kbps. Programs are always input or output in ASCII. In addition, the ENCORE can also off-line dump captured data to another device via the XIO port. This is accomplished under Level-1 program control. Data is always output in the same language as it was captured.

**1.15** The real time clock is a convenient means of triggering and time stamping events. It can be used to initiate or terminate tests at a specific time, conduct tests for a specific duration, and record the data and time at which a specific event occurs. It is extremely useful in diagnosing problems that reoccur at regular intervals.

## 2. USING THE MANUAL

**2.01** This manual contains information required for the operation, programming, and operator maintenance of the Data Network Diagnostic Monitor/Simulator, ENCORE 200. The manual is separated into nine chapters.

**2.02** Chapter 1 is a brief introduction to the ENCORE 200. It covers the unit's major features and provides the reader with a quick glance at the ENCORE's overall capabilities.



**2.03** Chapter 2, Unpacking and Inspection, provides the user with the information required for unpacking, installation, and inspection of the unit upon receipt. Since the unit is portable and designed for desk top use, installation is minimal. Illustrations included in this chapter give overall dimensions and show rear panel connections for ac power and for interface with the RS-232 circuit.

**2.04** Chapter 3, Product Description, is a more detailed description of the ENCORE 200 including design philosophy, architecture, basic configuration, options, and hardware controls. These controls are detailed using full page photographs of the front and rear of the unit.

**2.05** Chapter 4, Initial Set-Up and Operation, contains information of particular value to the first time user. It includes the instructions required for power-up and gives examples of operation in Level-1, Level-2, and Level-3.

**2.06** Chapters 5, 6, and 7 detail operation in Level-1, Level-2, and Level-3, respectively. These chapters cover software control of the ENCORE, use of all command and operating modes, and how to write programs in DICOL and COMBASIC.

**2.07** Chapter 8 is devoted to operation of the ENCORE in an X25/X75 environment. It includes a discussion of X25/X75 requirements at both the frame level and packet level and how the ENCORE operates as a monitor or interactive simulator while adhering to these requirements. The optional X25/X75 operating capability is provided as a result of highly developed applications software.

**2.08** Chapter 9 includes operator maintenance procedures and troubleshooting tips. It also provides the information necessary to obtain applications assistance and additional training.

**2.09** There are six appendices included at the end of the manual. They contain detailed equipment specifications, a complete description of both DICOL and COMBASIC instruction sets, X25/X75 scenario and link test instructions, control graphics and code charts, and programming worksheets.

## CHAPTER 2

### UNPACKING AND INSPECTION

#### 1. GENERAL

**1.01** This chapter covers environmental considerations, unpacking and handling, inspection, safety precautions, cleaning, storage, and the installation of the ENCORE. Also included are the pin assignments for the RS-232C interface used to access the circuit under test via the IO port and external peripherals via the XIO port.

#### 2. ENVIRONMENTAL CONSIDERATIONS

**2.01** In most geographical locations, the normal heating and ventilating facilities for personnel will provide a favorable environment for operation of the instrument. All performance specifications are met when operating at temperatures within the range of +5 to 50 C at a relative humidity of 20 to 80%.

#### 3. UNPACKING AND HANDLING

**3.01** When shipped from the factory, the ENCORE and its accessories are packed in a single cardboard carton using foam packing to absorb shock and vibration. The shipping weight, dimensions, and contents of the standard configuration are given in Table 2-1. Upon receipt, unpack the carton and verify its contents.

**NOTE:** Original packing material should be preserved for use in repacking and shipping.

**TABLE 2-1**  
**PACKING DATA**

Shipping Weight	Dimensions	Contents	Qty
45 lbs.	13 in. x 23½ in. x 24 in.	<u>STANDARD</u>	
		ENCORE 200	1
		Program Disc, Levels 1, 2, and 3	1
		Program Disc, blank	1
		T-Box	1
		Auxiliary Interface Cable	1
		Power Cord	1
		Technical Manual	1

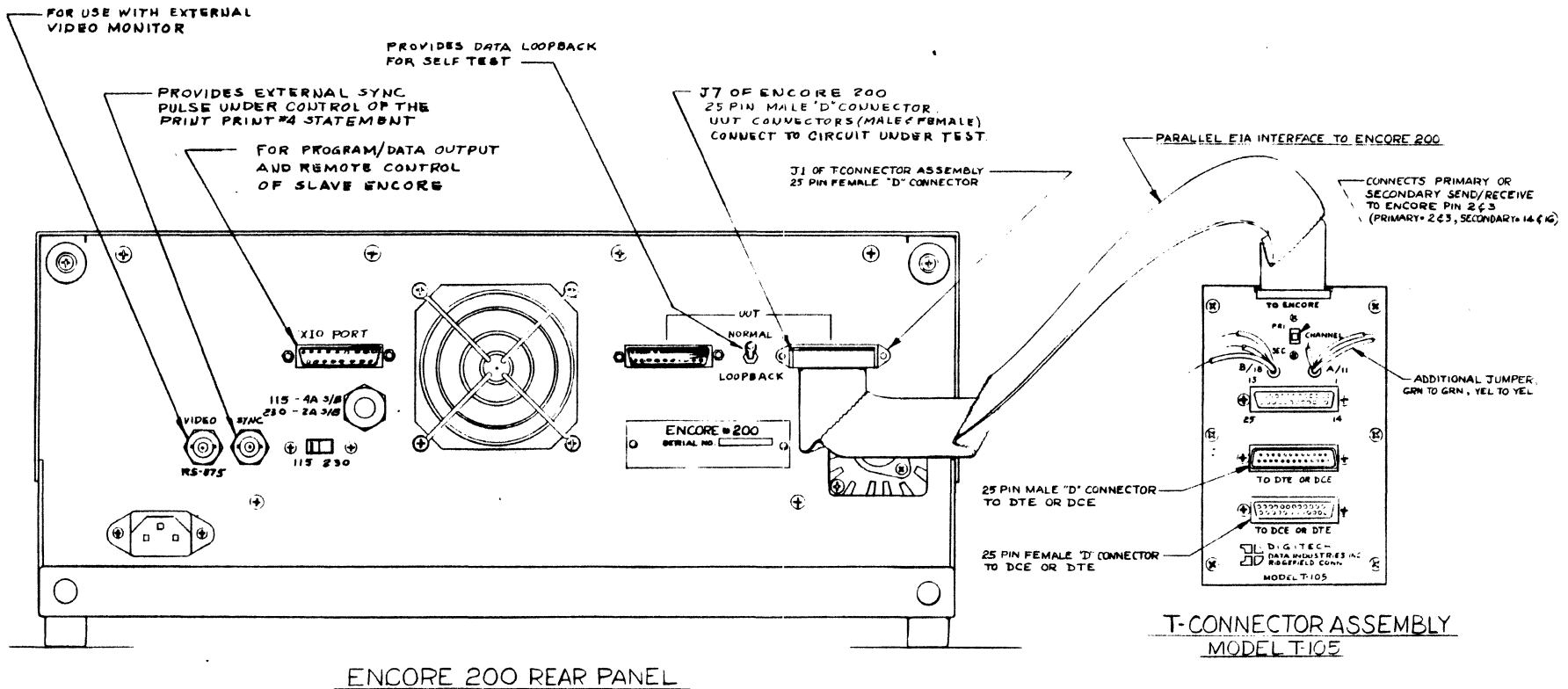
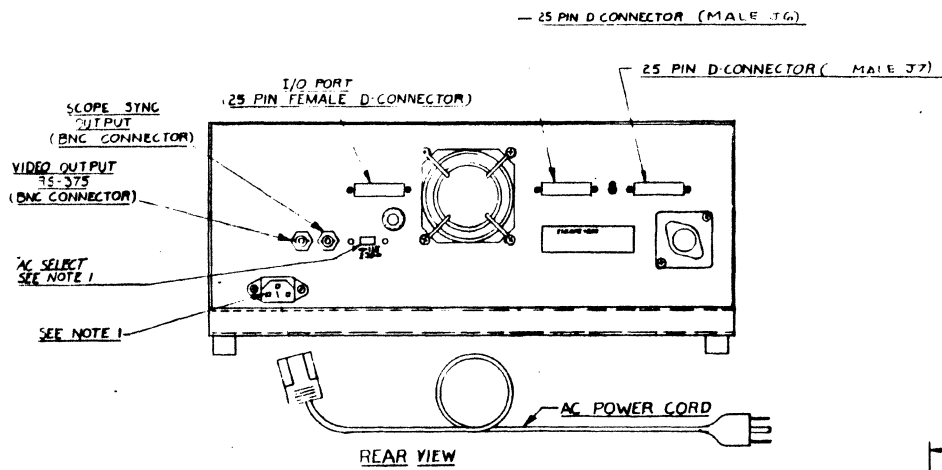


Fig. 2-1 ENCORE External Equipment Connections (D811-00487)



NOTES  
1- POWER REQUIREMENTS  
VOLTAGE - 95/130 OR 190/260 VAC SELECTABLE  
FREQUENCY - 47 Hz TO 440 Hz  
POWER - 100 WATTS MAXIMUM

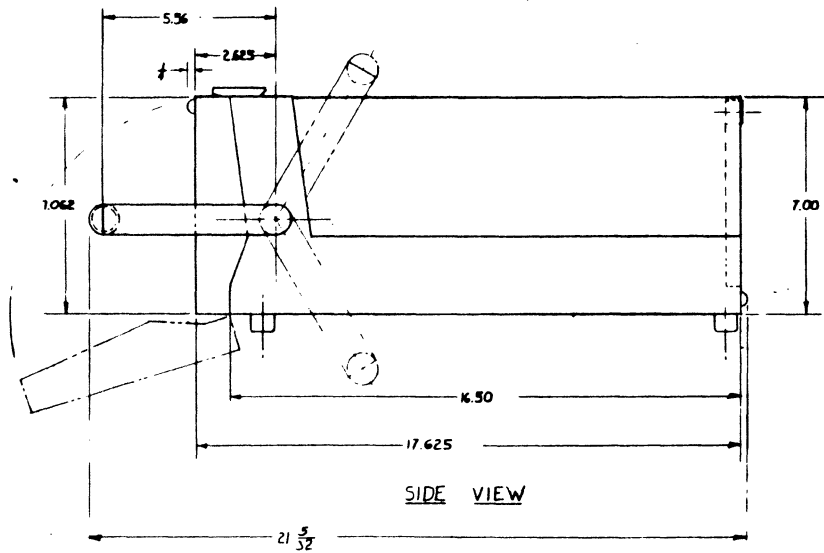
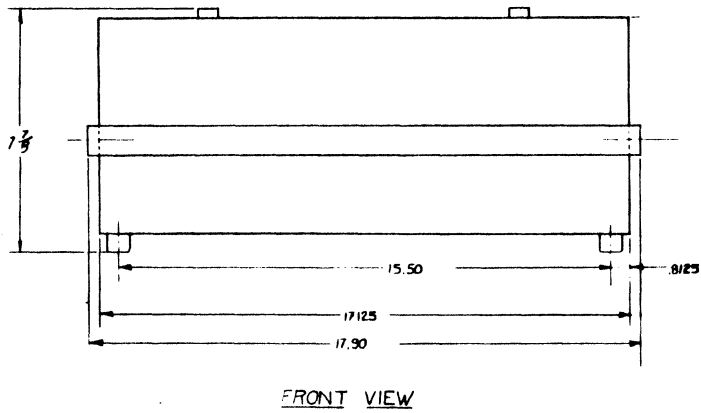


Fig. 2-2 Installation Data  
(D830-00117)

**3.02** If the instrument is to be shipped to DIGITECH for service or repair, contact DIGITECH for scheduling information. In any correspondence, identify this instrument using both the serial number and the product number of the ENCORE and the options included (see Chapter 3).

**3.03** Repack the instrument using the original packing material or wrap the instrument in heavy paper or plastic before placing it in the container. Place packing material all around the sides of the instrument. Seal the shipping container with strong tape or metal bands and mark it "DELICATE INSTRUMENT", etc.. See the Warranty on the last page of this manual for additional information.

#### **4. INSPECTION**

**4.01** The ENCORE was carefully inspected both mechanically and electrically before shipment. It should be physically free of marks or scratches and in perfect electrical order upon receipt. After unpacking, inspect the equipment for signs of damage incurred during shipment. If the instrument has been damaged, notify the carrier immediately.

#### **5. SAFETY PRECAUTIONS**

**5.01** Some of the voltages required for operation of the ENCORE may present a hazard to the operator if he fails to exercise reasonable care when handling the equipment. The unit should be properly grounded and any internal maintenance should be performed by qualified personnel.

#### **6. CLEANING**

**6.01** Cleaning of the ENCORE is usually required at least every four months. The outside front panel and other exposed surfaces are cleaned with mild soap and water. An air hose provides a satisfactory means of cleaning hard-to-reach areas.

#### **7. STORAGE**

**7.01** When stored, the ENCORE should be repacked in its original carton using the same packing material. The area should be cool and dry. Temperature should not exceed the limits given in Appendix A.

#### **8. INSTALLATION**

##### **A. Placement**

**8.01** Installation of the ENCORE is fairly straight forward. Since the unit is designed for desk top use, find a suitable location, make ac power and interface connections as detailed in Figures 2-1 and 2-2, and proceed to "power-up" the unit as described in the Initial Operating Procedure in Chapter 4.

**CAUTION:** Always remove the disc when turning power on or off.

##### **B. Power Requirements**

**8.02** The ENCORE is designed for worldwide installation. A rear panel voltage selecting switch eliminates the need for internal wiring changes when switching from a 95 to 130 or 190 to 260 Vac power source. A 4 amp S/B fuse (092-00154) or Littlefuse 3AG 313004 is used for 95-130 Vac operation and a 2 amp S/B fuse (092-00022) or Littlefuse 3AG 313002 for 190-260 Vac operation. The fuse is located on the rear panel shown in Figure 2-2.

## CHAPTER 3

### PRODUCT DESCRIPTION

#### 1. GENERAL

**1.01** This chapter provides the user with a physical description of the ENCORE, Figure 3-1, including its major components. It also discusses design philosophy and system architecture including an overall system block diagram. In addition, the standard equipment configuration is established and options are defined.

**1.02** Physical controls are detailed through the use of full page photographs which are keyed to an explanation of specific connectors, switches, etc..



Fig. 3-1 ENCORE 200

## **2. THREE LEVELS OF OPERATION**

**2.01** Level-1 is a disciplined test procedure with a directory of test programs that reside on disc. No programming is required. The operator simply enters a number, sets IO parameters, and executes the program by selecting it from one of the Level-1 Menus. These programs, sequenced into monitor and simulation functions, can be added to by the user or by DIGITECH at the user's request. The Level-1 Operating System is located on the program disc supplied with each ENCORE.

**2.02** Level-2 uses DICOL, a high order programming language designed for DIGITECH's PACER-103. This language is used for interface control and data manipulation and is widely accepted in the data communications industry as a simple, easy to use diagnostic language. All programs designed for use with the PACER are upward compatible with the ENCORE 200. The Level-2 Operating System is located on the same program disc as Level-1.

**2.03** Level-3 incorporates a form of BASIC (Beginners All purpose Symbolic Instruction Code) enhanced by DIGITECH to simplify its use and improve its relevancy to communications. This new language, hereafter referred to as COMBASIC (COMMunications BASIC), provides the user with a powerful and easy to use means of designing comprehensive and otherwise complicated programs. The Level-3 operating system is contained within the ENCORE's Read Only Memory (ROM).

## **3. DESCRIPTION**

**3.01** ENCORE is a rugged, portable, self-contained data communications diagnostic test set employing the latest techniques in microprocessor based integrated circuit design. The unit measures approximately 7 7/8 inches high, 18 1/4 inches wide, and 18 1/4 inches deep. It incorporates an image-enhanced CRT, a 5 1/4 inch floppy disc drive, and full ASCII keyboard.

**3.02** While operating as an intelligent Monitor or Simulator, ENCORE provides the operator with a view of real time data on an integral 7 inch CRT (Cathode Ray Tube). It analyzes communications problems and isolates troubles to specific network components by taking control of the RS-232C interface and simulating system hardware and software.

**3.03** While executing passive monitor programs, ENCORE allows the operator to view transmit or receive data and selectively load data in RAM (Random Access Memory). These types of programs are used to detect poll or select sequences and monitor terminal response while ignoring less significant transmissions. ENCORE will also detect the status of given interface leads and audibly alert the operator upon recognition of a specific event (high-low control lead, parity error, time out, etc.).

**3.04** When executing a program that actively simulates a network component, ENCORE may transmit data and clock signals while controlling the state of specified interface leads. It can also transmit and receive pseudorandom patterns for character error rate and block error rate testing; and it can calculate, transmit, and compare block or frame check sequences.

### **A. Primary Display**

**3.05** The Primary Display is a 7 inch CRT using an 8 x 16 dot matrix. This matrix permits display of 7 or 14 lines of 32 or 64 characters and provides display of control character mnemonics and HEX pairs while maintaining character registration. Displayed data may be converted to a HEX or binary representation through the use of COMBASIC subroutines. Data is normally displayed using a 128 character ANSI/ASCII repertoire with two character mnemonics for all control symbols. Send and receive data are displayed as normal and reverse video, respectively. Normal video appears as white characters on a black background. Reverse video is the opposite; black characters on a white background. Half intensity

subcharacters are used to identify such characteristics of the data as parity errors, block check characters, flag bytes, etc.. The cursor, a blinking white rectangle, indicates the current writing location on the CRT in all operational modes. The maximum capacity of the CRT memory including half-intensity subcharacters (attributes) is 4 096 bytes. Information displayed on the CRT originates at the keyboard, RS-232 interface (DCE or DTE data), and capture or program memory.

### **B. Real Time Display**

**3.06** The Real Time Display is an interface monitor using LED's (Light Emitting Diodes) to indicate the status of each control, data, and timing lead. The display is located to the right of the CRT, above the disc drive. Red LED's indicate the on (illuminated) or off (not illuminated) state of control leads. Tristate LED's are used to indicate the mark (green), space (red), or quiescent (not illuminated) state of data and timing leads.

### **C. Disc Drive**

**3.07** The Disc Drive uses a 5¼ inch mini diskette as the permanent storage media and is located below the Real Time Display. The ENCORE can record data and the status of six EIA control leads (circuits) or the attribute byte. The same type of information, if stored in RAM, may be transferred to the disc for a permanent record. The disc may also be used to store programs and to load these programs into the ENCORE's own nonvolatile storage buffer (program memory). The Disc Drive records data and status bytes at speeds up to 9 600 bps operating FDUX. The data storage capacity of the disc is at least 240K bytes (8-bit).

### **D. Memories**

**3.08** The ENCORE employs up to 128K of memory which is considered either system or bulk memory. Although the precise allocation of memory is quite complicated, it can be assumed for this discussion that the System and Bulk memories are each comprised of 32K of ROM (Read Only Memory) and 32K of RAM (Random Access Memory). The System ROM contains the software required for operation of the main processor including language ROM's, system configuration ROM's, etc.. System RAM includes up to 26 624 bytes of memory which are dynamically allocated to the source, object, and capture memories. The amount of memory allocated to each of these is established by the operator while in the Memory Allocation Mode as discussed in Chapter 7.

### **E. Keyboard**

**3.09** The ENCORE keyboard is a 76 array with a 56 key trimode touch typing area (main array), a 15 key control cluster to the right of the main array, and a 5 key special functions cluster to the left of the main array. The keyboard employs "n" key rollover protection and multiple key inhibit. In addition, up to 26 keys may be assigned to User Defined Functions greatly simplifying repetitive entry of large or multiple word commands.

### **F. Interface**

**3.10** The ENCORE is supplied with an RS-232/V.24 Interface for use with the Unit Under Test (UUT). Separate interface parameters may be established for both transmit and receive functions. This allows the user, for example, to transmit data at one speed while receiving it at another. Other standard interface provisions include scope SYNC and VIDEO (RS-375) outputs and an External IO port, which are detailed later in this chapter.



3.11 The RS-232 Interface connects ENCORE to the interchange circuit using one or both of the 25-pin D connectors (one male, one female). A pull to unlock miniature toggle switch allows the operator to loopback data for self-test purposes. Table 3-1 defines each pin of the RS-232 interchange circuits.

**TABLE 3-1**  
**RS-232 INTERFACE PIN ASSIGNMENT**

RS-232 CIRCUIT				ENCORE CIRCUIT BY MODE (3)				
PIN NUMBER	EIA DESIG	CCITT CKT	DESCRIPTION	SIG SOURCE	LED DISPLAY	INPUT	CONTROL DTE	OUTPUT DCE
1	AA	101	Protective ground	-	(1)(5)	(1)(5)	(1)(5)	(1)(5)
2	BA	103	Transmitted data (10)	DTE	Red/green	BR	T	BR
3	BB	104	Received data (10)	DCE	Red/Green	BR(2)	BR(2)	T
4	CA	105	Request to Send, RTS (8)(4)	DTE	Red	BR	T	BR
5	CB	106	Clear to Send, CTS (8)(9)	DCE	Red	BR	BR	T
6	CC	107	Data Set Ready, DSR (8)(9)	DCE	Red	-	-	T
7	AB	102	Signal ground	-	Red(1)(5)	(1)(5)	(1)(5)	(1)(5)
8	CF	109	Receive Line Signal Detect, RLSD (9)	DCE	Red(2)	BR(2)	BR(2)	T
9	-	-	Reserved for data set testing	-	Red(5)	-	-	-
10	-	-	Reserved for data set testing	-	Red(5)	-	-	-
11/A	-	-	Unassigned (8)	-	Red(4)	BR/T	BR/T	BR/T
12	SCF	122	Sec. receive line signal detect	DCE	Red(5)	-	-	-
13	SCB	121	Sec. clear to send, SCTS	DCE	Red(5)	-	-	-
14	SBA	118	Sec. transmitted data	DTE	Red(5)	-	-	-
15	DB	114	Transmit clock (11)	DCE	Red/green	BR	BR	T
16	SBB	119	Sec. received data (11)	DCE	Red(5)	-	-	-
17	DD	115	Receive clock	DCE	Red/green	BR	BR	T
18/B	-	-	Unassigned (8)	-	Red(4)	BR/T	BR/T	BR/T
19	SCA	120	Sec. request to send, SRTS	DTE	Red(5)	-	-	-
20	CD	108.2	Data terminal ready, DTR (8)(9)	DTE	Red	-	T	-
21	CG	110	Signal quality detect, SQ	DCE	Red(5)	-	-	-
22	CE	125	Ring indicator (8)(9)	DCE	Red	BR	BR	T
23	CH/CI	111/112	Data signal rate select	DTE	Red(5)	-	-	-
24	DA	113	External transmit clock (11)	DTE	Red/green(5)	-	-	-
25	-	-	Unassigned	-	Red(5)	-	-	-

**NOTES:**

- (1) Pin 1 is chassis/frame and is not connected to signal ground, pin 7.
- (2) Pin 8 (CF) must be on (+) to permit data input on pin 8.
- (3) BR = Bridging Receiver.  
T = Transmitter.
- (4) A and B are control leads or receive leads which may be selected using the external T-Connector.
- (5) Not Controlled.
- (6) 1 = (-V) - Marking = OFF.  
0 = (+V) = Spacing = ON.  
DTE = Data Terminal Equipment.  
DCE = Data Communications Equipment.
- (7) LED color definition:  
Green = Mark, no greater than -3V.  
Red = Space, no greater than +3V.  
Extinguished = No signal greater than +3V.  
Red and green = Transitions.
- (8) The status of pins 4, 5, 8, 11, 18, and 22 are recorded in the status byte.
- (9) Pins 4 and 2 are controlled using the COMBASIC STATE TERM instruction. Pins 5, 6, 8, and 22 are controlled using STATE MODEM.
- (10) Data is output via pin 2 using STATE TERM (clocked by pin 15) and via pin 3 using STATE MODEM (clocked by pin 17).
- (11) When simulating a terminal (STATE TERM), and using internal clock, the ENCORE outputs timing via pin 24. When simulating a modem (STATE MODEM), and using internal clock, timing is output via pins 15 and 17.

4. DESIGN PHILOSOPHY

4.01 ENCORE's design is based on the concept of distributed processing. Although the concept is not new, it has just recently become available for use with microprocessor based systems. This is due to the availability of programmable peripherals that relieve the microprocessor of its time consuming bookkeeping, timing, and IO tasks. Within ENCORE, a single main processor coordinates the activities of the entire system. The main processor has the capability of communicating with all system peripherals, but by no means controls every step of operation. Distributed processing relieves the main processor of this burden through the use of other microprocessors, dedicated function LSI circuits, and simple state machines.

5. ARCHITECTURE

A. Control Signal Attributes

5.01 The main processor uses three different methods of control and communications between itself and system peripherals as shown in Figure 3-2. The basic attributes of the different forms are discussed in the following paragraphs.

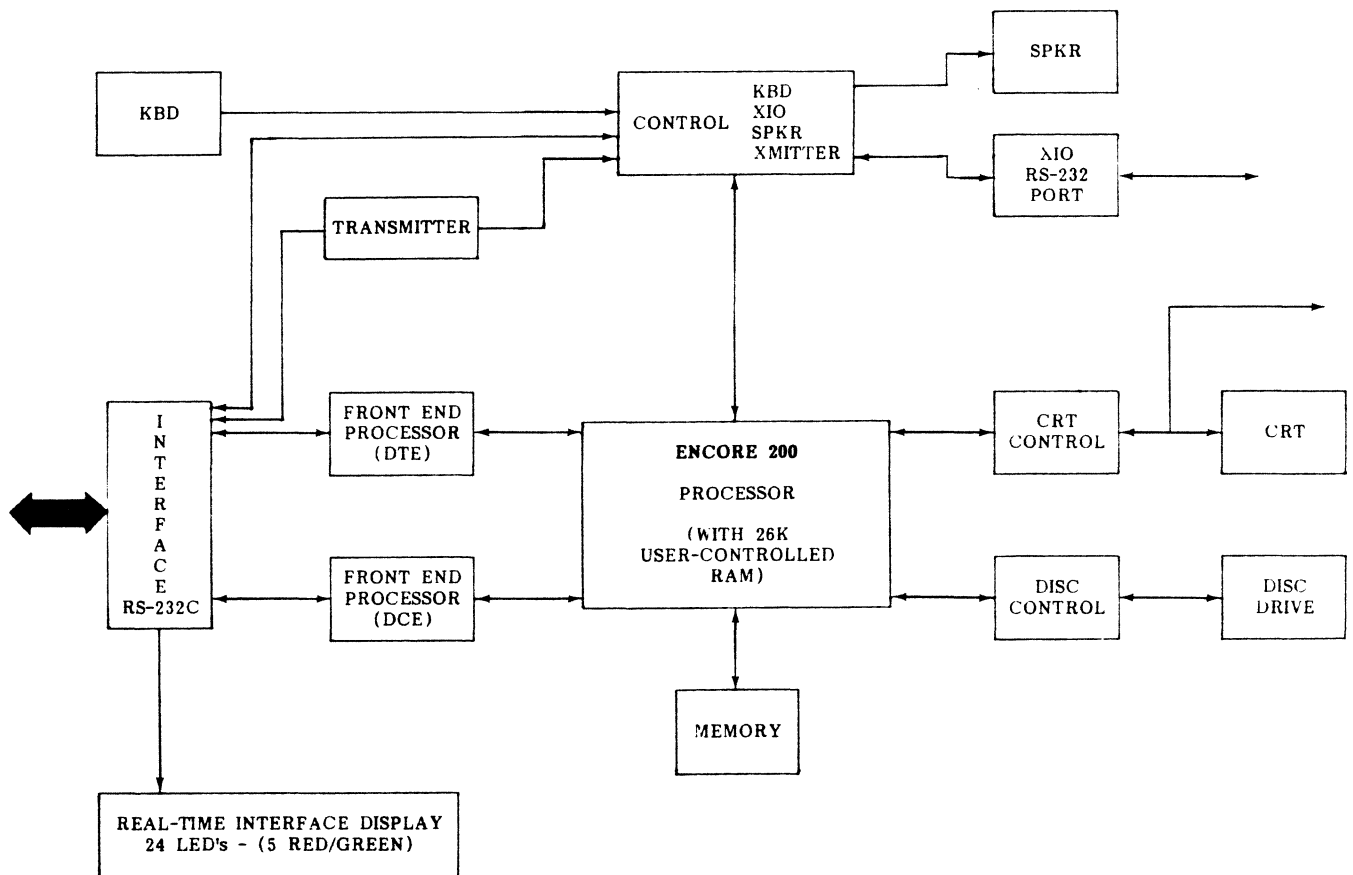


Fig. 3-2 ENCORE Block Diagram

**5.02 IO Control.** This method is totally under control of the microprocessor's set of program instructions. Therefore, control commands, and/or data will be written or read at a particular port at the discretion of the microprocessor's software. Usually this technique is used to transfer commands, status, or initialization data between the main processor and system peripherals. This is the main function of the IO because each IO demands main processor CPU dedication which is 10 system clock intervals per IO instruction. Therefore, transferring large amounts of data between the CPU and a system component would take even more clock states, since at a minimum:

- An instruction must be executed to get data from memory.
- An IO instruction must be executed.
- An instruction must be executed to increment the memory pointer.
- An instruction must be executed to determine if all data bytes have been transferred.

**5.03** A better and faster method available is DMA (Direct Memory Access). Another drawback of using IO is having to interrogate the status of system peripherals. If we are only interested in the status of the system peripheral because we wish to have activity with it, then a software loop must be made when using IO interrogation. This is inefficient because it demands that the microprocessor execute this IO interrogation over and over until the correct status is obtained. A better method, interrupt, would allow the microprocessor to continue its activity and be signalled asynchronously with software execution when a particular system peripheral has a particular status.

**5.04 DMA Control.** This is an efficient method requiring only four system clock states to transfer data from the main processor memory to a port or vice versa. When an ENCORE system component which is assigned a DMA channel asserts a DMA request line to the main processor, the following sequences will occur:

- The main processor software execution will stop.
- The DMA controller inside the main processor will become the system bus master.
- The DMA controller will acknowledge this via a DACK line to the system component.
- Data transfer will take place between the main processor and the system peripherals in either direction. The number of data transfers and the location of memory has been initialized by software prior to this sequence. Once the number of DMA transfers has been finished the main processor CPU will again become master.

**5.05** In order for the DMA controller to transfer data upon a system peripheral request as described, the only work the CPU has to do is to program the memory location in the DMA controller and the number of bytes to transfer. The four system clock cycles of DMA transfer are much shorter than the IO technique. Not only does this save time for a programmed data transfer, but it permits a system peripheral to initiate data transfers, either because they have data or because they need data, without any demand upon the CPU. In effect, this data transfer was transparent to the CPU software since it stopped executing instructions as soon as DMA became the master. A slightly different operation has been designed in the ENCORE which permits the main processor CPU to execute an instruction every 10 microseconds during DMA. This will be discussed in more detail later.

**5.06 INT (Interrupt) Control.** When a system peripheral needs to be serviced by the main processor, it sends a unique interrupt signal to the main processor. In this technique, the main processor stops what it is doing, saves pertinent useful data, and then starts executing software which was pointed to by the specific interrupt. This permits the main processor to execute its software and only service the system peripheral when the system component requires it.

### **B. Control Signal Assignment**

**5.07** Assignment of the three different methods of control and communication between the main processor and system peripherals as shown in Figure 3-2 is discussed in the following paragraphs.

**5.08 Disc Controller.** Because large amounts of data are transferred to and from the disc, it is assigned a DMA channel. When the data transfer is complete, the DMA controller in the main processor signals the disc controller, which then interrupts the main processor. This ensures that the main processor initiates disc data transfer and then forgets about it until an interrupt occurs. One IO port is assigned to the disc to be used by the DMA channel while other ports are assigned for command and status.

**5.09 CRT Controller.** Again, because large amounts of data are transferred to the CRT controller for subsequent display on the CRT, it is assigned a DMA channel in the main processor. In fact, its DMA channel also has the capability of autoloading. This means that the main processor can identify the memory location and the number of transferred bytes for this channel only once and forget about it, because the DMA will autoloading the base address and bytes count when data transfer is completed. The only need to ever change this is for a screen format change. Again an IO port is assigned for the DMA channel and other ports are assigned to permit initialization of the CRT controller.

**5.10 Keyboard, External RS-232, Speaker, and Transmitter.** These functions require all three methods of control/communications.

- **Keyboard:** Because this is the only operator interface, it is assigned one interrupt for asynchronous system intervention and an IO port for one byte data transfer per keystroke.
- **XIO (External IO Port):** Because the External IO Port transfers data on a byte by byte basis and the maximum speed is only 19.2K bit/sec, i.e., 1 300 system clocks between byte transfer, an interrupt driven IO scheme is appropriate. Two interrupts are assigned, one for receive data and one for transmit data. Two ports are assigned, one for initialization and status, while the other is for data bytes.
- **Speaker:** The speaker is driven by one command; therefore one IO port is assigned to it.
- **Transmitter:** This circuit is the main data transmitter used by ENCORE when emulating data modems or terminals. Since data transfer could occur at 100K bit/sec, a byte is required to be transferred every 250 system clocks. In order to maximize the efficiency of CPU operation, a DMA channel is assigned to the transmitter. Again, when the total byte transfer is complete, the main processor signals the transmitter with a signal (terminal count) which then turns around and interrupts the main processor. This sequence permits the main processor to initialize data transfer to the transmitter and then proceed with other activities until interrupted by the transmitter. Ports are assigned for the DMA channel transmitter commands and status.

- **EIA Control:** This circuitry requires IO ports to set or sense the status of EIA leads. Eight interrupts are also assigned to specific EIA leads so that asynchronous events can be sensed.

**5.11 Front End 2/3.** These system peripherals are the primary receivers for EIA leads 2 and 3. The main method of communications is via interrupts and IO. The front ends interrupt the main processor for every character received. The main processor then acquires this byte at the specific port assigned to the particular front end.

**5.12 Memory.** The method of data transfer is via interrupt and IO. The main processor initializes activity to a particular port associated with the specified memory. Then, the memory interrupts the main processor when the activity is completed, after reading or writing data. This technique permits slow, nonvolatile memories to be utilized in the memory.

**6. STANDARD CONFIGURATION**

**6.01** Equipment supplied with the standard ENCORE configuration is listed in Table 3-2 and indexed to Figure 3-3.

**TABLE 3-2  
EQUIPMENT SUPPLIED**

<b>FIGURE AND INDEX NO.</b>	<b>ITEM</b>	<b>QTY</b>	<b>DIGITECH PART NO.</b>
3-3-1	Power Cord	1	096-00008
3-3-2	Auxiliary Interface Cable	1	948-00081
3-3-3	T-Box	1	940-00016
3-3-4	ENCORE 200	1	702-00030
3-3-5	Program Disc, Level-1, Level-2, and Utilities	1	805-00016
	Program Disc, Blank	1	035-00117
3-3-6	Technical Manual	1	810-00184

STANDARD CONFIGURATION

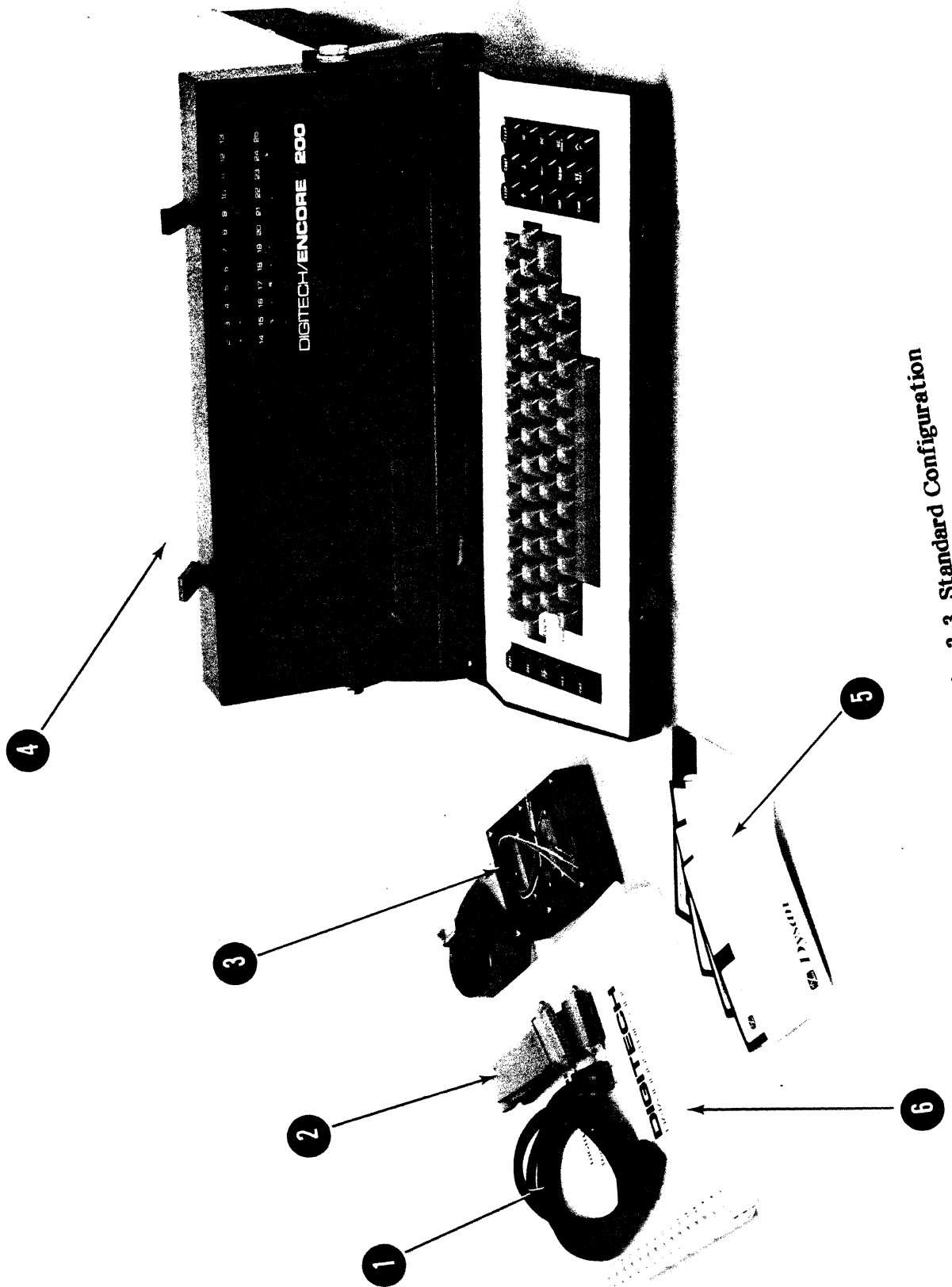


Fig. 3-3 Standard Configuration

**7. OPTIONS**

**7.01** Table 3-3 describes each of the options available with ENCORE. In referring to these options, use the product number given in the table.

**TABLE 3-3**  
**OPTIONS**

<b>PRODUCT NO.</b>	<b>DESCRIPTION</b>
E250-01	IPARS Language
E250-02	XS-3 Language
E250-03	BAUDOT Language
E250-04	SBT Language
E250-05	FIELDATA Language
E250-06	BIDIC Language
E250-07	RMS Language
E250-08	RMS-2 Language
E251	Additional Technical Manual
E252	X25 Monitor and Interactive Package
E254	100 Volt ac

## CONTROLS, DISPLAYS, AND CONNECTORS

### 8. CONTROLS, DISPLAYS, AND CONNECTORS

**8.01** Tables 3-4 and 3-5 describe the function of all ENCORE controls, displays, and connectors. These include the three keyboard arrays and both Primary and Real Time Displays as shown in Figure 3-4. Rear panel connectors are shown in Figure 3-5.

**TABLE 3-4**  
**CONTROLS, DISPLAYS, AND CONNECTORS**

<b>FIGURE AND INDEX NO.</b>	<b>NAME</b>	<b>FUNCTION</b>
<b>***** FRONT *****</b>		
3-4-1	Primary Display	The Primary Display is a 7 or 14 line by 32 or 64 character, seven inch CRT using an 8 x 16 dot matrix. The display presents the operator with a view of on-line data, captured data, programs, and operating mode formats.
3-4-2	Real Time Display	The Real Time Display is a 24 LED cluster to the right of the CRT. The display monitors the state of all the RS-232 leads except frame ground (pin 1) and signal ground (pin 7), although a LED exists for pin 7, it is not operative. When illuminated, a red LED indicates a positive, ON, condition for all control leads. Tristate LED's are used to indicate the mark (green), space (red), or quiescent (not illuminated) state of leads 2, 3, 15, 17, and 24.
3-4-3	Power On/Off	Level switch, applies primary ac power to the unit.
3-4-4	Disc Drive	The disc drive is a data recorder using a 5 $\frac{1}{4}$ inch mini diskette as the storage media. Each disc provides an additional 240K bytes of memory which may be used to store programs or real-time data.
3-4-5	Access Door	Protects disc drive mechanism from dust. Door should be closed when drive is not in use.
3-4-6	Door Release	Pushbutton, opens door to drive and ejects disc.
3-4-7	Activity LED	Illuminates whenever drive motor is on.



**TABLE 3-4**  
**CONTROLS, DISPLAYS, AND CONNECTORS (Cont'd)**

FIGURE AND INDEX NO.	NAME	FUNCTION
<b>***** FRONT *****</b>		
3-4-8	Control Cluster	The Control Cluster consists of 15 auxiliary keys that provide convenient control of mode selection and edit capabilities.
3-4-9	Main Array	The Main Array consists of 56 touch typing keys employing "n" key rollover protection and multiple key inhibit with clearly defined control legends.
3-4-10	Special Function Cluster	The Special Function Cluster consists of five keys. Four deal with special display functions and one, RWD, sets the disc read/write head to the beginning of the first disc record.



CONTROLS, DISPLAYS, AND CONNECTORS

Fig. 3-4 ENCORE 200 Front View

**TABLE 3-5**  
**CONTROLS, DISPLAYS, AND CONNECTORS**

FIGURE AND INDEX NO.	NAME	FUNCTION
<b>***** REAR *****</b>		
3-5-1	XIO Port	Auxiliary External IO Port (male) provided for serial down line load, etc.. Data output via pin 3 is asynchronous ASCII with a 2.0 unit stop interval. RS-232 voltage levels and appropriate 25-pin "D" connector are utilized.
3-5-2	UUT Connector (female)	Items 2 and 3 are parallel wired twenty-five pin "D" connectors used for interface between the ENCORE and the circuit under test. The status of the interface leads in the active and passive modes is covered in Table 3-1.
3-5-3	UUT Connector (male)	
3-5-4	Normal/Loopback	Two position locking level toggle switch (pull to unlock) provides normal operation or internal loop-back of data for self-test purposes.
3-5-5	Line Fuse	Protects the ac line from overload in the event of an equipment malfunction.  4 amp S/B at 95-130 Vac 2 amp S/B at 190-260 Vac
3-5-6	Voltage Selecting Switch	Provides for operation at 95-130 or 190-260 Vac. , 47-440 Hz.
3-5-7	SYNC	BNC connector provides an output port for external oscilloscope synchronization under programmable control.
3-5-8	RS-375 Video	BNC connector provides RS-375 composite video output.
3-5-9	AC Line Connector	Provides for ac input via removable line cord.

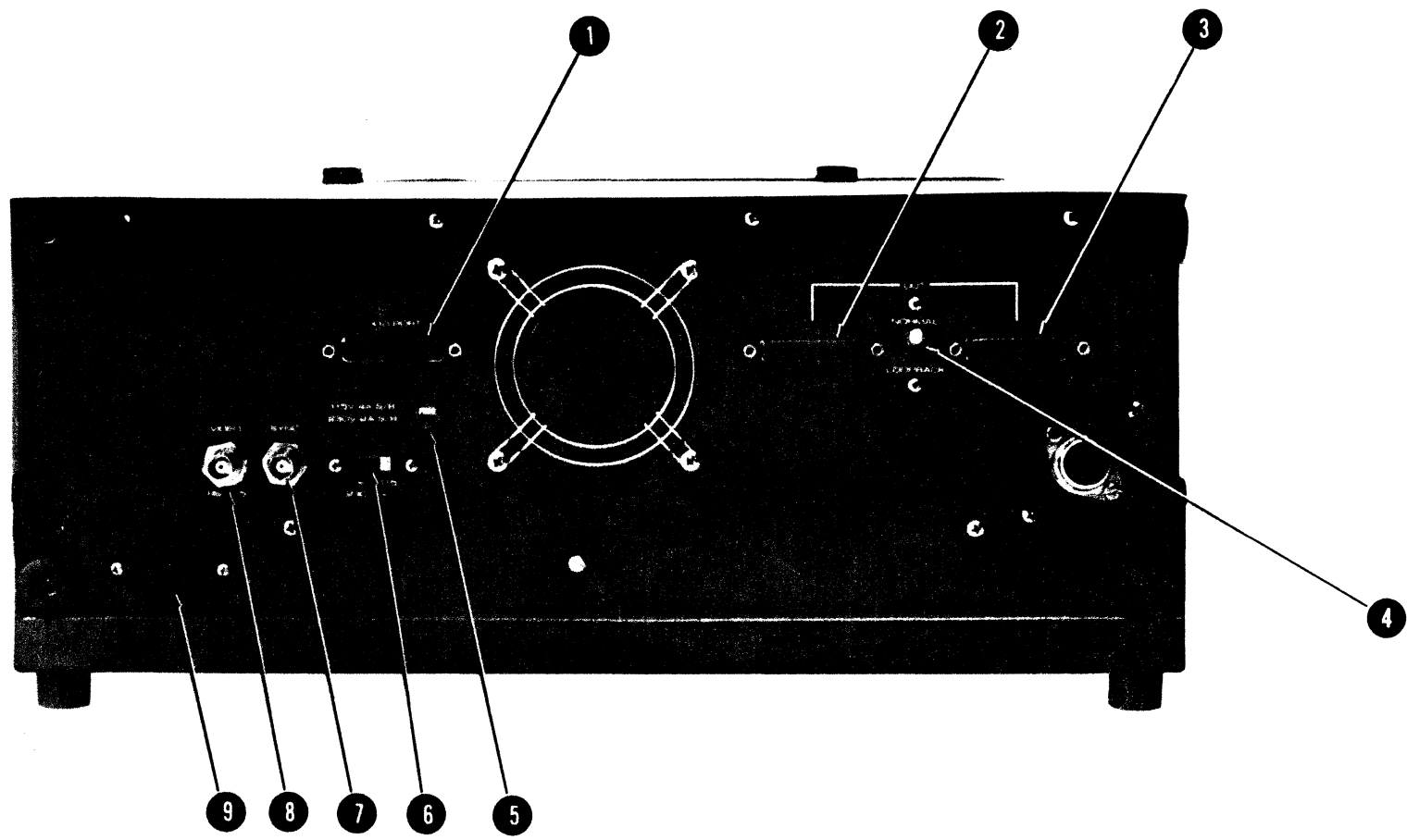


Fig. 3-5 ENCORE 200 Rear View

## 9. CONTROL CLUSTER AND SPECIAL FUNCTION KEYS

9.01 The operation of the system at any level requires some familiarity with the control cluster and special function keys along with the simple keystrokes that combine to form typewriter commands.

### A. CONTROL CLUSTER KEYS

STOP

This keystroke is only functional when monitoring data and only if a STATE LOCK has not been executed during the monitor program. Once STATE LOCK has been executed, the user must include some alternate means of program escape. When this key is depressed, the CRT display will halt (while the program continues) until the **CONT** key is depressed or until a screen position instruction is encountered in the program. The **CONT**, **CLEAR**, and **CMD** keys remain functional in this mode.

CONT

During program execution, this key is used to cancel the **STOP** function if a STATE LOCK has not been previously executed. If STATE LOCK has been executed, the user must include some alternate means of program escape. It is also used during the IO mode to permit entry of separate parameters for each front end.

CLEAR

During program execution this key is used to momentarily clear the CRT display and cancel the **STOP** function if a STATE LOCK has not been previously executed. If STATE LOCK has been executed, the user must include some alternate means of program escape. In the EDIT Mode, this key clears the current program line as noted by the cursor position.

↑ ↓ → ←

These keys are used to position the cursor. Display scroll is accomplished using the **↑** or **↓** keys alone or in combination with the **REPT** key.

IC

This is the "Insert Character" key. It is used in conjunction with the **←** and **→** keys to insert a character before the position of the cursor.

DC

This is the "Delete Character" Key. It is used in conjunction with the **←** and **→** keys to insert a character indicated by the position of the cursor.

UDF








A User Defined Function is executed when this key is depressed and then followed by an entry from the main array (A-Z). Up to 26 different functions may be assigned. These functions are defined by the user in the ASGN (assign) Mode.

BRDG








This keystroke is only functional when a program is not being executed. When the key is depressed, the ENCORE immediately assumes a nonintrusive state at the interface bridging all normally terminated interchange leads.

**NOTE:** If the STATE BRDG instruction is not used in the program to disconnect the ENCORE from the circuit under test, depress the **BRDG** key.

## CONTROL CLUSTER AND SPECIAL FUNCTION KEYS

-  Depressing this key normally returns the ENCORE to the Command Mode.
-  This is the Variable Entry Status key. It functions during the EDIT, MENU, TIME, XIO, IO, and ASGN modes and instructs the ENCORE to display the status of all four buffer memories and the IO parameters. Depressing the  again places the ENCORE in a memory modification mode (Variable Entry) where the operator may change the memory allocation for the object, source, and capture buffers.
-  This key is used for terminating the input of string variables during program execution and can be used in place of  while in the EDIT Mode to avoid placing carriage returns in a string constant.
-  This key provides keyboard entry of upper and lower case characters. It is normally used in the EDIT Mode (see top line of EDIT Mode Display) and permits conventional use of the keyboard where the  key must be depressed to enter upper case characters. Normal operation (upper case only) is similar to that of the Teletype Corporation's Model 33 terminal. Depressing this key a second time restores the ENCORE to the upper case only mode.

### B. SPECIAL FUNCTION KEYS

-  This key is used in conjunction with any of the other keys on the keyboard for multiple key entry. This key and the key to be repeated must be depressed and held down to repeat.
-  This key simply enters the ASCII delete symbol.
-  This key enters case shift symbols for languages requiring upper and lower case entries. It is used in conjunction with the  key to enter the upper case symbols and by itself to enter the lower case symbols.
-  This key is used during the IO mode to convert SYNC sequence entries to their HEX equivalents. The  key must be depressed immediately before the HEX entries. If depressed again, HEX entries will be converted to ASCII and the SYNC sequence will not be the desired HEX character.
-  Whenever this key is depressed, the disc automatically returns to the beginning of the first record.

9.02 Table 3-6 lists all control cluster and special function keys and indicates the operating mode in which they are active.

TABLE 3-6

USE OF CONTROL CLUSTER AND SPECIAL FUNCTION KEYS

	MENU	TIME	XIO	IO	ASGN	EDIT	CMD	RUN
<b>CONTROL CLUSTER KEYS</b>								
STOP								X
CONT								X
CLEAN						X		X
↑	X	X	X	X	X	X	X	X
↓	X	X	X	X	X	X	X	X
→	X	X	X	X	X	X	X	X
←	X	X	X	X	X	X	X	X
IC					X	X	X	X
DC					X	X	X	X
UDF	X		X	X		X	X	X
BRDG	X	X	X	X	X	X	X	X
CMD	X	X	X	X	X	X	X	X
VE STAT	X	X	X	X	X	X	X	X
^ P						X		X
UC ONLY						X		X
<b>SPECIAL FUNCTION KEYS</b>								
REPT	X	X	X	X	X	X		
DEL				X		X		X
IC TC								X
MEZ				X				X
RWD	X	X	X	X	X	X	X	X

## **T-CONNECTOR, AUDIO OUTPUT**

### **10. T-CONNECTOR**

**10.01** A T-Connector, provided for use with the RS-232 Interface, uses two jumpers that allow the operator to connect pins 11 (A) and 18 (B) of ENCORE to any of the 25 interchange circuits. A slide switch is provided to connect ENCORE to the secondary send and receive circuits (pins 14 and 16, respectively) in lieu of the normal primary connection (pins 2 and 3, respectively). Two additional jumpers are provided for the user's convenience.

**10.02** An auxiliary interface cable (approximately three feet long) is supplied and can be used with either the RS-232 Interface or the T-Connector.

### **11. AUDIO OUTPUT**

**11.01** The ENCORE is provided with a  $\frac{1}{4}$  watt audio output to a 2 inch speaker located under the keyboard. An audible output at a frequency of 1000 Hz is provided for programmed alarm. It may be turned on or off to indicate receipt of a parity error, NAK, or any other user defined event. In addition, an audible click provides instant confirmation of every typed entry.





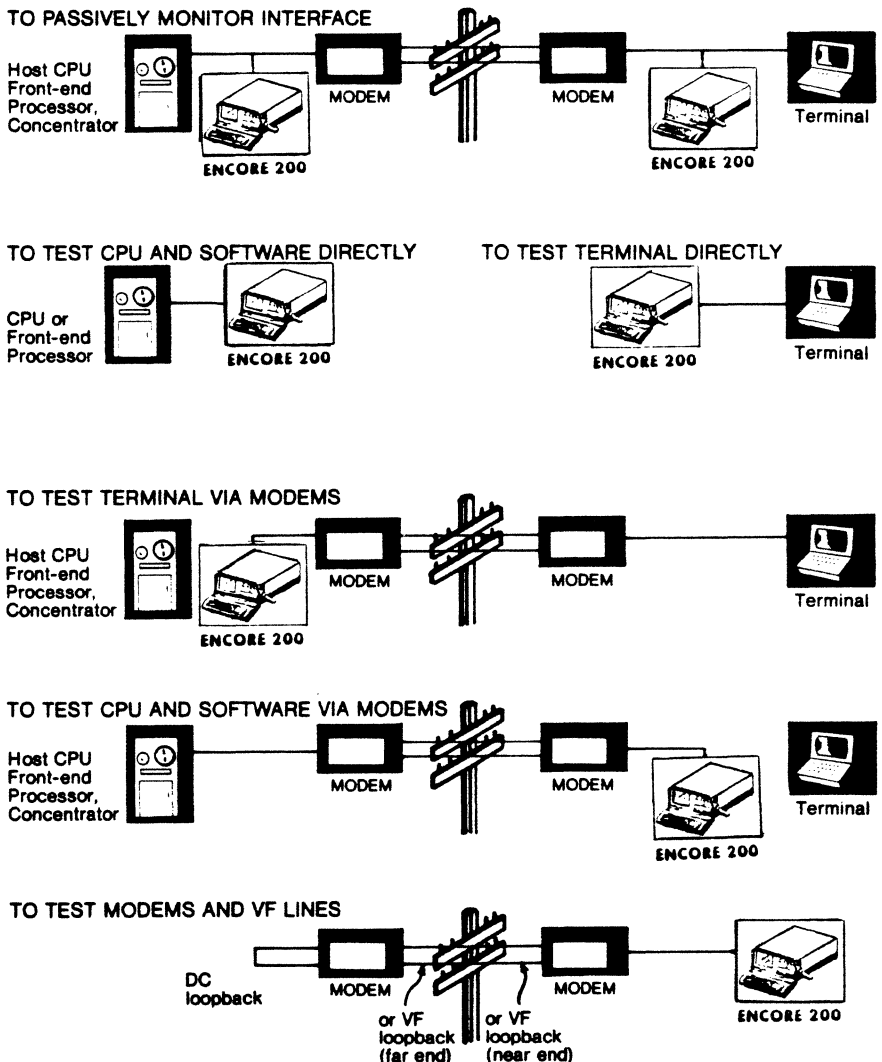
**CHAPTER 4**  
**INITIAL SET-UP AND OPERATION**

**1. GENERAL**

**1.01** In addition to the power-up procedure, this chapter includes examples of operation as a monitor in Level-1, Level-2, and Level-3. The synchronous monitor was chosen for all three levels of operation because it is one of the most commonly used features of the system. These examples will serve to familiarize you with the ENCORE, but by no means do they represent the full capabilities of the unit. These capabilities can best be realized by carefully reading the appropriate chapters of the manual and operating the unit. Select items of entry from the various menus to see what actually happens. Experiment with difficult commands and develop your own techniques for diagnosing problems. The more time you can spend with the ENCORE, the more you will be able to take advantage of its enormous power and flexibility.

**2. INITIAL SET-UP**

**2.01** The information in Chapter 2 regarding installation and the RS-232C interface should be read before attempting to connect the ENCORE to the circuit under test. With this information, the user must determine which of the configurations, shown in Figure 4-1, is required to perform the desired tests. Once the configuration is known, the user simply connects the ENCORE to the circuit under test via the rear panel UUT (Unit Under Test) connector.



**Fig. 4-1 Typical Operating Configuration**

POWER-UP

2.02 The procedures that follows are designed to take the user through the power-up sequence to the desired level of operation. It assumes that the user has read Chapters 5, 6, and 7 and has determined which level of operation is best suited to his current applications. A simple flowchart is included, Figure 4-2, showing the steps given in the procedure. Detailed information for operation in Level-1, Level-2, and Level-3 is supplied in Chapters 5, 6, and 7, respectively.

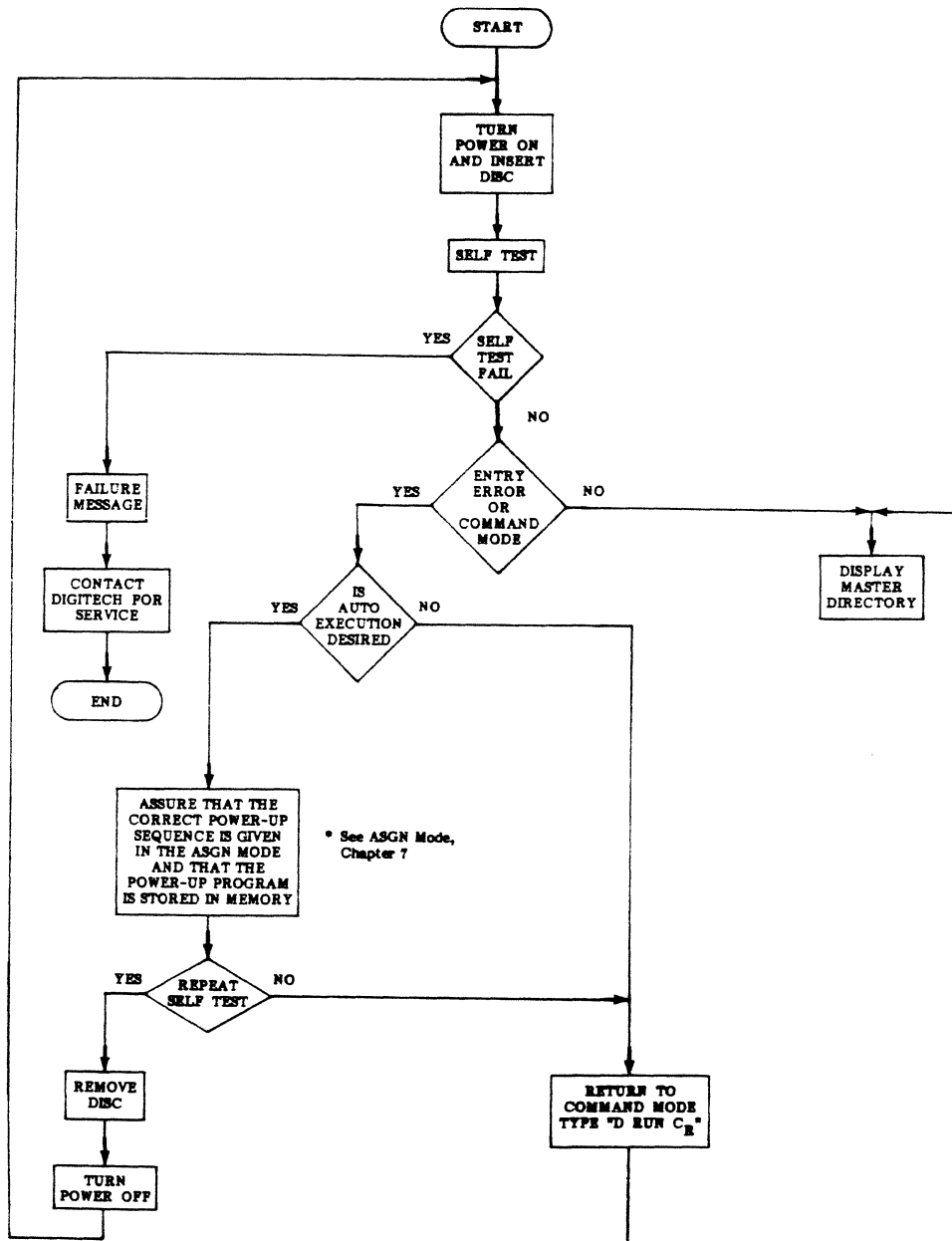


Fig. 4-2 Power-Up Flowchart

## INITIAL POWER-UP PROCEDURE

STEP	PROCEDURE
1.	Connect the ENCORE to the appropriate ac power source.
2.	Connect the ENCORE to the circuit under test.
3.	Assure that no disc is in the drive assembly.
4.	Set the ac power switch to the ON position and note that the ENCORE displays "SELF TEST IN PROGRESS".
5.	Insert the disc, write protected, tab to the left and toward the front (see Figure 4-3).

## WRITE PROTECT TAB



Fig. 4-3 Diskette Insertion

6. When the self test is successfully completed, the ENCORE will normally display the Master Directory as shown in Figure 4-4. If the ENCORE enters the Level-3 Command Mode or the display reads "ENTRY ERROR", type D RUN . This executes the disc stored program "RUN" which automatically allocates memory and eliminates the error message due to incorrect memory allocation in the power-up ASGN Mode.

INITIAL POWER-UP PROCEDURE (Cont'd)

STEP	PROCEDURE
7.	<p>If the display reads "ERROR #5", the program MD (Master Directory) must be loaded into memory from the diskette and stored prior to typing D RUN <input type="button" value="RETURN"/> . This is accomplished as follows:</p> <ul style="list-style-type: none"> <li>a) From the Level-3 Command Mode, type D EDIT MD <input type="button" value="RETURN"/> and note that MD appears in the EDIT Mode display.</li> <li>b) Type SAVE <input type="button" value="RETURN"/> .</li> <li>c) Strike the <input type="button" value="CMD"/> key and note that the ENCORE returns to the Level-3 Command Mode.</li> <li>d) Type D RUN <input type="button" value="RETURN"/> and note that the Master Directory is displayed.</li> </ul>

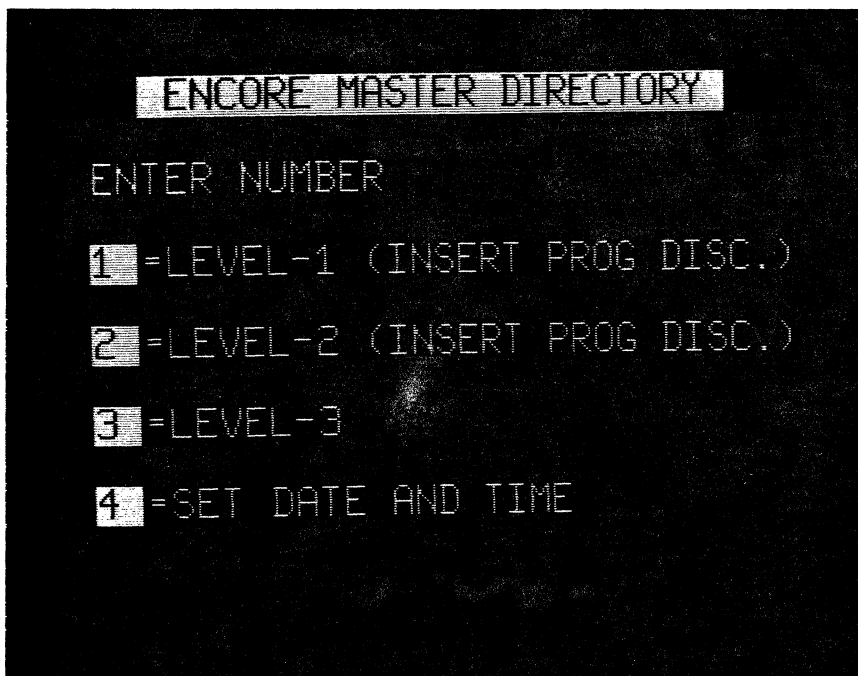


Fig. 4-4 Master Directory

8. From the Master Directory, select the desired level of operation.

This completes the Initial Operating Procedure

3. TYPICAL OPERATION IN LEVEL-1

3.01 For the purpose of this discussion, it is assumed that the user wishes to monitor synchronous data in ASCII and store that data on disc for later analysis.

**TYPICAL LEVEL-1 OPERATING PROCEDURE  
FOR  
MONITOR AND STORAGE**

STEP	PROCEDURE
1.	Connect the auxiliary interface cable between the ENCORE's rear panel UUT connector and the circuit under test. Assure that the NORMAL/LOOP-BACK switch is set to NORMAL.
2.	Power-up the unit as described in the procedure under paragraph 2.02.
3.	From the Master Directory, select Level-1 by striking the <input type="checkbox"/> key and note that the Level-1 Directory is displayed.
4.	From the Level-1 Directory, select the Synchronous Library by striking the <input type="checkbox"/> key and note display of the current IO parameters.
5.	Set IO parameters to agree with the parameters of the circuit under test. Refer to Chapter 7 for additional information.
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p style="text-align: center;">TYPICAL EXAMPLE:</p> <pre> SPEED -- 2400 SYNC --- S<sub>Y</sub> S<sub>Y</sub> NRZI --- NO MODE --- SYNC CLOCK -- EXT PARITY - ODD LANG --- ASCII                     </pre> </div> <div style="width: 45%;"> <p><b>NOTE:</b> These are the default parameters assumed on power-up.</p> </div> </div>
6.	Strike the <input type="checkbox"/> key when IO entries are completed and note display of the FRONT END SELECTIONS.
7.	Select the desired redundancy check by striking the appropriate key and note display of the Synchronous Library. Strike the <input type="checkbox"/> key for this example, because we have assumed the data to be in ASCII.
8.	Select MONITOR/TRAP from the library by striking the <input type="checkbox"/> key and note display of the MONITOR/TRAP Menu.
9.	Select MONITOR ALL by striking the <input type="checkbox"/> key and note display of the MONITOR ALL MENU which will now allow you to store data on disc.

**TYPICAL LEVEL-1 OPERATING PROCEDURE  
FOR  
MONITOR AND STORAGE (Cont'd)**

STEP	PROCEDURE
10.	For the purpose of this procedure, we will assume that data is to be stored on the program disc. Select DISC from the menu by striking the <span style="border: 1px solid black; padding: 0 2px;">I</span> key and note the user prompts to insert a non-write protected data disc.
11.	Eject the disc, remove the write protect tab, and re-insert the disc. Strike the <span style="border: 1px solid black; padding: 0 2px;">RETURN</span> key to continue and note that the synchronous data appearing at the interface is displayed on the screen and transferred to the disc.  <b>Note:</b> Pin 8 (RLSD) must be on to input data on pin 3.
12.	Use any of the keystrokes shown below to further enhance display of the monitored data or strike the <span style="border: 1px solid black; padding: 0 2px;">ESC</span> key to display of the MONITOR/TRAP Menu.

**Monitor/Trap Commands**

<span style="border: 1px solid black; padding: 2px;">" 2</span>	= P2 ONLY	<span style="border: 1px solid black; padding: 2px;">HT I</span>	= FLTR
<span style="border: 1px solid black; padding: 2px;"># 3</span>	= P3 ONLY	<span style="border: 1px solid black; padding: 2px;">IC</span>	= TIME STAMP
<span style="border: 1px solid black; padding: 2px;">SOH A</span>	= BOTH	<span style="border: 1px solid black; padding: 2px;">HEX</span>	= HEX
<span style="border: 1px solid black; padding: 2px;">DC4 T</span>	= TRAP RESTART	<span style="border: 1px solid black; padding: 2px;">ESC</span>	= EXIT
<span style="border: 1px solid black; padding: 2px;">BS H</span>	= HDUX	<span style="border: 1px solid black; padding: 2px;">ENQ E</span>	= EIA STATUS
<span style="border: 1px solid black; padding: 2px;">ACK F</span>	= FDUX	<span style="border: 1px solid black; padding: 2px;">STOP</span>	
<span style="border: 1px solid black; padding: 2px;">DC3 S</span>	= SCREEN SIZE	<span style="border: 1px solid black; padding: 2px;">CLEAR</span>	
<span style="border: 1px solid black; padding: 2px;">DC2 R</span>	= RESYNC	<span style="border: 1px solid black; padding: 2px;">CONT</span>	

13. This completes the example procedure for monitoring synchronous data. In most cases, the next step is to return to the Level-1 Directory and select OFF-LINE DATA DISPLAY in order to further analyze the data stored on disc. Menu selections and prompts are sufficient to lead the user through this mode of operation. To take full advantage of the ENCORE's Level-1 capabilities, the user must become familiar with the information in Chapter 5.

4. TYPICAL OPERATION IN LEVEL-2

4.01 For the purpose of this procedure, it is assumed that the user is familiar with DICOL and wishes to write a simple program that can be used to monitor synchronous data. The program shown in this procedure is designed to display all monitored data in a half- or full-duplex format until EOT is received. For additional information, please refer to Chapter 6.

**TYPICAL LEVEL-2 OPERATING PROCEDURE  
FOR  
SYNCHRONOUS MONITOR**

STEP	PROCEDURE
------	-----------

1. Connect the auxiliary interface cable between the ENCORE's rear panel UUT connector and the circuit under test. Assure that the NORMAL/LOOP-BACK switch is set to NORMAL.
2. Power-up the unit as described in the procedure under paragraph 2.02.
3. From the Master Directory, select Level-2 by striking the 2 key and note that the Level-2 Directory is displayed.
4. From the Level-2 Directory, select the Program Mode by striking the DLE P key and note display of the LEVEL-2 EDITOR.
5. If a program is displayed, you must either erase it by striking the ^ P key twice or move to the end of the program by holding the SHIFT key down while striking the CLEAR key. In either case, the first instruction must be "63", followed by the program name. If a program is not displayed, simply begin a new program by typing the program name.
6. Enter the following program. Strike the SPACE BAR for automatic colon entry after the instruction number and the ↑ key after the argument.

Line#	Instruction	Argument	Comments
00	: 63	: MONITOR	Program name
01	: 10	:	Resync both receivers
02	: 13	: <u>EOT</u>	<span style="border: 1px solid black; padding: 0 2px;">CTRL</span> <span style="border: 1px solid black; padding: 0 2px;">EOT D</span> . Receive and log through <u>EOT</u> .
03	: 04	: -2	Jump back 2 steps

7. Strike the CLEAR key and note that four lines (00-03) of the program are displayed. Use the ↑ and ↓ keys to scroll through the program and check for errors. If an error is found, use the keystrokes shown in the table below to edit the program.



**TYPICAL LEVEL-2 OPERATING PROCEDURE  
FOR  
SYNCHRONOUS MONITOR (Cont'd)**

STEP	PROCEDURE
------	-----------

**PACER/ENCORE Program Mode Commands**

ITEM	PACER	ENCORE	FUNCTION
1.	SHIFT ↑	SHIFT ↑	Advance to next program.
2.	SHIFT →	SHIFT →	Insert line above current line.
3.	↑	↑	Go ahead one line.
4.	↓	↓	Go back one line.
5.	←	←	Character delete.
6.	→	→	Enter/Exit HEX Mode.
7.	HOME	CLEAR	Go to program start.
8.	SHIFT HOME	SHIFT CLEAR	Go to program end.
9.	SHIFT →	SHIFT DC	Erase line.
10.	MODE SHIFT ENTER	^ P ^ P	Erase program.
11.	SHIFT	CONT	Invert parity.
12.		UC ONLY	Enable/Disable upper-case.



**NOTES:** Item #11 above is necessary because of the absence of an auxiliary shift key on the ENCORE. On the ENCORE, the **CONT** key will turn the parity invert functions on or off. Whenever the invert function is on, an indication will appear on the top line of the display.

Item #12 above is an added ENCORE function which allows only upper case alphas when enabled. Whenever the upper case function is on, there will be a "UC" displayed on the top line of the CRT. This function is especially useful since there is only one shift key on the ENCORE.

8. Strike the **CMD** key and note that the ENCORE displays the Level-2 Directory. It is good practice, at this point, to save the program on disc by selecting item **5** from the menu. This will prevent accidental loss of the program if power is turned off. Refer to Chapter 6, page 6-38 for additional information.
9. From the Level-2 Directory, select the IO Mode by striking the **HT** key and note that the current IO parameters are displayed.
10. Set IO parameters to agree with the parameters of the circuit under test. Refer to Chapter 7 for additional information.

## TYPICAL OPERATION IN LEVEL-2, TYPICAL OPERATION IN LEVEL-3


### TYPICAL LEVEL-2 OPERATING PROCEDURE FOR SYNCHRONOUS MONITOR (Cont'd)

STEP	PROCEDURE
	EXAMPLE:  SPEED -- 2400 SYNC --- S <sub>Y</sub> S <sub>Y</sub> NRZI --- NO CLOCK -- EXT PARITY - ODD LANG --- ASCII
11.	Strike the  key when IO entries are completed and note display of the Level-2 Directory.
12.	Execute the MONITOR program by striking the  key and note that both transmit and receive data (if present) are displayed in an interlaced format.
13.	This completes the example procedure for writing, saving, and executing a synchronous monitor program in Level-2. It is important to note that the data is stored in capture memory and can be recalled for further analysis while in the Display Mode. All Level-2 modes of operation are detailed in Chapter 6.

### 5. TYPICAL OPERATING PROCEDURE, LEVEL-3

**5.01** For the purpose of this procedure, it is assumed that the user wishes to write a synchronous COMBASIC monitor program that receives and logs data into the capture buffer.

### TYPICAL LEVEL-3 OPERATING PROCEDURE FOR SYNCHRONOUS MONITOR

STEP	PROCEDURE
1.	Connect the auxiliary interface cable between the ENCORE's rear panel UUT connector and the circuit under test. Assure that the NORMAL/LOOP-BACK switch is set to NORMAL.
2.	Power-up the unit as described in the procedure under paragraph 2.02.
3.	From the Master Directory, select Level-3 by striking the  key and note that the ENCORE enters the Level-3 Command Mode.

**TYPICAL LEVEL-3 OPERATING PROCEDURE  
FOR  
SYNCHRONOUS MONITOR (Cont'd)**

STEP	PROCEDURE
4.	From the command mode type IO <input type="button" value="RETURN"/> and note that the ENCORE enters the IO Mode.
5.	Set IO parameters to agree with the parameters of the circuit under test. Refer to Chapter 7 for additional information.
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>EXAMPLE:</p> <p>SPEED -- 2400</p> <p>SYNC --- S<sub>Y</sub> S<sub>Y</sub></p> <p>NRZI --- NO</p> <p>MODE --- SYNC</p> <p>CLOCK -- EXT</p> <p>PARITY - ODD</p> <p>LANG --- ASCII</p> </div> <div style="width: 45%;"> <p><b>Note:</b> These are the default parameters assumed on power-up.</p> </div> </div>
6.	Strike the <input type="button" value="CMD"/> key when IO entries are completed and note that the ENCORE returns to the Level-3 Command Mode.
7.	From the Level-3 Command Mode type EDIT MONITOR <input type="button" value="RETURN"/> and note that the ENCORE enters the Edit Mode.
8.	Enter the following program being careful to type each line exactly as shown. Terminate each line by striking the <input type="button" value="LMS FEED"/> key. Use the <input type="button" value="←"/> , <input type="button" value="→"/> , <input type="button" value="⌘"/> , and <input type="button" value="⌘"/> keys to correct mistakes. To correct a line already terminated, simply retype the line number, strike <input type="button" value="RETURN"/> and correct the error. For more information on editing a program, refer to Chapter 7.

Line#	Statement	Comments
10	SCREEN CP=6-1 "SYNCHRONOUS MONITOR"	Clear screen and print the message, "SYNCHRONOUS MONITOR".
20	SCREEN P=5-2 "STRIKE ""E <sub>C</sub> "" TO ESCAPE"	Print second line of message, "STRIKE ""E <sub>C</sub> "" TO ESCAPE".
30	WAIT 5000	Wait 5000 milliseconds.
40	STATE BRG, DISCON	Bridge both pins 2 and 3, and turn on the disc.

**TYPICAL LEVEL-3 OPERATING PROCEDURE  
FOR  
SYNCHRONOUS MONITOR (Cont'd)**

STEP	PROCEDURE	
Line#	Statement	Comments
50	TON P2 , P2B , P2D , P3 , P3B , P3D	Turn on the front ends, capture buffer (P2B, P3B), and display (P2D, P3D) for pins 2 and 3.
60	WHEN KBD a GOTO 80	* Interrupt the program whenever the keyboard is struck.
70	GOTO 70	Continuous return to WHEN KBD if the keyboard is not struck.
80	IF a=27 GOTO 100	* If the <input type="checkbox"/> key is struck, goto step 100.
90	GOTO 60	If a key other than <input type="checkbox"/> is struck, return to step 60.
100	PRINT #3, "!"	Fill the unused portion of the buffer with asterisks to assure transfer of a complete block of characters to the disc.
110	PRINT #3, "!"	Transfer an extra block of asterisks to assure complete transfer of all data.
120	STATE DISCOFF	Turn off the disc.
130	CHAIN ""	Return to the Level-3 Command Mode.
		* <b>Note:</b> Strike <input type="checkbox"/> before and after entering the byte variable a.
9.	When the program has been entered and corrected, strike the <input type="checkbox"/> key. Now type SAVE <input type="checkbox"/> to save the program in solid state memory.	
10.	For the purpose of this procedure, we will assume that data is to be stored on the program disc. Eject the disc, remove the write protect tab, and re-insert the disc.	
11.	Type RUN <input type="checkbox"/> and note that the program message is displayed, and after 5 seconds, synchronous data is displayed on the screen, logged into the buffer, and transferred to the disc. Strike the <input type="checkbox"/> key to exit the program and return to the Level-3 Command Mode.	

**TYPICAL LEVEL-3 OPERATING PROCEDURE**  
**FOR**  
**SYNCHRONOUS MONITOR (Cont'd)**

---

<b>STEP</b>	<b>PROCEDURE</b>
12.	If the program contains any additional errors, an audible alarm will sound, an error code will be displayed, and the ENCORE will return to the Edit Mode. If the program is executed, but the results are not as expected, strike the <b>END</b> key and note return to the Edit Mode. Type LIST <b>RETURN</b> and note display of the monitor program. Make the necessary changes and repeat step 11.
13.	This completes the example procedure for writing and executing a synchronous monitor program in COMBASIC. To take full advantage of the ENCORE's Level-3 capabilities, the user must become familiar with the information in Chapter 7.

---

## CHAPTER 5

### LEVEL-1 OPERATION

#### 1. GENERAL

**1.01** This chapter of the manual contains the detailed information required for Level-1 operation of the ENCORE.

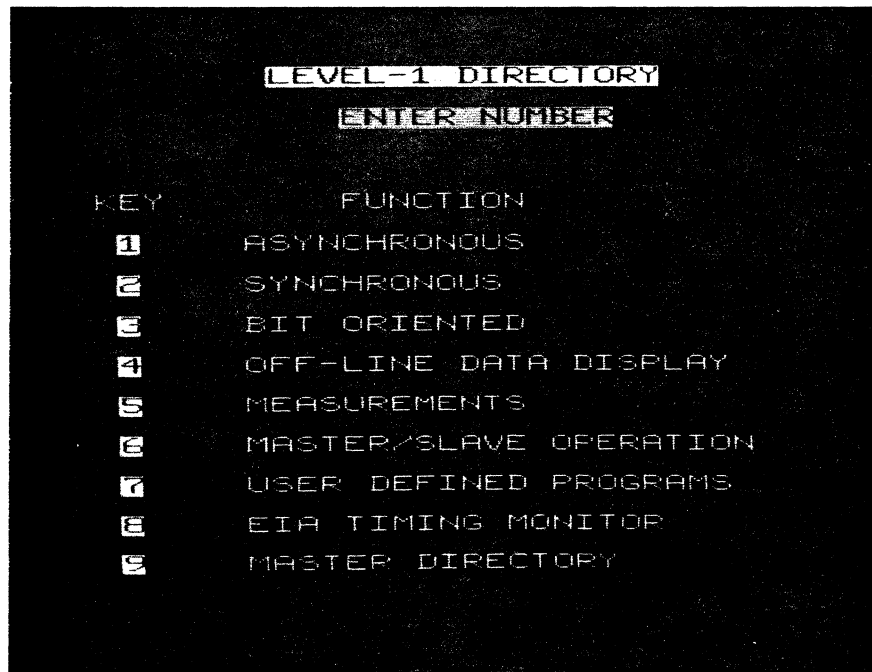
**1.02** Level-1 is a menu driven operating system designed for the non-programmer. It is entered by selecting item [1] from the Master Directory and consists of a series of programs selected and executed with minimum operator input. These are highly sophisticated programs designed for the comprehensive evaluation of a data communications network. At this level of operation, the user is not a programmer but he is familiar with network under test and understands the basics of data communications and the environment in which he is operating. To simplify operation, Level-1 provides the user with a Master Directory for selecting the desired level of operation and several program libraries to permit the selection and execution of existing programs. If, during the execution of Level-1 programs, the ENCORE enters the Level-3 Command Mode, as a result of user entry error, simply type D RUN [RETURN] to display the Master Directory.

---

#### LEVEL-1 DIRECTORY

---

**1.03** The following paragraphs briefly describe the contents of the various program libraries and operating routines listed in the Level-1 Directory, Figure 5-1. The last entry in each library will return the ENCORE to the Level-1 Directory. Select the desired program by typing the adjacent number and note that program execution begins. The keystrokes required to select an item from the directory and then return to the directory after execution, are shown in the keystroke flowchart, Figure 5-2. Detailed logic flowcharts are included in the Programmer's Notebook, Pub. No. 810-00181.



```
LEVEL-1 DIRECTORY
ENTER NUMBER

KEY      FUNCTION
1        ASYNCHRONOUS
2        SYNCHRONOUS
3        BIT ORIENTED
4        OFF-LINE DATA DISPLAY
5        MEASUREMENTS
6        MASTER/SLAVE OPERATION
7        USER DEFINED PROGRAMS
8        EIA TIMING MONITOR
9        MASTER DIRECTORY
```

**Fig. 5-1 Level-1 Directory**

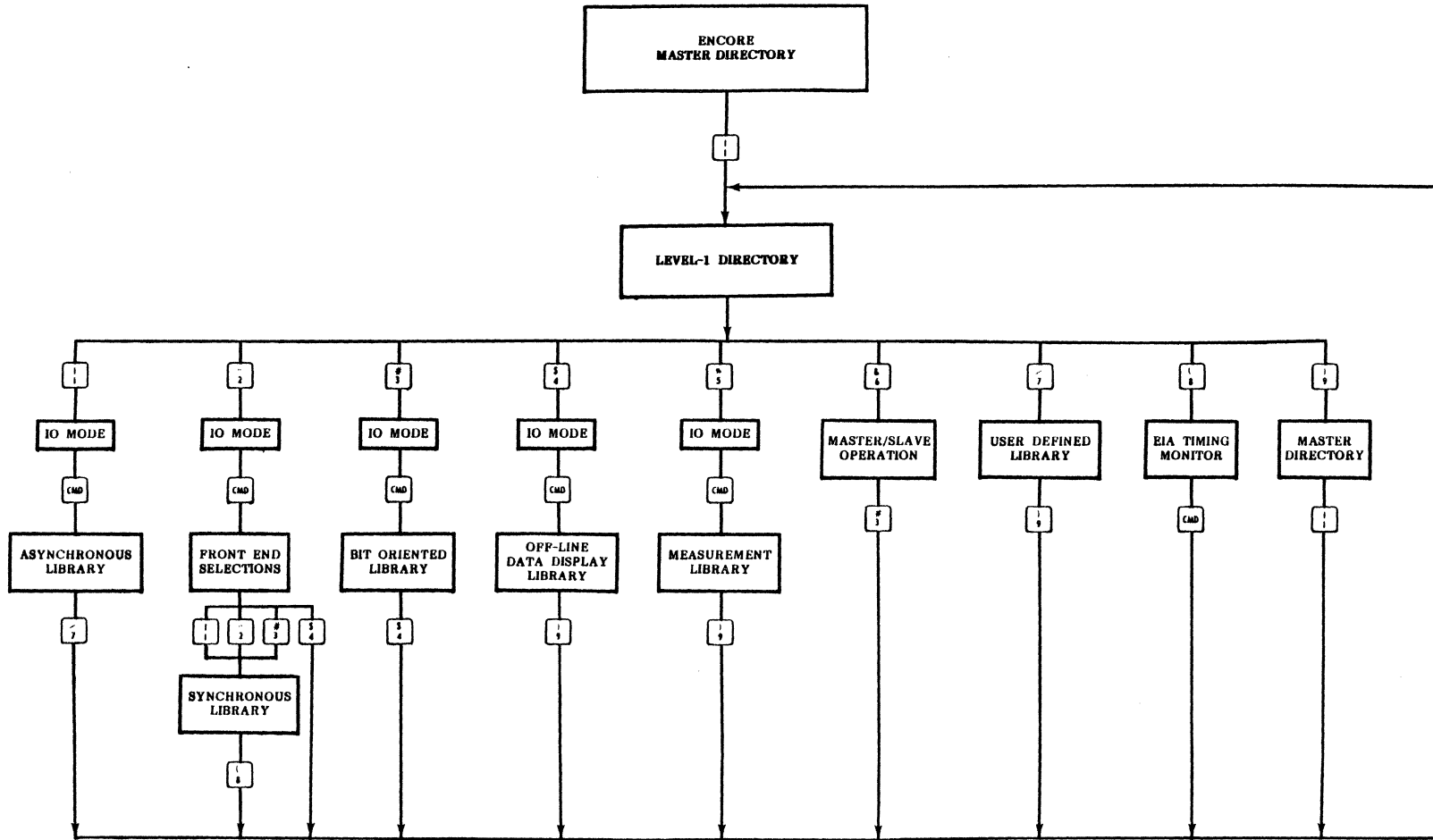


Fig. 5-2 Keystroke Flowchart, Level-1 Directory

2. ASYNCHRONOUS LIBRARY, [1]

2.01 The Asynchronous Library, Figure 5-3, includes several programs designed for operation in an asynchronous environment. When the Asynchronous Library is first selected, the Level-1 program CHAIN's the IO program to permit operator entry of IO parameters. When the desired IO parameters have been entered, program execution is continued by depressing the [CMD] key. At this point the ENCORE first loads the front end programs and then displays the Asynchronous Library. The following paragraphs, 2.02 through 2.07 discuss the operation of each program in the Asynchronous Library. The keystroke flowchart, Figure 5-4, shows how each item is selected from the library and how the user returns to the library after selection.

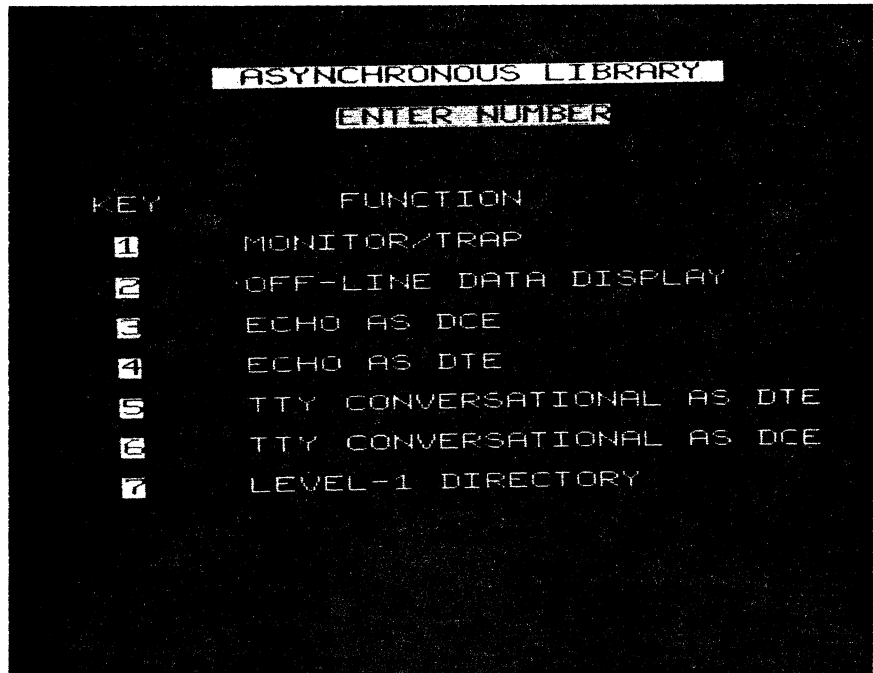


Fig. 5-3 Asynchronous Library

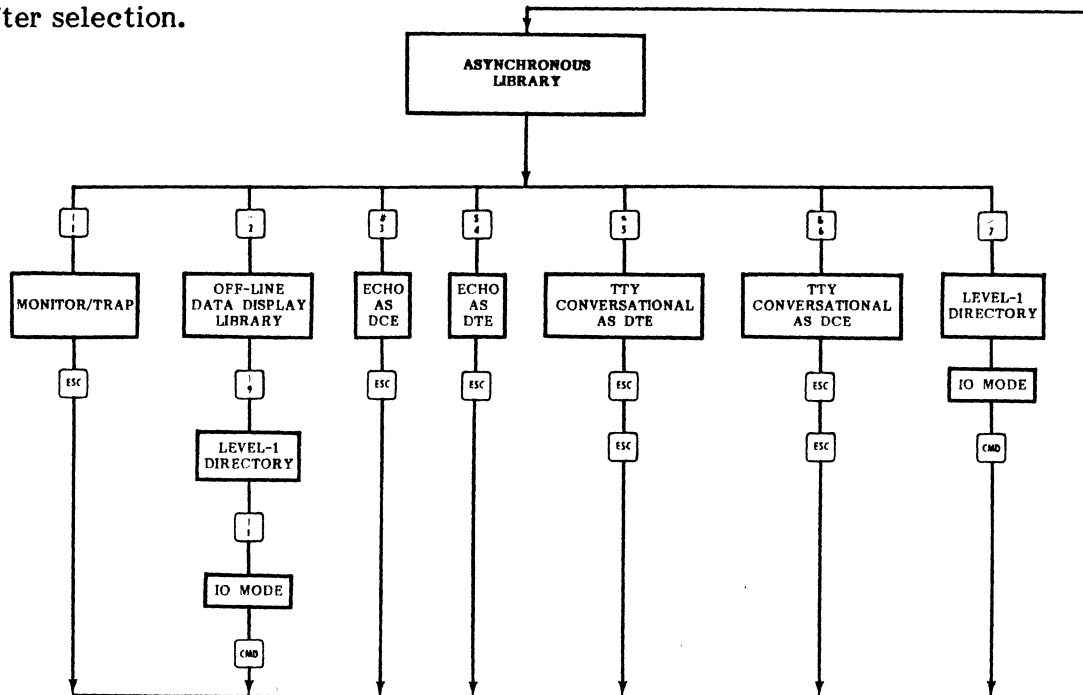


















Fig. 5-4 Keystroke Flowchart, Asynchronous Library




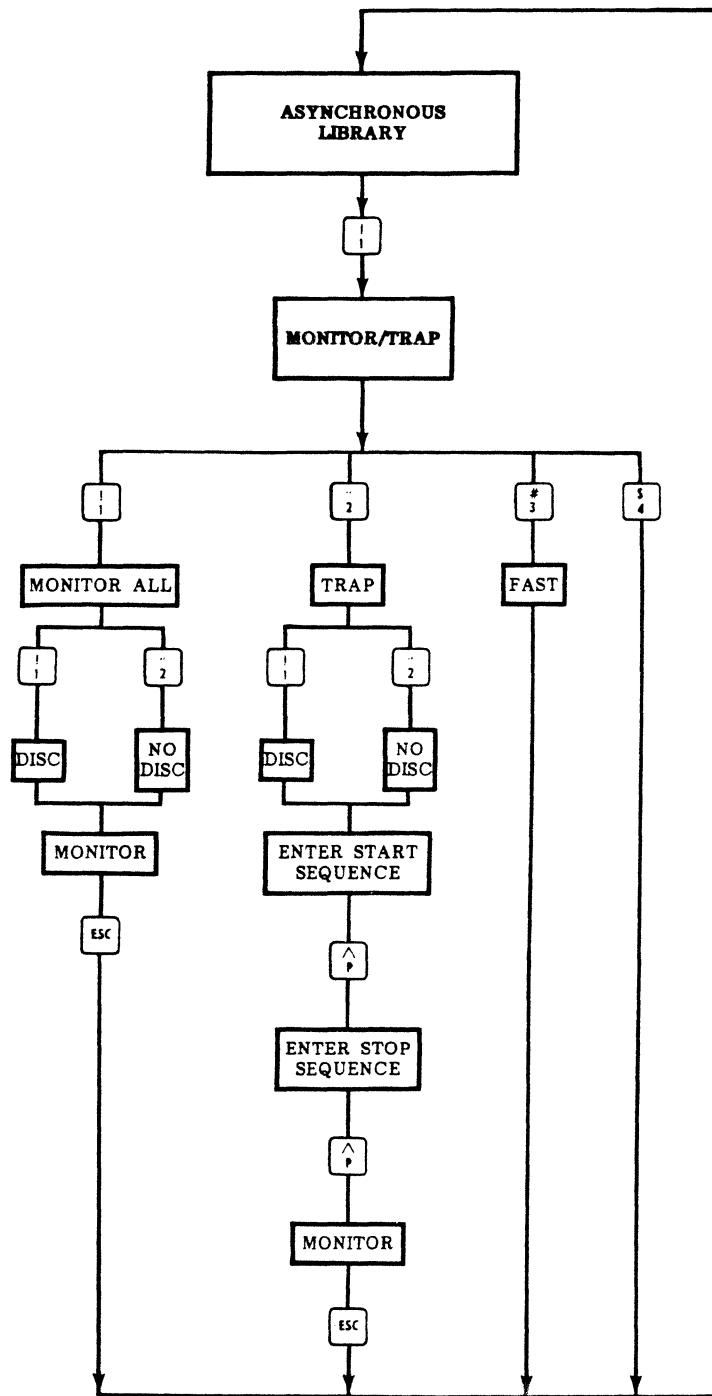
**Monitor/Trap**

**2.02** This is an asynchronous program (MONTRAP) designed to display and capture data while passively monitoring the interface. Captured data and the associated attribute or status byte may be stored in buffer memory or on disc. The program includes a routine to display and capture both data and attribute bytes occurring between specified start and stop sequences. The program makes extensive use of operator prompts and commands that may be used during program execution. These commands are defined in Table 5-1. Figure 5-5 shows the keystrokes required to enter and exit the Monitor/Trap program.

**TABLE 5-1  
MONITOR/TRAP COMMANDS**

COMMAND	DEFINITION	USAGE		
		MONITOR	ALL TRAP	FAST
	Display pin 2 and 3 data	*	*	
	Clear display (see note)	*	*	
	Continue display (see note)	*	*	
 / 	EIA status and data/Attribute and data	*	*	
	Exit program and return to Asynchronous Library	*	*	*
	Convert display to full duplex format	*	*	
	Convert display to half duplex format	*	*	
	Convert display to HEX/NO HEX	*	*	
	Suppress idle line/display idle line	*	*	
	Time stamp capture buffer	*	*	
	Resync front ends	*	*	
	Change screen size	*	*	
	Stop display	*	*	
	Display pin 2 data only	*	*	
	Display pin 3 data only	*	*	

**NOTE:** These keys are only active after the  key is depressed.



- NOTES:**
1. Data and attribute are captured, no status.
  2. To increase capture speed, data is transferred directly to the buffer, without attribute and status.

**Fig. 5-5 Keystroke Flowchart, Monitor/Trap**

### **Off-Line Data Display**

**2.03** This is the same series of programs described in paragraphs 5.01 through 5.07. They are made available in the Level-1 Directory and in the Asynchronous, Synchronous, and Bit Oriented Libraries, solely for the convenience of the operator.

### **Echo as DCE**

**2.04** This is an asynchronous program (ECHODCE) that configures the ENCORE as a MODEM. During program execution, EIA leads 5 (CTS), 6 (DSR), and 8 (RLSD) are turned ON and both transmit and receive timing signals are output via EIA pins 15 and 17, respectively. Send data is then input via pin 2 and retransmitted via pin 3. Both send and receive data are displayed on the CRT as normal and reverse video, respectively. To exit the program, depress the  ESC key. This will return program control to the Asynchronous Library.

### **Echo as DTE**

**2.05** This is an asynchronous program (ECHODTE) that configures the ENCORE as a terminal. During program execution, EIA leads 4 (RTS) and 20 (DSR) are turned on. External timing must be input via pins 15 and 17 if EXT clock has been chosen during the selection of IO parameters. Receive data is input via pin 3 and retransmitted via pin 2. Both send and receive data are displayed as normal and reverse video, respectively. To exit the program, depress the  ESC key and note that program control is returned to the Asynchronous Library.

### **TTY Conversational as DTE**

**2.06** This is an asynchronous program (TTY) that configures the ENCORE as a terminal and allows the operator to communicate, on-line, with a remote terminal. Upon execution, the ENCORE accepts typed messages from the keyboard and transmits the data in real time. Both send and receive data are displayed, real time, in half duplex format. Depressing the  ESC key twice will return program control to the Asynchronous Library.

### **TTY Conversational as DCE**

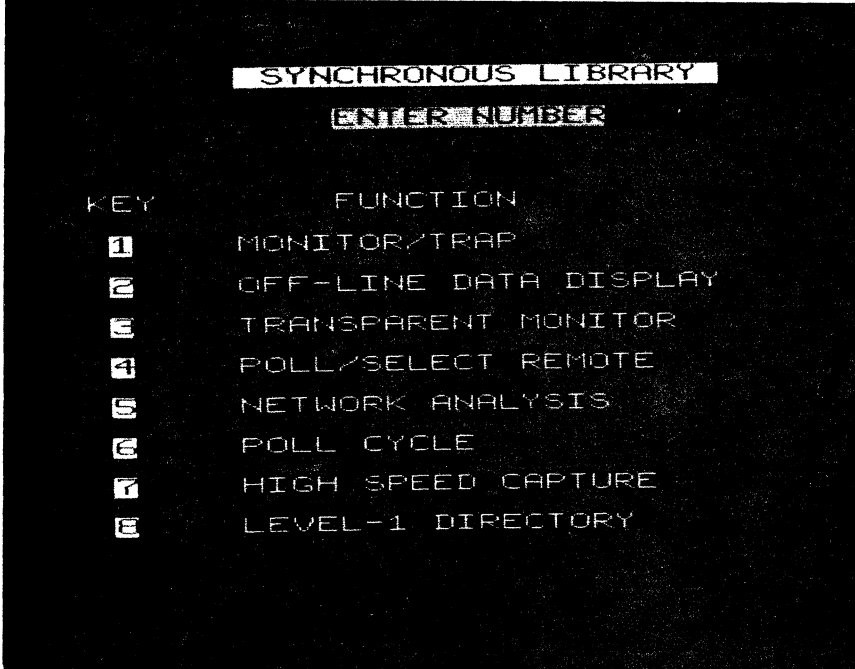
**2.07** This is the same program used for conversation while simulating DTE except that the ENCORE is configured as a modem instead of a terminal. Upon execution, data typed on the keyboard is transmitted, real time. Both send and receive data are displayed on the CRT in half duplex format. Depressing the  ESC key twice returns program control to the Asynchronous Library.

---

**3. SYNCHRONOUS LIBRARY,** 2

---

**3.01** The Synchronous Library lists programs designed for operation in a synchronous environment. When the Synchronous Library is selected from the Level-1 Directory, the Level-1 program CHAIN's the IO program to permit operator entry of IO parameters. When the desired IO parameters have been entered, program execution is continued by depressing the CMD key. The operator now responds to the CRT prompt selecting CRC (Cyclic Redundancy Check, a 16-bit character), LRC (Longitudinal Redundancy Check, an 8-bit character), or neither. Upon entering this selection, the synchronous front end program is loaded and then the Synchronous Library is displayed as shown in Figure 5-6. The following paragraphs, 3.02 through 3.05, discuss the operation of each program in the Synchronous Library. Figure 5-7 shows the keystrokes required to select an item from the Synchronous Library and return to the library after the selected program is executed.



```
SYNCHRONOUS LIBRARY
ENTER NUMBER

KEY          FUNCTION
1           MONITOR/TRAP
2           OFF-LINE DATA DISPLAY
E           TRANSPARENT MONITOR
4           POLL/SELECT REMOTE
5           NETWORK ANALYSIS
6           POLL CYCLE
7           HIGH SPEED CAPTURE
E           LEVEL-1 DIRECTORY
```

**Fig. 5-6 Synchronous Library**

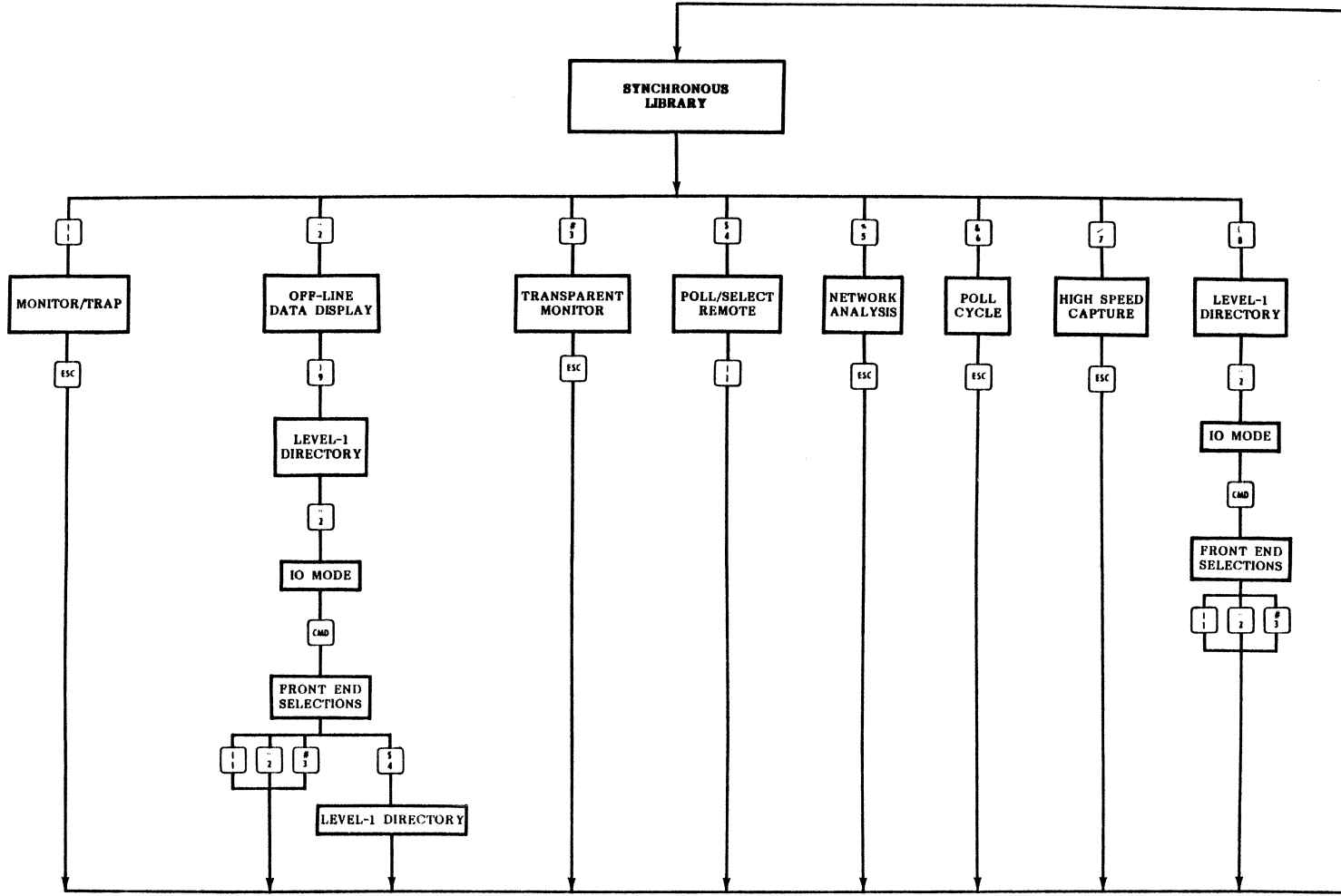


Fig. 5-7 Keystroke Flowchart, Synchronous Library

**Monitor/Trap**

**3.02** This is the same program used during asynchronous operation and is detailed in paragraph 2.02. The difference between synchronous and asynchronous monitor lies in the type of front end program that is automatically executed before "Monitor/Trap". In all other respects, program control is accomplished in the same manner as described for asynchronous operation. All commands listed in Table 5-1 are also valid during synchronous operation.

**Off-Line Data Display**

**3.03** This is the same library of programs described in paragraphs 5.01 through 5.07. This selection is made available in the Synchronous Library solely for the operator's convenience.

**Transparent Monitor**

**3.04** This program is designed to passively monitor data at the interface and capture both data and EIA status. The data is captured without regard to language and is normally displayed, after capture, using one of the Off-Line Data Display programs. CRT prompts allow the operator to store captured data in both the buffer and on disc or in the buffer only. During program execution, the display of captured data may be controlled using the commands listed in Table 5-2.

**TABLE 5-2**

**TRANSPARENT MONITOR COMMANDS**

COMMAND	DESCRIPTION
SOH A	Display pin 2 and pin 3 data.
CLEAR	Clear display.
CONT	Continue display.
ESC	Exit program and return to Library Display.
ACK F	Convert display to full duplex format.
BS H	Convert display to half duplex format.
HEX	Convert display to HEX or return to ASCII.
HT I	Suppress idle line/do not suppress idle line.
DC3 S	Change screen size.
STOP	Stop display.
" 2	Display pin 2 data only.
# 3	Display pin 3 data only.

### **Poll/Select Remote**

**3.05** This program is designed to poll or select a terminal in a BSC (Binary Synchronous Communications) environment using the EBCDIC language and a CRC 16 block check polynomial. During execution, the ENCORE simulates a front end processor in a data network. CRT prompts allow the operator to store captured data in the buffer or on disc and to enter the station poll address, select address, and device address. Additional prompts then allow the operator to send a general poll, a specific poll, or to select a remote terminal and send an operator entered message. If a general or specific poll is to be sent, further prompting allows the operator to select the number of successive polls to be transmitted (32000 max). If there is no response to the poll within three seconds, a "TIME OUT! NO RESPONSE!" message is displayed. After five consecutive time out messages, a final prompt allows the operator to exit the program, start the program from the beginning or repeat the poll. At this point, the operator should determine the reason for the "TIME OUT!" message and then restart the program.

### **Network Analysis**

**3.06** This program is designed to analyze network performance in a BI-SYNC environment at speeds up to 9 600 bps and is only run between the user modem and CPU. It will analyze the data for up to 20 stations with a maximum of 200 total devices. The language employed is EBCDIC and the terminal equipment is assumed to be IBM 3270; TELETYPE 40/4 and 4540; or equivalent. The maximum speed of 9 600 bps is valid for point-to-point communications. For multi-drop application, the speed is limited to 4 800 bps. User prompts are provided to input speed and test duration. To exit the program, strike the  key.

**3.07** During execution of this program, the ENCORE displays information relative to overall line performance and individual station and device activity. This information includes such items as: line utilization, response time, number of messages transmitted, message size, poll cycle time, number of retransmissions, and error performance. The information required for analysis of the line under test is obtained by passively monitoring that line for a period of 1 to 60 minutes as specified by the user. The ENCORE displays this information on its CRT and may output the same information to a hard copy device if desired.

**3.08** During the monitor phase of execution, the station polling address (shown in EBCDIC), number of polls, number of messages (IN, OUT) and retransmission for up to 20 stations is logged and displayed on the CRT as shown in Figure 5-8. This same display also includes the last, minimum and maximum poll cycle, and CPU response times in milliseconds. This information is then analyzed by the ENCORE and the results of analysis are subsequently displayed as two different reports: "LINE TOTALS" and "STATION TOTALS". Elapsed time (RUN TIME) and selected test duration (END TIME) are also shown in the display.

## SYNCHRONOUS LIBRARY

	LAST	MIN	MAX	MIN:SEC
POLL CYCLE	515	433	11058 MSEC.	RUN TIME 5:00
CPU RESPONSE	240	238	529 MSEC.	END TIME 5:00
SPA POLLS IN/OUT	REXMIT-IN/OUT		SPA POLLS IN/OUT	REXMIT-IN/OUT
C 416 12/ 11	/		@ 416 20/ 14	/
A 421 25/ 14	/		B 416 21/ 19	679/
P /	/		/	/
/	/		/	/
/	/		/	/
/	/		/	/
/	/		/	/
/	/		/	/

**Fig. 5-8 Typical Monitor Display**

**3.09 Overall Line Performance:** The overall performance of the circuit under test is determined by examining the "LINE TOTALS" as shown in Figure 5-9. This report totals measurement parameters for all stations monitored during the selected time period. The report also includes the RUN TIME and a date-time group to mark the beginning and ending of the test. These parameters are defined as follows:

- MSGS IN** - Total number of messages received.
- MSGS OUT** - Total number of messages transmitted.
- CHAR IN** - Total number of characters received including the minimum message size, maximum message size, and average message size (CHAR IN/MSGS IN) in characters.
- CHAR OUT** - Total number of characters transmitted including the minimum message size, maximum message size, and average message size (CHAR OUT/MSGS OUT) in characters.
- RESP TIME** - The minimum, maximum, and average response times measured from CPU input of "ETX" to CPU output of "SEL". Times are given in milliseconds.
- POLL CYCLE TIME** - The minimum, maximum, and average times required to cycle through an entire poll list (top of list to top of list). Times are given in milliseconds.
- REXMIT IN** - Total number of "NAK's" received.
- REXMIT OUT** - Total number of "NAK's" transmitted.
- LINE UTILIZATION** - The total number of characters transmitted or received divided by total possible for the selected speed and measurement period. Utilization is given in percent.



- LINE ERROR PERFORMANCE** - The percent of characters sent error free.
- WORST CASE RESPONSE TIME** - The sum of the maximum response time and maximum poll cycle time.

```

LINE TOTALS      RUN TIME = 5 MINUTES
TIME START      123 16:14:29      TIME STOP 123 16:19:29
                TOTAL          MIN          MAX          AVG
MSGs IN         78
MSGs OUT        58
CHAR IN        1866           4          364          23
CHAR OUT       8783          10         339          151
RESP TIME              238          529          266 MS.
POLL CYCLE TIME      433          11058         635 MS.
REXMIT IN        0
REXMIT OUT       679
LINE UTILIZATION           5.91 %
LINE ERROR PERFORMANCE      93.62 %
LINE TOTALS (CONTINUED)
WORST CASE RESPONSE TIME    11587 MS.
    
```

**Fig. 5-9 Line Totals**

**3.10 Station Performance:** Station performance is evaluated by examining the "STATION TOTALS" as shown in Figure 5-10. This report includes the same type of information provided in the "LINE TOTALS" report except that the information is only relative to the specified station (address now shown in HEX, not EBCDIC). Included with the "STATION TOTALS" are message totals for each station device. These totals include the total messages transmitted and received, total characters transmitted and received, and the minimum, maximum, and average message size.

```

STATION TOTALS
STATION ADDRESS  C3
                TOTAL          MIN          MAX          AVG
MSGs IN         12
MSGs OUT         11
CHAR IN         394           4          209          32
CHAR OUT       1736          10         339          157
REXMIT IN        0
REXMIT OUT       0
LINE ERROR PERFORMANCE  100 %
DEVICE ADDRESS  40
MSGs IN/OUT      12/ 11
CHARS IN-- 394 TOTAL           4 MIN          209 MAX          32 AVG
CHARS OUT- 1736 TOTAL          10 MIN          339 MAX          157 AVG
    
```

**Fig. 5-10 Station Totals**

4. BIT ORIENTED LIBRARY, #  
3

4.01 The Bit Oriented Library contains many of the same programs listed in the Asynchronous and Synchronous Libraries. When used in a bit oriented environment, however, a different front end program is executed before a selection from the library is made. The front end program (FESDLC) is designed to permit operation of the ENCORE in a SDLC (Synchronous Data Link Control) environment. When the Bit Oriented Library is selected, the Level-1 program CHAINS to the ROM stored IO program to permit operator entry of IO parameters. During IO set up, the MODE parameter must be set to SDLC. When the desired IO parameters have been entered, program execution is continued by depressing the CMD key. The front end program is then loaded and the Bit Oriented Library is displayed as shown in Figure 5-11. Each of the programs shown in the library are discussed in paragraphs 4.02 through 4.04. Figure 5-12 shows the keystrokes required to select an item from the library and to return to the library after the selected program is executed.

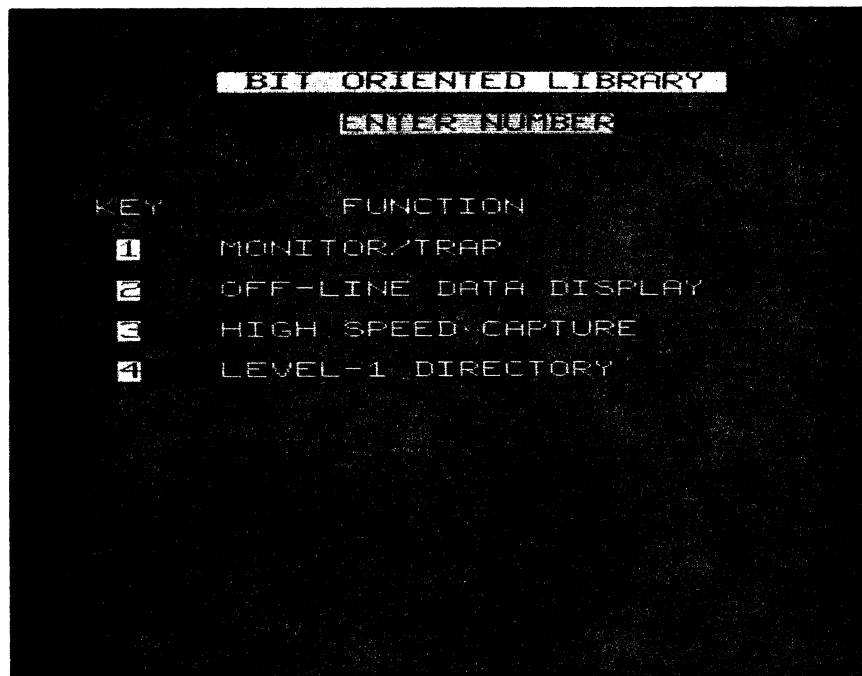


Fig. 5-11 Bit Oriented Library

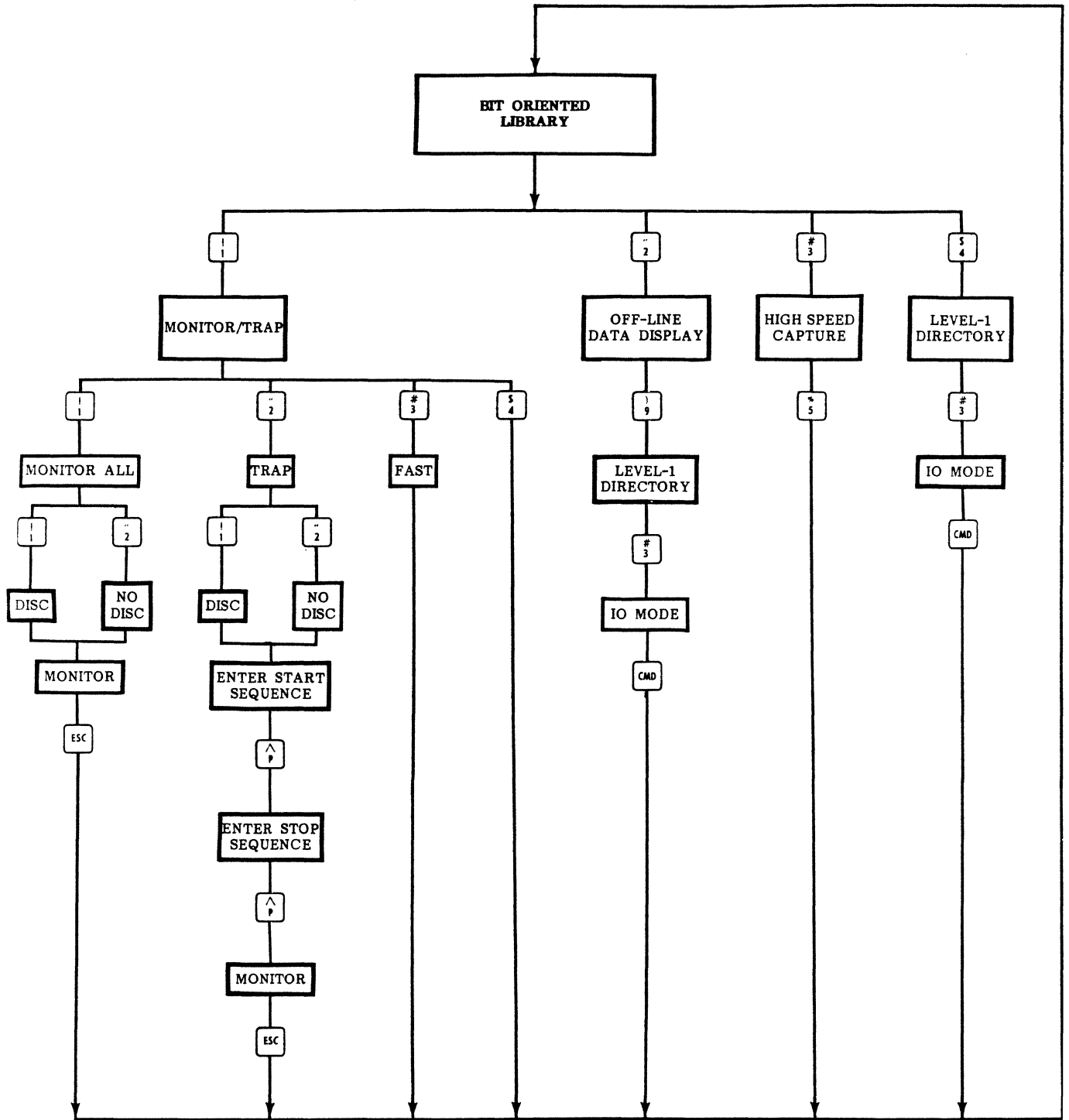


Fig. 5-12 Keystroke Flowchart, Bit Oriented Library

## BIT ORIENTED LIBRARY, OFF-LINE DATA DISPLAY LIBRARY

### Monitor/Trap

**4.02** This is the same program listed in both the Asynchronous and Synchronous Libraries and is detailed in paragraph 2.02. This program, like the synchronous MONITOR/TRAP, differs from the asynchronous version only by the type of front end program that precedes its execution. In all other respects, program control is accomplished in the same manner as described for asynchronous operation. All commands listed in Table 5-1 are still valid during execution in a bit oriented environment.

### Off-Line Data Display

**4.03** This is the same series of programs described in paragraphs 5.01 through 5.07. The selection is made available in the Bit Oriented Library solely for the operator's convenience.

---

## 5. OFF-LINE DATA DISPLAY LIBRARY,

5 4
--------

**5.01** The Off-Line Data Display Library, Figure 5-13, is a collection of programs designed to display data that has been captured and stored in the capture buffer or on disc. Other selections in the library allow the user to set IO and XIO parameters and to output the captured data via the External IO Port. The flowchart, Figure 5-14, shows how each item in the library is selected and how the user returns to the library after the selected program is run. Table 5-3 shows the attributes that may be displayed beneath each data byte when data and attribute are selected for display.

**TABLE 5-3**  
**DISPLAY ATTRIBUTES**

SYMBOL	DEFINITION
$\hat{P}$	Parity error.
$\hat{B}$	Block Check Character. If blinking, the BCC is bad.
$\hat{T}$	Inserted idle character generated by ENCORE.
$\hat{F}$	SDLC flag.
III	40 character FIFO buffer is overwritten.
■	End of data, incoming data.
-	Selected lead is on.

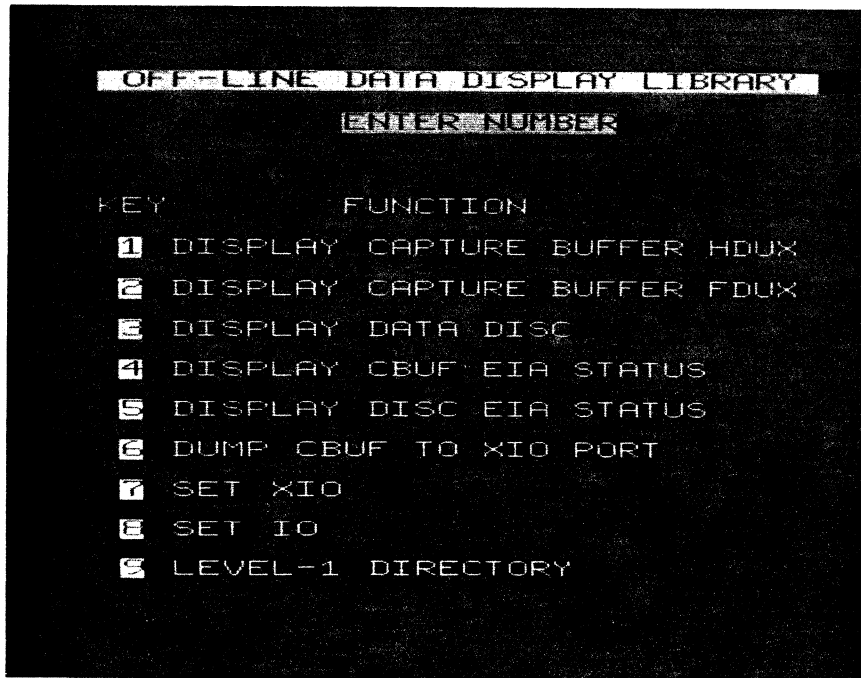


Fig. 5-13 Off-Line Data Display Library

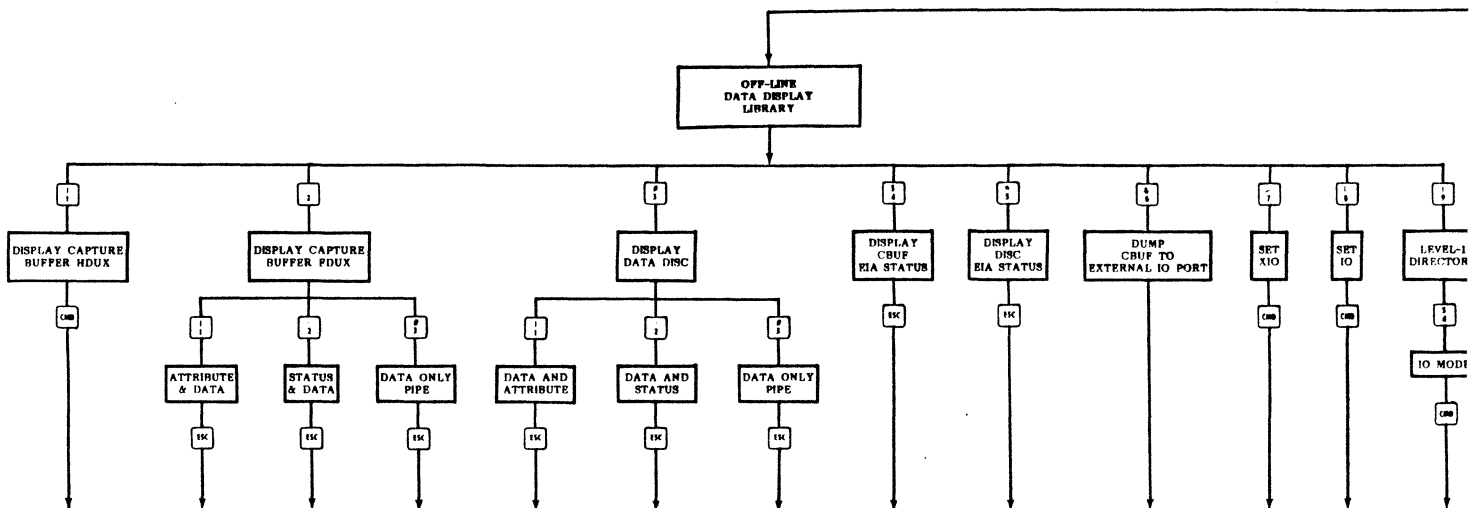


Fig. 5-14 Keystroke Flowchart, Off-Line Data Display Library
















**Display Capture Buffer HDUX**

**5.02** This program (DISP) is designed to display data in the capture buffer, stored there as a result of executing a capture program, e.g., Monitor/Trap. The IO parameters should be the same as those used when the data was captured. Because data may be captured with or without status or attribute bytes, it is important that the operator display data in the same manner as it was captured. For example, if data is captured in the FAST Mode, it should be displayed in the DATA ONLY (PIPE) Mode. This is accomplished through the use of easy to understand operator prompts. If an incorrect response is made, momentarily depress the **ESC** key and reselect the display program. During program execution, control of the display is provided through the use of operator commands as outlined in Table 5-4.

**TABLE 5-4**  
**HDUX BUFFER DISPLAY COMMANDS**

COMMAND	DEFINITION
<b>Ø</b> — <b>)</b> <b>9</b>	Slew speed factor.
<b>HEX</b>	Toggle display to HEX or octal or normal. Allow entry of HEX pair when entering the search sequence.
<b>VE</b> <b>STAT</b>	Enter and exit from sequence entry mode.
<b>?</b> <b>/</b>	Toggle mode from "AD" to "SD" to "D" where: "AD" = Attribute and Data, "SD" = Status and Data, and "D" = Data only.
<b>DC3</b> <b>S</b>	Search for sequence.
<b>STX</b> <b>B</b>	Search for B attribute. "AD" mode only.
<b>CTRL</b> <b>STX</b> <b>B</b>	Search for blinking B attribute. "AD" mode only.
<b>DLE</b> <b>P</b>	Search for P attribute. "AD" mode only.
<b>ACK</b> <b>F</b>	Search for F attribute. "AD" mode only.
<b>SPACE</b>	Toggle display to DTE only or DCE only or normal.
<b>FF</b> <b>L</b>	Language toggle.
<b>ETX</b> <b>C</b>	Case shift toggle.
<b>DCI</b> <b>Q</b>	Pin underline toggle select. "SD" mode only.
<b>^</b> <b>P</b>	Program restart. Used after changing data diskettes.
<b>SI</b> <b>O</b>	Output screen to XIO printer port.
<b>↑</b>	Forward one line.
<b>↓</b>	Backward one line.

**TABLE 5-4**  
**HDUX BUFFER DISPLAY COMMANDS (Cont'd)**

COMMAND	DEFINITION
	Auto slew forward. Slew speed controlled by  thru  .
	Auto slew backward. Slew speed controlled by  thru  .
 	Bit slip left.
 	Bit slip right.
 	Go to end of data.
 	Go to start of data.
	Allow interleaved data display in the "D" mode.

**Display Capture Buffer FDUX**

**5.03** This program (DISPFCBUF) is similar in function to the program discussed in the previous paragraph. The difference lies in the display format (full duplex) and in the use of display commands. The commands used with this program are listed in Table 5-5.

**TABLE 5-5**  
**DISPLAY CAPTURE BUFFER FDUX COMMANDS**



















COMMAND	DEFINITION
	Page forward (384 character/page).
	Page backward (384 character/page).
	Shift data byte one bit to the right.
	Shift data byte one bit to the left.
	Display pin 2 and 3 data.
	Display the contents of the capture buffer from the beginning.
	Clear screen.
	Output the contents of the capture buffer, non ASCII, via the XIO port.
	Display the contents of the capture buffer from the end.

TABLE 5-5

## DISPLAY CAPTURE BUFFER FDUX COMMANDS (Cont'd)

COMMAND	DEFINITION
	Convert display to full duplex from half duplex or HEX, etc..
	Exit program and return to Library Display.
	Convert display to HEX or return to ASCII.
	Returns display to normal from HEX.
	Output the contents of the capture buffer, in ASCII, via the XIO port.
	Convert display to upper or lower case character.
	Execute search routine.
	Display pin 2 data only.
	Display pin 3 data only.

## Display Data Disc






**5.04** This program (DISP-DISC) is designed to display data already captured and logged on disc. The IO parameters should be the same as those used when the data was captured. The method of displaying data is controlled by means of operator prompts. If an error is made in response to an operator prompt, the program may be executed again by depressing the  key and then reselecting the program. The data may be displayed with or without status or attribute bytes, depending on how it was captured. Data recorded on disc is displayed in half duplex format with 256 characters per page. Control of the display is provided through the use of operator commands as outlined in Table 5-6.

TABLE 5-6

## DISPLAY DATA DISC COMMANDS

COMMAND	DEFINITION
	Page forward.
	Page backward.
	Shift data byte one bit to the left.
	Shift data byte one bit to the right.



**TABLE 5-6**  
**DISPLAY DATA DISC COMMANDS (Cont'd)**

COMMAND	DEFINITION
STX B	Back up one record.
CLEAR	Clear display.
CONT	Continue search.
EOT D	Output recorded data, non ASCII, via the XIO port.
ENQ E	Display end record.
ESC	Exit program.
HEX	Convert display to HEX or return to ASCII.
DLE P	Output recorded data, in ASCII, via the XIO port.
STOP	Stop display.
UC LC	Convert display to upper or lower case character.
UDF	Execute search routine.
" 2	Display pin 2 data only.
# 3	Display pin 3 data only.









**Display Capture Buffer and EIA Status**

**5.05** This program (DISP-STAT) is designed to display the contents of the capture buffer including both data and EIA status. Status is defined as the condition of the EIA leads, as shown in Table 5-7, at the time a given character is logged. The contents of the capture buffer are displayed twelve characters at a time. The display includes the data character (send or receive) shown in ASCII, HEX, octal, and binary, and the EIA status. During execution of this program, the display is controlled using the commands listed in Table 5-8. In order to display the EIA status associated with a data byte, the EIA status byte must be captured while monitoring data. If attribute and data or data only are captured, the results of this program will be incorrect.

**TABLE 5-7**  
**CAPTURE EIA STATUS**

EIA LEAD	DESCRIPTION	EIA LEAD	DESCRIPTION
2 (DTE)	Transmit data	8 (RLSD)	Carrier on detect
3 (DCE)	Receive data	11 (A)	Programmed by user
4 (RTS)	Request to send	18 (B)	Programmed by user
5 (CTS)	Clear to send	22 (RI)	Rind indicator

**TABLE 5-8**  
**DISPLAY CAPTURE BUFFER AND EIA STATUS COMMANDS**

COMMAND	DESCRIPTION
	Page forward (12 characters).
	Page backward (12 characters).
	Display beginning of buffer (first 12 characters).
	Continue search routine.
	Display end of buffer (last 12 characters).
	Exit program and return to Library Display.
	Page forward 10 pages.
	Execute search routine.

**Display Disc and EIA Status**

**5.06** This program (DISP-DSTAT) is similar to the previous one in that data and EIA status are displayed in the same format. The difference is that now the data and status are loaded into the ENCORE from the disc. There are also several additional commands available for display control as shown in Table 5-9.

**TABLE 5-9**  
**STATUS AND DATA DISC DISPLAY COMMANDS**

COMMAND	DESCRIPTION
↑	Page forward (12 characters/page).
↓	Page backward (12 characters/page).
STX B	Display first page of disc record (first 12 characters).
CONT	Continue search routine.
ENQ E	Display last page of disc record.
ESC	Exit program and return to Library Display.
ACK F	Read next disc record.
HEX	Enter search sequence in HEX.
FF L	Read last disc record.
DC2 R	Set disc read/write head to beginning of first record.
STOP	Stop search.
UDF	Execute search routine.

### Dump Capture Buffer to XIO Port

**5.07** This program (DUMPXIO) is designed to output the contents of the capture buffer asynchronously via the XIO port (pin 3). The data is output exactly as recorded. If, for example, the data was recorded in EBCDIC with the status byte, it will be output the same way. Each status or attribute byte is transmitted immediately preceding its associated data byte. The character length (BITS) and parity, selected when XIO parameters are set, should agree with the data as it was recorded in EBCDIC, i.e., 8 bits, no parity. IO parameters should also be the same as those used when the data was recorded. During output, XIO port pin 9 (normally reserved for testing) is set to +12V to provide the equipment under test with a data set ready (DSR, pin 6) or a receive line signal detect (RLSD, pin 8) signal as required. The +12V output at pin 9 is jumpered to the appropriate EIA pin by means of the T-connector supplied with each ENCORE. When the entire contents of the capture buffer have been transmitted, program control will return to the Asynchronous Library.

### 6. MEASUREMENT LIBRARY, 5

**6.01** The Measurement Library contains a series of programs designed to provide the operator with the means of performing specific measurements on the circuit under test. The Measurement Library is shown in Figure 5-15. Figure 5-16 shows the keystrokes required to select an item from the library and return to the library after the selected program is executed.

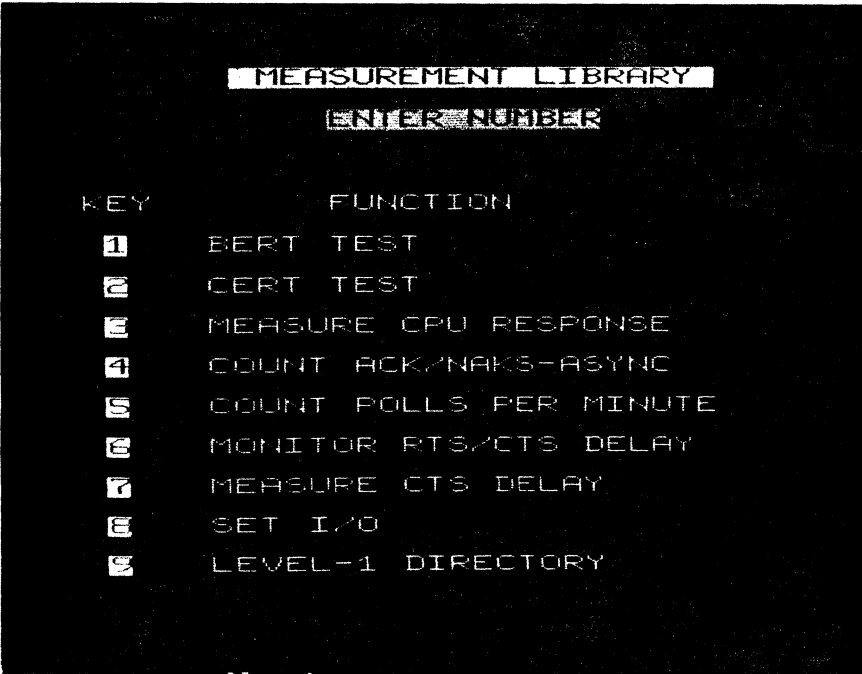


Fig. 5-15 Measurement Library

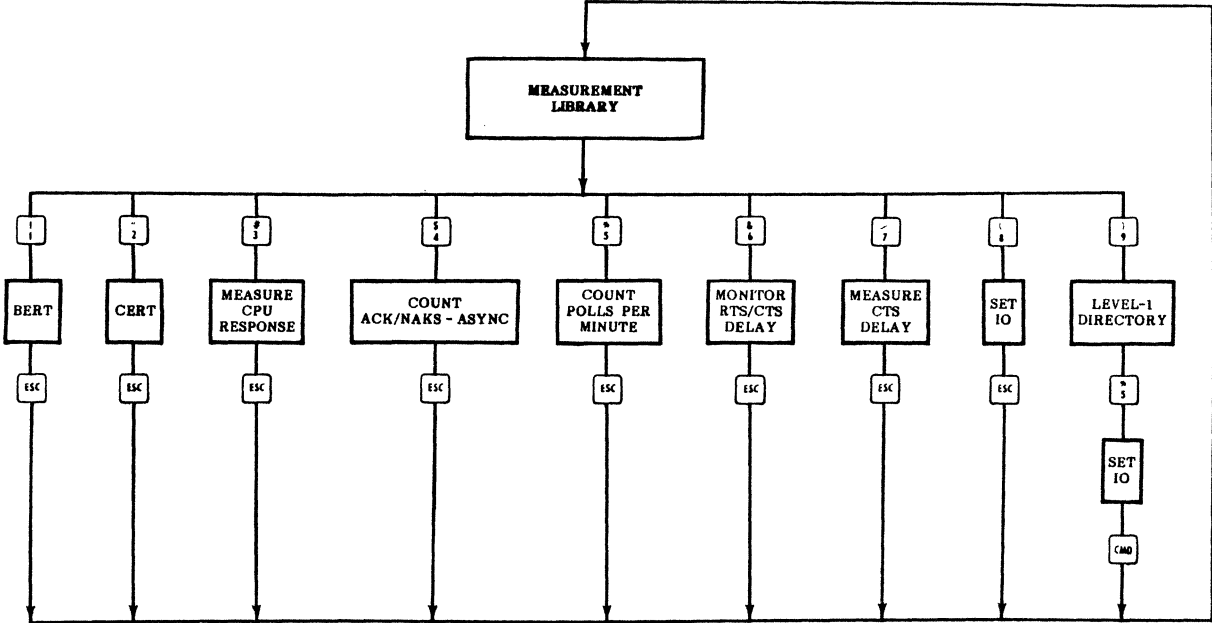


Fig. 5-16 Keystroke Flowchart, Measurement Library

### Bert

**6.02** This program is designed to transmit, receive, and count the errors that have occurred on any one of the three pseudorandom test patterns. CRT prompts allow the operator to select the desired pattern (63, 511, or 2 047 bits in length) and the duration of the test (1, 5, or 15 minutes or continuous). During execution, the ENCORE simulates a terminal and therefore requires external timing and proper handshaking. When the pattern lengths and duration of the test have been selected, transmission of the pseudorandom pattern begins immediately. Once the transmit and receive patterns have synchronized, the words "IN SYNC" are displayed in the upper right hand corner of the CRT and measurement data is displayed in a format similar to that shown in Figure 5-17. When the test is completed, an audible alarm sounds and the operator responds to CRT prompts to exit (  ) or rerun (  ) the program.

```
*** BERT ***      IN SYNC

START TIME---- 000 02 24 15
END TIME-----
TEST DURATION- 0.3 MINUTES

BAD BLOCKS---    6 ( 13.33%)
GOOD BLOCKS--   39 ( 86.66%)
TOTAL BLOCKS-   45

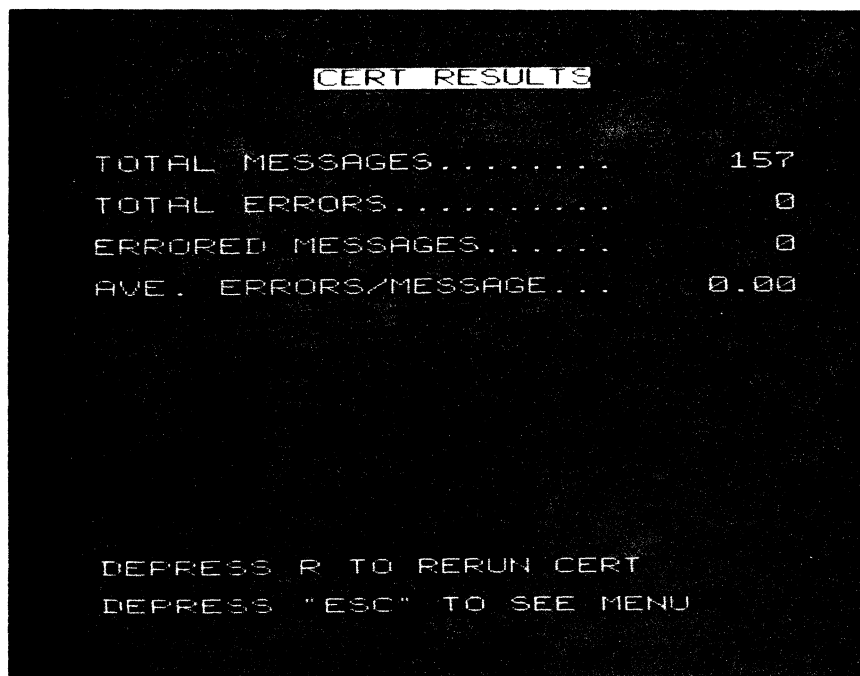
ERRORED BLOCKS / MINUTE- 20.0

SPEED= 2400      PATTERN=63
1 MINUTE TEST
```

**Fig. 5-17 BERT Measurement Results**

**Cert**

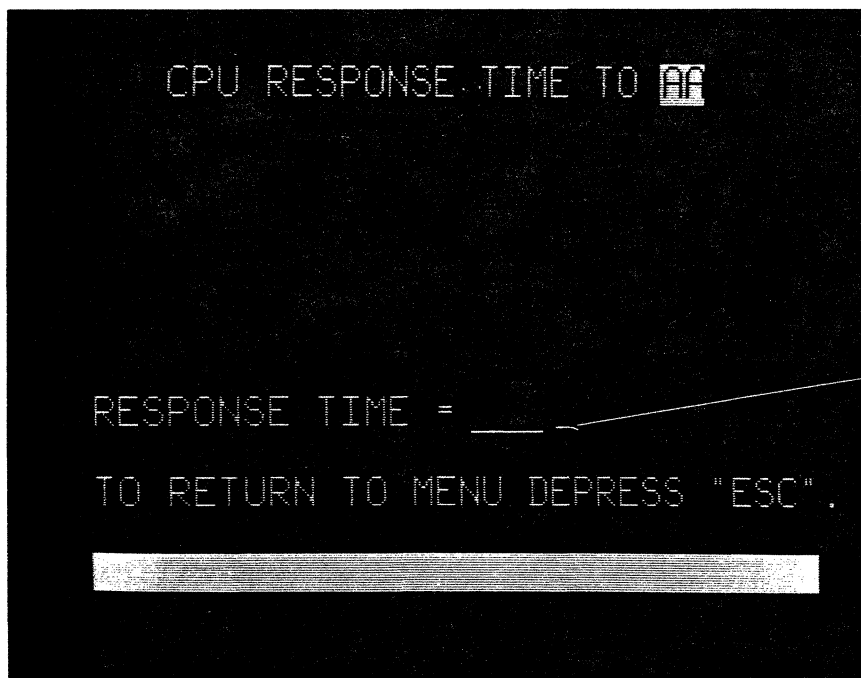
**6.03** This program is designed to transmit, receive, and count the errors on either a standard fox message or an operator entered message. During execution, the ENCORE simulates a terminal and therefore requires external timing and proper handshaking. The local or remote modem should be configured to loopback the transmitted message. As the message is received, it is displayed on the CRT until the ESC key is depressed. This will instruct ENCORE to display the results of the CERT, as shown in Figure 5-18, and, through CRT prompts, allow the operator to rerun the CERT or return program control to the Measurement Library.



**Fig. 5-18 CERT Measurement Results**

### Measure CPU Response

**6.04** This is a synchronous program (CPURESP1) designed to measure response time between CPU and 40/4 terminal. Upon execution, CRT prompts allow the operator to select the station number and the device number of the terminal to be polled. Both entries are followed by . After selecting this device number, depressing the  key instructs ENCORE to start its internal measurement timer and stop the timer when the correct response is detected. The program measures the length of time between the poll and select sequences being received at the terminal. The response time is displayed on the CRT as shown in Figure 5-19.



Response time shown here in milliseconds

**Fig. 5-19 CPU Response Time**

Count ACK/NAK's - Async

6.05 This program (ACK/NACK-A) is used in an asynchronous environment to passively monitor the interface and count the number of ACKs (acknowledgements) and NAKs (negative acknowledgements) sent (pin 2) and received (pin 3). Prior to selection of this program, IO parameters for speed, language, and parity must be set by the user. The program sets the mode to ASYN and the clock to INT. Upon conclusion of the test, the ratio of ACKs to NAKs is displayed for those sent (pin 2), for those received (pin 3), and for the total sent and received, as shown in Figure 5-20. In addition, the CURRENT TIME/START TIME display is changed to show the time at which the test was terminated. Commands used for control of the program are listed in Table 5-10. To exit the program and return to the Library display, depress the  key.

TABLE 5-10

COUNT ACK/NAKs - ASYNC COMMANDS

COMMAND	DESCRIPTION
<input type="button" value="STOP"/>	Terminate test, computes and displays results.
<input type="button" value="CLEAR"/>	Clear counters and restart test.
<input type="button" value="ESC"/>	Exit program and return to Measurement Library.
<input type="button" value="CONT"/>	Clear counter and restart test after the <input type="button" value="STOP"/> key is depressed.

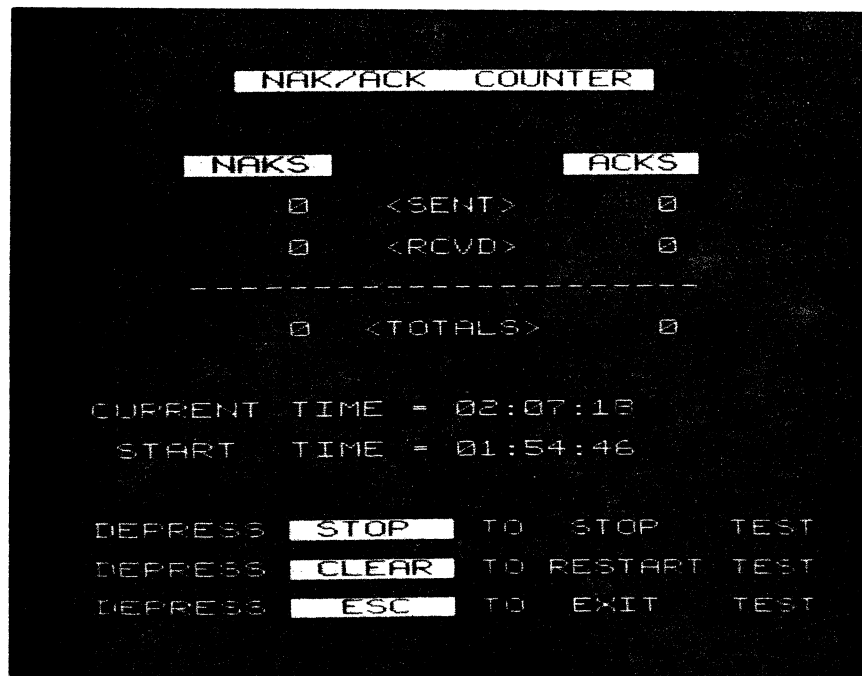
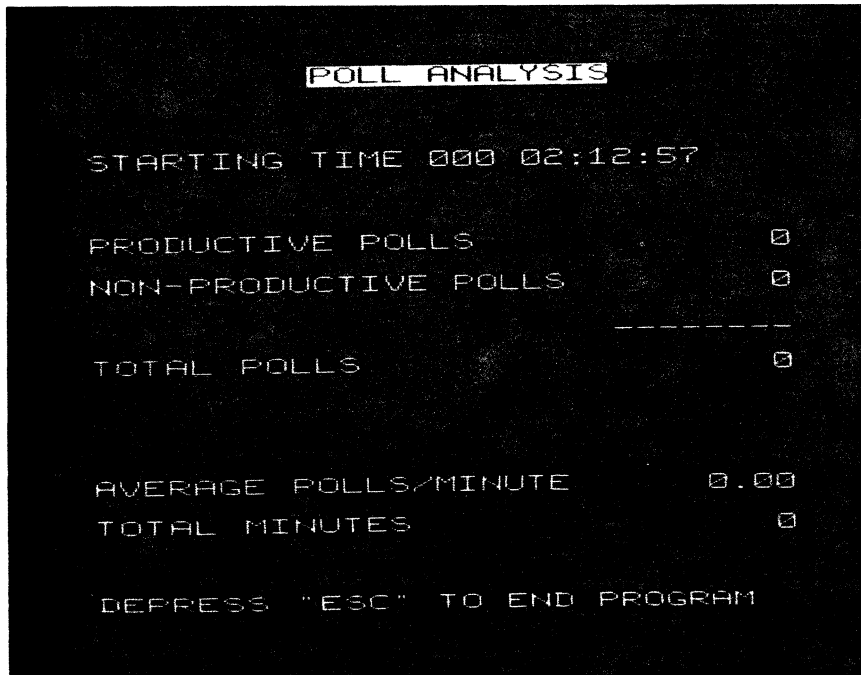


Fig. 5-20 ACK/NAK Analysis



### Count Polls Per Minute

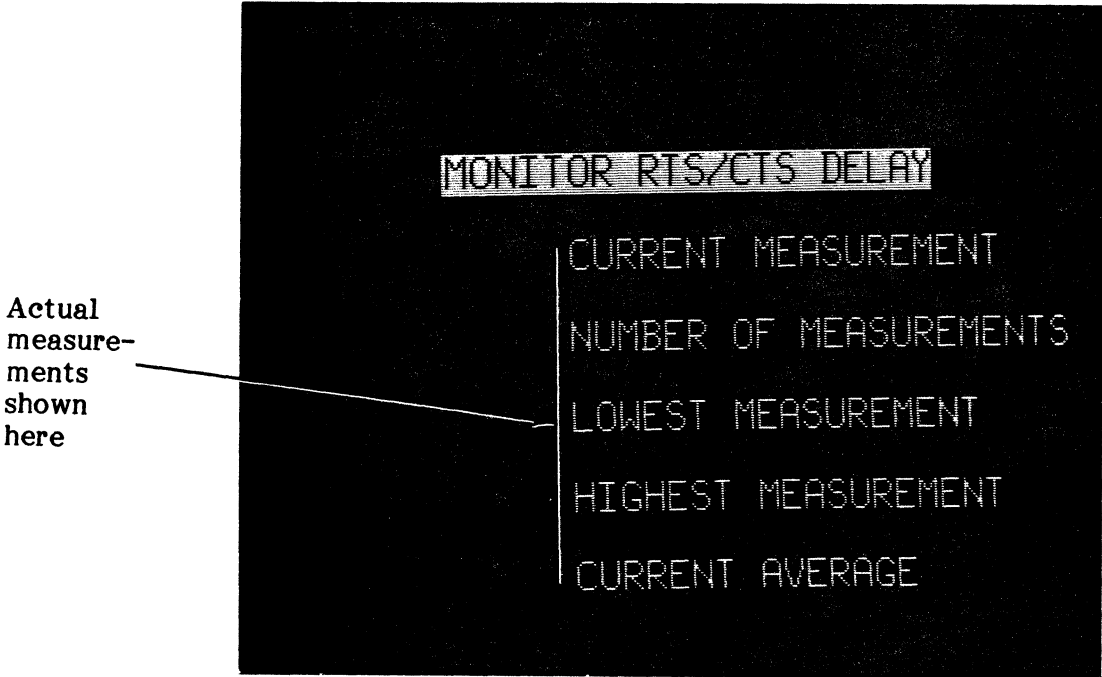
**6.06** This program (CP1) is designed to bridge both transmit (pin 2) and receive (pin 3) leads in a synchronous environment, and count the number of polls appearing on pin 2 (DTE). If an ET or NK is detected on pin 3 (DCE), the poll is considered to be nonproductive. If an SX, SH, or DL is detected on pin 3, the poll is considered productive. Polls are counted and timed on the CRT as shown in Figure 5-21. To return to the Measurement Library, simply depress the  key.



**Fig. 5-21 Poll Analysis**

**Monitor RTS/CTS Delay**

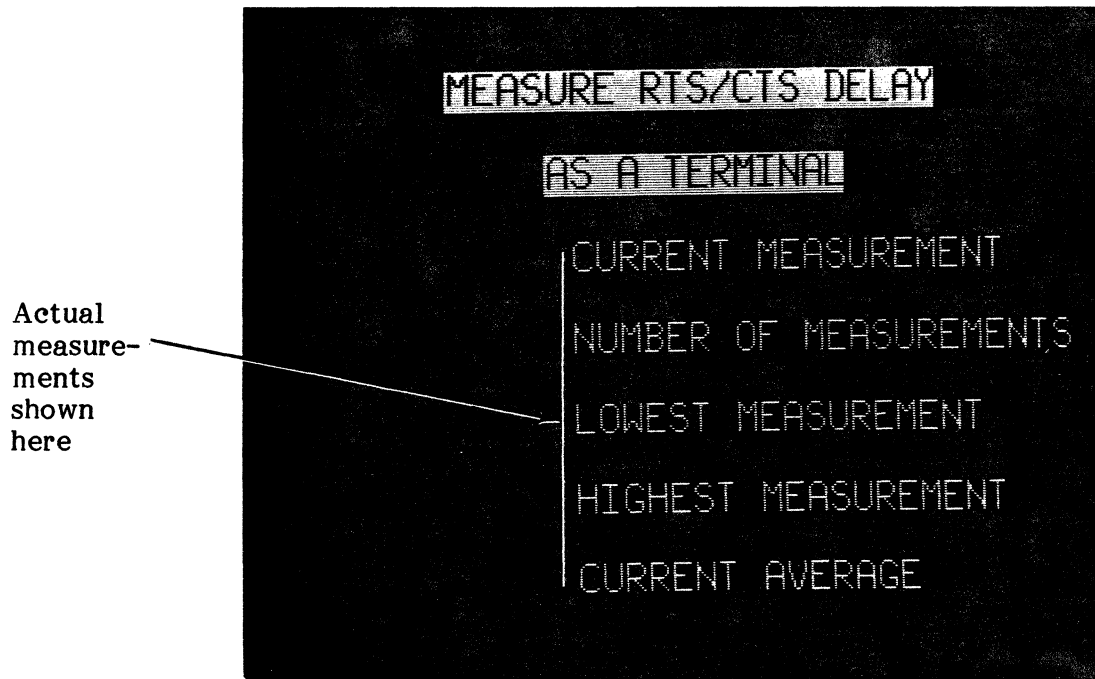
**6.07** This program (MONRTSCTS) is designed to passively measure the time between an RTS (Request To Send, pin 4) and a CTS (Clear To Send, pin 5). Since the program only measures the time between these two events, and since it is not concerned with data or timing, IO parameters are of no significance. RTS/CTS measurements are displayed on the CRT as shown in Figure 5-22. Measurements are continually updated until the ESC key is depressed. This returns program control to the Measurement Library.



**Fig. 5-22 Monitor RTS/CTS Delay**

### Measure RTS/CTS Delay

**6.08** This program (CTSDLY) is similar to the program described in paragraph 6.07 except that the ENCORE actively simulates DTE and generates the RTS by turning on EIA pin 4. Since the program only measures the elapsed time between RTS and CTS, and since data and timing have no affect on the measurement, IO parameters need not be established. The delay measurements are displayed as shown in Figure 5-23. Measurements are continually updated until the **[ESC]** key is depressed. This returns program control to the Measurement Library.



**Fig. 5-23 Measure RTS/CTS Delay as DTE**

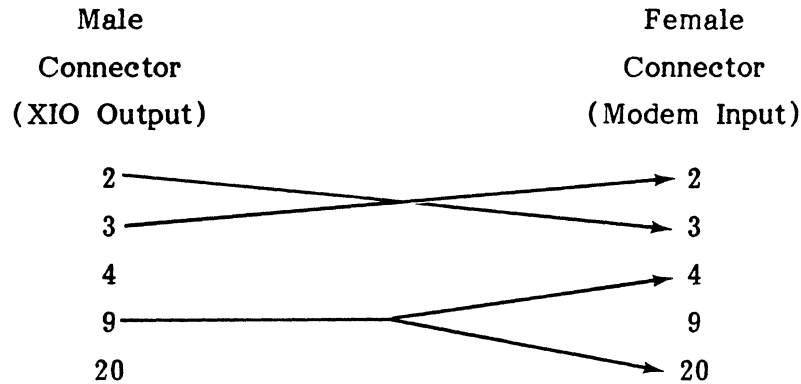
## 7. MASTER/SLAVE OPERATION, **[6]**

**7.01** This feature is designed to permit remote control of one ENCORE by another, or some other external device.

**7.02** Upon selection, user prompts permit operation with the ENCORE at either of two revision levels, below H and H or above. The revision level of your machine can be determined by holding down the **[SHIFT][7]** keys during power-up. To establish the master/slave link, each ENCORE is connected to the data circuit via the XIO port, which is RS-232 compatible. Since the XIO port is permanently configured as DCE and since each ENCORE must simulate DTE communicating through modems, the XIO port must be reconfigured as DTE to operate in the master/slave mode. The DTE configuration is provided by means of a special cable which must be constructed by the operator. The cable is shown schematically in Figure 5-24.

**NOTE:** If the ENCORE enters the Level-3 Command Mode, double check XIO connectors and then type D RUN **[RETURN]** to display the Master Directory once again.

**MASTER/SLAVE OPERATION**



**Fig. 5-24 XIO DCE/DTE Cable**

**7.03** Regardless of the program being executed, any ENCORE will assume the slave condition whenever \*DC2 appears at pin 2 of the XIO port. In addition, signals simulating control cluster keystrokes must also be preceded by an asterisk (\*). If an asterisk is to be input it must appear twice (\*\*). Finally, a \*DC3 at pin 2 instructs the ENCORE to exit the slave mode.

**7.04** All ASCII characters (00 through 7F) and most special function keystrokes received by the slave ENCORE are interpreted in exactly the same manner as they would be if input from the keyboard. The HEX equivalents of special function keys and unique master/slave commands are shown in Table 5-11.










**TABLE 5-11**

**MASTER/SLAVE SPECIAL FUNCTION KEYS**




(UNSHIFT) HEX CHARACTER	KEY	(SHIFT) HEX CHARACTER
80	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     UC LC                 </div>	C0
81	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     HEX                 </div>	C1
82	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     RWD                 </div>	C2
83 <sup>1</sup>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     STOP                 </div>	C3
84	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     ↑                 </div>	C4
85	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     →                 </div>	C5
86	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     UDF                 </div>	C6
87	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     CMD                 </div>	C7
88 <sup>2</sup>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">                     CONT                 </div>	C8

TABLE 5-11

MASTER/SLAVE SPECIAL FUNCTION KEYS (Cont'd)

(UNSHIFT) HEX CHARACTER	KEY	(SHIFT) HEX CHARACTER
89		C9
8A		CA
8B		CB
8C		CC
8D <sup>3</sup>		CD
8E		CE
8F		CF
90		D0
91		D1

NOTES:

1.  If this key is depressed while the slave is transmitting the contents of the CRT, transmission will stop. If depressed at any other time, the Master ENCORE will present the operator with a description of the STOP, CONT, and CLEAR Master/Slave commands.
2.  This command instructs the slave ENCORE to transmit the contents of the screen to the Master.
3.  This command instructs the remote ENCORE to exit the Master/Slave Mode.

8. USER DEFINED LIBRARY, 

**8.01** Item 7 in the Level-1 Directory permits user selection of any one of eight custom programs written by the user or by DIGITECH for the user. If custom programs have not been prepared, demonstration programs are supplied instead. There is no attempt made here to define these programs because they will vary from unit to unit. To examine the programs supplied, place the desired program in the source buffer (EDIT Mode) and LIST or DUMP as required.

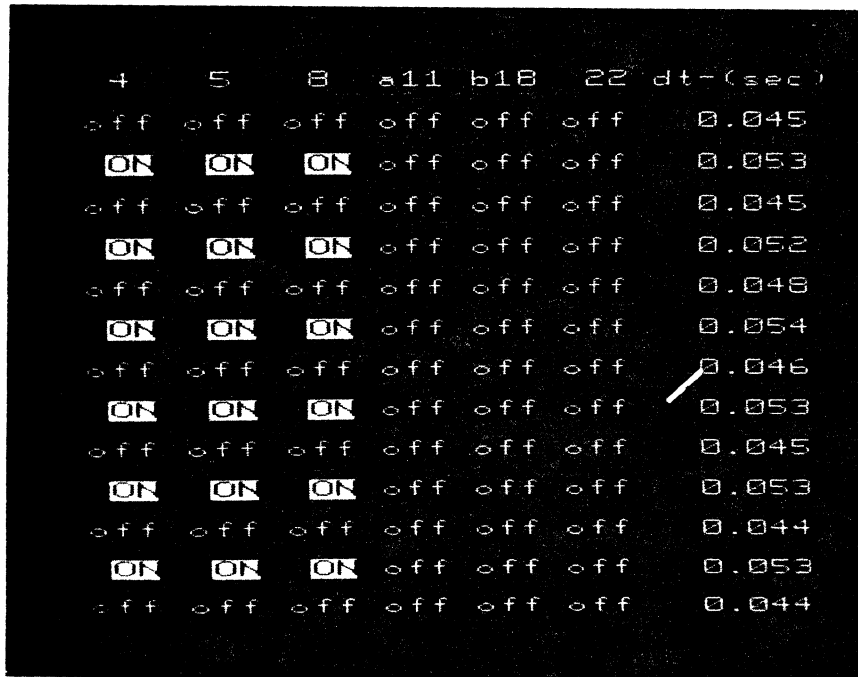
## USER DEFINED LIBRARY, EIA TIMING MONITOR, MASTER DIRECTORY

---

### 9. EIA TIMING MONITOR, 8

---

**9.01** This program is item 8 in the Level-1 Directory. It is designed to measure the time between a change in status of specified EIA-RS-232C leads. It is completely independent of timing or data signals and requires no change in IO parameters. As shown in Figure 5-25, the leads monitored are 4 (RTS), 5 (CTS), 8 (RLSD), 11 (A), 18 (B), and 22 (Ring indicator). The current status of the lead is shown on the bottom line of the display. The time between changes in the status of the line is shown in seconds to the right of the ON or OFF status. To exit this program and return to the Level-1 Directory, strike the CMD key.



4	5	8	a11	b18	22	dt-(sec)
off	off	off	off	off	off	0.045
ON	ON	ON	off	off	off	0.053
off	off	off	off	off	off	0.045
ON	ON	ON	off	off	off	0.052
off	off	off	off	off	off	0.048
ON	ON	ON	off	off	off	0.054
off	off	off	off	off	off	0.046
ON	ON	ON	off	off	off	0.053
off	off	off	off	off	off	0.045
ON	ON	ON	off	off	off	0.053
off	off	off	off	off	off	0.044
ON	ON	ON	off	off	off	0.053
off	off	off	off	off	off	0.044

**Fig. 5-25 EIA Timing Monitor**

---

### 10. MASTER DIRECTORY, 9

---

**10.01** The selection of item 9 from the Level-1 Directory instructs the ENCORE to display the Master Directory. This provides the user with an escape from Level-1 and offers him the opportunity to once again enter any desired level of operation.



## CHAPTER 6

### LEVEL-2 OPERATION AND PROGRAMMING

#### 1. GENERAL

**1.01** This chapter of the manual is devoted to a description of the DICOL/Level-2 operating system. It details the enhancements to both the Level-2 Mode Commands and the DICOL Instruction Set. In addition to these enhancements, each item in the Level-2 Menu is discussed in detail, and is provided with typical operating procedures where appropriate. Additional information is found in the PACER-103 Technical Manual, Pub. No. 810-00066.

**1.02** Level-2 is a powerful menu driven operating system. When combined with disc storage, it offers the DICOL programmer far greater flexibility and ease of operation than previously obtained through use of the PACER alone. Programs are quickly accessed and may be transferred to or from a PACER by simple menu selection. Level-2 modes of operation are nearly identical to those of the PACER.

- The Execute Mode runs the DICOL program currently residing in memory.
- The Program Mode allows the operator to enter and edit programs using basically the same techniques, instructions, and edit commands developed for the PACER.
- The IO Mode permits operator selection and entry of IO parameters.
- The Display Mode provides for off-line analysis of captured data which is greatly enhanced by the high volume of data which can be displayed on the ENCORE's 7 inch CRT.

**1.03** Further enhancements directly attributed to the power of the ENCORE and its menu driven Level-2 operating system include:

- The ability to output the entire DICOL menu to an external printer (via ENCORE's XIO port).
- The ability to transfer the entire contents of the PACER program memory to the Level-2 PSC ( Pacer Source Code) file (via XIO port).
- The ability to quickly load the PSC file with DICOL/Level-2 programs stored on disc.
- The ability to transfer the entire contents of the Level-2 PSC file to the PACER (via XIO port).
- The ability to store the entire contents of the Level-2 PSC file on the ENCORE's 5 1/4 inch disc.

**1.04** Compatibility is not limited to the modes of operation, commands, and instructions. It has been extended to include the CRT display as well. During the Execute Mode, for example, the ENCORE display includes information designed to simulate the PACER's Operating Status and Secondary displays. The Operating Status is displayed as follows:



- A = ALL, combining the functions of both SIM DTE and SIM DCE.
- B = BOTH, send and receive leads (2 and 3) are bridged.
- M = MODEM, simulating Data Communications Equipment (input send data on via pin 2 and output receive data via pin 3).
- T = TERMINAL, simulating Data Terminal Equipment (output send data via pin 2, input received data via pin 3).

1.05 The Secondary Display indicates program steps as they are executed. The familiar "AA" indicating BERT or CERT synchronization is appropriately displayed directly beneath the program step numbers.

1.06 From the user's stand-point, control of the interface (front ends) is similar to that experienced when using the PACER and easier though not as flexible as provided in the ENCORE's Level-3 operating system. Program control of the interface employs the same DICOL instructions used with the PACER but includes some benefits inherent with the design of the ENCORE.

## 2. OPERATION

2.01 For the purpose of this discussion, it is assumed that the user has read Chapter 4, is familiar with disc operation, and is capable of displaying the Level-2 Directory, Figure 6-1. The information that follows details each item in the directory in the order of appearance.

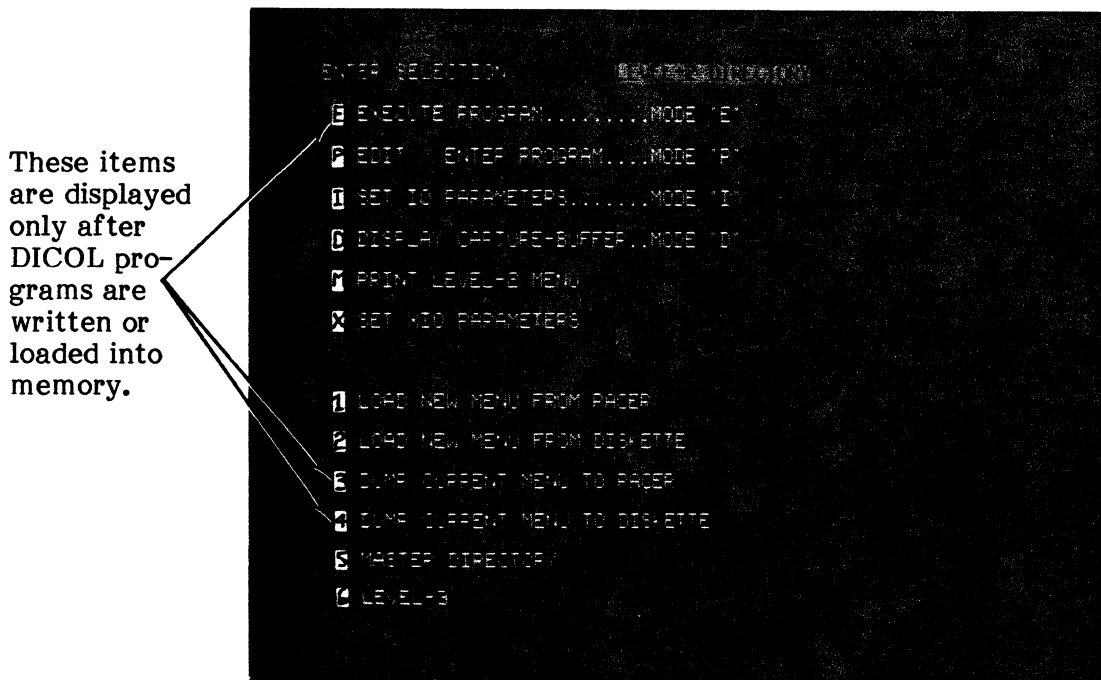


Fig. 6-1 Level-2 Directory

---

### 3. EXECUTE PROGRAM . . . MODE "E" ENQ E

---

**3.01** The Level-2 Execute Mode is selected from the Level-2 Directory by striking the ENQ  
E key, but only if the PSC (Pacer Source Code) file is in memory (see paragraph 4.01). Once selected, the system program, LVL2, executes the current DICOL program one line at a time, in numerical order, from beginning to end. Execution may be prematurely terminated by striking the CMD key once. Striking it again returns the Level-2 Directory. The ^  
P key is used to enter one-bit-character errors in the data stream while a program is running. Counting these errors in a BERT or CERT loopback helps to verify circuit continuity.

#### Typical Execute Procedure

**3.02** This procedure is designed to provide the user with step-by-step instructions for executing a DICOL/Level-2 program. This procedure is simplified when IO parameters are known to be correct or when the program contains an "01" (auto IO) instruction.

STEP	PROCEDURE
1.	From the Level-2 Directory, select Mode I by striking the <span style="border: 1px solid black; padding: 2px;">HT I</span> key. Then examine the IO parameters and change if necessary (see paragraph 5.01).  <b>Note:</b> If the program to be executed includes a "01" instruction, IO Mode parameters change automatically upon execution.
2.	Strike the <span style="border: 1px solid black; padding: 2px;">CMD</span> key after the correct IO parameters are established and note that the Level-2 Directory is displayed once again.
3.	From the directory, select Mode P by striking the <span style="border: 1px solid black; padding: 2px;">^ P</span> key. Then select the program to be executed using the <span style="border: 1px solid black; padding: 2px;">SHIFT ↑</span> keys. If there are no programs in the PSC file, enter a program manually as described in paragraph 4.02 or load programs as described in paragraph 9.01 or 10.01.  <b>Note:</b> The <span style="border: 1px solid black; padding: 2px;">SHIFT</span> key must be held down as the <span style="border: 1px solid black; padding: 2px;">↑</span> key is struck.
4.	Return to the Level-2 Directory by striking the <span style="border: 1px solid black; padding: 2px;">CMD</span> key.
5.	Now execute the selected program by striking the <span style="border: 1px solid black; padding: 2px;">ENQ E</span> key.

---



---

### 4. EDIT/ENTER PROGRAM . . . MODE "P" DLE P

---

**4.01** The Level-2 Program Mode is selected from the Level-2 Directory by striking the DLE  
P key. At this point, the system program "P MODE" is executed. This program allows the Level-2 operator to write and edit programs using the DICOL instruction set and edit commands in a manner similar to that used with DIGITECH's PACER-103. The display that follows Program Mode entry shows the contents of the PSC file. This will normally include previously loaded programs. Typical displays are shown in Figures 4-2 and 4-3.

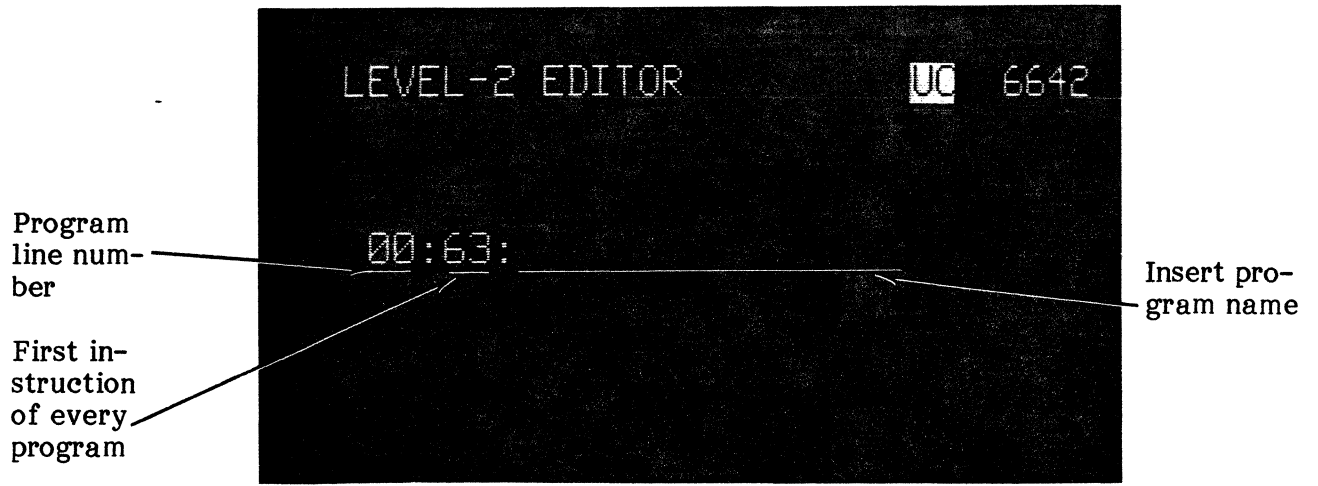


Fig. 6-2 Program Mode Display, Menu Not Loaded

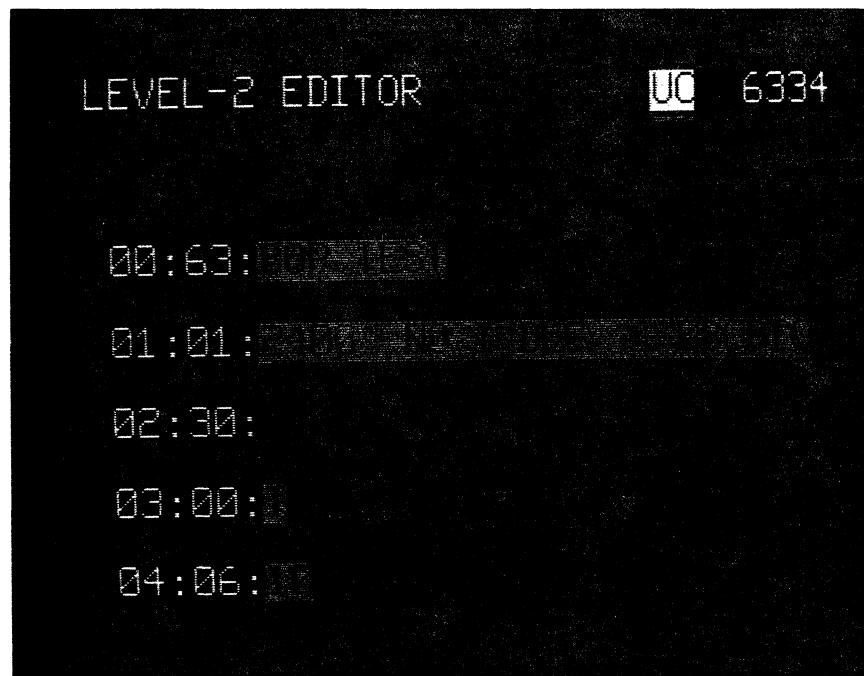


























Fig. 6-3 Program Mode Display, Menu Loaded


**4.02** You will notice that each program line appears in exactly the same format as it would on the PACER-103 except that instruction arguments are highlighted. You should also note that the top line of the display includes a case shift mode indicator (UC), that disappears when the  key is depressed (lower case selected). It also includes a number that indicates the amount of remaining program memory.

**Program Mode Commands**

**4.03** The Level-2 Program Mode Commands differ slightly from those of the PACER due to differences in the keycap legends and in the arrangement of keys in the auxiliary keypad. Both PACER and equivalent ENCORE commands are shown in Table 6-1.

**TABLE 6-1  
PACER/ENCORE PROGRAM MODE COMMANDS**

ITEM	PACER	ENCORE	FUNCTION
1.			Advance to next program.
2.			Insert line above current line.
3.			Go ahead one line.
4.			Go back one line.
5.			Character delete.
6.			Enter/Exit HEX Mode.
7.			Go to program start.
8.			Go to program end.
9.			Erase line.
10.			Erase program.
11.			Invert parity.
12.			Enable/Disable upper-case.


**NOTES:** Item #11 above is necessary because of the absence of an auxiliary shift-key on the ENCORE. On the ENCORE, the  key will turn the parity invert functions on or off. Whenever the invert function is on, an indication will appear on the top line of the display.

Item #12 above is an added ENCORE function which allows only upper case alphas when enabled. Whenever the upper case function is on, there will be a "UC" displayed on the top line of the CRT. This function is especially useful since there is only one shift key on the ENCORE.









**Typical Program Entry Procedure**

**4.04** DICOL/Level-2 programs are normally written in the Level-2 Program Mode. Although they may also be written in the Level-3 Edit Mode, the examples given here deal strictly with Level-2 operation.

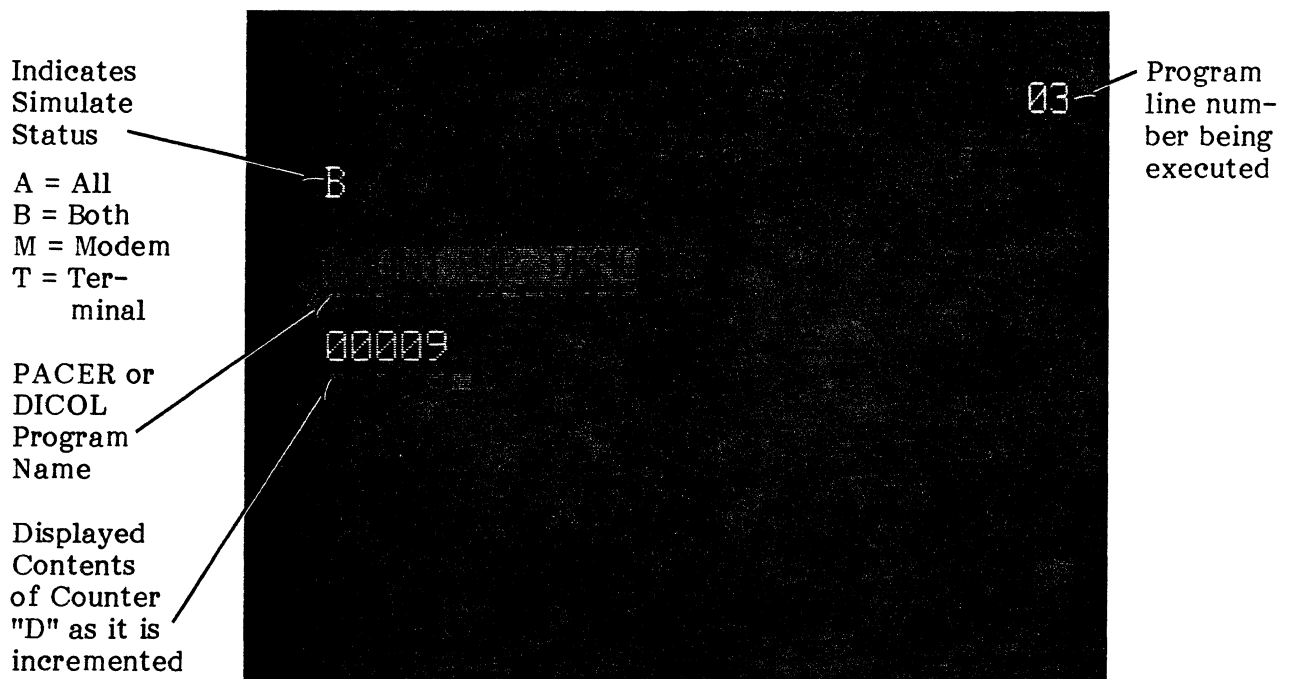
**Programming Hints**

- Momentarily depress the SPACE BAR to terminate the DICOL instruction.
- Momentarily depress the  key to terminate the argument.

**4.05** The following procedure is designed to demonstrate the method by which a DICOL/Level-2 program is written. Additional information covering the selection of IO parameters is found in paragraph 5.01.

STEP	PROCEDURE																												
1.	From the Level-2 Directory, momentarily strike the  key (Program Mode) and note that the Program Mode Display (LEVEL-2 EDITOR) appears in a format similar to that shown in Figures 6-2 and 6-3.																												
2.	If the Program Mode display is similar to that shown in Figure 6-3, you must either erase the existing program(s) by striking the  key twice for each program to be erased or move to the end of the program by holding the  key down while striking the  key. In either case, the first instruction must be "63", followed by the program name. If the display appears as shown in Figure 6-2, simply begin the program by typing the program name.																												
3.	Enter the following program. Remember, strike the SPACE BAR for automatic colon entry after the instruction number, and the  key after the argument.																												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Line#</th> <th style="text-align: left;">Instruction</th> <th style="text-align: left;">Argument</th> <th style="text-align: left;">Comments</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>: 63 :</td> <td>COUNT UP DEMO</td> <td>Program name</td> </tr> <tr> <td>01</td> <td>: 30 :</td> <td></td> <td>Clear display</td> </tr> <tr> <td>02</td> <td>: 55 :</td> <td>D</td> <td>Display the contents of counter "D"</td> </tr> <tr> <td>03</td> <td>: 50 :</td> <td>100</td> <td>Wait 100 ms</td> </tr> <tr> <td>04</td> <td>: 53 :</td> <td>D</td> <td>Increment counter "D" by 1</td> </tr> <tr> <td>05</td> <td>: 04 :</td> <td>-4</td> <td>Jump back 4 steps</td> </tr> </tbody> </table>	Line#	Instruction	Argument	Comments	00	: 63 :	COUNT UP DEMO	Program name	01	: 30 :		Clear display	02	: 55 :	D	Display the contents of counter "D"	03	: 50 :	100	Wait 100 ms	04	: 53 :	D	Increment counter "D" by 1	05	: 04 :	-4	Jump back 4 steps
Line#	Instruction	Argument	Comments																										
00	: 63 :	COUNT UP DEMO	Program name																										
01	: 30 :		Clear display																										
02	: 55 :	D	Display the contents of counter "D"																										
03	: 50 :	100	Wait 100 ms																										
04	: 53 :	D	Increment counter "D" by 1																										
05	: 04 :	-4	Jump back 4 steps																										
4.	Strike the  key and note that 5 lines of the program are displayed. Use the  and  keys to scroll through the program and check for errors. If an error is found, use the keystrokes defined in Table 6-1 to edit the program.																												

STEP	PROCEDURE
5.	Strike the <b>CMD</b> key and note that the ENCORE displays the Level-2 Directory. It is good practice, at this point, to save the program on disc by selecting item <b>3</b> from the menu (see paragraph 12.01). This will prevent accidental loss of the program if power is turned off.
6.	If the program were designed to input or output data, the IO Mode would have to be entered and IO parameters established (see paragraph 5.01). Since this example does not require input or output via the interface, strike the <b>END</b> key to execute the program and note a display similar to that shown in Figure 6-4.



**Fig. 6-4 Typical Execute Display**

7. Strike the **CMD** key twice to exit this program and note that the Level-2 Directory is displayed once again.

**Instruction Set**

**4.06** The DICOL/Level-2 instruction set consists of high order language instructions that may or may not require an argument. An instruction specifies an operation to be performed. An argument provides the data/information require by the instruction. The information in the following table differs from that provided in the PACER manual because the ENCORE makes certain enhancements possible. Whenever the explanation of an instruction is changed as a result of these enhancements, an asterisk (\*) will appear to the right of the instruction number. The time required to execute each instruction is shown in parenthesis and given in microseconds (us), e.g. (execute = 125 us + 65 us/char.). Where the execute time is not given, the maximum speed is shown, e.g. (60 kbps).

**NOTE:** Each instruction entered must be followed by a space and each argument by the  key. Colons are entered automatically.

**TABLE 6-2**

**DICOL/LEVEL-2 INSTRUCTION SET**

KEYSTROKE	OPERATION
-----------	-----------

**INITIALIZE INSTRUCTIONS**



SELECT TEST MODE. This instruction is normally one of the first instructions in a program. It allows the PACER to bridge both inputs (lead 2 and 3), to simulate DCE by receiving on lead 2 only or to simulate DTE and receive on lead 3 only (initiates a sync search on the appropriate lead). If the 00 instruction is not part of the program, the PACER automatically assumes a bridging input. When the instruction is used, one of the following arguments is required:

Instruction	Argument	Comment
00	: B	Both inputs are bridged (leads 2 and 3). The same condition exists when the 00 instruction is deleted or no argument is entered.
00	: M	This argument allows the PACER to simulate DCE and monitor data received on lead 2 while sending on lead 3.
00	: T	This argument allows the PACER to simulate DTE and receive data on lead 3 while sending on lead 2.
00	: A	This instruction is used primarily for self-testing purposes. It combines the functions of both the SIM DTE and SIM DCE into a single test mode (A = M + T).

TABLE 6-2

DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION
-----------	-----------

INITIALIZE INSTRUCTIONS (Cont'd)

**Note:** In practice, to conserve program memory, delete the 00 instruction to bridge both inputs.

If the program directs the PACER to assume an active simulate state (instruction "00":T or "00":M), the PACER normally takes the place of the CPU (DTE) or modem (DCE), but that component must be disconnected from the circuit. When simulating DTE, the PACER automatically inputs data via lead 3, outputs data via lead 2, and turns on lead 20 (Data Terminal Ready). If instructed to simulate DCE, the PACER inputs data via lead 2, outputs data via lead 3, turns on lead 6, and supplies send and receive clock via leads 15 and 17, respectively.

In addition, the PACER may be programmed to control lead 4 and T when simulating DTE and leads 5, 8, M, and 21 when simulating DCE (execute = 32 us).

1

**STORE IO PARAMETERS.** When this instruction is entered, all network parameters listed during the IO Mode are copied in the argument field. In addition, this instruction presets the memory for over-write on buffer full (capture memory), clears the A and B counters, clears the parity error flag, and turns on the data display. To change IO parameters when writing a program, enter the IO Mode and make the necessary changes there. Then return to the Program Mode, delete the "01" instruction and re-enter. This will change the "01" parameters to agree with the current IO parameters. The "01" instruction is not to be used with the BERT instruction number 25 (execute = 4-21 ms).

2

\* **SELECT BCS TYPE.** This instruction permits selection of the proper redundancy check scheme for use with synchronous and asynchronous data.

LRC-8 (ASCII)	=	SHIFT CR M	(Parity corrected to agree with IO selection)
LRC-8	=	{ GS }	(Actual accumulation no parity correction)
CRCC-16 (EBCDIC)	=	SHIFT BS H	
CRCC-CCITT (SDLC)	=	SO N	



TABLE 6-2

DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION
<b>INITIALIZE INSTRUCTIONS (Cont'd)</b>	
	An upper case argument character resets the BCC accumulation to all zero's (BCS). The lower case argument presets the accumulation to all one's (SDLC). This instruction is used only once during a program and prior to the first "03" instruction (execute = 10 ms).
# 3	<u>START BCS.</u> This is an instruction to reset/begin the <u>Block Check Sequence</u> accumulation for data received on RS-232 lead 2 or 3. This is accomplished by entering a "2" or "3" argument (execute = 50 ms).
5 4	* <u>JUMP.</u> This is an instruction to jump forward or backward to some other instruction in the program. The number of steps required by the jump instruction are entered as the argument (PACER = +127, ENCORE = +319). A backward jump must be preceded by a "-" (minus). A forward jump may be preceded by a "+" (plus) for clarity, but need not be (execute = 34 us). GOTO and GOSUB (+319) may be used in place of "04" when the DICOL program is executed in Level-2.
% 5	<u>RETURN TO MODE SELECT.</u> Upon execution, this instruction causes the PACER to exit the EXECUTE Mode and enter the MODE SELECT state (no argument required). To assure compile, this instruction, a "04", or a "09" instruction should end every program.
& 6	<u>SUBROUTINE CALL.</u> This is an instruction to execute a subroutine. The argument is any number of steps up to +127 (to the first step of the subroutine). When the subroutine is completed, the program will return to the step that follows the "06" instruction (execute = 32 us).
^ 7	<u>RESET LOGGING TO BEGINNING OF BUFFER.</u> Upon execution, the memory is cleared and all incoming data will be logged in capture memory starting at the first memory location.
( 8	<u>RETURN TO MODE SELECT ON BUFFER FULL.</u> Upon execution, this instruction causes the PACER to assume the MODE SELECT state when the capture memory is full (no argument required). This instruction can be used only once during a program and must follow the "01" instruction.
) 9	<u>RETURN FROM SUBROUTINE.</u> This is an instruction to return from a subroutine to the program step following the "06" call. The "09" instruction must always be preceded by a "06" instruction at some point in the program (execute = 32 ms).

TABLE 6-2

## DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)



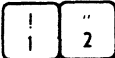
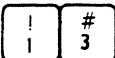
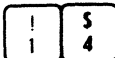
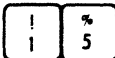
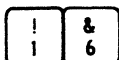
KEYSTROKE	OPERATION
<b>RECEIVE INSTRUCTIONS</b>	
	<b>TURN ON BOTH RECEIVERS.</b> This instruction turns on the PACER receivers for RS-232 leads 2 and 3, and initiates a sync search. The PACER normally assumes this instruction unless directed otherwise. A "00:T" or "M" will automatically turn off one of the receivers. Used following 00:M or 00:T to receive data sent as well as received (execute = 20 us).
	<b>TURN OFF RECEIVER 2 OR 3.</b> Upon execution, this instruction turns off receiver 2 or 3. (The number "2" or "3" is entered as the argument). This instruction is also used to drop sync by turning off the active receiver. Turning off one receiver automatically turns on the other and initiates a new sync search (execute = 20 ms).
	<b>TRAP SEQUENCE.</b> This instruction causes the PACER to receive and display incoming characters, one character at a time, in the left-most position of the Primary Display window. When incoming characters match the argument characters, they are displayed in the right-most position of the window. Each match character displayed shifts the previous match character one position to the left. If any character in the sequence does not match, the search for a complete sequence begins when the first argument character is received again. When the complete sequence has been received, the PACER proceeds to the next instruction. The Trap Sequence is not displayed or logged. The sequence may be any one to 25 characters (60 kbps).
	<b>RECEIVE AND LOG THROUGH SEQUENCE.</b> This instruction causes the PACER to receive and log all incoming data up to and including the character(s) entered in the argument field. The sequence may be any one to 25 characters. If the instruction is not followed by an argument, the PACER will receive and log indefinitely (automatic overwrite) or until the "MODE" key is depressed. If the "08" instruction is used, the PACER will receive and log until the buffer is filled. The speed of operation is increased when preceded by a "37" instruction (60 kbps).
	* <b>RECEIVE AND LOG "n" CHARACTERS.</b> This instruction causes the PACER to receive and log "n" number of characters. The number entered in the argument field can be any number from one to 999 for operation with the PACER or from one to 65K for Level-2 operation (60 Kbps, "37" instruction).
	<b>RECEIVE AND LOG THROUGH ANY ONE OF UP TO FIVE CHARACTERS.</b> Upon execution, the PACER will receive and log all incoming data until one of the five characters entered in the argument field is recognized (30 kbps).

TABLE 6-2

DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION
-----------	-----------

RECEIVE INSTRUCTIONS (Cont'd)



CRC COMPUTATION IN TRANSPARENT BSC. This instruction (Optional) is designed to permit update of the Block Check Sequence on a character-by-character basis deleting unwanted characters from the accumulation. To accomplish this, an additional instruction is required and the current block check instruction is altered. In the following example the new instruction ("16") is used in conjunction with instruction "14" and "18" to perform the block check accumulation on all characters between SOH and ETX excluding any specified characters in the message (e.g. SYN DLE). Once the accumulation is made, the altered instruction ("17") simply checks the accumulation for all "0's" or all "1's" as determined by the "02" instruction.

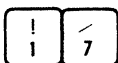
**Note:** Once this option is supplied, all BCS accumulations must be made in a manner similar to that shown in the following example.

Instruction	Code	Argument	Bytes	Comments
<b>PROGRAM BEGINS</b>				
13	:	<u>SOH</u>	3	Receive and log through SOH.
03	:	2	3	Start BCS accumulation on pin 2.
14	:	1	3	Receive and log one character.
18	:	<u>SYN</u>	3	Skip next instruction unless last logged character equals <u>SYN</u> .
				<b>Note:</b> Select any desired argument character.
04	:	2	4	Jump back two instructions.
16	:	2	3	Update BCS accumulation on pin 2.

**TABLE 6-2**  
**DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)**

KEYSTROKE	OPERATION	
<i>RECEIVE INSTRUCTIONS (Cont'd)</i>		
18 : <u>ETX</u>	3	Skip next instruction unless last logged character equals <u>ETX</u> .
04 : 2	3	Jump ahead two instructions.
04 : -6	4	Jump back six instructions to receive and log one character.
14 : 1	3	Receive and log one character (first BCC received).
16 : 2	3	Update BCS accumulation on pin 2.
14 : 1	3	Receive and log one character (second BCC received).
16 : 2	3	Update BCS accumulation on pin 2.
17 : 2	3	Check BCS.

**PROGRAM  
CONTINUES**



RECEIVE NEXT SEQUENCE AS BCS AND SKIP NEXT INSTRUCTION IF VALID. This instruction causes the PACER to read the next incoming character or two character sequence as the Block Check Sequence resulting from longitudinal or cyclic redundancy checks. If the BCS is correct, the next instruction is skipped. If the BCS is incorrect, the program is advanced to the next instruction. The appropriate receiver (2 or 3) is selected by entering a "2" or "3" in the argument field. The BCS is neither displayed nor logged into the capture memory. A "03" instruction must precede with same receiver 2 or 3.

**Note:** When the PACER is equipped with the E014 option, this instruction simply checks the accumulation (see instruction "16"). ("N" - Appears in the power-up display). Option E014 (BCS/Transparent BSC) provides for cyclic or longitudinal redundancy check in a transparent BSC environment.

TABLE 6-2

## DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION				
<b>RECEIVE INSTRUCTIONS (Cont'd)</b>					
<table border="1"> <tr> <td> </td> <td>(</td> </tr> <tr> <td> </td> <td>8</td> </tr> </table>		(		8	<p>* <u>SKIP NEXT INSTRUCTION UNLESS ARGUMENT EQUALS LAST LOGGED CHARACTER(S).</u> This instruction normally follows a receive and log instruction. If the last character(s) received is the same as the character(s) in the argument field, the program will advance to the next instruction. Otherwise, the next instruction is skipped. The sequence may be any one to 25 characters (execute = 67 us + 27 us/char).</p>
	(				
	8				
<table border="1"> <tr> <td> </td> <td>)</td> </tr> <tr> <td> </td> <td>9</td> </tr> </table>		)		9	<p><u>RECEIVE AND LOG ANY CHARACTERS EXCEPT ARGUMENT.</u> Receipt of any argument character prevents the program from being advanced until a single non-argument character is received. Receipt of any other character will advance the program. Any non-argument character received is logged. This instruction is typically used for editing purposes. For example, assume that a test message is being received and it contains unwanted delete characters. The delete characters can be edited out of the text by combining instructions "19": <u>DEL</u> and "04": 1. The "19" instruction allows the PACER to log any character but <u>DEL</u>. The "04":-1 is then used to repeat the "19" instruction so that all received characters, except <u>DEL</u>'s are logged (30 kbps).</p> <p><b>Note:</b> The "19" instruction is especially useful in the transparent mode on synchronous circuits to prevent logging an idle line condition.</p>
	)				
	9				
<table border="1"> <tr> <td>"</td> <td>2</td> </tr> <tr> <td></td> <td>∅</td> </tr> </table>	"	2		∅	<p><u>TRANSMIT CHARACTER(S).</u> This instruction allows the PACER to transmit from one to 25 characters as entered in the argument field. Use of the argument chain (instruction "38") extends the test message beyond 25 characters (60 kbps). The characters can not be displayed or logged into the capture memory.</p>
"	2				
	∅				
<table border="1"> <tr> <td>"</td> <td> </td> </tr> <tr> <td>2</td> <td>1</td> </tr> </table>	"		2	1	<p><u>TRANSMIT BCS CHARACTER(S).</u> This is an instruction to transmit the next character(s) as the block check sequence (LRC-8) or (CRC-CCITT, CRC-16). This instruction does not require an argument. At some point in the program, this instruction must be preceded by "02" and "03" instructions. Special programming must be used to display all data up to and including the BCS (60 kbps).</p>
"					
2	1				
<table border="1"> <tr> <td>"</td> <td>"</td> </tr> <tr> <td>2</td> <td>2</td> </tr> </table>	"	"	2	2	<p><u>TRANSMIT CHARACTERS, RECEIVE AND LOG SIMULTANEOUSLY.</u> This is an instruction to operate full-duplex. Upon execution, the PACER will receive and log incoming data while simultaneously transmitting from one to 25 characters as entered in the argument field. To log transmitted data, the program must include a "10" instruction at some point preceding the "22" instruction (30 kbps). Use of the argument chain instruction "38" automatically extends the argument field beyond 25 characters.</p>
"	"				
2	2				

TABLE 6-2

## DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION				
<b>RECEIVE INSTRUCTIONS (Cont'd)</b>					
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">"</td> <td style="text-align: center;">#</td> </tr> <tr> <td style="text-align: center;">2</td> <td style="text-align: center;">3</td> </tr> </table>	"	#	2	3	<p>* <u>CHARACTER ERROR RATE TEST, SKIP ON ERROR.</u> This is an instruction to perform a character error rate test for asynchronous testing. The test message of up to 25 characters is entered in the argument field. Each unerrored block received increments an internal block counter by one. When 10 000 message blocks are received, the program is advanced to the next instruction. If an error is detected the PACER immediately skips the next instruction. Use of the argument chain (instruction "38") extends the test message beyond 25 characters. For operation with the PACER, program logic, after an error, must be completed within one character time to prevent loss of data. In Level-2, the 128 character buffer prevents the loss of data (60 kbps asynchronous). "AA" appears in the Secondary Display when synchronization is attained.</p>
"	#				
2	3				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">"</td> <td style="text-align: center;">S</td> </tr> <tr> <td style="text-align: center;">2</td> <td style="text-align: center;">4</td> </tr> </table>	"	S	2	4	<p>* <u>ENTER CONVERSATION MODE, EXIT ON ARGUMENT.</u> This instruction allows the operator to converse with a remote station through the keyboard. When the character entered in the argument field is typed, the program is advanced to the next instruction. The conversation is not logged in buffer memory. When selecting the argument character, use a key that is easily reached and would not normally appear in the conversation unless a single key operator input is desired to advance in the program. The argument character is not transmitted. When used in Level-2, send and received data are displayed as normal and reverse video, respectively.</p>
"	S				
2	4				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">"</td> <td style="text-align: center;">%</td> </tr> <tr> <td style="text-align: center;">2</td> <td style="text-align: center;">5</td> </tr> </table>	"	%	2	5	<p><u>BLOCK ERROR RATE TEST, SKIP ON ERROR.</u> This is an instruction to perform a block error rate test using a 63 or 511 bit pattern for synchronous testing. The desired pattern is selected by entering one of the following argument characters:</p>
"	%				
2	5				

<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">&amp;</td> </tr> <tr> <td style="text-align: center;">6</td> </tr> </table>	&	6	=	63 bit pattern
&				
6				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">%</td> </tr> <tr> <td style="text-align: center;">5</td> </tr> </table>	%	5	=	511 bit pattern
%				
5				

TABLE 6-2

DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION
-----------	-----------

**RECEIVE INSTRUCTIONS (Cont'd)**

To ensure synchronization, always precede the "25" instruction with one of the instructions that initiates a sync search ("00", "10", "11"). Synchronization should then occur after one complete message pattern has been transmitted and received. A sync search instruction should also precede the BERT instruction each time that it is executed. "AA" appears in the Secondary Display when synchronization is attained. The complete bit error rate test is performed on a total of 10 000 message blocks each consisting of 1 000 bits. A message block received without errors, increments an internal block counter by one. If an error is detected, the PACER immediately skips the next program instruction. The program is advanced to the next step after 10 000 valid message blocks are received. When the test is performed, IO parameters are automatically set as follows:

- Character length = 7BIT
- Parity = NON
- Transmission Mode = SYNC
- Sync Characters = Z (511 bit pattern) or 7<sub>B</sub>V (63 bit pattern)
- Transmission Code = ASCII

**Notes:** 1) The "B" Subscript = blinking character. 2) This instruction will not be properly executed if the PACER is connected to an asynchronous circuit, unless options for synchronous internal clock are included (20 kbps).

" 2	& 6
-----	-----

DISPLAY CONTENTS OF BLOCK COUNTER. This instruction causes the PACER to display the contents of the block counter. The block count accumulates unerrored blocks until the count of 10 000 is reached (no argument required, execute = 3.1 ms).

" 2	/ 7
-----	-----

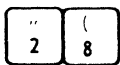
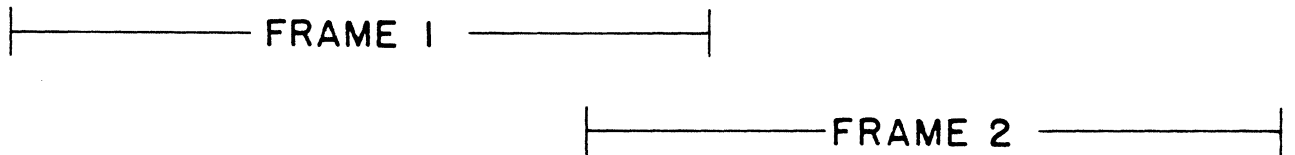
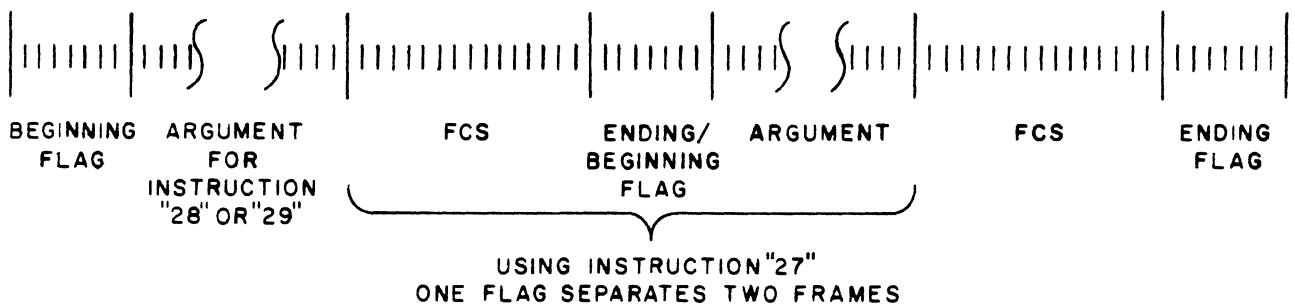
SDLC FRAME CHAIN (OPTION). This instruction serves the same function as a "38" instruction except that an FCS and flag are inserted automatically after each argument chain. In this case, each argument chain becomes an SDLC frame: sharing the end flag of the previous frame. To insert more than one flag between information fields, use the "20" or "22" instruction with argument characters appropriate for ASCII or EBCDIC. The "27" instruction is only used with another "27", "28", or "29" instruction.

TABLE 6-2  
 DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

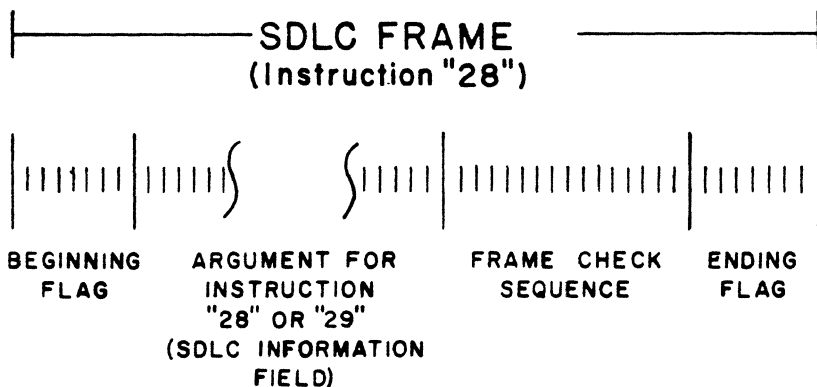
KEYSTROKE	OPERATION
-----------	-----------

**RECEIVE INSTRUCTIONS (Cont'd)**

**Note:** To compute the FCS, a "02" instruction with an "n" argument must precede the "27", "28", or "29" instruction. Any one to 25 characters may be used. A "38" instruction extends the argument field.



**SDLC FRAME TRANSMIT (OPTION).** This instruction is used to frame an SDLC information field (data block) with a beginning flag, an FCS, and an ending flag. The instruction argument is the SDLC information field. The PACER frames the information field for data transmission. Any one to 25 characters may be used. A "38" instruction extends the argument field.



AUTOMATIC FRAMING WHEN INSTRUCTION IS EXECUTED



TABLE 6-2

DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION
-----------	-----------

**RECEIVE INSTRUCTIONS (Cont'd)**

**Notes:** 1) Use instruction "27" to chain frames. Each "27" instruction will cause the previous block to be terminated by an FCS flag, and another block will be started. There will be only one flag byte between blocks when using the "27" instruction. A "27" instruction with no argument will cause only a flag to be transmitted. 2) To transmit the correct FCS, a "02" instruction with an "n" argument must be given during program initialization before any "28" or "29" instruction.

"	)
2	9

SDLC FRAME TRANSMIT CHARACTERS, RECEIVE AND LOG SIMULTANEOUSLY (OPTION). This is an instruction to operate full-duplex in SDLC. The argument for this instruction is framed and chained in the same manner as instruction "28". The difference between instructions "28" and "29" lies in the fact that the "29" instruction will allow the PACER to receive and log data while the argument is transmitted. A "10" instruction must precede this instruction to permit logging the data sent. (No flags are logged).

**MISCELLANEOUS INSTRUCTIONS**

#	)
3	ø

CLEAR DISPLAY. Upon execution, this instruction clears all data from the display window (execute = 100 us).

#	!
3	1

\* DISPLAY CHARACTERS. Upon execution, the argument appears in the display window. The argument will normally consist of up to 25 characters. The argument chain instruction ("38") must be used to extend the displayed argument to a full 32 characters (execute = 125 us + 65 us/char). When used in Level-2, characters appear underscored.

#	"
3	2

CLEAR PARITY ERROR. This instruction clears an internal parity detect flag (no argument required). Instructions "01" and "33" also clear the parity detect flag (execute = 14 us).

#	#
3	3

SKIP NEXT INSTRUCTION ON PARITY ERROR. If an error has occurred, the PACER will skip the next instruction and clear the parity detect flag. If parity error has not occurred, the program is advanced to the next instruction (no argument required, execute = 40 us).

#	\$
3	4

\* TURN ON AUDIO. This is an instruction to sound a 1000 Hz audible alarm for a period of not less than 0.25 seconds. When operating with the PACER, the tone remains on until the TONE OFF instruction ("35") is executed or until the PACER returns to MODE SEL (no argument required, execute = 17 us).

TABLE 6-2

## DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION
<b>MISCELLANEOUS INSTRUCTIONS (Cont'd)</b>	
# 3 % 5	* <u>TURN OFF AUDIO.</u> This instruction turns off the audible alarm. The minimum tone-on time is 0.25 seconds. In Level-2, the tone is automatically turned off after 0.25 seconds (no argument required, execute = 17 us).
# 3 & 6	<u>TURN ON DATA DISPLAY.</u> This instruction turns on the display for presentation of all data applied to receivers 2 and 3 as conditioned by instructions "10", "11", "22", "29", etc.. This condition is normally assumed until the TURN OFF DATA DISPLAY instruction ("37") is executed (no argument required, execute = 17 us).
# 3 / 7	<u>TURN OFF DATA DISPLAY.</u> This instruction prevents display of data received via EIA pins 2 and 3 and must be used in all programs operating at or above 50 kbps. Use instruction "36" to turn the display back on (no argument required, execute = 17 us).
# 3 ( 8	<u>ARGUMENT CHAIN.</u> This instruction automatically follows any instruction that requires more than the allotted 25 argument characters. Whenever this instruction is used, the length of the previous argument field is increased by 25 characters. No more than nine continuous "38" instructions may be entered in a single program (for a maximum of 255 characters).
# 3 ) 9	Reserved for future use.
<b>CONTROL LEAD AND COUNT INSTRUCTIONS</b>	
5 4 ∅	<u>TURN OFF ALL CONTROLLED RS-232 LEADS (4, 5, 8, T, M, AND 21) EXCEPT DTR (20) AND DSR (6).</u> This instruction turns off all RS-232 leads previously turned on with the exception of leads 6 and 20 (no argument required).
5 4 ! 1	<u>TURN ON RS-232 LEAD _____.</u> This is an instruction to turn on a single RS-232 lead. Only the following leads are controlled in the simulate modes: 4, 20, 11/A (SIM DTE), and 5, 6, 8, 22, and 18/B (SIM DCE). Leads 2, 3, 15, and 17 are automatically controlled (see instruction "00"). A "41 : 3" turns on all controlled DTE or DCE leads depending on the mode of operation. The lead number is entered in the argument field (execute = 17 us).
<b>Note:</b> Since the PACER is designed to control lead 21 instead of 22, the instruction "41 : 21" actually turns on lead 22 when executed in Level-2.	

TABLE 6-2

DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION				
<b>CONTROL LEAD AND COUNT INSTRUCTIONS (Cont'd)</b>					
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">"</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">2</td></tr> </table>	5	"	4	2	<p><u>TURN OFF RS-232 LEAD</u> . This instruction counters the "TURN ON" instruction (41). A "42 : 3" turns off controlled DTE or DCE leads depending on the mode of operation. The lead number is entered in the argument field (execute = 17 us).</p> <p><b>Note:</b> Since the PACER is designed to control lead 21 instead of 22, the instruction "42 : 21" actually turns off lead 22 when executed in Level-2.</p>
5	"				
4	2				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">#</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">3</td></tr> </table>	5	#	4	3	<p><u>SKIP NEXT INSTRUCTION IF RS-232 LEAD</u> IS ON. Upon execution, the PACER detects the state of the RS-232 lead entered in the argument field. If the lead is ON, the next program instruction is skipped. If the lead is OFF, the program is advanced to the next instruction (execute = 30 us).</p>
5	#				
4	3				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">\$</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">4</td></tr> </table>	5	\$	4	4	<p><u>SKIP NEXT INSTRUCTION IF RS-232 LEAD</u> IS OFF. Upon execution, the PACER detects the state of the RS-232 lead entered in the argument field. In this case, the next program instruction is skipped if the lead is OFF. If the lead is ON, the program is advanced to the next instruction (execute = 30 us).</p>
5	\$				
4	4				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">%</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">5</td></tr> </table>	5	%	4	5	<p><u>SET COUNTER "C" (OPTION)</u>. This is an instruction to set the internal "C" counter to any number from 0 to 9 999. The number is entered in the argument field (execute = 24 us).</p>
5	%				
4	5				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">&amp;</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">6</td></tr> </table>	5	&	4	6	<p><u>SET COUNTER "D" (OPTION)</u>. This instruction is similar to instruction "45" except that it presets a second four-digit counter (counter "D"). The number to which the counter is preset is entered in the argument field (execute = 24 us).</p>
5	&				
4	6				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">/</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">7</td></tr> </table>	5	/	4	7	<p>Reserved for future use.</p>
5	/				
4	7				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">(</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">8</td></tr> </table>	5	(	4	8	<p><u>DISPLAY HALF-DUPLEX (OPTION)</u>. This is an instruction to display data on a CRT in half-duplex fashion on 16 lines. Send data is displayed in white characters on black, while receive data is displayed in black characters on white. Send and receive data are displayed in sequence. For additional information see the PACERSCOPE Manual, Pub. No. 810-00073.</p>
5	(				
4	8				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td style="text-align: center;">5</td><td style="text-align: center;">)</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">9</td></tr> </table>	5	)	4	9	<p><u>DISPLAY FULL-DUPLEX (OPTION)</u>. This is an instruction to display data on a CRT in dual line, time correlated full-duplex fashion using while receive data is displayed in black characters on white. For additional information see the PACERSCOPE Manual, Pub. No. 810-00073.</p>
5	)				
4	9				

TABLE 6-2

## DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION				
<b>CONTROL LEAD AND COUNT INSTRUCTIONS (Cont'd)</b>					
<table border="1"><tr><td>%</td><td></td></tr><tr><td>5</td><td>∅</td></tr></table>	%		5	∅	* <u>WAIT</u> <u>MILLISECONDS.</u> This is an instruction to delay execution of the program from 1 to 9 999 milliseconds. Typically a delay should follow any display instruction to insure readability. The delay, in milliseconds, is entered in the argument. In Level-2, the delay may be entered from 1 to 65 535. (A zero argument results in a 65,535 ms delay).
%					
5	∅				
<table border="1"><tr><td>%</td><td> </td></tr><tr><td>5</td><td>1</td></tr></table>	%		5	1	<u>SET COUNTER A.</u> This is an instruction to preset an internal counter, (counter "A") to any number from 0 to 99. The number is entered in the argument field (execute = 24 us). The counter is automatically preset to 0 by a "01" instruction.
%					
5	1				
<table border="1"><tr><td>%</td><td>"</td></tr><tr><td>5</td><td>2</td></tr></table>	%	"	5	2	<u>SET COUNTER B.</u> This instruction is similar to "51" except that it presets a second counter, (counter "B"). The number to which the counter is preset is entered in the argument field (execute = 24 us). The counter is automatically preset to 0 by a "01" instruction.
%	"				
5	2				
<table border="1"><tr><td>%</td><td>#</td></tr><tr><td>5</td><td>3</td></tr></table>	%	#	5	3	<u>ADD ONE COUNT TO COUNTER.</u> This is an instruction to increment an internal counter by one. The proper counter is selected by entering any character "A" through "L" in the argument field (execute = 34 us).
%	#				
5	3				
<table border="1"><tr><td>%</td><td>\$</td></tr><tr><td>5</td><td>4</td></tr></table>	%	\$	5	4	<u>SUBTRACT ONE COUNT FROM COUNTER.</u> This is an instruction to decrement an internal counter by 1. The proper counter is selected by entering any character "A" through "L" in the argument field (execute = 34 us).
%	\$				
5	4				
<table border="1"><tr><td>%</td><td>%</td></tr><tr><td>5</td><td>5</td></tr></table>	%	%	5	5	<u>DISPLAY CONTENTS OF COUNTER.</u> This is an instruction to display the contents of one of the internal counters. The proper counter is selected by entering any character "A" through "L" in the argument field. In Level-2, the argument may be entered for counters A through L (execute = 2.8 ms).
%	%				
5	5				
<table border="1"><tr><td>%</td><td>&amp;</td></tr><tr><td>5</td><td>6</td></tr></table>	%	&	5	6	<u>SKIP NEXT INSTRUCTION IF COUNTER EQUALS "00".</u> This is an instruction to detect a non-zero balance in one of the internal counters and advance to the next instruction. If the contents of the selected counter is "00", the PACER will skip the next instruction. The proper counter is selected by entering "A", "B", "C", or "D" in the argument field (execute = 23 us).
%	&				
5	6				
<b>Note:</b> The "A" and "B" counters are limited to a count from 00 to 99. To extend a count beyond that limit the counters must be chained.					

**TABLE 6-2**  
**DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)**

KEYSTROKE	OPERATION
-----------	-----------

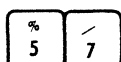
**CONTROL LEAD AND COUNT INSTRUCTIONS (Cont'd)**

**EXAMPLE:**

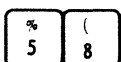
```

00 : 63 : COMPLEMENTARY SP COUNT
01 : 30 :
02 : 55 : A
03 : 55 : B
04 : 50 : 100
05 : 54 : A
06 : 53 : B
07 : 04 : -6
    
```

**TIMER INSTRUCTIONS**



- \* CLEAR TIMER. This is an instruction to reset to 0 and start an internal timer. The timer counts from 0 to 127 ms and 0.1 to a maximum of 12.7 seconds for operation with the PACER. The same timer is used by instructions "57" and "58". In Level-2, two distinct timers are provided (execute = 23 us).



- \* SET TIMER AND SKIP NEXT INSTRUCTION, ADVANCE TO NEXT INSTRUCTION WHEN TIMER RUNS OUT. This is an instruction to run an internal timer for a period of 1 to 127 milliseconds or 0.1 to 9.9 seconds. If the argument does not include a decimal point, the time interval is in milliseconds. An entry in seconds must include a decimal point.

**Example:**

Step	Code	Argument	Comment
	58	2.5	Set internal timer for a period of 2.5 seconds.

As soon as the timer is set, the next instruction is skipped. When the timer runs out, the program is interrupted and returns to the skipped instruction (execute = 54 us). To avoid interrupts, use a "57" instruction to clear the timer. For operation with the PACER, the same timer is used by instructions "57" and "58". In Level-2, two distinct timers are provided.

TABLE 6-2

## DICOL/LEVEL-2 INSTRUCTION SET (Cont'd)

KEYSTROKE	OPERATION		
<b>TIMER INSTRUCTIONS (Cont'd)</b>			
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>% 5</td><td>) 9</td></tr></table>	% 5	) 9	* <u>DISPLAY TIMER.</u> This is an instruction to display the contents of the interval timer. If preceded by a "57" instruction, the display reads accumulated time and cannot show more than 12.7 seconds (at which time the count proceeds from 00 and repeats). If preceded by a "58" instruction, the display reads time remaining to 00 and will not exceed 9.9 seconds in the PACER to 65 seconds in Level-2. In Level-2, the contents of this timer may be stored in counter "L" using instruction "60".
% 5	) 9		
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>&amp; 6</td><td>ø</td></tr></table>	& 6	ø	* In Level-2, this instruction stores the contents of the interval timer in counter "L".
& 6	ø		
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>&amp; 6</td><td>!</td></tr></table>	& 6	!	<u>OFF-LINE PROGRAM LOAD (OPTION).</u> This is an instruction to transfer programs to and from the PACER or some remote device defined as follows:
& 6	!		
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>SI O</td></tr></table>	SI O	= Output user programs via RS-232 pin 2 or 3, determined by preceding "00" instruction.	
SI O			
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>HT I</td></tr></table>	HT I	= Input programs via RS-232, compute check character and sound audible alarm if programs do not check.	
HT I			
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>ETX C</td></tr></table>	ETX C	= Check programs received against programs stored in program memory and sound audible alarm if data does not check.	
ETX C			
<b>Note:</b> Remember to turn on AUDIO/VOLUME control prior to execution of program.			
Each of the above argument instructions may be preceded by a file name assigned to the programs. The file name allows the PACER to ignore any recorded data block that is not preceded by the desired file name. The last character of the argument must be one of three characters described above.			
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>&amp; 6</td><td>" 2</td></tr></table>	& 6	" 2	Reserved for future use.
& 6	" 2		
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>&amp; 6</td><td># 3</td></tr></table>	& 6	# 3	<u>PROGRAM IDENTIFIER.</u> This instruction allows the operator to enter from one to 25 characters of the program identification data. Each and every program must begin with the "63" instruction.
& 6	# 3		

---

**5. SET IO PARAMETERS . . . MODE "T"** HT  
I

---

**5.01** Once the HT  
I key is struck, IO parameters are established in the same manner as described in Chapter 7. Striking the CONT key allows the user to enter separate parameters for each of the front ends (pin 2 and pin 3). Cursor position and parameter entry are controlled using the ↑, ↓, ←, and → keys. Striking the END key will return to the Level-2 Directory.

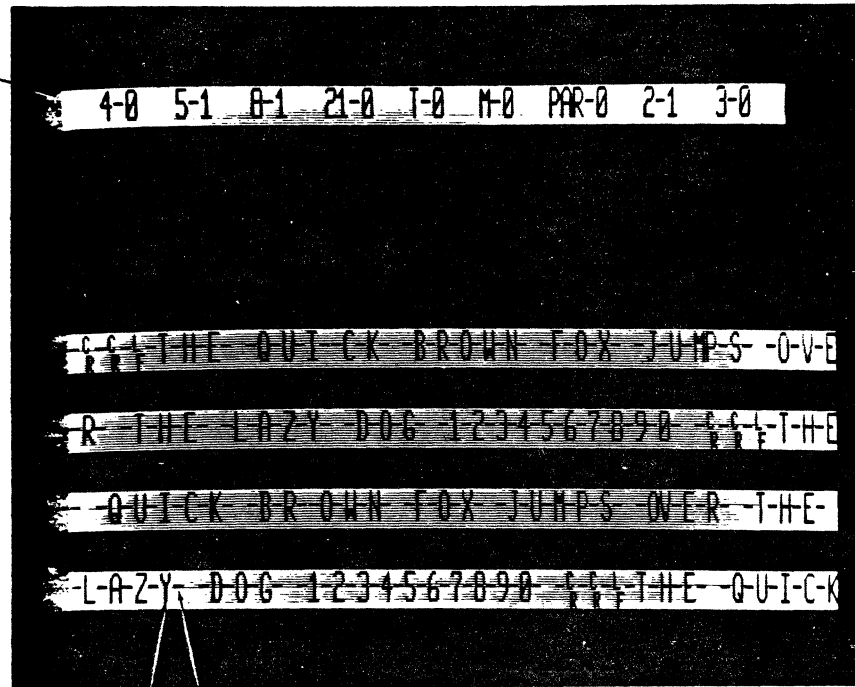
---

**6. DISPLAY CAPTURE BUFFER . . . MODE "D"** EOT  
D

---

**6.01** The Level-2 Display Mode is selected from the menu by striking the EOT  
D key. This mode allows the user to view the contents of the capture buffer including both data or status bytes. Data is displayed in the full or half duplex format, depending on how it was captured, while status is indicated on the top line of the CRT. The status is the condition, on or off, of selected interface leads at the moment each data byte is captured. The status displayed is that of the top left-most character appearing in the data display. A typical capture buffer display is shown in Figure 6-5. Figure 6-6 shows two lines of data as they might appear in the full or half duplex displays.

Status Display  
of First Char-  
acter in Data  
Display



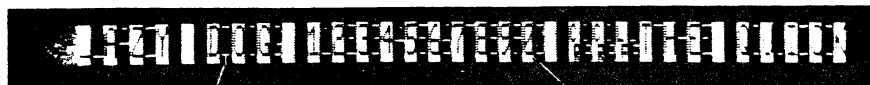
4 lines of  
data, 64-  
characters  
each for  
a total of  
256-char-  
acters

Pin 3 Data

Pin 2 Data  
Not Displayed (See Table 6-3)

Fig. 6-5 Typical Captured Data Display

FULL DUPLEX



PIN 2 DATA

PIN 3 DATA



HALF DUPLEX

Fig. 6-6 Typical Full and Half Duplex Lines



Display Mode Commands

6.02 The following commands are used in the Display Mode after data has been received and logged in capture memory. These commands require the use of keys in both the control cluster and main array.

TABLE 6-3  
DISPLAY MODE COMMANDS

KEYSTROKE	COMMAND
<code>   </code>	Display only that data received and logged while lead 22 (SQ) was ON. A hyphen indicates that the logged character was received while lead 22 was OFF. To counter this command and display all data, strike the <code>SOH A</code> key.
<code>" 2</code>	Display only the data logged on lead 2. To counter this command and display all data, strike the <code>SOH A</code> key.
<code># 3</code>	Display only the data logged on lead 3. To counter this command and display all data, strike the <code>SOH A</code> key.
<code>\$ 4</code>	Display only that data received and logged while lead 4 (RTS) was ON. A hyphen indicates that the logged character was received while lead 4 was OFF. To counter this command and display all data, strike the <code>SOH A</code> key.
<code>% 5</code>	Display only that data received and logged while lead 5 (CTS) was ON. A hyphen indicates that the logged character was received while lead 5 was OFF. To counter this command and display all data, strike the <code>SOH A</code> key.
<code>&amp; 6</code>	Display only parity flawed characters. An underscore indicates that the logged character was without parity error. To counter this command and display all data, strike the <code>SOH A</code> key.
<code>( 8</code>	Display only those characters received and logged while lead 8 (COD) was ON. An underscore indicates that the logged character was received while lead 8 was OFF. To counter this command and display all data, strike the <code>SOH A</code> key.
<code>SOH A</code>	Display all data received on leads 2 and 3.
<code>STX B</code>	Strike to display the number of characters in capture memory. The contents are measured from the left-most character on the display to the last character in memory. When the memory has been overwritten, or if a "08" instruction is included in the program, and the capture memory is filled, the display will read the number of characters in a full buffer, i.e., "SIZE = 4607".

TABLE 6-3

DISPLAY MODE COMMANDS (Cont'd)

















KEYSTROKE	COMMAND
 / 	<p>Strike to display the contents of all internal counters (A through L). To counter this command and return to the previous display, strike the  key.</p>
	<p>Output the contents of the capture buffer to an external device. The buffer contents includes all data from the characters appearing in the top left-most position of the data display to the end of memory. The output is in serial form via pin 3 of the rear panel XIO port. The language is the same as received, but all other parameters are set in the XIO Mode (see paragraph 8.01).</p>
	<p>Display the contents of the capture memory in binary. To counter this command and display normal data, strike the  key.</p>
	<p>Display the contents of capture memory in hexadecimal equivalents. To counter this command and display normal data, strike the  key.</p>
	<p>Display the contents of capture memory in inverted hexadecimal equivalents. To counter this command and display data, strike the  key. Inverted = one's complement.</p>
	<p>Display only that data received and logged while lead M was ON. A hyphen indicates that the logged character was received while lead M was OFF. To counter this command and display all data, strike the  key.</p>
	<p>Output the contents of capture memory to a hard copy device. The hard copy will include all data from the characters appearing in the left-most position of the display to the end of memroy. An automatic CR/LF will be generated every 64-characters unless a CR is detected in the data. A delay of 256 ms. is generated after any CR sent (whether added by the ENCORE or not) to permit execution by a printing device. The output is serial ASCII via pin 3 of the rear panel XIO port. All other parameters are set in the XIO Mode (see paragraph 8.01).</p>
	<p>Search captured data for the next parity error. All characters received and logged in parity error are half intensity. Each time the  key is depressed, the next errored character in memory appears in the extreme top left-most position of the data display. If the  key is depressed and there are no errored characters remaining in memory, the display will read "NONE".</p>

TABLE 6-3

DISPLAY MODE COMMANDS (Cont'd)

























KEYSTROKE	COMMAND
	<p>Search for the next match sequence (up to 15 characters). Each time the  key is depressed, the next match sequence in memory appears in the top left-most position of the data display. If the  key is depressed and there are no match sequences remaining in memory, the display will read "NO MATCH". The search is performed on displayed characters only. If, for example, the display shows lead 2 data, the search is performed on that data only. The data on lead 3 is not searched. (Useful in FDUX, character interlaced data).</p>
	<p>Display only that data received and logged while lead T was ON. A hyphen indicates that the logged character was received while lead T was OFF. To counter this command and display all data, strike the  key.</p>
	<p>Cancel the previous command.</p>
	<p>Strike once to move displayed data one character to the left. Depress and hold the  key to repeat.</p>
	<p>Strike once to move displayed data one character to the right. Depress and hold the  key to repeat.</p>
	<p>Strike once to move displayed data 64 characters to the left. Depress and hold the  key to repeat.</p>
	<p>Strike once to move displayed data 64 characters to the right. Depress and hold the  key to repeat.</p>
	<p>Strike once to move displayed data forward 256 characters.</p>
	<p>Strike once to move displayed data backward 256 characters.</p>
	<p>Display first 256 characters in buffer memory.</p>
	<p>Depress and hold the  key and strike the  key to display the last 256 characters in buffer memory.</p>
	<p>Shift displayed data one bit to the left. Use this command to decipher transparent data.</p>
	<p>Shift displayed data one bit to the right. Use this command to decipher transparent data.</p>
<p>SPACE</p>	<p>Invert case if displayed data is in a case shift code.</p>









TABLE 6-3

DISPLAY MODE COMMANDS (Cont'd)

KEYSTROKE	COMMAND
	<p>Following this command, key in a match sequence of up to 15 characters. When the sequence is entered, strike the  key again for normal display of captured data.</p>

**Typical Display Mode Operation**

**6.03** The purpose of this procedure is to provide the user with step by step instructions for analyzing captured data using Display Mode Commands. Before entering the Display Mode, a program to be capture data must be executed. It is assumed that such a program is resident in the current PSC file.

STEP	PROCEDURE
1.	<p>From the Level-2 Directory, select Mode I by striking the  key. Then examine the IO parameters and change if necessary.</p> <p><b>Note:</b> If the program to be executed includes a "01" instruction, IO Mode parameters change automatically upon execution.</p>
2.	<p>Strike the  key after the correct IO parameters are established and note that the Level-2 Directory is displayed once again.</p>
3.	<p>From the directory, select Mode "P" by striking the  key. Then select the program to be executed using the  keys.</p>
4.	<p>When the desired program is displayed, return to the Level-2 Directory by striking the  key.</p>
5.	<p>Execute the selected program by striking the  key.</p> <p><b>Note:</b> For Display Mode analysis of captured data, the program must include some receive and log instruction.</p>
6.	<p>When a sufficient quantity of data is captured, strike the  key twice to return to the Level-2 Directory.</p>
7.	<p>From the directory select Mode "D" by striking the  key and note the display of captured data.</p>

STEP	PROCEDURE
8.	<p data-bbox="409 344 1500 436">Refer to Table 6-3 and exercise the appropriate Display Mode commands to analyze captured data. For example, the following commands can be used to determine the conditions under which the data was received:</p> <ul style="list-style-type: none"> <li data-bbox="483 466 1500 499"> <input type="checkbox"/>       Display data logged while lead 21 (SQ) was ON.         </li> <li data-bbox="483 529 1500 562"> <input type="checkbox"/> 2     Display data logged on lead 2 (transmitted data).         </li> <li data-bbox="483 592 1500 625"> <input type="checkbox"/> # 3     Display data logged on lead 3 (received data).         </li> <li data-bbox="483 655 1500 688"> <input type="checkbox"/> S 4     Display data logged while lead 4 (RTS) was ON.         </li> <li data-bbox="483 718 1500 751"> <input type="checkbox"/> * 5     Display data logged while lead 5 (CTS) was ON.         </li> <li data-bbox="483 781 1500 814"> <input type="checkbox"/> &amp; 6     Display parity flawed characters.         </li> <li data-bbox="483 844 1500 877"> <input type="checkbox"/> ( 8     Display data logged while lead 8 (COD) was ON.         </li> <li data-bbox="483 907 1500 940"> <input type="checkbox"/> SON A     Display all data.         </li> <li data-bbox="483 970 1500 1003"> <input type="checkbox"/> EOT D     Output to hard copy device.         </li> <li data-bbox="483 1033 1500 1108"> <input type="checkbox"/> CR M     Display data logged while lead M (strapped to any DCE lead) was ON.         </li> <li data-bbox="483 1138 1500 1213"> <input type="checkbox"/> SI O     Output to hard copy device and include CR/LR every 64 characters.         </li> <li data-bbox="483 1243 1500 1297"> <input type="checkbox"/> DCA T     Display data logged while lead 7 (strapped to any DTE lead) was ON.         </li> </ul>
9.	<p data-bbox="409 1331 1500 1394">Use the following commands to manually control data flow across the display:</p> <ul style="list-style-type: none"> <li data-bbox="483 1423 1500 1457"> <input type="checkbox"/> ←     Move one character to the left.         </li> <li data-bbox="483 1486 1500 1520"> <input type="checkbox"/> →     Move one character to the right.         </li> <li data-bbox="483 1549 1500 1583"> <input type="checkbox"/> ↑     Move 64 characters to the left.         </li> <li data-bbox="483 1612 1500 1646"> <input type="checkbox"/> ↓     Move 64 characters to the right.         </li> <li data-bbox="483 1675 1500 1709"> <input type="checkbox"/> SHIFT ↑     Move displayed data forward 256 characters.         </li> <li data-bbox="483 1738 1500 1772"> <input type="checkbox"/> SHIFT ↓     Move displayed data backward 256 characters.         </li> <li data-bbox="483 1801 1500 1835"> <input type="checkbox"/> DC     Display the first 256 characters captured.         </li> <li data-bbox="483 1864 1500 1898"> <input type="checkbox"/> SHIFT DC     Display the last 256 characters captured.         </li> </ul>

STEP	PROCEDURE
10.	To search all data for parity errors or a specific character sequence, use the following commands: <ul style="list-style-type: none"> <li data-bbox="488 436 1495 499"> <span style="border: 1px solid black; padding: 2px;">DLE P</span> Search for and display next parity error (top left-most character).             </li> <li data-bbox="464 531 1495 594"> <span style="border: 1px solid black; padding: 2px;">K</span> / <span style="border: 1px solid black; padding: 2px;">K</span> Enter match sequence (up to 15 characters) and follow with <span style="border: 1px solid black; padding: 2px;">K</span> again.             </li> <li data-bbox="488 625 1495 688"> <span style="border: 1px solid black; padding: 2px;">DC3 S</span> Search for and display next match sequence (top left-most characters).             </li> </ul>
11.	To display the number of characters from the top left-most position on the display to the end of memory, or the contents of the internal counters, use the following commands: <ul style="list-style-type: none"> <li data-bbox="488 846 1243 888"> <span style="border: 1px solid black; padding: 2px;">STX R</span> Depress and hold for number of captured bytes.             </li> <li data-bbox="488 909 1292 951"> <span style="border: 1px solid black; padding: 2px;">ETX C</span> Depress and hold for contents of internal counters.             </li> </ul>
12.	For data conversion use the following commands: <ul style="list-style-type: none"> <li data-bbox="488 1035 1276 1077"> <span style="border: 1px solid black; padding: 2px;">ENQ E</span> Convert/expand normal data to binary equivalent.             </li> <li data-bbox="488 1098 1260 1140"> <span style="border: 1px solid black; padding: 2px;">BS H</span> Convert normal data to hexadecimal equivalent.             </li> <li data-bbox="488 1161 1382 1203"> <span style="border: 1px solid black; padding: 2px;">HT I</span> Convert normal data to inverted hexadecimal equivalent.             </li> <li data-bbox="427 1224 1341 1266"> <span style="border: 1px solid black; padding: 2px;">SHIFT</span> <span style="border: 1px solid black; padding: 2px;">←</span> <span style="border: 1px solid black; padding: 2px;">SHIFT</span> <span style="border: 1px solid black; padding: 2px;">←</span> Convert transparent data to normal (intelligible) data.             </li> <li data-bbox="451 1287 1065 1329"> <b>SPACE</b> Invert case (case shift codes only).             </li> </ul>
13.	To transfer data from the ENCORE to an external device, strike the <span style="border: 1px solid black; padding: 2px;">EOT D</span> key. The external device must be RS-232C compatible serial ASCII and must be connected to the external IO port (XIO). XIO parameters are set as described in paragraph 8.01.

---

## 7. PRINT LEVEL-2 MENU CR M

**7.01** A listing of the current Level-2 Menu (all programs currently residing in the PSC file) may be output to an external device (e.g., PACER or cassette recorder) by selecting "PRINT LEVEL-2 MENU" from the Level-2 Directory. The listing is output, in DICOL, via the XIO port once the proper XIO parameters are established. The Level-2 system program, L2P, executed by striking the CR  
M key when the directory is displayed, prompts the operator for entry of XIO parameters and for output of the listing.

### Typical Menu Print Operation

**7.02** This procedure provides the user with simple step-by-step instructions for output of the Level-2 Menu to an external device. It is primarily used to obtain a hard copy listing of the DICOL programs resident in the Level-2 PSC file.

---

STEP	PROCEDURE
1.	From the Level-2 Directory select "PRINT LEVEL-2 MENU" by striking the <input type="checkbox"/> key and note that the CRT displays:  STRIKE "RETURN" TO SET XIO  STRIKE "SPACE BAR" TO PRINT MENU
<b>Fig. 6-7 Level-2 Menu Print</b>	
2.	Unless XIO parameters have already been established, strike the <input type="checkbox"/> key, examine XIO parameters, and change if necessary (see paragraph 8.01).
3.	Strike the <input type="checkbox"/> key and note the same display shown in Step 1.
4.	Assure that the external device is properly connected to the ENCORE.
5.	Strike the "SPACE BAR" and note that the Level-2 Menu is displayed on the CRT as it is output to the external device. Also note that the Level-2 Directory is displayed when menu output is completed.

---

---

### 8. XIO CAN X

---

**8.01** With one exception, each item in the Level-2 Directory requiring the use of the XIO port provides for operator entry of external IO parameters. The exception is Mode "D", Display of the Capture Buffer. When displaying the contents of the capture buffer, the user may execute the "DUMP" command by striking the  key. This outputs the contents of the buffer via the XIO port. On power-up, the XIO parameters are set as described in Chapter 7. If these parameters are satisfactory, then the DUMP command may be used. If, however, these parameters must be changed, the user must set XIO parameters prior to entering Mode "D". The most convenient method of accomplishing this is to:

1. Select  from the Level-2 Directory.
2. Change parameters using the  ,  ,  , and  keys.
3. Strike the  key and note return to the Level-2 Directory.

**8.02** The External IO Mode (XIO) is designed to permit operator selection of external IO parameters for interface with equipment connected to the rear panel XIO port. With the exception noted in the previous paragraph, XIO Mode entry is controlled by the Level-2 system programs. This permits user selection of XIO parameters with a minimum of keystrokes. A complete explanation of how these parameters are selected is found in Chapter 7.

---

### 9. LOAD NEW MENU FROM PACER ! |

---

**9.01** This selection allows the user to input DICOL programs from the PACER to the Level-2 PSC file via the XIO port. Striking the  key selects this item from the Level-2 Directory and executes the system program "P61I". The video prompts that follow remind the user to set ENCORE's XIO parameters equal to the PACER IO, to connect the PACER to the ENCORE XIO port, and, on the PACER, to key in and execute the PACER "DUMP" program. Once the PACER program is executed, the contents of the PACER program memory is transferred to the ENCORE source buffer. If transfer is successful, the ENCORE returns to the Level-2 Directory.



### Typical PACER to ENCORE Menu Load

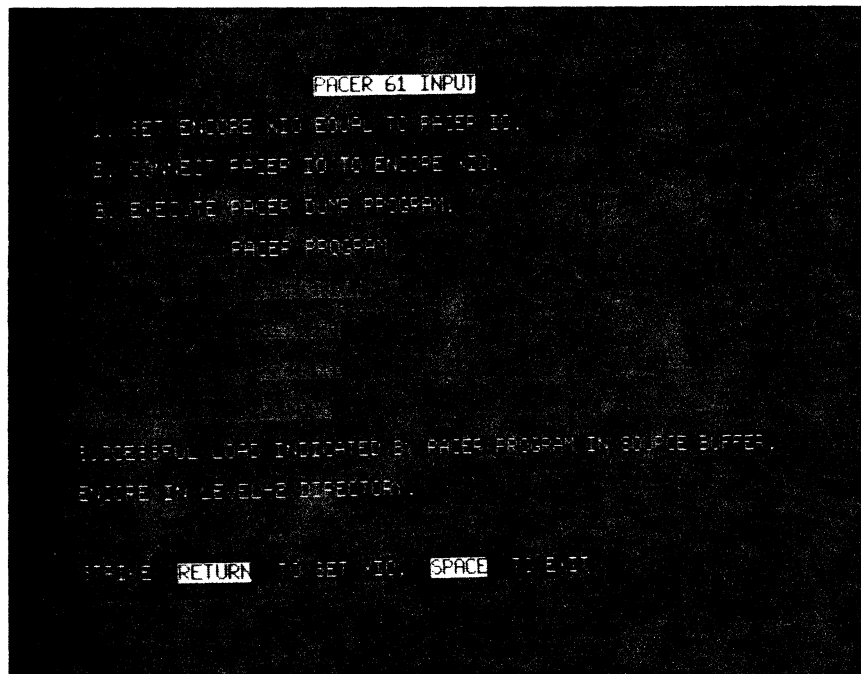
**9.02** This procedure provides the step-by-step instructions required for transfer of DICOL programs from PACER to ENCORE.

---

STEP	PROCEDURE
------	-----------

---

1. From the Level-2 Directory, select "LOAD NEW MENU FROM PACER" by striking the **[1]** key and note the following display.



**Fig. 6-8 PACER to ENCORE Menu Load**

2. Unless XIO parameters have already been established, strike the **[RETURN]** key, examine XIO parameters, and change if necessary (see paragraph 8.01).
  3. Strike the **[CMD]** key and note the same display shown in Step 1.
  4. Assure that the PACER is connected to the ENCORE XIO port.
  5. Key the "DUMP" program into the PACER and execute it.
  6. When transfer has been successfully completed, the ENCORE will return to the Level-2 Directory.
-

**10. LOAD NEW MENU FROM DISKETTE** 2

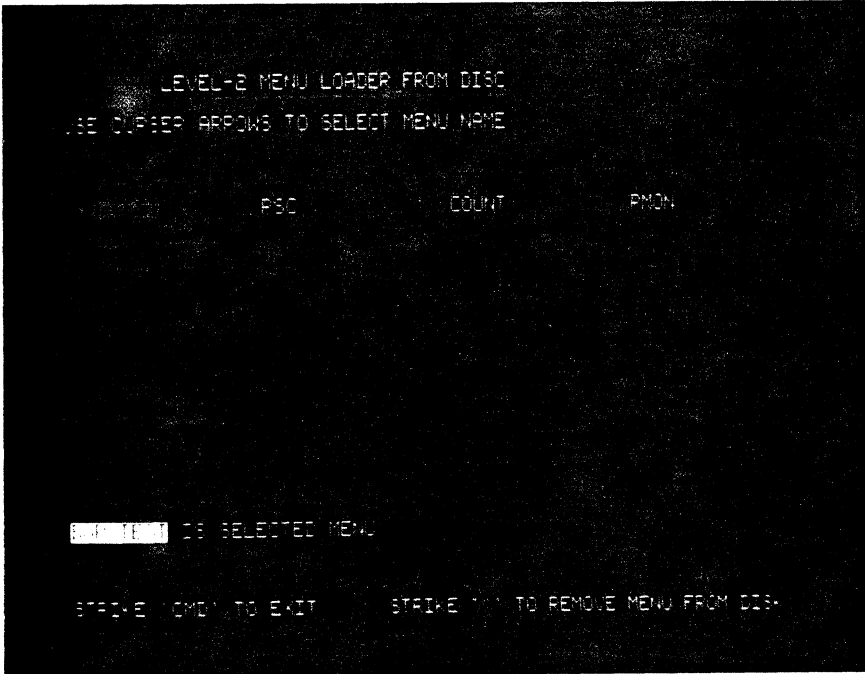
**10.01** This selection allows the user to transfer DICOL/Level-2 programs from diskette to the PSC file. Striking the 2 key selects this item from the Level-2 Directory and executes the system program DISC-GET. This program searches the entire diskette for DICOL programs and displays their names on the CRT for selection by the user. In addition to selecting programs for transfer, the user, if desired, may also erase any of the displayed programs. In either case, the transfer or removal of programs from diskette is accomplished by responding to the appropriate video prompts.

**Typical Menu Load from Diskette**

**10.02** The procedure that follows provides the instructions required for transfer of DICOL programs from diskette to the ENCORE's source buffer.

STEP	PROCEDURE
------	-----------

1. From the Level-2 Directory, select "LOAD NEW MENU FROM DISKETTE" by striking the 2 key and note a display similar to that shown in Figure 6-9.



**Fig. 6-9 Menu Load from Diskette**

---

STEP	PROCEDURE
2.	Use the <input type="button" value="↑"/> , <input type="button" value="↓"/> , <input type="button" value="←"/> , and <input type="button" value="→"/> keys to highlight the desired program name.
3.	To transfer the desired program from disc to PSC file, strike the <input type="button" value="CMD"/> key and note the return of the Level-2 Directory.
4.	To erase the desired program from diskette, strike the <input type="button" value="^"/> key twice. Successful deletion of the program is indicated when the same display shown in Step 1 returns, minus the deleted program. Strike the <input type="button" value="CMD"/> key and note the return of the Level-2 Directory.

---

---

**11. DUMP CURRENT MENU TO PACER**

---

**11.01** This item is selected from the Level-2 Directory by striking the  key. This executes the system program "P610" which is designed to transfer DICOL programs from the ENCORE's PSC file to the PACER program memory via the XIO port. For successful transfer, the ENCORE's XIO parameters must be set to agree with the PACER IO parameters and pin 8 of the interface must be on. In addition, the "LOAD" program must be keyed into the PACER and executed. DICOL program transfer is then initiated by striking ENCORE's "SPACE BAR". If transfer is successful, the ENCORE returns to the Level-2 Directory.

**Typical ENCORE to PACER Menu Dump**

**11.02** The following procedure details those steps required for successful transfer of DICOL program from ENCORE to PACER.

STEP	PROCEDURE
1.	From the Level-2 Directory, select "DUMP CURRENT MENU TO PACER" by striking the <b>[F]</b> key and note the display shown in Figure 6-10.

```

OUTPUT TO PACER
1. SET ENCORE XIO EQUAL TO PACER IO.
2. CONNECT PACER IO TO ENCORE XIO.
3. PACER MUST SIMULATE DTE.
4. INSURE PACER'S PIN-8 IS ON
5. EXECUTE PACER LOAD PROGRAM.

PACER PROGRAM
00:00:LOAD
01:00:T
02:00:I
03:00:

PRESS [RETURN] TO SET XIO .. PRESS [F] TO TRANSMIT PROGRAM

```

**Fig. 6-10 ENCORE to PACER Menu Dump**

2. Unless XIO parameters have already been established, strike the **[RETURN]** key, examine XIO parameters, and change if necessary (see paragraph 8.01).
3. Strike the **[CMD]** key and note the same display shown in Step 1.
4. Assure that the PACER is connected to the ENCORE XIO port via T-Connector assembly or equivalent.
5. Connect a jumper between pins 20 and 8 of the T-Connector (COD on).
6. Key the "LOAD" program into the PACER and execute it.
7. Strike the ENCORE's "SPACE BAR" and note that the DICOL programs are transmitted and that the Level-2 Directory is displayed following successful program transfer.

---

**12. DUMP CURRENT MENU TO DISKETTE** 5  
4

---

**12.01** Striking the ↓ key selects this item from the Level-2 Directory and executes the system program "DISC-PUT". Proper response to video prompts will then allow the user to transfer the contents of the current PSC file to the disc using a name of no more than 10 characters in length. When entering the file name, it must begin with a letter and include no spaces or control characters. If these conditions are met, the file will be stored on disc and the ENCORE will return to the Level-2 Directory.

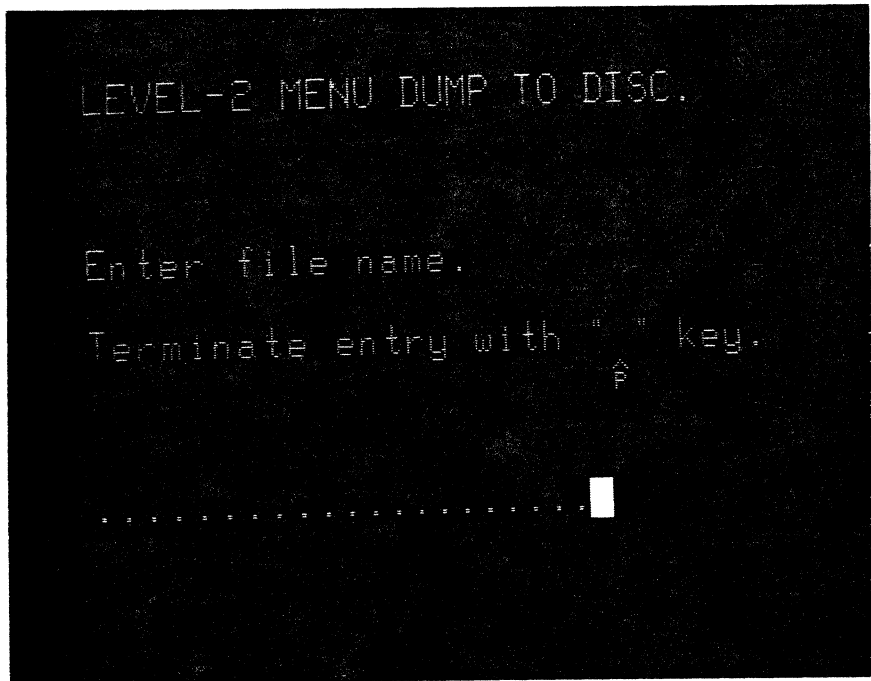
**Typical Menu Dump to Diskette**

**12.02** The procedure that follows provides the instructions required for transfer of DICOL programs from the ENCORE's source buffer to diskette.

---

STEP	PROCEDURE
1.	From the Level-2 Directory, select "DUMP CURRENT MENU TO DISKETTE" by striking the <span style="border: 1px solid black; padding: 2px;">↓</span> key and note the display shown in Figure 6-11.

---



**Fig. 6-11 Menu Dump to Diskette**

## DUMP CURRENT MENU TO DISKETTE

---

STEP	PROCEDURE
2.	Assure the diskette is not write protected.
3.	Type the desired file name including no more than 10 characters, no spaces, and no control characters. Name must begin with an alpha. Terminate the entry by striking the <input data-bbox="753 474 786 516" type="checkbox"/> key and note that the ENCORE returns to the Level-2 Directory.

---



## CHAPTER 7

### LEVEL-3 OPERATION AND PROGRAMMING

#### 1. GENERAL

**1.01** This chapter of the manual is devoted to operation of the ENCORE in Level-3. The chapter includes a detailed explanation of all Level-3 operating commands, general information of use to the programmer, and a complete description of all COMBASIC instructions including those used to write front end programs.

**1.02** Level-3 is intended to provide the user with a means of creating his own custom programs individually tailored to his specific needs. This requires the use of COMBASIC, an easy to learn, yet powerful language developed by DIGITECH for the ENCORE. COMBASIC is designed to preserve the fundamental characteristics which have made BASIC such a universal programming language, while enhancing its capabilities for use in a data communications environment. The commands and instructions used in COMBASIC are detailed in this chapter. Other tools available to the COMBASIC user are instruction variables, operators, and various ports with which the user may communicate. To the maximum extent possible, symbology, semantics, and syntax are in agreement with accepted practices in computer programming. COMBASIC provides the solution to such a wide range of problems that its versatility is limited only by the user's imagination.


**1.03** This discussion assumes that the user has had previous programming experience in a high level language. For additional information, the user may refer to any number of BASIC primers currently available.


#### 2. LEVEL-3 COMMANDS

**2.01** The Level-3 Command Mode is the mode from which all Level-3 commands are entered. The Command Mode display echoes keyboard entry, displays error messages when appropriate, and includes a real time clock. Level-3 Commands for control of the disc drive are preceded with D space, i.e., D EDIT, D MENU, etc.. All commands must be separated from a program name by a space, e.g., EDIT *program name*. In addition, the program name must begin with an alphabetical character, include no spaces, pound signs (#), exclamation points (!), or commas and consist of no more than ten characters. Every command or command statement (command plus program name) must be terminated with a carriage return (  ), e.g., MENU . Each of the commands shown in Table 7-1 is detailed in the paragraphs that follow.






**TABLE 7-1**  
**LEVEL-3 COMMANDS**








COMMAND	REF PARA	COMMAND	REF PARA
ASGN	2.02	KILL	2.23
D*	2.05	MENU	2.24
D DXFR	2.06	RUN	2.25
D FORMAT	2.07	SAVE	2.26
EDIT	2.08	TIME	2.27
HELP	2.12	STATUS (  )	2.28
IO	2.13	XIO	2.31

**Note:** The Status Mode is entered from the Level-3 Command Mode by striking the  key. It is the only Level-3 Mode entered in this manner.

**ASGN**

**2.02** The ASGN Command allows the user to enter the Assign Mode and establish up to 26 different User Defined Functions (A through Z). Each function may be an instruction(s), a command(s), or a series of keystrokes, up to 27 characters in length. Figure 7-1 gives several examples of user defined functions. The "A = @ASGNCR", for example, allows the user to enter the Assign Mode by simply depressing   in any mode. The @ is equivalent to depressing the  key prior to executing the command that follows. The advantage of using these functions is that a series of frequently used keystrokes can be reduced to two simple entries.

**2.03** Another feature of the Assign Mode allows the operator to assign a power-up sequence. The sequence shown in Figure 7-1 will, on power-up, cause the ENCORE to execute the disc stored program, RUN. This is especially useful when programming has been completed because it allows an inexperienced user to turn on the ENCORE and immediately execute a program on power-up. Memory may also be allocated in the power-up sequence using the number sign (#) and exclamation point (!) as explained later in this chapter.

**2.04** In the Assign Mode, cursor position is controlled using the , , , and  keys. The  and  keys are used to delete and insert characters, respectively. Entries are terminated by striking the  key, thereby exiting the Assign Mode and returning to the Level-3 Command Mode.

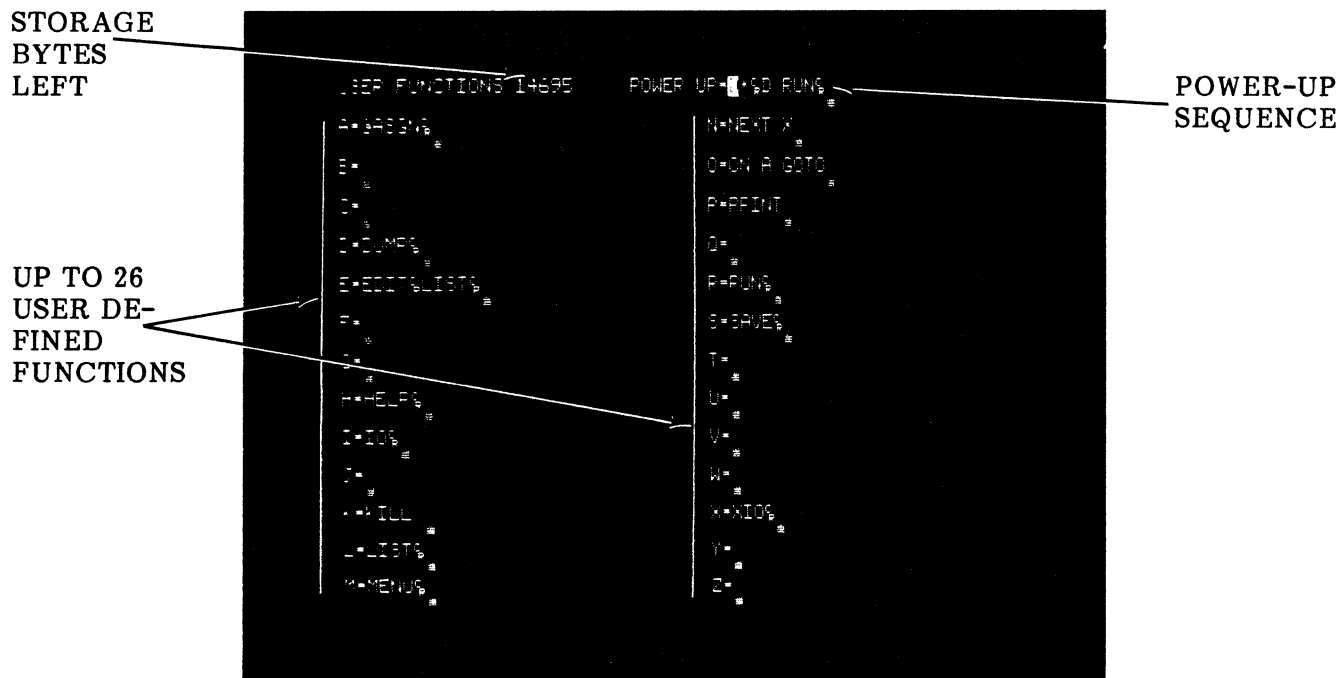


Fig. 7-1 ASGN Mode Format

**D\***

**2.05** This command is designed to automatically search both storage and disc whenever a program is executed using *program name*. It can be included as part of the power-up sequence, e.g., D\*  D RUN , or may be entered from the Level-3 Command Mode at the user's discretion. It need be entered only once after power-up, and remains in effect until power is turned off.

**D DXFR**

**2.06** This is a command to execute the disc stored program DXFR. The program is designed to copy any data or program disc onto another disc that has been properly formatted (D FORMAT). The program will make up to nine copies of a single disc each time it is executed. Because an entire disc cannot be duplicated in one pass, the master and copy discs must be removed and reinserted several times during the copy process. The program provides all the user prompts required for swapping the discs and will alert the user when the copy process is complete. Upon completion, simply strike the  key to return to the Level-3 Command Mode. To exit this program prior to completing the copy disc, type OUT.

**NOTE:** Always write protect the master disc.

**D FORMAT**

**2.07** This is a disc command that must be used to format a new disc. When executed, it instructs the ENCORE to record track and sector boundaries on both sides of the disc. Each side contains 34 tracks separated into 16 sectors each. A sector may contain up to 256 bytes of program information or captured data. Dynamic allocation of disc space provides a minimum of 27 904 bytes and a maximum of 196 608 bytes of memory for program storage. A minimum of 49 152 bytes and a maximum of 237 568 bytes is available for captured data.

## EDIT

**2.08** The EDIT command is designed to provide the user with program entry and edit capabilities. It may be used in a command statement that accesses a program in storage or on disc. It accesses a program by name, and may be used with the pound sign (#) and exclamation point (!) to automatically allocate memory. Upon successful execution, the user may then execute any of the EDIT commands shown in Table 7-2 to aid in writing and debugging a COMBASIC program. Techniques for writing a COMBASIC program, while in the EDIT Mode, are covered under paragraphs 3 and 4. The basic forms and uses of the EDIT command are covered in the following paragraphs.

- **EDIT.** This command allows the user to edit the program currently residing in storage or to write a new program that must be named using the SAVE command.
- **EDIT *program name*.** When used with a program name, the EDIT command loads the specified program from storage into source. If the specified program does not reside in storage (has not been SAVE'd), a new program is initiated under the specified name. Any program residing in source is lost if it has the specified name.
- **EDIT *program name*!** The addition of an exclamation point following the program name automatically allocates source and object memory sufficient to edit and execute the specified program. Re-allocation occurs only if the current allocation is insufficient.
- **# EDIT *program name*!** This command performs the same function described above except that the object and source memories are first reduced to their minimum values of 4 096 and 256 bytes, respectively. Both memories are then incremented in steps of 256 bytes each until sufficient memory is allocated for edit and execution of the program. Since the minimum allocation is made, extensive editing may result in the need to increase memory allocation beyond the original requirements.
- **D EDIT *program name*.** The D and space before EDIT initiates a search of the disc for the specified program. If the program resides on disc, it is then transferred to source where it can be edited. If it does not reside on disc (has not been D SAVE'd), the ENCORE displays the message "ERROR #18" (NO SUCH DISC FILE).
- **D EDIT *program name*!** The addition of an exclamation point, as shown in this statement, automatically allocates source and object memory for the disc edited program.
- **# D EDIT *program name*!** This command performs the same function as D EDIT *program name* except that the object and source memories are first reduced to their minimum values of 4 096 and 256 bytes, respectively. Both memories are then incremented in steps of 256 bytes each until sufficient memory is allocated for edit and execution of the program. Since the minimum allocation is made, extensive editing may result in the need to increase memory allocation beyond the original requirements.

**2.09** In addition to the EDIT commands already discussed, there are two other methods of entering the EDIT Mode:

- a) If the user executes a program while in the EDIT Mode, the ENCORE will return to the EDIT Mode when program execution is terminated by the CHAIN " " instruction or by striking the **END** key. A program is executed while in the EDIT Mode by using the RUN command as described in Table 7-2.
- b) A user defined function may also be used to enter the EDIT Mode. For example, the **UDF** **END** keys could be used to first enter the EDIT Mode and then list the resident program. The function would be entered in the ASGN Mode as follows: E = @ EDIT **RETURN** LIST **RETURN**. The @ simulates striking the **END** key; EDIT is then automatically executed from the Command Mode while LIST is executed from the EDIT Mode.










**2.10** Upon entering the EDIT Mode, the user may execute a wide variety of commands designed to simplify the entry, edit, and debugging of COMBASIC programs. These commands and edit keys are described in the following table. Each command must be terminated by a carriage return, **RETURN**. The edit keys require no termination.

**TABLE 7-2**

**EDIT COMMANDS AND KEYSTROKES**

COMMAND/KEY	FUNCTION
<div style="display: inline-block; border: 1px solid black; padding: 2px; margin-right: 10px;">→</div> <div style="display: inline-block; border: 1px solid black; padding: 2px;">←</div>	These keys position the cursor at any point on a program line.
<div style="display: inline-block; border: 1px solid black; padding: 2px; margin-right: 10px;">↑</div> <div style="display: inline-block; border: 1px solid black; padding: 2px;">↓</div>	These keys are used to scroll through the program, line by line.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">IC</div>	Each time this key is struck, the program line is opened for insertion of another character. The point of insertion is indicated by the position of the cursor.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">DC</div>	Each time this key is struck, a character is deleted from the program line. The point of deletion is indicated by the position of the cursor.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">REPT</div>	Simultaneously depressing this key and any of the above keys will cause continuous repeat of the desired keystroke.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">VE STAT</div>	This key allows the user to enter the Memory Allocation Mode, check memory size, and then return to the Edit Mode by striking any other key in the main array.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">RETURN</div>	This key terminates the current program line if the cursor position is not within a string (inside quotation marks). It will also delete a program line when the line number is typed as a new line and terminated by two <b>RETURN</b> 's, e.g., 10 <b>RETURN</b> <b>RETURN</b> .

**TABLE 7-2**  
**EDIT COMMANDS AND KEYSTROKES (Cont'd)**

COMMAND/KEY	FUNCTION
	This key terminates the current program line and automatically numbers the next line.
	This key terminates a program line regardless of cursor position. It is normally used to terminate a line while editing inside a string (inside quotation marks).
	This key is used to exit the Edit Mode and enter the Level-3 Command Mode.
	This key is used to permit entry of lower case characters used in strings or as byte variables. Striking the key once permits entry of lower case characters. Striking the key a second time returns to the normal upper case entry mode. The status of the  key is shown on the top line of the Edit Mode display. As an aid in programming, each line of the program can be typed in lower case. When the line is terminated, it will be transformed to upper case and spaced automatically.
	This key will erase the program line being written regardless of cursor position.
DUMP	The DUMP command will output the current program via the XIO port (External IO) to a line printer, data recorder, etc..
DUMP <i>line #</i>	When the DUMP command is followed by a space and a line number, the remainder of the program, beginning at the specified line number, is output via the XIO port.
HELP	This command will allow the ENCORE to define an error message.
LIST	This command will list the first 10 steps of the program. The  and  cursor position keys are then used to scroll through the program.
	<b>Note:</b> Programs written in object code cannot be listed.
LIST <i>line #</i>	When LIST is followed by a space and a line number, 10 lines of the source program are listed beginning with the specified line number.
LOAD	This command clears source memory and inputs program instructions via the XIO port. After a program is loaded, it may be assigned a name using the Level-3 Command SAVE <i>program name</i>  .

**TABLE 7-2**  
**EDIT COMMANDS AND KEYSTROKES (Cont'd)**

COMMAND/KEY	FUNCTION
NEW	This command will erase the current ENCORE program from memory (source buffer).
RENUMBER	This command will instruct the ENCORE to renumber all program steps beginning with 10 and in 10 step increments. When the command is followed by a number (1-99), each step will be incremented by that number, i.e., RENUMBER 2 <input type="button" value="RETURN"/> . Type LIST to display the renumbered program. For debugging, execute RENUMBER to locate GOTO or GOSUB instructions referencing non-existent line numbers.
RUN	This command will execute the current program. To run any other program, you must first depress <input type="button" value="CMD"/> and then type the program name.
RUN!	This command instructs ENCORE to continually reallocate memory to the object buffer until the entire object program can be compiled. It is suggested that the program be fully debugged before using RUN! because this command causes compile without recording the line number on which an error occurred. This command causes less object code to be generated and therefore, faster program execution.
SAVE	Transfers the current source program to storage overwriting any program previously stored under the same name.

**2.11** The EDIT Mode is also used to obtain a complete and up to date listing of any non-proprietary program. This is accomplished by loading the desired program into the ENCORE source buffer and then LISTing the program to the CRT or DUMPing it to a hard copy device. A typical procedure to LIST via the CRT is given below:

STEP	PROCEDURE
1.	From the Level-3 Command Mode, type D MENU <input type="button" value="RETURN"/> and note a display of the disc menu.
2.	Locate the desired program and note its correct name.
3.	Strike the <input type="button" value="CMD"/> key to return to the Level-3 Command Mode and type D EDIT <i>program name</i> <input type="button" value="RETURN"/> . Note that the desired program is transferred to the source buffer (ENCORE enters the EDIT Mode).
4.	Type LIST <input type="button" value="RETURN"/> to list the program. A hard copy may be obtained, using the DUMP command, provided the correct XIO parameters are set and a hard copy device is connected to the XIO port.

## HELP

**2.12** The HELP command is designed as an aid to the user by providing a short explanation of the ENCORE error codes. For example, assume that the user has forgotten to include a hyphen in a screen position instruction and the ENCORE has responded by displaying "ERROR 10". Once the user recognizes "ERROR 10" as a syntax error, this display is sufficient to suggest corrective measures. However, when the user is not familiar with the error message, he simply types HELP  and an explanation of the error code is displayed, e.g., "SYNTAX ERROR". Chapter 9 of the manual includes a complete list and description of all the error codes.

## IO

**2.13** The IO Mode is designed to simplify operator selection of Input/Output parameters. It is a read only memory object program and cannot be edited. It is accessed through the Level-3 Command Mode by typing IO  or in Level-1 and Level-2 by menu selection. It may also be accessed through the COMBASIC program by executing the CHAIN "IO" statement. Upon entering the IO Mode, use the cursor position keys to change parameters. Striking the  key allows the user to enter separate parameters for each of the front ends (pin 2 and pin 3). Strike the  key to exit the IO Mode.

**NOTE:** The language and number of stop bits must be the same for both front ends.

**2.14 Speed.** Any speed from 10.0 up to 99 000 bits per second to three significant digits may be entered (five digits maximum). For speeds greater than 999 baud, key in spaces to delete digits or zeros as appropriate. For speeds under 999 baud, a decimal point must follow the units digit. The three keystroke sequences that follow illustrate the use of the space and decimal point when entering speeds:

```

100   baud = 1 0 0 SP 
75    baud = 7 5 . 
45.5  baud = 4 5 . 5 

```

**2.15 Sync.** The sync sequence may consist of one or two characters as may be required for synchronous operation. The normal sequence, also the power-up default, is SY SY. In a bisynchronous environment employing a HEX 55 leading pad, the sequence should be changed to  SY. To enter the sync sequence in HEX, simply strike the  key, enter the sequence in HEX, and strike the  key again to continue normal entries.

**NOTES:** 1) All characters are converted to the language and parity specified. 2) HEX entries are not altered.

**2.16 NRZI.** This is an IBM encoding technique (Non-Return-to-Zero-Inverted) used in an SDLC environment when a terminal must provide its own timing. It provides a signal change any time a binary 0 is received, but not when a binary 1 is received. Strings of 0's provide transitions used for receive synchronization. Strings of 1's cause zero insertion which again provides synchronizing transitions. NRZI selections are only made in the SYNC, SDLC, or TRAN modes.

**2.17 Mode.** The standard ENCORE operates in any one of four transmission modes: SYNC, SDLC, TRAN, and ASYN. For synchronous (SYNC) operation, the ENCORE employs an external clock and uses a sync sequence for data synchronization. When SDLC (Synchronous Data Link Control) is selected, the ENCORE is capable of framing data blocks and calculating

the Frame Check Sequences in compliance with the SDLC line protocol. In the transparent mode (TRAN), an external clock is used for bit synchronization and unsynchronized data is logged into the capture buffer (use 8-bit, no parity). In Level-1 and Level-2 display modes, captured transparent data can be shifted, by the operator, one bit at a time until it becomes intelligible. For asynchronous operation (ASYN), the intelligence bits are framed with a 2.0 unit stop pulse which the user may change to a 1.0 unit if desired. Operation in an asynchronous environment usually requires an internal clock.

**2.18 Stops.** Selection of a stop interval is only offered in the asynchronous mode. The user may choose either a 1.0 or 2.0 unit stop interval when external timing is provided, or 1.0, 1.5, or 2.0 when internal timing is selected.

**2.19 Clock.** The choice of an internally or externally derived timing device is offered in all modes. This is especially useful during operation as a terminal in a synchronous environment when external timing is unavailable or suspected of being incorrect.

**2.20 Parity.** Six choices are available for parity selection: odd, even, none, mark, space, and ignore. The correct choice is essential for several reasons: 1) In a synchronous environment, character synchronization will not be attained without proper parity; 2) Certain program mode instructions require the recognition of received characters. If the wrong parity is selected, the characters will never be recognized; 3) When displaying logged characters in their hexadecimal equivalent, selection of the wrong parity will change the HEX character; and 4) In some cases peripheral equipment will require that the parity-bit of all characters be the same logical state. Table 7-3 gives the number of bits and indicates whether a parity bit is included for each of the languages currently available.

**TABLE 7-3  
LANGUAGES WITH PARITY**

LANG	BITS	PARITY	TOTAL
ASCII	7	YES	8
BAUDOT	5	NO	5
BCD	6	YES	7
BCDIC	6	YES	7
EBCDIC	8	NO	8
EXT BCD	7	YES	8
IPARS	6	YES	7
SBT	7	YES	8
SELECT	6	YES	7
XS-3	6	YES	7
RMS	7	NO	7
RMS-2	7	NO	7
ASCII/KANA	7	YES	8



**2.21 Lang.** In its standard configuration, the ENCORE is equipped to operate in five different languages: ASCII, EBCDIC, SELECTRIC, BCD, and EXTERNAL BCD. Other languages including IPARS, XS-3, BAUDOT, SBT, FIELDATA, BCDIC, ASCII/KANA, RMS, and RMS-2 are available as options. No more than three optional languages can be supplied in any one unit.

**Typical IO Entry Procedure**

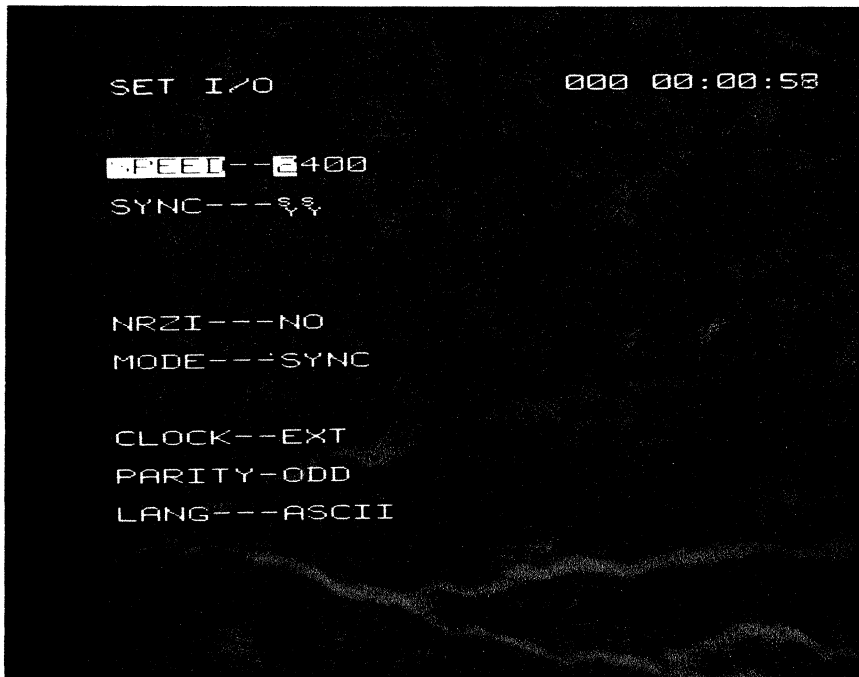
**2.22** The purpose of this procedure is to demonstrate, by example, the steps required for proper selection of all IO parameters. Figure 7-2 shows the power-up default parameters.

---

STEP	PROCEDURE
------	-----------

---

1. Upon entering the IO Mode, note a display similar to that shown below.



**Fig. 7-2 Power-Up Default IO**

2. Type in the desired speed entering spaces where digits are not required. Follow the last entry with the  key.
3. If SYNC is not displayed and a sync sequence is required, strike the  key until MODE is selected. Then strike the  key until SYNC appears. Strike the  key twice and enter the sync sequence followed by the  key. If SYNC is displayed and a sync sequence is not required, simply strike the  key.

STEP	PROCEDURE
4.	Select the appropriate response to NRZI by striking the <input type="checkbox"/> key. Strike the <input type="checkbox"/> key to advance.
5.	Select the operating mode by striking the <input type="checkbox"/> key until the desired mode is displayed. Strike the <input type="checkbox"/> key to advance.
6.	If operating asynchronously, select the stop unit interval by striking the <input type="checkbox"/> key until the desired interval is displayed. Strike the <input type="checkbox"/> key to advance.  <b>Note:</b> To choose a 1.5 unit stop interval, internal clock must be selected.
7.	Select the timing source by striking the <input type="checkbox"/> key until the desired source is displayed. Strike the <input type="checkbox"/> key to advance.
8.	Select the parity by striking the <input type="checkbox"/> key until the desired parity is displayed. Strike the <input type="checkbox"/> key to advance.
9.	Select the language by striking the <input type="checkbox"/> key until the desired language is displayed.
10.	These parameters now apply to both inputs (pins 2 and 3). To establish separate parameters for pins 2 and 3, strike the <input type="checkbox"/> key and note that the appropriate pin appears on the top line of the display. Strike the <input type="checkbox"/> key again to change from pin 2 to pin 3 or both (no pin number displayed). The method of entering parameters is the same regardless of which pin is selected.
11.	Strike the <input type="checkbox"/> key to exit the IO Mode.  <b>Note:</b> The <input type="checkbox"/> , <input type="checkbox"/> , <input type="checkbox"/> , and <input type="checkbox"/> keys may all be used to edit IO parameters.




## KILL

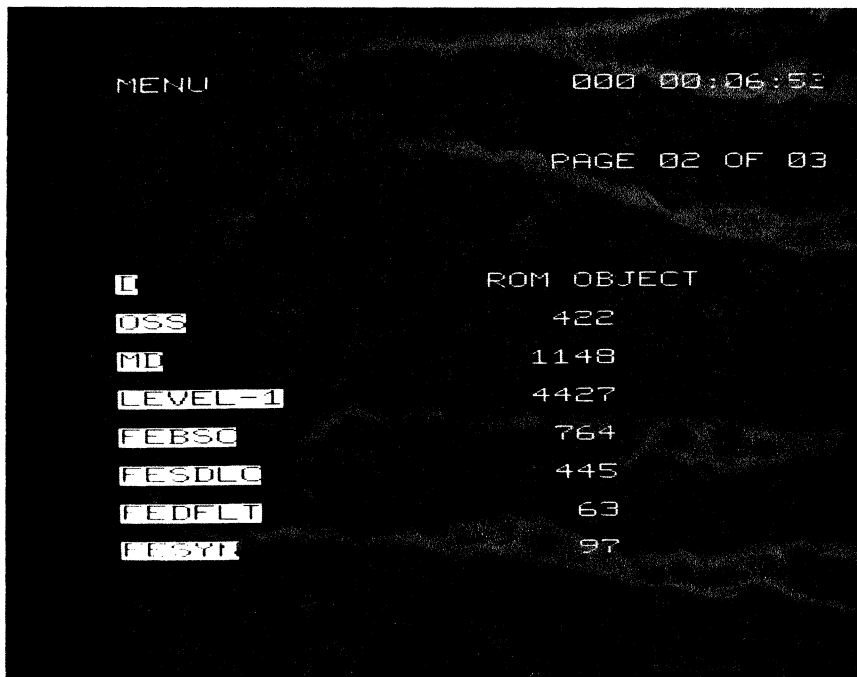
**2.23** The KILL command is used to erase programs from storage or disc, not from source. It always includes a program name, which is separated from the word KILL by a space, and is terminated with a carriage return.

- o **KILL *program name*.** This command erases a specified program from storage, not from source or disc. When successfully executed, the program name and the words "FILE GONE" are displayed. Any programs residing in storage are then shifted to avoid fragmentation and a new program index is written.
- o **D KILL *program name*.** This command performs the same function as described above except that the program to be erased must reside on disc.

## MENU

**2.24** The MENU command instructs the ENCORE to display a complete list of all programs in storage or on disc. As with all commands, it is terminated with a carriage return.

- **MENU.** This command displays the list of programs currently residing in storage, including system programs stored in ROM (Read Only Memory). The display, as shown in Figure 7-3, always includes the real time clock and the name and total number of source bytes required for each program. Programs labeled "ROM OBJECT" are available to the user only in their existing forms and cannot be listed or edited. The MENU display always begins with page 2 because page 1 contains unalterable ROM programs. Programs are added to or deleted from the MENU using the KILL and SAVE commands as described in paragraphs 2.23 and 2.26, respectively. The MENU normally consists of two or more pages as indicated in the display. To move from one page to another, use the  and  keys. Depress the  key to return to the Level-3 Command Mode.



The screenshot shows a terminal window with the following text:

```
MENU                                000 00:06:53
                                     PAGE 02 OF 03

  [  ROM OBJECT
  [  422
  [  1148
  [  4427
  [  764
  [  445
  [  63
  [  97
```

**Fig. 7-3 Typical Storage Menu**

- **D MENU.** This command performs the same function as MENU except that the display contains a list of the programs currently residing on disc. Up to 48 programs per page are displayed as shown in Figure 7-4.

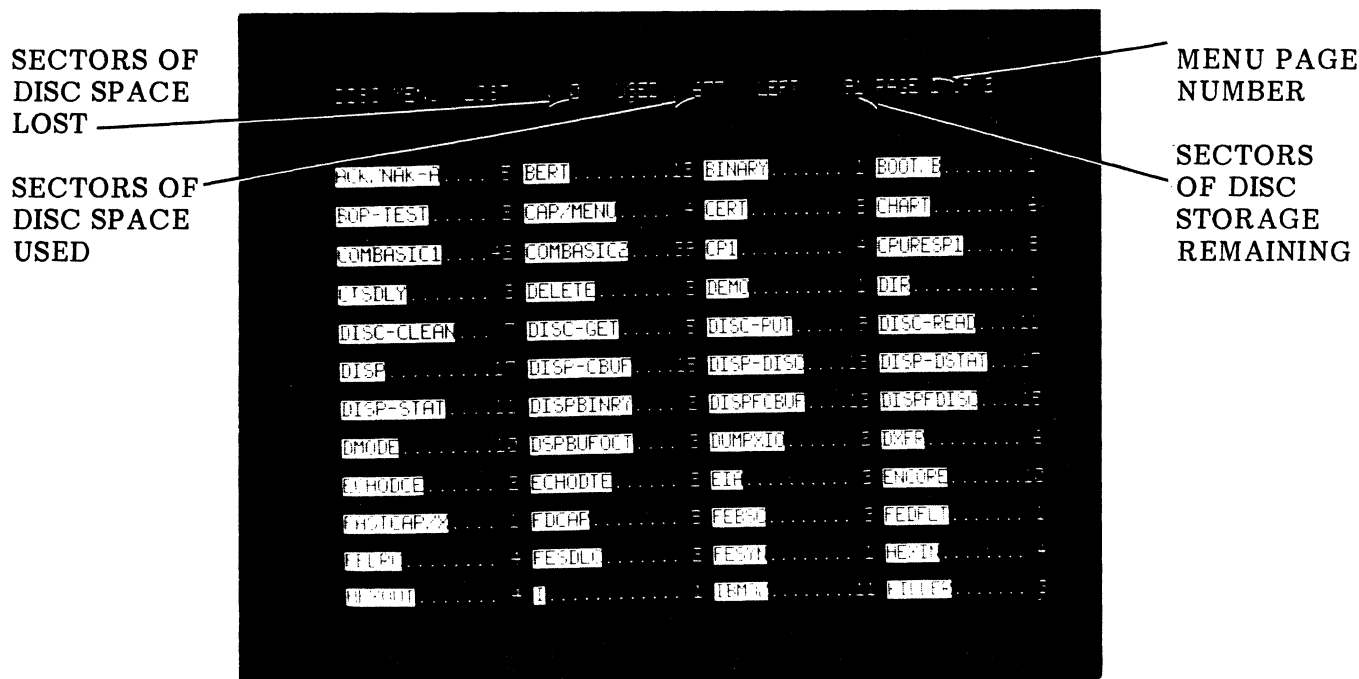


Fig. 7-4 Typical Disc Menu

## RUN

2.25 The RUN command is designed to execute programs. It is entered in several different forms and can be used to allocate both object and source memory. In some forms, the key word RUN is understood and need not be typed when the command is entered. The various forms of RUN are described below.

- **RUN.** When using this form of the RUN command, the program to be run must currently reside in the source memory and there must be sufficient object memory allocated to permit execution. If there is insufficient memory allocated, the ENCORE will display "ERROR #07", indicating that there is "NOT ENOUGH ROOM" to compile the program.
- **RUN!.** When using this form of the RUN command, the program to be executed must be stored in the source buffer. The ENCORE is then instructed to automatically reallocate memory from the capture buffer to the object buffer in 256 byte increments until the program compiles. In addition, 7 bytes of instruction overhead are dropped from the object code. These 7 bytes are normally used to identify the line number on which an error has occurred during processing. Because this information is dropped, the line number is not displayed with the error message. For this reason, programs executed with RUN! must be thoroughly debugged before execution. If RUN! is used and there is more than sufficient memory allocated, the object buffer will not be reduced, but the overhead is still dropped. RUN! may be used to execute a program using the least possible amount of memory.

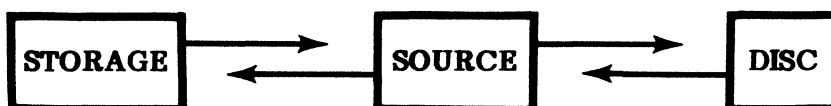
- **Program name.** This command transfers the specified program from storage to the source buffer and then compiles and executes it.

**NOTE:** Do not modify and execute an existing program until the modified program has been saved under a new name. If this command is used and the modified program has not been saved, the original program will be executed and all modifications will be lost.

- **Program name!.** This command transfers the specified program from storage to the source buffer and then allocates memory, compiles, and executes the program in the same manner as RUN!.
- **# program name!.** This command performs the same function as *program name* except that the object and source memories are first reduced to their minimum values of 4 096 and 256 bytes, respectively. Both memories are then incremented in steps of 256 bytes each until sufficient memory is allocated for edit and execution of the program.
- **D program name.** This command is used to execute a program stored on disc. It instructs the ENCORE to search the disc for the named program, transfer the program to source, and then execute it. A program with the same name already in source or storage will not be executed.
- **D program name!.** This command performs the same function described above except that the required amount of source and object memory are automatically allocated.
- **# D program name!.** This command performs the same function as *D program name* except that the object and source memories are first reduced to their minimum values of 4096 and 256 bytes, respectively. Both memories are then incremented in steps of 256 bytes each until sufficient memory is allocated for edit and execution of the program.

## SAVE

**2.26** The SAVE command is designed to provide permanent storage of programs in non-volatile storage memory or on a properly formatted disc. In either case, programs are only transferred to storage or disc via source memory as shown below.



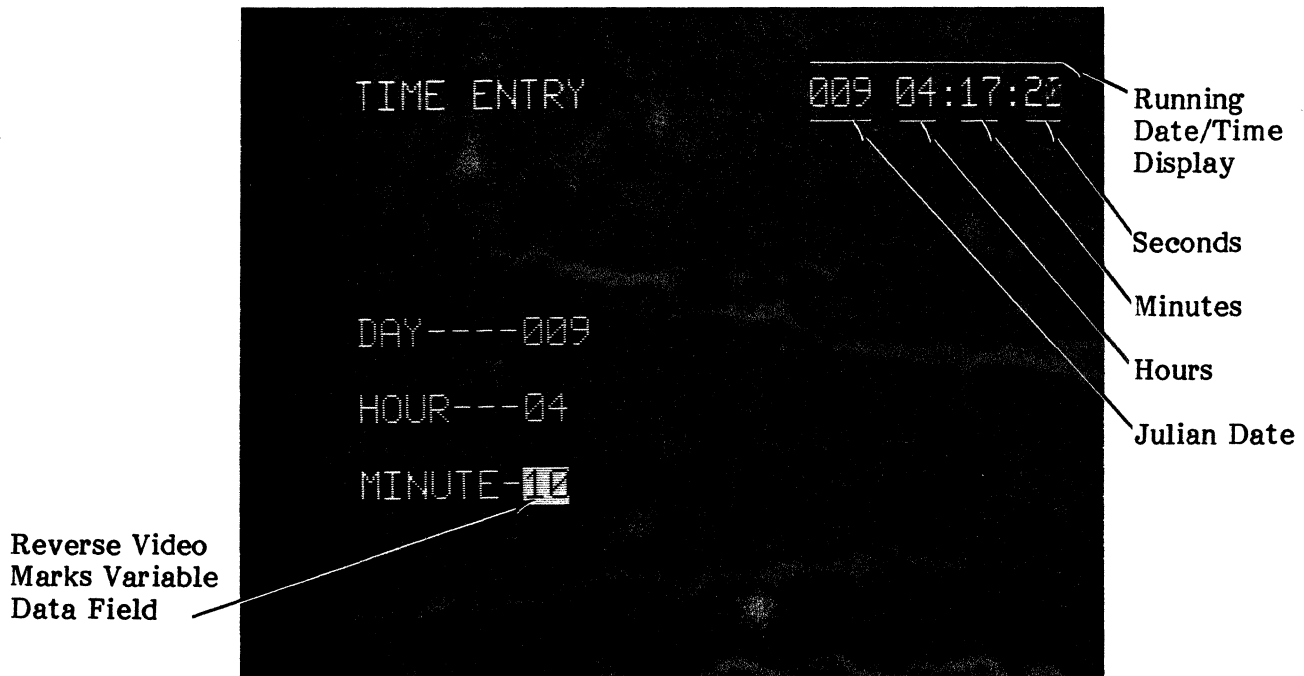
In addition, programs can only be placed in source through use of the EDIT commands as described earlier. As an example, assume that a program on disc is to be SAVE'd in storage. The user must first execute a disc edit command, e.g., *D EDIT program name*, to transfer the desired program from disc to source. Successful execution of this command then places the ENCORE in the Edit Mode where the program is now transferred to storage using the SAVE command. This places the program in storage under its original name. To save the program under a different name, the *SAVE program name* command is executed from the Level-3 Command Mode. The following paragraphs detail the various methods of executing the SAVE command.

- **SAVE.** This command transfers an already named program from source to storage and it is executed from the Edit Mode or the Level-3 Command Mode. It instructs the ENCORE to search the program index until the original version of the program is found. Delete the original version, move all subsequent programs forward in storage, store the revised version behind all other programs, and update the index. If the program is not in the index, it will be stored behind all other programs and the index will be updated. If the program is not named, the ENCORE will display the message "NO NAME".
- **SAVE *program name*.** This command is used to name or rename the program in source and transfer it to storage. If renaming a program, the original program is erased from storage using the KILL command as described earlier under paragraph 2.22. If the new program name is already used, the ENCORE will display the message "NAME USED". At this point, a new name must be chosen or the previously saved program with the same name must be erased. To save a new program under a previously used name, use the exclamation point (!) in conjunction with the SAVE command as described below.
- **SAVE *program name!*.** The addition of an exclamation point (!) to the SAVE *program name* command instructs the ENCORE to erase the original program from storage and transfer the current source program to storage under the same name. Do not use this form of the SAVE command unless you are sure that the previously stored program is no longer needed.
- **D SAVE.** This command performs the same function as SAVE except that the program to be saved is transferred from source to disc.

- **D SAVE *program name*.** This command performs the same function as SAVE *program name* except that the program to be saved is transferred from source to disc. The message "NAME USED" indicates the need to D KILL the previous program or repeat the command using an exclamation point (!) as shown below.
- **D SAVE *program name*!** This command performs the same function as SAVE *program name* except that the program to be saved is transferred from source to disc. Again, do not use this command unless you are sure that the program, bearing the same name on disc, is no longer needed.


**TIME**

2.27 The TIME Entry Mode allows the user to set the internal real time clock. TIME is a system program and cannot be edited. It is accessed from the Master Directory by selecting item  or from Level-3 by typing TIME . It may also be accessed from a COMBASIC program by executing the statement CHAIN "TIME". While in the TIME Entry Mode, the ENCORE display appears as shown in Figure 7-5. The  and  keys are used to direct the cursor to the DAY, HOUR, or MINUTE where the date and time are entered in numerical form and terminated by striking the , , or  key. The Julian date is the first entry required. It is made by typing the appropriate date (1 through 365), but any number from 0 through 999 is acceptable. The hour is entered by typing any number from 1 to 23. The minute is the last entry and is entered by typing any number from 0 to 59. When the desired date and time are entered, strike the  key to restart the clock and return to the Level-3 Command Mode.



**Fig. 7-5 TIME Entry Mode**

STATUS/MEMORY ALLOCATION/ 

2.28 The Memory Allocation or Variable Entry Status Mode is entered from the Level-3 Command Mode or the Edit Mode by striking the  key. This mode allows the user to view the present memory allocation and to change it if desired. Upon entering the Status Mode, the ENCORE first displays the status of the memory as shown in Figure 7-6. This display includes the total memory available, the amount used, and the amount left. The date and time are also shown.

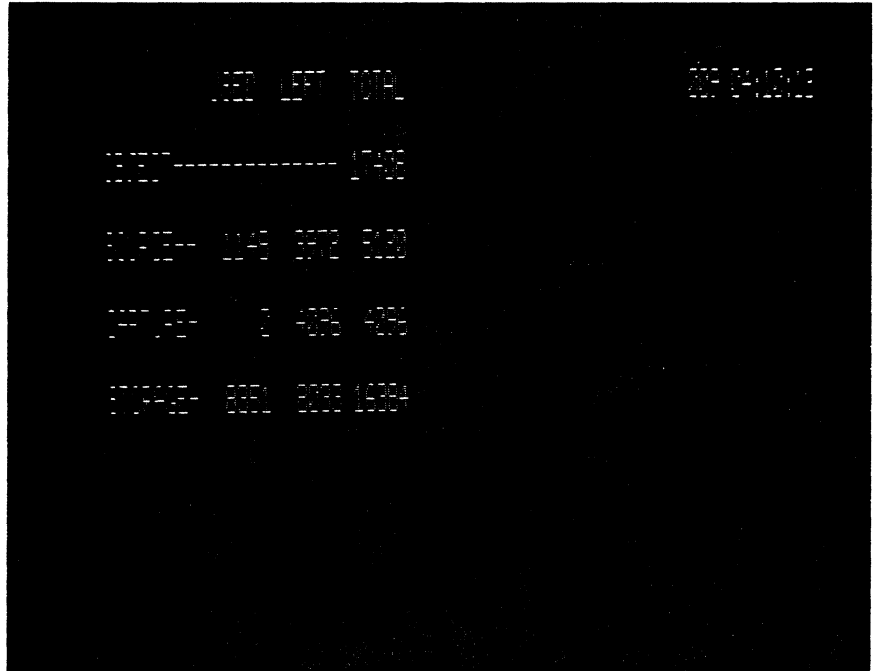



Fig. 7-6 Status Mode

2.29 While in the Status Mode, the user may manually reallocate memory by striking the  key again to obtain the Variable Entry display shown in Figure 7-7. In this mode, the cursor position keys are used to move from object to source and to increase or decrease memory allocation. The figure shows the default allocation made during power-up (no automatic memory allocation).

**NOTE:** When changing memory allocation in this mode, the source, object, and capture memories are effectively cleared.

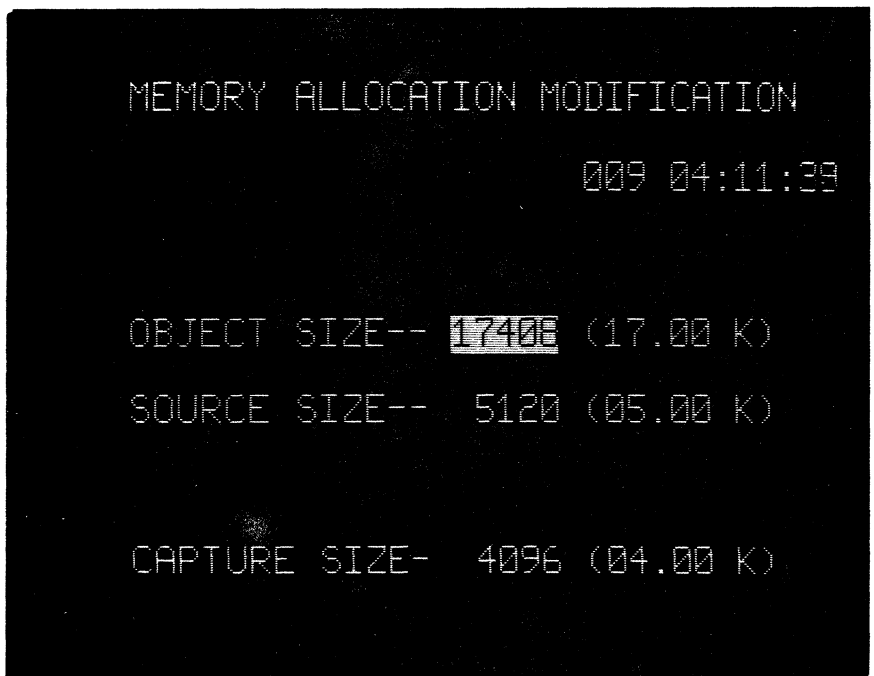


Fig. 7-7 Variable Entry Mode



**2.30** Automatic memory allocation, as mentioned earlier, allows the user to assign memory during the execution of the various EDIT and RUN commands. This is accomplished by including one or two special symbols in the command statement. The pound sign (#) is used to reduce the object and source memories to their minimum values. The exclamation point (!) is used to reallocate object and source memory to the minimum values required for execution of the desired program. To allocate specific amounts of memory, the pound sign is first followed by the object allocation and then by the source allocation. Some examples of acceptable syntax are shown below:

EXAMPLES:

RUN!

program name!

D program name!

# D program name !

Reallocate object and source buffers to minimum required to execute program.

# 16, 10, program name

Increase source buffer 17 increments (17 x 256) above minimum (256) allocation.

Increase object buffer 62 increments (62 x 256) above minimum allocation (4K).

Reduce object and source buffers to minimum allocations; Object = 4 096 bytes, Source = 256 bytes, and Capture = 22 272.

XIO

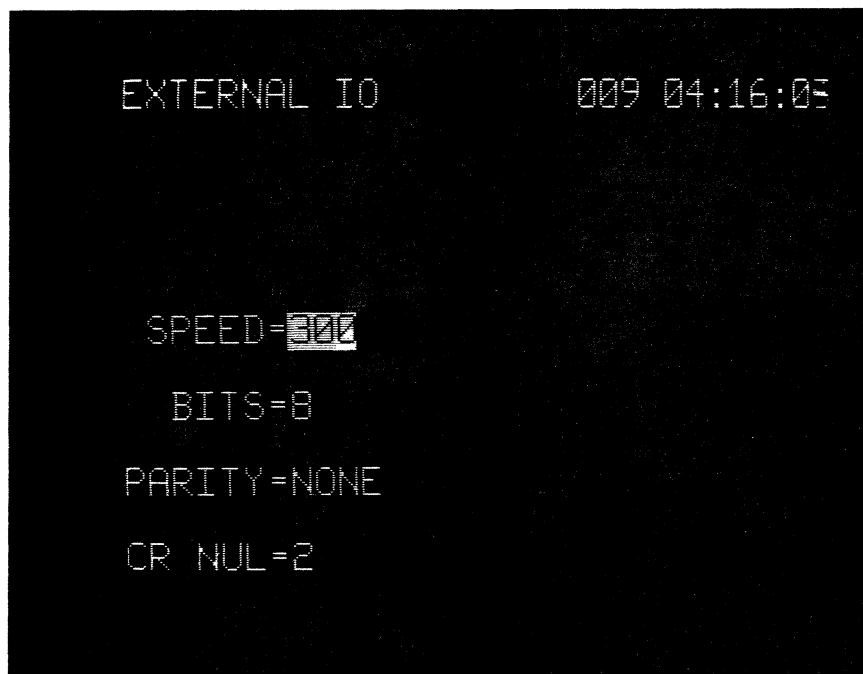
**2.31** The External IO Mode (XIO) is designed to permit operator selection of external IO parameters for interface with equipment connected to the rear panel XIO port (configured as DCE, data input on pin 2, output on pin 3). It is accessed in Level-3 by typing XIO or in Level-1 and Level-2 by menu selections. It may be accessed through the COMBASIC program by executing the CHAIN "XIO" statement. The XIO Mode allows the user to set external IO parameters to input/output programs while in the EDIT Mode using the LOAD or DUMP commands or to output data from the capture buffer under Level-1 program control. Programs are always output in ASCII. Data from the capture buffer is always output in the same language as captured (language set in IO) with the data byte set as shown in Table 7-4. While in the XIO Mode, the user selects any one of 16 operating speeds, shown in Table 7-5, where the data byte may consist of 5, 6, 7, or 8 bits, may have odd, even, or no parity, and may include from 0 to 9 null character period to permit actual carriage return of an external device. All entries are made by simply using the  and  keys to change cursor position and the  and  keys to change parameters. Once the correct parameters are established, striking the  key terminates the entries. A typical XIO Mode display is shown in Figure 7-8.

**TABLE 7-4**  
**XIO BIT AND PARITY SELECTION**

<b>LANGUAGE</b>	<b>BITS (Excluding Parity)</b>	<b>PARITY</b>
BAUDOT	5	NONE
IPARS	6	NONE
SELECTRIC	6	ODD or EVEN
BCD	6	ODD or EVEN
ASCII	7	ODD or EVEN
EXT BCD	7	ODD or EVEN
EBCDIC	8	NONE

**TABLE 7-5**  
**XIO SPEEDS**

50	150	1 800	4 800
75	300	2 000	7 200
110	600	2 400	9 600
134.5	1 200	3 600	19 200



**Fig. 7-8 XIO Mode Display**

### 3. COMBASIC PROGRAMMING

**3.01** The following paragraphs include general information of value to the user in writing COMBASIC programs. Immediately following this information are several example programs that serve to illustrate programming techniques and will help you to write your own programs. These examples are followed by the complete instruction set, listed alphabetically for easy reference. Throughout the instruction set, you will find many examples of COMBASIC statements showing the correct syntax used when entering an instruction. In some cases, these examples may run as short programs; in other cases, they only illustrate syntax.

#### PROGRAMMING RULES AND AIDS

**3.02** The following rules and aids will simplify the writing and debugging of your COMBASIC program. The rules always apply, the aids and hints are used at your discretion.

##### RULES

- Each program line must begin with a line number.
- No two lines may have the same line number.
- Program lines are executed in order by line number.
- Each line must include at least one COMBASIC instruction.
- Multiple instruction lines are permitted with the following restrictions:
  - 1) FOR/NEXT instructions cannot be included on the same line.
  - 2) WHEN \_\_\_ GOTO must be the last statement on the line.
- Each program line must be terminated using the  RETURN ,  LINE FEED , or  ^ key.
- Keyboard entry of spaces to separate line numbers, instructions, and variables is optional.
- Commas must be used to separate variables.
- String constants must be enclosed in quotation marks.
- Programs accessed by BOOT, CALL, or CHAIN must reside in storage.
- The maximum number of program lines is 320.
- A program name must begin with a letter (A-Z), contain no spaces, commas, exclamation points, or pound signs, and consist of no more than ten characters.
- Quotation marks within a string must be entered twice, e.g., " ""TEST"" ".
- String variables are automatically reinitialized upon program execution; integer, floating point, and byte variables are not and must therefore be reinitialized to avoid transfer from one program to another.

## COMBASIC PROGRAMMING, PROGRAMMING RULES AND AIDS, OPERATORS

### PROGRAMMING AIDS

- Use the REMARK statement whenever possible to describe significant events within the program.
- Increment line numbers by 10 to make room for future revisions. After line 10 is typed, subsequent line numbers may be generated automatically by terminating each line with `LINE FEED` instead of `^P` or `RETURN`.
- The `^P` key is used to terminate a line regardless of cursor position. The `LINE FEED` and `RETURN` keys are only used if the cursor is outside quotation marks.
- To edit an existing line, type the line number and `RETURN`. Strike the `RETURN` key twice to delete the line.
- Line numbering errors are pinpointed using the RENUMBER command.
- Unless memory is short, do not reuse variables.
- Whenever programming in a real time environment, consider the overhead processing time required to execute the COMBASIC instruction. Any values that can be established prior to interrogation of the data stream should be established in the beginning of the program. Also, when executing the program, use the RUN! or the *program name!* command. Instruction execution times are given in Appendix C.

### OPERATORS

**3.03** Operators perform mathematical or logical operations on numeric values, and, in some cases, on strings. COMBASIC operators include those which perform arithmetic, relational, and logical operations as shown in Table 7-6. The arithmetic and relational operators are detailed in the discussion of PRINT, LET, and IF statements, beginning on pages 7-87, 7-77, and 7-74, respectively. Logical operators are used only with the LET statement as shown in the following examples.

- **AND:** This operator instructs the ENCORE to perform a logical AND of two byte variables. Execute the example program given below, typing an upper case A and B in response to the prompts "ENTER a" and "ENTER b". Note that the results agree with the proof shown below. For an example of how to enter a COMBASIC program, refer to paragraph 4, EXAMPLE PROGRAMS, beginning on page 7-51.

EXAMPLE:

```
10 LET PARITY= IGNORE
20 LET LANG= "ASCII"
30 SCREEN CP=7-1"AND BYTE VARIABLES"LR"a AND b=c"LR
40 INPUT "ENTER a, ANY CHARACTER",a
50 SCREEN LR
60 INPUT "ENTER b, ANY CHARACTER",b
70 SCREEN LR
80 LET c=a AND b
90 PRINT %3.0, a, " AND ", b, " = ", c, " DECIMAL "
100 PRINT " ", [a], " AND ", [b], " = ", [c], " ASCII "
110 WAIT 5000
120 GOTO 30
```

SAMPLE RUN:

```

        AND BYTE VARIABLES
a AND b=c
ENTER a, ANY CHARACTER
ENTER b, ANY CHARACTER
65 AND 66 = 64 DECIMAL
A AND B = @ ASCII
    
```

**NOTE:** The LET statement may also be used to AND a byte variable with an integer with a value from one to 255 (LET c=a AND 66).

PROOF, LOGICAL AND:

	128	64	32	16	8	4	2	1
A=65	0	1	0	0	0	0	0	1
B=66	0	1	0	0	0	0	1	0
@=64	0	1	0	0	0	0	0	0

- **OR:** This operator instructs the ENCORE to perform a logical OR of two byte variables. Execute the example program below, entering an upper case A and B in response to the prompts "ENTER a" and "ENTER b". Note that the results agree with the proof shown below.

EXAMPLE:

```

10 LET PARITY= IGNORE
20 LET LANG= "ASCII"
30 SCREEN CP=6-1'OR BYTE VARIABLES"LR"a OR b=c"LR
40 INPUT "ENTER a, ANY CHARACTER",a
50 SCREEN LR
60 INPUT "ENTER b, ANY CHARACTER",b
70 SCREEN LR
80 LET c=a OR b
90 PRINT %3.0, a, " OR ", b, " = ", c
100 PRINT " ", [a], " OR ", [b], " = ", [c]
110 WAIT 5000
120 GOTO 30
    
```

SAMPLE RUN: (Enter A and B)

```

        OR BYTE VARIABLES
a OR b=c
ENTER a, ANY CHARACTER
ENTER b, ANY CHARACTER
65 OR 66 = 67
A OR B = C
    
```

**NOTE:** The LET statement may also be used to OR a byte variable with an integer (LET c=a OR 66).

PROOF, LOGICAL OR:

	128	64	32	16	8	4	2	1
A=65	0	1	0	0	0	0	0	1
B=66	0	1	0	0	0	0	1	0
C=67	0	1	0	0	0	0	1	1

- **RTL:** This operator instructs the ENCORE to take a given byte variable and rotate the bits comprising that variable to the right a specified number of times; no bits are dropped, regardless of the number of rotates. The number of bits moved to the right is indicated by the integer immediately following RTL (LET a=a RTL 3). In addition, the RTL argument can be used in combination with any of the logical operators (LET a=a RTL 3 AND a RTL 1). Execute the example program given below and note that the results agree with the proof.
- **RTR:** This operator is the same as RTL except that the bits are rotated to the left.

EXAMPLE:

```

10 SCREEN CP=3-1"ROTATE BITS TO THE RIGHT"LR
20 LET LANG= "ASCII"
30 LET PARITY= IGNORE
40 INPUT "ENTER a, ANY CHARACTER",a
50 SCREEN LR
60 PRINT "a =", a, " = ", [a]
70 LET a=a RTR 1
80 PRINT "a RTR 1 = ", a, " = ", [a]
90 WAIT 5000
100 GOTO 10
    
```

SAMPLE RUN: (Enter B)

```

ROTATE BITS TO THE RIGHT
ENTER a, ANY CHARACTER
a = 66 = B
a RTR 1 = 33 = !
    
```

PROOF, LOGICAL RTR:

	128	64	32	16	8	4	2	1
A=65=A	0	1	0	0	0	0	0	1
A RTR 1=160=	1	0	1	0	0	0	0	0
a RTR 2= 80= p	0	1	1	1	0	0	0	0
a RTR 3= 40= (	0	0	1	0	1	0	0	0

- XOR:** This operator instructs the ENCORE to perform a logical XOR of two byte variables. Execute the following example entering an upper case A and B in response to the prompts "ENTER a" and "ENTER b". Note that the results agree with the proof.

**EXAMPLE:**

```

10 LET LANG= "ASCII"
20 LET PARITY= IGNORE
30 SCREEN CP=6-1"XOR BYTE VARIABLES"LR"a XOR b=c"LR
40 INPUT "ENTER a, ANY CHARACTER",a
50 SCREEN LR
60 INPUT "ENTER b, ANY CHARACTER",b
70 SCREEN LR
80 LET c=a XOR b
90 PRINT %3.0, a " XOR ", b, " = ", c
100 PRINT " ", [a], " XOR ", [b], " = ", [c]
110 WAIT 5000
120 GOTO 30
  
```

**SAMPLE RUN:**

```

      XOR BYTE VARIABLES
a XOR b=c
ENTER a, ANY CHARACTER
ENTER b, ANY CHARACTER
  65 XOR 66 = 3
A XOR B =
  
```

**NOTE:** The LET statement may also be used to XOR a byte variable with an integer (LET c=a XOR 66).

**PROOF, LOGICAL XOR:**

	128	64	32	16	8	4	2	1
A=65	0	1	0	0	0	0	0	1
B=66	0	1	0	0	0	0	1	0
EX=3	0	0	0	0	0	0	1	1

TABLE 7-6

INSTRUCTION OPERATORS

INPUT	OPERATION	INPUT	OPERATION
<b>ARITHMETIC</b>		<b>LOGICAL</b>	
+	Add	AND	Logical "AND"
-	Subtract	OR	Logical "OR"
*	Multiply	XIO	Logical "XIO"
/	Divide	RTR n	Rotate to the right n bits
		RTL n	Rotate to the left n bits
<b>RELATIONAL</b>			
<	Less than		
=	Equal to		
>	Greater than		
< >	Not equal to		
= < or < =	Less than or equal to		
= > or > =	Greater than or equal to		

**NOTES:** 1) The logical operators are not used in the IF statement. 2) RTR and RTL include a wrap around of the carry bit.

**VARIABLES**

**3.04** Variables are names used to represent values in a COMBASIC program. There are four types of variables in COMBASIC: integer, floating point, string, and byte variables.

**A. Integer Variables**

**3.05** An integer variable, represented by any upper case letter A through Z, may contain any positive or negative whole number from zero through 32 767. Integer variables are used when working with relatively small numbers. They are stored within the Level-3 operating system RAM as two 8 bit bytes and are initialized to zero only during the power-up sequence or under program control. It is the programmer's responsibility to determine whether initialization is required and to include the appropriate steps within the program. Since initialization is under program control, it is possible to pass values from one program to another. The contents of integer variables may be moved to byte variables provided the value is between 0 and +255. The contents of any integer variable may be moved to a floating point variable regardless of the value of the integer variable. Table 7-7 lists those COMBASIC statements that may be used with integer variables. Additional information is found in the alphabetical listing of COMBASIC instructions beginning on page 7-61. The following are examples of integer variables and proper syntax used in the COMBASIC statements.



**EXAMPLE: Addition**

```
10 LET A=0, B=0, C=0
20 LET A=A+1, B=B+1, C=A+B
30 SCREEN Z
40 WAIT 1000
50 PRINT %5.0, C
60 GOTO 20
```

**EXAMPLE: Subtraction**

```
10 LET A=1000, B=500, C=0
20 LET C=A-B
30 PRINT %5.0, C
```

**EXAMPLE: Multiplication. The product must be stored in a floating point variable.**

```
10 LET A=1000, B=5
20 LET FA=A*B
30 PRINT %5.0, FA
```

**EXAMPLE: Division. The quotient must be stored in a floating point variable.**

```
10 LET A=1000, B=5
20 LET FA=A/B
30 PRINT %5.0, FA
```

**EXAMPLE: All math is performed left to right without regard to parenthetical notation.**

```
10 LET A=100, B=5, C=2
20 LET FA=A*B/C
30 PRINT %5.0, FA
```

**EXAMPLE: MOVE**

```
10 LET A=0, B=100
20 LET A=B
30 PRINT %5.0, A, B
```

**EXAMPLE: RELATIONSHIPS >, <, =, >, = <**

```
10 LET A=50, B=100, C=500, D=250
20 IF A+B < C-D GOTO 40
30 STOP
40 PRINT %5.0, A+B, C-D
```

TABLE 7-7

## COMBASIC STATEMENTS USING INTEGER VARIABLES

STATEMENT	COMMENTS
FOR A=B TO C NEXT A	In the FOR/NEXT loop, the value of A is incremented from the value of B to C.
IF A=B GOTO 100	If the value of A and B are equal, a conditional jump (GOTO or GOSUB) is executed.
INPUT A	The value of A is a numerical input from the keyboard. Use <input type="button" value="RETURN"/> to terminate.
LET A=1000	The value of A is set to 1 000.
LET A=B	The value of A is set equal to the value of B.
LET A=LEN A\$	A = the number of characters in the string.
ON A GOTO 90, 210, 430	A is equal to the positions following GOTO, e.g., 90 = position 1 (A=1), 210 = position 2 (A=2), etc..
PRINT %5.0, A	The value of A is printed with 5 digits before the decimal point and 1 digit after.
PRINT A\$(A, B)	A and B represent beginning and ending characters within the string.
READ A	The value of A is an integer read from the DATA statement.
SCREEN P=A-B	A and B represent screen positions. The A is character position, the B is line number.
SCREEN S=A-B	A and B represent the number of characters and number of lines that can be displayed. Variables are limited to the characters A through Y.
WAIT A	A = time in milliseconds.

## B. Floating Point Variables

**3.06** Floating point variables are represented in COMBASIC by an upper case F followed immediately by any upper case letter A through Z (FA-FZ). They may contain any positive or negative number from zero through .999999 E 37 and are primarily used for calculating with very large numbers. Floating point variables are initialized to zero only during the power-up sequence. It is, therefore, the programmer's responsibility to initialize the values of the floating point variables within the program. Since initialization is under program control, it is possible to pass values from one program to another. The contents of floating point variables may be moved to integer variables provided the value is between zero and 32 767 or to byte variables provided the value is a whole number between 0 and +255. Floating point variables are accurate to six significant digits. Table 7-8 lists those COMBASIC statements that may be used with floating point variables. Additional information is found in the alphabetical listing of COMBASIC instructions beginning on page 7-61.

### EXAMPLE: Addition

```
10 LET FA=0, FB=100, FC=100
20 LET FA=FB+FC
30 PRINT %5.0, FA, FB, FC
```

### EXAMPLE: Subtraction

```
10 LET FA=0, FB=50, FC=100
20 LET FA=FB-FC
30 PRINT %5.0, FA, FB, FC
```

### EXAMPLE: Multiplication

```
10 LET FA=-123, FB=1.23, FC=FA*FB
20 PRINT %3.3, FA, FB, FC
```

### EXAMPLE: Division

```
10 LET FA=0, FB=10, FC=3.2
20 LET FA=FB/FC
30 PRINT %3.3, FA, FB, FC
```

TABLE 7-8

COMBASIC STATEMENTS USING FLOATING POINT VARIABLES

STATEMENT	COMMENTS
IF FA=122.2 GOTO 100	FA may be set equal or not equal to another floating point variable, integer, or arithmetic expression.
INPUT FA	The value of FA may be input to any number between 0.000000 and <u>+0.999999</u> E 37.
LET FA=12.1288/2	FA may be set equal to an arithmetic expression or any number from 0.000000 to <u>+0.999999</u> E 37.
LET FA=MSTM	FA may be set equal to the value of the measurement timer.
LET FA=SPEED	FA may be set equal to the IO speed.
LET SPEED=FA	The IO speed may be set to the value of FA.
PRINT %2.4, FA	Used with PRINT %, the value of FA is printed in the specified format, e.g., 12.1234.

C. String Variables

**3.07** String variables are represented by an upper case letter immediately followed by a dollar sign (A\$-Z\$). String variables are used for storing alphanumeric information. The maximum length of a string variable is limited only to the amount of object memory available. At compile time, the starting and ending addresses for each string variable are established. Before using a string variable, it must be dimensioned by a DIM statement. At compile time, the DIM statement establishes the maximum number of characters that may be contained within the string. The actual length of a string variable during program execution is determined by the length of the data moved to the string. It is possible to select certain positions of a string by indexing with integer variables or constants. Table 7-9 lists those COMBASIC statements that may be used with string variables. Additional information is found in the alphabetical listing of COMBASIC instructions beginning on page 7-61.

EXAMPLES:

```

10 DIM A$=100, B$=100
20 LET A$="THE QUICK BROWN FOX"
30 PRINT A$
40 LET B$=A$ (1,9)
50 PRINT B$

```

```

10 DIM A$=100
20 LET A$="THE QUICK BROWN FOX"
30 LET A=1, B=1
40 PRINT A$ (A,B):
50 LET A=A+1, B=B+1
60 IF A$ (A,B)="X" GOTO 80
70 GOTO 40
80 PRINT A$ (A,B)
90 STOP

```

EXAMPLES: (Cont'd)

```
10 DIM A$=100
20 LET A=0
30 FOR T=1 TO 100
40 LET A=A+1
50 LET A$ (A,A)="X"
60 NEXT T
70 PRINT A$
```

**3.08** It is also possible to pass string variables from one program to another, but the user must be careful to observe the following techniques:

1. Use the CHAIN statement to execute the second and all subsequent programs to which the string(s) is transferred.
2. In each subsequent program, redimension the string(s), e.g., DIM A\$=100, B\$=200.
3. In each subsequent program, save the original string using the LET statement, .eg., LET A\$(18,18)="\*".

**3.09** The LET statement (item 3 above) is used to save the original string by resetting the string pointers in memory. Normally cleared during execution of the CHAIN statement, these pointers allocate original memory to the original string. As a result, the area in memory reallocated to the string is not overwritten during compile of the CHAINED program.

EXAMPLES:

PROGRAM A

```
10 DIM A$=100
20 LET A$="PROGRAM A MESSAGE"
30 LET A=LEN A$
40 CHAIN "B"
```

PROGRAM B

```
10 DIM A$100
20 LET A$(A,A)="*"
30 PRINT A$
```

OR

PROGRAM A

```
10 DIM A$=18
20 LET A$="PROGRAM A MESSAGE"
30 CHAIN "PROGMB"
```

PROGRAM B

```
10 DIM A$=18
20 LET A$(18,18)="*"
30 PRINT A$
```

**NOTES:**

- (1) Integer variables remain unchanged from one program to another unless re-initialized, e.g., A=0, B=0.
- (2) Use any character including "space" to save the desired string, .e.g, LET A\$(18,18)="\*", A\$(18,18)=" ", or A\$(18,18)="A".
- (3) The saving character ("\*", "space", "A", etc.) is also printed.

**NOTES:** (Cont'd)

- (4) The portion of the LET statement in parenthesis determines the number of characters to be printed from the new string and the size of the original string.
- (5) Dimension statements for transferred strings must be executed before the dimension statements associated with any new strings.

**TABLE 7-9****COMBASIC STATEMENTS USING STRING VARIABLES**


<b>STATEMENT</b>	<b>COMMENTS</b>
DIM A\$=10 or DIM A\$(10)	A\$ may not consist of more than 10 characters.
DISP SA\$	Search the capture buffer for the string A\$.
DISP IA\$	Search the capture buffer for the string A\$ ignoring parity.
DLC A\$=B\$C\$	Format the strings as required for one SDLC frame.
DLC A\$=B\$, C\$	Format the string required for two frames showing a flag.
DLC A\$= B\$,, C\$	Format the string as required for two SDLC frames using two flags.
IF A\$=B\$ GOTO 100	If the strings A\$ and B\$ are equal, a conditional jump (GOTO or GOSUB) is executed.
IF A\$="ABCD" GOSUB 100	If the string A\$ is equal to the string constant ABCD, a conditional jump (GOTO or GOSUB) is executed.
IF A\$(2,3)="BC" GOTO	If specified characters within a string (2,3) are equal to the string constant ("BC"), a conditional jump (GOTO or GOSUB) is executed.
INPUT A\$	Letters and/or numbers comprising A\$ are entered on the keyboard. The entry is terminated by striking the  key.
LET A\$="ABCD"	Set A\$ equal to the string constant "ABCD".
LET A\$="AB(C)D"	Set A\$ equal to the string constant "ABCD", but the third character's parity bit is incorrect.

TABLE 7-9

COMBASIC STATEMENTS USING STRING VARIABLES (Cont'd)

STATEMENT	COMMENTS
LET A\$=B\$(2,4)	Set A\$ equal to the second through the fourth characters of B\$.
LET A\$(2)="BCD"	Set A\$, beginning with the second character, to "BCD".
LET A=LEN A\$	Integer variable A is set equal to the number of characters in A\$.
LET A\$=RTC	Set A\$ equal to the value of the <u>Real Time Clock</u> .
LET A\$(2,4)= a to c	Set the second, third, and fourth characters of A\$ equal to byte variables a, b, and c.
LET A\$(1,5)=CBUF6(6,10)	Set the first five characters of A\$ equal to the sixth through tenth characters in the capture buffer.
LET A\$(1,5)=CBUFE(1,5)	Set the first five characters of A\$ equal to the last five characters of the capture buffer.
LET A\$(2,3)="BC"	Set the second and third characters of A\$ equal to B and C.
LET BCS ab=A\$, B\$	Set byte variables equal to the block check characters produced by the contents of A\$ and B\$.
PRINT A\$	Print the contents of A\$.
PRINT A\$(2,4)	Print the second, third, and fourth characters of A\$.
WHEN P2=" 3 "A\$ GOTO	Execute a conditional jump (GOTO or GOSUB) after the first three characters of A\$ are received.

**D. Byte Variables**

**3.10** Byte variables are represented by the lower case letters a through z and may contain any whole, positive number from zero through 255. Each byte variable is stored within the Level-3 operating system RAM in one 8 bit byte. Byte variables are used when working with small numbers. They can also be used to store any single character, since all characters have a decimal value between zero and 255. Byte variables are initialized to zero only during the power-up sequence. It is, therefore, the programmer's responsibility to initialize the values of all byte variables used within his program. Since initialization is under program control, it is possible to pass values from one program to another. The contents of byte variables may be moved to integer variables, floating point variables, and string variables. Whenever the byte variable is used with the PRINT statement, the decimal value (zero to 255) will be printed

unless the byte variable is enclosed in brackets. When enclosed in brackets, the ASCII character is printed. Table 7-10 lists those COMBASIC statements that may be used with byte variables. Additional information is found in the alphabetical listing of COMBASIC instructions beginning on page 7-61.

EXAMPLES:

```

10 LET a=0
20 FOR T=1 TO 128
30 SCREEN Z
40 PRINT %3.0, a, "=", [a]
50 WAIT 1000
60 LET a=a+1
70 NEXT T

10 DIM A$=128
20 LET a=0
30 FOR T=1 TO 128
40 LET A$(T,T)=a TO a
50 LET a=a+1
60 NEXT T
70 PRINT A$

10 LET X=0, a=X, B=2
20 PRINT a, [a], b, [b]
30 LET A=a, B=b
40 PRINT A, B
50 LET a=a-1, b=b-1
60 PRINT a, [a], b, [b]

```

TABLE 7-10

COMBASIC STATEMENTS USING BYTE VARIABLES

STATEMENT	COMMENTS
DISP C	Convert data in byte variable d to no parity ASCII.
DISP D	Display data and attribute as stored in byte variable d and a, respectively.
DISP F	Set byte variable o to 1 or 0 to correspond with the overwritten or non-overwritten state of the capture buffer.
DISP G	Get data and attribute at previously specified CBUF address, and place in byte variables d and s, respectively.
DISP H	Convert data in byte variable d to HEX.
DISP L	Shift byte variable d one bit to the left.
DISP P	Set byte variable a to binary 10000000 if parity of byte variable d does not agree with IO.
DISP R	Shift byte variable d one bit to the right.



TABLE 7-10

COMBASIC STATEMENTS USING BYTE VARIABLES (Cont'd)

STATEMENT	COMMENTS
DISP dBG	Set byte variable o to indicate data overwrite and/or data lost.
DISP SA\$	Set byte variable a to 0 after successful search of capture buffer for specified string, A\$.
DISP SAI\$	Same as above but parity is ignored.
IF a=b GOSUB	If the a is equal to b, a conditional jumper (GOTO or GOSUB) is executed. The byte variable may range in value from 0 to 255 and may be used with < or >.
IF a="A" GOTO	If a is equal to the character "A" of the IO language and parity, a conditional jump (GOTO or GOSUB) is executed.
IF a=A GOSUB	If a is equal to an integer variable (0-255), a conditional jump (GOTO or GOSUB) is executed.
IF a+2=b/2 GOTO	Byte variables may be used in an IF expression employing arithmetic operators and a conditional jump (GOTO or GOSUB).
IF a=32 GOTO	If a is equal to a decimal 32, a conditional jump (GOTO or GOSUB) is executed. The variable may range in value from 0 to 255 and may be used with < or >.
INPUT a	Keystroke is input as a byte variable in the IO language.
LET a=aLANG	Convert ASCII byte variable to the equivalent in the IO specified language.
LET a=aLPAR	Convert parity of the byte variable to agree with the IO specified parity.
LET a=aHEX	For BCD, IPARS, and SELECTRIC, reverses order of bits 0 through 6.

TABLE 7-10

## COMBASIC STATEMENTS USING BYTE VARIABLES (Cont'd)

STATEMENT	COMMENTS
LET a="A"	Assign the 8-bit value of "A" (11000001) to byte variable a.
LET a="(A)"	Assign the 8-bit value of "A" to byte variable a with incorrect parity (01000001).
LET a=A\$(A)	Assign the 8-bit value of the character within A\$ specified by the value of integer variable A.
LET a=DSTAT	Assign the disc status byte to byte variable a.
LET a="[FF]"	Assign the HEX value, FF, to byte variable a.
LET a=0B "11111111"	Assign the bit configuration, "11111111", to byte variable a.
LET a=0H "FF"	Assign the HEX value, FF, to byte variable a.
LET a=0O "377"	Assign the octal value, 377, to byte variable a.
LET a=255	Assign the decimal value, 255, to byte variable a.
LET A\$(2,4)=a to c	Set second, third, and fourth characters of A\$ equal to a, b, and c.
LET BCS ab=A\$	Set two character BCS for A\$ into a and b.
LET c=a AND b+2	Assign the value of the expression a AND b+2 to byte variable c. Used with all logical operators and arithmetic operators + and -.
LET XIOP=a	Set XIO parameters as specified in byte variable a.
ON a GOTO	Provide multiple branching to the line number whose order following GOTO corresponds with the value of the byte variable.
ON a-64 GOTO	Same as above using an expression and the operators + and -.
PRINT a	Print decimal value of a.
PRINT [a]	Print character represented by the 8-bit binary configuration of a.

**TABLE 7-10**  
**COMBASIC STATEMENTS USING BYTE VARIABLES (Cont'd)**

STATEMENT	COMMENTS
SCREEN Ga	Place byte at current screen position into byte variable a.
WHEN KBD a GOTO	Executes a conditional jump (GOTO or GOSUB) upon input of a byte via keyboard.
WHEN P2a	Input byte variable via pin 2.
WHEN P3a	Input byte variable via pin 3.
WHEN XIOIN a GOTO	Execute a conditional jump (GOTO or GOSUB) upon input of a byte via pin 2 of the XIO port.

**E. Reserved Variables**

**3.11** There are certain COMBASIC statements that, when executed, assign values to a series of variables. These variables must be reserved whenever the associated statement is used in a COMBASIC program. If the statement is not used, the variables need not be reserved. As an example, the byte variables d and s must never be assigned values when the DISP G statement is used. DISP G automatically assigns these variables the value of the data and status bytes at the memory address specified by the value of integer variable A. Integer variable A is also reserved in this situation as it is always assumed to contain an address in the capture buffer. Table 7-11 lists all variables that must be reserved when the associated COMBASIC statement is used.

**TABLE 7-11**  
**RESERVED VARIABLES**

VARIABLE	STATEMENT	ASSIGNMENT
a	DISP G	Attribute byte.
	DISP P	Attribute byte.
	DISP D	Attribute byte.
d	DISP G	Data byte.
	DISP P	Data byte.
	DISP D	Data byte.

TABLE 7-11

RESERVED VARIABLES (Cont'd)

VARIABLE	STATEMENT	ASSIGNMENT
e	DISP dR	Set to zero if the end of data is reached.
	DISP dBS	Set to zero if the beginning of data is reached.
	WHEN ERROR	Set to the error number.
o	DISP F	Set to zero if the capture buffer is not overwritten.
	DISP dBG	Set to zero if the disc is not overwritten.
r	DISP dR	Set to zero if read errors occur during the transfer of data from disc to the capture buffer.
s	DISP G	Status byte with STATE STAT, NFAST. Status byte with STATE STAT, FAST. Attribute byte with STATE STAT, NFAST.
A	DISP G	Set to address of desired data and status bytes within the capture buffer.
E	WHEN ERROR	Set to source code line number of error.

**F. Other Variables**

**3.12** The COMBASIC language employs a series of additional numeric and string variables. The numeric variables are assigned values from the real time clock, measurement timer, and interrupt timer. The additional string and one more numeric variable, are assigned values for establishing IO parameters within the COMBASIC program. These variables are detailed under paragraphs headed: Real Time Clock, Timers and Triggers, and IO Parameters in COMBASIC.

**REAL TIME CLOCK**

**3.13** The Real Time Clock (RTC) is set in the TIME Mode. If the value of the RTC is needed for a COMBASIC program, its value may be assigned to a string variable, using the statement LET A\$=RTC. This places the entire date time group into the string. If the entire date time group is not needed, the desired information may be separated from the string using a

statement similar to LET B\$=A\$(5,9). The integers 5 and 9 in parenthesis indicate the beginning and ending characters within the string.

**EXAMPLE:**

```
10 DIM A$=12, B$=3, C$=2, D$=2, E$=2
20 LET A$=RTC
30 LET B$=A$(1,3)
40 LET C$=A$(5,6)
50 LET D$=A$(8,9)
60 LET E$=A$(11,12)
70 SCREEN C
80 PRINT "JULIAN DAY: ", B$
90 PRINT "HOUR: ", C$
100 PRINT "MINUTE: ", D$
110 PRINT "SECOND: ", E$
120 GOTO 20
```

**3.14** RTC is also used with the WHEN statement to detect a specified date time group. The date time group is placed in a string variable. Then, using a statement similar to WHEN RTC=A\$ GOTO 100, the user may transfer COMBASIC program control to a specified line number at a specified time. To cancel this statement, use ABORT RTC. For additional information, please see the discussion of string variables, and the explanation of the ABORT, LET, and WHEN statements included at the end of this chapter.

**TIMERS/TRIGGERS**

**3.15** As described in the preceding paragraph, the Real Time Clock is one timer that can be used as an event trigger, but there are three more timers; the measurement timer, the WAIT timer, and the WHEN timer.

**3.16** Using the CLEAR and LET FA-FZ instructions with the argument MSTM, the user can clear and start the ENCORE's measurement timer and then assign the value of the timer to a floating point variable. As a floating point variable, the contents of the measurement timer can be compared with another floating point variable to trigger an event. The number of triggers then, is limited only by the number of floating point variables available in the program. The measurement timer is designed to count from 0 to 8 388 608 ms. or 2 hours, 19 minutes, and 48 seconds in 1 millisecond intervals.

**EXAMPLE:**

```
10 STATE TERM
20 WHEN P5=1 GOTO 60
30 TON P4
40 CLEAR MSTM
50 GOTO 50
60 LET FA=MSTM
70 PRINT %5.2, FA
```

**NOTE:** The measurement timer requires approximately 1 millisecond to clear. Any attempt to read the MSTM in less than 1 millisecond may produce invalid results.

**3.17** The WAIT timer differs from the other ENCORE timers in that it halts further execution of the program until time out. However, data continues to be transmitted or received if initiated under previous instructions. The WAIT argument is entered in milliseconds as a whole number or integer variable. In addition, the WAIT argument may be set equal to the sum of the variables or integers when they are separated by commas, e.g., WAIT A, B, 1000.

## EXAMPLE:

```

10 SCREEN C "MEASURING RTS/CTS DELAY"
20 LET A=1000 STATE TERM
30 WHEN TIMER=5000 GOTO 50
40 GOTO 70
50 SCREEN C "NO RESPONSE AFTER 5 SECONDS"
60 ABORT TIMER WAIT A GOTO 10
70 WHEN P5=1 GOTO 100
80 TON P4 CLEAR MSTM
90 GOTO 90
100 ABORT TIMER LET FA=MSTM
110 SCREEN C PRINT %5.2, "RESPONSE = ", FA, " MS"

```

**3.18** The WHEN timer is designed to permit continued execution of the program, transferring program control to a specified line number when the time has run out. Although the WHEN TIMER statement may be used as often as needed within the program, it only sets one timer. Whenever the statement is executed, it sets or resets the timer as appropriate. It can be used with an argument entered as a whole number in milliseconds or as an integer variable. The statement ABORT TIMER is used to stop the timer.

## EXAMPLE:

```

10 SCREEN C "MEASURING RTS/CTS DELAY"
20 STATE TERM
30 WHEN TIMER=5000 GOTO 50
40 GOTO 60
50 SCREEN C "NO RESPONSE AFTER 5 SECONDS" GOTO 110
60 WHEN P5=1 GOTO 90
70 TON P4 CLEAR MSTM
80 GOTO 80
90 LET FA=MSTM
100 SCREEN C, PRINT %5.2, "RESPONSE = ", FA, " MS"
110 ABORT TIMER
120 SCREEN LR "STRIKE THE SPACE BAR TO REPEAT"
130 WHEN KBD a GOTO 10
140 GOTO 140

```

**3.19** Another WAIT argument, XDONE, is designed to assure complete output of a transmitted message before the program is allowed to continue. XDONE executes a wait until the front end transmitter has been loaded with the last two characters before proceeding to the next instruction. It also executes a wait of two additional character times before turning off any interface pins.

## EXAMPLE:

```

100 FOR T=0 TO X
110 SCREEN Ga
120 PRINT #2, [a]
140 WAIT XDONE
150 NEXT T

```

## **IO PARAMETERS IN COMBASIC**

**3.20** Although the IO Mode can be accessed in any level of operation, IO parameters can also be entered directly into a COMBASIC program. This eliminates the need for user intervention in setting up IO parameters. In the COMBASIC program, IO parameters are set using the LET statement. Parameters not set in the program assume the condition set in the IO Mode. In addition, parameters set in the program apply to both front ends unless the STATE P3 instruction immediately follows the parameters set for pin 2. In this case, a second set of parameters may be entered for pin 3. Multiple LET statements are permitted. For additional information, see the LET statement beginning on page 7-77.

### **EXAMPLES:**

```
100 LET FA=2400
110 LET SPEED=FA
120 LET SYNC="SY SY"
130 LET NRZI=NO
140 LET MODE=SYN
150 LET CLOCK=EXT
160 LET PARITY=NONE
170 LET LANG="ASCII"
```

### **OR**

```
110 LET SPEED=FA, SYNC="SY SY", NRZI=NO
120 LET MODE=SYN, CLOCK=EXT, PARITY=NONE
130 LET LANG="ASCII"
```

### **OR**

```
110 LET SPEED=FA, SYNC="SY SY", NRZI=NO
120 LET MODE=SYN, CLOCK=EXT, PARITY=NONE
130 LET LANG="ASCII"
140 STATE P3ONLY
150 LET PARITY=MARK
```

## **ATTRIBUTE AND STATUS**

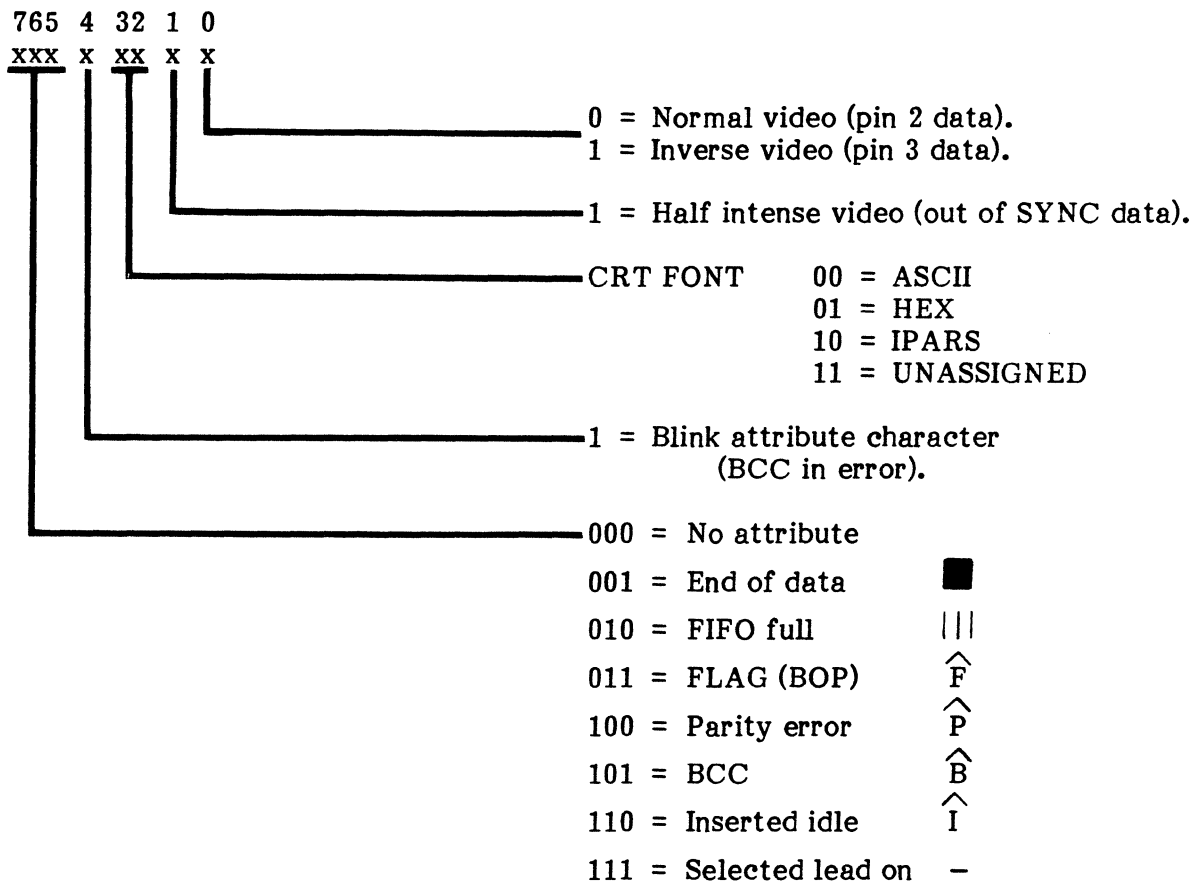
**3.21** The attribute and status bytes are used to define each data byte and the status of the interface as each data byte is captured. The user may, at his discretion, choose not to capture attribute or status, thereby increasing the speed at which the data alone may be input. The attribute byte is used to generate the display attributes shown directly beneath each character on the screen. The status byte is used to drive the RS-232 display under the same conditions. The captured information may include data only, data and attribute, or data and status as determined by use of the STATE instruction (see STATE STAT, NSTAT, and NFAST). Both attribute and status bytes are broken down as follows.

## IO PARAMETERS IN COMBASIC, ATTRIBUTE AND STATUS, Attribute Byte

### A. Attribute Byte

**3.22** The attribute byte, Figure 7-9, is comprised of 8 bits which provide the ENCORE with a variety of information concerning each character logged in the capture buffer. The attribute byte may be stored in the capture buffer, in lieu of the status byte, through use of the STATE instruction. Once the attribute byte is in memory, each bit can be examined using the AND, OR, XOR, RTL, and RTR arguments of the LET instruction to determine the characteristics of the data and ENCORE status. The display attributes are described in Table 7-12.

#### ATTRIBUTE BYTE BREAKDOWN



**Fig. 7-9 Attribute Byte**



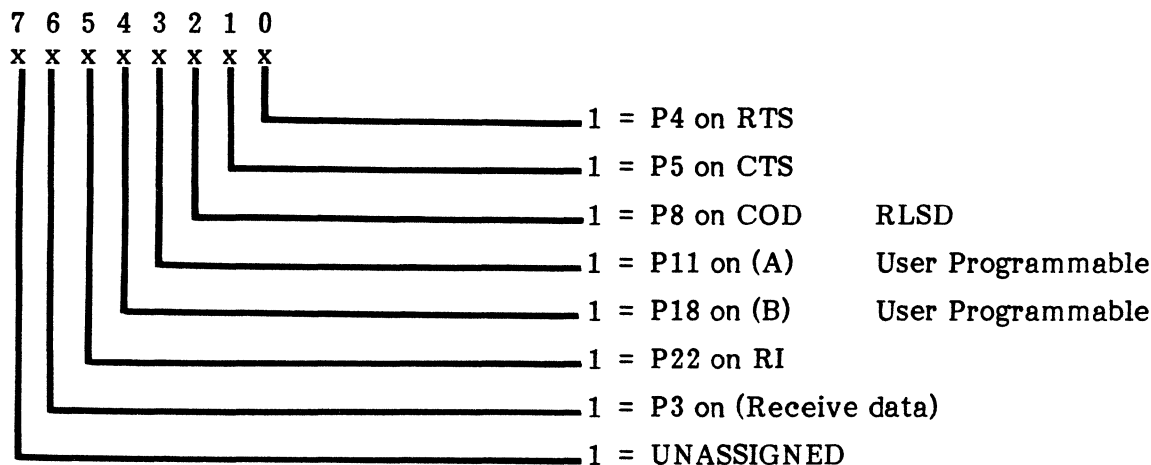
**TABLE 7-12**  
**DISPLAY ATTRIBUTES**

ATTRIBUTES	EXPLANATION
^P	This symbol indicates a parity flaw on any character in the data except those that are part of the block check sequence.
^B	This symbol indicates a block check character. If the symbol is blinking, the block check is bad.
^I	This symbol indicates inserted fill (by the ENCORE) for proper presentation of the full-duplex information.
^F	This symbol indicates a flag byte and is used only during SDLC operation.
-	This symbol is displayed when the program contains a STATE LEAD instruction. This instruction, when followed by an interface pin number, instructs the ENCORE to underscore data logged while the specified lead was on.
	This indicates that the FIFO (first in-first out) buffer is full. This buffer normally stores each incoming character (40 max) until it can be processed by the main CPU. If processing delays the input of data beyond forty characters, the     symbol is displayed, indicating that the buffer is full and that data has been lost.
■	This symbol indicates the last received byte of incoming data. Incoming data continually writes over old data on the screen.

**B. Status Byte**

**3.23** The status byte, Figure 7-10, like the attribute byte, is comprised of 8 bits which provide the ENCORE with information regarding the status of the RS-232 interface at the moment each character is input. The status byte may also be stored in the capture buffer, but only if the attribute is not.

**STATUS BYTE BREAKDOWN**



**Fig. 7-10 Status Byte**

**CAPTURE DATA CONTROL**

**3.24** Data may be captured and stored in the ENCORE's solid state memory (capture buffer or CBUF) or on disc. In either case, data is only input when the appropriate interface pins are turned on (TON P2B, P3B), and is always input to the buffer whether stored on disc or not. In addition to data, the CBUF may include the attribute or status bytes at the user's discretion. The size of the CBUF is dynamically allocated in 256 byte increments from a minimum of 4K to a maximum of 20K bytes.

**3.25** When data is to be stored on disc, it must be transferred via the CBUF. This requires execution of the STATE DISCON instruction to permit automatic transfer of data when the CBUF is full. Transfer from disc to buffer requires execution of a DISP dR for each 1K of data. The instruction, STATE DISCOFF must always be used to terminate transfer or the data will be lost. Tables 7-13 and 7-14 list the various instructions used to control data capture and display. A detailed explanation of each instruction is included in the alphabetical listing of COMBASIC instructions beginning on page 7-61.

TABLE 7-13

## COMBASIC STATEMENTS FOR CONTROL OF THE CAPTURE BUFFER

STATEMENT	COMMENTS
CLEAR CBUF	Erase contents of the capture buffer.
DISP B	Backup display of captured data or character.
DISP E	Set integer variable A to ending address of capture buffer (data or fill).
DISP F	Set byte variable o to zero upon overwrite.
DISP G	Put data and attribute in byte variables d and s.
DISP SA\$	Search capture buffer for A\$ and set a=0 if found.
DISP SIA\$	Same as above, but parity is ignored.
LET A\$(2,4)=CBUFB(6,8)	Put A\$ characters 2, 3, and 4 into the CBUF at character positions 6, 7, and 8.
LET A\$(2,4)=CBUFE(6,8)	Put A\$ characters 2, 3, and 4 into the CBUF at positions 6, 7, and 8, counting from the end of buffer toward the beginning.
PRINT #3	Transfer the remainder of the PRINT #3 statement to the CBUF.
STATE NSTAT, FAST	Only data is input to the CBUF, no status or attribute bytes.
STATE STAT, FAST	Data and status bytes are input to the CBUF, no attribute bytes.
STATE NSTAT, NFAST	No status bytes are input to the CBUF, only data and attribute.
TON P2B/P3B	TON is used to turn on pins 2 or 3 for input to the CBUF.
TOFF P2B/P3B	TOFF is used to halt CBUF input via pins 2 or 3.
WHEN FULL GOTO	Execute a conditional jump (GOTO or GOSUB) when the CBUF is full.
ABORT FULL	Cancel WHEN FULL.

TABLE 7-14

## COMBASIC STATEMENTS FOR DISC CONTROL

STATEMENT	COMMENTS
DISP dBG	Set disc head to beginning of first data record. Also set byte variable o to indicate data overwrite if applicable.
DISP dBS	Read previous data record and set byte variable e to indicate that the first record has been reached.
DISP dE	Set disc head to beginning of last data record.
DISP dR	Read data from disc to CBUF in 1K increments (four sectors).
LET a-z=DSTAT	Transfer disc status byte into specified byte variable.
STATE DISCON	Transfer CBUF data to disc in 1K increments when full. The appropriate pin must be turned on, e.g., TON P2B, P3B.
STATE DISCOFF	Mark end of data record and cancel DISCON. Must be executed to end data record or data is lost.

**BLOCK CHECK CALCULATIONS**

**3.26** The ENCORE performs redundancy checks as defined in Table 7-15. COMBASIC instructions available for use in performing these checks are shown in Table 7-16. A detailed explanation of these statements is included in the alphabetical listing of COMBASIC instructions beginning on page 7-61.

**TABLE 7-15**  
**BLOCK CHECK SCHEMES AND POLYNOMIALS**

STATEMENT	POLYNOMIAL
Calculator preset to all zeros	
STATE CRC = CRC 16	$X^{16} + X^{15} + X^2 + 1$
STATE CRC = CRC 12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$
STATE CRC = CRC 6 (IPARS)	$X^6 + X^5 + 1$
STATE CRC = LRC 7	$X^7 + 1$
STATE CRC = LRC 8	$X^8 + 1$
STATE CRC = CCITT	$X^{16} + X^{12} + X^5 + 1$
Calculator preset to all ones	
STATE FCS = CRC 16	$X^{16} + X^{15} + X^2 + 1$
STATE FCS = CRC 12	$X^{12} + X^{11} + X^3 + X^2 + 1$
STATE FCS = LRC 8	$X^8 + 1$
STATE FCS = CCITT	$X^{16} + X^{12} + X^5 + 1$

TABLE 7-16

## COMBASIC STATEMENTS TO USE FOR BLOCK CHECK CALCULATIONS

STATEMENT	COMMENTS
DLC A\$=B\$	Set A\$ equal to B\$ and include flags.
DLC A\$=B\$ C\$	Set A\$ equal to B\$ and C\$ and include one beginning and one ending flag.
DLC A\$=B\$, C\$	Set A\$ equal to B\$ and C\$ and include a beginning flag, a shared flag between B\$ and C\$, and an ending flag.
DLC A\$=B\$,, C\$	Set A\$ equal to B\$ and C\$ and include a beginning and ending flag for each string.
LET BCS ab=A\$	Perform a block check on A\$, setting the block check character into byte variables a and b.
WHEN P2/P3=BCC GOTO/GOSUB	Execute a conditional jump (GOTO or GOSUB) when the block check characters are input at the specified pin.
WHEN P2/P3=BCG GOTO/GOSUB	Execute a conditional jump (GOTO or GOSUB) when the block check characters input at the specified pin are good.
WHEN P2/P3=BCB GOTO/GOSUB	Execute a conditional jump (GOTO or GOSUB) when the block check characters input at the specified pin are bad.

Front End Instructions

CLR	Set BCC calculator to all zeros.
CMA	Complement each byte after input to register A.
HEX	Permit screen print in HEX.
INU 0-255	Input specified number of characters, updating the BCC calculator with each byte input.
IXU 0-255	Input specified number of characters, update BCC calculator and transfer bytes to main processor.

TABLE 7-16

## COMBASIC STATEMENTS TO USE FOR BLOCK CHECK CALCULATIONS (Cont'd)

STATEMENT	COMMENTS
<u>Front End Instructions</u>	
PRE	Preset BCC calculator to all ones.
UPD	Update BCC calculator for byte in register A.

**FRONT ENDS**

**3.27** The ability to write programs that will control both of the ENCORE's front end processors permits monitor or interactive communications using any existing or future protocol. Typically, the system default front end programs and parameters established in the IO Mode will handle most user applications. Additional front end programs, supplied on the program disc, are designed for unique applications in a synchronous, bisynchronous, or SDLC environment and can perform redundancy checks. These are automatically executed during Level-1 operation, but can be edited by the user to suit his unique applications. Wherever the existing front end programs are inadequate, the user has the capability of designing his own through the use of the front end instructions supplied in COMBASIC. The underlying purpose of the programmable front end processors is to free the host or main processor of the task of qualifying large amounts of data. This enables the host to concentrate on the processing of data that complies with the network protocol. The result is more efficient operation at higher speeds.

**3.28** The COMBASIC instruction set includes a series of instructions designed to control the front end processors. For most applications, the front end programs supplied on disc will suffice. They should be included, CALLED, or CHAINED at the beginning of your program. This is especially important if the CALL instruction is used, as it will compile the front end program to the first position of the object buffer, overwriting previously compiled instructions. Existing front end programs are easily recognized in the disc menu (D MENU), as they are prefixed FE, e.g., FESYN, FESDLC, etc.. Paragraph 3.29 explains the method used to execute a separate program for each front end. Paragraph 3.30 breaks down two of the most commonly used front end programs to show how each instruction is used. A detailed explanation of each front end instruction follows the alphabetical listing of all COMBASIC instructions at the end of this chapter.

**3.29** Whenever a front end program is compiled, it is transferred to both front ends. The statement, STATE P3ONLY, allows any front end programming that follows to overwrite the program already transferred to the pin 3 front end. When the P3 program has been compiled and transferred, STATE NP3ONLY prevents any further change in the P3 program. Any additional COMBASIC statements that affect the front ends will affect both the P2 and P3 programs, e.g., LET SPEED=FA, LET PARITY=ODD. The example below illustrates the use of the STATE instruction to permit CHAIN of two separate front end programs.

## EXAMPLE:

Line#	Instruction	Comments
10	CHAIN "FE#2"	Front end program "FE#2" is compiled and transferred to both front ends.
20	STATE P3ONLY	Permit modification of P3 front end program and IO parameters.
30	CHAIN "FE#3"	Front end program "FE#3" is compiled and transferred to the pin 3 front end processor, overwriting "FE#2".
40	STATE NP3ONLY	Prevent further modification of the program and IO parameters for pin 3 only. Both front ends are now affected by any changes in front end programming or IO parameters.

**3.30** Two front end programs, supplied on disc, are shown here as examples of how data can be handled before it is passed on to the main processor and from there to the capture buffer.

## EXAMPLE: FEDFLT

Line#	Instruction	Comments
20	IXO	Input 256 bytes, immediately transferring each one to the main processor.
30	JMP 20	Jump back to line 20 and input another 256 bytes.
40	CHAIN ""	Executed by main processor, not front end. Return to previous (calling) program.
		<b>Note:</b> Whenever the Level-3 Command Mode is entered, front end programs are overwritten by the system default program.

## EXAMPLE: FESYN

Line#	Instruction	Comments
20	RSN	Resync. Drop sync and search data for new sync sequence.
30	IXO	Input 256 bytes, immediately transferring each one to the main processor.
40	TDS 11111111	Skip next instruction if character input is not 11111111 (FF).
50	JMP 20	Jump back to line 20 and resync if FF is input.



EXAMPLE: FESYN (Cont'd)

Line#	Instruction	Comments
60	JMP 30	Jump back to line 30 and continue to input data with no resync until FF is received.
70	STATE NSYNC	Executed by main processor, not front end. Allows transfer of all data to front end, in sync or not.
80	CHAIN ""	Executed by main processor, not front end. Return control to previous program.

**EXECUTION TIMES**

**3.31** When writing COMBASIC programs that are RUN in real time, instruction execute time is an important consideration because it can be substantially limited by the data line speed. In other words, data may be input faster than it can be processed. The use of a 40 byte buffer between the front end and main processor helps to prevent this from occurring. The buffer is used to accumulate the input data, including status and attribute bytes generated by the front end processor, at the line speed, while the main processor accepts the data, from the buffer, at a rate determined by the execute time of the COMBASIC program. If the buffer overflows before data can be processed, a logically correct program will produce invalid results. To prevent this from happening, begin the program with all statements that can be executed once, and in a non-real time environment. These non-real time or initializing statements include such instructions as STATE and DIM, and in some cases, LET, SCREEN, etc.. The STATE instruction is especially useful as shown in the following examples:

EXAMPLES:

10 STATE	FLTR= FF	(eliminates idle line FF)
20 STATE	SYNC	(pass only in-sync data to the main processor)
30 STATE	NSTAT	(delete status byte)
40 STATE	FAST	(delete attribute byte)
50 STATE	HDUX	(interleaved display mode)

**OR**

10 STATE FLTR= FF, SYNC , NSTAT , FAST , HDUX

**3.32 Timing Chart:** Appendix C includes a list of typical execute times for each type of COMBASIC instruction. These instruction times are used to determine the execute time for a given program and are compared with the available time as determined by the line speed. For example, assume that synchronous EBCDIC data is input at 9 600 baud. The period, in milliseconds, of a single 8-bit byte is then equal to  $8/9\ 600 \times 1\ 000$ , or 0.833 ms. All real time instructions must, therefore, be executed within that period in order to process the next byte as it is input. However, the input buffer can only provide a delay of up to 40 bytes. If execution time accumulates beyond that allowed by the buffer, the COMBASIC program will be processing data in the wrong time frame and results will not be as expected.

**4. EXAMPLE PROGRAMS**

**4.01** The example program that follows will serve to familiarize you with the more commonly used COMBASIC instructions. It is suggested that you key in these programs and execute them to discover for yourself exactly how the instruction works. Additional examples are provided by examining the programs supplied on your program disc. This is accomplished using the EDIT commands described in paragraph 2.08. A more detailed explanation of individual instructions is found in the alphabetical listing of COMBASIC instructions beginning on page 7-61. For the purpose of these procedures, it is assumed that the user has entered the Level-3 Command Mode.

**SCREEN CONTROL**

**4.02** This procedure is designed to acquaint you with screen formatting instructions used in many COMBASIC programs. In addition, the procedure uses all of the program editing commands and will help you to develop successful programming techniques. The following COMBASIC instructions are used in this procedure.

FOR/NEXT    GOTO    INPUT    LET    PRINT    SCREEN    WAIT    WHEN

**SCREEN CONTROL**

STEP	PROCEDURE
1.	From the Level-3 Command Mode, type EDIT <input type="button" value="RETURN"/> and then NEW <input type="button" value="RETURN"/> upon entering the Edit Mode.
2.	Enter the following one line program shown below. Do not enter spaces following 10 and SCREEN, the ENCORE will do this for you.  10 SCREEN "TEST MESSAGE"
3.	After terminating the program line ( <input type="button" value="RETURN"/> ), strike the <input type="button" value="CMD"/> key and type SAVE SCREEN <input type="button" value="RETURN"/> . Notice that to assign a program name using SAVE, you must be in the Level-3 Command Mode. Be sure to include a space between SAVE and SCREEN.
4.	Type EDIT <input type="button" value="RETURN"/> and follow with LIST <input type="button" value="RETURN"/> upon entering the Edit Mode. Note that the program line is displayed.
5.	Type 10 <input type="button" value="RETURN"/> and move the cursor ( <input type="button" value="←"/> ) to the space following SCREEN.
6.	Strike the <input type="button" value="X"/> key 14 times and type: S=64-14P=20-6IU. Terminate these entries by striking <input type="button" value="RETURN"/> and note that the program line appears as shown below:  10 SCREEN S=64-14P=20-6IU"TEST MESSAGE"
7.	Type RUN <input type="button" value="RETURN"/> and note that TEST MESSAGE is displayed in the center of the screen, underscored, and in reverse video.
8.	Strike the <input type="button" value="CMD"/> key, type LIST <input type="button" value="RETURN"/> and note that the program is displayed.

SCREEN CONTROL (Cont'd)

STEP	PROCEDURE
9.	<p>Enter a second program line as follows:</p> <p>(a) Type 10 <input type="button" value="RETURN"/> and move the cursor to the 1 in the line number.</p> <p>(b) Strike the <input type="button" value="I"/> key and move the cursor to the S before the equal sign.</p> <p>(c) Strike the <input type="button" value="BK"/> key 7 times. Move the cursor to the 6 and type 7NH <input type="button" value="RETURN"/>. Your edited program should now appear as shown below:</p> <pre data-bbox="553 720 1170 779"> 10 SCREEN S=64-14P=20-6IU"TEST MESSAGE" 20 SCREEN P=20-7NH"TEST MESSAGE" </pre>
10.	<p>Type RUN <input type="button" value="RETURN"/> and note that TEST MESSAGE is displayed as before, but now the HEX equivalent of the message appears directly beneath it.</p>
11.	<p>Strike the <input type="button" value="CMD"/> key and then type SAVE <input type="button" value="RETURN"/>. Notice that the program is now saved while in the Edit Mode.</p>
12.	<p>Type LIST <input type="button" value="RETURN"/> to display the program once again.</p>
13.	<p>Expand the program to display the last HEX pair of the HEX message in its ASCII and decimal equivalents as follows. Enter a second and third program line as shown below. Terminate line 20 by striking the <input type="button" value="LINE FEED"/> key and line 30 by striking <input type="button" value="RETURN"/>.</p> <pre data-bbox="444 1230 732 1289"> 20 SCREEN P=31-7Ga 30 PRINT [a], a </pre>
	<p><b>Note:</b> The lower case character "a" is entered with the following keystrokes: <input type="button" value="BK ONLY"/> <input type="button" value="SCN A"/> <input type="button" value="BK ONLY"/>.</p>
14.	<p>Type RUN <input type="button" value="RETURN"/> and note display of the character E and its decimal equivalent on the line below TEST MESSAGE. The new line starts under the last E is MESSAGE.</p>
15.	<p>Strike the <input type="button" value="CMD"/> key and type NEW <input type="button" value="RETURN"/>.</p>
16.	<p>Enter a new SCREEN program as shown below:</p> <pre data-bbox="444 1677 873 1921"> 10 SCREEN SA 20 FOR C=1 TO B 30 SCREEN Z"TEST MESSAGE" 35 WAIT 500 40 NEXT C 50 SCREEN SZ 55 PRINT #5 60 GOTO 10 </pre>

SCREEN CONTROL (Cont'd)

STEP	PROCEDURE
17.	Type RUN <input type="button" value="RETURN"/> and note the following: <ul style="list-style-type: none"> <li>(a) TEST MESSAGE is displayed on the bottom line of the screen and is scrolled to the top.</li> <li>(b) Each time TEST MESSAGE is scrolled up one line, there is a ½ second pause.</li> <li>(c) Each time TEST MESSAGE reaches the top of the screen, the screen size changes and an audible alarm sounds.</li> </ul>
18.	Strike the <input type="button" value="CMD"/> key and then type SAVE <input type="button" value="RETURN"/> .
19.	Modify the program as shown to allow user input of the beginning screen size and program halt when the <input type="button" value="ESC"/> key is struck.

Line#	Instruction	Comments
10	SCREEN S=327	Sets screen size to permit display of the following prompts.
20	SCREEN P=00"SELECT THE DESIRED SCREEN SIZE"	Positions user prompts on lines 0 through 5 of the screen. Note that I in the SCREEN statement highlights the numbers 1 through 4.
30	SCREEN P=02I" 1 "N" 32 CHARACTERS / 14 LINES"	
40	SCREEN P=03I" 2 "N" 64 CHARACTERS / 14 LINES"	
50	SCREEN P=04I" 3 "N" 32 CHARACTERS / 7 LINES"	
60	SCREEN P=05I" 4 "N" 64 CHARACTERS / 7 LINES"	Halts further execution of the program until the keyboard is struck.
65	SCREEN P=06"STRIKE THE ""ESC"" KEY TO ESCAPE"	
70	WHEN KBD a GOTO 90	When the keyboard is struck, these two statements execute a jump to one of the SCREEN SZ statements or return to line 20 if the <input type="button" value="1"/> , <input type="button" value="2"/> , <input type="button" value="3"/> , or <input type="button" value="4"/> keys were not struck. 1, 2, 3, and 4 equal decimal 49, 50, 51, and 52, respectively.
80	GOTO 80	
90	ON a49 GOTO 120, 110, 140, 130	
100	GOTO 20	

SCREEN CONTROL (Cont'd)

STEP		PROCEDURE	Comments
Line#	Instruction		
110	SCREEN SZ		These statements increment the screen size to agree with the selection made and assures screen clear when item <input type="checkbox"/> is selected.
120	SCREEN SZ		
130	SCREEN SZ		
140	SCREEN C		
150	WHEN KBD a GOTO 240		This statement interrupts the program the next time the keyboard is struck and executes a jump to line 240.
160	SCREEN SA		This is the original program written in step 16 of this procedure, but renumbered here.
170	FOR C=1 TO B		
180	SCREEN A"TEST MESSAGE"		
190	WAIT 500		
200	NEXT C		
210	SCREEN SZ		
220	PRINT #5		
230	GOTO 160		
240	IF a <> 27 GOTO 180		
250	SCREEN C"PROGRAM TERMINATED"LR		
260	SCREEN "STRIKE ""SPACE BAR"" TO CONTINUE"		When any key is struck following execution of line 150, this statement checks the key and allows the program to continue unless the <input type="checkbox"/> key was struck. "ESC" equals decimal 27.
			This prompt is displayed if the <input type="checkbox"/> key was struck.

SCREEN CONTROL (Cont'd)

STEP		PROCEDURE
Line#	Instruction	Comments
270	WHEN KBD a GOTO 290	These statements allow the user to repeat the program by striking the SPACE BAR. SPACE BAR equals decimal 32.
280	GOTO 280	
290	IF a=32 GOTO 10	
300	GOTO 250	
20.	Type RUN <input type="button" value="RETURN"/> and note the program is executed according to the modifications incorporated in step 19.	
21.	Strike the <input type="button" value="CMD"/> key and type SAVE <input type="button" value="RETURN"/> to save the program in solid state memory.	
22.	Strike the <input type="button" value="CMD"/> key and type D SAVE <input type="button" value="RETURN"/> to save the program on disc.  <b>Note:</b> The disc write protect tab must be removed.	
23.	This completes the procedure and example program for screen control.	

CALCULATE

4.03 This procedure will demonstrate the techniques needed to take advantage of the ENCORE's calculating ability. It includes methods to perform calculations using integer, floating point, and byte variables using the LET and PRINT statements. It assumes that you are familiar with program editing techniques.

CALCULATE

STEP		PROCEDURE
Line#	Instruction	Comments
1.	From the Level-3 Command Mode type EDIT <input type="button" value="RETURN"/> followed by NEW <input type="button" value="RETURN"/> upon entering the Edit Mode.	
2.	Enter the following program. Do not enter spaces.	
10	LET A=2,B=3,a=5	Assign values to integer variables A and B and byte variable a.

CALCULATE (Cont'd)

STEP		PROCEDURE
Line#	Instruction	Comments
20	PRINT %2.0, "A=",A	Print A=2, allowing 2 integers and no decimal places.
30	PRINT "B=",B	Print B=3 in the same manner as above.
40	PRINT "a=",a	Print a=5 in the same manner as line 20.
50	PRINT "AXB+a=",A*B+a	Print the result of A multiplied by B+a.
3.	Type RUN <input type="button" value="RETURN"/> and note the desired results.	
4.	Strike the <input type="button" value="CMD"/> key to re-enter the Edit Mode. Type NEW <input type="button" value="RETURN"/> and enter the following program.	

Line#	Instruction	Comments
10	LET FA=2,FB=3,FC=FB/FA	Assign values to floating point variables FA, FB, and FC.
20	PRINT %2.3	Print results will permit display of 2 integers and 3 decimal places.
30	PRINT "FA=",FA	Print FA=2.000.
40	PRINT "FB=", FB	Print FB=3.00.
50	PRINT "FC=", FC	Print FC=1.500.
60	PRINT "FB/FA=", FC	Print FA/FB=1.500.
5.	Type RUN <input type="button" value="RETURN"/> and note how the use of the PRINT % statement affects the display of the floating point variables.	
6.	Strike the <input type="button" value="CMD"/> key to re-enter the Edit Mode and modify the program as follows:	

CALCULATE (Cont'd)

STEP		PROCEDURE
Line#	Instruction	Comments
10	LET FA=2,FB=3,FC=FB/FA	Assign values to floating point variables FA, FB, and FC.
15	LET D=FC	Add a new program line to assign the value of floating point variable FC to integer variable D.
20	PRINT %2.3	Print results will permit display of 2 integers and 3 decimal places.
30	PRINT "FA=", FA	Print FA=2.000.
40	PRINT "FB=", FB	Print FB=3.000.
50	PRINT "FC=", FC	Print FC=1.500.
55	PRINT "D=", D	Add a new program line to print D=1.  <b>Note:</b> That integer variable D does not contain the decimal value of FC. To print the decimal, the number must be assigned to a floating point variable.
60	PRINT "FA/FB=",FC	Print FA/FB=1.500.
7.	Type RUN <span style="border: 1px solid black; padding: 0 2px;">RETURN</span> and note that the integer variable D, when printed, does not include the decimal value .500.	
8.	This concludes the procedure and example programs to demonstrate calculations.	

DISPLAY DISC

**4.04** This procedure is designed to simplify use of the DISP instructions. It allows you to access data already stored on disc and display that data on the CRT. It assumes that a disc containing captured data is in the ENCORE disc drive and that you are familiar with program editing techniques.

**NOTE:** Data is easily captured using the Level-1 Monitor/Trap program.



DISPLAY DISC

STEP	PROCEDURE	
1.	From the Level-3 Command Mode, type EDIT <input type="button" value="RETURN"/> followed by NEW <input type="button" value="RETURN"/> upon entering the Edit Mode.	
2.	Enter the following program. Do not enter spaces.	
Line#	Instruction	Comments
10	SCREEN S=64-14	Set screen size for display of 14 lines of 64 characters each.
20	SCREEN SB	Set screen size into two consecutive intervariables, B and C. B = 64, C = 14.
30	DISP dBG	Position the disc read/write head to the beginning of the first data record and set byte variable o (see page 7-65).
40	DISP dR	Transfer 1K of data from the disc to positions 1 024 through 2 047 of the capture buffer.
50	DISP dR	Transfer the first 1K of data from positions 1 024 through 2 047 to 0 through 1 023 and a second 1K from the disc to positions 1 024 through 2 047. This fills the 2K capture buffer with data from the disc.
60	LET A=0	DISP G reads the value of integer A as the address of the data and status to be placed in byte variables d and s, respectively. 0 represents the beginning of data in the capture buffer.
70	SCREEN P=0-C	Position the cursor at the bottom left hand corner of the CRT, C = 14.

DISPLAY DISC (Cont'd)

STEP		PROCEDURE
Line#	Instruction	Comments
80	FOR X=1 TO C	Begin a for/next loop to display 14 lines of data on the CRT.
90	SCREEN Z	Move the cursor up one line from the bottom of the CRT. The instruction is placed here to permit display of a full 14 lines including the space that occurs between lines. Each time that it is executed, data is scrolled up one line.
100	FOR Y=1 TO B	Begin a for/next loop to display 64 characters on a single line.
110	DISP GD	This is a multiple DISP instruction to get the data and status bytes at the specified address (A), place those bytes in byte variables d and s, increment the value of A, and display the data and status beginning at the specified screen position.
120	NEXT Y	Complete the for/next loop to display 64 characters of data on a single line.
130	NEXT X	Complete the for/next loop to display a total of 14 lines on the CRT.
140	INPUT x	Halt further execution of the program until the keyboard is struck.
150	IF A<1 023 GOTO	Assure that all data transferred to buffer positions 0 through 1 023 is displayed before another 1K is transferred from disc.
160	GOTO 50	Return to step 50 and transfer another 1K from the disc.

DISPLAY DISC (Cont'd)

---

STEP	PROCEDURE
3.	Type RUN <input data-bbox="570 352 630 388" type="button" value="RETURN"/> and note that the disc is accessed (activity LED illuminated) and that data is first displayed at the bottom of the CRT and scrolls upward until the screen is filled (14 lines x 64 characters per line = 896 characters).
4.	Strike the space bar and note that an additional 14 lines of data are displayed. Note that each time the space bar is struck, another 14 lines of data are displayed.
5.	This completes the procedure and example program for display data stored on disc.

---

**5. COMBASIC INSTRUCTION SET**

**5.01** The following paragraphs describe the entire COMBASIC instruction set in detail. Examples are included with nearly every instruction to clarify the use of each keyword with its respective argument. With the exception of front end instructions, which are found on pages 7-101 through 7-104, the instruction set is listed in alphabetical order. The top outside corner of each page lists the instructions included on that page. A complete list of the entire instruction set is found in the Table of Contents. Each instruction also appears in the subject index.

**ABORT** This instruction is used to cancel any WHEN statements that are still in effect because the WHEN condition has not been met. If the WHEN condition has been met, it is automatically aborted. When any one of the four (maximum) WHEN conditions for pin 2 has been met, the remaining three WHEN conditions are also aborted. The only WHEN condition that cannot be aborted is WHEN ERROR.

**EXAMPLE:**

```

580 WHEN   KBD y GOSUB   740
590 WHEN   TIMER =5000 GOTO  700
600 WHEN   XIOIN a GOTO  620
610 GOTO  610
620 IF a <> 255 GOTO  650
630 SCREEN RL
640 GOTO  590
650 WHEN   XIOIN d GOTO  670
660 GOTO  660
670 ABORT   TIMER
680 DISP D
690 GOTO  590
700 ABORT  ALL
740 STOP
    
```

- ABORT ALL**        Aborts all WHEN statements except WHEN ERROR.
- ABORT FULL**     Aborts the WHEN FULL statement (when capture buffer is full).
- ABORT KBD**       Aborts WHEN KBD statement.
- ABORT PIN**       Aborts all of the following WHEN statements: WHEN P4, WHEN P5, WHEN P6, WHEN P8, WHEN P11, WHEN P18, WHEN P20, WHEN P22, WHEN PA, and WHEN PB.
- ABORT P2/P3**     Aborts all WHEN P2 or WHEN P3 statements.
- ABORT RTC**       Aborts the WHEN RTC statement.
- ABORT TIMER**     Aborts the WHEN TIMER statement.
- ABORT XIOIN**     Aborts the WHEN XIOIN statement.

# BOOT

**BOOT** This instruction allows the user to execute specific Level-3 commands from the program. These commands are part of the BOOT statement as shown in the examples. The BOOT argument must be enclosed in quotation marks and must include one @ character after the opening quotation mark, one to separate commands, and another @ before the closing quotation mark. The @ instructs the ENCORE to enter the Command Mode and after executing the specified command, to return to the BOOT statement. In addition, Special Function and Control Cluster entries are made using the keystrokes shown in Table 7-17. The example following the table instructs ENCORE to enter the Memory Allocation Mode and increase the size of object memory by two 256 byte increments and then return to the Level-3 Command Mode. The example also shows the control symbols that correspond with the desired commands.

**TABLE 7-17  
BOOT CONTROL ENTRIES**

SPECIAL FUNCTION		CONTROL ENTRY	SPECIAL FUNCTION		CONTROL ENTRY
UC LC	=	CTRL NUL ⓪	↓	=	CTRL HT I
HEX	=	CTRL SOH A	←	=	CTRL LF J
RWD	=	CTRL STX B	BRDG	=	CTRL VT K
STOP	=	CTRL ETX C	VE STAT	=	CTRL FF L
↑	=	CTRL EOT D	CLEAR	=	CTRL CR M
→	=	CTRL ENQ E	IC	=	CTRL SO N
UDF	=	CTRL ACK F	DC	=	CTRL SI O
CMD	=	CTRL BEL G	UC ONLY	=	CTRL DLE P
CONT	=	CTRL BS H	^ P	=	CTRL DCI Q

EXAMPLE: VE = CTRL FF L      ← = CTRL BRDG E

```

10 BOOT "@F@F@E@E@E@E@E@E@E@E"
20 CHAIN "@F@F@E@E@E@E@E@E@E@E"
  
```

**NOTES:**

- (1) SAVE the program before executing.
- (2) All of the Level-3 commands except RUN and D RUN can be executed using the BOOT statement.

**NOTES: (Cont'd)**

- (3) When using BOOT to load programs, precede the BOOT statement with the IF PROGRAM NAME statement to skip BOOT if the program is already in storage.

**EXAMPLE:**

```

10 IF "PROGRAM-1" GOTO 30
20 BOOT "D PROGRAM-1 CR @SAVE CR @"
30 CHAIN "PROGRAM-1"
40 STOP

```

- (4) The program containing the BOOT instruction must reside in storage.

**CALL**

When executed, CALL instructs the ENCORE to load the named program into the source buffer beginning at the location of the CALL instruction. The entire contents of the source buffer is then compiled to the object buffer as one program. When front end programs are called, they are always compiled to the first position of the object buffer and then transferred to both front end receivers. A front end CALL must, therefore, be the first statement of the main program. If it is not, previously compiled instructions will be overwritten by the front end instructions. Because the CALL statement instructs the ENCORE to compile the programs as one, the overall execution time of the program is less than it would be if CHAIN were used in place of CALL.

**NOTES:**

- (1) When using CALL, the two programs to be compiled must be in storage.
- (2) Although compiled as one program, the master program cannot execute a GOTO instruction that references a line number in the called program. The same conditions exist for the called program; it cannot GOTO a line number in the main program.
- (3) Programs in ROM cannot be CALLED.
- (4) Only one CALL statement may be used during each program.
- (5) A CALL statement must not transfer control to another program that also includes a CALL.

**EXAMPLE: Front-end-Call**

```

10 CALL "FEPROG"
20 STATE BRG , HDUX
30 TON P2 , P2B , P2D , P3 , P3B , P3D

```

**EXAMPLE: Called program (FEPROG)**

```

10 IXO
20 JMP 10

```

## CALL, CHAIN

### EXAMPLE: Master program CALL

```
10 STATE BRG , HDUX
20 SCREEN C "MAIN ROUTINE"
30 WAIT 1000
40 CALL "RTNE"
```

### EXAMPLE: Called program (RTNE)

```
10 SCREEN C "RTNE"
20 WAIT 100
30 CHAIN ""
```

**CHAIN** The CHAIN instruction is used to execute up to four programs residing in storage or on disc, from within a COMBASIC program. CHAIN may be used to execute the ROM object programs "IO", "ASGN", "TIME", and "XIO", and then return to the master program at the line number immediately following the CHAIN instruction. The name of the program to be executed by the CHAIN instruction must follow the word CHAIN and must be enclosed in quotes. Upon execution, the main program is first transferred from memory to the source buffer and then compiled to the object buffer. Whenever a CHAIN instruction is encountered, the program, named within quotes, is transferred from memory to the source buffer, compiled, and executed. When a CHAIN "" is encountered (no space between quotes), program control returns to the line number immediately following the previously executed CHAIN instruction. In this manner, it is possible to first CHAIN through four programs, (PROG-A, PROG-B, PROG-C, PROG-D), and then return back through the same programs, (PROG-D, PROG-C, PROG-B, PROG-A), to the master program. If a CHAIN "" is encountered and the CHAIN stack is empty, the ENCORE will return to the Level-3 Command Mode. To alter this return sequence, use the CLEAR CHAIN instruction to clear the entire CHAIN stack. The program can now CHAIN another program or CHAIN "" and return to the Level-3 Command Mode.

### NOTES:

- (1) The CHAIN stack contains the name of the program executed prior to the CHAIN instruction and the line number immediately following each CHAIN instruction encountered.
- (2) CHAIN instructions may be nested to four levels.
- (3) As each chained program is executed, its name and following line number are removed from the stack.
- (4) To execute a program containing the CHAIN instruction, the program must first be placed in storage or on disc using the SAVE or D SAVE commands.

## CHAIN, CLEAR, DATA/READ/RESTORE

### EXAMPLES:

```
10 CHAIN "IO"  
20 CHAIN "XIO"  
30 SCREEN C, "CHAIN EXAMPLE"  
40 WAIT 3000  
50 SCREEN C  
60 CHAIN ""
```

```
10 CHAIN "IO"  
20 SCREEN C, "CHAIN DISC PROGRAM EXAMPLE"  
30 SCREEN LR, "INSERT LEVEL-1 DISC"  
40 CHAIN "D PROGRAM-8"  
50 SCREEN C  
60 CHAIN ""
```

**CLEAR** This instruction is used in conjunction with the CHAIN, CBUF, MSTM, and RETURN arguments, as detailed in the following paragraphs.

**CLEAR CBUF** This statement clears the capture buffer by resetting the capture buffer pointer to zero. This allows the ENCORE to store data beginning with the first character location in the capture buffer memory. Upon execution, all previously stored data is lost.

**CLEAR CHAIN** This statement is intended to provide the programmer with the means of altering the return from one program to another in an order other than that established by the normal use of the CHAIN instruction. It cancels or clears the CHAIN stack completely. The programmer may then CHAIN "" or CHAIN "program name".

**CLEAR MSTM** This statement sets the measurement timer to zero and allows it to begin counting in ONE MILLISECOND increments to a maximum of 8 388 608 milliseconds or 2 hours, 19 minutes, and 48 seconds.

**NOTE:** Resolution of the measurement timer is one millisecond.

**CLEAR RETURN** This statement is similar to the CLEAR CHAIN statement, except that it cancels or clears the stack used for subroutines.

**DATA/READ/RESTORE** This instruction is used in conjunction with READ and RESTORE to provide data storage in the program. The READ instruction reads data sequentially beginning with the first item in the first DATA statement and ending with the last item in the last DATA statement. The RESTORE statement is used to reset the DATA pointer so the next READ statement will read the first item of data. If a READ statement is executed after the last DATA item has been read and a RESTORE has not been executed, the ENCORE will display "ERROR 27" (data error).

### EXAMPLE:

### SAMPLE RUN

```
10 FOR J=1 TO 3  
20 READ A  
30 PRINT %3.0, A;  
40 NEXT J
```

1	2	4	1	2	4
1	2	4	1	2	4



## DATA/READ/RESTORE, DIM, DISP

EXAMPLE: (Cont'd)

```
50 RESTORE
60 READ A
70 READ B
80 READ C
90 PRINT A, B, C
100 RESTORE
110 GOTO 10
120 DATA 1, 2, 4, 8, 16, 32, 64, 128, 256
```

### NOTES:

- (1) All DATA items must be in contiguous line numbers in the program.
- (2) Items in the DATA statement are entered as numeric constants and may be read as integer variables, floating point variables, or byte variables.
- (3) It is good practice to place DATA statements at the end of a program.

### DIM

This instruction consists of the key word DIM followed by a string variable and the maximum number of bytes to be reserved for that variable. Each string variable must be dimensioned before it is used in the program. If it is not dimensioned, it does not exist. The DIM statement may be written using the equal sign or parentheses as shown in the example.

EXAMPLE:

```
10 DIM A$=4096, B$=156, C$=3500, D$ (80)
```

**NOTE:** The maximum length of the dimensioned string is limited only by the amount of object buffer allocated. The dimensioned string value may exceed the actual length of the data to be stored in the string, but it must not be smaller.

### DISP

The DISP statement consists of the key word DISP followed by an argument that will control the display of captured data (data stored in the capture buffer or on a data disc). During the execution of a capture program, data may be captured with EIA status (STATE STAT, NFAST), with attributes (STATE NSTAT, NFAST), or as data only. Since it is impossible to determine which method was used by examining the data, the operator must assure that the same method is used for both the capture and display programs. If the method of capture and display are not the same, the final display will be incorrect. For a breakdown of the attribute and status bytes, refer to paragraphs 3.21 through 3.23.

### NOTES:

- (1) STATE NSTAT, FAST (data only). When using this statement, each byte in the capture buffer is treated as data. The DISP GPD statement then places the data byte into byte variable d and sets byte variable r to 0 and byte variable a to an octal 200 (10000000 binary) if the parity bit in d does not agree with the IO parity parameter. When displaying data, the EIA status (pins 2, 3, 4, 5, 8, 11, 18, and 22) and attribute byte cannot be recreated. However, parity can be determined by using DISP P.

**NOTES:** (Cont'd)**EXAMPLE:**

```
50 DISP GPD
```

- (2) STATE STAT, FAST (data and status). When using this statement, the first byte in the capture buffer (byte variable s) is always the EIA status and the second byte is always data (byte variable d). The attribute byte can be partially recreated using the example shown below. This example will create a parity error and display pin 2 data as normal video and pin 3 data as inverse video. EIA status is available in byte variable s. Any interrogation of the byte variables should be accomplished prior to the bit rotation and logical AND shown in the example.

**EXAMPLE:**

```
110 DISP GP
120 LET s=s RTL 2 AND 1, a=a+s
130 DISP D
```

- (3) STATE NSTAT, NFAST (data and attribute). When using this statement, the first byte in the capture buffer is always the attribute byte and the second byte is always data. The status byte cannot be recreated from the information contained in data and attribute bytes. The example routine shown below can be used to display the captured attributes. Pin 2 data is displayed as normal video and pin 3 as inverse video. Step 120 moves the captured attribute byte to the display attribute byte.

**EXAMPLE:**

```
110 DISP GP
120 LET a=s
130 DISP D
```

- DISP B** This statement instructs the ENCORE to back up the capture buffer pointer which is integer variable A, one character position. This has the effect of LET A=A-1. A will not be decremented past the beginning of the capture buffer (0).
- DISP C** This statement converts the data in byte variable d from the line language specified in the IO set-up to no parity ASCII. No conversion takes place if the line language is ASCII.
- DISP D** This statement instructs the ENCORE to display the data and screen attributes contained in byte variable d and a, respectively. This argument is usually part of a multiple DISP statement.

**EXAMPLE: (commas are optional)**

```
10 DISP G, P, D
200 DISP GPD
```

## DISP

**NOTE:** The status or attribute that was captured and placed in byte variable s must be used to supplement the attribute byte contained in byte variable a.

**DISP E** This statement sets integer variable A to the physical ending address of the capture buffer (not end of data).

**DISP F** This statement sets byte variable o to zero if the capture buffer has not been overwritten, and not equal to zero if it has been overwritten. This statement can be used in a program to warn the operator that some captured data may have been lost due to buffer overwrite during data capture.

### EXAMPLE:

```
10 DISP F
20 IF o=0 GOTO 50
30 SCREEN "CAPTURE BUFFER OVERWRITTEN"
40 STOP
50 SCREEN "CAPTURE BUFFER NOT OVERWRITTEN"
```

**DISP G** This statement instructs ENCORE to get the data and status bytes at a specific address in the capture buffer and place those bytes in byte variable d and s, respectively. The address is specified by the contents of integer variable A which must be set using the LET statement and cannot exceed the length of the capture buffer (0 to 1023 bytes if using disc or the limit specified in the Variable Entry Status Mode). Each time DISP G is executed, the value of integer variable A (capture buffer memory) is incremented by one. In the event that integer variable A addresses unused memory, the ENCORE will reset the variable to the highest address of captured data. To determine the address of the last character in memory, set A higher than the size of the buffer. Integer variable A will then contain the highest address after the next DISP G is executed.

### EXAMPLE:

```
10 DISP E
20 DISP G
30 LET B=A, C=A-1, A=0
40 PRINT A
```

**NOTE:** A contains the starting address. B contains the address of the next available position for data capture. C contains the address of the last captured data byte.

**DISP H** This statement instructs the ENCORE to convert the data byte variable d to its HEX equivalent. It is usually part of a multiple DISP statement. The data byte will be displayed in HEX with character registration maintained.

**EXAMPLE:**

CHAR		HEX
		1
<u>A</u>	=	<u>0</u>

- DISP L** This statement is similar to DISP R except that it shifts the contents of the entire capture buffer one bit to the left. The data in the buffer is then unintelligible until shifted back one bit to the right.
- DISP P** This statement instructs the ENCORE to set byte variable a to an octal 200 (binary 1000000) if the parity of the data in byte variable d does not agree with the parity set in the IO parameters. The attribute P appears beneath the parity flawed character when displayed by DISP D on the CRT.
- DISP R** This argument is similar to DISP L except that it shifts the contents of the entire capture buffer one bit to the right. The data in the buffer is then unintelligible until shifted back one bit to the left.
- DISP SA\$-SZ\$** This statement instructs the ENCORE to search the capture buffer for the contents of a specified string variable. When found, byte variable a is set to zero. Integer variable A is always used as the capture buffer pointer and cannot be used as a variable for any other reason when executing the DISP statement. Integer variable A will be set to the address preceding the address of the first character in the matched string.

**NOTES:**

- (1) When data is displayed on the CRT, the ENCORE uses the display attributes shown in Table 7-12 to identify specific data characters. The attribute can be displayed directly beneath the associated data character if it was captured.
- (2) When searching for a string variable, and the data captured was full duplex, the data will be in the capture buffer in an interleaved fashion; therefore, the string value must be set in an interleaved fashion in order to find a match.

**EXAMPLE:**

```

10 STATE  FUDX , NLEAD8 , NSTAT , NFAST
20 TON   P2 , P2B , P2D , P3 , P3B , P3D
30 CLEAR CBUF
40 WAIT  500
50 PRINT #1, "SySy1234567890ABCDEFGHI1234567890123"
60 WAIT 2500
70 STATE  HDUX
80 TOFF  P2 , P3
90 SCREEN C
100 DIM A$=10
110 LET A$="AABBCC"
120 LET A=30000
130 DISP G
140 LET B=A, A=0
150 FOR T=1 TO B
160   DISP G

```

## DISP

### EXAMPLE: (Cont'd)

```
170  DISP SA$
180  IF a=0 GOTO 210
190  NEXT T
200  GOTO 270
210  SCREEN C
220  FOR T=1 TO 6
230  DISP GPD
240  NEXT T
250  PRINT " MATCH FOUND AT", %4.0, A-10
260  STOP
270  PRINT A$, "NOT IN CBUF"
280  STOP
```

**DISP SIA\$-SIZ\$** This statement performs the same function as DISP SA\$-SZ\$ except that parity is ignored during search.

**DISP dBG** This statement instructs the ENCORE to position the read/write head at the beginning of the first captured data record. It also sets byte variable o to indicate an overwrite or lost data condition as follows:

<u>Byte Variable o</u>	<u>Program/Data Condition</u>
0	No data overwrite, no lost data.
1	Data overwrite, no lost data.
2	No data overwrite, data lost.
3	Data overwrite, data lost

**CAUTION:** SAVE'd programs overwrite previously captured data.

**DISP dBS** This statement instructs the ENCORE to read in the previous data record (backspace one record) and set byte variable e as follows:

<u>Byte Variable</u>	<u>Data Pointer</u>
e ≠ 0	Beginning of first data record reached.
e = 0	Beginning of first data record not reached.

**DISP dE** This statement instructs the ENCORE to place the read/write head at the beginning of the last data record.

**DISP dR** This statement instructs the ENCORE to read data into the capture buffer. Upon execution, 4 096 bytes of capture memory are allocated to the data record, 4K for use when writing data to the disc and 2K when reading data to the capture buffer. Each DISP dR moves the data in bytes 1 024 through 2 047 to bytes 0 through 1 023, and reads an additional 1K of data in bytes 1 024 through 2 047. The additional 2K of buffer memory used during write operations significantly reduces overhead and permits maximum thru-put at speeds of up to 125 kbps. Byte variables e and r are used to indicate the end of data and read errors when not equal to zero.

**NOTES:**

- (1) Byte variable e is not set to zero if the last record is read.
- (2) Byte variable e is set to zero if the last record is not read.
- (3) Byte variable r is set to zero if a data disc read is good.
- (4) Byte variable r is not set to zero if a data disc read error occurs.
- (5) Both byte variable e and r should be checked after each DISP dR.
- (6) All DISP disc instructions automatically allocate the capture buffer to 4K bytes. Any excess is allocated to the source buffer. It is important to note that the capture buffer is cleared whenever memory is reallocated.

**DISP T1** This statement sets a tab at every other character position on each line displayed. Beginning at position zero, tabs are then located at positions 0, 2, 4, 6, 8, 10, etc..

**DISP T2** This statement performs the same function as DISP T1, except that the tabs are set two character positions apart; i.e., 0, 3, 6, 9, 12, etc..

**DISP T4** This statement is similar to the other tab statements, except that the tabs are set four character positions apart; i.e., 0, 5, 10, 15, 20, etc..

**DISP T8** This statement is similar to the other tab statements, except that the tabs are set 8 character positions apart; i.e., 0, 9, 17, 25, 33, etc..

**DLC**

**DLC** This instruction is used in conjunction with string variables A\$ through Z\$ to format messages as required for SDLC (Synchronous Data Link Control) operation. DLC must be preceded by a STATE instruction to establish the method for calculating the frame check sequence. STATE CRC is used when the calculation must be preset to zero; STATE FCS is used when the calculation must be preset to ones. All flags except the beginning flag and end flag are inserted by placing commas between the string variables, one comma for each flag.

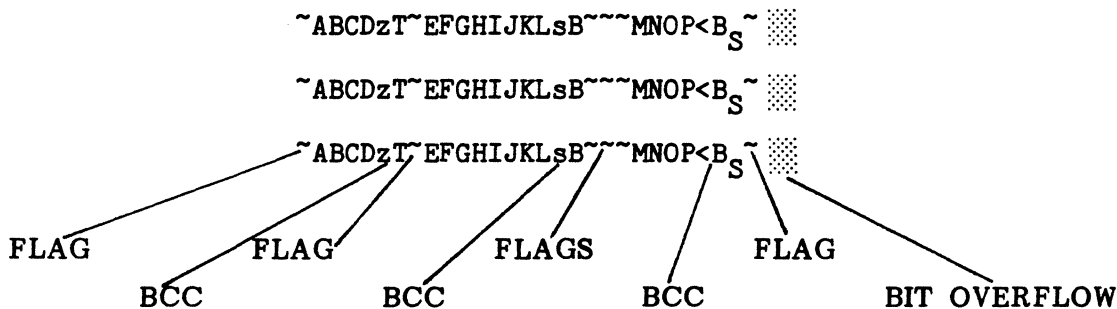
**DLC A\$-Z\$** The argument characters A\$ through Z\$ denote string variables. When used with DLC, the same string variable can never appear on both sides of the equal sign.

**EXAMPLE:**

```

10 STATE CRC= CRC16
20 STATE TERM , NLEAD8
30 CHAIN "IO"
40 REM SET MODE TO SDLC
50 REM SET PARITY TO NONE
60 REM DEPRESS "CMD" KEY TO EXIT IO MODE
70 TON P2 , P2D , P3 , P3D
80 WAIT 500
90 DIM A$=29, B$=4, C$=4, D$=4, E$=4
100 LET B$="ABCD"
110 LET C$="EFGH"
120 LET D$="IJKL"
130 LET E$="MNOP"
140 DLC A$=B$, C$D$,, E$
150 SCREEN C
160 PRINT A$
170 GOTO 160
  
```

**SAMPLE RUN:**



**NOTE:** Zero bit stuffing for all non-flag characters is also accomplished with this instruction. The receive string, to the left of the equal sign, must be large enough to hold inserted flags, as well as inserted zero bits, and FSC characters.

## FESYN, FOR/NEXT, GOSUB/RETURN

**FESYN** This is an instruction to initiate a new sync search (drop sync) in both front ends. It performs the same function as the front end instruction RSN. No argument is required.

**FOR/NEXT** This instruction creates a program loop that is repeated a specified number of times. Directly following the word FOR must be an integer variable whose value changes each time the loop is executed. The initial and final values of the running variable are set by the remainder of the argument which may be written as an integer or integer variable. The loop may be run in the stepped increments by adding the key word STEP and the increment value, i.e., FOR X=0 TO 60 STEP 2. The STEP increment must be a positive value.

### EXAMPLES:

```
10 LET a=0
20 FOR X=1 TO 128
30 PRINT [a];
40 LET a=a+1
50 NEXT X
60 FOTO 10

10 LET A=2, B=100
20 FOR X=A TO B STEP 2
30 SCREEN Z
40 PRINT %3.0, X
50 WAIT 1000
60 NEXT X
```

**GOSUB/RETURN** The GOSUB instruction transfers program control to the line number specified in the argument. The RETURN instruction transfers control back to the next line following GOSUB. Nesting up to eight levels of subroutines is permissible. CLEAR RETURN cancels the RETURN statement (clears RETURN stack). A GOTO statement is required in order to exit a subroutine after executing CLEAR RETURN.

### EXAMPLE:

```
10 SCREEN C"STRIKE THE ""ESC"" KEY TO OCNTINUE"
20 WHEN KBD a GOTO 40
30 GOTO 30
40 GOSUB 100
50 WAIT 1000
60 GOTO 10
100 SCREEN C"THIS IS YOUR ENCORE 200"
110 RETURN
```



## GOTO, IF

**GOTO *line number*** This instruction is used when an unconditional jump is required. It transfers control of the program to a specified line number. It must not be used to transfer program control inside a FOR/NEXT loop.

**IF** This instruction provides the ENCORE with conditional jump or branch capability. The branch is established through the use of a GOTO or GOSUB instruction and the appropriate line number if an expression is true (IF A=B GOTO 100). Operators used in the IF statement are listed in Table 7-6 on page 7-25.

### EXAMPLE:

```
10 IF "PROGRAM-1" GOTO 1000
20 IF FA=123 GOTO 1000
30 IF A$="ABCDE" GOTO 1000
40 IF A=B GOTO 1000
50 IF A<>B GOTO 1000
60 IF A>B GOTO 1000
70 IF A<B GOTO 1000
80 IF A=>B GOTO 1000
90 IF A=<B GOTO 1000
100 IF A+2-C=B+D-3 GOTO 1000
110 IF a="SH" GOTO 1000
120 IF a=27H GOTO 1000
130 IF A$=B$ GOTO 1000
140 IF FA>123.4 GOTO 1000
150 IF a=A GOTO 1000
160 IF A=FA GOTO 1000
```

**NOTE:** The above examples are all valid IF statements.

**IF A-Z** The upper case argument characters A through Z are used to denote integer variables. These variables may be used with all of the arithmetic operators but cannot receive the results of \* (multiply) and / (divide). The integer variable is limited to a value between 0 and +32 767. Numbers in excess of these values must be assigned to floating point variables.

### EXAMPLE: IF statement using integer variables

```
20 IF A=B GOTO 1000
30 IF A<>B GOTO 1000
40 IF A>B GOTO 1000
50 IF A<B GOTO 1000
60 IF A=>B GOTO 1000
70 IF A=<B GOSUB 1000
80 IF A+2-C=B+D-3 GOTO 1000
```

**IF FA-FZ** The upper case argument characters FA through FZ denote floating point variables which may be any number between 0.000000 and +0.999999 E 37. These variables may be used with all arithmetic operators. Floating point accuracy is up to six significant digits.

**EXAMPLE:**

```
10 IF FA=1493.7 GOTO 1000
20 IF FB=FC GOTO 1000
```

**IF a-z** The lower case argument characters a through z denote byte variables which represent any whole number between 0 and 255. The value of the byte variable may be entered as a decimal number, an ASCII character (quotes required), as another byte variable, or as an expression. Acceptable syntax is shown below. If you add +1 to +255, the result will be 0.

**NOTE:** <> cannot be used with "A".

**EXAMPLE:**

```
100 IF a="A" GOTO 1000
150 IF a=65 GOTO 1100
200 IF a=b GOTO 1200
300 IF a+21=65 GOTO 1300
400 IF a<>b GOTO 1400
500 IF a<>66 GOTO 1500
```

**IF A\$-Z\$** The characters A\$ through Z\$ denote string variables. Each variable may contain up to 46 characters which must be enclosed in quotation marks and dimensioned earlier in the program. To separate selected characters from a string, follow the string variable with the number of the beginning and ending characters in parentheses. To separate one character from the string, the beginning and ending character are the same.

**EXAMPLE:**

```
10 DIM A$=4, B$=4
20 LET A$="ABCD"
30 INPUT B$
40 IF B$=A$ GOTO 100
50 IF B$(4,4)="D" GOTO 200
100 STOP
200 STOP
```

**IF "program name"** This statement, used in conjunction with a conditional jump (GOTO or GOSUB), transfers program control to a specified line number if the named program is in storage. It normally precedes a BOOT or CHAIN statement which loads a program from disc if not already in storage.

## IF, INPUT

### EXAMPLE:

```
110 IF "OSS" GOTO 130
120 CHAIN "D OSS"
130 SCREEN CS=32-7P=2-0I"ENCORE MASTER DIRECTORY"
```

**INPUT** This instruction directs ENCORE to input data entered from the keyboard. Upon execution, the program halts until the required keyboard entry is made. The input may be an integer, integer variable, string variable, or byte variable. If enclosed in quotation marks, an operator prompt or other comments may be printed during execution immediately following the key word INPUT.

### INPUT A-Z

The upper case argument characters A through Z are used to denote integer variables. The integer variable is limited to a value between 0 and +32 767. Whole numbers in excess of this value must be assigned a floating point variable. Upon execution, a cursor is displayed on the CRT until a response is entered. All entries must be numeric and must be terminated by depressing the  key. A number greater than 32 767 will result in a "NUMBER TOO LARGE" error message at execution time.

### EXAMPLE:

### SAMPLE RUN:

```
10 INPUT "HOW MANY: ", A          HOW MANY
```

### INPUT FA-FZ

The upper case argument characters FA through FZ denote floating point variables which may be any number between 0.000000 and +0.999999 E 37. These variables may be used with all arithmetic operators. When a floating point number exceeds the PRINT % format, it is printed in exponential form.

### EXAMPLE:

```
10 INPUT "ENTER NUMBER OF SHARES ", A
20 SCREEN LR
30 INPUT "ENTER PRICE PER SHARE $", FA
40 SCREEN LR
50 PRINT %6.2, "TOTAL VALUE = $", A*FA
```



### INPUT a-z

The lower case argument characters a through z denote byte variables which may be any whole number between 0 and 255. With a byte variable INPUT, a cursor is not provided, only one alphanumeric entry can be made and no terminator is required.

### EXAMPLE:

```
10 LET LANG= "ASCII", PARITY= ODD
20 INPUT "STRIKE NAY KEY",a
30 SCREEN C
40 PRINT %3.0, "DECIMAL = ",a," CHARACTER = ",[a]
50 GOTO 20
```

## INPUT, LET

**INPUT A\$-Z\$** The argument characters A\$ through Z\$ denote string variables. Each variable is limited to the value of the DIM statement or the number of characters which may be typed between the cursor and the end of the line, whichever is less. Entries must be terminated by depressing the  key instead of .

### EXAMPLE:

```
10 SCREEN S=64-7
20 LET LANG= "ASCII", PARITY= ODD
30 DIM A$=32
40 SCREEN "MESSAGE MUST BE 32 CHARACTERS OR LESS"LR
50 SCREEN "TERMINATE ENTRY WITH THE ""^P"" KEY"LR
60 INPUT "TYPE MESSAGE: ",A$
70 SCREEN C
80 PRINT "A$= ",A$
90 GOTO 40
```

**NOTE:** With all INPUT statements, the data entered via the keyboard is moved to specified variables in the language and parity selected during IO set-up.

**LET** This instruction assigns a value to a variable. The LET statement is always interpreted as performing the operation on the right side of the equal sign and assigning the resulting value to the variable on the left. The statement consists of the key word LET and an argument that may include string constants (messages enclosed in quotes), string variables (A\$, B\$, C\$), numeric constants (numbers), and integer variables (A, B, C), or expressions involving all of the above.

**LET A-Z** The upper case argument characters A through Z are used to denote integer variables. These variables may be used with all of the arithmetic operators but cannot receive the results of \* (multiply) and / (divide). The integer variable is limited to a value between 0 and 32 767. Numbers in excess of these values must be assigned to floating point variables. In addition, an integer variable used with the argument LEN and a string variable returns the number of characters in the string to the integer variable.

### EXAMPLE:

```
100 LET A=B
110 LET A=C+B
120 LET FA=C+B*D/E
130 LET A=1, B=2, C=3
140 LET A=LEN A$
```

## LET

### LET A\$-Z\$

The argument characters A\$ through Z\$ denote string variables. Each variable may contain up to the number of characters specified by the DIM statement. The value of the string must be enclosed in quotation marks. Specific characters may be separated from the string by noting their sequential position within parentheses and assigning them a new variable.

#### EXAMPLE:

```
10 SCREEN S=64-7
20 DIM A$=50, B$=50, C$=50, D$=50, E$=50
30 LET A$="THE QUICK BROWN FOX JUMPED OVER A LAZY DOG"
40 PRINT A$
50 LET B$=" 1234567890"
60 PRINT A$,B$
70 LET C$=A$(33,42)
80 LET D$=A$(20,32)
90 LET E$=A$(1,19)
100 PRINT C$, D$, E$
```

#### SAMPLE RUN:

```
THE QUICK BROWN FOX JUMPED OVER A LAZY DOG
THE QUICK BROWN FOX JUMPED OVER A DOG 1234567890
A LAZY DOG JUMPED OVER THE QUICK BROWN FOX
```

**NOTE:** The numbers within parentheses denote the beginning and ending characters within a string. To separate a single character from the string, the beginning and ending characters are the same, e.g., A\$=B\$(2,2).

**LET A\$-Z\$=CBUF** This argument is always on the right of the equal sign in a LET expression and is used to assign all or part of the contents of the capture buffer to a string variable. That part of the capture buffer assigned to the variable is selected using two digits enclosed in parentheses immediately following the argument CBUF, i.e., LET B\$=CBUF(1,88). When using CBUF, the last character in the key word indicates that characters are counted from the beginning of the buffer and only the specified characters are stored in the buffer.

#### EXAMPLE:

```
10 SCREEN S=64-14
20 CLEAR CBUF
30 DIM A$=50, B$=50
40 LET A$="THE QUICK BROWN FOX JUMPED OVER A LAZY DOG"
50 PRINT #0,#3,A$
60 SCREEN LR
70 LET B$= CBUF(1,20)
80 PRINT B$
```

## LET

**LET A\$-Z\$=C~~BU~~FE** This argument performs the same function as CBUFB except that characters are counted from the end of the capture buffer as indicated by the last character in the key word (C~~BU~~FE).

**LET A\$-Z\$=RTC** The argument RTC is always on the right side of the equal sign in a LET expression and is used to assign the value of the Real Time Clock to a string variable.

### EXAMPLE:

```
10 DIM A$=12
20 LET A$=RTC
30 PRINT A$
40 WAIT 1000
50 SCREEN C
60 GOTO 20
```

### SAMPLE RUN:

```
000 00:00:01
```

## LET FA-FZ

The upper case argument characters FA through FZ denote floating point variables which may be any number between 0.000000 and +0.999999 E 37. These variables may be used with all arithmetic operators. In addition, this statement may be used with the argument MSTM to transfer the contents of the measurement timer to a floating point variable. It may also be used with the argument SPEED to transfer the IO speed to a floating point variable, e.g., LET FA=SPEED. The contents of the timer can then be displayed using the PRINT statement.

### EXAMPLE: Arithmetic Operators

```
10 SCREEN S=64-7
20 INPUT "ENTER MULIPLICAND: ",FA
30 SCREEN LR
40 INPUT "ENTER MULTIPLIER: ", FB
50 SCREEN LRC
60 LET FC=FA*FB
70 PRINT "%4.3,FA," C ",FB," = ',FC
80 GOTO 20
```

### SAMPLE RUN:

```
ENTER MULTIPLICAND: 12
ENTER MULTIPLIER: 12
12.000X 12.000= 14.000
```

### EXAMPLE: MSTM

```
10 CLEAR MSTM
20 GOSUB 100
30 LET FA=MSTM
40 PRINT FA
```

## LET

**LET FA-FZ=MSTM** When a floating point variable is equal to MSTM, the variable is set to a number in whole milliseconds which is equal to the time the measurement timer has been running. The measurement timer starts when the CLEAR MSTM statement is executed.

### EXAMPLE:

```
10 CLEAR MSTM
20 WAIT 1000
30 LET FA=MSTM
40 PRINT FA/1000
```

**LET a-z** The lower case argument characters a through z denote byte variables which may be any whole number between 0 and 255. When enclosed in brackets, the argument represents a character, otherwise they are treated as decimal numbers.

### EXAMPLE:

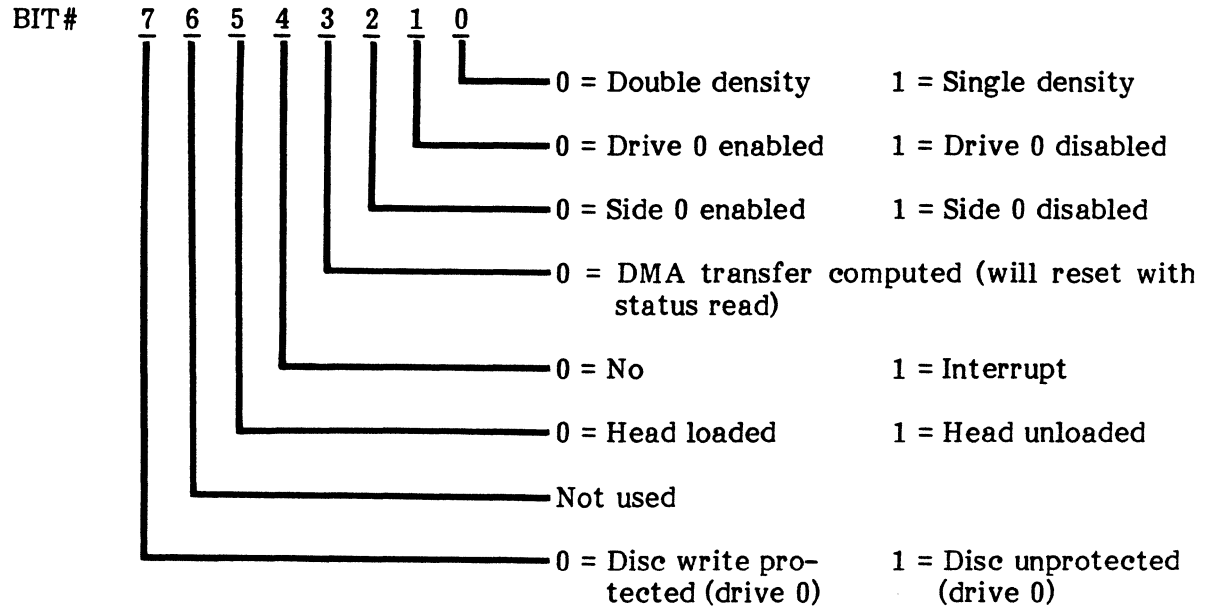
```
10 LET LANG= "ASCII", PARITY= ODD
20 SCREEN S=64-14
30 LET a=0
40 FOR T=0 TO 255
50   WAIT 100
70   PRINT %3.0,a," = ",[a];
80   LET a=a+1
90 NEXT T
```

**LET a-z=a-z LPAR** This argument is used in conjunction with byte variables to force the parity of a byte to agree with the parity in the IO parameters.

### EXAMPLE:

```
10 LET a=a+1
20 LET a=a LPAR
30 PRINT #1, a
40 WAIT XDONE
```

**LET a-z=DSTAT** This statement is used to determine the status of the disc. The disc status is placed in any byte variable and then interpreted as follows:



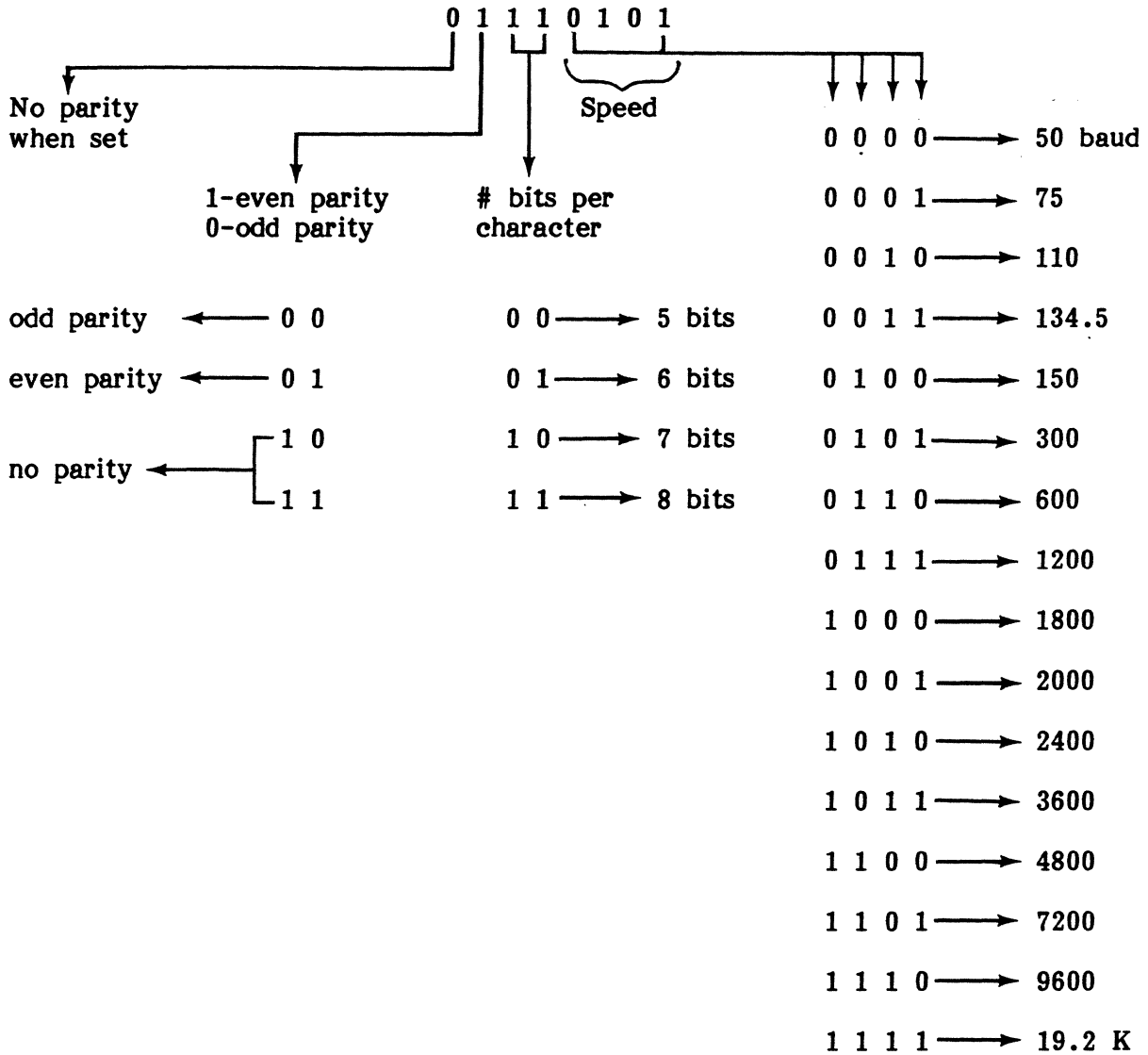
**LET a-z=a-z HEX** This statement is used when BCD, IPARS, or SELECTRIC is the language selected. It reverses the bit sequence of the byte variable as follows:

- Bit 0 is moved to the bit 6 position.
- Bit 1 is moved to the bit 5 position.
- Bit 2 is moved to the bit 4 position.
- Bit 3 does not change.
- Bit 4 moves to the bit 2 position.
- Bit 5 moves to the bit 1 position.
- Bit 6 moves to the bit 0 position..



**LET**

**LET a-z=XIOP or LET XIOP=a-z** When this argument is set equal to a byte variable, XIO parameters are established according to the byte variable as shown below. Existing XIO parameters may be transferred to a byte variable using **LET XIOP=a**.



**EXAMPLE: (even parity, 8-bits, 300 baud)**

```

10 LET a=0B "01110101", XIOP =a
20 LET a=0H "75", XIOP =a
30 LET a=00 "165", XIOP =a
40 LET a=117, XIOP =a
50 LET a=XIOP

```

**LET**

**LET a-z=0B "binary number"** This argument followed by an eight bit binary number enclosed in quotation marks assigns that bit configuration to a byte variable.

**EXAMPLE:**

```
10 LET a=0B "01111110"
20 PRINT a, " = ", [a]
```

**LET a-z=0H "HEX pair"** This argument followed by a HEX pair enclosed in quotes assigns that bit configuration to a byte variable.

**EXAMPLE:**

```
10 LET a=0H "FE"
20 PRINT a, " = ", [a]
```

**LET a-z=0O "octal number"** This argument followed by an octal number enclosed in quotes assigns that bit configuration to a byte variable.

**EXAMPLE:**

```
10 LET a=0O, "176"
20 PRINT a, " = ", [a]
```

**LET BCS**

This argument instructs the ENCORE to perform a block check calculation. This calculation is made using string variables and storing the results in one or more byte variables. The polynomial to be used must be selected by the STATE command.

**EXAMPLE:**

```
10 REM FEBSC MUST BE IN STORAGE
20 CHAIN "FEBSC"
30 SCREEN S=64-14
40 LET LANG= "EBCDIC", PARITY= NONE, MODE= SYN
50 STATE MODEM, CRC= CRC16
60 TON P3 , P3D , P5 , P8
80 DIM A$=50, B$=50
90 LET A$="THE QUICK BROWN FOX EX "
100 LET BCS ab=A$
110 LET B$="SySySy", A$,a TO b, "[FFFFFFFF]"
120 PRINT #1,B$
130 WAIT XDONE
140 GOTO 120
```

## LET

### LET CLOCK

This statement sets the clock equal to INT or EXT, instructing the ENCORE to derive timing internally or from an external source. Upon execution, IO Mode update occurs.

#### EXAMPLE:

```
10 LET FA=300
20 LET SPEED =FA
30 LET PARITY= ODD
40 LET CLOCK= INT
```

### LET LANG

This statement is used during program compile to change the operating language of the ENCORE. When compiled, the language set into the IO format is changed to agree with this statement. The desired language is entered within quotation marks. It is also used in conjunction with byte variables to convert a byte from ASCII to the language specified in the IO parameters.

#### EXAMPLES:

```
10 LET LANG= "ASCII"
20 LET LANG= "EBCDIC"
30 LET LANG= "BCD"
40 LET LANG= "SELECTRIC"
50 LET a=48
60 LET a=a LANG
```

#### NOTES:

- (1) With the exception of LET SPEED = FA and LET FA = SPEED, all other arguments establishing IO parameters are accomplished during the compile and not during program execution. It is not possible, therefore, to select the IO values using operator prompts in the program. If, for example, there are multiple "LET LANG=" statements in your program, the IO language will be set to the last entry in your source code.
- (2) To convert a byte variable from the line language specified in the IO set-up to NO PARITY ASCII, use the DISP C instruction. Move the byte to be converted to byte variable d, then DISP C. Byte variable d will be converted from the line language to NO PARITY ASCII.

#### EXAMPLE:

```
10 LET d=a
20 DISP C
30 PRINT #1,[d]
```

**LET MODE** This statement instructs ENCORE to operate in any one of the following modes: TRAN (transparent), ASYN (asynchronous), SYN (synchronous), or SDLC (Synchronous Data Links Control). During compile, this statement updates the IO parameters.

**EXAMPLE:**

```
10 LET MODE= TRAN
20 LET MODE= ASYN
30 LET MODE= SYN
40 LET MODE= SDLC
```

**LET NRZI** This statement, followed by an equal sign and YES, instructs ENCORE to use the NRZI method of SDLC transmission coding. When followed by NO, NRZI is not used.

**EXAMPLE:**

```
10 LET NRZI= YES
20 LET NRZI= NO
```

**LET PARITY** This statement sets the parity equal to ODD, EVEN, NONE, MARK, SPACE, or IGNORE. During compile, this statement updates the IO parameters.

**EXAMPLE:**

```
10 LET FA=300
20 LET SPEED =FA
30 LET PARITY= ODD
```

**LET SPEED** This statement, when equal to a floating point variable, sets the operating speed of the ENCORE to the value of the variable (10 to 99900). Upon execution, the IO Mode is automatically updated. If the statement is written LET FA=SPEED, FA is then equal to the speed set in the IO parameters.

**EXAMPLE:**

```
10 LET FA=100
20 LET SPEED =FA
30 LET FS=SPEED
40 CHAIN "IO"
```

**LET STOPS** This statement, when equal to ONE, ONE.5, or TWO, instructs the ENCORE to operate with a 1, 1½, or 2 unit stop pulse. Upon execution, IO parameters are updated.

## LET, MICRO, ON

### EXAMPLE:

```
10 LET FA=300
20 LET SPEED =FA
30 LET PARITY= ODD
40 LET CLOCK= INT
50 LET MODE= ASYN
60 LET STOPS= TWO
```

### LET SYNC

This statement is used to change the sync sequence set into the IO format. It may be used with any two characters or any two HEX pairs. Because IO parameters are updated during execution, this statement must be placed near the beginning of the program.

### EXAMPLE:

```
10 LET SYNC="SY SY"
20 LET SYNC="[3232]"
```

**MICRO** A MICRO is similar to the machine language (MICRO PROCESSOR CODE) of the INTEL 8085 used in the ENCORE. It permits access to the same areas accessed by the ENCORE host processor. The use of MICRO is limited to specially trained programmers.

### EXAMPLE:

```
40 MICRO "324097"
```

### ON

This statement provides the ENCORE with multiple branching capabilities. It includes a variable, or expression, followed by the key word GOTO or GOSUB, and a list of line numbers. The integer value of the variable, or formula, transfers program execution to the line number whose order in the list corresponds with the integer. The ON expression may include integers, integer variables, byte variables, and appropriate operators. If the integer value of the expression is less or greater than any number in the list of numbers, program execution continues with the next statement.

### EXAMPLE:

```
10 INPUT A
20 ON A GOTO 100, 200, 300, 400
30 STOP
100 STOP
200 STOP
300 STOP
400 STOP
```

**PRINT** The PRINT instruction is used to write data to the CRT (default) or to an external device via any of the addressable output ports. The PRINT statement consists of the PRINT instruction and an argument that may include string constants (messages in quotes), string variables (A\$, B\$, C\$), numeric constants (numbers), integer variables (A, B, C), floating point variables (FA, FB), or expressions involving all of the above. A summary of all instruction variables is found in the following paragraphs.

**NOTE:** Two error messages are provided to alert the operator in the event of an illegal PRINT statement:

MATH ERROR #1 indicates that a floating point variable is greater than  $0.425 \times 10^{38}$ .

MATH ERROR #2 indicates that division by zero is attempted.

**RULES:**

- (1) String constants ("HI THERE") must be enclosed in quotes.
- (2) Successive items in the PRINT argument must be separated by commas.
- (3) Spaces are not required between instruction and argument or between argument items.
- (4) Each PRINT statement writes one new line of data.
- (5) A PRINT statement containing no argument will produce a blank line.
- (6) Two or more successive PRINT statements may be printed on the same line when the first statement is followed by a semicolon.

**EXAMPLE:**

**SAMPLE RUN:**

```
10 PRINT "HI THERE!";           HI THERE! I AM YOUR NEW ENCORE
20 PRINT "I AM YOUR NEW ENCORE"
```

- (7) Integer variables that do not exceed 32 767 may be used as fixed point variables for addition and subtraction. Whenever an integer value exceeds 32 767 or is used in multiplication or division, it should be handled as a floating point variable (FA, FB, FC) and should be formatted using PRINT %. An example is given in the explanation of the PRINT statement using floating point variables.
- (8) An exclamation point (!), as the last entry, will instruct the ENCORE to continuously transmit the last string in the PRINT statement.

**EXAMPLE:**

```
10 STATE NLEAD 8
20 TON P2 , P2D , P3 , P3D
30 PRINT #1, "SySy1234567890!"
```

## PRINT

### PRINT A-Z

The upper case argument characters A through Z are used to denote integer variables. These variables may be used with all of the arithmetic operators except \* (multiply) and / (divide). The integer variable is limited to whole number values between 0 and 32 767. Numbers other than these values must be assigned to a floating point variable.

### PRINT A\$-Z\$

The argument characters A\$ through Z\$ denote string variables. Each variable may contain up to 46 characters which must be enclosed in quotation marks and dimensioned earlier in the program. A variable may contain more than 46 characters if it is set equal to two or more string variables or the contents of the capture buffer, e.g., LET A\$=B\$, C\$ or LET A\$=C\$ (1,88). To print selected characters from a string, follow the string variable with the number of beginning and ending characters in parentheses. To print one character from the string, the beginning and ending character are the same.

#### EXAMPLE:

```
10 DIM A$=12
20 LET A$="HELLO THERE!"
30 PRINT A$
40 PRINT A$ (1,5)
50 PRINT A$ (7,12)
60 PRINT A$ (5,5)
```

#### SAMPLE RUN:

```
HELLO THERE!
HELLO
THERE!
0
```

### PRINT FA-FZ

The upper case argument characters FA through FZ denote floating point variables which may be any number between 0.000000 and +0.999999 E 37. These variables may be used with all arithmetic operators. When a floating point number exceeds the PRINT % format, it is printed in exponential form.

#### EXAMPLE:

```
10 LET FA=1493.7
20 PRINT %3.2, FA
```

#### SAMPLE RUN:

```
0.1493700E 04
```

### PRINT a-z

The lower case argument characters a through z denote byte variables which may be any whole number between 0 and 255. When enclosed in brackets, the argument is printed as a character, otherwise it is printed as a decimal.

#### EXAMPLE:

```
10 LET a=0
20 PRINT %3.1, a, " = ", [a]
```

## PRINT

**PRINT "[*HEX number*]"** This statement is used to PRINT the ASCII equivalent of the specified HEX pair.

EXAMPLE:

```
10 LET LANG= "ASCII", PARITY= ODD
20 PRINT "[5555]"
```

SAMPLE RUN:

UU

**PRINT "*string constant*"** Quotation marks are used to denote the beginning and ending of a string constant. When quotation marks appear in the string, they must be entered twice.

EXAMPLE:

```
10 PRINT "HELLO"
20 PRINT "DEPRESS THE ""CMD"" KEY"
```

SAMPLE RUN:

```
HELLO
DEPRESS THE "CMD" KEY
```

**PRINT "*(string constant)*"** Same as above except the parity bit of each character does not agree with the parity set in the IO Mode.

**PRINT %** This statement is used to format numbers. It is immediately followed by a number, a decimal point, and another number to limit the digits at the left and right of the decimal point. Format numbers may range from 0 to 7.

EXAMPLE:

SAMPLE RUN:

```
10 LET FA=0273.614      273.61
20 PRINT %3.2, FA
```

**PRINT #0** This statement instructs ENCORE to output the remainder of the PRINT statement to the CRT. This is the default condition assumed when no other port is entered.

**PRINT #1** This statement instructs ENCORE to output the remainder of the PRINT statement via the RS-232 interface. Always follow this statement with a new line and the statement WAIT XDONE to assure transmission of data before the next instruction is executed.

**PRINT #2** This statement instructs ENCORE to output the remainder of the PRINT statement via the XIO port. Always follow this statement with a new line and the statement WAIT XDONE to assure transmission of data before the next instruction is executed.



## **PRINT, READ, REM, RESTORE, RETURN, SCREEN**

**PRINT #3** This statement instructs ENCORE to output the remainder of the PRINT statement to the capture buffer.

**PRINT #4** This statement instructs ENCORE to output a SYNC pulse via the rear panel BNC SYNC connector. The pulse is from 0 to 5 volts and approximately 10 us. in duration.

**PRINT #5** This statement instructs ENCORE to sound an audible alert for approximately  $\frac{1}{4}$  second.

**READ** See DATA.

**REM** This instruction allows the programmer to place remarks anywhere within the COMBASIC program to explain the use of specific program steps or subroutines. It is the only statement in the COMBASIC language that is not executed, and it only appears when the program is listed (Edit Mode, LIST command). Each REM statement may contain up to 51 characters.

**NOTE:** To conserve memory, the REM statements may be deleted from the source program.

**EXAMPLE:**

```
10 REM *THIS PROGRAM WILL PASSIVELY MONITOR A
20 REM *LINE FDUX, STORING DATA IN
30 REM *THE CAPTURE BUFFER FOR LATER ANALYSIS
```

**RESTORE** See DATA.

**RETURN** See GOSUB.

**SCREEN** This instruction allows the operator to format the CRT. It provides arguments to control character size, position the cursor, and display messages. It does not perform arithmetic operations.

**SCREEN B** This statement will move the cursor backwards one space for each B in the argument.

**EXAMPLE:**

```
10 SCREEN NI " " "BB
```

**SCREEN C** Upon execution, this statement clears the display and returns the cursor to the top left corner of the screen.

EXAMPLE:

```
10 SCREEN C, "SEARCH STOPPED"LR
```

**SCREEN E** This statement erases data from its present cursor position to the end of the line.

**SCREEN G a-z** This statement, followed by a byte variable (Ga), will place the byte at the current screen position into the byte variable. It may be preceded by a SCREEN position argument to locate the desired character. The cursor is automatically advanced one position after executing SCREEN Ga.

EXAMPLE:

```
10 LET LANG= "ASCII", PARITY= ODD
20 CHAIN "XIO"
30 SCREEN S=32-7
40 FOR T=1 TO 224
50   PRINT "X";
60 NEXT T
70 LET A=31, B=6
80 FOR X=0 TO B
90   PRINT #2, "C L C "
100  FOR Y=0 TO A
110   SCREEN P=X-YGa
120   PRINT #2, [a]
130   WAIT XDONE
140  NEXT Y
150 NEXT X
```

**SCREEN H** All data displayed on the CRT after this statement is executed will appear in HEX.

EXAMPLE:

```
10 SCREEN H "SYSYET"
```

**SCREEN I** All data displayed on the CRT after this statement is executed will appear as inverse video (black characters on a white line).

EXAMPLE:

```
10 SCREEN I "ENTER NUMBER"
```

## SCREEN

**SCREEN L** Upon execution, this statement moves the cursor down one line maintaining the same horizontal position (L=Line Feed).

EXAMPLE:

```
10 SCREEN "DISPLAY CAPTURE BUFFER"LR
20 GOTO 10
```

**SCREEN N** This statement is used to cancel all other screen arguments (N=Normal).

EXAMPLE:

```
10 SCREEN IP=15-1 "ENTRY ERROR"N
```

**SCREEN P =**      This statement is used in conjunction with integers or integer variables to position the cursor on the CRT. The first integer represents the character position and the second represents the line number (P=Position).

EXAMPLE:

```
10 LET A=0, B=1
20 SCREEN P=A-B, I"LAST RECORD: "N
30 SCREEN P=0-3 "DEPRESS ""CMD"" KEY"
```

**SCREEN R** Upon execution, this statement moves the cursor to the first character position on its present line (R=Return).

EXAMPLE:

```
10 SCREEN R, "COMMUNICATIONS ESTABLISHED"L
20 GOTO 10
```

**SCREEN S** This statement is used to determine the number of characters and lines displayed on the CRT. Four configurations are available:

S=64-14	64 characters on each of 14 lines
S=64-7	64 characters on each of 7 lines
S=32-14	32 characters on each of 14 lines
S=32-7	32 characters on each of 7 lines

EXAMPLE:

```
10 SCREEN C S=64-14 P=23-6 I"BY DIGITECH"N
```

## SCREEN

**SCREEN SA-SY** This statement places the screen size into any two consecutive integer variables beginning with the letters A through Y.

**EXAMPLE:**

```
10 SCREEN S=64-14
20 SCREEN SA
30 PRINT A,B
```

**NOTE:** A=64, B=14.

**SCREEN "string constant"** Quotation marks are used to denote the beginning and ending of a string constant. When quotation marks appear within the string, they must be entered twice. When used in the SCREEN statement, a string will be output only to the CRT.

**EXAMPLE:**

```
10 SCREEN S=64-14 "THIS IS A TEST"LR
20 SCREEN "DEPRESS THE ""CMD"" KEY TO CONTINUE"
```

**SCREEN SZ** This statement changes the screen size to the next size.

**EXAMPLE:**


```
10 SCREEN SX
20 FOR T=1 TO Y
30 SCREEN ZUI "SCREEN COMMANDS"N
40 PRINT %2.0, "SCREEN S=", X, "-", Y;
50 WAIT 500
60 NEXT T
70 SCREEN SZ
80 GOTO 10
```

**SCREEN U** This statement underlines all data displayed on the CRT.

**SCREEN Z** This statement scrolls the data on the CRT up one line.

## STATE

**STATE** The STATE instruction is used to establish circuit parameters other than those covered in the IO parameters and is usually the first statement in a program. Each of the STATE arguments and default conditions are defined in the following paragraphs.

**STATE BRG** The BRG statement is the default for MODEM or TERM, and accomplishes the same function as the  key in the control cluster. This function serves as an electrical disconnect. ENCORE is physically connected to the circuit but is not terminated as a modem or terminal. It does, however, passively monitor the interface. STATE BRG should be used to terminate programs when disconnect is required.

### EXAMPLE:

```
10 STATE BRG , HDUX , NLOCK , STAT
```

**NOTE:** Whenever STATE BRG is used prior to a CHAIN instruction, STATE BRG must be executed in the program being CHAINED to and executed again by the main program upon return from CHAIN. When used prior to a BOOT instruction, it must be re-entered upon return from BOOT.

**STATE CRC = \_\_\_** This statement is used with several arguments to establish the type of cyclic redundancy check to be performed. It always presets the internal CRC calculator to zero. CRC 16 is the default condition when STATE CRC = \_\_\_ is not used. STATE CRC=CRC 6 is only available when the IPARS option E250-01 is provided.

### EXAMPLE:

```
10 STATE CRC = CRC 16
20 STATE CRC = CRC 12
30 STATE CRC = CRC 6 (IPARS option)
40 STATE CRC = LRC 7
50 STATE CRC = LRC 8
60 STATE CRC = CCITT
```

**STATE DISCON/DISCOFF** The DISCON argument instructs the ENCORE to position the read/write head at the beginning of the first data record. The DISCOFF argument instructs the ENCORE to mark the end of the data record. It also cancels the DISCON argument and is the default condition assumed when STATE DISCON is not entered. If a STATE DISCON is executed, a STATE DISCOFF must be executed before any data captured onto the disc can be recovered. In addition, the contents of the buffer are automatically read to the disc, in 1K increments, as it is filled. This requires prior execution of TON P2B or P3B in order to transfer incoming data to the buffer. If a DISCOFF instruction is executed prior to filling the buffer, the partial contents of the buffer is not output to the disc and can, therefore, not be recovered off-line. To assure complete data recovery, always fill the buffer with some type of data before executing DISCOFF. This is easily accomplished by preceeding the DISCOFF instruction with two PRINT #3 statements as follows:

## EXAMPLE:

```
100 PRINT #3, " *"!
110 PRINT #3, " *"!
120 STATE DISCOFF
```

The exclamation point (!) causes the buffer to be filled with spaces immediately following the asterisk. The second print statement assures that the data is in the proper position of the buffer for easy recall.

**STATE FAST/NFAST** The FAST argument allows the ENCORE to capture and display data at high speeds by not creating attribute bytes. NFAST is the default condition assumed when FAST is not entered. It is also used to cancel FAST.

## EXAMPLE:

```
10 STATE BRG , FAST
```

**STATE FCS =** \_\_\_ This statement is similar to STATE CRC except that the calculator is preset to all ones as required in SDLC.

## EXAMPLE:

```
10 STATE FCS = CRC 16
20 STATE FCS = CRC 12
30 STATE FCS = LRC 7
40 STATE FCS = LRC 8
50 STATE FCS = CCITT
```

## STATE

**STATE FDUX** This statement formats the CRT for full-duplex operation where send and receive data are presented in a dual-line time correlated format. When the FDUX argument is not entered, ENCORE defaults to HDUX.

**EXAMPLE:**

```
10 STATE BRG , FAST , FDUX
```

**STATE FLTR/NFLTR** The FLTR argument, followed by a HEX pair (e.g. FLTR = 96) will delete the selected character from the incoming data stream whenever it appears. It will not be seen by the front end. The character to be deleted must be entered in HEX (see Code Charts, Appendix D). The NFLTR argument cancels FLTR and is the default condition assumed when FLTR is not entered.

**EXAMPLE:**

```
10 STATE BRG , FAST , FDUX , FLTR=FF
```

**STATE HDUX** This statement formats the CRT for half-duplex operation where send and receive data are displayed on the same line in an interlaced format. Send data appears as normal video, receive data appears as inverse video. This is the default format, assumed when FDUX is not entered.

**EXAMPLE:**

```
10 STATE BRG , FAST , HDUX , FLTR=FF
```

**STATE LEAD=/NLEAD** The LEAD= argument, followed by an interface pin number (4, 5, 8, A, B, 20, and 22) instructs the ENCORE to underline data logged while the specified lead was on. A new STATE LEAD= or STATE NLEAD cancels the previous STATE LEAD= and is the default condition assumed when STATE LEAD= is not entered.


**EXAMPLE:**

```
10 STATE BRG , FAST , HDUX , FLTR=FF  
20 STATE LEAD= 4
```

**STATE NLEAD8** Under normal operating conditions, RS-232 lead 8 (COD) must be on for DTE to input receive data at pin 3. This argument ignores pin 8, as though the modem had supplied COD, allowing ENCORE to input pin 3 data as DTE or a bridging device.

**EXAMPLE:**

```
10 STATE BRG , FDUX , NLEAD8  
20 TON P2 , P2D , P3 , P3D
```

**STATE LOCK/NLOCK** The LOCK argument is used in conjunction with the WHEN KBD instruction to inhibit keyboard entry. This provides the programmer with a means for eliminating accidental keyboard entries that might alter the program during execution. As an example, accidentally depressing the  key during execution would interrupt the program and return ENCORE to the Level-3 Command Mode. In the example subroutine shown below, ENCORE responds to only one keyboard entry while all others are ignored. The NLOCK argument is the default condition assumed when LOCK is not entered. It is also used during the program to cancel the LOCK argument.

**EXAMPLE:**

```

10 STATE BRG , FDUX , NLEAD8
20 TON P2 , P2D , P3 , P3D
30 STATE LOCK
40 WHEN KBD x GOTO 60
50 GOTO 50
60 IF x<>27 GOTO 40
70 REM 27 IS THE "ESC" KEY
80 STATE NLOCK
90 CHAIN ""

```

**NOTE:** Use of the WHEN KBD        statement permits recognition of specific keys during the lock state. If WHEN KBD        is not used, escape is accomplished by turning power off and on again.

**STATE MODEM** This argument instructs ENCORE to simulate a modem (DCE) exercising control over the appropriate interface leads. If neither the BRG, MODEM, or TERM arguments is entered, ENCORE will default to BRG.

**EXAMPLE:**

```

10 STATE MODEM
20 TON P2 , P2D , P3 , P3D , P8
30 WHEN P4=1 GOTO 50
40 GOTO 40
50 TON P5
60 WHEN P2 = "SYSYD" "EQ" GOTO 100
70 GOTO 70
100 STOP

```

**NOTE:** Whenever STATE MODEM is used prior to a CHAIN instruction, STATE MODEM must be executed in the program being CHAINED to and executed again by the main program upon return from CHAIN. When used prior to a BOOT instruction, it must be re-entered upon return from BOOT.



## STATE

**STATE P3ONLY/NP3ONLY** Under normal circumstances a single front end program will control both front end processors. If however, the P3ONLY argument is used immediately following the front end program, a second front end program may be written to control receive data (pin 3). Assuming two front end programs, FE1 and FE2, you may use the following example to call in each program. The NP3ONLY argument is the default condition assumed when P3ONLY is not entered. It allows one front end program to control both front end processors.

### EXAMPLE:

```
10 REM P2 FRONT END IS ASYNCHRONOUS
20 IXO
30 JMP 20
40 STATE P3ONLY
50 REM P3 FRONT END SYNCHRONOUS
60 RSN
70 IXO
80 JMP 70
90 STATE NP3ONLY
100 STATE BRG , FDUX , NLEAD8
110 TON P2 , P2D , P3 , P3D
```

**STATE PIPE** The PIPE argument allows ENCORE to capture data at speeds up to 64K bytes FDUX by turning off the CRT.

### EXAMPLE:

```
10 STATE NLEAD8
20 LET MODE= SYN , LANG= "EBCDIC"
30 LET CLOCK= INT , SYNC= "SYSY"
40 CLEAR CBUF
50 WHEN FULL GOTO 110
60 SCREEN C "PIPE RUNNING"
70 WAIT 500
80 STATE PIPE
90 TON P2 , P2B , P3 , P3B
100 PRINT #1, "SYSYSY1234567890"!
105 GOTO 105
110 CHAIN " "
```

**STATE STAT/NSTAT** The STAT argument is the default condition assumed when NSTAT is not entered. It instructs ENCORE to transfer a status byte from front end processor for each incoming character. The NSTAT argument instructs ENCORE to delete the status byte from the data transferred to the main processor. This provides for greater storage of data in both the capture buffer and mass storage media.

## EXAMPLE:

```

10 STATE NLEAD8 , NSTAT , FAST
20 TON P2 , P2D , P3 , P3D
30 PRINT #1, "SYSY1234567890"!

```

**STATE SYNC/NSYNC** The SYNC argument instructs ENCORE to transfer data from front end to main processor, but only if the data is in sync. ENCORE will then log and display synchronized data only. The NSYNC argument instructs ENCORE to transfer all data from front end to main processor and display out-of-sync data at half intensity. This cancels SYNC and is the default condition assumed when SYNC is not entered.

## EXAMPLE:

```

10 CHAIN "FEBS"
20 REM FEBS MUST BE IN STORAGE
30 DIM A$=50, B$=100
40 LET A$="1234567890EX"
50 LET BCS ab=A$
60 LET B$="SYSYSY", A$, a TO b, "ABCDEFGHJKLM"
70 STATE FDUX , NLEAD8 , NSTAT , NFAST
80 TON P2 , P2D , P3 , P3D
90 PRINT #1, B$

```

**STATE TERM** This statement instructs ENCORE to simulate a terminal (DTE) exercising control over the appropriate interface leads. If neither the BRG, MODEM, or TERM arguments are entered, ENCORE will default to BRG.

## EXAMPLE:

```

10 STATE TERM , HDUX , NLEAD8
20 TON P4
30 WHEN P5 =1 GOTO 50
40 GOTO 40
50 PRINT "SYSY1234567890"

```

**NOTE:** Whenever STATE TERM is used prior to a CHAIN instruction, STATE TERM must be executed in the program being CHAINED to and executed again by the main program upon return from CHAIN. When used prior to a BOOT instruction, it must be re-entered upon return from BOOT.

**STATE UPC/NUPC** The UPC argument is used in conjunction with the WHEN KBD and INPUT statements instructing the ENCORE to accept only upper case keyboard entries. The NUPC argument cancels UPC and is the default condition assumed when UPC is not entered.

## STATE, STOP, TON/TOFF

### EXAMPLE:

```
10 DIM A$=32, B$=32
20 STATE UPC
30 INPUT A$
40 SCREEN LR
50 STATE NUPC
60 INPUT B$
70 SCREEN LR
80 PRINT A$
90 PRINT B$
```

### STATE XIDL

This statement followed by a HEX pair (e.g., XIDL=00) allows the programmer to select the character (idle space=00) transmitted during an idle condition. The character must be entered in HEX. When XIDL is not entered, the ENCORE defaults to all 1's (HEX FF=11111111).

### EXAMPLE:

```
10 STATE NLEAD8 , FDUX , XIDL= 01
20 TON P2 , P2D , P3 , P3D
30 PRINT #1, "SYSY123456"
```

### STOP

This instruction halts program execution and requires no argument. When executed, the CRT will display the words "STOP AT LINE" followed by the line number of the STOP instruction.

### TON/TOFF

This instruction is used to control the interface and to turn on the capture buffer and CRT display. It may be cancelled by TOFF which is the default condition assumed when TON is not entered. In asynchronous transmission, it is advisable to WAIT 500 ms after turning on pins, and before transmitting the first message. Do not attempt to TON/TOFF more than ten pins on one COMBASIC line. Pin numbers must be entered in ascending order, as shown in the example.

### EXAMPLE:

```
10 TON P2 , P2B , P2D , P3 , P3B , P3D
20 TON P4 , P5 , P8 , P11 , P18
30 PRINT #1, "SYSY123456"!
```

**NOTE:** Always turn on pins 2 and 3 before turning on other pins.

### TONP2

This statement turns on the pin 2 front end processor (send data). Data can then be transmitted or received in accordance with the STATE instruction (STATE MODEM, STATE TERM, etc.).

- TONP2B** This statement allows the pin 2 front end processor (transmit data) to transfer incoming data to the capture buffer. It must always be executed prior to STATE DISCON in order to place data in the buffer for transfer to the disc.
- TONP2D** This statement allows ENCORE to display pin 2 data on the CRT (normal video).
- TONP3** This statement performs the same function as P2 except that it turns on the P3 front end processor.
- TONP3B** This statement serves the same purpose as P2B except that transfer occurs between the Pin 3 front end processor (receive data) and the capture buffer.
- TONP3D** This statement allows ENCORE to display pin 3 data on the CRT (inverse video).
- TONP4** This statement sets RS-232 pin 4 (REQUEST TO SEND) to the ON condition, but only if ENCORE is simulating a terminal (STATE TERM).
- TONP5** This statement sets RS-232 pin 5 (CLEAR TO SEND) to the ON condition, but only if ENCORE is simulating a modem (STATE MODEM).
- TONP6** This statement sets RS-232 pin 6 (DATA SET READY) to the ON condition, but only if the ENCORE is simulating a modem (STATE MODEM).
- TONP8** This statement sets RS-232 pin 8 (RECEIVE LINE SIGNAL DETECT) to the ON condition, but only if ENCORE is simulating a modem. If simulating a terminal, use STATE NLEAD8 as ENCORE will only input data on pin 3 when lead 8 is on.
- TONP11** This statement sets RS-232 pin 11 to the ON condition and may be written as TONP11 or TONPA. Pin 11 is one of two leads (pin 11 and 18) that are controlled during STATE BRG, MODEM, and TERM. This pin is typically used to control interface leads through the T-Connector. This includes any lead not under direct ENCORE control. The ON condition of this lead can be used as an event marker since its on or off state is recorded in the status byte logged for each incoming character (4th LSB).

## TON/TOFF, WAIT

- TONP18** This statement serves the same purpose as P11 except that its state is recorded in the 5th LSB of the status byte. It can be written as TONP18 or TONPB.
- TONP20** This statement sets RS-232 pin 20 (DATA TERMINAL READY) to the ON condition, but only if the ENCORE is simulating a terminal (STATE TERM).
- TONP22** This statement sets RS-232 pin 22 (RING INDICATOR) to the ON condition, but only if ENCORE is simulating a modem (STATE MODEM).
- TONP24** This statement outputs a transmit timing signal via RS-232 pin 24, but only if ENCORE is simulating a terminal (STATE TERM).

**WAIT** This instruction is used to create a delay in further execution of the program. The argument is time, entered in milliseconds as an integer or integer variable. During execution of the WAIT statement, data continues to be transmitted or received if initiated under prior instructions.

### EXAMPLE:

```
10 PRINT "THIS PROGRAM DEMONSTRATES WAIT"  
20 WAIT 5000  
30 PRINT "WAIT = 5 SECONDS"  
40 LET A=10000  
50 WAIT A  
60 PRINT "WAIT = 10 SECONDS"  
70 STOP
```

**WAIT A-Z** When used with the WAIT instruction, the upper case characters A through Z denote integer variables in milliseconds.

**WAIT XDONE** This argument instructs ENCORE to halt further program execution until the last two bytes have been loaded into the transmit buffer. This gives the slower external device time to copy a line of data before a new line is transmitted. Always use WAIT XDONE after every PRINT #1 or PRINT #2 statement.

### EXAMPLE:

```
10 PRINT #2, "TEST SEQUENCE VALID"  
20 WAIT XDONE
```

## WHEN

**WHEN** This is an interrupt instruction that will alter the course of the program when the condition of the argument is met. The argument always includes a GOTO or GOSUB instruction. In the case of a WHEN P2/P3=, there must be no more than four WHEN statements pending.

**WHEN ERROR** When an error occurs in program execution, this statement stores the error line number and definition number in variables E and e, respectively, provided the program has not been executed with an exclamation point. It then transfers program control to a line number specified by a GOTO or GOSUB.

**EXAMPLE:**

```
10 WHEN ERROR GOTO 100
20 GOTO 20
100 STOP
```

**WHEN FULL** The FULL argument is used in conjunction with GOTO or GOSUB to alter the course of the program when the capture buffer is full.

**EXAMPLE:**

```
10 WHEN FULL GOTO 100
20 GOTO 20
100 STOP
```

**WHEN KBD a-z** This statement is used in conjunction with a single byte variable (a-z) and GOTO or GOSUB to transfer program control to a given line number when a specified key is depressed. When a keyboard entry is made, the variable assumes the NO PARITY ASCII equivalent of that key. The entry is not converted to the language selected in the IO Mode and therefore, will not print to the CRT in that language. This is done to permit recovery of the ASCII equivalent for any keystroke and to allow escape from the program being executed. To display the byte variable in the IO language, use the statement LET a=aLANG.

**EXAMPLE:**

```
10 WHEN KBD a GOTO 30
20 GOTO 20
30 IF a=97 GOTO 100
100 STOP
```

**WHEN P \_\_\_ = 0/1** This statement is used to transfer program control to a specified line number when the appropriate pin is ON. The statement is used with the following pins: PA, PB, P4, P5, P6, P8, P11, P18, P20, and P22. As soon as a single WHEN P \_\_\_ condition is met, the instruction is canceled. If more than one WHEN P \_\_\_ instruction appears in the program, the first WHEN condition met cancels all WHEN instructions.

## WHEN

**NOTE:** Pins A and B are 11 and 18, respectively. 0 means OFF and 1 means ON.

### EXAMPLE:

```
10 WHEN P4 =1 GOTO 100
20 WHEN PA =1 GOTO 100
30 GOTO 30
100 STOP
```

**WHEN P2/P3** This statement is used in conjunction with GOTO, GOSUB, TOFF, and TON to transfer program control to a specified line number when data is present at the appropriate pin.

### EXAMPLE:

```
10 WHEN P2 TOFF P3 , P2="XYZ" GOTO 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3 a-z** The lower case argument characters a through z denote byte variables which are used in conjunction with the WHEN P2/P3 statement. When encountered, this statement halts program execution until a character appears at the specified pin. The byte variable then assumes the decimal value of that data character (0-255) and program execution continues.

### EXAMPLE:

```
10 WHEN P2 a
20 IF a="S" GOTO 40
30 GOTO 10
40 STOP
```

**WHEN P2/P3=A\$-Z\$** The argument characters A\$ through Z\$ denote string variables which are used in conjunction with the WHEN P2/P3 statement. When encountered, this statement will vector program execution when the specified string appears at the appropriate pin. The string variable must be preceded by spaces within quotation marks to denote the number of characters within the string variable, always starting from position one, that must appear before the WHEN condition will be met. The delimiting factor is the length of the data in the string or the number of spaces within the quotation marks, whichever is shorter. The maximum length of a string is determined by the DIM statement. The actual length is determined by the amount of data moved to the string.

**NOTE:** Any character may be used in place of the spaces to denote the number of characters required by the WHEN statement, e.g., (###), (1,2,3), etc..

**EXAMPLE:**

```
10 DIM A$=20, B$=30
20 LET A$="123"
30 LET B$="456"
40 WHEN P3="123" A$ GOTO 70
50 WHEN P2=" " B$ GOTO 80
60 GOTO 60
70 STOP
80 STOP
```

**WHEN P2/P3=BCB** This argument is used in conjunction with the WHEN P2/P3 statement and a GOTO or GOSUB instruction to determine whether the block check character on pin 2 or pin 3 is bad. If bad, program control is transferred to the line number following GOTO or GOSUB.

**EXAMPLE:**

```
10 WHEN P2=BCB GOTO 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3=BCC** This argument is used in conjunction with the WHEN P2/P3 statement to transfer program control to a specified line when the BCC characters appear at the appropriate pin.

**EXAMPLE:**

```
10 WHEN P2=BCC GOTO 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3=BCG** This argument is used in conjunction with the WHEN P2/P3 statement to transfer program control to a specified line number when the block check sequence at the appropriate pin is good.

**EXAMPLE:**

```
10 WHEN P2=BCG GOTO 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3=FLAG** This argument is used in conjunction with the WHEN P2/P3 statement to transfer program control to a specified line number when the SDLC flag appears at the appropriate pin.



## WHEN

### EXAMPLE:

```
10 WHEN P2=FLAG GOSUB 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3="[HEX pair]"** This argument is used in conjunction with the WHEN P2/P3 statement to transfer program control to a line number specified by GOTO or GOSUB. Transfer occurs when the data at the appropriate pin is equal to the byte specified by the bracketed HEX pair.

### EXAMPLE:

```
10 WHEN P2="[7E]" GOTO 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3=PE** This argument is used in conjunction with the WHEN P2/P3 statement to transfer program control to the line number specified by GOTO or GOSUB. Transfer occurs when a parity error is detected.

### EXAMPLE:

```
10 WHEN P2=PE GOTO 100
20 GOTO 20
100 STOP
```

**WHEN P2/P3="string constant"** This argument is used in conjunction with the WHEN P2/P3 statement to transfer program control to a line number specified by a GOTO or GOSUB. Transfer occurs when data at the appropriate pin is equal to the string.

### EXAMPLE:

```
10 WHEN P2="ABC" GOTO 100
20 GOTO 20
100 STOP
```

**WHEN RTC** This statement is used to transfer program control to a specified line number when the Real Time Clock is equal to a string variable.

### EXAMPLE:

```
10 DIM A$=12
20 LET A$="123 10:20:0"
30 WHEN RTC =A$ GOTO 100
40 GOTO 40
100 STOP
```

## WHEN, Front End Programming, CLR, CMA, DNZ, DSZ

**WHEN TIMER** This statement is used in conjunction with an integer and GOTO or GOSUB to transfer program control to a specified line number when the internal timer is equal to the integer in milliseconds. Upon execution of the WHEN TIMER statement, TIMER is set to the specified value and started.

**EXAMPLE:**

```
10 WHEN TIMER =1000 GOTO 100
20 GOTO 20
100 STOP
```

**WHEN XIOIN** This statement is used to transfer program control to a specified line number when data is present at pin 2 of the XIO port. The data byte is moved to the byte variable specified.

**EXAMPLE:**

```
10 WHEN XIOIN x GOTO 100
20 GOTO 20
100 STOP
```

### Front End Programming

**5.02** Each front end processor employs six operating registers and one counter. The six registers are labeled A, B, C, D, Y, and Z. Register A serves as the main operating register while the Y and Z registers are general purpose registers, each with a special buffer for handling block check schemes (CRC and LRC).

- CLR** This instruction is used to clear the BCC calculator by setting it to all zeros. It does not require an argument.
- CMA** This instruction complements the data in register A. It is used primarily for handling FCS calculations in bit oriented protocols and does not require an argument.
- DNZ** This instruction decrements an internal counter and skips the next instruction if the counter is not equal to zero. It does not require an argument.
- DSZ** This instruction decrements an internal counter and skips the next instruction if the counter is equal to zero. It does not require an argument.

## **FEM, HEX, INP, INU, IXO, IXU, JMP, MOV**

**FEM** A FEM (Front End Micro) code is similar to the machine language (MICRO PROCESSOR CODE) of the Intel 8085 used in the ENCORE. It permits access to the same areas accessed by the ENCORE's front end processor. The use of FEM is limited to specially trained programmers.

**EXAMPLE:**

```
10 FEM "320030"
```

**HEX** This instruction allows a byte to print on the screen in HEX. It can be used with the Y and Z registers only.

**INP** This instruction allows the ENCORE to input up to 256 characters. The number of characters to be input is specified by an argument of 0 to 255. A 0 argument will input 256 characters, while no argument defaults to 1 character. Immediately after execution, the A register will contain the last character received.

**EXAMPLE:**

```
10 INP 25
```

**INU** This instruction inputs up to 256 characters in the same manner as INP except that the BCC calculator is continually updated as each character is received.

**IXO** This instruction inputs up to 256 characters in a manner similar to that described for the INP instruction. This instruction, however, transfers each character to the main processor as it is received.

**IXU** This instruction combines the functions of IXO and INU. It inputs up to 256 characters, transfers each one to the main processor, and updates the BCC calculator.

**JMP** This instruction transfers program control to the line number specified by the number following JMP.

**EXAMPLE:**

```
10 JMP 20
```

**MOV** This instruction is used to transfer a data byte from one register to another without changing the contents of the source register. When entered, the specified registers are separated by the word "TO".

**EXAMPLE:**

```
10 MOV A TO Z  
20 MOV B TO Y
```

## MSK, NPE, PRE, QPE, RSN, SET, TAS

**MSK** This instruction performs a logical AND of the data in register A. The argument must be entered as a binary number. Leading zeros are optional. If an argument is not entered, the A register is set to all zeros.

**EXAMPLE:**

10 MSK 101

**NPE** This instruction turns off the parity flaw attribute regardless of data byte parity.

**PRE** This instruction presets the BCC calculator to all ones. No argument is required.

**QPE** This instruction turns on parity check ability.

**RSN** This instruction causes the ENCORE to drop sync and initiate a new search. It does not require an argument.

**SET** This instruction is used to set an internal counter to any value between 0 and 255. If an argument is not included, the counter is set to all zeros.

**EXAMPLE:**

10 SET 144

**TAS** This instruction tests the contents of register A for a flag or sync sequence, a binary number, or a character as outlined below:

- (a) If an argument is not entered, the register is tested for a flag or sync sequence and the next instruction is skipped if a flag or sync sequence is present.
- (b) If the argument is a binary number and that number is in register A, the next instruction is skipped.
- (c) If the argument is a character entered in quotes and the binary equivalent of that character is in register A with the correct parity bit, the next instruction is skipped.

**EXAMPLE:**

10 TAS  
20 TAS 101  
30 TAS "X"

## **TAS A, TDS, TDS A, UPD, XBC, XMT**

- TAS A** This instruction is used in the SDLC mode to detect the occurrence of an abort (7 or more 1's in succession). If an abort is detected, the next instruction is skipped.
- TDS** This instruction takes the same form as TAS except that the next instruction is skipped if the contents of register A does not agree with the argument.
- TDS A** This instruction takes the same form as TAS A except that the next instruction is skipped if an abort (7 or more 1's in succession) is not encountered.
- UPD** This instruction updates the BCC calculator with the data in register A. No argument is required.
- XBC** This instruction transfers the block check byte(s) from the Y and Z registers to the main processor. If the block check scheme employs a single character check, the transfer is between the Z register and the main processor. No argument is required.
- XMT** This instruction is used to transfer a data byte from any of the six operating registers to the main processor. When entered, the source register must be specified.

### **EXAMPLE:**

```
10 XMT A
20 XMT Z
```

**CHAPTER 8**  
**X25/X75 OPERATION AND PROGRAMMING**

**1. GENERAL**

**1.01** This chapter provides the user with a background to the development of X25 and briefly describes the frame and packet level formats used for the exchange of information between DTE and DCE. It also contains the information required for operation of the ENCORE as an X25/X75 monitor or interactive simulator, including a detailed explanation of each operating mode followed by simple step-by-step operating procedures. The X25/X75 Applications Package is offered as option E252. For additional information, please call our Customer Service Department at (203) 438-3731.

**PURPOSE**

**1.02** Acceptance of the X25 and X75 recommendations as international standards for data communications has prompted DIGITECH to develop comprehensive applications software packages for use with the ENCORE 200. It employs a user-friendly, menu driven operating system requiring the absolute minimum in user intervention.

**2. BACKGROUND**

**2.01** Adopted by the International Telegraph and Telephone Consultative Committee (CCITT) in 1976, X25 has been accepted by public and private networks throughout the world as the standard interface protocol for packet-switched communications. It was developed to permit communication between dissimilar equipment while sharing installed bandwidth, centralizing the message buffering hardware, and providing point-to-point error checking. It allows the terminal equipment of one subscriber to communicate with the equipment of any other subscriber by simply presenting data to the network in a specified format.

**2.02** From an operational point of view, packet switching is the transmission of individually addressed packets which may comprise all or only part of the entire message. Because each packet carries its own address information, it may be transmitted over a different circuit to the same destination. This eliminates wasting network bandwidth when waiting for replies. The user simply delivers packets to the network and the network assumes responsibility for managing the flow of data.

**2.03** The X25 recommendation describes the interface between Data Terminal Equipment (DTE) and the Data Circuit Equipment (DCE) used on a packet-switched network. It is comprised of three levels of standards. Level-1 defines the physical and electrical interface between DTE and DCE. Level-2 defines the characteristics and format of the data frame. Level-3 defines the characteristics and format of the data or information packet within the frame. The following paragraphs, 4.01 through 6.13, are supplied as a quick reference to the frame level and packet level formats described in the X25 Standard. Operation of the ENCORE in the X25/X75 environment is covered in paragraphs 7.01 through 18.01.

**3. LEVEL-1, PHYSICAL INTERFACE**

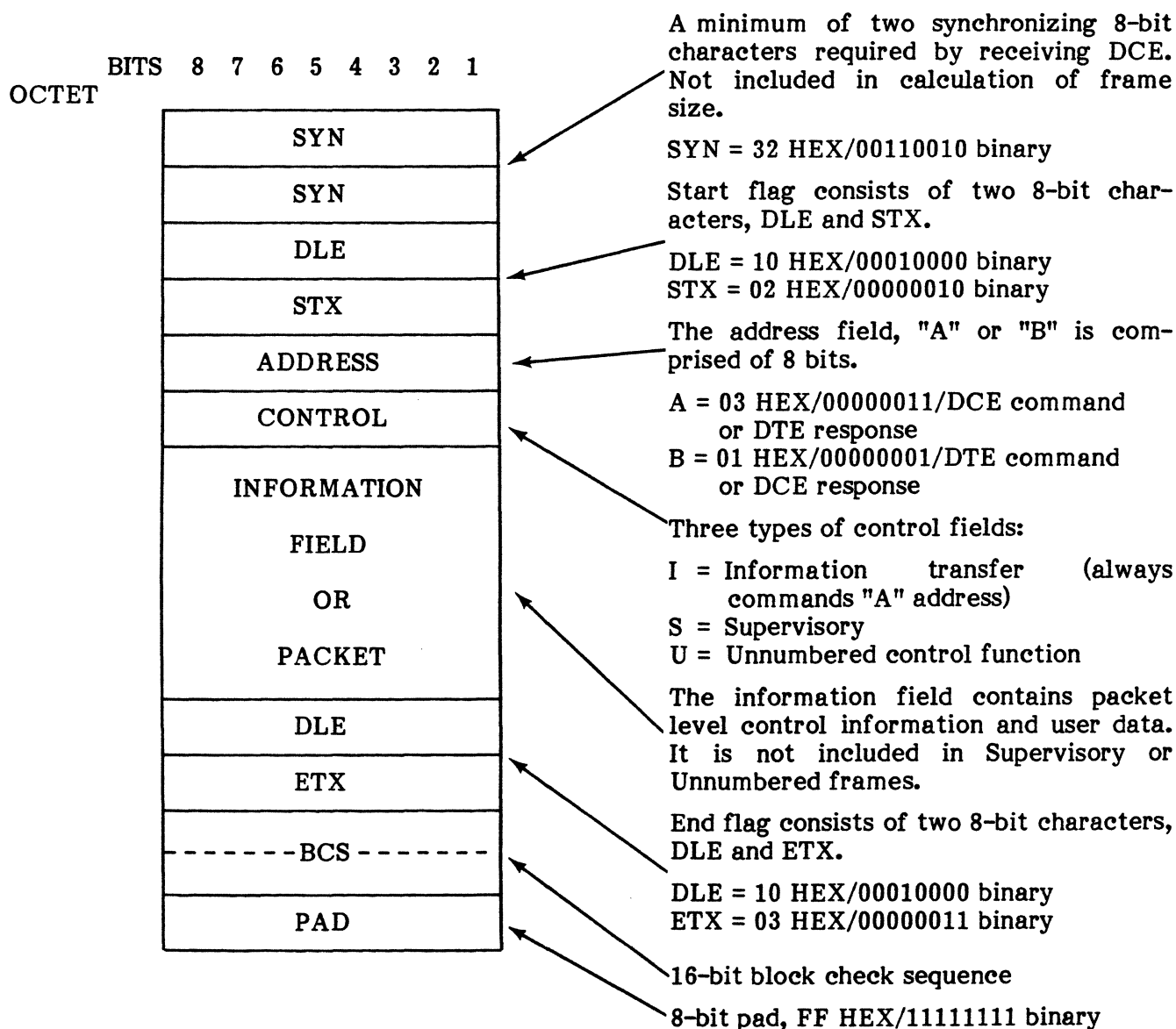
**3.01** The Level-1 interface is that portion of the X25 recommendation describing the physical, electrical, and functional characteristics of the link between DTE and DCE. The ENCORE assumes that this interface is RS-232 compatible. Information regarding this interface can be found in the appropriate specifications.

**4. LEVEL-2, FRAME LEVEL**

**4.01** Level-2 describes the full-duplex Link Access Procedure (LAP or LAPB) used between DTE and DCE. The frame structure differs slightly depending on whether the bit oriented (HDLC/SDLC) or character oriented (BSC) mode is initialized.

**CHARACTER ORIENTED FRAMING (BISYNC, BSC)**

**4.02** Figure 8-1 shows the structure of a character oriented frame. This type of frame begins with two special synchronizing bit patterns used to place the transmitter and receiver in character phase. These synchronization characters and the pad character shown are not considered part of the actual frame when calculating frame size. Each frame transmitted by DCE is preceded by three synchronizing characters (SYN, HEX 32, binary 00110010) used to establish character phase. Each frame received by DCE must be preceded by at least two synchronizing characters. The pad character (HEX FF, binary 11111111) is required by some interface hardware to ensure that the last 8 bits of the Block Check Sequence (BCS) are shifted into the hardware register. Therefore, the pad character must immediately follow the BCS, but need not be checked by the receiver. The address field, control field, and information field are the same for both character oriented and bit oriented frames and are discussed in paragraphs 4.07 through 4.12. The flag sequences, transparency, and error checking scheme of the character oriented frame are discussed in the following paragraphs. Bit number 1 is the least significant bit and it is transmitted first.



**Fig. 8-1 Character Oriented (BSC) Frame Structure**

## CHARACTER ORIENTED FRAMING (BISYNC, BSC)

**4.03 Start Flag.** The start flag sequence consists of two characters, Data Link Escape (DLE, HEX 10, binary 00010000) and Start of Text (STX, HEX 02, binary 00000010). Once synchronization is established, the receiving station expects this flag sequence. If any other sequence is received, the station initiates a sync search reverting to bit synchronization in an effort to establish character phase once again.

**4.04 End Flag.** The end flag sequence consists of the two characters DLE and End of Text (ETX, HEX 03, binary 00000011). Once the end flag is correctly received, the receiving station is prepared to receive the 16-bit BCS. To accomplish this, the ETX must be preceded by an odd number of DLE characters.

**4.05 Transparency.** Character-oriented framing provides transparency for data within the information field. Whenever a DLE (HEX 10, binary 00010000) appears in the data, an additional DLE is inserted adjacent to it to identify it as data. The inserted DLE is later discarded and is not included in the BCS calculation. Whenever a 2-character DLE sequence is encountered, (e.g., DLE STX, DLE ETX), the second DLE is not inserted. The DLE sequences are interpreted as follows:

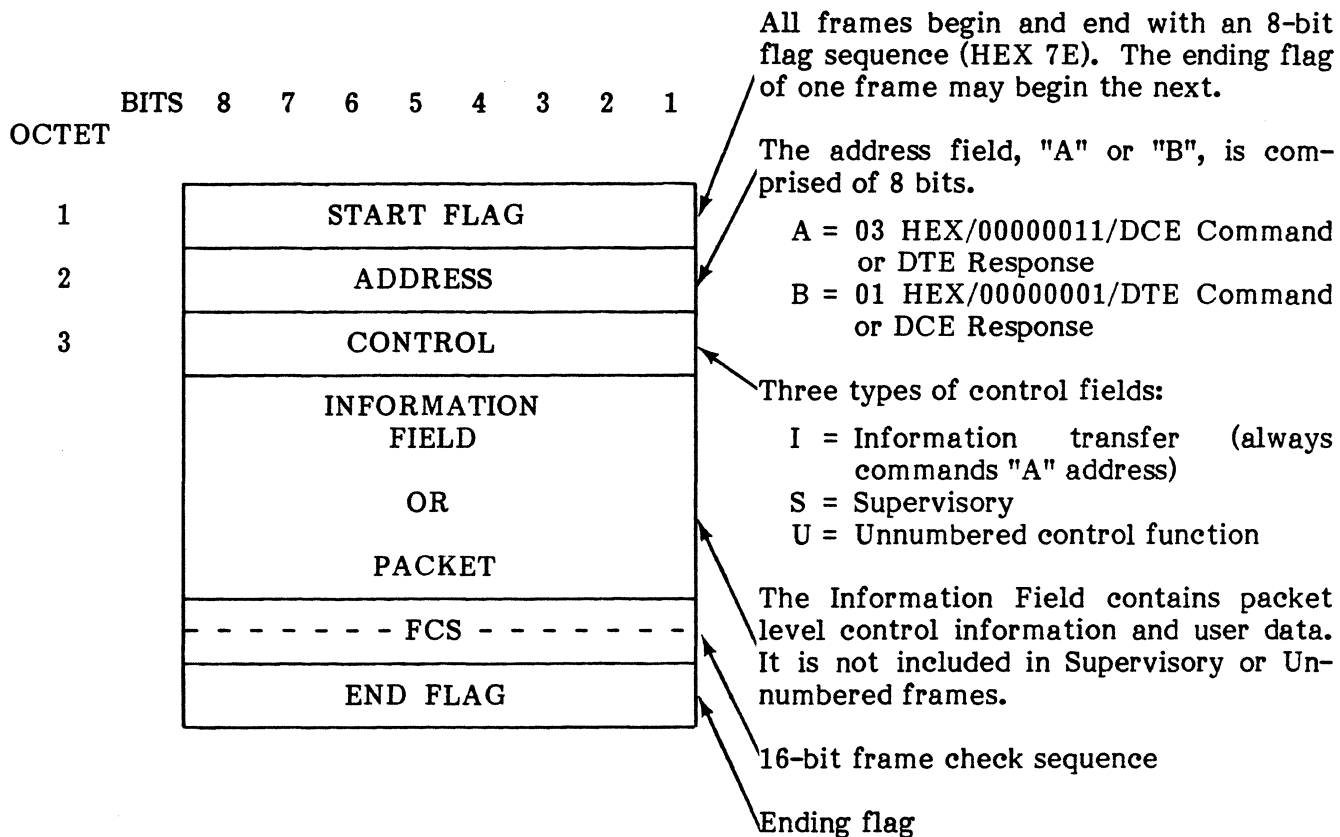
- DLE STX indicates the start of transparency.
- When two DLE characters appear adjacent to each other, one is suppressed and the other is considered data.
- If the 2-character sequence DLE ETX is not immediately preceded by an odd number of DLE characters, it indicates the end of transparency.
- If the 2-character sequence DLE SYN is not immediately preceded by an odd number of DLE characters, it is discarded. The discarded sequence is not included in the BCS calculation.

**4.06 Block Check Sequence.** All character oriented frames include a 16-bit Block Check Sequence (BCS) immediately following the end flag for error detection purposes. The start flag (DLE STX) and the synchronization time-fill (DLE SYN) are not included in the BCS calculation. The calculation is based on the contents of the address, control, and information fields, excluding inserted DLE characters. The end flag ETX character is also included in the BCS calculation, but the end flag DLE is not.



**BIT ORIENTED FRAMING (HDLC/SDLC)**

**4.07** A typical bit oriented frame is constructed as shown in Figure 8-2. It includes a beginning and ending flag, an address field, a control field, and depending on packet type, an information field. Each type of frame and the various fields which may comprise a frame are discussed in the following paragraphs. Bit number 1 is the least significant bit and it is transmitted first. Bits 1 through 8 comprise a single octet.



**Fig. 8-2 Bit Oriented (HDLC/SDLC) Frame Structure**

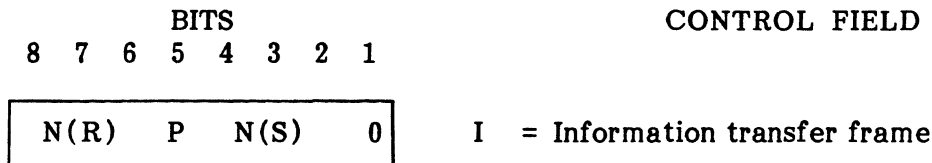
**4.08 Flag Sequence Field.** The flag is a unique sequence of eight bits (01111110, or HEX 7E) which begin and end a frame. A single flag can be used to end one frame while beginning another.

**4.09 Address Field.** The address is comprised of eight bits and is either an "A" or "B". The "A" (HEX 03, 00000011) indicates a DCE command or DTE response. The "B" (HEX 01, 00000001) indicates a DTE command or DCE response.

**4.10 Control Field.** Information is transferred as either a command or a response and is controlled using one of the following three basic control field formats. As shown in Figures 8-3 and 8-5, these formats specify the type of frame and appropriate commands or responses.

## BIT ORIENTED FRAMING (HDLC/SDLC)

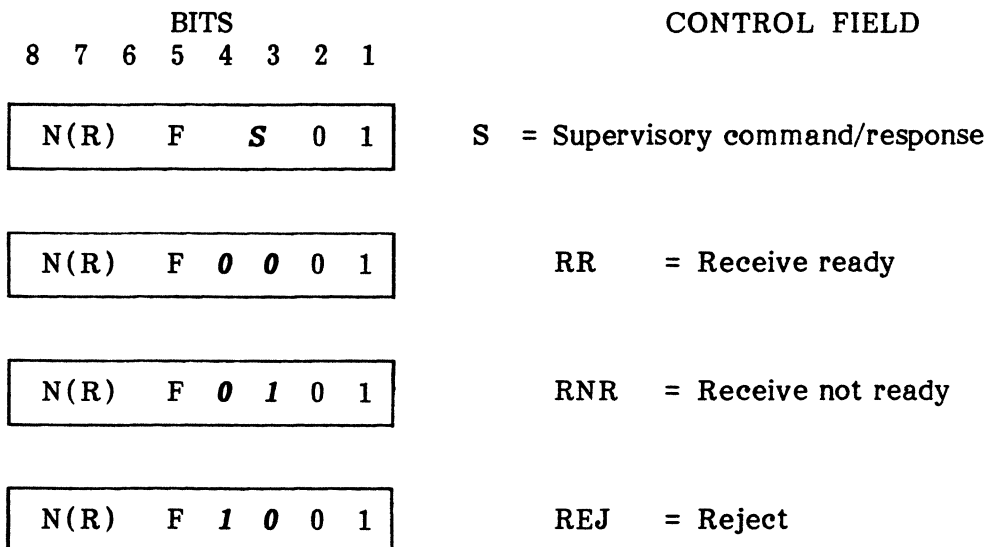
- (a) **Information Frames.** Frames of this type are designed to sequentially transfer information. The control field includes the send and receive frame counts, which may or may not acknowledge additional frames received by the DTE or DCE, and the P/F (poll or final) bit.



**Notes:** N(S) = Transmitter send sequence number  
 N(R) = Transmitter receive sequence number  
 P = Poll bit (primary)

**Fig. 8-3 Information Transfer Frame**

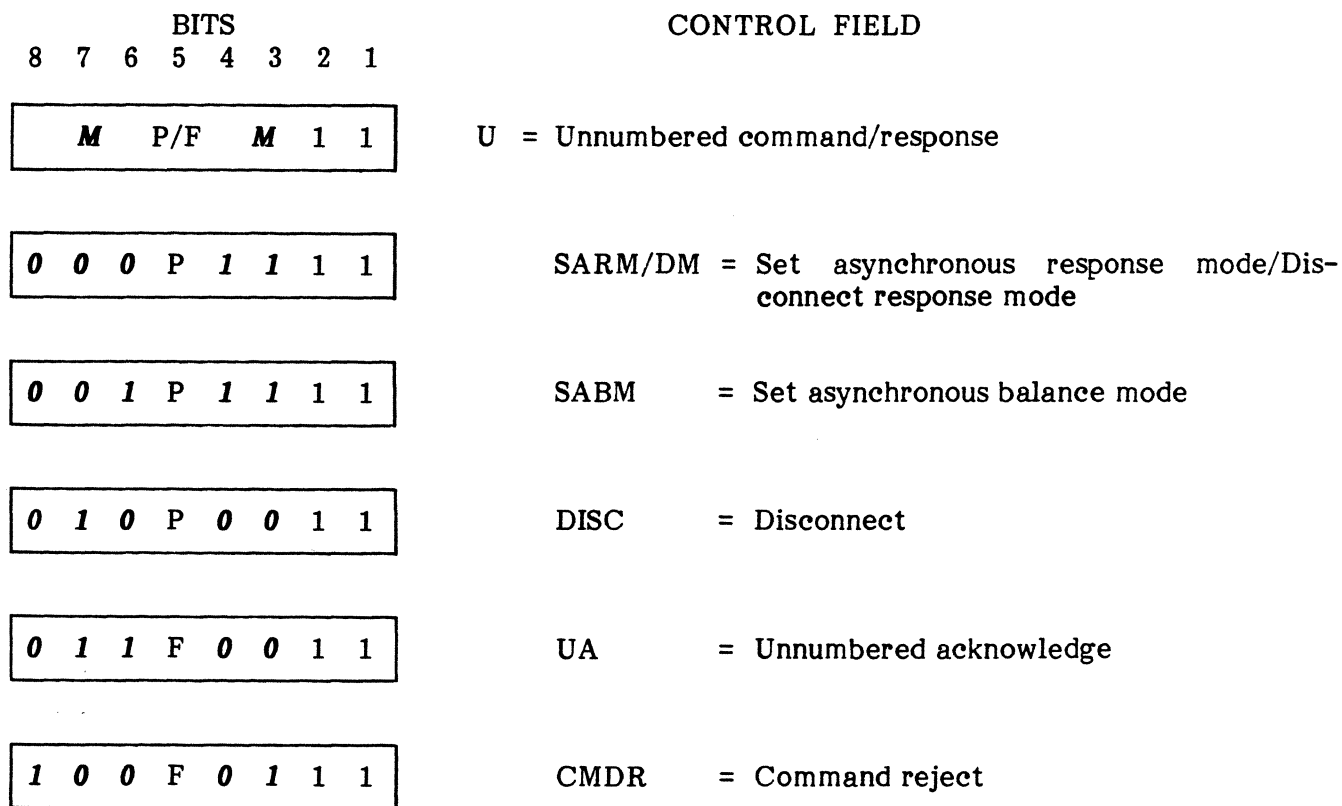
- (b) **Supervisory Frames.** Supervisory frames are used to perform such link control functions as the request to retransmit information frames (REJ), the acknowledgement of previously received information frames (RR), and the temporary suspension of information frame transmissions (RNR).



**Notes:** N(R) = Transmitter receive sequence number  
 F = Final bit (secondary)  
 S = Supervisory function

**Fig. 8-4 Supervisory Command/Response Frames**

- (1) **REJ.** The Reject supervisory frame is used to request retransmission of the information frames. These frames begin with the rejected frame and end with the last unacknowledged frame. Additional information frames may be transmitted after these frames are retransmitted.
  - (2) **RR.** The Receive Ready supervisory frame is used as a response to indicate that a station is ready to receive frames after a temporary suspension due to a busy (RNR) condition. It is also used to acknowledge previously received information frames up to and including N(R)-1, when not acknowledged by an information frame and as a command to query the state of the other station.
  - (3) **RNR.** The Receive Not Ready supervisory frame is a secondary response used to indicate a busy condition. The transmission of a valid UA, RR, SABM, or SARM indicates that the busy condition is cleared.
- (c) **Unnumbered Frames.** Unnumbered frames are used to provide additional link control functions. They contain no send or receive frame sequence numbers.



**Notes:** **M** = Modifier function  
 P/F = Poll bit (final) or final bit (secondary)

**Fig. 8-5 Unnumbered Command/Response Frames**

## BIT ORIENTED FRAMING (HDLC/SDLC)

- (1) **SARM/DM.** The Set Aynchronous Response Mode (SARM) is an unnumbered command used to place the addressed station in the Aynchronous Response Mode (ARM). No information field is permitted with the SARM command. A DTE or DCE confirms acceptance of SARM by the transmission of a UA response at the first opportunity. Upon acceptance of this command, the DTE or DCE receive state variable V(R) is set to zero. Previously transmitted information frames that are unacknowledged when this command is actioned remain unacknowledged. The SARM command may be used by the DTE or DCE in the Information Transfer Phase to reset one direction of information transmission.

The Disconnect Mode (DM) response is used to report a status where the DTE or DCE is logically disconnected from the link and is in the Disconnect Phase. The DM response is sent to request a SARM/SABM command, informing the primary that the secondary is still in Disconnect Phase and cannot action the previous SARM/SABM command. No information is permitted with the DM response.

- (2) **SABM.** The Set Aynchronous Balance Mode (SABM) is an unnumbered command used to place the addressed station in the Aynchronous Balance Mode (ABM) Information Transfer Phase. No information field is permitted with the SABM command. A DTE or DCE confirms acceptance of SABM by the transmission of a UA response at the first opportunity. Upon acceptance of this command, the DTE or DCE send state variable V(S) and the receive state variable V(R) are set to zero. Previously transmitted information frames that are unacknowledged when this command is actioned remain unacknowledged.
- (3) **DISC.** The Disconnect (DISC) is an unnumbered primary command used to terminate the mode previously set. It informs the secondary that operation is suspended and is acknowledged by secondary transmission of a UA response at the first opportunity. This type of frame does not include an information field. Unacknowledged frames transmitted prior to DISC remain unacknowledged.
- (4) **UA.** The Unnumbered Acknowledge is a secondary response used to acknowledge the Unnumbered (U) control frame command. The secondary transmits the response prior to any action directed by the command. This type of frame does not include an information field.
- (5) **CMDR.** The Command Reject (CMDR) is a secondary response used to report an error condition that retransmission of the error frame will not correct. The information field immediately following the control field consists of three octets and is returned with this response to provide the reason for the CMDR.

**4.11 Information Field.** The information field normally contains the packet data which is transmitted as part of an information frame. Although unrestricted with respect to code or grouping of bits, the field length is a specified parameter of the system. An information field is also transmitted as part of an unnumbered Command Reject response (CMDR). In this case, it consists of three octets which include the reason for reject.

**4.12 Frame Check Sequence (FCS).** The FCS is a 16-bit sequence immediately preceding the frame end flag. It is used by the receiving device to detect the presence of transmission errors.

**5. LEVEL-3, PACKET LEVEL**

**5.01** This section of the X25 recommendation describes the packet structure and procedures for the exchange of packets between DTE and DCE. It explains how calls are established, maintained and cleared, and how user data and control information are structured into packets for presentation to the networks. The following paragraphs present this information to the user in an abbreviated form. For additional information refer to the X25 or appropriate network specification.

**5.02** Level-3 packets contain control information and user data issued by the called and calling DTE's. Each packet includes some or all of the following information depending on its type.

- General Format Identifier
- Logical Channel Group Number
- Logical Channel Number
- Packet Type Identifier
- Called DTE Address Length
- Calling DTE Address Length
- Called DTE Address
- Calling DTE Address
- Facility Field Length
- Facility Field
- User Data

**5.03 General Format Identifier.** The general format identifier consists of the first four bits of octet one. It is used to indicate the packet sequence numbering scheme which is either modulo 8 or 128 as shown in Table 8-1. If the numbering scheme is modulo 8, no more than 8 outstanding packets are permitted. If the scheme is modulo 128, no more than 128 outstanding packets are permitted.

**TABLE 8-1**  
**GENERAL FORMAT IDENTIFIER**

PACKET TYPE	MODULO	OCTET 1			
		BITS 8	7	6	5
Data (0-7 outstanding packets)	8	x	0	0	1
Data (0-127 outstanding packets)	128	x	0	1	0

**Note:** x = 1 or 0 as described in subsequent paragraphs.

## LEVEL-3, PACKET LEVEL

**5.04 Logical Channel Group Number.** This binary number is coded into bits 1 through 4 of octet one. It is included in every packet except those used for restart. For switched circuits, this number (0 to 15) is assigned during call set-up. For permanent virtual circuits, the number is assigned according to prior agreement between network and subscriber.

**5.05 Logical Channel Number.** This 8-bit binary number is coded into octet two. It is included in every packet except those used for restart. For switched circuits, this number (0 to 255) is assigned during call set-up. For permanent virtual circuits, the number is assigned according to prior agreement between network and subscriber.

**5.06 Packet Type Identifier.** As shown in Table 8-2, octet three is used to identify the various types of packets. Packet types are detailed later in paragraphs 5.14 through 5.31. A question mark (?) following the packet name indicates a request; an exclamation point (!) indicates an accept.

**TABLE 8-2**  
**PACKET TYPE IDENTIFIER**

DCE/DTE	PACKET TYPE NAME	DTE TO DCE	HEX LD.	OCTET 3								
				BITS	8	7	6	5	4	3	2	1
<i>Call Set-up and Clearing</i>												
Incoming Call	CAL?	Call Request	0B		0	0	0	0	1	0	1	1
Call Connected	CAL!	Call Accepted	0F		0	0	0	0	1	1	1	1
Clear Indication	CLR?	Clear Request	13		0	0	0	1	0	0	1	1
DCE Clear Confirmation	CLR!	DTE Clear Confirmation	17		0	0	0	1	0	1	1	1
<i>Data and Interrupt</i>												
DCE Data		DTE Data			x	x	x	x	x	x	x	0
DCE Interrupt	INT?	DTE interrupt	23		0	0	1	0	0	0	1	1
DCE Interrupt Confirmation	INT!	DTE Interrupt Confirmation	27		0	0	1	0	0	1	1	1
<i>Flow Control and Reset</i>												
DCE Receive Ready	RR	DTE Receive Ready	01		x	x	x	0	0	0	0	1
DCE Receive Not Ready	RNR	DTE Receive Not Ready	05		x	x	x	0	0	1	0	1
	REJ	DTE Reject	09		x	x	x	0	1	0	0	1
Reset Indication	RSR?	Reset Request	1B		0	0	0	1	1	0	1	1
DCE Reset Confirmation	RSR!	DTE Reset Confirmation	1F		0	0	0	1	1	1	1	1
<i>Restart</i>												
Restart Indication	RST?	Restart Request	FB		1	1	1	1	1	0	1	1
DCE Restart Confirmation	RST!	DTE Restart Confirmation	FF		1	1	1	1	1	1	1	1

**Note:** x = 0 or 1 as described in subsequent paragraphs.

**5.07 Called DTE Address Length.** Bits 1, 2, 3, and 4 of octet 4 indicate the length of the called DTE address in semi-octets (half an octet or 4 bits). The address length is binary coded and bit 1 is the least significant bit.

**5.08 Calling DTE Address Length.** Bits 5, 6, 7, and 8 of octet 4 indicate the length of the calling DTE address in semi-octets. The address length is binary coded and bit five is the least significant bit.

**5.09 Called DTE Address.** Octet 5 contains the first one or two digits (one digit per semi-octet) of the called address. It is followed by the number of semi-octets required to complete the address as specified by bits 1 through 4 of octet 4.

**5.10 Calling DTE Address.** This address is contained in the semi-octets immediately following the called address. The number of semi-octets in the address is determined by bits 5 through 8 of octet 4. Bits 1 through 4 of the last octet in the address may be zero filled as the address field must always contain an integral number of octets.

**5.11 Facility Field Length.** This field consists of bits 1 through 6 of the first octet to follow the calling address. This field is a binary number indicating the total number of octets in the facility field (62 max).

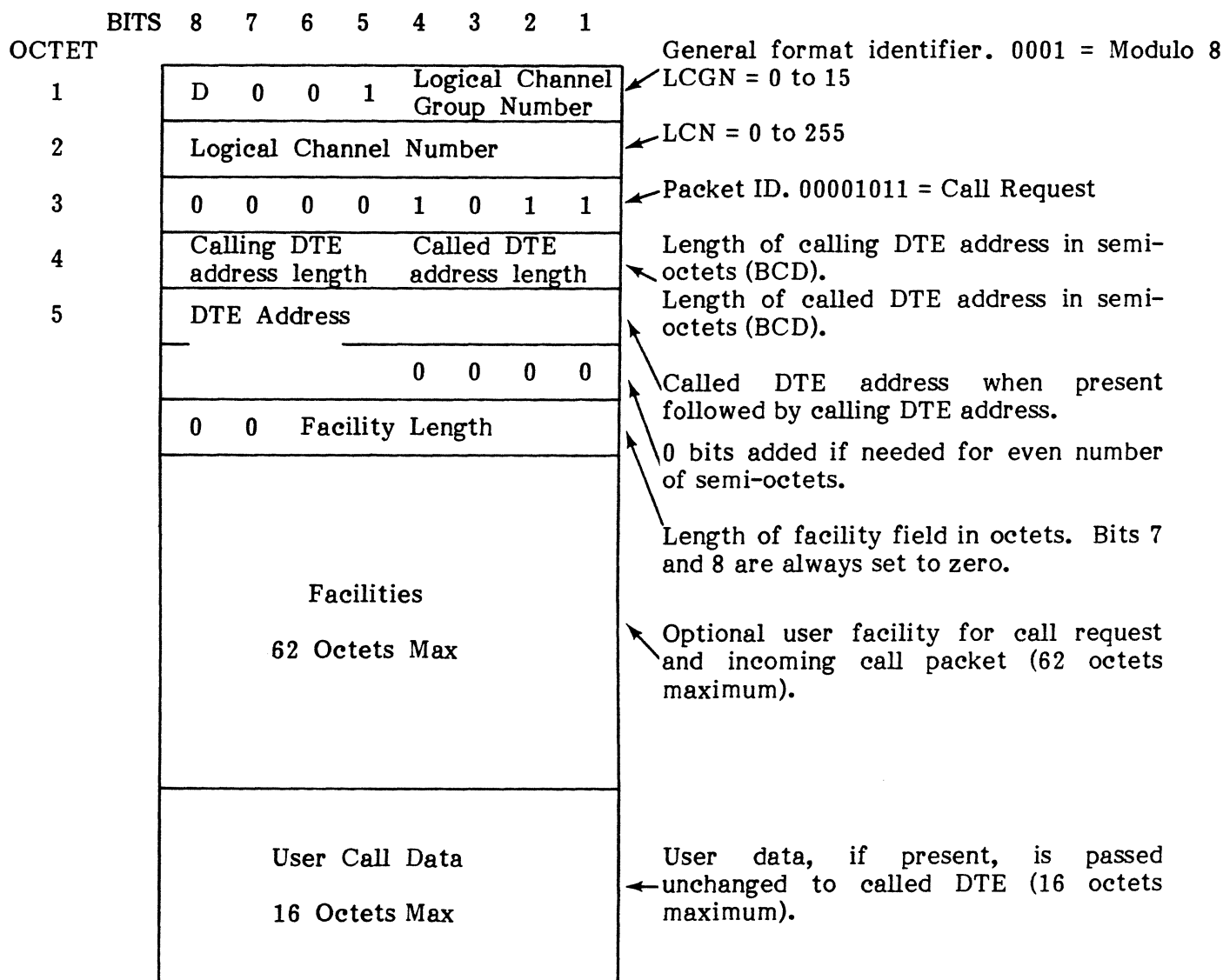
**5.12 Facility Field.** This field is only present when DTE is using an optional user facility requiring some indication in the call request and incoming call packets. The field contains a sequence of octet pairs. The first octet of each pair is a facility code indicating the facility or facilities requested. The second octet is the facility parameter. The facility field contains an integral number of octets. The actual length of the field depends on the facilities offered by the network but it cannot exceed 62 octets.

**5.13 User Data.** In the call request or incoming packet, up to 16 octets of user data may follow the facility field. In the absence of an optional user facility field, the maximum data field length is a fixed system parameter common to all logical channels.

CALL SET-UP AND CLEARING PACKETS

5.14 Call set-up and clearing packets include four packet types: the call request or incoming call packet, the call connected or call accepted packet, the clear request or clear indication packet, and the clear confirmation packet. These packets are used to establish switched virtual circuits between DTE's.

5.15 CAL? If this packet is transmitted by DTE, it is considered a call request and indicates that DTE wishes to set-up a virtual circuit. When received by DTE, it is considered an incoming call packet and indicates that a calling DTE wishes to establish a virtual circuit. In a call request packet, bit 8 of the general format identifier is considered the D-bit (delivery configuration). It is set to 1 if DTE wishes to receive an end-to-end acknowledgement of its transmitted data. A typical call request or incoming call packet format is shown in Figure 8-6.

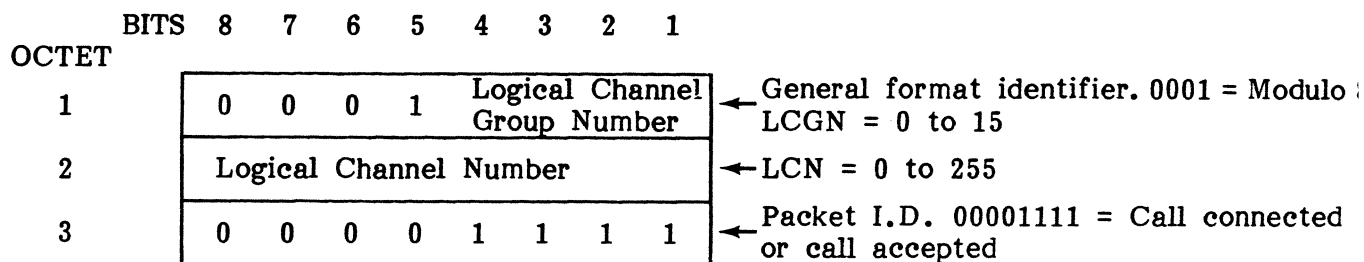


**Note:** The figure assumes that a single address is present consisting of an odd number of digits and that the call user data field is an integral number of octets in length.

Fig. 8-6 Call Request or Incoming Call Packet

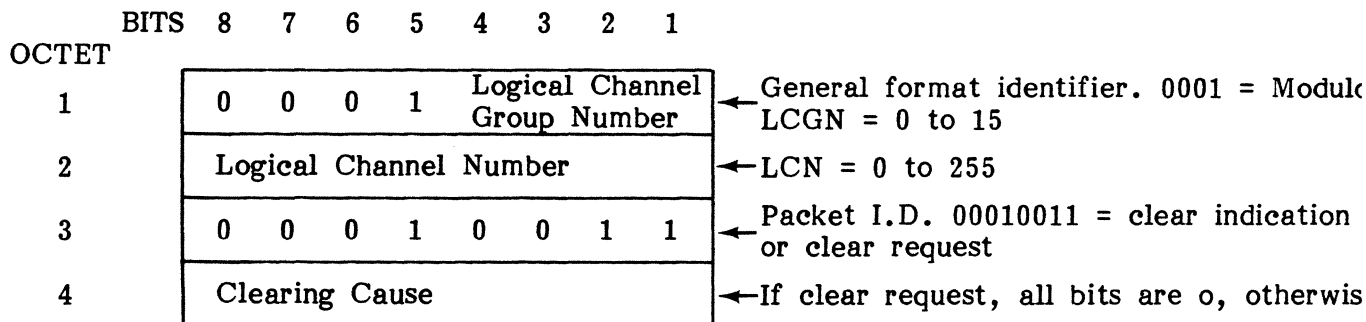


**5.16 CAL!** If a called DTE accepts an incoming call, it transmits this packet as a call accept to the calling DTE. The calling DTE receives the packet as a call connected indication. The call accepted or call connected packet format, consisting of three octets, is shown in Figure 8-7.



**Fig. 8-7 Call Accepted or Call Connected Packet**

**5.17 CLR?** If the called DTE cannot accept an incoming call, it transmits a clear request packet. The calling DTE receives this as a clear indication packet. The clear request or clear indication packet consists of four octets as shown in Figure 8-8.

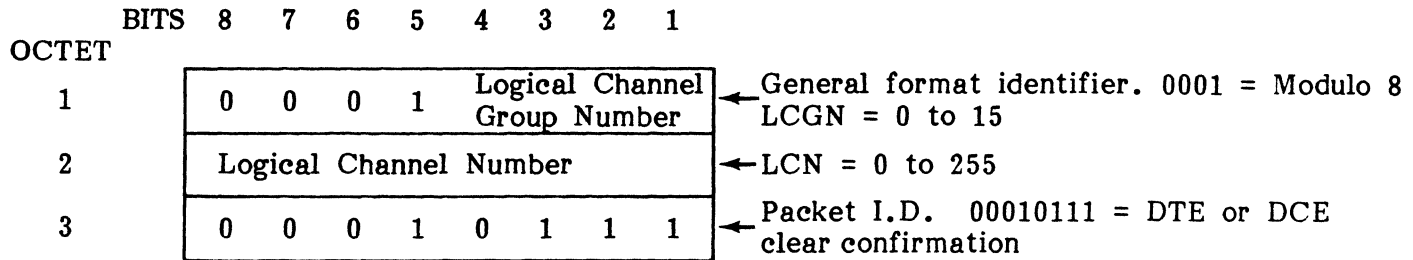


CLEARING CAUSE	HEX	BITS							
		8	7	6	5	4	3	2	1
Number Busy	= 01	= 0	0	0	0	0	0	0	1
Out of Order	= 09	= 0	0	0	0	1	0	0	1
Remote Procedure Error	= 11	= 0	0	0	1	0	0	0	1
Number Refuses Reverse Charging	= 19	= 0	0	0	1	1	0	0	1
Invalid Call	= 03	= 0	0	0	0	0	0	1	1
Access Barred	= 0B	= 0	0	0	0	1	0	1	1
Local Procedure Error	= 13	= 0	0	0	1	0	0	1	1
Network Congestion	= 05	= 0	0	0	0	0	1	0	1
Not Obtainable	= 0D	= 0	0	0	0	1	1	0	1

**Fig. 8-8 Clear Request or Clear Indication Packet**

## CALL SET-UP AND CLEARING PACKETS, DATA AND INTERRUPT

**5.18 CLR!.** The clear confirmation packet is transmitted by a calling DTE to confirm a clear indication packet when the called DTE cannot accept an incoming call. It may also be transmitted by either DTE when data transfer is complete. The three octets of the clear confirmation packet are shown in Figure 8-9.

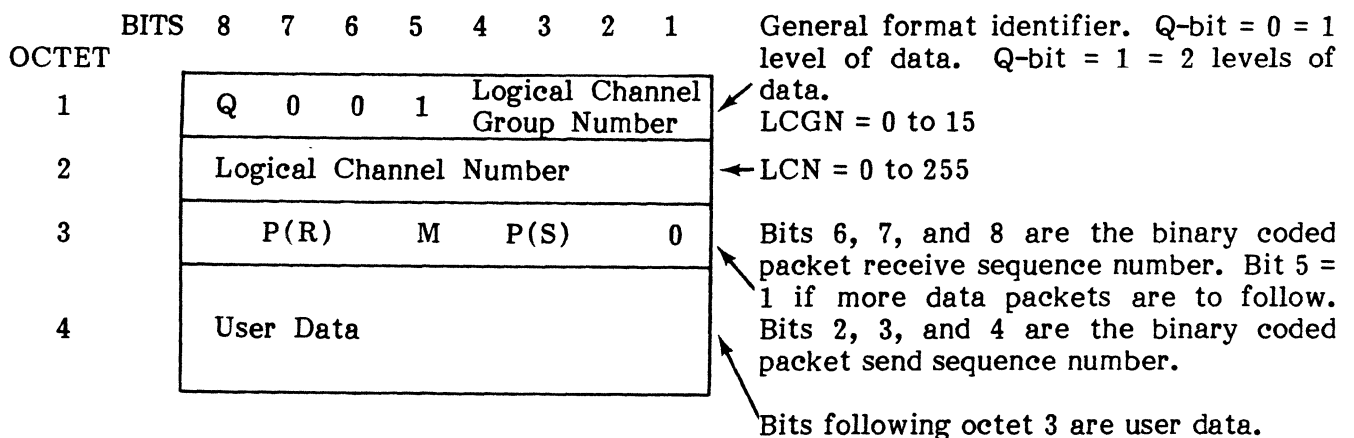


**Fig. 8-9 Clear Confirmation Packet**

### DATA AND INTERRUPT

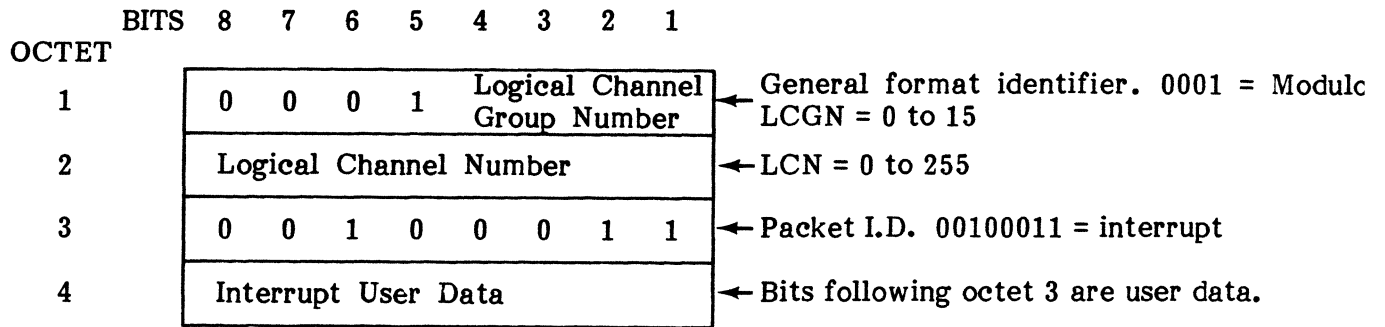
**5.19** Data and interrupt packets provide for the transfer of user data and interrupt conditions between DTE and DCE during the information transfer state. These packets include the data packet, the interrupt packet, and the interrupt confirmation packet as described in the following paragraphs.

**5.20 Data Packet.** Packet data may be transmitted in two levels as indicated by the Q-bit. Each data packet is sequentially numbered from 0 to 7 (modulo 8) or 0 to 127 (modulo 128). This number is called the packet send or receive sequence number. The first packet transmitted is always given the number zero. Level-1 and Level-2 packets may not be intermixed. Typical data packet format is shown in Figure 8-10.



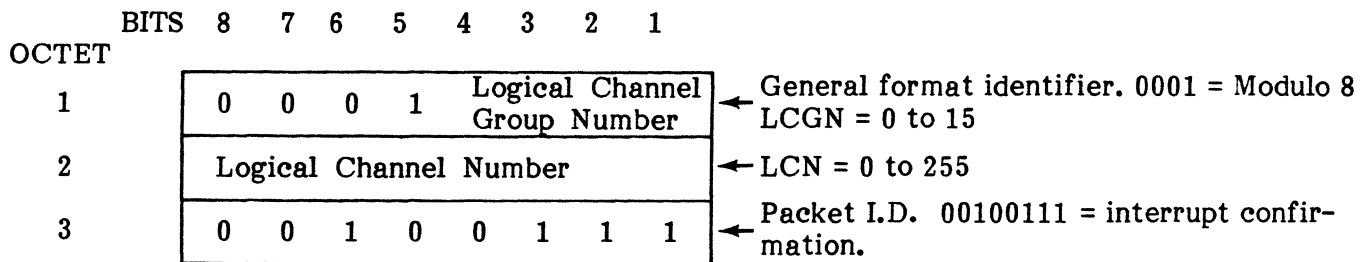
**Fig. 8-10 Data Packet**

**5.21 INT?** Nonsequenced interrupt packets permit bypass of the normal data packet transmission sequence. This allows an interrupt (e.g., break key on terminal keyboard) to be transmitted without being subject to the flow control imposed on data packets. The interrupt packet consists of four octets as shown in Figure 8-11.



**Fig. 8-11 Interrupt Packet**

**5.22 INT!** This packet is used to confirm the interrupt condition. It consists of three octets as shown in Figure 8-12.



**Fig. 8-12 Interrupt Confirmation Packet**

FLOW CONTROL AND RESET

5.23 Flow control and reset packets control the transfer of data packets and reset conditions on each logical channel. The transmission of data is controlled separately for each direction and is based on the appropriate response from the receiver. These packets include the receive ready, receive not ready, reject, reset request, and reset confirmation packets as described in the following paragraphs.

5.24 RR. Receive Ready packets are used by DTE or DCE to indicate that it is ready to receive data packets beginning with a specified sequence number, P(R). The RR packet consists of three octets as shown in Figure 8-13.

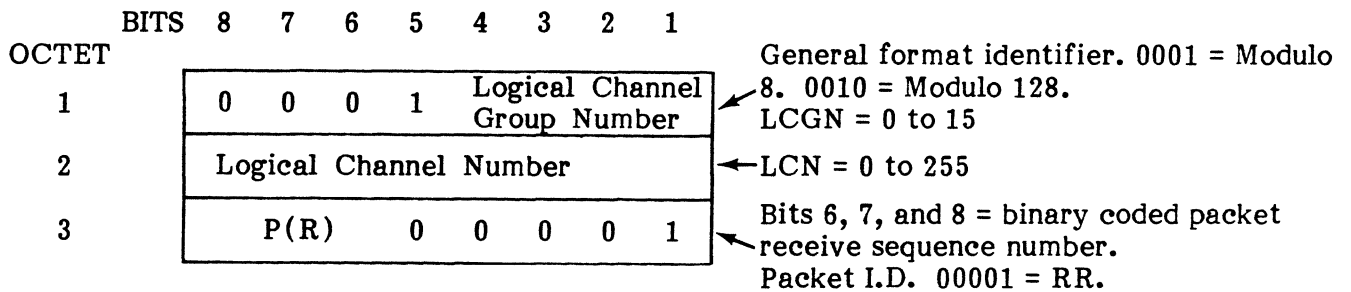


Fig. 8-13 Receive Ready Packet

5.25 RNR. The Receive Not Ready packet is used by DTE or DCE to indicate a temporary inability to receive data packets. When an RNR packet is received, it signals the DTE or DCE to stop transmitting data packets on the specified logical channel. RNR is cleared by RR or reset. The RNR packet consists of three octets as shown in Figure 8-14.

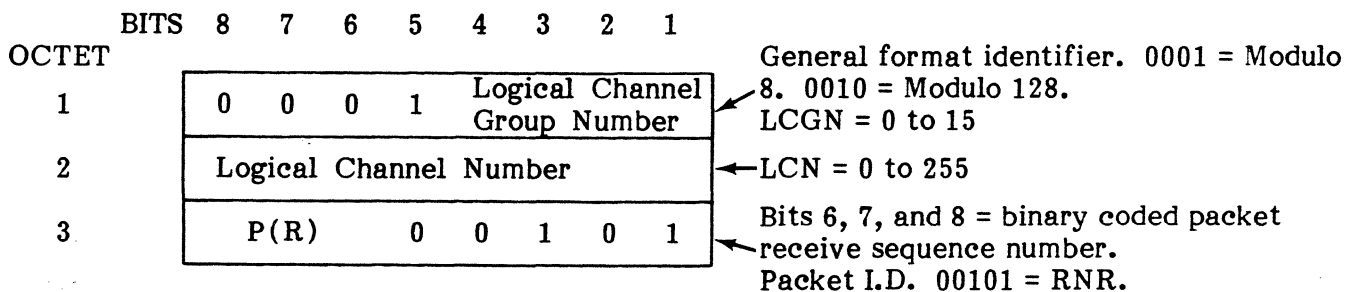
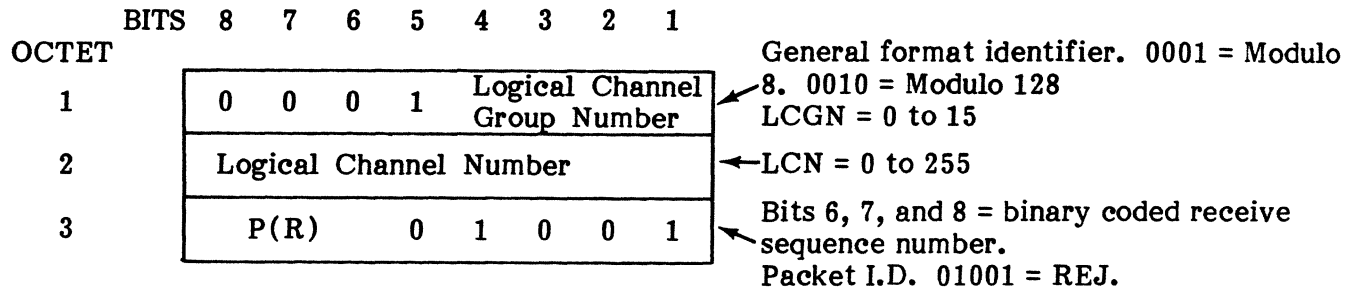


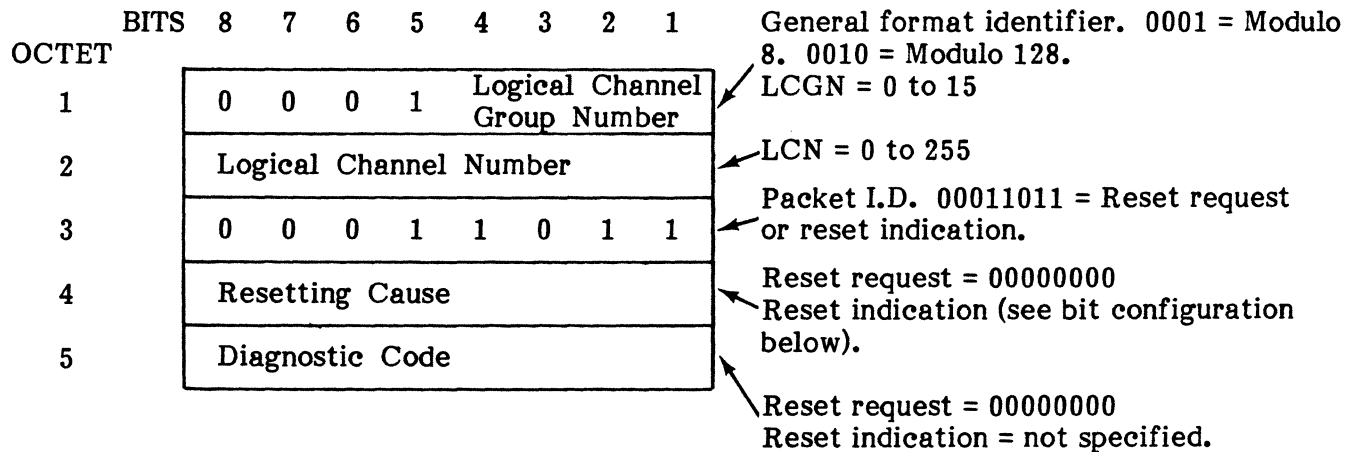
Fig. 8-14 Receive Not Ready Packet

**5.26 REJ.** The DTE reject packet initiates retransmission of data packets as specified by the P(R). The REJ packet consists of three octets as shown in Figure 8-15.



**Fig. 8-15 Reject Packet**

**5.27 RSR?** The reset request is used to reinitialize the virtual circuit removing all data and interrupt packets from the network. The reset request is initiated by DTE and specifies the logical channel (LCN) to be placed in the Reset Request State. DCE indicates reset by transmitting a reset indication packet which specifies the LCN and the reason for reset. This places the logical channel in the DCE Reset Indication State where the DCE will ignore data, interrupt, RR, and RNR packets. This type of packet consists of five octets and is shown in Figure 8-16.

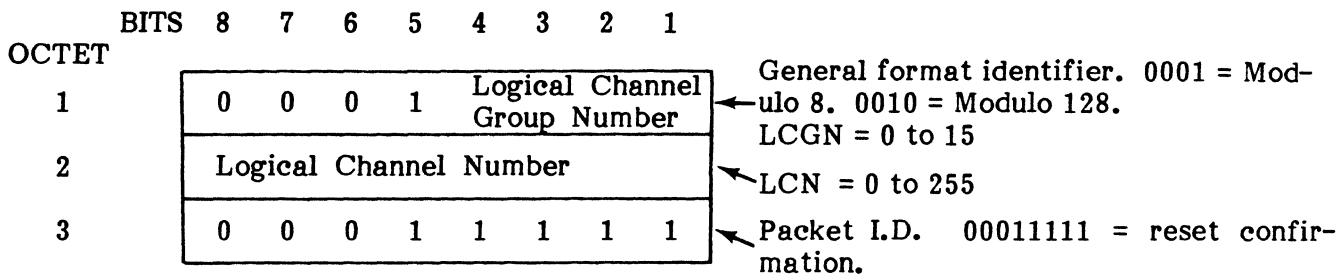


RESETTING CAUSE	HEX	BITS								
		8	7	6	5	4	3	2	1	
Out of Order	= 01	= 0	0	0	0	0	0	0	0	1
Remote Procedure Error	= 03	= 0	0	0	0	0	0	0	1	1
Local Procedure Error	= 05	= 0	0	0	0	0	0	1	0	1
Network Congestion	= 07	= 0	0	0	0	0	0	1	1	1

**Fig. 8-16 Reset Request Packet**

## FLOW CONTROL AND RESET, RESTART

**5.28 RSR!.** When in the DTE Reset Request State, DCE confirms reset by sending DTE a reset confirmation packet. When in the DCE Reset Indication State, DTE confirms reset by sending DCE a reset confirmation packet. The reset confirmation packet consists of three octets as shown in Figure 8-17.

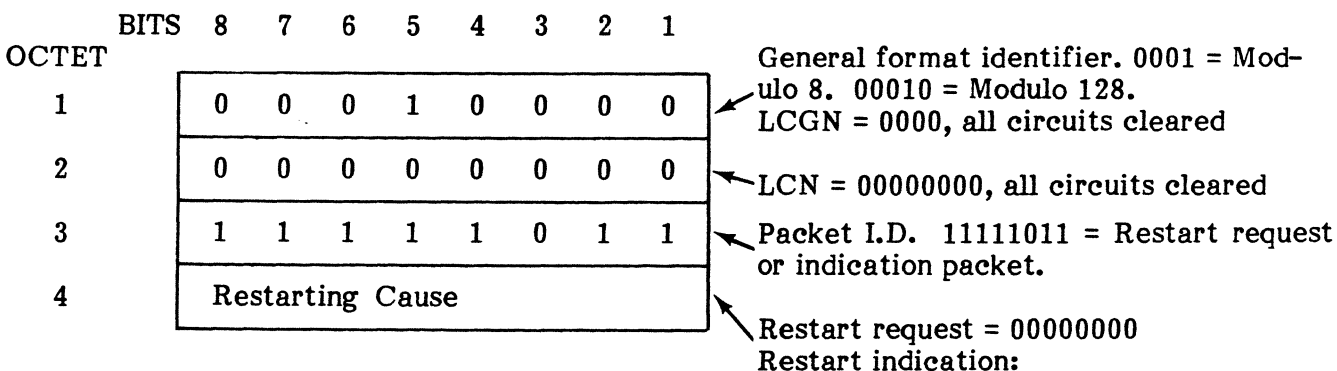


**Fig. 8-17 Reset Confirmation Packet**

### RESTART

**5.29** Restart packets are used to clear all switched virtual circuits and reset all permanent virtual circuits of the DTE/DCE interface. This allows the system to recover from major failures and return to the state that it was in when service was initiated. These packets include the restart request and restart confirmation packets as described in the following paragraphs.

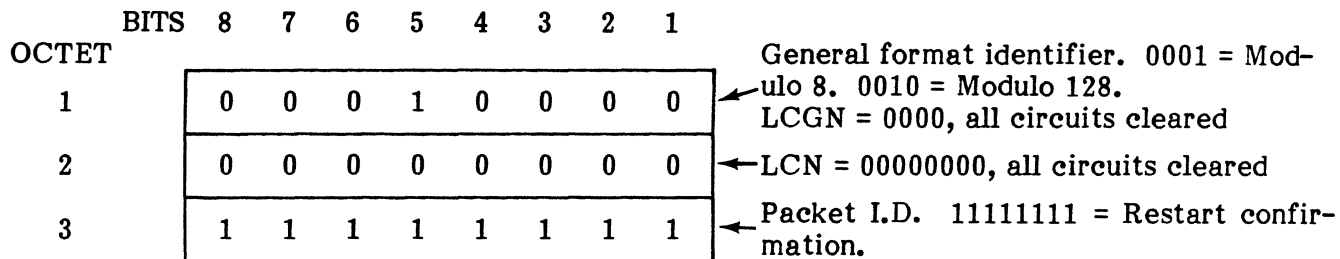
**5.30 RST?.** The restart is used to simultaneously clear all switched virtual circuits and reset all permanent virtual circuits at the DTE/DCE interface. The DTE may initiate a restart request at anytime by transmitting a restart request packet. This places the logical channel interface in the DTE restart request state. DCE indicates a restart by transmitting a restart indication packet. This places the logical channel interface in the DCE restart indication state, where DCE will ignore data, interrupt, call set-up and clearing, flow control, and reset packets. The restart request or indication packet consists of four octets as shown in Figure 8-18.



<u>RESTARTING CAUSE</u>	<u>HEX</u>	<u>BITS</u>	8	7	6	5	4	3	2	1
Local Procedure Error	= 01	=	0	0	0	0	0	0	0	1
Network Congestion	= 03	=	0	0	0	0	0	0	1	1

**Fig. 8-18 Restart Request Packet**

**5.31 RST!** This packet is used by DCE to confirm the DTE restart request. Confirmation places the logical channels used for switched virtual circuits in the Ready State and those used for permanent virtual circuits in the Flow Control Ready State. When used by DTE, the restart confirmation packet places the logical channels for switched virtual circuits in the Ready State and those used for permanent virtual circuits in the Flow Control Ready State. The restart confirmation packet consists of three octets as shown in Figure 8-19.



**Fig. 8-19 Restart Confirmation Packet**

**6. TRANSMISSION PROCEDURES**

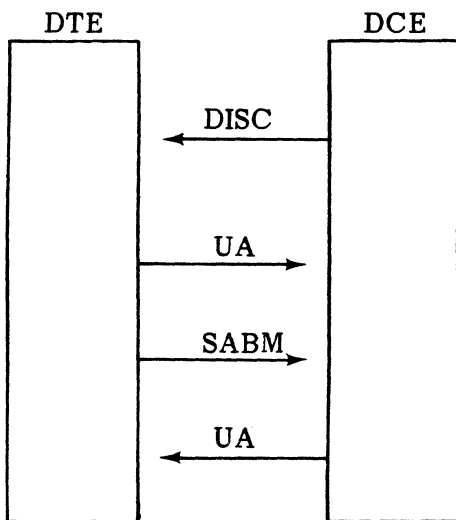
**6.01** For the purpose of this discussion, the characteristics of the DTE and DCE are described as interpreted by the ENCORE.

**LINK DOWN STATE**

**6.02** In the Link Down State, DCE continually transmits DISC commands at intervals established by the value of an internal DCE timer, T1. In this state, the DISC poll bit is set to 1 (reference Figure 8-5). This continued transmission of DISC commands is an indication to DTE that DCE is prepared to bring the link up. The Link Down State is the same regardless of whether balanced (LAPB) or unbalanced (LAP) procedures are used.

**LINK SET-UP (LAPB)**

**6.03** As shown in Figure 8-20, balanced link set-up is accomplished with fewer frame transmissions than unbalanced set-up. Prior to set-up, as mentioned in the preceding paragraph, DCE continually transmits DISC frames until DTE responds with an unnumbered acknowledge frame, UA. Upon receipt of the UA, DCE stops transmitting DISC frames and starts an internal timer, T3. DCE now expects the SABM frame to follow the UA within the T3 time. If T3 runs out before the SABM is received, it returns to the Link Down State. However, if the SABM is received prior to T3 time out, DCE transmits the UA bringing the link up and entering the Information Transfer State. It is important to note that DCE can only respond to the SABM during T3. Upon completion of the Link Set-Up Procedure, all counters and timers are initialized to zero.



In the link down state, DCE transmits DISC commands at "T1" intervals.

DTE indicates a request for link set-up by transmitting UA followed by SABM.

DCE responds with a UA and the link is up.

**TYPICAL PRINTOUT OF ABOVE FRAME TRANSMISSIONS**

FLAGS	A	TYPE	NR	NS	PF	FCS	FLAGS	A	TYPE	NR	NS	PF	FCS
							6663	A	DISC			1	1E
							7196	A	DISC			1	1E
							7196	A	DISC			1	1E
							7196	A	DISC			1	1E
							7196	A	DISC			1	1E
							7196	A	DISC			1	1E
OVER	A	UA			1	3d							
0002	B	SABM			1	k_							
							0028	B	UA			1	W

**Fig. 8-20 LAPB Set-Up Diagram and Typical Display**

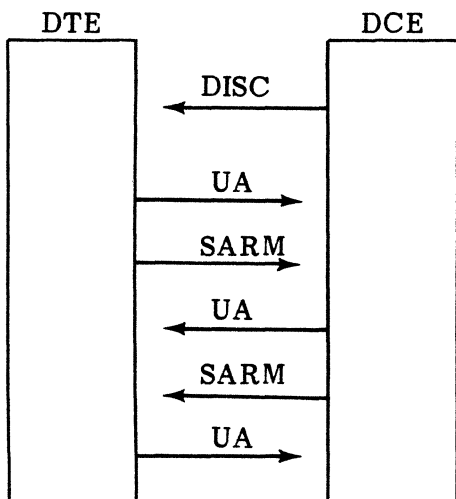


**LINK CLEAR-DOWN (LAPB)**

**6.04** While DCE is in the Information Transfer State, DTE may clear-down the link by transmitting a DISC command. Upon receipt of this command, DCE responds by retransmitting the UA frame and starting timer T3. If DTE does not transmit the SABM before T3 runs out, DCE transmits the DISC and enters the Link Down State.

**LINK SET-UP (LAP)**

**6.05** The unbalanced method of setting up the link is similar to the balanced method except that both sides of the link (DTE and DCE) must be set-up separately, i.e., both sides must transmit and acknowledge the SARM command. As shown in Figure 8-21, DCE continually transmits DISC commands, while in the Link Down State, until DTE responds with the UA, followed by the SARM command. DCE then acknowledges the DTE SARM with the UA, sets timer T3, and transmits its own SARM. Finally DTE must acknowledge the SARM by responding with the UA before T3 runs out. If T3 runs out, DCE will retransmit its SARM N2 times before returning to the Link Down State. N2 is another internal DCE timer set for the number of SARM retransmission. If the final UA is received within the allotted time, the link is up and all timers and counters are initialized to zero.



In the link down state, DCE transmits DISC commands at "T1" intervals.

DTE indicates a request for link set-up by transmitting a UA response followed by a SARM command.

DCE responds with a UA followed by a SARM command.

DTE responds with a UA and the link is up.

**TYPICAL PRINTOUT OF ABOVE FRAME TRANSMISSION**

FLAGS	A	TYPE	NR	NS	PF	FCS	FLAGS	A	TYPE	NR	NS	PF	FCS
							7645	A	DISC			1	1E
							7195	A	DISC			1	1E
							7197	A	DISC			1	1E
							7197	A	DISC			1	1E
							7196	A	DISC			1	1E
							7196	A	DISC			1	1E
							7195	A	DISC			1	1E
OVER	A	UA			1	3d							
0002	B	SARM			1	i~							
							0028	B	UA			1	W
							0002	A	SARM			1	YM
0028	A	UA			1	3d							

**Fig. 8-21 LAP Set-Up Diagram and Typical Display**

### LINK CLEAR-DOWN (LAP)

**6.06** As described for the balanced mode, Link Clear-Down is initiated by DTE while DCE is in the Information Transfer State. DTE transmits the DISC command and DCE acknowledges by transmitting the UA. A UA response from DTE at this point is optional in that DCE will enter the Link Down State with or without a response.

### INFORMATION TRANSFER (LAP and LAPB)

**6.07** During the Information Transfer State, the link employs a frame sequence number scheme used to acknowledge the receipt of an information frame, and in the case of a reject, to indicate which frames must be retransmitted. The scheme uses a send and receive frame sequence number, N(S) and N(R) respectively, and a send a receive variable, V(S) and V(R) respectively. N(S) and N(R) are included in the information frame. The variables are used by DTE and DCE to control N(S) and N(R).

**6.08** When DCE or DTE transmits an information frame, the frame send sequence number, N(S), is set equal to its send variable, V(S), and the receive sequence number, N(R), is set equal to its receive variable, V(R). After the frame is transmitted, V(S) is incremented by 1. If V(S) is now greater than the number of outstanding frames allowed (window size), no further information frames will be transmitted until one or more outstanding frames are acknowledged or until the REJ frame requests retransmission of the invalid frame. Information frames may be acknowledged by the receipt of an information frame (INFO), a receiver ready frame (RR), a receiver not ready frame (RNR), or a reject frame (REJ). When a valid acknowledgement is received, V(R) is incremented by 1. In the next frame transmitted, N(R) is set to the value of V(R). A valid receive, N(R), must be greater than the last received N(R) and equal to or less than the current V(S). If the N(R) is outside this range, it will cause a reset.

**6.09** When a received frame is invalid due to an incorrect FCS, it is discarded. If it is invalid because the received N(S) is not equal to V(R), the information content of the frame is discarded and a REJ response is transmitted with the N(R) set one higher than the last correctly received N(S). The information content of all subsequently received frames is then discarded until the expected frame is received.

#### A. Receiving Acknowledgement

**6.10** When correctly receiving an information or supervisory frame, (RR, RNR, or REJ), the N(R) contained in this frame is considered an acknowledgement of all the frames it has transmitted with a N(S) up to the received N(R) minus one.

#### B. Receiving Reject

**6.11** When receiving a REJ, V(S) is set to the N(R) received in the REJ control field. The corresponding INFO frame and subsequently transmitted frames are then retransmitted as soon as they are available.

#### C. Reset (LAPB)

**6.12** Reset of the balanced link is accomplished during the Information Transfer State when DTE transmits the SABM command. DCE responds to the SABM by transmitting a UA at the first opportunity and resetting its send and receive state variable to zero. Under certain reject conditions, DCE may initiate the reset by transmitting a command reject response, CMDR.

**D. Reset (LAP)**

**6.13** Reset of the unbalanced link is also accomplished during the Information Transfer State. DTE may initiate the reset by transmitting the SARM command. DCE then transmits the UA response at the earliest convenience and sets its V(R) to zero. When DCE initiates a reset, it transmits the SARM and starts timer T1. DTE must then respond with a UA before T1 runs out. If the response is in time, DCE stops T1, sets its send state variable to zero, and the link is reset. However, if T1 runs out, DCE restarts T1 and retransmits the SARM N2 times. If there is no response after N2 retransmission, DCE transmits a DISC command and enters the Link Down State.

7. X25/X75 OPERATING SYSTEM

7.01 This disc based package employs a powerful menu driven operating system which allows the user to monitor, capture, and display information in a packet-switched environment. It allows the user to define new packets or redefine existing packets to comply with network specifications. In the interactive modes, the user may write, store, and execute scenarios that allow the ENCORE to actively simulate network components (DCE or DTE). The operating system requires 126K bytes of disc storage, leaving 112K bytes for storing captured data.

7.02 Included in the following paragraphs is a detailed explanation of each item in the X25/X75 MENU. This includes a description of monitor commands, display commands, and typical operating procedures where appropriate. The Initial Operating Procedure includes the instructions required to power-up the ENCORE using the X25/X75 Interactive Monitor Applications Software.

7.03 Following the Initial Operating Procedure, each item of the X25/X75 MENU is discussed in order of appearance, which is the logical order of operation. For convenience, the submenus repeat some of the selections offered in the main menu and in other submenus.

7.04 An important feature of this applications package is the ability to define individual packets. For ease of operation, the packets described in the X25 Standard are predefined as the power-up default configurations. Should the user need to reconfigure these packets or define new packets not covered in the X25 Standard, he has that option, as described later in this chapter. However, the user must possess a thorough knowledge of the network's protocol before any attempt to alter the packet specification is made.

8. INITIAL OPERATION (Power-Up)

8.01 This procedure is designed to take the user through the power-up sequence using the X25/X75 Operating System supplied on diskettes numbered 805-00017 and 805-00018. It includes step-by-step instructions that will leave the user in a position to select any one of the X25/X75 operating modes. Also included with the procedure is a simple flow chart, Figure 8-22, illustrating the steps given in the procedure.

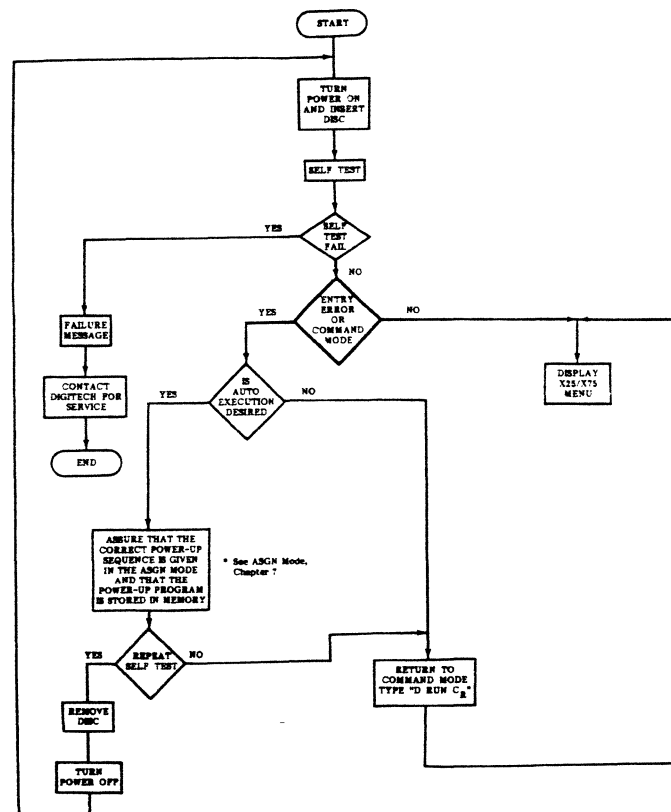


Fig. 8-22 X25/X75 Power-Up Flowchart

### INITIAL POWER-UP OPERATING PROCEDURE

STEP	PROCEDURE
1.	Connect the ENCORE to the appropriate ac power source.
2.	Connect the ENCORE to the circuit under test.
3.	Set the ac power switch to the ON position and note that the ENCORE displays "SELF TEST IN PROGRESS".
4.	Insert the disc, non-write protected, tab toward the left (see Figure 8-23).

**Note:** Use disc 805-00017, PART I for bit oriented operation (HDLC/SDLC). Use disc 805-00018, PART II for character oriented operation (BISYNC).



**Fig. 8-23 Diskette Insertion**

5. When the self-test is successfully completed, the ENCORE will normally display the X25/X75 MENU as shown in Figures 8-24 or 8-25. If the display reads "ENTRY ERROR", type D RUN . This executes the disc stored program RUN which automatically allocates memory and eliminates the error message resulting from incorrect memory allocation in the power-up ASGN Mode. This sequence is illustrated in Figure 8-22.

## INITIAL POWER-UP OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
------	-----------

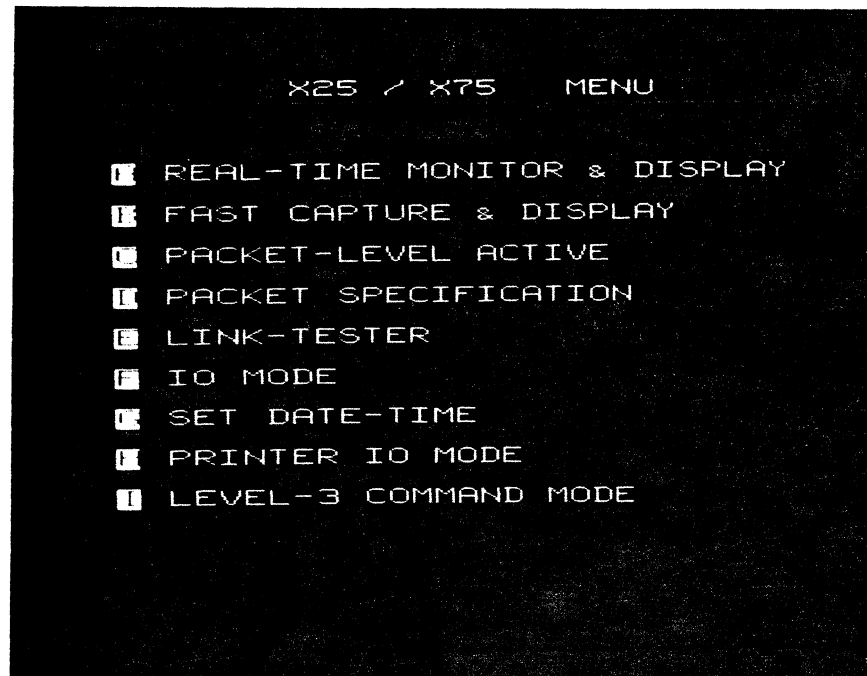


Fig. 8-24 X25/X75 Menu (HDLC/SDLC)

**Note:** Item ☐ is the ENCORE Level-3 Command Mode, not X25 Level-3.

INITIAL POWER-UP OPERATING PROCEDURE (Cont'd)

---

STEP	PROCEDURE
------	-----------

---

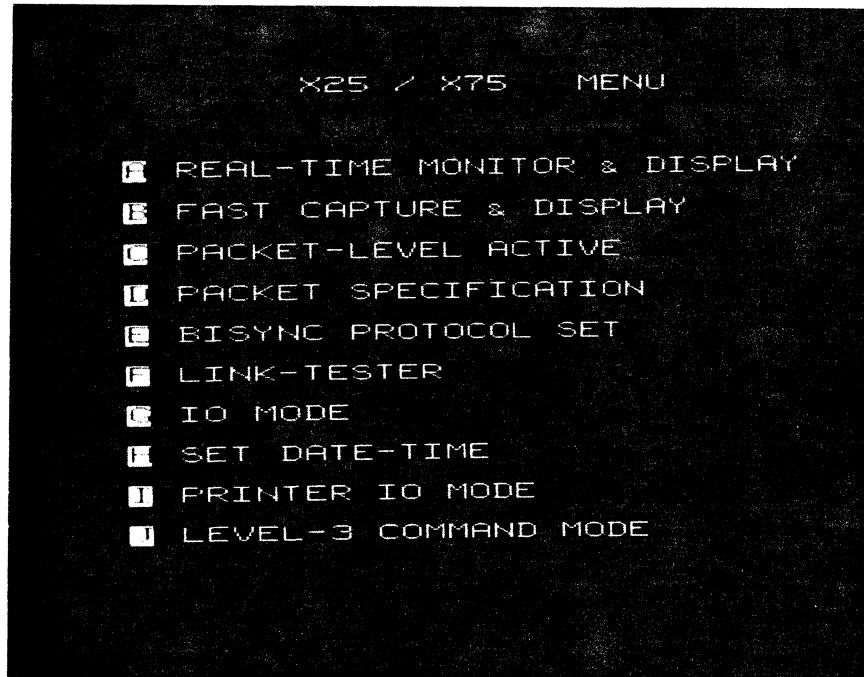


Fig. 8-25 X25/X75 Menu (BISYNC)

**Note:** Item [ ] is the ENCORE Level-3 Command, not X25 Level-3.

6. From the X25/X75 MENU, select the desired mode of operation.

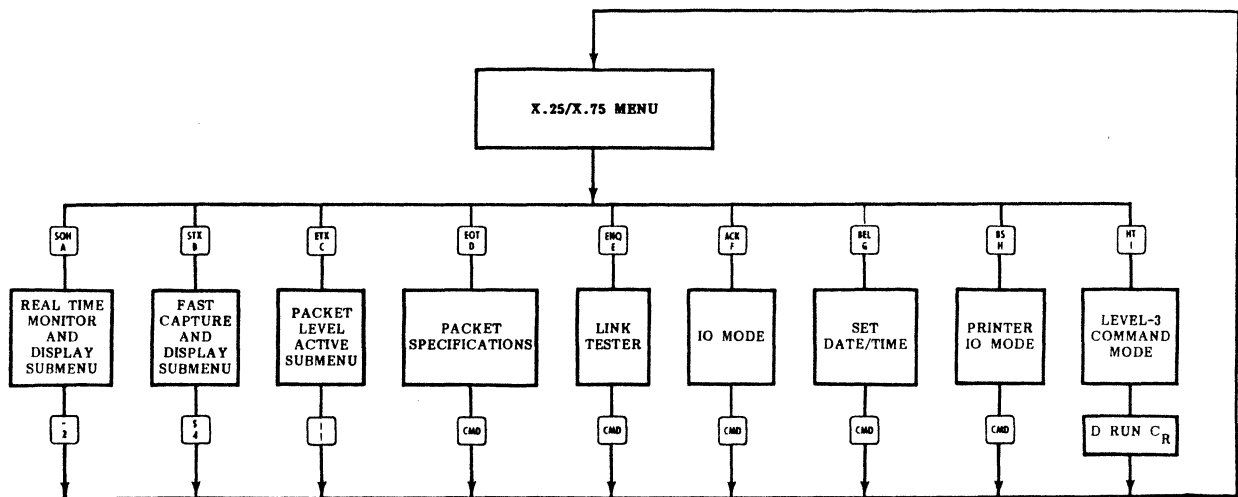
**Note:** Each item in the menu is detailed in the following paragraphs.

This completes the Initial Power-Up Procedure

---

## 9. X25/X75 MENU

**9.01** The X25/X75 MENU is designed to provide the user with the means to monitor data real time, actively simulate DTE or DCE, and log data into memory (2K solid state buffer and/or 112K disc) for later analysis. There are two items in the menu designed to monitor and log data. The first item, **SOM**, REAL TIME MONITOR AND DISPLAY, allows the user to monitor and capture data at speeds up to 19.2 kbps. It is the slower of the two monitors because incoming data is decoded for display in a split screen format as described later in this chapter. The second monitor, **SIX**, FAST CAPTURE AND DISPLAY, only displays a single line of data or a count of the frames received and is therefore capable of operating at speeds up to 30 kbps. The PACKET-LEVEL ACTIVE selection, **ETX**, allows the user to write scenarios which control the ENCORE in the DTE or DCE simulate modes and in the Scenario Driven Monitor Mode. It is important to note that first time execution of these monitors or scenario driven modes requires prior entry of IO parameters and the date time group. The IO parameters establish operating speed, clock source, language, etc., while the date time group permits use of the date time stamp immediately following captured data. These and each of the other items in the menu are detailed in the following paragraphs. In addition, keystroke flowcharts are included to supplement the text and further clarify menu selection. Figure 8-26, for example, shows each keystroke required to select an item from the X25/X75 MENU (HDLC/SDLC) and each keystroke required to return to the menu from the corresponding submenus. Figure 8-27 shows the addition of the BISYNC PROTOCOL SET supplied on the PART II disc for character oriented operation.



**Fig. 8-26 X25/X75 Menu Keystrokes (HDLC/SDLC)**



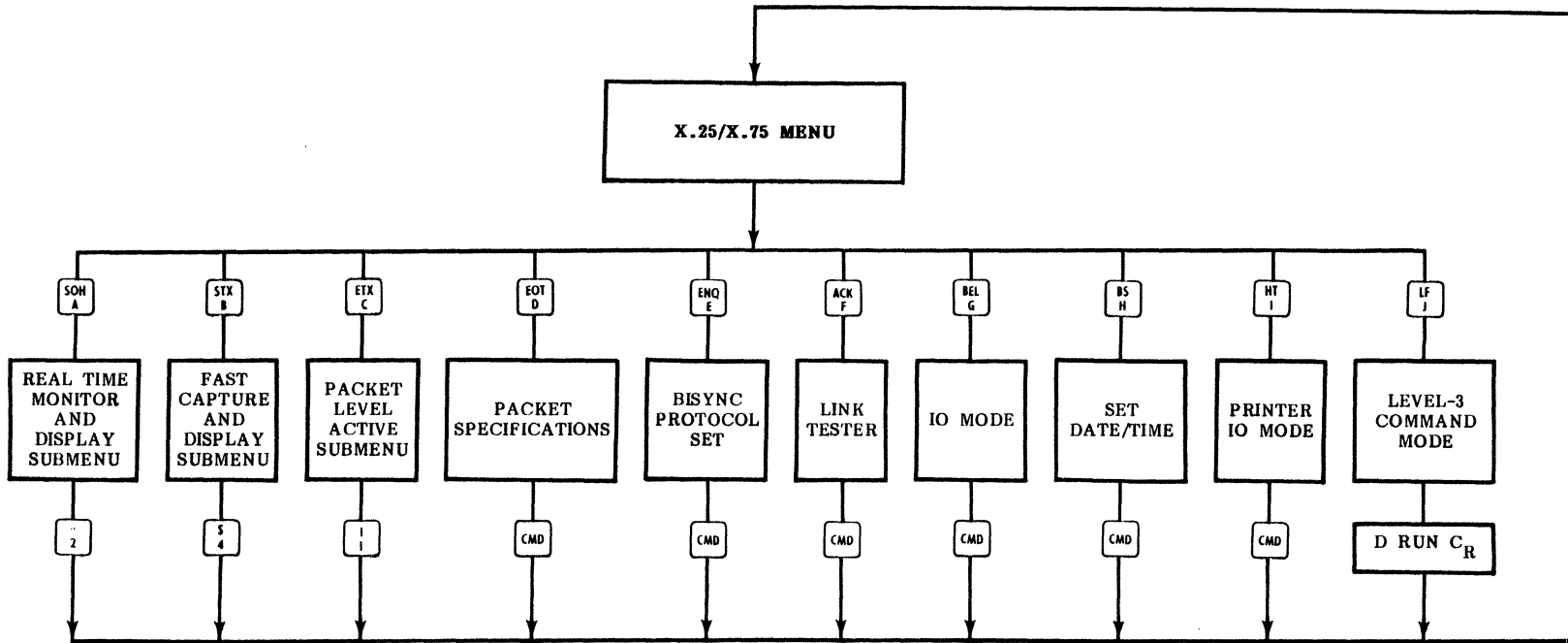


Fig. 8-27 X25/X75 Menu Keystrokes (BISYNC)

10. REAL TIME MONITOR AND DISPLAY, SOH  
A

10.01 The Real Time Monitor and Display Submenu, shown in Figure 8-28, allows the user to: 1 set monitor parameters and monitor data in real time, 2 return to the X25 MENU, 3 set IO parameters, 4 set external IO parameters, 5 display previously captured data, and 6 erase the capture buffer if buffer data is on disc.

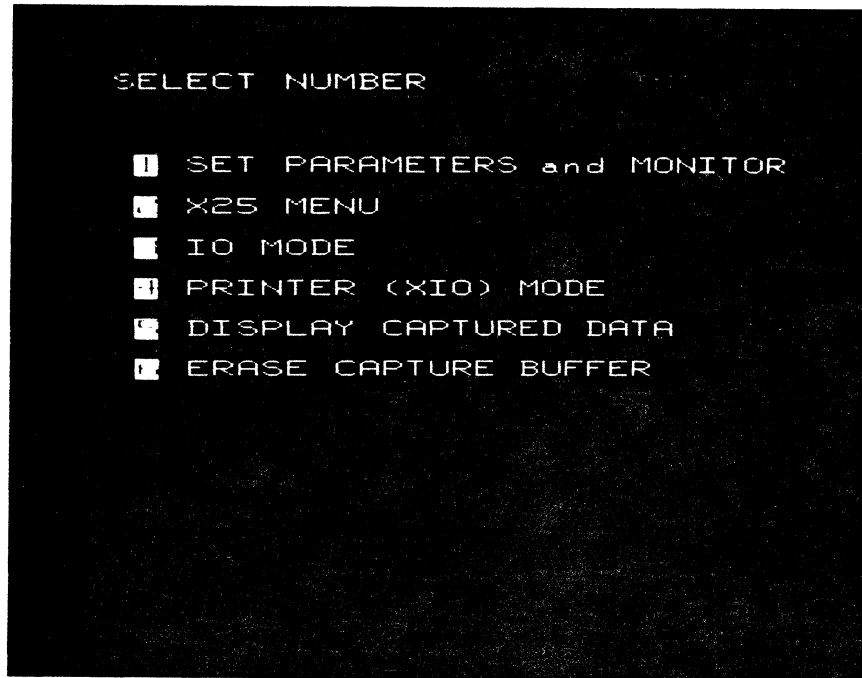


Fig. 8-28 Real Time Monitor and Display Submenu

10.02 Figure 8-29 is a keystroke flowchart showing how each item in the Real Time Monitor Submenu is selected and how the user returns to the submenu after the selection has been made.

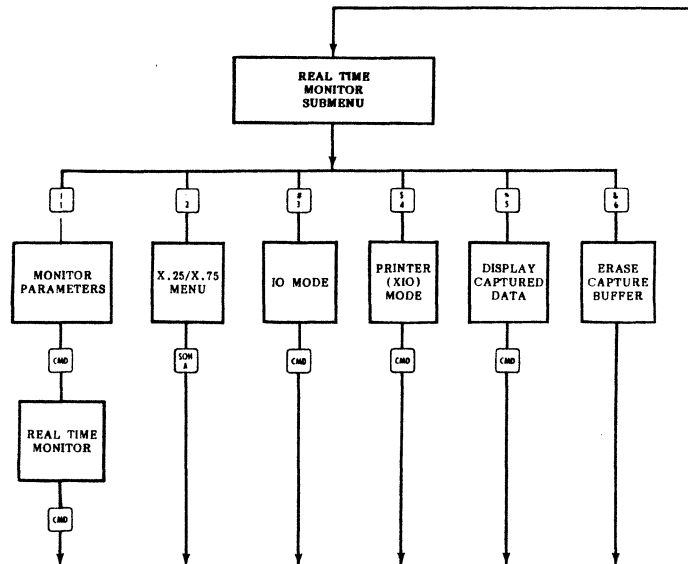
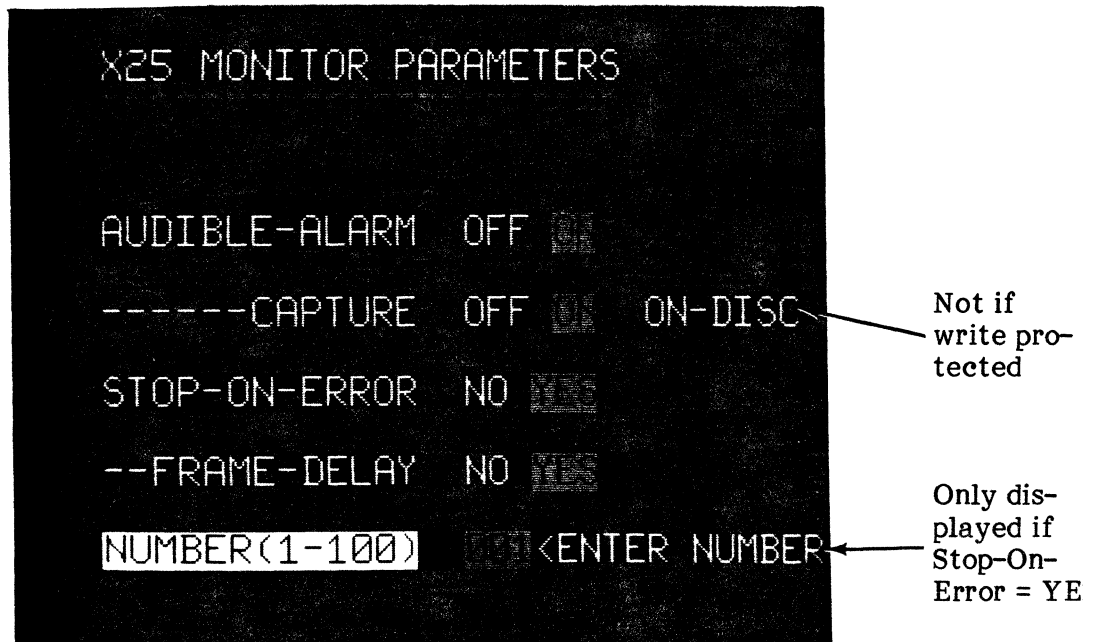


Fig. 8-29 Real Time Monitor Flowchart

**MONITOR PARAMETERS,**

**10.03** This item is selected from the Real Time Monitor and Display Submenu by striking the  key. The Monitor Parameters, as shown in Figure 8-30, instruct the ENCORE to audibly alert the user on error, to capture data, to freeze the display on error, and/or to display from 1 to 100 frames following that error. The first four parameters are selected using the , , , and  keys to position the reverse video and half-intensity field behind the desired parameters. The number of frames to be included in the frame delay are entered by typing the desired number. All entries are terminated and the monitor executed by striking the  key. A detailed explanation of each of these parameters follows in paragraphs 10.04 through 10.07.



**Fig. 8-30 Monitor Parameter Prompts**

**10.04 Audible Alarm.** The audible alarm is a 1 kHz tone which may be set, by the user, to sound whenever one of the errors shown in Table 8-3 is diagnosed by the ENCORE. The error mnemonic appears in the undefined position of the split screen display following the octet count.

TABLE 8-3  
DIAGNOSED ERRORS

DISPLAYED MNEMONIC	NAME	DEFINITION
FCS	<u>F</u> rame <u>C</u> heck <u>S</u> equence	Incorrect frame check sequence received.
* TMO	<u>T</u> oo <u>M</u> any <u>O</u> ctets	The received packet contains more than the specified number of octets.
* TFO	<u>T</u> oo <u>F</u> ew <u>O</u> ctets	The received packet contains less than the specified number of octets.
IC	<u>I</u> ncorrect <u>C</u> ontrol Field (frame level)	The received frame control field is incorrect.
* IP	<u>I</u> ncorrect <u>P</u> acket <u>I</u> den- <u>t</u> ification (packet level)	The received packet ID is incorrect.
NSS	Frame level N(S) sequence error	The received frame contains the wrong send sequence number.

**NOTE:** The asterisk (\*) indicates an error condition as defined during the Packet Specification Mode (item  on the X25/X75 MENU).

**10.05 Capture.** If the user chooses to capture data, as well as display it, he may input that data to the capture buffer and continually overwrite the oldest data. He may also input data to the buffer and transfer it to the disc in 1K blocks. In either case, this data can be recalled during the Display Mode for further evaluation. If captured data is to be displayed in the Real Time Monitor Mode, it must have been captured in that mode. If captured in the Capture and Display Mode, the data will be unintelligible.

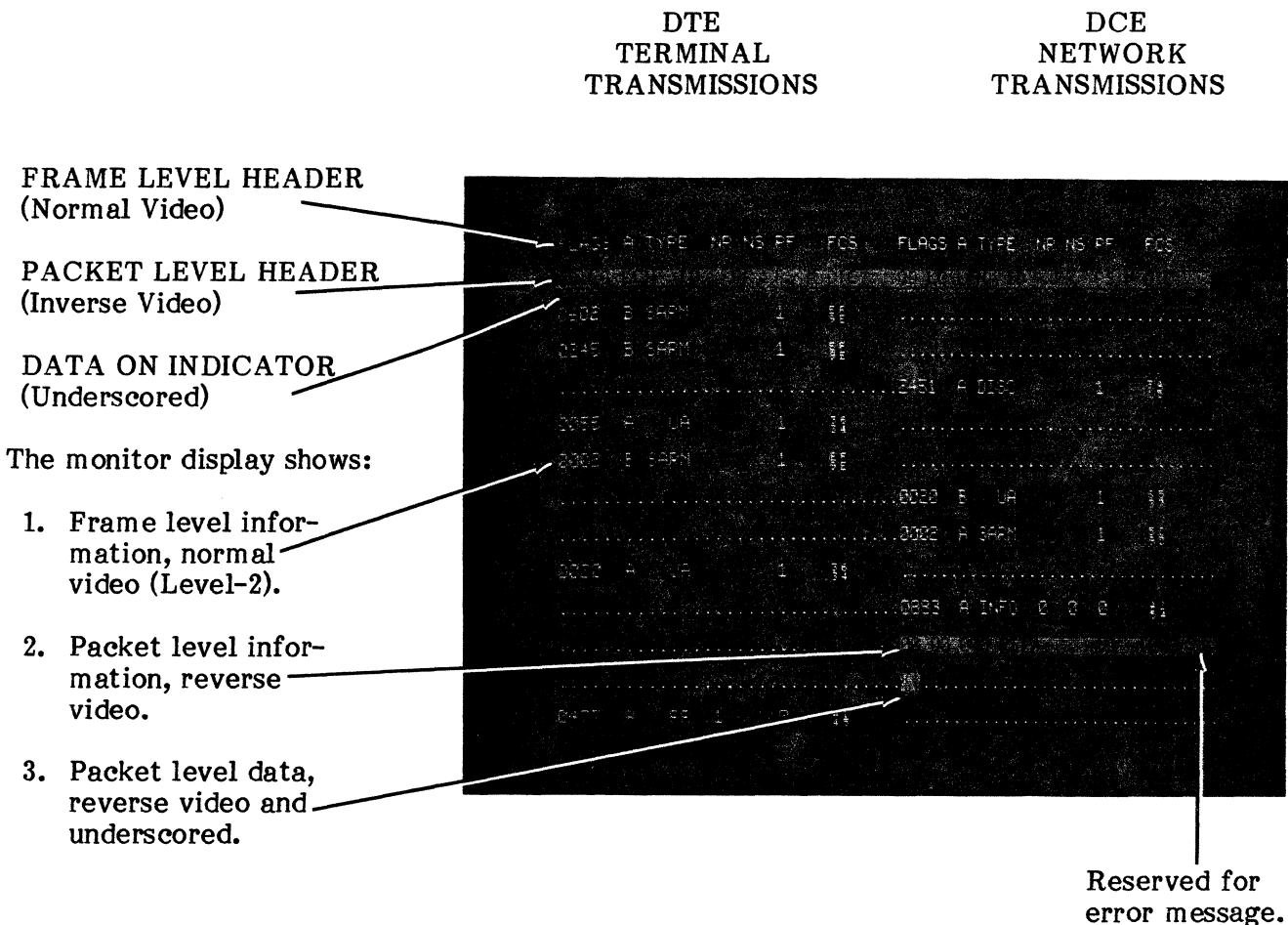
**10.06 Stop-on-Error.** During the display of on-line data, the user may choose to freeze the display if an error is encountered. The error conditions are shown in Table 8-3 and are diagnosed, in some cases, as a result of incompatibility between established packet specifications and the incoming packet. Errors at the frame level are diagnosed without user intervention.

**10.07 Frame Delay.** The Frame Delay allows the user to continue monitoring and/or recording a specified number of frames following an error. This choice can only be made if the response to "STOP-ON-ERROR" is "YES". The user may also enter the number of frames (1 to 100) to follow an error before the monitor is stopped. This can be useful when analyzing the system's response to an error.

**REAL TIME MONITOR**

**10.08** The REAL TIME MONITOR is automatically executed, when the **[CMD]** key is depressed, following selection of the last monitor parameter. It will be executed using the default IO parameters unless the user establishes IO parameters prior to selecting item **[SON A]** from the X25/X75 MENU. A complete explanation of the IO parameters including the default conditions is given in paragraph 10.31.

**10.09** When executed, the REAL TIME MONITOR allows the ENCORE to display send and receive data in the split screen format as shown in Figure 8-31. Depending upon the monitor parameters, it also allows the ENCORE to transfer data to the solid state buffer or to the disc for later recall and analysis.



**Fig. 8-31 Typical Display Mode Format**

**10.10** While data is displayed in the split screen format, the user may execute up to six different display commands to further enhance the presentation of data. These commands are listed in Table 8-4. Figures 8-32 through 8-34 are examples of the display produced when the **[EOT D]** key is depressed (disable data), when the **[EOT D]** and **[PLE P]** keys are depressed (disable data and packet), and when the **[EOT D]** and **[ACK F]** keys are depressed (disable data and frame).

**TABLE 8-4**  
**MONITOR DISPLAY COMMANDS**

COMMAND	DESCRIPTION
<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">ACK F</div> <div style="font-size: 2em; margin: 0 5px;">/</div> <div style="border: 1px solid black; padding: 2px;">ACK F</div> </div>	Enable or disable display of frame information.
<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">DLE P</div> <div style="font-size: 2em; margin: 0 5px;">/</div> <div style="border: 1px solid black; padding: 2px;">DLE P</div> </div>	Enable or disable display of packet information.
<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">EOT D</div> <div style="font-size: 2em; margin: 0 5px;">/</div> <div style="border: 1px solid black; padding: 2px;">EOT D</div> </div>	Enable or disable display of packet level user data.
<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">CAN X</div> <div style="font-size: 2em; margin: 0 5px;">/</div> <div style="border: 1px solid black; padding: 2px;">CAN X</div> </div>	Freeze or continue display.
<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">HEX</div> <div style="font-size: 2em; margin: 0 5px;">/</div> <div style="border: 1px solid black; padding: 2px;">HEX</div> </div>	Enable or disable HEX display of packet level user data.
<div style="border: 1px solid black; padding: 2px; width: 40px; text-align: center;">DC3 S</div>	Clear screen and change format.
<div style="border: 1px solid black; padding: 2px; width: 40px; text-align: center;">CMD</div>	Exit Monitor Mode and return to submenu.

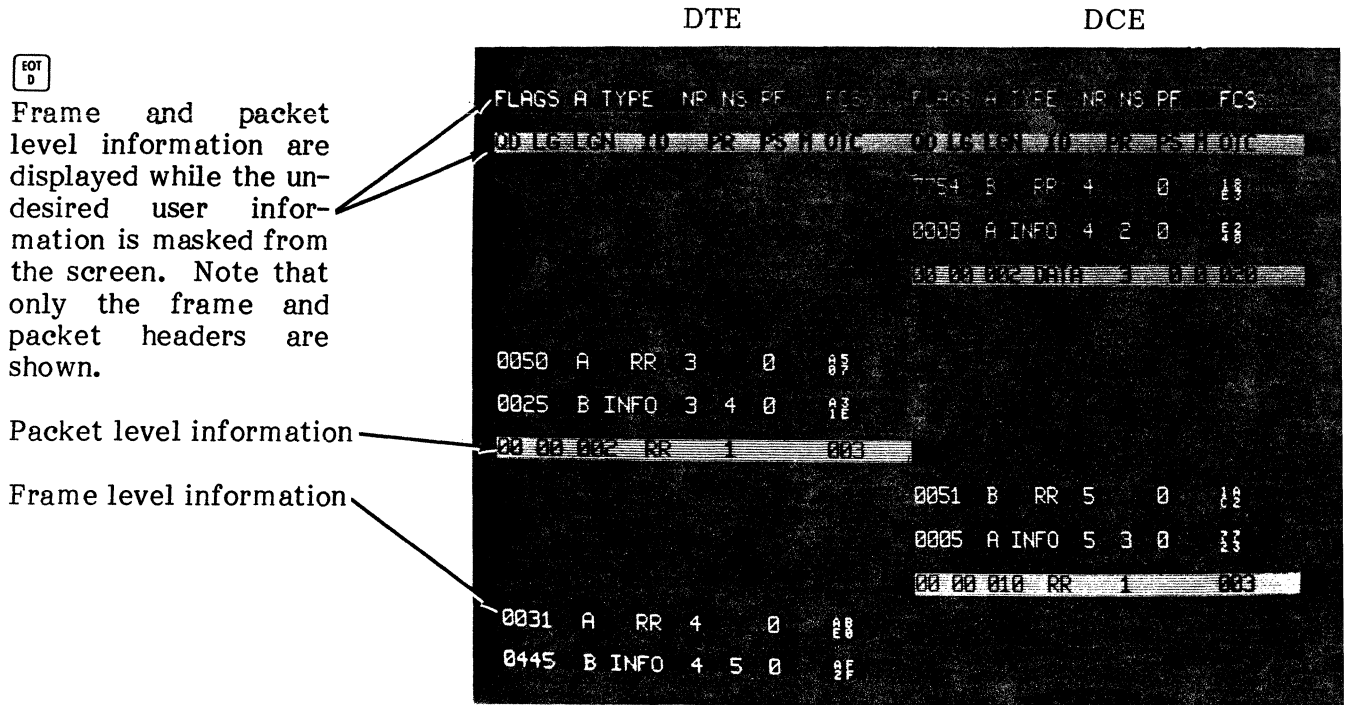


Fig. 8-32 Typical Monitor Display Frame and Packet

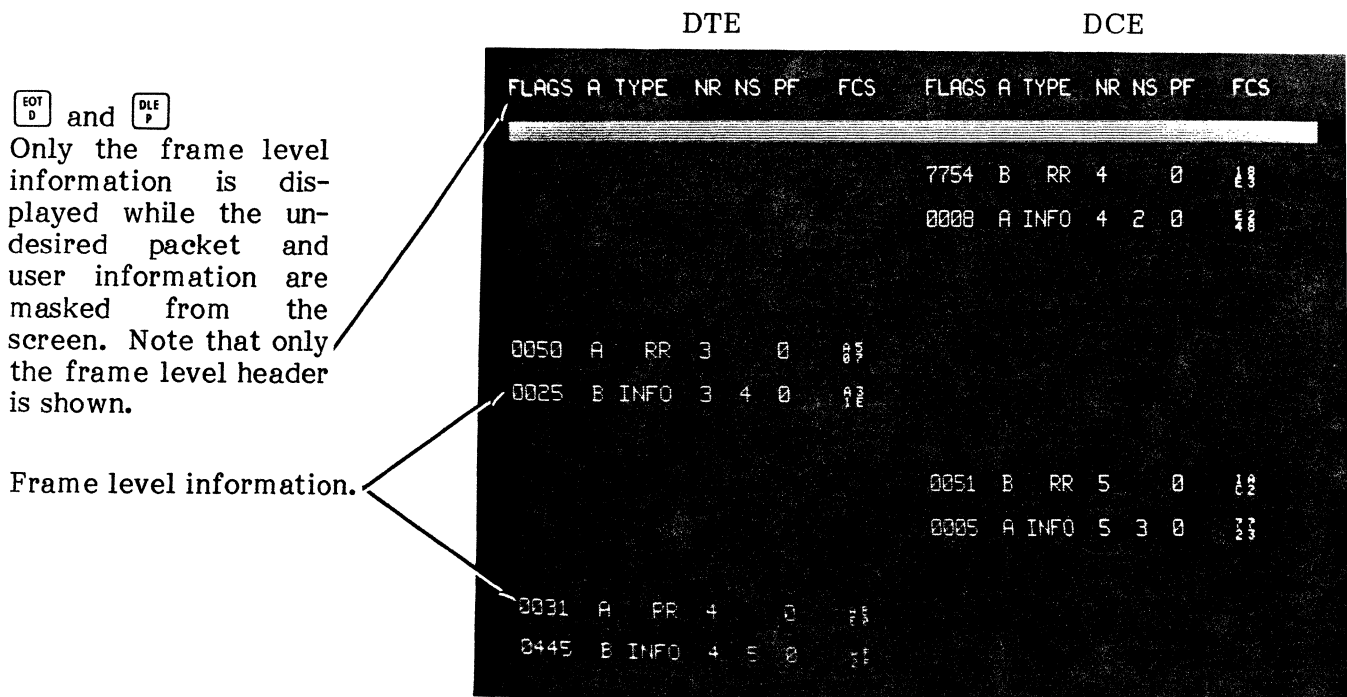


Fig. 8-33 Typical Monitor Display Frame Only

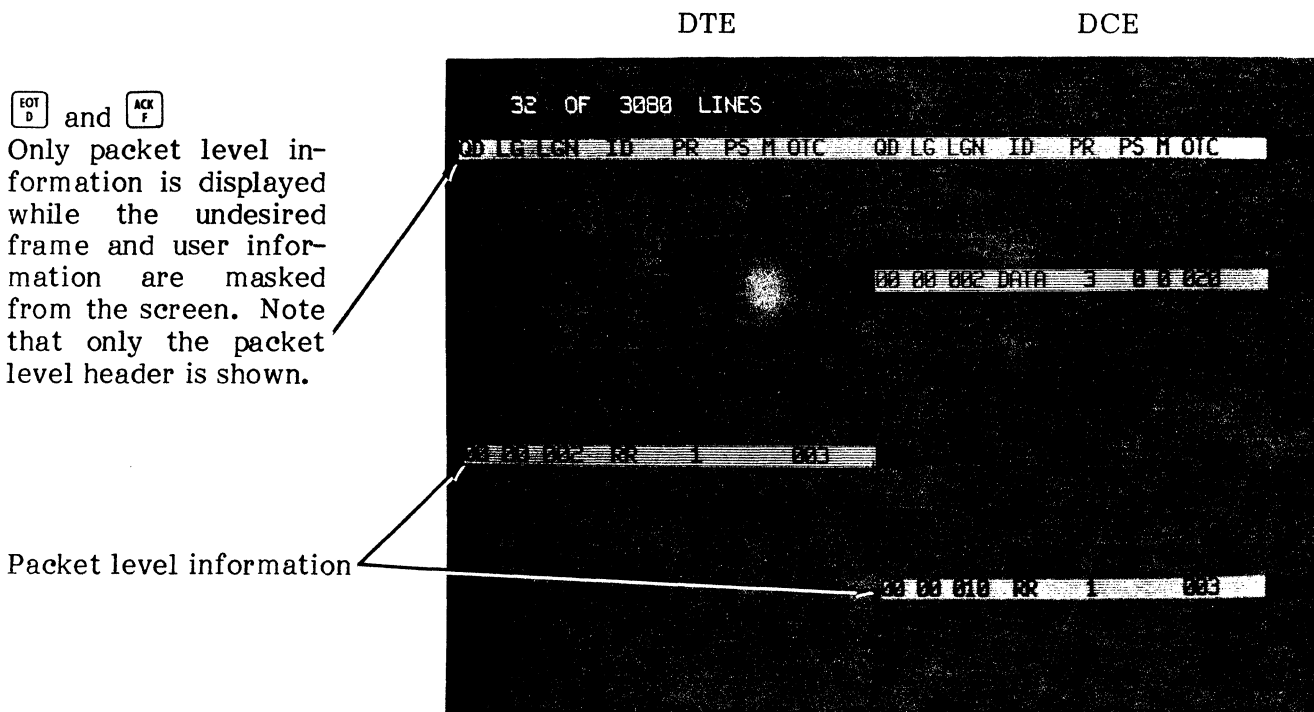


Fig. 8-34 Typical Monitor Display Packet Only

**SPLIT SCREEN FORMAT**

**10.11** The split screen format used in both the monitor and display modes places DTE and DCE information on the left and right hand sides of the screen respectively. The format also identifies the various fields within the frame and packet by supplying the header information shown in the previous illustrations. A description of the information contained in these headers is supplied in the following paragraphs. It is important to note that the same headers are used in the Real Time Monitor and for the display of data captured in both the Real Time Monitor and Fast Capture Modes.

**10.12** The split screen format shows frame level information in normal video and packet level information in reverse video. User data (packet level) is always shown in reverse video, half-intensity, and underscored. The frame level information includes the number of flags between transmissions, the address (A or B), the type of frame, N(R) and N(S) counts, poll/final bit status, and the frame check sequence. Packet level information includes the status of the Q-bit, D-bit, logical group number, logical channel number, and the octet count.

**A. Frame Level Header**

**10.13 FLAGS.** The number in this column is the number of flags that have appeared between transmissions on the same pin.

**10.14 A.** The letter in this column is the frame address. Frames containing commands transferred from the DCE to DTE or responses from DTE to DCE will contain the address A. Frames containing commands transferred from DTE to DCE or responses from DCE to DTE will contain address B.



**10.15 TYPE.** This column contains the frame identification.

- INFO = Information frame.
- RR = Receive ready, supervisory frame.
- RNR = Receive not ready, supervisory frame.
- REJ = Reject, supervisory frame.
- SARM = Set asynchronous response mode, unnumbered frame.
- SABM = Set asynchronous balance mode, unnumbered frame.
- DM = Disconnect mode, unnumbered frame.
- DISC = Disconnect, unnumbered frame.
- UA = Unnumbered acknowledge, unnumbered frame.
- CMDR = Command reject, unnumbered frame.

**10.16 NR.** This column contains the frame level receive sequence number which is the expected sequence number of the next received frame. This number is included in all information and supervisory frames.

**10.17 NS.** This column contains the frame level send sequence number which is the number of the frame being transmitted. Only information frames include the NS.

**10.18 PF.** This column contains the poll/final bit which is set to either "1" or "0". If the DCE receives a SARM, DISC, or INFO frame with the poll bit set to "1", its response includes a final bit set to "1".

**10.19 FCS.** This column contains the frame check sequence in HEX.

## **B. Packet Level Header**

**10.20 Q.** This column contains the Q-bit (data qualifier) which is set to "1" or "0" in data and interrupt packets to indicate the level of data being transmitted. The "0" indicates one level of data, and the "1" indicates two levels of data.

**10.21 D.** This column contains the D-bit which is used to indicate whether or not an end to end acknowledgement of delivery is required for data being transmitted. Acknowledgement is provided by means of the packet receive sequence number.

**10.22 LG.** This column contains a 2-digit logical channel group number (0-15) which appears in all but the restart packet.

**10.23 LCN.** This column contains a 3-digit logical channel number (0-255) which appears in all but the restart packet.

**10.24 ID.** This column contains the packet type identifier.

- CAL? = Incoming call or call request.
- CAL! = Call connected or call accepted.
- CLR? = Clear indication or clear request.

## REAL TIME MONITOR AND DISPLAY

- CLR! = Clear confirmation.
- DATA = Data packet.
- INT? = Interrupt.
- INT! = Interrupt confirmation.
- RR = Receive ready.
- RNR = Receive not ready.
- REJ = Reject.
- RSR? = Reset indication or reset request.
- RSR! = Reset confirmation.
- RST? = Restart indication or restart request.
- RST! = Restart confirmation.

**10.25 PR.** This column contains the packet receive sequence number (PR = 0-7 in Modulo 8 or 0-127 in Modulo 128). It appears in DATA, RR, RNR, and REJ packets. It must be within the window established by the last PR received and the next PS to be transmitted. If it is not, DCE resets the virtual circuit.

**10.26 PS.** This column contains the packet send sequence number (PS = 0-7 in Modulo 8 or 0-127 in Modulo 128). It must be within the window established by the last PR received and the next PS expected. If it is not, DTE resets the virtual circuit. Only data packets contain the PS.

**10.27 M.** This column contains the "more bit". "0" indicates no more data, "1" indicates that more data will follow.

**10.28 OTC.** This column contains a count of the number of octets transmitted in the packet.

**10.29 Errors.** The undefined positions following the octet count will contain an error message if an error condition occurs. The errors are defined in Table 8-3.

### X25 MENU,

**10.30** Selecting "X25 MENU" from the Real Time Monitor and Display Submenu, by striking the  key, simply instructs the ENCORE to display the X25/X75 MENU once again.

### IO MODE,

**10.31** This item is included in the Real Time Monitor and Display Submenu to permit user entry of IO parameters. Once the  key is struck, IO parameters are established in the same manner as described in Chapter 7 with some noted exceptions. Striking the  key allows the user to enter separate parameters for each of the front ends (pin 2 and pin 3). Cursor position and parameter entry are controlled using the , , , and  keys. Striking the  key will return display of the submenu. During X25 operation, the SYNC, MODE, and STOPS parameters are to be ignored by the user.

**10.32 Speed.** Any speed from 10.0 to 99 900 bits per second, to three significant digits, may be entered (five digits maximum). For speeds greater than 999 bps, key in spaces or zeros as appropriate. For speeds under 999 bps, a decimal point must follow the units digit. The three keystroke sequences that follow illustrate the use of the space and decimal point when

entering speeds:

100 baud = 1 0 0 SP

75 baud = 7 5 .

45.5 baud = 4 5 . 5

The maximum speed at which the X25/X75 Applications Package can be run depends on the mode of operation. In the interactive modes, for example, the speed of operation is controlled as a function of the window size. If information arrives too fast, the received frames are simply not acknowledged and will be retransmitted when the window size falls below the maximum of 7 frames. In the monitor modes, however, the speed of operation is affected by several conditions. These include the percent of line utilization and the type of monitor selected. As discussed in previous paragraphs, the X25/X75 Applications Package employs two different monitors which are designated A and B in the X25/X75 Menu. The B monitor is capable of operation at speeds significantly higher than the A monitor, especially if the monitor is limited to only one side of the line (DTE or DCE, not both). The operating speed of both monitors is also affected by line utilization. Under normal circumstances, line utilization ranges from 30 to 75%, and as utilization decreases, the maximum speed of operation increases from 1.5 to 3 times. The maximum operating speeds during 100% line utilization are shown below. The actual speed of operation in a real environment is many times faster and varies in proportion to the line utilization.

**X25/X75 MONITORING SPEEDS**

TYPE OF MONITOR	NO. OF PINS MONITORED	DISC STATUS	DISPLAY STATUS	SPEED	COMMENTS
A	DTE & DCE (both) info	DISC on	DISPLAY on	9.6	
A	DTE & DCE (both) info	DISC off	DISPLAY on	19.2	
B	Monitoring DTE or DCE (only one)	DISC off	DISPLAY off	80	Message size limited to 4000 characters (data, packet, and frame headers)
B	Monitoring DTE or DCE (only one)	DISC on	DISPLAY off	56	
B	DTE & DCE (both) info	DISC on	DISPLAY off	9.6	

**10.33 Sync.** Not applicable.

**10.34 NRZI.** This is an IBM encoding technique (Non-Return-to-Zero-Inverted) used in an SDLC environment when a terminal must provide its own timing. It provides a signal change any time a binary 0 is received, but not when a binary 1 is received. Strings of 0's provide transitions used for receive synchronization. Strings of 1's cause zero insertion which again provides synchronizing transitions.

**10.35 Mode.** During execution of the X25 software, the ENCORE automatically sets the transmission mode to SDLC (ENCORE's equivalent to HDLC).

**10.36 Stops.** Not applicable.

**10.37 Clock.** The choice of an internally or externally derived timing device is offered in all modes. This is especially useful during operation as a terminal in a synchronous environment when external timing is unavailable or suspected of being incorrect.

**10.38 Parity.** Six choices are available for parity selection: odd, even, none, mark, space, and ignore. The correct choice is essential for several reasons: 1) In a synchronous environment, character synchronization will not be attained without proper parity; 2) Certain program mode instructions require the recognition of received characters. If the wrong parity is selected, the characters will never be recognized; 3) When displaying logged characters in their hexadecimal equivalent, selection of the wrong parity will change the HEX character; and 4) In some cases peripheral equipment will require that the parity-bit of all characters be the same logical state.

**10.39 Lang.** In its standard configuration, the ENCORE is equipped to operate in five different languages: ASCII, EBCDIC, SELECTRIC, BCD, and EXTERNAL BCD. Other languages including IPARS, XS-3, BAUDOT, SBT, UNIVAC Fielddata, BCDIC, RMS, and RMS-2 are available as options.

**TYPICAL IO OPERATING PROCEDURE**

**10.40** The purpose of this procedure is to demonstrate, by example, the steps required for proper selection of all IO parameters. Figure 8-35 shows the power-up IO default parameters.

**TYPICAL IO OPERATING PROCEDURE**

STEP	PROCEDURE
1.	From the Real Time Monitor and Display Submenu, select the IO MODE by striking the <input type="checkbox"/> key and note a display similar to that shown in Figure 8-35.

TYPICAL IO OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
------	-----------

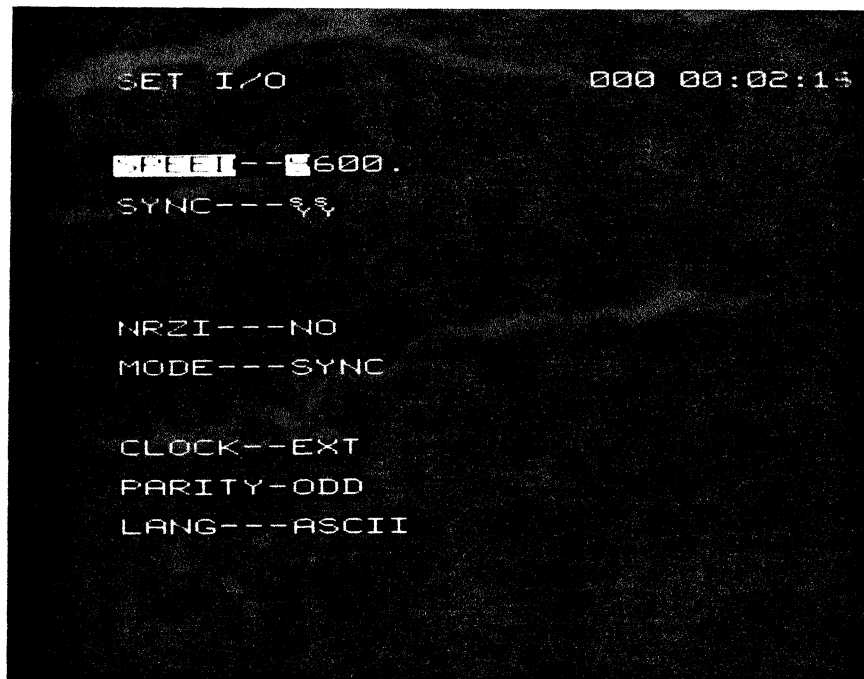


Fig. 8-35 Power-Up Default IO

2. Type the desired speed (10 to 99,900 bps) entering spaces where digits are not required. Follow the last entry with the  key.
3. For X25 operation, ignore the SYNC sequence entry. Strike the  , if required, to advance to the NRZI parameter.
4. Select the appropriate response (YES or NO) to NRZI by striking the  key. Strike the  key to advance.
5. For X25 operation, ignore the MODE entry.
 

**Note:** This step is skipped because the ENCORE automatically sets the transmission mode to SDLC during the monitor. The ENCORE SDLC parameter is equivalent to HDLC as specified in X25.
6. If the STOPS parameter is shown, ignore it.
7. Select the timing source by striking the  key until the desired source is displayed. Strike the  key to advance.

TYPICAL IO OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
8.	Select the desired parity by striking the <input type="checkbox"/> key until the desired parity is displayed. Strike the <input type="checkbox"/> key to advance.
9.	Select the desired language by striking the <input type="checkbox"/> key until the desired language is displayed.
10.	Strike the <input type="checkbox"/> key to terminate entries and note that the menu is displayed once again.

This completes the Typical IO Operating Procedure

PRINTER IO MODE,

10.41 The PRINTER IO MODE, also referred to as the XIO Mode, allows the user to establish the IO parameters associated with the rear panel XIO port. It is used in the display modes in conjunction with the  command to output the contents of the screen to some external device. Selecting item  from the X25/X75 MENU instructs the ENCORE to display the external IO prompts. For additional information, see Chapter 7.

DISPLAY CAPTURED DATA,




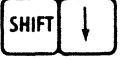









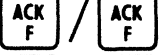
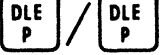
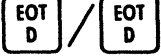


10.42 This item, "DISPLAY CAPTURED DATA", allows the user to display data captured during the Real Time Monitor. This can only be accomplished if the user indicates the desire to capture data during selection of the monitor parameters. If there is no captured data in the buffer or on disc, this item will not be displayed in the Real Time Monitor and Display Submenu.

10.43 Data captured for display in this mode includes a time stamp following the last line of data. Proper use of the stamp, although not required, is accomplished by setting the date time group listed as item  in the X25/X75 MENU. To be used effectively, it should be set prior to selecting items , , or  from this menu. Because each item in the menu is discussed in order of appearance, the procedure for entering the date time group is given in paragraph 16.02.

**10.44** Assuming that data has been captured for display in this mode, the user will find that it is displayed in the same split screen format described for the Real Time Monitor. Display of the data begins with the last page of the capture buffer including the time stamp. When there is insufficient data to completely fill the capture buffer, the ENCORE will insert spaces until the buffer is filled. This allows the ENCORE to store transfer and display data in 1K blocks. The use of additional Capture Display Commands, detailed in Table 8-5, allow the user to further enhance the presentation of data. In addition to masking unwanted data from the screen, he may scroll through the data, search to an error or string, and output the data to an external device for hard copy.

**NOTE:** To display captured data in the Real Time Monitor and Display Mode, it must have been previously captured in this mode.

**TABLE 8-5**  
**CAPTURE DISPLAY COMMANDS**

COMMAND	DESCRIPTION
	Scroll forward one line.
	Scroll backward one line.
	Scroll forward to the last page of captured data.
	Scroll backward to the first page of captured data.
	Scroll forward continuously at speeds of 0 (fastest) through 9 (slowest).
	Scroll backward continuously at speeds of 0 (fastest) through 9 (slowest).
	Search to next error.
	Enter string sequence and search to first appearance (14 character maximum). Use the  ,  , and  keys to speed search by deleting unwanted information.
	Advance to next appearance of string sequence.
	Output the contents of the screen via the XIO port.
	Enable or disable display of frame information.
	Enable or disable display of packet information.
	Enable or disable display of packet level user data.
	Enable or disable HEX display of packet level user data.
	Exit Display Mode and return to submenu.

**ERASE CAPTURE BUFFER, 6**

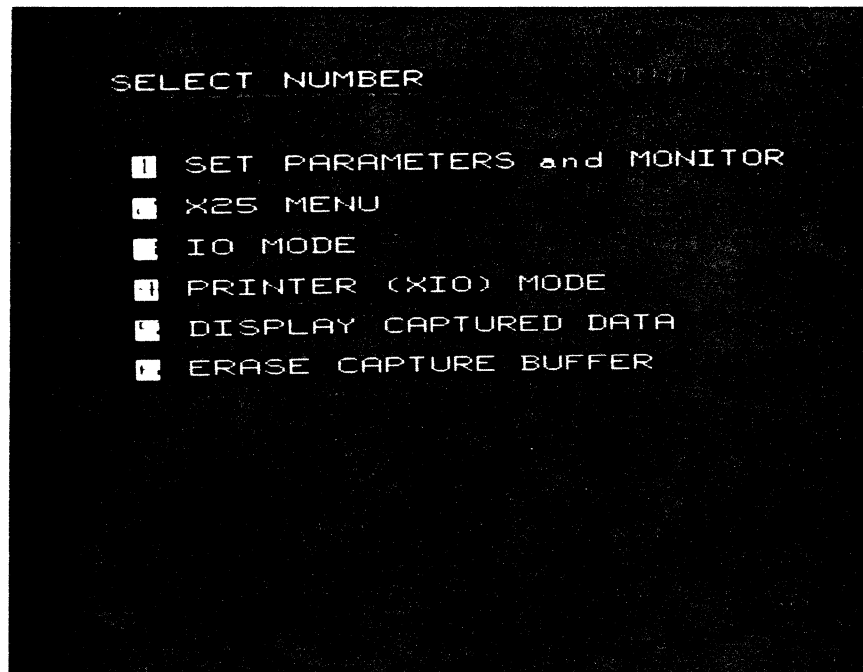
**10.45** When item 6, "ERASE CAPTURE BUFFER", is selected from the Real Time Monitor and Display Submenu, the contents of the 2K capture buffer are erased. This feature is used prior to the transfer of data from the disc to buffer to prevent accidental loss of data. If the user, while in one of the Display Modes, scrolls to the end of the buffer, data from the disc will not be transferred to the buffer until its contents are erased. To accomplish this, the user must leave the display mode ( CMD key), select item 6 from the Real Time Monitor and Display Submenu, and then return to the Display Mode. On the other hand, data cannot be erased from the buffer unless there is some captured data on the disc. Item 6 only appears in the submenu if data has been input to the buffer via the X25/X75 interface. In addition, the buffer is automatically erased whenever a monitor (Real Time or Fast) is executed.

**TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE**

**10.46** For the purpose of this procedure, it is assumed that the ENCORE is connected to the circuit under test and powered-up in accordance with the Initial Operating Procedure given in paragraph 8.01. It is also assumed that the default packet specifications are adequate for successful monitoring of the packet-switched data.

**TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE**

STEP	PROCEDURE
1.	From the X25/X75 MENU select "REAL TIME MONITOR AND DISPLAY" by striking the <span style="border: 1px solid black; padding: 0 2px;">SON</span> key and note that the ENCORE displays a submenu similiar to that shown in Figure 8-36.



**Fig. 8-36 Real Time Monitor and Display Submenu**



TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
2.	Select "MONITOR PARAMETERS" from the submenu by striking the <input type="checkbox"/> key and note that the ENCORE displays monitor parameter prompts in a manner similar to that shown in Figure 8-37.

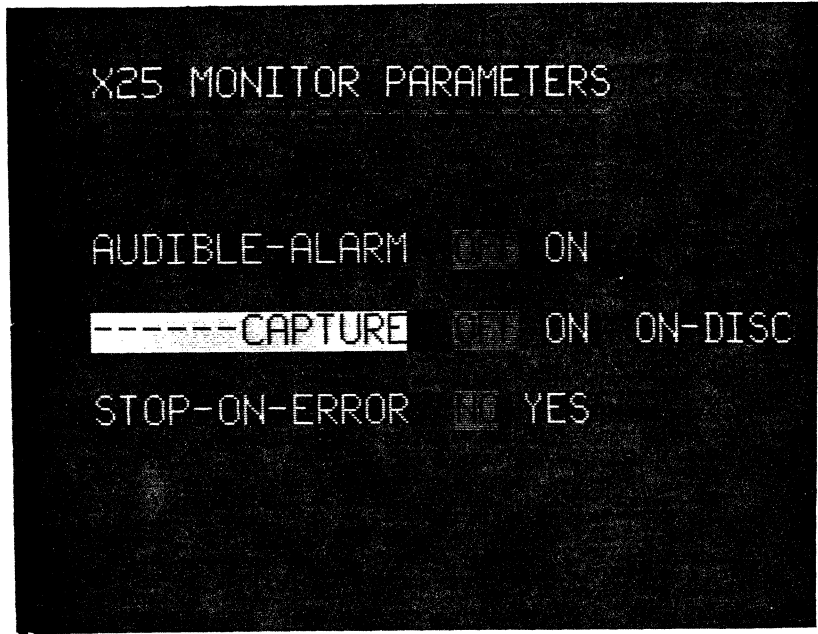


Fig. 8-37 Monitor Parameter Prompts

3. Using the , , , and  keys, enter the desired monitor parameter as described in paragraphs 10.04 through 10.07.
4. Strike the  key and note that the ENCORE displays information in the split screen format as shown in Figure 8-38.

TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
DTE TERMINAL TRANSMISSIONS	DCE NETWORK TRANSMISSIONS

Fig. 8-38 Typical Monitor Format

5. Use any of the following Monitor Display Commands to further enhance the display of information.

- ACK F /  ACK F Enable or disable display of frame information.
- DLE P /  DLE P Enable or disable display of packet information.
- EOT D /  EOT D Enable or disable display of packet level user data.
- CAN X /  CAN X Freeze or continue display.

**TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE (Cont'd)**

STEP	PROCEDURE
	<p> <input type="checkbox"/> HEX / <input type="checkbox"/> HEX Enable or disable HEX display of packet level user data.  <input type="checkbox"/> DC3 S Clear screen and change format.  <input type="checkbox"/> CMD Exit Monitor Mode and return ot submenu.                 </p>
6.	<p>To exit the Real Time Monitor Mode, simply strike the <input type="checkbox"/> CMD key and note that the ENCORE displays the Real Time Monitor and Display Submenu once again. Also note that if data was captured, the menu now includes items <input type="checkbox"/> S and <input type="checkbox"/> S to display captured data or erase the capture buffer.</p>
7.	<p>To display captured data, select item <input type="checkbox"/> S from the menu and use the following Capture Display Commands to further enhance the display of information.</p>
	<p> <input type="checkbox"/> ↑ Scroll forward one line.  <input type="checkbox"/> ↓ Scroll backward one line.  <input type="checkbox"/> SHIFT ↑ Scroll forward to the last page of captured data.  <input type="checkbox"/> SHIFT ↓ Scroll backward to the first page of captured data.  <input type="checkbox"/> → Scroll forward continuously at speeds of 0 (fastest) through 9 (slowest).  <input type="checkbox"/> ← Scroll backward continuously at speeds of 0 (fastest) through 9 (slowest).  <input type="checkbox"/> ENQ E Search to next error.  <input type="checkbox"/> DC3 S Enter string sequence and search to first appearance (14 characters max).  <input type="checkbox"/> SOH A Advance to next appearance of string sequence.  <input type="checkbox"/> SI O Output the contents of the screen via the XIO port.                 </p>
	<p> <input type="checkbox"/> ACK F / <input type="checkbox"/> ACK F Enable or disable display of frame information.  <input type="checkbox"/> DLE P / <input type="checkbox"/> DLE P Enable or disable display of packet information.  <input type="checkbox"/> EOT D / <input type="checkbox"/> EOT D Enable or disable display of packet level user data.  <input type="checkbox"/> HEX / <input type="checkbox"/> HEX Enable or disable HEX display of packet level user data.  <input type="checkbox"/> CMD Exit Display Mode and return to submenu.                 </p>
8.	<p>To exit the Display Mode, simply strike the <input type="checkbox"/> CMD key and note that the ENCORE again displays the Real Time Monitor and Display Submenu.</p>

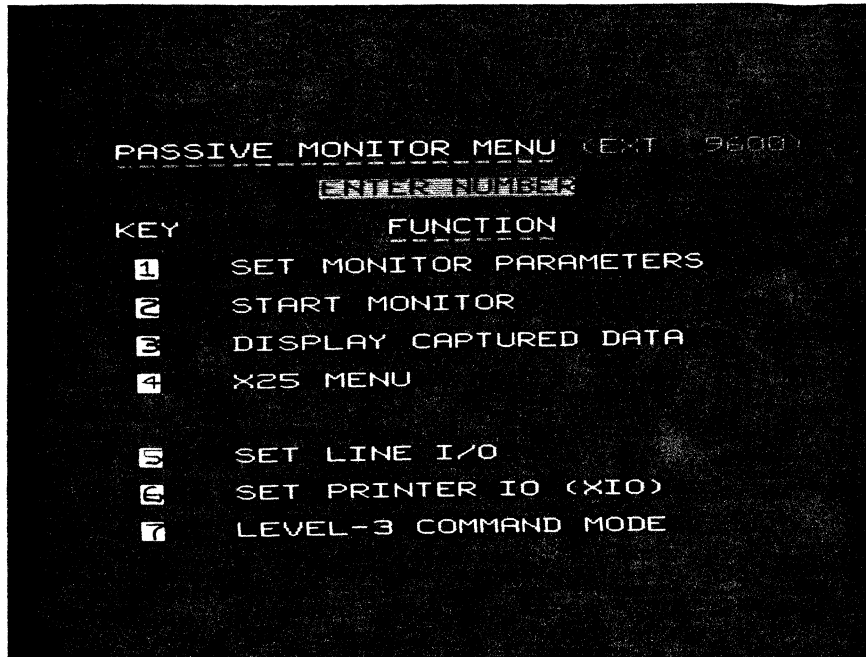
TYPICAL REAL TIME MONITOR AND DISPLAY OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
9.	To erase the capture buffer, simply strike the <input type="checkbox"/> key. If there is no captured data on the disc, the buffer will not be cleared until item <input type="checkbox"/> is selected from the submenu. If the buffer is cleared, the submenu will be updated, if not, item <input type="checkbox"/> will remain in the display.

This completes the Typical Monitor and Display Operating Procedure

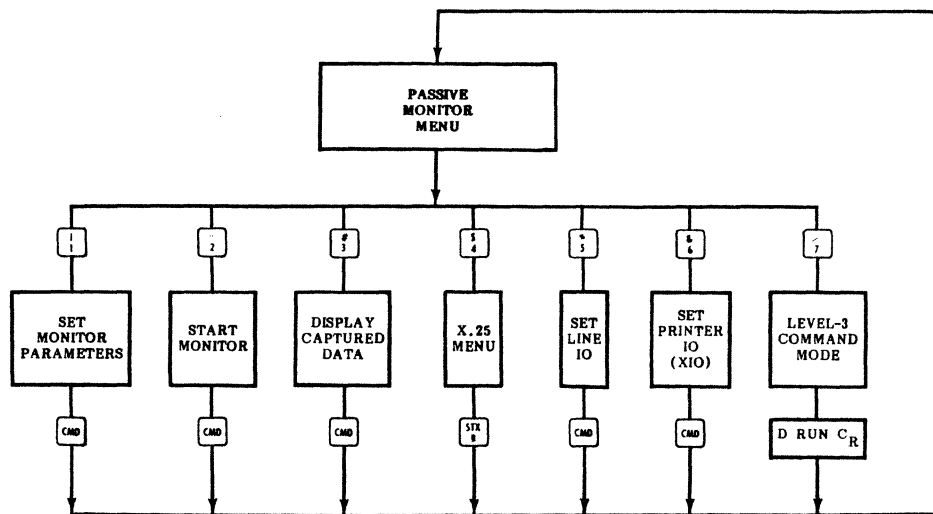
**11. FAST CAPTURE AND DISPLAY,** STX  
B

**11.01** This mode of operation is designed to input data directly to the capture buffer while eliminating the on-line display of data in the split screen format. Combined with the ability to capture send or receive data only, this mode significantly increases the speed of operation as described in paragraph 10.32. When selected from the X25/X75 MENU, this item presents the user with the PASSIVE MONITOR MENU (Figure 8-39) listing those items discussed in paragraphs 11.03 through 11.12.



**Fig. 8-39** Passive Monitor Menu

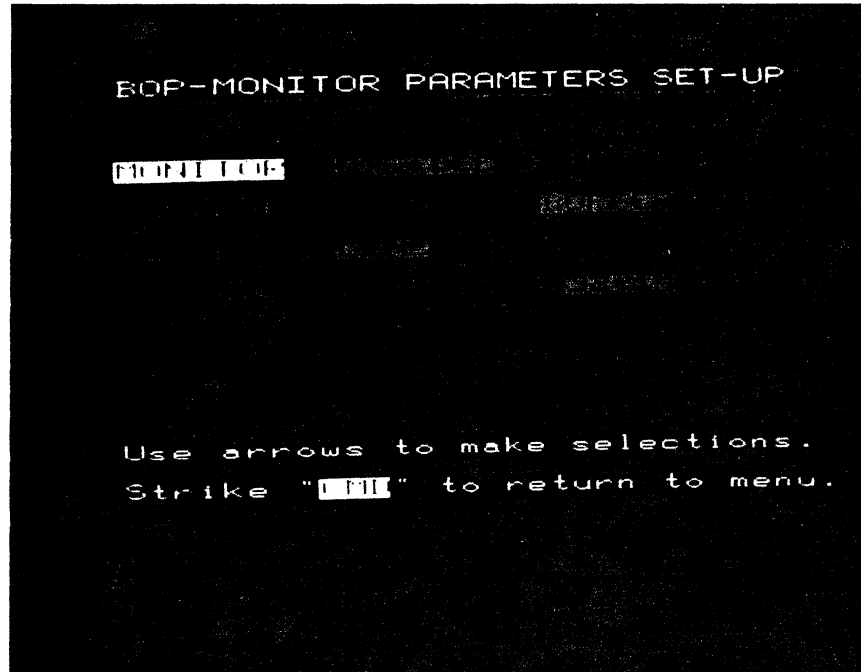
**11.02** Figure 8-40 is a keystroke flowchart showing how each item in the PASSIVE MONITOR MENU is selected and how the user returns from that selection to the menu once again.



**Fig. 8-40** Passive Monitor Menu Flowchart

**SET MONITOR PARAMETERS,** 

**11.03** The monitor parameters, shown in Figure 8-41, allow the user to capture send or receive data, or both, to store data in the buffer only or on disc, and to display either a line of real time data or a frame count in any of the ENCORE's four screen sizes. "SET MONITOR PARAMETERS" is selected from the PASSIVE MONITOR MENU by striking the  key.



**Fig. 8-41 BOP Monitor Parameters Set-Up**

**11.04 Monitor.** The monitor parameter allows the user to input data via pin 2, pin 3, or both pins simultaneously. This ability to input in a half-duplex format rather than full-duplex helps to boost operating speed by reducing the amount of data that must be input at one time.

**11.05 Capture.** This parameter allows the user to capture data using only the 2K solid state buffer or to capture and then transfer the data to disc. If the 2K buffer is sufficient for data storage, the speed of operation can again be increased by not transferring data to the disc. If captured data is to be displayed in the Fast Capture and Display Mode, it must have been previously captured in this mode.

**11.06 Display.** This parameter allows the user to view data real time without the advantage of frame level and packet level headers to define the X25 information. This also helps to increase the speed of operation. In addition, the user may choose to simply display a count of the number of frames received or he may turn the screen completely off. In general, the less data presented to the screen, the higher the speed of operation.

**11.07 Screen.** The display of real time data or the frame count may be further enhanced by changing the screen size. The ENCORE will display data in any one of four configurations:

- 32-7 = Seven lines of 32 characters each
- 64-7 = Seven lines of 64 characters each
- 32-14 = Fourteen lines of 32 characters each
- 64-14 = Fourteen lines of 64 characters each

**START MONITOR, 2**

**11.08** The PASSIVE MONITOR MENU includes the item "START MONITOR" because fast capture does not begin automatically following the selection of monitor parameters. This affords the user an opportunity to make another selection from the submenu prior to executing the Fast Monitor. To start this monitor, select item 2 from the PASSIVE MONITOR MENU.

**DISPLAY CAPTURED DATA, #  
3**

**11.09** "DISPLAY CAPTURED DATA" is the same item described in paragraph 10.11 for the display of data captured using the Real Time Monitor. All of the Capture Display Commands given in Table 8-5 are also used to control the display of fast captured data. Data is displayed in the same split screen format described earlier except that send (DTE) or receive (DCE) data may not be present depending on how the data was captured (full- or half-duplex). Data captured in the Fast Capture and Display Mode must have previously captured in the same mode.

**X.25 MENU, 5  
4**

**11.10** Selecting "X25 MENU" from the PASSIVE MONITOR MENU simply instructs the ENCORE to display the X25/X75 MENU once again.

**SET LINE IO,**

**11.11** "SET LINE IO" is the same item described in paragraph 10.31 for operation with the Real Time Monitor. It is offered again in the PASSIVE MONITOR MENU strictly for operator convenience.

**SET PRINTER IO (XIO),**

**11.12** SET PRINTER IO (XIO) is the same item described in paragraph 10.41 for operation with the Real Time Monitor. It is offered again in the PASSIVE MONITOR MENU strictly for operator convenience.














**LEVEL-3 COMMAND MODE, **

**11.13** Selection of the "LEVEL-3 COMMAND MODE" from the PASSIVE MONITOR MENU and from the X25/X75 MENU permits escape from the X25 Operating System and allows the user to enter ENCORE's Level-3 mode of operation. Level-3, COMBASIC, is detailed in Chapter 7.

**TYPICAL FAST CAPTURE AND DISPLAY OPERATING PROCEDURE**

**11.14** For the purpose of this procedure, it is assumed that the ENCORE is connected to the circuit under test and is powered-up in accordance with the Initial Operating Procedure given in paragraph 8.01. It is also assumed that the default packet specifications are adequate for successful monitoring of the packet-switched data.

**TYPICAL FAST CAPTURE AND DISPLAY OPERATING PROCEDURE**

STEP	PROCEDURE
1.	From the X25/X75 MENU select FAST CAPTURE AND DISPLAY by striking the  key. Note that the ENCORE displays the PASSIVE MONITOR MENU.
2.	Select "SET MONITOR PARAMETERS" from the submenu by striking the  key. Note that the ENCORE displays the "BOP MONITOR PARAMETERS SET-UP".
3.	Using the  ,  ,  , and  keys, select the desired parameters (see paragraphs 10.03 through 10.07).
4.	Strike the  key and note that the ENCORE again displays the "PASSIVE MONITOR MENU".
5.	From the "PASSIVE MONITOR MENU" begin the fast monitor and capture of X25 information by striking the  key.
6.	When sufficient data has been captured, strike the  to end the monitoring and capture of data and note that the ENCORE again displays the "PASSIVE MONITOR MENU".
7.	Select "DISPLAY CAPTURE DATA" by striking the  key and note that the ENCORE displays data in the same split screen format as described in the Typical Monitor and Display Procedure (paragraph 10.44).
8.	Use the following Capture Display to further enhance the display of information.
	 Scroll forward one line.
	 Scroll backward one line.
	 Scroll forward to the last page of captured data.



**TYPICAL FAST CAPTURE AND DISPLAY OPERATING PROCEDURE (Cont'd)**

STEP	PROCEDURE
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">SHIFT</div> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">↓</div> <div>Scroll backward to the first page of captured data.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">→</div> <div>Scroll forward continuously at speeds of 0 (fastest) through 9 (slowest).</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">←</div> <div>Scroll backward continuously at speeds of 0 (fastest) through 9 (slowest).</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">ENC E</div> <div>Search to next error.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">DC3 S</div> <div>Enter string sequence and search to first appearance.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">SOH A</div> <div>Advance to next appearance of string sequence.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">SI O</div> <div>Output the contents of the screen via the XIO port.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">ACK F</div> <div style="margin-right: 10px;">/</div> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">ACK F</div> <div>Enable or disable display of frame information.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">DLE P</div> <div style="margin-right: 10px;">/</div> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">DLE P</div> <div>Enable or disable display of packet information.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">EOT D</div> <div style="margin-right: 10px;">/</div> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">EOT D</div> <div>Enable or disable display of packet level user data.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">HEX</div> <div style="margin-right: 10px;">/</div> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">HEX</div> <div>Enable or disable HEX display of packet level user data.</div> </div>
	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 10px;">CMD</div> <div>Exit Display Mode and return to submenu.</div> </div>
9.	To exit the Display Mode simply strike the <span style="border: 1px solid black; padding: 2px;">CMD</span> key and note that the ENCORE again displays the "PASSIVE MONITOR MENU".

This completes the Typical Fast Capture and Display Operating Procedure

12. PACKET-LEVEL ACTIVE, ETX  
C

12.01 The Packet-Level Active Mode allows the user to write and execute a scenario (program) that will instruct the ENCORE to actively simulate DTE, DCE, or passively monitor the interface in an X25/X75 environment. Once written, the scenario may be stored on disc and recalled for execution or edit at some later time. In addition, the user may choose either of two interactive display modes which emphasize the scenario or the line activity. The scenario instruction set is detailed in paragraph 12.19. The information needed to write, edit, and execute a scenario are included in paragraphs 12.12 through 12.18. To select the PACKET-LEVEL ACTIVE Mode, strike the ETX  
C key while the X25/X75 MENU is displayed and note the display of the X25 ACTIVE MENU as shown in Figure 8-42.

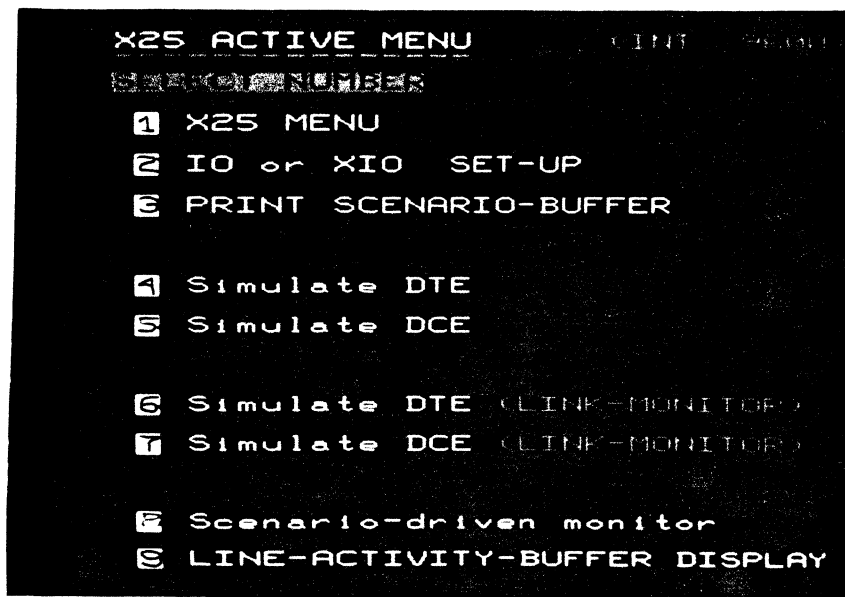


Fig. 8-42 X25 Active Menu

12.02 Figure 8-43 is a keystroke flowchart showing how each item in the X25 ACTIVE MENU is selected and how the user returns from that selection to the menu once again. Each of these items is discussed further in paragraphs 12.03 through 12.11.

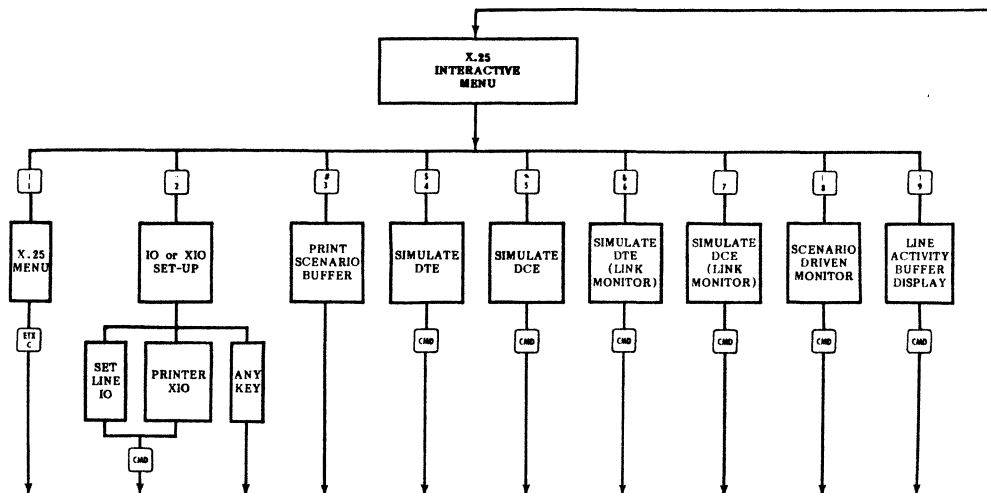


Fig. 8-43 X25 Active Menu Flowchart

**X25 MENU,  1**

**12.03** Selecting this item from the X25 ACTIVE MENU instructs the ENCORE to, once again, display the X25/X75 MENU.

**IO or XIO SET-UP,  2**

**12.04** This item prompts the user for entry of LINE IO or PRINTER XIO parameters. Response to the video prompt instructs the ENCORE to enter the appropriate mode, IO or XIO, as described in paragraphs 10.31 and 10.41, respectively.

**PRINT SCENARIO-BUFFER,  # 3**

**12.05** When the desired XIO parameters are set, this item instructs the ENCORE to output the contents of the scenario buffer via the XIO port. This is especially useful when a hard copy of a scenario is required for record purposes. XIO parameters must be set prior to selection.

**SIMULATE DTE,  4**

**12.06** This is the first of two packet-level simulate modes. In this mode, the ENCORE simulates DTE and in addition to the scenario, displays only packet-level information in the Line Activity Window. All of the scenario and display commands shown in Table 8-6 may be used in this mode and all scenario instructions except DTE-ENABLE, DCE-ENABLE, and BOTH-ENABLE may be used.

**SIMULATE DCE,  5**

**12.07** This is the second packet-level simulate mode. It offers the same features as item  4 except that the ENCORE simulates DCE instead of DTE.

**SIMULATE DTE (LINK MONITOR),  6**

**12.08** This is the first of two simulate modes that display only link-level information in the Line Activity Window. In all other respects it is the same as the SIMULATE DTE mode listed as item  4 in the Interactive Menu.

**SIMULATE DCE (LINK MONITOR),  7**

**12.09** This is the second simulate mode that displays only link-level information in the Line Activity Window. It functions in the same manner as the SIMULATE DTE (LINK MONITOR) Mode except that it simulates DCE.

**SCENARIO-DRIVEN MONITOR,  8**

**12.10** The SCENARIO-DRIVEN MONITOR Mode is similar to the simulate modes because it too uses the scenario and display commands shown in Table 8-6 and many of the scenario instructions. The instructions that cannot be used in this mode are automatically changed to NOP instructions (no operation) when the scenario is displayed. In addition, the DTE-ENABLE, DCE-ENABLE, and BOTH-ENABLE can now be used to limit the type of information displayed.




**LINE-ACTIVITY-BUFFER DISPLAY, **

12.11 This mode allows the user to recall captured information from the buffer for display or output to an external device via the XIO port. It also permits buffer search for a specified sequence of up to 15 characters in length and manual or automatic scrolling. The keystrokes required for all of these functions are defined in Table 8-6.

**NOTE:** The disc must not be write protected when data is captured for display in this mode.

**HOW TO WRITE A SCENARIO**

12.12 A scenario is a program, up to 2 048 bytes in length, that can be written and executed by the ENCORE in any of the interactive simulate or monitor modes. In addition, a scenario can be stored on disc and recalled for later execution. The tools required by the user for writing, executing, and saving a scenario include a series of scenario commands and a comprehensive set of program or scenario statements. The commands are defined in Table 8-6. The complete set of scenario instructions is included under paragraph 12.19.

12.13 When operating in any of the interactive simulate or monitor modes, the following keystrokes may be used to write, edit, and control execution of the scenario. The desired instruction may be selected by scrolling through the entire instruction set using the  and  keys or by scrolling alphabetically using the key that corresponds with the first letter of the desired instruction. Either one of two display modes, LINE ACTIVITY or SCENARIO may also be selected as noted below. Where the  key is shown, depress and hold while striking the adjacent key.

**TABLE 8-6  
INTERACTIVE SCENARIO AND DISPLAY COMMANDS**


































COMMAND	DESCRIPTION
 	Displays last ten lines of scenario when in the SCENARIO DISPLAY MODE or END if in the LINE ACTIVITY DISPLAY MODE. Displays last page of buffer in the LINE ACTIVITY BUFFER DISPLAY Mode.
 	Displays first line of scenario in either the SCENARIO DISPLAY or LINE ACTIVITY DISPLAY Mode. Displays first page of captured information in the LINE ACTIVITY BUFFER DISPLAY Mode.
	Used in conjunction with the  or  keys to insert instructions when writing or editing a scenario.  <b>NOTE:</b> For easy selection of instructions, follow  with the first letter of the desired instruction and repeat to scroll.
	Used in conjunction with the  key to delete scenario instructions.
 	Used to scroll through the scenario or the capture buffer and to terminate entry of scenario instructions.

TABLE 8-6

INTERACTIVE SCENARIO AND DISPLAY COMMANDS (Cont'd)

COMMAND	DESCRIPTION
 	<p>These keys permit entry of instruction arguments and the scrolling through possible arguments for selected instructions such as WAIT-FOR, TRANSMIT, etc.. These arguments must be terminated by the key. In the LINE ACTIVITY BUFFER DISPLAY Mode, these keys initiate automatic scroll.</p>
	<p>Used in conjunction with  (read scenario from disc) or  (save scenario on disc) for storing and recalling any one of up to eight scenarios (0 through 7). Once  or  is entered, it is immediately followed by entry of any numbered key from  through  to select the desired scenario.</p>
	<p>Used to alternate between the two interactive display modes (LINE ACTIVITY and SCENARIO).</p>
	<p>Terminates entry of scenario instructions and/or arguments.</p>
	<p>Stops execution of the buffer stored scenario in the interactive modes. Stops automatic scroll and buffer output in the LINE ACTIVITY BUFFER DISPLAY Mode.</p>
	<p>Executes the buffer stored scenario with automatic write to disc.</p>
	<p>Executes the buffer stored scenario.</p>
	<p>Increases the speed of operation by halting further update of the scenario display while the scenario continues to be executed.</p>
	<p>Automatically terminates a response at any point in the execution of the scenario if a response is pending.</p>
	<p>Outputs the contents of the capture buffer via the XIO port. Data may be output in HEX when so displayed.</p>
	<p>In the LINE ACTIVITY BUFFER DISPLAY Mode, this command permits capture buffer search of specified character sequence (15 characters max). When located, the specified sequence is underscored.</p>
<p><b>SPACE</b></p>	<p>Single steps, or permits execution of the buffer stored scenario one line at a time.</p>
	<p>Permits entry of HEX pairs when keying user data in the argument for the TRANSMIT instruction. The  key must be struck prior to entry of each HEX pair. In the LINE ACTIVITY BUFFER DISPLAY Mode, this command changes the data display to HEX.</p>
	<p>Instructs the ENCORE to display the X25 ACTIVE MENU.</p>

**TABLE 8-6**  
**INTERACTIVE SCENARIO AND DISPLAY COMMANDS (Cont'd)**

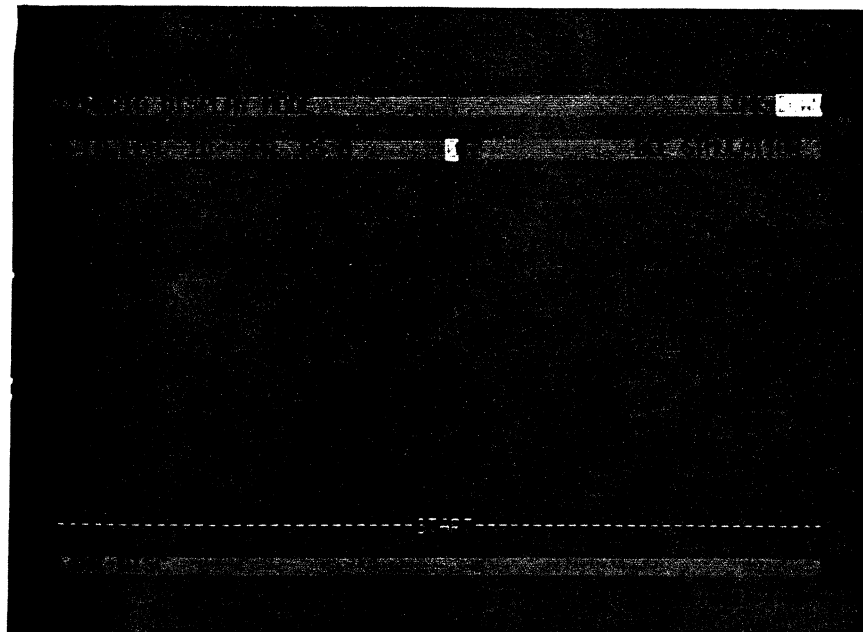
COMMAND	DESCRIPTION
<div style="border: 1px solid black; padding: 2px; display: inline-block;">CLEAR</div>	<p>Permits review of buffer controls from active mode (note REVIEW in display header). This mode can only be selected while in the Line Activity Display Mode and only if information is contained in the buffer. In the review mode, scrolling is accomplished using the <span style="border: 1px solid black; padding: 0 2px;">↑</span>, <span style="border: 1px solid black; padding: 0 2px;">↓</span>, <span style="border: 1px solid black; padding: 0 2px;">SHIFT ↑</span>, and <span style="border: 1px solid black; padding: 0 2px;">SHIFT ↓</span> keys. Strike any key to escape.</p> <p><b>Note:</b> The first line of information in the Line Activity Buffer cannot be displayed in the Review Mode.</p>

**A. TYPICAL SCENARIO ENTRY PROCEDURE**

**12.14** A scenario may be entered in any of the active simulate or monitor modes. These modes are listed as items 4, 5, 6, 7, and 8 in the X25 ACTIVE MENU. The procedure that follows demonstrates the recommended method for entering scenario instructions. For the purpose of this procedure, we will enter the first four instructions of the example scenario provided under paragraph 12.18 on page 8-61.

**TYPICAL SCENARIO ENTRY PROCEDURE**

STEP	PROCEDURE
1.	From the X25/X75 MENU, select "PACKET-LEVEL ACTIVE" by striking the <span style="border: 1px solid black; padding: 0 2px;">EX C</span> key and note display of the X25 ACTIVE MENU.
2.	From the X25 ACTIVE MENU, select item <span style="border: 1px solid black; padding: 0 2px;">5</span> (example scenario, paragraph 12.18 simulates DCE) and note a display similar to that shown below.



**Fig. 8-44 Scenario Entry Display**

TYPICAL SCENARIO ENTRY PROCEDURE (Cont'd)

STEP	PROCEDURE
3.	Strike the $\boxed{\kappa}$ key and note that "INSERT-FOR" is displayed on the line between START and END.
4.	Enter the SET WINDOW instruction by continually striking the $\boxed{\text{DC3}}$ key until SET WINDOW appears on the display. Then strike the $\boxed{\wedge}$ key to terminate entry of the instruction.
5.	Enter the SET WINDOW argument by striking the $\boxed{-}$ key. Then type the number 7, and terminate by striking the $\boxed{\wedge}$ key.
6.	Strike the $\boxed{\uparrow}$ key to move the first instruction up one line. Then strike the $\boxed{\kappa}$ key and note that "INSERT-FOR" appears as the second line of the scenario.
7.	Enter the next instruction by striking the $\boxed{\text{RM}}$ key and note that "MARKER" appears as the second line of the scenario. Terminate this instruction by striking the $\boxed{\wedge}$ key.
8.	Enter the "MARKER" argument by striking the $\boxed{-}$ key. Then enter the number 0, and terminate by striking the $\boxed{\wedge}$ key.
9.	Strike the $\boxed{\uparrow}$ key to move this instruction up one line. Then strike the $\boxed{\kappa}$ key and note that "INSERT-FOR" again appears as the next line of the scenario.
10.	Enter the next instruction by striking the $\boxed{\text{REL 6}}$ key and note that "GOSUB-MARKER-#" appears as the next line of the scenario. Terminate this instruction by striking the $\boxed{\wedge}$ key.
11.	Enter the "GOSUB-MARKER-#" argument by striking the $\boxed{-}$ key. Then type the number 100, and terminate by striking the $\boxed{\wedge}$ key.
12.	Strike the $\boxed{\uparrow}$ key to move this instruction up one line. Then strike the $\boxed{\kappa}$ key and note that "INSERT-FOR" is displayed once again.
13.	Enter the next instruction by continually striking the $\boxed{\text{DC3}}$ key until "SKIP-UNLESS-LAST =" appears as the next line of the scenario. Terminate this instruction by striking the $\boxed{\wedge}$ key.

TYPICAL SCENARIO ENTRY PROCEDURE (Cont'd)

STEP	PROCEDURE
14.	<p>Enter the argument for this instruction by continually striking the <input type="checkbox"/> key until the mnemonic "DATA" is displayed and then terminate by striking the <input type="checkbox"/> key.</p> <p>The above steps demonstrate the method by which scenario instructions and their arguments can be entered. You will notice that some arguments are typed in and others are entered by scrolling through predetermined mnemonics using the <input type="checkbox"/> and <input type="checkbox"/> keys. This method eliminates syntax errors and greatly simplifies the entry of instructions and arguments. The remaining lines of the example scenario, or the lines of your own scenario, are all entered as described in the preceding steps. Using the <input type="checkbox"/>, <input type="checkbox"/>, <input type="checkbox"/>, and <input type="checkbox"/> keys, you may scroll through the scenario and make corrections as necessary (see Table 8-6). Once the scenario is completed, be sure that the disc is <u>not write protected</u> and save the scenario using the <input type="checkbox"/> <input type="checkbox"/> keys as described in Table 8-6. In addition, be sure that the disc is <u>not write protected</u> if data is to be captured in the active modes.</p>

This completes the Typical Scenario Entry Procedure.

**12.15** When one of the scenario driven modes is selected, the ENCORE enters the SCENARIO DISPLAY Mode showing the first line of the scenario currently residing in the scenario buffer. If the scenario buffer is empty, a new scenario may be written or an existing scenario may be transferred from the disc to the buffer using the  key (see Table 8-6). A scenario may then be edited, executed, and saved using the commands given in Table 8-6.

**A. Interactive Displays**

**12.16** In each of the interactive monitor or simulate modes, the ENCORE offers the user a choice of two displays. The first, Figure 8-45, shows the user up to 10 lines of the scenario at one time while the Line Activity Window displays only one line of packet-level activity. This display is the default condition assumed whenever an interactive simulate or monitor mode is selected. The second display, Figure 8-46, shows up to 10 lines of activity while displaying the current scenario instruction on a single line at the top of the CRT. This display replaces the default display when the  key is depressed. The default display is returned by striking the  key again.



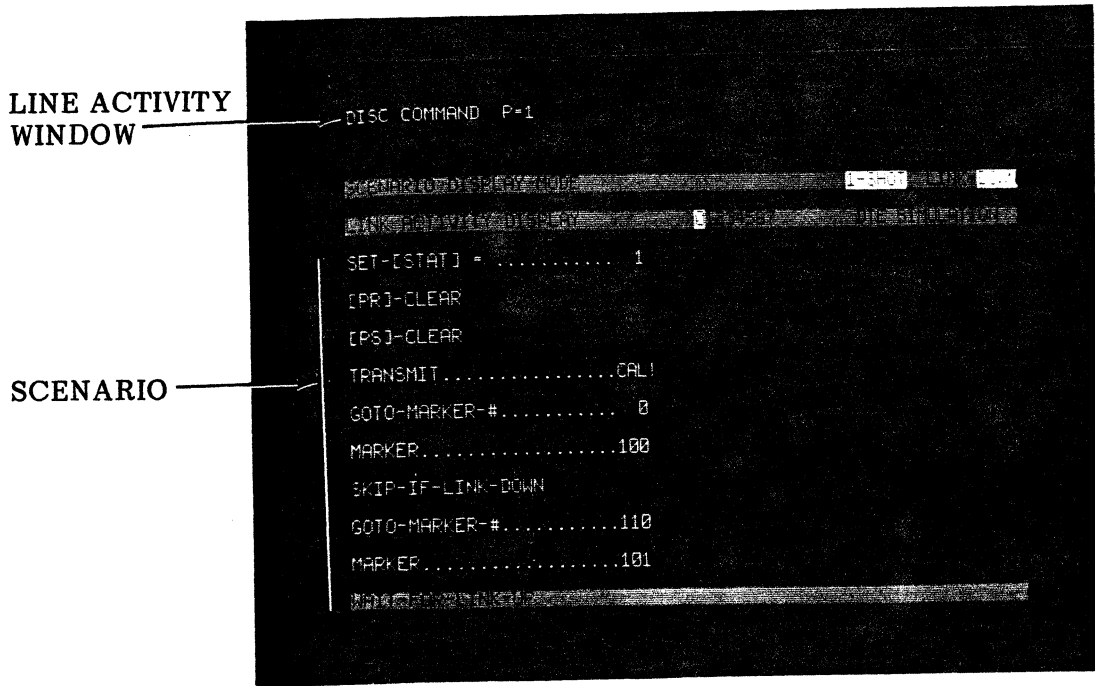


Fig. 8-45 Scenario Display Mode

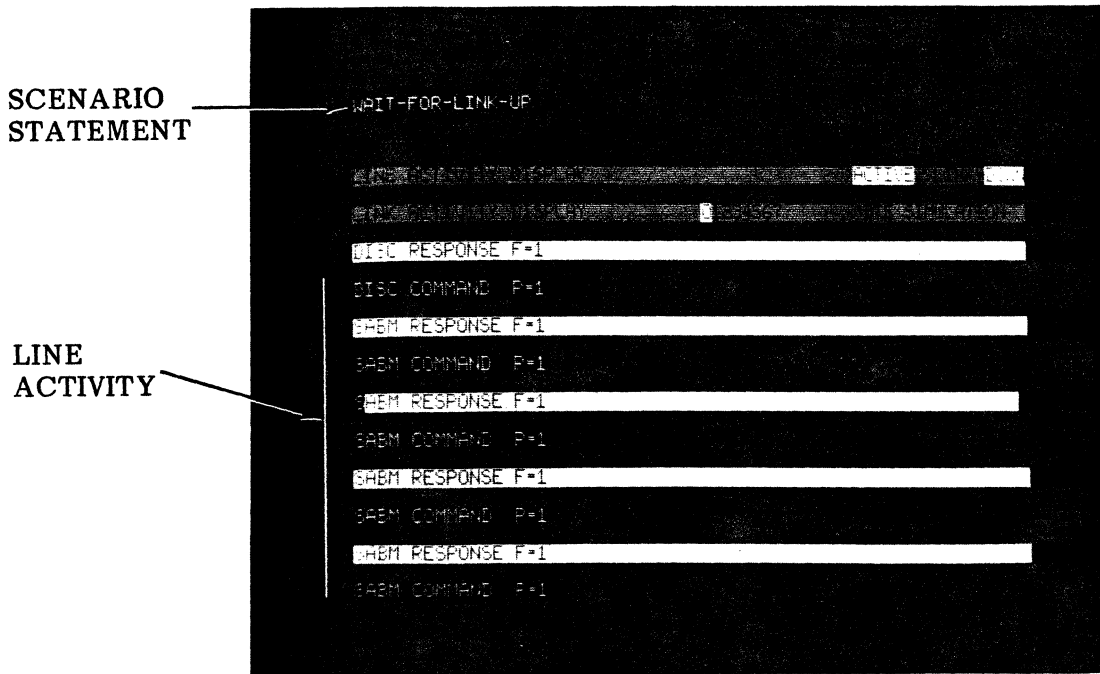
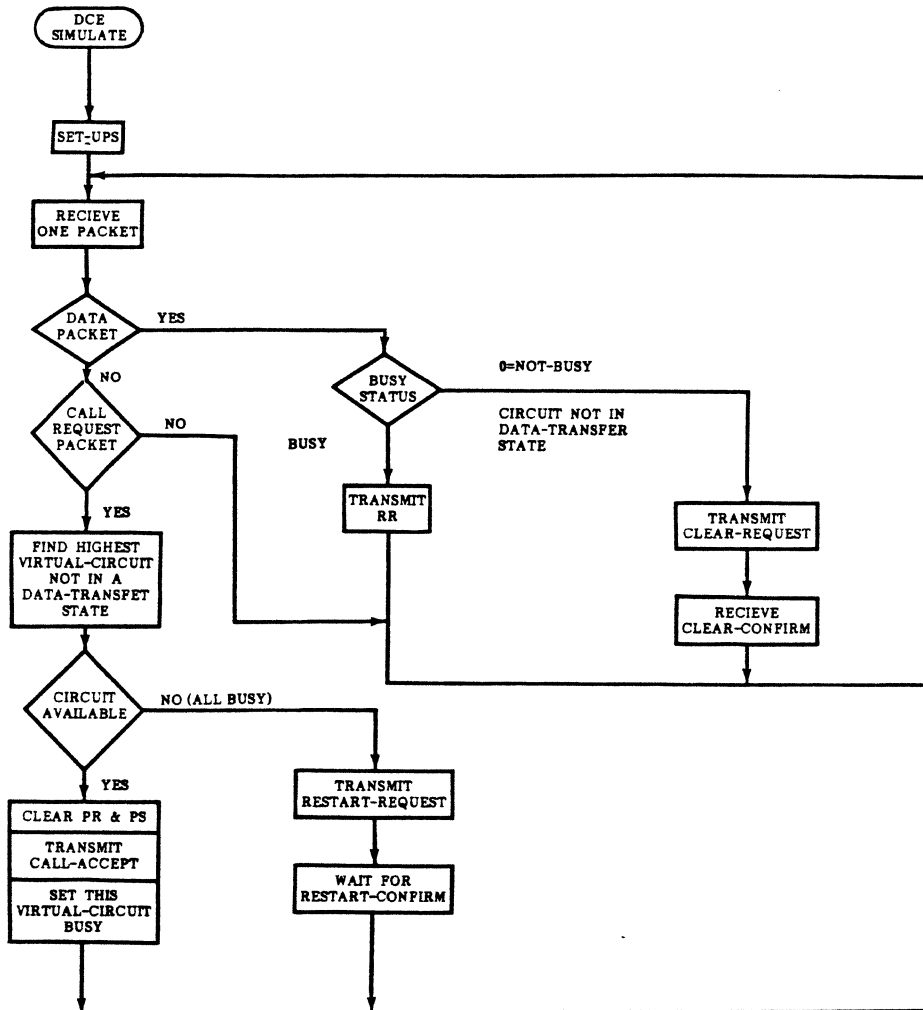


Fig. 8-46 Line Activity Display

**B. Example Flowchart**

**12.17** The following flowchart, Figure 8-47, and scenario are provided as an example of how a scenario may be written. The complexity of the program depends entirely upon the needs of the user in a test situation. The example scenario is written to accomplish the following:

- Transmit a Restart Request (RST?) and wait for a Restart Confirmation (RST!) whenever the link assumes the LINK-UP state.
- Transmit the proper response to the following packets:
  - Restart-Request (RST?)
  - Clear-Request (CLR?)
  - Call-Request (CAL?)
- Respond to a data packet with a Receive-Ready (RR) packet.
- Maintain a state variable for each virtual circuit.



**Fig. 8-47 Scenario Flowchart**

**C. Example Scenario**

**12.18** The example scenario shown below simulates DCE and includes a short explanation of each statement. Additional information is found in paragraph 12.19 and Appendix E.

SCENARIO BUFFER RECORD ***** START *****	EXPLANATION OF STATEMENT
SET WINDOW.....7	Sets the maximum number of outstanding frames to 7.
MARKER.....0	Marks a point within the scenario for later use with GOTO or GOSUB.
GOSUB-MARKER-#.....100	Executes a program call to marker #100 where a Restart Request is transmitted and a Reset Confirmation is expected when the link transfers from a down to up state.
SKIP-UNLESS-LAST = .....DATA	Skips the next instruction unless the last packet received was a data packet.
GOTO-MARKER-#.....1	Executes a program jump to marker #1.
SKIP-UNLESS-LAST = .....CAL?	Skips the next instruction unless the last packet received was a Call Request (CAL?).
GOTO-MARKER-#.....2	Executes a program jump to marker #2 if the CAL? was received.
GOTO-MARKER-#.....0	Executes a program jump back to marker #0 if the CAL? was not received.
MARKER.....1	Marks a point within the scenario where execution continues if the last packet received is a data packet.
SKIP-UNLESS-[STAT] = ...0	Skips the next instruction if the status byte for the current logical channel is not 0 (busy, not in data transfer state).
GOTO-MARKER-#.....3	Executes a program jump to marker #3 where a Clear Request is transmitted prior to a program jump to marker #100.
TRANSMIT..... RR	Transmits a Receiver Ready if the current status byte is not 0.
GOTO-MARKER-#.....0	Executes a program jump back to marker #0.
MARKER.....3	Marks a point within the scenario where execution continues if the status byte for the current logical channel is 0.

## PACKET LEVEL ACTIVE

### SCENARIO BUFFER RECORD (Cont'd)

### EXPLANATION OF STATEMENT (Cont'd)

TRANSMIT.....CLR?

Transmits a Clear Request if the current status byte is 0.

GOSUB-MARKER-#.....100

Executes a program jump to marker #100.

SKIP-UNLESS-LAST = .....CLR!

Skips the next instruction if the last packet received was not a Clear Confirmation.

GOTO-MARKER-#.....0

Executes a program jump back to marker #0.

MARKER.....4

Marks a point within the scenario where circuits must be reset.

GOSUB-MARKER-#.....101

Transfers scenario control to marker 101 where the ENCORE waits for LINK-UP.

GOTO-MARKER-#.....0

Executes a program jump back to marker #0.

MARKER.....2

Marks a point within the scenario where execution continues if the last packet received was a Call Request. Search for free channel begins.

SET-LCN.....0

Sets the logical channel number to 0.

FOR.....255

Begins a program loop that can be repeated 255 times.

LCN-1

Decrements the logical channel number.

SKIP-UNLESS-[STAT] = ...0

Skips the next instruction if the channel is busy.

GOTO-MARKER-#.....5

Transfers scenario control to marker #5 if the channel is not busy.

NEXT

Repeats the program loop beginning with the FOR statement.

GOTO-MARKER-#.....4

Transfers scenario control to marker #4 if all 255 channels are busy.

MARKER.....5

Marks a point within the scenario where execution continues when a free channel is found.

**PUB. NO. 810-00184A**

SCENARIO BUFFER RECORD (Cont'd)

EXPLANATION OF STATEMENT (Cont'd)

SET-[STAT] = .....1	Sets the status of the current channel to 1 (busy).
[PR]-CLEAR	Sets the packet receive sequence number on the current channel to 0.
[PS]-CLEAR	Sets the packet send sequence number on the current channel to 0.
TRANSMIT.....CAL!	Transmits a Call Accept packet.
GOTO-MARKER-#.....0	Executes a program jump back to marker #0.
MARKER.....100	Marks the beginning of the subroutine to input a packet.
SKIP-IF-LINK-DOWN	Skips the next instruction if the link is down.
GOTO-MARKER-#.....110	Transfers scenario control to marker #110 if the link is up.
MARKER.....101	Marks the beginning of a routine to wait for link up and then place the logical channels in the ready state.
WAIT-FOR-LINK-UP	Transmits DISC commands at T1 intervals until the link is established.
SET-TIMER.....4	Sets the internal count down timer to 4 seconds.
SET-LCN.....0	Sets the current logical channel number to 0.
TRANSMIT.....RST?	Transmits a restart indication packet.
WAIT-FOR.....RST!	Waits for receipt of a restart confirmation packet. If it is not received within 4 seconds, scenario control is transferred to the next ERROR ENTRY statement.
CLEAR-TIMER	Clears the internal count down timer if RST! is received before time out.
MARKER.....102	Marks the point within the scenario where execution continues after the ENCORE responds to DTE's restart request.
CLEAR-STATS	Clears all 255 status bytes.
GOTO-MARKER-#.....100	Transfers scenario control back to marker 100 after the ENCORE enters the ready state.

## PACKET LEVEL ACTIVE

### SCENARIO BUFFER RECORD (Cont'd)

### EXPLANATION OF STATEMENT (Cont'd)

ERROR-ENTRY

Marks the point within the scenario where execution continues if reset confirmation is not received within 4 seconds.

GOTO-MARKER-#.....101

Transfers scenario control back to marker 101 where the ENCORE again waits for link up.

MARKER.....110

Marks the point within the scenario where execution continues if the link is up.

SET-TIMER.....4

Sets the internal count down timer to 4 seconds.

RECEIVE-ANY

Receives any packet type.

CLEAR-TIMER

Clears the internal timer if a packet is received within 4 seconds.

SKIP-UNLESS-LAST = .....RST?

Skips the next instruction unless the last packet received was a restart request.

GOTO-MARKER-#.....103

Transfers scenario control to marker 103 if the last packet received was a restart request.

SKIP-UNLESS-LAST = .....CLR?

Skips the next instruction unless the last packet received was a clear packet.

GOTO-MARKER-#.....104

Transfers scenario control to marker 104 if the last packet received was a clear request.

COPY-LAST-LCN

Sets the current logical channel number (LCN) to the same value as the last received LCN.

RETURN

Returns scenario control back to the instruction immediately following the last executed GOSUB-MARKER-#.

.....ERROR-ENTRY

Marks the point within the scenario where execution continues if no packets are received after link up.

GOTO-MARKER-#.....100

Transfers scenario control back to marker 100 if no packets are received after link up.

MARKER.....103

Marks the point within the scenario where execution continues when a restart request is received.

SCENARIO BUFFER RECORD (Cont'd)	EXPLANATION OF STATEMENT (Cont'd)
AUTO-REPSONSE	Automatically transmits a restart confirmation in response to restart request.
GOTO-MARKER-#.....102	Transfers scenario control back to marker 102 where all status bytes are cleared prior to receiving the next packet.
MARKER.....104	Marks the point within the scenario where execution continues when a clear request packet is received.
AUTO-RESPONSE	Automatically transmits a clear confirmation in response to clear request.
SET-[STAT]= .....0	Sets the status byte for the current logical channel to 0.
GOTO-MARKER-#.....100	Transfers scenario control back to marker 100 where the link status is determined once again.
***** END *****	

**D. Scenario Instruction Set**

**12.19** The following is a complete description of the entire X25/X75 Scenario Instruction Set. An abbreviated listing is found in Appendix E. Each of these instructions requires from 1 to 7 bytes of memory.

**1. FOR:** Creates a program loop that is repeated a specified number of times. The argument is entered as any integer from 1 to 255. The NEXT instruction indicates the end of the program loop. Default is to 1.

EXAMPLE:       FOR.....3  
                   TRANSMIT.....DATA  
                   WAIT-FOR..... RR  
                   NEXT

**2. NEXT:** Used in conjunction with the FOR instruction to mark the end of a program loop. No argument required.

**3. SET-TIMER:** Sets an internal count down timer to any value from 1 to 255 seconds. When executed, the contents of the timer are shown in the SCENARIO or LINE ACTIVITY DISPLAY headers. If the timer runs out (not cleared), scenario control is transferred to the next ERROR ENTRY instruction listed in the scenario. Default is to 1 second.

EXAMPLE:       MARKER.....110  
                   SET-TIMER.....4  
                   RECEIVE-ANY  
                   CLEAR-TIMER  
                   SKIP-UNLESS-LAST = .....RST?  
                   GOTO-MARKER-#.....103  
                   SKIP-UNLESS-LAST = .....CLR?  
                   GOTO-MARKER-#.....104

EXAMPLE: (Cont'd)

```
COPY-LAST-LCN  
RETURN  
.....ERROR-ENTRY  
GOTO-MARKER-#. ....100
```

- 4. **CLEAR-TIMER:** Clears the internal timer. No argument required.
- 5. **WAIT-FOR:** Halts further execution of the scenario until a specified packet is received. Default is to RR.

EXAMPLE: WAIT-FOR.....RR

- 6. **RECEIVE-ONLY:** Specifies the next packet to be received. If a non argument packet is received, program control is transferred to the next ERROR ENTRY instruction. Default is to RR.

EXAMPLE: RECEIVE-ONLY.....RR

- 7. **TRANSMIT:** Specifies the packet to be transmitted. During entry of the argument provision is made, via operator prompts, for entry of user data, address, etc. depending on the packet type. Default is to RR. The argument is changed from one packet to another each time the  or  key is depressed.

EXAMPLE: TRANSMIT.....DATA

- 8. **SET-Q-BIT:** Sets the Q-bit (data qualifier) in transmitted packets to 1 or 0. Default is to 0.

EXAMPLE: SET-Q-BIT.....0

- 9. **SET-D-BIT:** Sets the D-bit (delivery confirmation) transmitted packets to 1 or 0. Default is to 0.

EXAMPLE: SET-D-BIT.....0

- \*7. **DTE-ENABLE:** Permits monitor of pin 2 (DTE) information only. No argument required.

- \*8. **DCE-ENABLE:** Permits monitor of pin 3 (DCE) information only. No argument required.

- \*9. **BOTH-ENABLE:** Permits monitor of both pin 2 (DTE) and pin 3 (DCE) information. This is the default condition assumed when the DTE-ENABLE or DCE-ENABLE instruction is not used. No argument required.

- 10. **SET-LCGN:** Sets the logical channel group number, for all packet transmissions, to any number from 0 to 15. Default is to 0.

EXAMPLE: SET-LCGN.....4

**NOTE:** Multiple SET-LCGN statements are premissable.



**11. SET-LCN:** Sets the logical channel number, for all packets, to any number from 0 to 255. Default is to 0.

EXAMPLE: SET-LCN.....0

**NOTE:** Multiple SET-LCN statements are permissible.

**12. SET-M-BIT:** Sets the M-bit (more data), in data packets, to 1 or 0.

EXAMPLE: SET-M-BIT.....0

**13. PAUSE:** Creates a delay in execution of the scenario of 1 to 255 seconds. Default is to 1.

EXAMPLE: PAUSE.....1

**14. SET-N2:** Sets the N2 counter to any value from 1 to 255. This value represents the maximum number of frame transmissions or retransmission after the primary timer, T1, runs out. Default is to 1.

EXAMPLE: SET-N2.....1

**15. SET-T1:** Sets the primary timer, T1, to any value from 1 to 63 seconds. T1 is normally greater than the time between transmission of a command frame (SARM, DISC, or INFO frame) and its response. Default is to 1.

EXAMPLE: SET-T1.....1

**16. [PR]+1:** Increments the packet receive sequence number, P(R), by 1. P(R) and P(S) are automatically incremented by the ENCORE as normally required in the X25/X75 environment. This instruction allows the user to increment P(R) beyond its correct value, producing an intentional error. It is important to note that all bracketed instructions are only addressed by the current LCN. No argument required.

**NOTE:** Separate P(R) and P(S) counts are maintained for each LCN.

**17. [PR]-1:** Decrements the packet receive sequence number, P(R), by 1. This instruction allows the user to decrement P(R) beyond its correct value, producing an intentional error. It is important to note that all bracketed instructions are only addressed by the current LCN. No argument required.

**18. [PS]+1:** Increments the packet send sequence number, P(S), by 1. P(R) and P(S) are automatically incremented by the ENCORE as normally required in the X25/X75 environment. This instruction allows the user to increment P(S) beyond its correct value, producing an intentional error. It is important to note that all bracketed instructions are only addressed by the current LCN. No argument required.

**NOTE:** Separate P(R) and P(S) counts are maintained for each LCN.

**19. [PS]-1:** Decrements the packet send sequence number, P(S), by 1. This instruction allows the user to decrement P(S) beyond its correct value, producing an intentional error. It is important to note that all bracketed instructions are only addressed by the current LCN. No argument required.

## PACKET LEVEL ACTIVE

**20. [PR]-CLEAR:** Sets the packet receive sequence number, P(R), to 0 (initial requirement of some networks). No argument required.

**21. [PS]-CLEAR:** Sets the packet send sequence number, P(S), to 0 (initial requirement of some networks). No argument required.

**22. HALT:** Halts further execution of the scenario and may be used to separate multiple scenarios in a single buffer. GOTO and GOSUB statements will not jump a HALT instruction. To resume execution, use the  or  keys to scroll to the point where execution should continue, then strike the  or  key. No argument required.

**23. SET-WINDOW:** Allows the user to set the frame window size (number of outstanding frames) to any value from 1 to 7. The window size is shown in the LINE ACTIVITY DISPLAY headers.

**24. BAD-FCS-NEXT:** Instructs the ENCORE to transmit the next frame with an incorrect frame check sequence.

**25. MARKER:** Marks a point within the scenario that may be accessed whenever a GOTO-MARKER-# or GOSUB-MARKER-# is encountered. A HALT instruction located anywhere in the scenario between a GOTO or GOSUB and MARKER will prevent the program jump. The argument may be any integer from 0 to 255.

EXAMPLE:        MARKER.....0

**26. GOSUB-MARKER-#:** Executes a program call to the specified marker number. The specified marker indicates the beginning of a subroutine that may be terminated with a RETURN instruction. The RETURN executes a program jump back to the next instruction following the GOSUB-MARKER-#. The argument may be any integer from 0 to 255.

EXAMPLE:        GOSUB-MARKER-#.....100  
                 SET-LCN.....0  
                 FOR.....255  
                 LCN-1  
                 SKIP-UNLESS-[STAT]=.....0  
                 GOTO-MARKER-#.....155  
                 TRANSMIT.....CLR?  
                 WAIT-FOR.....CLR!  
                 SET-[STAT]=.....0  
                 MARKER.....155  
                 NEXT  
                 GOTO-MARKER-#.....199  
                 MARKER.....100  
                 SET-LCN.....0  
                 ALL-LCN'S  
                 TRANSMIT.....CAL?  
                 WAIT-FOR.....CAL!  
                 COPY-LAST-LCN  
                 [PR]-CLEAR  
                 [PS]-CLEAR  
                 SET-[STAT]=.....1  
                 FOR.....3  
                 TRANSMIT.....DATA

EXAMPLE: (Cont'd)

WAIT-FOR..... RR  
NEXT  
RETURN

**27. ERROR-ENTRY:** Used in conjunction with the SET TIMER and RECEIVE-ONLY instructions to mark the point within a scenario to which control is transferred on time out or receipt of a non-argument packet. No argument required.

**28. ALARM SOUND:** Sounds a 1 kHz audible alarm for a duration of not less than 0.2 seconds. No argument required.

**29. CLEAR-COUNT-#:** Clears any one of 16 specified internal counters numbered 0 through 15. Counter number 15 is cleared to 30 000 and when used in conjunction with SECONDS > COUNTER, will only input a number lower than its present contents. In a similar fashion, counter 0 will only input a number if it is higher than its present contents. Three other counters are automatically incremented by the ENCORE for use in counting the number of frames that must be retransmitted, the number of frame rejects received, and the number of frame rejects sent. These counters are numbered 12, 13, and 14, respectively, and although they can be cleared, incremented, and displayed, they are normally reserved for use by the ENCORE as described in the preceding sentences.

EXAMPLE: CLEAR-COUNT-#.....1

**30. COUNT-COUNTER-#:** Increments any one of 15 internal counters by 1. The argument specifies the counter and may be any integer from 0 through 15.

EXAMPLE: COUNT-COUNTER #.....1

**NOTE:** Counters 12, 13, and 14 are reserved (see instruction 29).

**31. DISPLAY-COUNT-#:** Displays the contents of any one of 16 internal counters in the SCENARIO or LINE ACTIVITY DISPLAY headers. The argument specifies the counter and may be any integer from 0 through 15.

EXAMPLE: DISPLAY-COUNT-#.....1

**32. CLEAR SECONDS:** Sets the value of the internal timer to 0. No argument required.

**33. SECONDS > COUNTER:** Transfers the seconds accumulated in the internal timer to any one of 16 specified counters. Counters 12, 13, and 14 should not be used as they are reserved for special functions (see instruction 29). Counter 15 will only input a number lower than its current value.

**34. LCN-1:** Decrements the logical channel number by 1. The logical channel number may range from 0 through 255. No argument required.

**35. LCN+1:** Increments the logical channel number by 1. The range and assignment of the LCN are discussed in the preceding paragraph. No argument required.

## PACKET LEVEL ACTIVE

**36. LOG-COUNT-#:** Logs the contents of a specified counter into the buffer while displaying the same information on the CRT. Any one of 16 counters may be specified as the argument. The counters are numbered 0 through 15.

**37. BUFFER-CLEAR/OPEN:** Clears the capture buffer and automatically writes captured information to the disc as the buffer is filled. Striking the COM key to execute the scenario accomplishes the same function. No argument required.

**38. TURN-OFF-LAW:** In the packet display simulate modes (items 4 and 5 in Active Menu), this instruction turns off the Line Activity Window. No argument required.

**39. TURN-ON-LAW:** This is the default condition assumed when TURN-OFF-LAW is not used. It allows the user to counter TURN-OFF-LAW as described in the preceding paragraph. No argument required.

**40. TRANSFER TO SCENARIO #:** Transfers execution to the specified scenario. During the transfer, however, "write to disc" is temporarily suspended and a "BUFFER SUSPENSION" message is logged to note the possible loss of data. The argument may be any integer from 0 through 7.

EXAMPLE:       TRANSFER-TO-SCENARIO-#.5

**41. LAPB:** Instructs the ENCORE to access the link using the balanced mode procedures. No argument required.

**42. LINK-DISC:** Instructs the ENCORE to transmit a DISC command to assure link disconnect. No argument required.

**43. WAIT-FOR-LINK-UP:** Depending on the simulate mode, this instructs the ENCORE to transmit SABM and/or DISC commands and responses in an effort to bring up the link. This continues at T1 intervals until the appropriate command or response is received. When the link is established, the scenario advances to the next instruction. No argument required.

**44. ALL-LCN'S:** Used primarily in the Scenario Driven Monitor Mode to counter the NOT-ALL-LCN'S instruction. No argument required.

**45. RETURN:** Used in conjunction with GOSUB-MARKER # to return control of the scenario to the instruction immediately following the GOSUB. No argument required.

**46. GOTO-MARKER-#:** Transfers control of the scenario to the instruction following the specified marker. The marker number may be any integer from 0 to 255.

EXAMPLE:       GOTO-MARKER-#.....2

**47. COPY LAST LCN:** Sets the current LCN to the LCN of the last packet received. No argument required.

**48. RECEIVE-ANY:** Instructs the ENCORE to receive any single packet. No argument required.

**49. SKIP-UNLESS-LAST =:** Skips the next instruction if the last packet received does not agree with the packet specified in the argument.

EXAMPLE: SKIP-UNLESS-LAST =.....DATA

**50. LOG-COMMENT:** Logs user comments into the capture buffer while displaying the same information on the CRT. Provision is made for entering an argument (comment) of up to 64 characters in length. Strike the  and  keys to enter the comment. This instruction can be used as the first instruction of a scenario to include a name, date, and other important test information.

EXAMPLE: LOG-COMMENT.....DATA

**51. !CHEAT-LINK-UP!:** Used primarily for off-line testing, this instruction tells the ENCORE to assume that the link is up. No argument required.

**52. DCE-LAP-MOD:** Modifies the link access procedure used in the two Simulate DCE modes,  and . Any one of the 6 numeric arguments shown in Table 8-7 may be used to modify the first frame transmitted during the link access procedure. This is used where the network procedure may differ slightly from the procedure defined in the X25/X75 recommendations. During DCE simulation, the ENCORE normally transmits DISC's at T1 intervals until a response is received.

TABLE 8-7

LAP MOD

FRAME	P/F	ARGUMENT
SARM	0	15
SARM	1	31
DISC	0	67
DISC	1	83
UA	0	99
UA	1	115

**53. AUTO RESPONSE:** Automatically transmits the proper response to the last packet received. These responses are defined in the PACKET SPECIFICATION Mode, item  in the X25/X75 MENU. No argument required.

**54. SKIP-IF-LINK-DOWN:** Skips the next instruction if the link is down. No argument required.

**55. SKIP-IF-LCN-MATCH:** Skips the next instruction if the current LCN matches the LCN of the last packet received. No argument required.

## PACKET LEVEL ACTIVE, PACKET SPECIFICATIONS

**56. SET-[STAT]:** Sets the status byte for the current LCN to any value from 0 through 255. If, for example, the channel is in the information transfer state, the status byte could be set to 1. During execution of the scenario then, the status byte for that channel or any channel could be examined and its status determined. The subroutine shown in the example checks the status of all 255 logical channels and sets them to 0. It is important to note that this instruction and all other bracketed instructions are addressed by the current LCN.

```
EXAMPLE:  SET-LCN.....0
          FOR.....255
          LCN-1
          SKIP-UNLESS-[STAT]=.....0
          GOTO-MARKER-#.....155
          SET-[STAT]=.....0
           MARKER.....155
          NEXT
```

**57. SKIP-UNLESS-[STAT]=:** Skip the next instruction if the status of the current LCN does not agree with the argument. The argument may be any integer from 0 to 255. An example of the instruction is shown above.

**58. NOT ALL LCN'S:** Used primarily in the Scenario Driven Monitor Mode to limit monitoring to the LCN specified in the SET-LCN instruction. All LCN's are monitored when this instruction is not used. No argument required.

**59. CLEAR-STATS:** Sets the status for LCN's 1 through 255 to 0. No argument required.

**60. MODULO-8:** Sets the packet numbering scheme to 0 through 7. After the 7th packet, the numbering sequence is repeated. Using this scheme, the maximum number of outstanding packets is 8. No argument required. This instruction is the default condition.

**61. MODULO-128:** Sets the packet numbering scheme to 0 through 127. After packet 127, the number sequence is repeated. Using this scheme, the maximum possible number of outstanding packets is 128. No argument required.

---

### 13. PACKET SPECIFICATIONS, EOT D

---

**13.01** The Packet Specification Mode allows the user to reconfigure the X25 packets as described earlier in this chapter or to specify the parameters for a new or previously undefined packet. When selected from the X25/X75 MENU, by striking the EOT  
D key, the ENCORE immediately displays the packet definition for the packet with a HEX ID of 01. Using the ← key at this point allows the user to scroll through the definitions of all existing packets. On power-up, the ENCORE normally defaults to the packet specifications shown in Table 8-8. The default specifications will be changed to agree with user modifications if the disc is not removed or not write protected when specifications are modified. Each column heading in the table corresponds with a packet specification parameter. These parameters are discussed in paragraphs 13.02 through 13.13.

TABLE 8-8

DEFAULT PACKET DEFINITIONS

PACKET TYPE NAME	HEX ID	RESPONSE	CLASS	MIN OCTETS	MAX OCTETS	ADDRESSES Y/N	VARIABLE LENGTH ADDRESSES Y/N	CALLING ADDRESS LENGTH	CALLED ADDRESS LENGTH	NO. FACILITY FIELDS	AUXILIARY INFORMATION
RR Notes 1,4	01	NONE	FINITE	03	03						
	03		ILLEGAL								
RNR Notes 2,4	05	NONE	FINITE	03	03						
	07		ILLEGAL								
REJ Notes 3,4	09	NONE	FINITE	03	03						
	CAL?	CAL!	FORMATTED			Y	Y	08	08	1	00000000
	0D		ILLEGAL								
	CAL!	0F	NONE	FINITE	03	03					
		11		ILLEGAL							
	CLR?	13	CLR!	FINITE	03	03					
		15		ILLEGAL							
	CLR!	17	NONE	FINITE	03	03					
		19		ILLEGAL							
	RSR?	1B	RSR!	FINITE	05	05					
		1D		ILLEGAL							
	RSR!	1F	NONE	FINITE	03	03					
	INT?	23	INT!	FINITE	04	04					
	INT!	27	NONE	FINITE	03	03					
		23 thru 55		ILLEGAL							
		57		IGNORE							
		59 thru EF		ILLEGAL							
		F1 thru F9		ILLEGAL							
	RST?	FB	RST!	FINITE	04	04					
		FD		ILLEGAL							
	RST!	FF	NONE	FINITE	03	03					

- NOTES:
- (1) Possible RR HEX ID's are 01, 21, 41, 61, 81, A1, C1, E1.
  - (2) Possible RNR HEX ID's are 05, 25, 45, 65, 85, A5, C5, E5.
  - (3) Possible REJ HEX ID's are 09, 29, 49, 69, 89, A9, C9, E9.
  - (4) When set for ID 01, parameters are automatically set for all other RR ID's.

**13.02 Packet Type Name.** This is a mnemonic identifying each of the X25 packets defined earlier in this chapter. The user may change this mnemonic when modifying a previously defined packet or name a new packet to be defined. The packet name is strictly a label for the user's convenience. It has no effect in detecting errors. Whenever an incoming packet ID (Octet 3) matches the ID of the ENCORE's defined packet, the packet type name assigned by the ENCORE will be the name shown in the monitor display.

**13.03 HEX ID.** The HEX or packet ID is contained in Octet 3. For most packet types, the ID is fixed, but for the RR, RNR, and REJ packets, the ID will change because bits 0, 1, and 2 contain the packet receive sequence number. The HEX ID will also change for DATA packets as Octet 3 includes the packet receive sequence number, the more bit, and the packet send sequence number.

**13.04 Response.** The response is simply the packet type expected in response to the packet shown in the first column. The response to a call request packet (CAL?), for example, is the call accept packet (CAL!).

**13.05 Class.** The ENCORE defines four packet classifications:

- (a) **Finite.** A finite packet contains a specified number of octets. If the number of octets in an RR, for example, is more or less than the number specified in the packet definition, an error is detected.
  - TFO = Too few octets
  - TMO = Too many octets
- (b) **Formatted.** A formatted packet contains a variable number of octets. A call request packet, for example, includes an address field, facility field, and user data field, each containing a number of octets that may change from transmission to transmission. During monitor operation, it is important that a formatted packet be properly classified or an octet error may result (TFO, TMO).
- (c) **Illegal.** Packets specified as "ILLEGAL" will produce an "IP" error on the display when logged.
- (d) **Ignore.** Packets classified as "IGNORE" do not cause an alarm condition when logged, but they are displayed.

**13.06 Minimum Octets.** This is a characteristic of a finite packet. When a number of octets is specified, an error will be detected if that number of octets is not received. RR, RNR, and REJ packets are comprised of three octets if modulo 8 or four octets if modulo 128.

**13.07 Maximum Octets.** This is also a characteristic of finite packets. It specifies the maximum number of octets that may comprise the packet. If this number is exceeded, an error is detected. RR, RNR, and REJ packets are comprised of three octets if modulo 8 or four octets if modulo 128.

**13.08 Addresses.** In a formatted packet, a YES or NO indicates whether or not the packet includes the DTE and DCE address.

**13.09 Variable Length Addresses.** In a formatted packet, a YES or NO indicates whether or not the packet includes variable length addresses.



**13.10 Calling Address Length.** In a formatted packet, the length of the calling address may be from 1 to 15 semi-octets. Each semi-octet is a binary coded decimal number (0-9).

**13.11 Called Address Length.** The characteristics of the called address length are the same as those discussed in the preceding paragraph.


**13.12 Number Facility Fields.** Up to three facility fields may be selected. The facility field is only present when DTE is using an optional user facility, i.e., reverse charging, high priority traffic, etc..

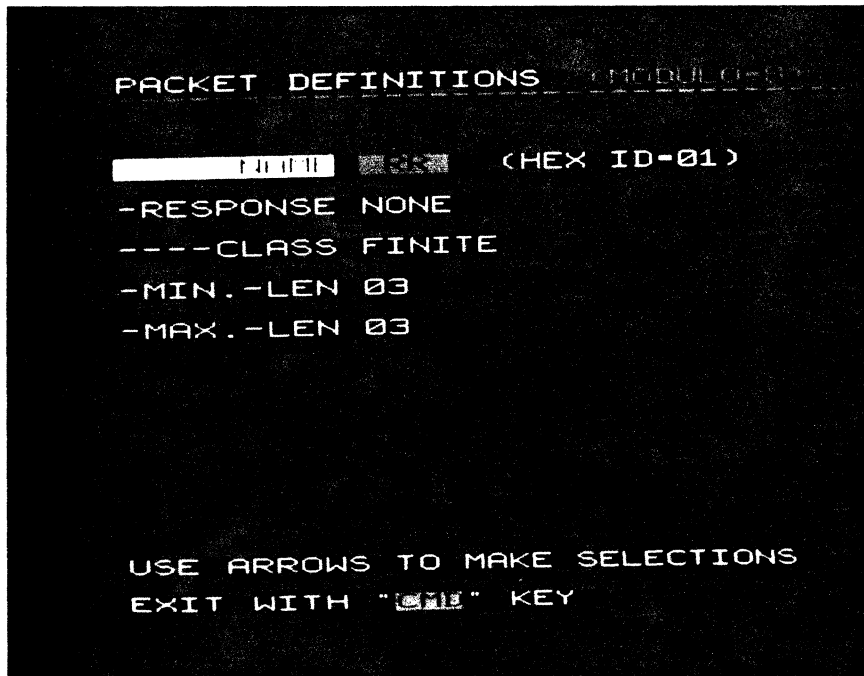
**13.13 Auxiliary Information.** Reserved for future use.

**TYPICAL PACKET SPECIFICATION OPERATING PROCEDURE**

**13.14** For the purpose of this procedure, it is assumed that the ENCORE is connected to the circuit under test and is powered-up in accordance with the Initial Operating Procedure given in paragraph 8.01. It is also assumed that the default packet specifications are adequate for successful monitoring of the packet switched data.

**TYPICAL PACKET SPECIFICATION OPERATING PROCEDURE**

STEP	PROCEDURE
1.	From the X25/X75 MENU select the Packet Specification Mode by striking the  key. Note that the ENCORE displays the finite RR packet definition as shown in Figure 8-48.



**Fig. 8-48 Finite Packet Definition**

TYPICAL PACKET SPECIFICATION OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
2.	Using the <input type="checkbox"/> and <input type="checkbox"/> keys, select the packet (by HEX ID) to be defined or modified and type a new name if required.
3.	Strike the <input type="checkbox"/> key and note that the word RESPONSE appears in reverse video. Although not required for monitor operation, the appropriate response (packet name) may be entered.
4.	Strike the <input type="checkbox"/> key and note that the word CLASS appears in reverse video. Use the <input type="checkbox"/> or <input type="checkbox"/> key to select one of the following packet classifications: <ul style="list-style-type: none"> <li>● FINITE</li> <li>● ILLEGAL</li> <li>● IGNORE</li> <li>● FORMATTED</li> </ul>
5.	Strike the <input type="checkbox"/> key and note that using the <input type="checkbox"/> and <input type="checkbox"/> keys: (a) The minimum number of octets permitted in a finite packet may be selected; (b) In a formatted packet, Figure 8-49, "YES" or "NO" may be selected to indicate that the packet includes both DTE and DCE addresses.

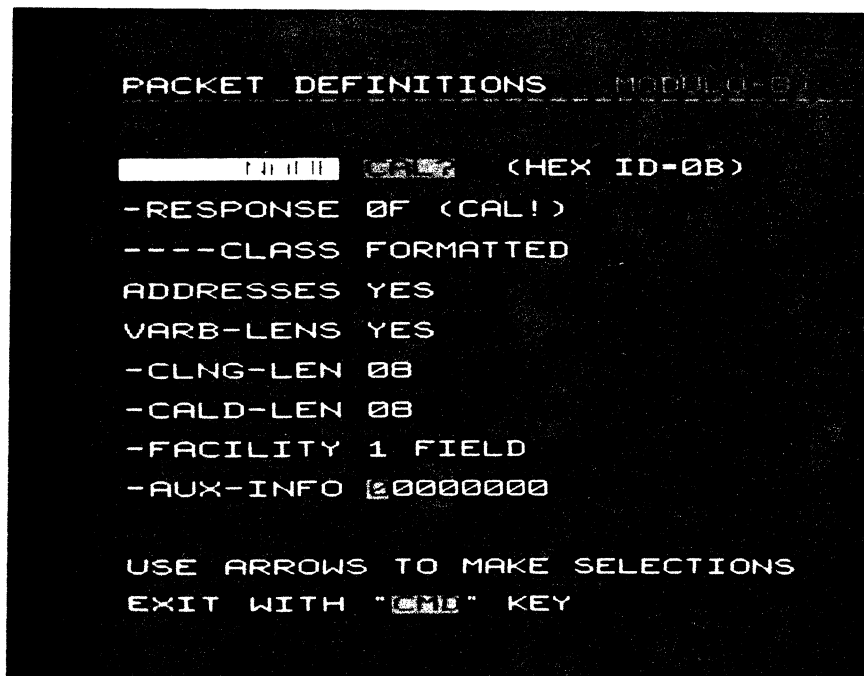


Fig. 8-49 Formatted Packet Definition

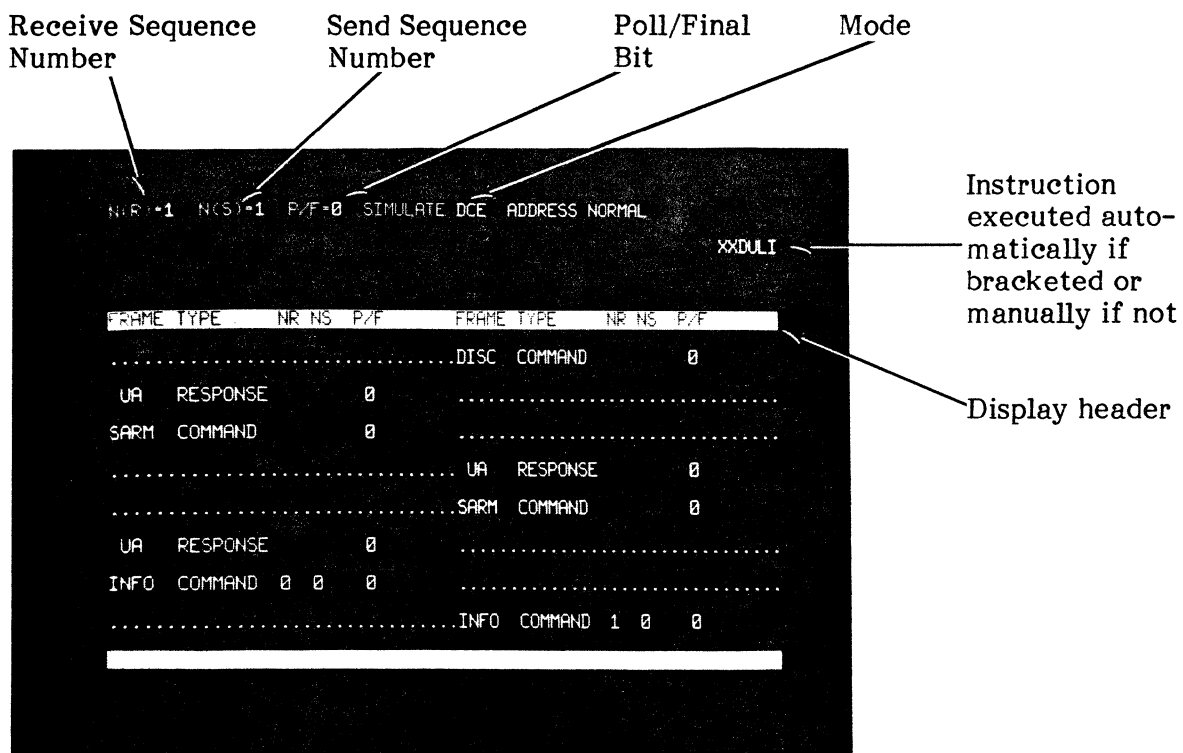
TYPICAL PACKET SPECIFICATION OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
6.	Strike the <input type="checkbox"/> key and note that using the <input type="checkbox"/> and <input type="checkbox"/> keys: (a) The maximum number of octets permitted in a finite packet may be selected; (b) In a formatted packet "YES" or "NO" may be selected to indicate that the packet includes variable length addresses.
7.	Strike the <input type="checkbox"/> key and note that using the <input type="checkbox"/> and <input type="checkbox"/> keys a calling address length of up to 15 semi-octets may be selected. For finite packets, no further definition is required. To continue defining a formatted packet, proceed with steps 8 through 10. To define another packet, simply select another HEX ID. Strike the <input type="checkbox"/> key to exit the Packet Specification Mode.
8.	If defining a formatted packet, strike the <input type="checkbox"/> key and note that the <input type="checkbox"/> and <input type="checkbox"/> keys may be used to select a called address length of up to 15 semi-octets.
9.	Strike the <input type="checkbox"/> and note that the <input type="checkbox"/> and <input type="checkbox"/> keys may be used to select up to three facility fields.
10.	The entry of auxiliary information is required for future use and is not discussed at this time. At this point, all parameters for a given packet are defined. Select another packet by returning to the NAME parameter. Return to the X25/X75 MENU by striking the <input type="checkbox"/> key.

This completes the Typical Packet Specification Operating Procedure

**14. LINK-TESTER,** ENQ  
E

**14.01** This mode of operation is designed for interactive testing at the link level. It employs its own unique set of instructions which can be executed manually, one at a time, or automatically in program form up to 62 at a time. Wherever an instruction is inappropriate for the simulate or bridging mode selected, it will be ignored. The LINK-TESTER display, Figure 8-50, is similar to other X25/X75 displays in that DTE information appears on the left side of the screen and DCE on the right. The top line of the display shows the current receive frame, send frame, and poll/final bit. It also indicates the simulate mode, as selected by striking the CAN  
X key, and the address state, normal or inverted. The second line of the display echoes the instruction entered manually or if enclosed within brackets, the series of instructions to be executed automatically by striking the COM1 key. The third line of the display is the frame level header used to identify the frame and depending upon its address, note whether it is a command or response (A or B address). The header also shows the N(R), N(S), and P/F for the displayed frame. A complete explanation of all LINK-TESTER instructions is included in Table 8-9. A display of the instructions is obtained by holding the SHIFT key down and striking the Z key. When the instructions are displayed, strike any key to return to the LINK-TESTER Mode.










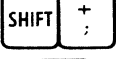






**Fig. 8-50 Link-Tester Display**

**TABLE 8-9**  
**LINK TEST INSTRUCTIONS**


KEY	FUNCTION	DESCRIPTION
SOH A	Address invert	Changes the address in the transmitted frame from A to B or B to A. The A address indicates a DCE command or DTE response. The B address represents a DTE command or a DCE response.
STX B	SABM transmit	Transmits an unnumbered Set Asynchronous Balanced Mode (SABM) command.
ETX C	CMDR transmit	Transmits an unnumbered Command Reject (CMDR) response.
EOT D	DISC transmit	Transmits an unnumbered Disconnect (DISC) command.
HT I	INFO transmit	Transmits an Information (INFO) frame which includes N(S), N(R), and P/F including data packet with user data = ABC.
LF J	REJ transmit	Transmits a Reject (REJ) supervisory response.
FF L	SARM transmit	Transmits an unnumbered Set Asynchronous Response Mode (SARM) command.
CR M	DM transmit	Transmits an unnumbered Disconnect Mode (DM) response.
SO N	RNR transmit	Transmits a Receive Not Ready (RNR) supervisory frame.
DLE P	P/F toggle	Changes the state of the current poll/final bit, 1 or 0.
DC2 R	RR transmit	Transmits a Receive Ready (RR) supervisory frame.
NAK U	UA transmit	Transmits an Unnumbered Acknowledge (UA) response.
ETB W	Wait ¼ sec	Creates a ¼ second delay in program execution during auto-run.
CAN X	DTE/DCE/BRG	Successively depressing the <span style="border: 1px solid black; padding: 2px;">CAN X</span> key changes the operating mode to simulate DTE, simulate DCE, or passively bridge the interface.
SUB Z	N(S) + N(R) = 0	Sets the frame send and receive sequence numbers to 0.
∅	Set N(R) = 0	Sets the frame receive sequence number to 0.

TABLE 8-9


## LINK TEST INSTRUCTIONS (Cont'd)

KEY	FUNCTION	DESCRIPTION
	Set N(R) = 1	Sets the frame receive sequence number to 1.
	Set N(R) = 2	Sets the frame receive sequence number to 2.
	Set N(R) = 3	Sets the frame receive sequence number to 3.
	Set N(R) = 4	Sets the frame receive sequence number to 4.
	Set N(R) = 5	Sets the frame receive sequence number to 5.
	Set N(R) = 6	Sets the frame receive sequence number to 6.
	Set N(R) = 7	Sets the frame receive sequence number to 7.
	N(S) + 1	Increments the frame send sequence number by 1.
	N(S) - 1	Decrements the frame send sequence number by 1.
	Open string	Permits user entry of up to 62 instructions for serial execution (left to right) during auto-run.
<b>SPACE</b>	Character delete	Deletes the last character shown in the auto-run string.
	Auto-run	Begins execution of the instructions entered in the auto-run string.
	Auto-run stop	Halts further execution of the instruction entered into the auto-run string. Depressing the  key a second time clears the string from the display and returns the user to the manual mode. Striking the  key returns the string (manual).

**15. IO MODE,** 

**15.01** The IO Mode offered as item  in the X25/X75 MENU is the same mode of operation selected earlier while in the Real Time Monitor or Fast Capture Modes. It is described in paragraphs 10.31 through 10.39 of this chapter.

**16. SET DATE AND TIME,** 

**16.01** Selecting this item from the X25/X75 MENU allows the user to establish the Julian date and time which may be used to time stamp captured data. Selecting item  from the menu instructs the ENCORE to display the date/time prompts as shown in the following procedure.

User entry of the time and date is described in this procedure.

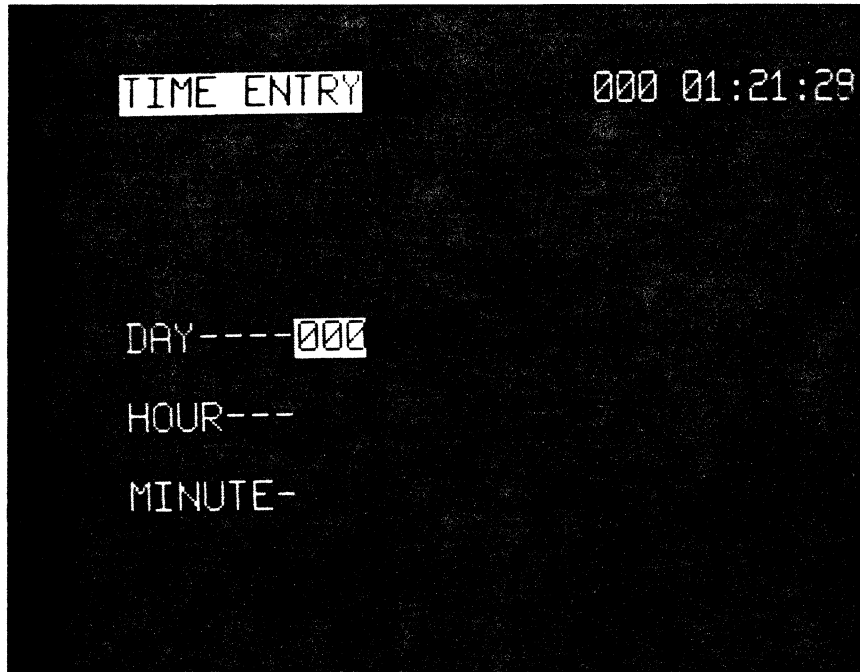
### TYPICAL DATE AND TIME OPERATING PROCEDURE

**16.02** The date/time group begins counting from zero upon completion of the self-test procedure during the power-up sequence. To set or change the display, proceed as follows:

**NOTE:** The  and  keys are used to move the cursor to another line for changing values.

### TYPICAL DATE AND TIME OPERATING PROCEDURE

STEP	PROCEDURE
1.	From the X25/X75 MENU, select "SET DATE-TIME" by striking the <input type="text" value="DEL"/> key and note a display similar to that shown in Figure 8-51.



**Fig. 8-51 Time Entry Format**

2. The Julian date is the first entry required. It is made by simply typing the appropriate date (1 through 365) and striking the  key.
3. The hour is entered in the same manner by typing the appropriate number (1 through 23) and striking the  key.

## SET DATE AND TIME, PRINTER IO MODE, LEVEL-3 COMMAND MODE

### TYPICAL DATE AND TIME OPERATING PROCEDURE (Cont'd)

STEP	PROCEDURE
4.	The minute is the last entry and it is also made by typing the appropriate number (0 through 59) and striking the <input type="button" value="END"/> key. When this entry is completed, the ENCORE will return to the X25/X75 MENU.

This completes the Set Date and Time Operating Procedure

---

#### 17. PRINTER IO MODE,

17.01 The PRINTER IO MODE offered as item  in the X25/X75 MENU is the same mode of operation selected while in the monitor and interactive modes. It is described in paragraph 10.41.

---

#### 18. LEVEL-3 COMMAND MODE,

18.01 The selection of item  from the X25/X75 MENU instructs the ENCORE to enter the Level-3 Command Mode. This provides the user with an escape from the X25 Operating System into the Command Mode where he may operate the ENCORE in any of its three levels of operation as detailed in Chapters 5 through 7 of this manual. This item is offered in both the X25 MENU and in the PASSIVE MONITOR MENU, primarily as a convenience for the user.





## CHAPTER 9 OPERATOR MAINTENANCE

### 1. GENERAL

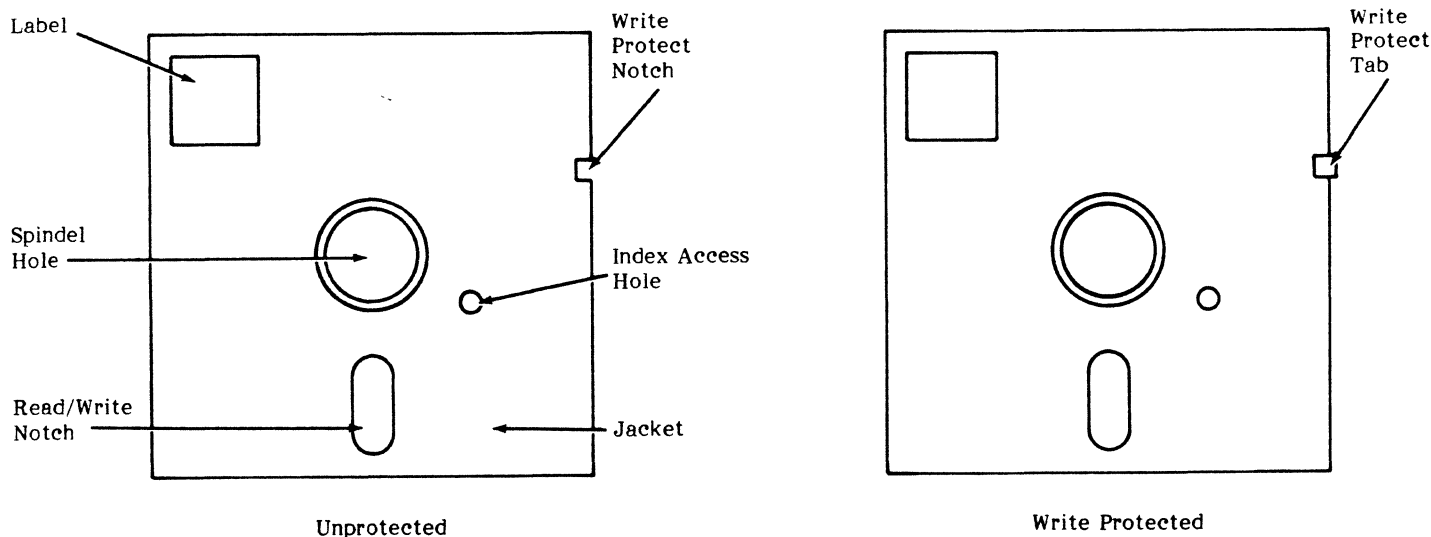
**1.01** This chapter of the manual includes information for handling and cleaning of the diskette and some tips on troubleshooting the ENCORE at the operator's level. If you suspect that the ENCORE may require factory service, contact DIGITECH for return authorization.

### 2. THE DISKETTE

**2.01** The flexible disc is a circular plastic sheet coated on both sides with a layer of ferromagnetic material. It has a large spindle hole to accommodate the disc drive hub and a small hole, used to index the disc as it rotates. A blank disc (new or erased) contains no information and cannot be used to store data or programs until it is formatted. A formatted disc is organized into numbered concentric "tracks" or sectors which are used to store programs, data, and a small amount of system bookkeeping information.

**2.02** Each disc (Figure 9-1) is supplied in a protective jacket to prevent damage as a result of handling. The jacket lining is treated to clean the disc as it rotates. A "write protect notch" is cut into each disc jacket. When the notch is uncovered, data may be written to the disc. When covered, with the write protect tab, the disc is protected and data can only be read from the disc and not written to it. Information is recorded on both sides of the disc providing for up to 192K bytes of program storage or up to 240K bytes of data storage. The amount of memory available for captured data depends on the amount of program memory used. The less memory used for programs, the more available for captured data.

**NOTE:** If new programs are to be stored after data has been captured, the new programs will overwrite the captured data.



**Fig. 9-1 Typical Diskette**

**DISC HANDLING**

**2.03** The disc is a precision recording media normally containing vital information. It must therefore be handled with reasonable care to insure long life and trouble-free operation. Some suggestions for care of the disc are listed below.

- Do not write on the disc with a hard point device such as a ball point pen or lead pencil. Use a felt tip pen.
- Do not fasten paper clips to the disc jacket.
- Handle the disc by the jacket only, avoid touching exposed surfaces.
- Keep the disc away from magnetic fields and from materials that may be magnetized.
- Protect the disc from dust, cigarette ashes, etc..
- Do not leave the disc in the drive when the ENCORE is unattended.
- Do not turn the ENCORE on or off with a disc in the drive.
- Keep the disc in its protective envelop when not in use.
- Do not exceed the following environmental storage conditions:

Temperature: 50 to 125 F  
10 to 51 C

Humidity: 8 to 80%

**CLEANING**

**2.04** Clean the disc drive read/write heads using the Scotch Head Cleaning Diskette, catalogue number 7440, or equivalent. Clean the heads weekly if the drives are in moderate use or daily for heavy use.

**3. ERROR MESSAGES**

**3.01** When the ENCORE encounters an error in a self test routine or operator input, an error code is displayed on the CRT. This error message points to the offending program line number (i.e., SYNTAX ERROR @ 100) or problem area within the operating system. A complete list of error codes is provided in Table 9-1. Additional error messages may appear on the CRT immediately following the power-up self test routine (i.e., SELF TEST FAILURE #1). If such a message appears, contact DIGITECH for authorization to return the unit for service.

**TABLE 9-1  
ERROR CODES**

<b>ERROR CODE</b>	<b>ERROR MESSAGE</b>	<b>PROBABLE CAUSE</b>
00	NO ERROR	
01	ILLEGAL LINE NUMBER	An attempt has been made to transfer program control to a nonexistent line number
02	MISSING LINE NUMBER	A program statement is encountered with no line number.
03	BAD LOAD PARITY	A parity error has occurred during an attempt to load a program in the EDIT Mode.
04	LINE NUMBER > 9999	During an attempt to renumber a program in the EDIT Mode, line numbers have exceeded 9999.
05	BAD DISC	In an attempt to read information from the disc, the ENCORE has detected a bad sector, incorrect format, or data error.
06	DISC FILE OPEN	Not used.
07	OBJECT BUFFER TOO SMALL	An attempt to execute the RUN command has been made, but the object buffer is too small to accommodate the compiled program. Use RUN! to automatically allocate the required memory.
08	STACK OVERFLOW	An attempt to execute a GOSUB has been made twice and the required RETURN has not been encountered.

**TABLE 9-1**  
**ERROR CODES (Cont'd)**

ERROR CODE	ERROR MESSAGE	PROBABLE CAUSE
09	STACK UNDERFLOW	An attempt to execute a RETURN has been made before encountering a GOSUB.
10	SYNTAX ERROR	An attempt to execute a COMBASIC statement has been made, but the statement has been entered improperly.
11	SOURCE BUFFER TOO SMALL	An attempt to transfer a program from nonvolatile memory, or disc, has been made, but the source buffer is too small to accomodate the program. Enter the Memory Allocation Mode, <span style="border: 1px solid black; padding: 2px;">VE STAT</span> , and increase the size of the source buffer.
12	BAD CHAIN	An attempt to execute the CHAIN instruction has been made, but the argument program is not in memory.
13	ILLEGAL NEXT	An attempt to execute a NEXT instruction has been made before encountering the required FOR instruction.
14	NUMBER TOO LARGE	The results of arithmetic operations produce a number larger than that permitted for the specified integer or floating point variable.
15	ILLEGAL FOR	An attempt to execute a FOR instruction has been made, but the required NEXT is not in the program.
16	DISC ERROR	An attempt to read a disc has been unsuccessful, indicating that the wrong disc is in the drive, or that the disc is faulty.
17	PROGRAM WON'T FIT	An attempt to load and compile a program has been unsuccessful because both source and object buffers are too small to accomodate the selected program. Automatic allocation to both buffers can be made by typing # program name !.

**TABLE 9-1**  
**ERROR CODES (Cont'd)**

<b>ERROR CODE</b>	<b>ERROR MESSAGE</b>	<b>PROBABLE CAUSE</b>
18	NO SUCH DISC FILE	An attempt to load a program from disc has been made, but the specified program is not on the disc.
19	CALL OVERFLOW	More than one CALL instruction has been encountered in a single program or more than 8 programs have been called.
20	BAD CALL	A CALL instruction has been executed, but the called program is not in memory.
21	RCV RAM TOO SMALL	An attempt to transfer a front end program from the object buffer to the front end has been made, but the receiver RAM is too small to accommodate the program.
22	SKIP	An attempt to execute the front end instruction DSZ or DNA has been made, but the required SET instruction is not in the program.
23	NO PROGRAM	An attempt to execute a front end program has been made, but no host program exists.
24	STRING ERROR	A string variable has been improperly dimensioned or the variable has been entered incorrectly.
25	DOUBLE DIM	A string variable has been dimensioned more than once in a single program.
26	INPUT ERROR	An attempt to execute a WHEN P2a or WHEN P3a statement has been made, but the required pin has not been turned on.
27	DATA ERROR	An attempt to execute a READ statement has been made, but the required DATA statement does not exist or the data pointer has past the last data item.

**TABLE 9-1**  
**ERROR CODES (Cont'd)**

<b>ERROR CODE</b>	<b>ERROR MESSAGE</b>	<b>PROBABLE CAUSE</b>
28	WHEN OVERFLOW	An attempt to execute more than four WHEN P2 or WHEN P3 statements has been made at one time.
29	TOO MANY LINES	A COMBASIC program, in the source buffer, contains more than 350 lines.
30	TOO MANY VECTORS	A COMBASIC program, in the source buffer, contains more than 128 GOTO or GOSUB instructions.
31	WRITE PROTECT	An attempt to record on disc has been made, but the write protect tab is in the protect position.

**4. TROUBLESHOOTING TIPS**

**4.01** When trouble first develops as a result of trying to enter a command or execute a program and the error codes offer no solution, use the troubleshooting tips given below.

**TABLE 9-2**  
**TROUBLESHOOTING TIPS**

<b>SYMPTOM</b>	<b>PROBABLE CAUSE</b>	<b>CORRECTIVE ACTION</b>
Known good program does not run.	1) Incorrect IO parameters. 2) Network configuration changed.	1) Return to IO Mode and check for correct parameters.
Newly designed program does not run.	1) Incorrect IO parameters. 2) Program design error.	1) Return to IO Mode and check for correct parameters. 2) Enter EDIT Mode and type in STOP instruction at the key position within the program. Run the program and check operation to the point where program execution stops.

# TROUBLESHOOTING TIPS, APPLICATIONS ASSISTANCE, TRAINING

TABLE 9-2

TROUBLESHOOTING TIPS (Cont'd)

SYMPTOM	PROBABLE CAUSE	CORRECTIVE ACTION
Improper response to keyboard entries.	1) LOCK key depressed on main array.	1) Release LOCK key and reenter command.

## 5. APPLICATIONS ASSISTANCE

**5.01** Applications assistance is one of the most important facets of DIGITECH's operation. It begins during the instrument design phase; applications engineers work closely with the design, quality control, and manufacturing departments to assure that every instrument and special applications package meets all user requirements.

**5.02** When you need applications assistance, the quickest most efficient source of information is our applications engineer. He is as near as your telephone. He will provide you with sound technical information and accurate, concise answers to your questions. For applications assistance, call DIGITECH at (203) 438-3731 and ask for our Applications Department.

## 6. TRAINING

**6.01** The field of data communications presents a variety of problems that need to be identified and solved. The personnel performing this task must understand the nature of the problems and how to solve them. DIGITECH's training classes provide assistance through:

- o An overview of communication principles.
- o Discussion of trouble-shooting techniques.
- o Hands on experience.
- o On-line application simulations.
- o Individualized instruction.

**6.02** These seminars coupled with the capability of DIGITECH's PACER-103 and ENCORE 100 and ENCORE 200, provide the user unlimited flexibility in a variety of problem solving situations.

### PACER-103 INTRODUCTION

**6.03** This course prepares the student to be a proficient troubleshooter with the PACER-103. Data communications and troubleshooting techniques are discussed in detail. Each student receives hands on utilization of the PACER-103. Programming in our DICOL language is taught. Many typical problems are isolated while the student learns to operate the PACER-103.

### ENCORE INTRODUCTION

**6.04** This course begins with a discussion of data communications principles. Next, the physical characteristics of the ENCORE are explained in detail. Troubleshooting techniques



are taught via our Level-1 diagnostic system. Level-1 provides menu access to a library of diagnostic programs. On-line tests and analysis of results are performed by each student. This course prepares customers to troubleshoot efficiently with the ENCORE.

**ENCORE ADVANCED PROGRAMMING**

**6.05** The programming language of the ENCORE is called COMBASIC and the focus of this course is COMBASIC programming. Also included is a discussion of the front end processors and their function within the system. All the programmable aspects of the ENCORE are presented in a logical progression. The main objective is the preparation of each student to be able to develop customized software for the ENCORE.

**6.06** If you would like to take advantage of our normally scheduled training sessions or would like to schedule on-site training, please call DIGITECH at (203) 438-3731 and ask for our Training Department.

**APPENDIX A**  
**SPECIFICATION SUMMARY**

Underlying equipment specifications are listed in this appendix. The specifications include physical characteristics, overall operating parameters, and power requirements.

SPECIFICATION SUMMARY

ITEM	CHARACTERISTICS
<u>Network Interface</u>	RS-232 interface for use with the circuit under test. Adapters available for V.35/DDS, 300 series dataset, EIA RS-449, 20/60 ma neutral current.  <b>Note:</b> For detailed specifications, see Chapter 3.
<u>Output Ports</u>	
RS-375	BNC composite interlaced video output for use with remote monitor.
Scope Sync	BNC output port for scope sync output under program control. The output pulse measures 0 to +5 volts nominal and is 10 microseconds in duration.
XIO Port	RS-232 port for off-line load/dump. Output language is determined by IO language selection, XIO bit selection, and language of captured data. The XIO Port looks like an asynchronous modem.
<u>Audible Alert</u>	A programmed alarm (1000 Hz) of no less than 0.2 seconds in duration alerts the operator that some programmed event has occurred.
<u>Program Memory</u>	32K nonvolatile battery supported.
<u>Disc</u>	ANSI standard 5¼ inch disc, dual density, double sided with the ability to record data and status bytes at speeds up to 9 600 bps FDUX.
Tracks	68 tracks, 34 per side.
Life	3 x 10 <sup>6</sup> passes on a single track.
Capacity	240 K bytes storage.
<u>Clock Source</u>	Internal or external (conventional or NRZI).

## SPECIFICATION SUMMARY (Cont'd)

ITEM	CHARACTERISTICS																
<u>Speeds</u>																	
IO	Frequency synthesizer provides for selection at any speed from 10.0 to 99.9 Kbps with three significant digits being the maximum resolution. Programs can be run at speeds of up to 64 Kbps FDUX by deleting the status byte from the captured data (see STATE PIPE). Accuracy is <u>+0.005%</u> .																
XIO	Any one of the following 16 speeds, accurate to <u>+0.005%</u> :  <table data-bbox="824 863 1382 982"> <tr> <td>50</td> <td>150</td> <td>1 800</td> <td>4 800</td> </tr> <tr> <td>75</td> <td>300</td> <td>2 000</td> <td>7 200</td> </tr> <tr> <td>110</td> <td>600</td> <td>2 400</td> <td>9 600</td> </tr> <tr> <td>134.5</td> <td>1 200</td> <td>3 600</td> <td>19 200</td> </tr> </table>	50	150	1 800	4 800	75	300	2 000	7 200	110	600	2 400	9 600	134.5	1 200	3 600	19 200
50	150	1 800	4 800														
75	300	2 000	7 200														
110	600	2 400	9 600														
134.5	1 200	3 600	19 200														
<u>Timers and Counters</u>																	
Up to 26 programmed controlled measurement timers count from 0 to 32 767 milliseconds in 1 millisecond intervals. Up to 78 additional counters and 27 triggers.																	
<u>Code Levels</u>																	
Standard	5, 6, 7, and 8 level including parity.																
<u>Parity</u>																	
Odd, even, none, ignore, always mark (in transmit), always space (in transmit).																	
<u>Languages</u>																	
Standard	ASCII, BCD, EBCDIC, Selectric, and External BCD.																
Optional	Any 3 others.																
<u>Real Time Clock</u>																	
Level-3 Time Mode provides for operator entry and edit of Julian date and time.																	
<u>Keyboard</u>																	
76-key full ASCII keyboard with special function and cursor controls. Up to 26 keys may be assigned to user defined functions.																	

SPECIFICATION SUMMARY (Cont'd)

ITEM	CHARACTERISTICS
<u>Displays</u>	
CRT	<p>The CRT measures 7 inches diagonally and displays characters in an 8x16 dot matrix with normal/reverse video; single/dual line displays; half/full duplex; 224 to 896 characters per screen, selectable in four formats:</p> <p style="padding-left: 40px;">32 characters on 7 lines          64 characters on 7 lines          32 characters on 14 lines          64 characters on 14 lines</p>
RS-232	<p>24-LED display for real time status of the interface.</p>
<u>Transmission Modes</u>	
Asynchronous, synchronous, isochronous.	
<u>Asynchronous Stop Length</u>	
1.0, 1.5, 2.0 units (1.5 with INT clock only).	
<u>Protocols</u>	
Standard	<p>SDLC, SDLC/NRZI, HDLC, ADCCP.          Bisync (normal and transparent), X.25/X.75.</p>
<u>Sync Scheme</u>	
<p>Will sync on any two characters. Applies to all patterns from 10-16 bits.</p>	
<u>Drop Sync Scheme</u>	
<p>Will drop sync under program control, or optionally on loss of COD (pin 8) for data received on pin 3 only.</p>	
<u>Block Check Schemes</u>	
Standard	<p><math>LRC_1 = X^8 + 1</math> (parity corrected; reset to zero).          Also for 7-bit characters (<math>X^7 + 1</math>).</p> <p><math>LRC_2 = X^8 + 1</math> (nonparity corrected; reset to zero).          Also for 7-bit characters (<math>X^7 + 1</math>).</p> <p><math>CRC-16 = X^{16} + X^{15} + X^2 + 1</math> (reset to zero).</p>

## SPECIFICATION SUMMARY (Cont'd)

ITEM	CHARACTERISTICS
<u>Block Check Schemes</u>	
Standard (Cont'd)	CRC-12 = $X^{12} + X^{11} + X^3 + X^2 + X + 1$ (reset to zero).
	CCITT CRC-16 = $X^{16} + X^{12} + X^5 + 1$ (preset to all ones).
Optional	IPARS CRC = $X^6 + X^5 + 1$ (reset to zero).
<u>Primary Power Source</u>	
Voltage	95-130 or 190-260 Vac
Frequency	47 Hz to 440 Hz
Power	100 watts maximum.
<u>Dimensions</u>	
Height	7.8 inches (198.120 millimeters).
Width	17.9 inches (454.660 millimeters).
Depth	18.1 inches (459.740 millimeters).
<u>Weight</u>	34.5 lb (15.6 kilograms).
<u>Temperature Range</u>	
Operating	+5 to 50 C
Storage	-55 to 75 C
<u>Humidity</u>	20 to 80% relative humidity at 50 C (no condensation).



**APPENDIX B**

**DICOL**



DICOL SUMMARY

<u>INSTRUCTION</u>	<u>OPERATION</u>
00	SELECT TEST MODE
01	STORE I/O PARAMETERS
02	SELECT BCS TYPE
03	START BCS
04	JUMP
05	RETURN TO MODE SELECT
06	SUBROUTINE CALL
07	RESET LOGGING TO BEGINNING OF BUFFER
08	RETURN TO MODE SELECT ON BUFFER FULL
09	RETURN FROM SUBROUTINE
10	TURN ON BOTH RECEIVERS
11	TURN OFF RECEIVER 2 OR 3
12	TRAP SEQUENCE
13	RECEIVE AND LOG THROUGH SEQUENCE
14	RECEIVE AND LOG "n" CHARACTERS
15	RECEIVE AND LOG THROUGH ANY ONE OF UP TO FIVE CHARACTERS
16	CRC COMPUTATION IN TRANSPARENT BCS
17	RECEIVE NEXT SEQUENCE AS BCS AND SKIP NEXT INSTRUCTION IF VALID
18	SKIP NEXT INSTRUCTION UNLESS ARGUMENT EQUALS LAST LOGGED CHARACTER(S)
19	RECEIVE AND LOG ANY CHARACTERS EXCEPT ARGUMENT
20	TRANSMIT CHARACTER(S)
21	TRANSMIT BCS CHARACTER(S)
22	TRANSMIT CHARACTERS, RECEIVE AND LOG SIMULTANEOUSLY
23	CHARACTER ERROR RATE TEST, SKIP ON ERROR
24	ENTER CONVERSATION MODE, EXIT ON ARGUMENT
25	BLOCK ERROR RATE TEST, SKIP ON ERROR
26	DISPLAY CONTENTS OF BLOCK COUNTER
27	SDLC FRAME CHAIN (OPTION)
28	SDLC FRAME TRANSMIT (OPTION)
29	SDLC FRAME TRANSMIT CHARACTERS, RECEIVE AND LOG SIMULTANEOUSLY (OPTION)
30	CLEAR DISPLAY
31	DISPLAY CHARACTERS
32	CLEAR PARITY ERROR
33	SKIP NEXT INSTRUCTION ON PARITY ERROR
34	TURN ON AUDIO
35	TURN OFF AUDIO
36	TURN ON DATA DISPLAY
37	TURN OFF DATA DISPLAY
38	ARGUMENT CHAIN
39	Reserved for future use
40	TURN OFF ALL CONTROLLED RS-232 LEADS (4, 5, 8, T, M, AND 21) EXCEPT DTE (20) AND DSR (6)

## DICOL SUMMARY (Cont'd)

<u>INSTRUCTION</u>	<u>OPERATION</u>
41	TURN ON RS-232 LEAD
42	TURN OFF RS-232 LEAD
43	SKIP NEXT INSTRUCTION IF RS-232 LEAD ___ IS ON
44	SKIP NEXT INSTRUCTION IF RS-232 LEAD ___ IS OFF
45	SET COUNTER "C" (OPTION)
46	SET COUNTER "D" (OPTION)
47	Reserved for future use
48	DISPLAY HALF-DUPLEX (OPTION)
49	DISPLAY FULL-DUPLEX (OPTION)
50	WAIT ___ MILLISECONDS
51	SET COUNTER A
52	SET COUNTER B
53	ADD ONE COUNT TO COUNTER
54	SUBTRACT ONE COUNT FROM COUNTER
55	DISPLAY CONTENTS OF COUNTER
56	SKIP NEXT INSTRUCTION IF COUNTER EQUALS "00"
57	CLEAR TIMER
58	SET TIMER AND SKIP NEXT INSTRUCTION, ADVANCE TO NEXT INSTRUCTION WHEN TIMER RUNS OUT
59	DISPLAY TIMER
60	Reserved for future use
61	OFF-LINE PROGRAM LOAD (OPTION)
62	Reserved for future use
63	PROGRAM IDENTIFIER



**APPENDIX C**  
**COMBASIC INSTRUCTION SUMMARY**  
**AND**  
**EXECUTION TIMES**



**COMBASIC SUMMARY**  
**INSTRUCTIONS/ARGUMENTS**

## COMBASIC1

```

10 REM COMBASIC INSTRUCTIONS
20 REM COMBASIC1 21 AUG 81 LH
30 REM COMBASIC1 AND COMBASIC2 ARE A SERIES OF NOTES
40 REM . ATTEMPTING TO EXPLAIN COMBASIC SYNTAX WITH
50 REM . SOME EXAMPLES. THE INSTRUCTIONS ARE LISTED IN
60 REM . ALPHABETIC ORDER TO FACILITATE REFERENCE.
70 REM . THESE NOTES ARE STORED ON THE PROGRAM
80 REM . DISKETTE AS COMBASIC1 AND COMBASIC2. ALL STATEMENTS
90 REM . ARE REMARKS, THEREFORE TO EXECUTE AN INSTRUCTION
100 REM . WHICH IS LISTED WITHIN DO NOT KEY IN "REM".
110 REM WHERE YOU SEE /CR/ OR /SH/ ETC THIS IS REALLY A
120 REM . CONTROL CHARACTER THAT OUR PRINTER CANNOT
130 REM . PRINT. TO ENTER PRESS 'CONTROL' AND THE
140 REM . CHARACTER WANTED.
150 REM LOWER CASE LETTERS ARE BYTE VARIABLES.
160 REM UPPER CASE LETTERS ARE INTEGER VARIABLES.
170 REM
180 REM ABORT
190 REM -----
200 REM USED FOR ABORTING "WHEN" INSTRUCTIONS
210 REM -----
220 REM ABORT TIMER - ABORT FULL - ABORT PIN
230 REM ABORT P2 - ABORT KBD - (PINS 4, 5, 6, 8, 11, A
240 REM ABORT P3 - ABORT XIOIN - 18, B, 20, 22)
250 REM ABORT RTC - ABORT ALL
260 REM
270 REM -----
280 REM BOOT
290 REM -----
300 REM . USED TO EXECUTE A PROGRAM
310 REM . ALLOWS YOU TO BRING IN A PROGRAM FROM DISC
320 REM . ALLOWS YOU TO SAVE A PROGRAM IN STORAGE
330 REM
340 REM IF "PROGRAM-1" GOTO 36
350 REM . IF PROGRAM-1 ALREADY IN STORAGE DON'T BOOT
360 REM . BOOT "@T PROGRAM-1\CR\@SAVE\CR\@"
370 REM
380 REM -----
390 REM CALL
400 REM -----
410 REM CALL "PROGRAM-1"

```

## COMBASIC SUMMARY

## INSTRUCTIONS/ARGUMENTS (Cont'd)

```

420 REM . PRIMARILY USED FOR FRONT END PROGRAMS
430 REM . MOVES SOURCE FROM STORAGE TO SOURCE BUFFER
440 REM . RECOGNIZES FRONT END INSTRUCTIONS
450 REM . COMPILES TO OBJECT BUFFER
460 REM . SHIPS TO FRONT END RECEIVERS
470 REM . COMPILES REMAINDER OF COMBASIC PROGRAM
480 REM . PUTS COMBASIC PORTION AT BEGINNING OF
490 REM . OBJECT BUFFER
500 REM
510 REM -----
520 REM CBUF CAPTURE BUFFER
530 REM -----
540 REM . LET A$(A,B)=CBUFB(A,B)
550 REM . LET A$(A,B)=CBUFE(A,B)
560 REM . A=STARTING POSITION,B=ENDING POSITION
570 REM NOTE: IF USING DISC (STATE DISCON) OR (DISP D )
580 REM . THE CAPTURE BUFFER IS AUTOMATICALLY
590 REM . ALLOCATED TO 4096 POSITIONS (4K).
600 REM . CBUF CAN STORE:
610 REM . DATA ONLY - STATE NSTAT,FAST
620 REM . STATUS & DATA - STATE STAT,NFAST
630 REM . STATUS & DATA - IS THE DEFAULT
640 REM . ATTRIBUTE & DATA - STATE NSTAT,NFAST
650 REM . CBUF CONTENTS CAN BE DISPLAYED USING THE
660 REM . DISP INSTRUCTION
670 REM
680 REM ----
690 REM CHAIN
700 REM ----
710 REM . CHAIN ""
720 REM . CHAINS TO LEVEL-3 COMMAND MODE IF PROGRAM NOT
730 REM . CHAINED INTO
740 REM . IF PROGRAM CHAINED INTO IT WILL CAUSE RETURN
750 REM . TO PREVIOUSLY CHAINED PROGRAM
760 REM . CHAINS CAN BE NESTED TO 4 LEVELS
770 REM . CHAIN STACK CAN BE CLEARED BY A CLEAR CHAIN
780 REM . INSTRUCTION
790 REM CHAIN "PROGRAM-1"
800 REM -----
810 REM CLEAR
820 REM -----
830 REM CLEAR CBUF - SETS CBUF POINTER TO POSITION 0
840 REM CLEAR MSTM - CLEARS MEASUREMENT TIMER TO 0,
850 REM . STARTS MEASUREMENT TIMER
860 REM MSTM MAX VALUE 8,388,608MS (2 HRS, 19 MIN, 48 SEC)

```

**COMBASIC SUMMARY**  
**INSTRUCTIONS/ARGUMENTS (Cont'd)**

```

870 REM CLEAR RETURN - CLEARS GOSUB/RETURN STACK
880 REM
890 REM -----
900 REM DISP
910 REM -----
920 REM . NOTE: DISP 'D' INSTRUCTIONS ARE FOR USE WITH
930 REM . DISC UNITS ONLY
940 REM . DISP 'T' INSTRUCTIONS ARE FOR USE WITH
950 REM . TAPE UNITS ONLY AND ARE NOT LISTED
960 REM . IN THIS NARRATIVE.
970 REM DISP DBG - POSITIONS DISC TO BEGINNING
980 REM . . . . o = 0 = NOT OVERWRITTEN & NO DATA LOST
990 REM . . . . o = 1 = OVERWRITTEN & NO DATA LOST
1000 REM . . . . o = 2 = NOT OVERWRITTEN & DATA LOST
1010 REM . . . . o = 3 = OVERWRITTEN & DATA LOST
1020 REM . . . . DATA LOST MEANS WHILE CAPTURING DATA
1030 REM . . . . ON DISC, DISC GOT BEHIND AND DATA WAS
1040 REM . . . . NOT RECORDED ON DISC.
1050 REM -----
1060 REM DISP DR - READS ONE BLOCK OF DATA (1K), SHIFTS
1070 REM . . . CBUF UP 1K , PUTS RECORD READ INTO 2ND K
1080 REM . . . . e <> 0 IF END OF DATA REACHED
1090 REM . . . . e = 0 IF END OF DATA NOT REACHED
1100 REM . . . . r <> 0 IF READ ERROR
1110 REM . . . . r = 0 IF NO READ ERROR
1120 REM DISP DBS - BACKS UP 3 RECORDS, READS 2
1130 REM . . . . e <> 0 IF BEGINNING OF DATA REACHED
1140 REM . . . . e = 0 IF BEGINNING OF DATA NOT REACHED
1150 REM DISP DE - POSITIONS DISC TO END
1160 REM DISP B - BACK UP 1 POSITION (LET A=A-1)
1170 REM DISP C - CONVERT d TO ASCII (NO PARITY)
1180 REM DISP D - DISPLAY DATA IN d
1190 REM DISP E - SET A TO PHYSICAL ENDING CBUF ADDRESS
1200 REM . (A+1)=NUMBER OF CHARACTERS
1210 REM DISP F - SET BYTE VARIABLE o TO
1220 REM . . . . . <> 0 IF CBUF OVERWRITTEN
1230 REM . . . . . = 0 IF CBUF NOT OVERWRITTEN
1240 REM DISP G - GET A BYTE FROM CBUF, INTEGER VARIABLE A IS
1250 REM . THE POSITION, START FROM 0, (LET A=0).
1260 REM . DATA PUT IN BYTE VARIABLE d; STATUS
1270 REM . OR ATTRIBUTE PUT IN BYTE VARIABLE s.
1280 REM DISP H - DISPLAY IN HEX
1290 REM DISP L - BIT SHIFT 1 BIT LEFT (CBUF CHANGED)
1300 REM DISP P - CHECK PARITY TO I/O PUT ATTRIBUTE IN a
1310 REM DISP R - BIT SHIFT 1 BIT RIGHT (CBUF CHANGED)

```



COMBASIC SUMMARY

INSTRUCTIONS/ARGUMENTS (Cont'd)

```
1320 REM DISP SX* - SET BYTE VARIABLE a TO 0 IF X* =
1330 REM ..... CBUF CONTENTS AT LOCATION A
1340 REM DISP SIX* - SAME AS SX* BUT IGNORE PARITY
1350 REM DISP TAB - TAB SCREEN POSITIONS
1360 REM DISP T1 =EVERY POSITION
1370 REM DISP T2 =EVERY OTHER POSITION
1380 REM DISP T4 =EVERY FOUR POSITIONS
1390 REM DISP T8 =EVERY EIGHT POSITIONS
1400 REM ATTRIBUTE BYTE BREAKDOWN
1410 REM -----
1420 REM B7, B6, B5, B4, B3, B2, B1, B0
1430 REM B7, B6, B5 OCTAL NUMBER
1440 REM ..... 000=DEFAULT
1450 REM ..... 001=END OF DATA
1460 REM ..... 002=FIFO FULL
1470 REM ..... 003=FLAG (SDLC ONLY)
1480 REM ..... 004=PARITY ERROR (NON BCC CHAR)
1490 REM ..... 005=BCC
1500 REM ..... 006=IDLE
1510 REM ..... 007=SELECTED LEAD ON
1520 REM .. PRIORITY OF ABOVE LO 007, 004, 005, 003, 002 HI
1530 REM B4
1540 REM 1=BCC BYTE IN ERROR (BLINK SUB CHARACTER)
1550 REM B3, B2
1560 REM ..... 00=ASCII
1570 REM ..... 01=HEX
1580 REM ..... 10=? UNASSIGNED
1590 REM ..... 11=? UNASSIGNED
1600 REM B1
1610 REM ..... 1=OUT OF SYNC (HALF INTENSE VIDEO)
1620 REM B0
1630 REM 1=P3 DATA
1640 REM STATUS BYTE BREAKDOWN
1650 REM -----
1660 REM B7, B6, B5, B4, B3, B2, B1, B0
1670 REM ..... B7=UNASSIGNED
1680 REM ..... B6=P3 DATA
1690 REM ..... B5=RI (P22) 1=ON
1700 REM ..... B4=B (P18) 1=ON
1710 REM ..... B3=A (P11) 1=ON
1720 REM ..... B2=COD(P08) 1=ON
1730 REM ..... B1=CTS(P05) 1=ON
1740 REM ..... B0=RTS(P04) 1=ON
1750 REM
1760 REM ----
```

## COMBASIC SUMMARY

## INSTRUCTIONS/ARGUMENTS (Cont'd)

```

1770 REM DIM
1780 REM -----
1790 REM DIM - DIMENSION - STRING MEMORY ALLOCATION A*-Z*
1800 REM . RESERVE MEMORY FOR STRINGS *
1810 REM .... DIM A*=10, B*=100, C*=1000
1820 REM .... DIM A*(10), B*(100), C*(1000)
1830 REM
1840 REM -----
1850 REM FESYN
1860 REM -----
1870 REM FESYN - USED TO CAUSE FRONT ENDS TO RESYNC
1880 REM . FROM COMBASIC PROG DURING EXECUTION.
1890 REM
1900 REM -----
1910 REM FOR / NEXT LOOP
1920 REM -----
1930 REM .. EXECUTES INSTRUCTIONS WITHIN THE LOOP T TIMES
1940 REM .. FOR T = A TO B STEP C
1950 REM .. T IS THE NUMBER OF TIMES & MUST BE A VARIABLE
1960 REM .. A IS THE STARTING NUMBER & CAN BE A VARIABLE
1970 REM .. B IS THE ENDING NUMBER & CAN BE A VARIABLE
1980 REM .. C IS THE INCREMNT NUMBER & CAN BE A VARIABLE
1990 REM FOR T = 1 TO 50
2000 REM INSTRUCTIONS GO HERE (DISP G, P, D) ETC.
2010 REM NEXT T
2020 REM
2030 REM -----
2040 REM IF
2050 REM -----
2060 REM IF GOTO/GOSUB
2070 REM IF "PROGRAM-1" GOTO 180
2080 REM IF FA=-123 GOTO 140
2090 REM IF A$ = "ABCDE" GOTO 140
2100 REM IF A=B GOTO 140
2110 REM IF A<>B GOTO 140
2120 REM IF A> B GOTO 140
2130 REM IF A< B GOTO 140
2140 REM IF A=>B GOTO 140
2150 REM IF A=<B GOTO 140
2160 REM IF A+2-C = B+D-3 GOTO 140
2170 REM IF A+2-C = B+D-3 GOSUB 140
2180 REM IF a = b goto 140
2190 REM IF a <> b goto 140
2200 REM IF a = "X" goto 140 (CANNOT SAY <> WITH THIS)
2210 REM IF a <> 1 goto 140

```

## COMBASIC SUMMARY

### INSTRUCTIONS/ARGUMENTS (Cont'd)

```
2220 REM IF [VARIABLE STATEMENT] [CONDITION] GOTO LINE NO
2230 REM IF [VARIABLE STATEMENT] [CONDITION] GOSUB LINE NO
2240 REM
2250 REM -----
2260 REM INPUT
2270 REM -----
2280 REM INPUT "YOUR MSG TO CRT" [VARIABLE]
2290 REM INPUT - USED TO INPUT FROM KEYBOARD ONLY, LANG
2300 REM ..... IS LANGUAGE & PARITY IN I/O
2310 REM ..... INPUT a - NO CURSOR, ONE CHARACTER, GOES
2320 REM ..... INTO BYTE VARIABLE a
2330 REM ..... INPUT a SAVE z - IF z=0 NOT FUNCTION KEY
2340 REM ..... INPUT A -CURSOR, NUMBER ONLY , GOES
2350 REM ..... INTO A, TERMINATE WITH RETURN KEY
2360 REM ..... INPUT A$-CURSOR, ALPHA/NUM , GOES
2370 REM ..... INTO A$, TERMINATE WITH ^P KEY
2380 REM . NOTE: MAX INPUT OF A$= 32 CHARACTERS OR FROM
2390 REM . CURSOR TO END OF SCREEN LINE, WHICHEVER
2400 REM . IS LESS.
2410 REM . NOTE: WHEN INPUT INSTRUCTION INTERRUPTED BY
2420 REM . WHEN INST. INPUT INST. IS TERMINATED.
2430 REM
2440 REM -----
2450 REM ON
2460 REM -----
2470 REM ON [VARIABLE STATEMENT] GOTO/GOSUB
2480 REM ... ON A GOTO 100,200,300,400, ETC.
2490 REM .... IF A=1 GOTO 100, IF A=2 GOTO 200 ETC.
2500 REM ... ON A-50 GOTO 100,200,300,400, ETC.
2510 REM . SUBTRACT 50 FROM A THEN DO ON/GOTO
2520 REM
2530 REM ---
2540 REM GOTO/GOSUB SEE IF
2550 REM ---
2560 REM -----
2570 REM LET
2580 REM -----
2590 REM LET A=0, a=0, FA=0, LET A$="12345", LET A$(4,5)="45"
2600 REM LET FA=2400, SPEED=FA
2610 REM LET SYNC="\XX\XX\" (ANY 2 CHARACTERS)
2620 REM LET SYNC="[3232]" (HEX PAIRS)
2630 REM LET SYNC = "\XX\XX\" OR LET SYNC = "[1616]"
2640 REM LET PARITY = ODD,EVEN,NONE,MARK,SPACE,IGNORE
2650 REM . IGNORE FORCES ZERO PARITY BIT
2660 REM LET NRZI = YES or NO
```

## COMBASIC SUMMARY

## INSTRUCTIONS/ARGUMENTS (Cont'd)

```

2670 REM LET MODE = ASYN, SYN, SDLC, TRAN
2680 REM .... TRAN FOR RECEIVE ONLY NOT TRANSMIT
2690 REM .... CLOCK MUST BE INT FOR ONE.5 STOPS, MODE ASYN
2700 REM LET STOPS = ONE, ONE.5, TWO
2710 REM LET CLOCK= EXT , INT
2720 REM LET LANG ="ASCII", "EBCDIC", "BCD", "SELECTRIC"
2730 REM .           "SBT", "SELECTRIC", "IPARS", "BAUDOT"
2740 REM LET FA=FB*FC/FD+FE-FF (MATH LEFT TO RIGHT)
2750 REM . FA-FZ ARE FLOATING POINT VARIABLES +, -, *, /
2760 REM . a-z ARE BYTE VARIABLES +, - ONLY, MAX VAL 255
2770 REM . A-Z ARE INTEGER VARIABLES +, - ONLY, MAX VAL 32K
2780 REM LET b=b RTR 3 TAKE b ROTATE RIGHT 3 PSTNS PUT b
2790 REM LET b=b RTL 3 TAKE b ROTATE LEFT 3 PSTNS PUT b
2800 REM LET a=a LPAR - FORCES PARITY OF a TO I/O PARITY
2810 REM LET a=a LANG -CONVERTS a FROM ASCII TO I/O LANG
2820 REM LET a= DSTAT (BITS 76543210)
2830 REM . B7 - 0=WRITE PROTECTED 1=NOT PROTECTED
2840 REM . B6 - 0=NOT USED
2850 REM . B5 - 0=HEAD LOADED 1=HEAD UNLOADED
2860 REM . B4 - 0=NO 1=INTERRUPT
2870 REM . B3 - 0=DMA TRANSFER COMPUTED (WILL RESET WITH
2880 REM .           STATUS READ)
2890 REM . B2 - 0=SIDE 0 ENABLED 1=SIDE 0 DISABLED
2900 REM . B1 - 0=DRIVE 0 ENABLED 1=DRIVE 0 DISABLED
2910 REM . B0 - 0=DOUBLE DENSITY 1=SINGLE DENSITY
2920 REM LET FA=MSTM - PUT MEASUREMENT TIMER IN FA
2930 REM LET a="[FF]" - HEX VALUE
2940 REM LET a="(A)"- PARITY FLAWED LETTER A
2950 REM LET A$(1,5)=a TO e, PUTS abcde IN A$ 1 THRU 5
2960 REM LET A$(2,2)=b TO b PUTS CNTNTS b IN A$ POS 2
2970 REM LET A = LEN A$ - PUT THE LENGTH OF A$ IN A
2980 REM LET a="0H"7E" HEX FLAG IN a
2990 REM LET a="0O"176" OCTAL FLAG IN a
3000 REM LET a="0B"01111110" BINARY FLAG IN a
3010 REM LET a=126 DECIMAL FLAG IN a
3020 REM LET a=a HEX FOR BCD, IPARS & SELECTRIC
3030 REM . MOVES BIT 0 TO BIT 6
3040 REM . MOVES BIT 1 TO BIT 5
3050 REM . MOVES BIT 2 TO BIT 4
3060 REM . MOVES BIT 3 TO BIT 3
3070 REM . MOVES BIT 4 TO BIT 2
3080 REM . MOVES BIT 5 TO BIT 1
3090 REM . MOVES BIT 6 TO BIT 0
3100 REM ----
3110 REM LOGICAL AND - BOTH HAVE TO BE 1 TO GET 1

```

COMBASIC SUMMARY

INSTRUCTIONS/ARGUMENTS (Cont'd)

```
3120 REM LET a=b AND c
3130 REM ----
3140 REM LOGICAL OR - EITHER HAS TO BE 1 TO GET 1
3150 REM LET a =b OR c
3160 REM ----
3170 REM LOGICAL EXCLUSIVE OR - IF BOTH 0 YOU GET 0
3180 REM LET a = b XOR c
3190 REM
```

**COMBASIC SUMMARY**  
**INSTRUCTIONS/ARGUMENTS (Cont'd)**

**COMBASIC2**

```

10 REM COMBASIC2  21 MAY 82   4A4B
20 REM PRINT
30 REM -----
40 REM PRINT#0    CRT DEFAULT
50 REM PRINT#1    TRANSMITTER
60 REM PRINT#2    XIO PORT(ALWAYS ASYNC MODEM)
70 REM .          (TRANSMIT P3,RECEIVE P2)
80 REM .          (+12V ON PIN 9, -12V ON P10)
90 REM PRINT#3    CBUF(STATUS OR ATTRIBUTE WILL BE 0)
100 REM PRINT#4   SYNC PULSE FOR SCOPE BNC CONNECTOR
110 REM PRINT#5   BEEPER (HALF SECOND TONE)
120 REM PRINT"YOUR MESSAGE WITHIN QUOTES";
130 REM .....;   STOPS LINE FEED, RETURN
140 REM PRINT a, [a], %7.7, FA,
150 REM PRINT a   BYTE VARIABLE DECIMAL VALUE
160 REM PRINT [a] BYTE VARIABLE CHARACTER
170 REM PRINT %7.7, 7 PLACES BEFORE DECIMAL, 7 AFTER
180 REM .        PLUS SIGN(-)IF NEG, ( )IF POS
190 REM PRINT FA  FLOATING POINT VARIABLE
200 REM PRINT A   INTEGER VARIABLE
210 REM PRINT [EXPRESSION] A+B*FA/FB+D-a+b
220 REM PRINT "[FF] (A)"
230 REM [FF] HEX, (A) PARITY FLAWED LETTER A
240 REM PRINT A$, B$, "ABC[FF](A)XYZ";
250 REM PRINT A$!  PRINT CONTINUOUSLY
260 REM PRINT A$(1-15), A$(16, 20);
270 REM PRINT #1, "\XX\XX\AA\EQ\" SAMPLE POLL TRANSMIT
280 REM -----
290 REM READ / DATA / RESTORE
300 REM -----
310 REM READ A
320 REM READ a
330 REM READ FA
340 REM .   MULTIPLE READS MUST BE ON DIFFERENT LINES
350 REM .   DATA STATEMENTS MUST BE NUMERIC
360 REM .   STATEMENTS MUST BE CONTIGUOUS IN PROGRAM
370 REM .   RECOMMEND PUTTING AT END OF PROGRAM
380 REM .   DATA 1, 2, 3, 4, 5, 06, 6, 03, 7, 0, 12
390 REM RESTORE-RESTORE READ/DATA POINTER TO BEGINNING
400 REM -----
410 REM SCREEN

```

COMBASIC SUMMARY

INSTRUCTIONS/ARGUMENTS (Cont'd)

```
420 REM -----
430 REM SCREEN P=0-0 POSITION
440 REM SCREEN S=64-7 SIZE
450 REM SCREEN C CLEAR
460 REM SCREEN L LINE FEED
470 REM SCREEN R RETURN CURSOR TO BEGINNING OF LINE
480 REM SCREEN H PRINT IN HEX
490 REM SCREEN I INVERSE VIDEO
500 REM SCREEN U UNDERLINE
510 REM SCREEN N RETURN SCREEN TO NORMAL MODE
520 REM SCREEN C READ FROM SCREEN INTO BYTE VAR a
530 REM SCREEN B BACK UP CURSOR ONE POSITION
540 REM SCREEN SA PUT SCREEN SIZE INTO A & B
550 REM . INTEGER VARIABLES A&B, CAN BE
560 REM . ANY 2 CONSECUTIVE INTEGER VARIABLES
570 REM SCREEN SZ CHANGE SCREEN SIZE TO NEXT SIZE
580 REM SCREEN E CLEAR LINE TO END OF LINE
590 REM SCREEN Z SCROLL SCREEN UP, CLEAR BOTTOM LINE
600 REM SCREEN "YOUR MESSAGE IN QUOTES"
610 REM SCREEN ATTRIBUTES- SEE DISP INST
620 REM -----
630 REM STATE
640 REM -----
650 REM . NOTE * INDICATES DEFAULT CONDITION
660 REM STATE MODEM - SIMULATE MODEM
670 REM . CONTROL MODEM LEADS
680 REM STATE TERM - SIMULATE TERMINAL
690 REM . CONTROL TERMINAL LEADS
700 REM STATE BFG *- PASSIVELY MONITOR
710 REM STATE HDUX *- INTERLEAVED DISPLAY ON CRT
720 REM STATE FDUX - P2 & P3 SEP DISPLAY ON CRT
730 REM STATE STAT *- RECORD STATUS BYTE
740 REM STATE NSTAT - DO NOT RECORD STATUS BYTE
750 REM STATE P3ONLY - FOLLOWING INSTRUCTIONS FOR
760 REM . PIN 3 RECEIVER ONLY
770 REM STATE NP3ONLY *-FOLLOWING INSTRUCTIONS FOR
780 REM . PIN 2 AND PIN 3 RECEIVERS
790 REM STATE XIDL=FF -(FF IS A HEX PAIR)TRANSMIT IDLE
800 REM . CHARACTERS(NORMALLY ONES)
810 REM STATE LEAD=4 -CAN BE 4, 5, 8, A, B OR 22
820 REM . RECORDED IN ATTRIBUTE BYTE.
830 REM . DATA ON CRT UNDERLINED IF LEAD
840 REM . STATED IS 'ON'WHEN CHARACTER
850 REM . IS RECEIVED
860 REM STATE NLEAD *-CANCELS STATE LEAD INSTR
```

## COMBASIC SUMMARY

## INSTRUCTIONS/ARGUMENTS (Cont'd)

```

870 REM STATE FLTR =FF -(FF) IS A HEX PAIR, CHARACTER
880 REM . SPECIFIED WILL NOT BE SEEN BY
890 REM . FRONT END RECEIVERS
900 REM STATE NFLTR *-CANCELS FILTER
910 REM STATE PIPE -TURNS OFF THE CRT COMPLETELY TO
920 REM . ELIMINATE PROCESSING OVERHEAD.
930 REM . PRIMARILY USED FOR HIGH-SPEED
940 REM . DATA CAPTURE WITH STATUS OR
950 REM . ATTRIBUTE.
960 REM STATE SYNC -RECEIVERS PASS ONLY DATA
970 REM . THAT IS IN SYNC
980 REM STATE NSYNC -RECVRS PASS DATA IN SYNC OR NOT
990 REM STATE FCS=CRC16,CRC12,LRC7,LRC8,CCITT
1000 REM . FCS PRESETS CALCULATOR TO ONES
1010 REM STATE CRC=CRC16,CRC12,LRC7,LRC8,CCITT
1020 REM . CRC PRESETS CALCULATOR TO ZERO
1030 REM STATE CRC=CRC16 *- DEFAULT
1040 REM STATE CRC=CRC6 - IPARS OPTION
1050 REM STATE DISCON - WHEN CAPTURE BUFFER FILLS 1/4K
1060 REM . DATA WILL BE WRITTEN ON DISC
1070 REM . CAPTURE BUFFER AUTOMATICALLY
1080 REM . ALLOCATED TO 4K
1090 REM STATE DISCOFF *- IF DISCON WAS STATED THIS WILL
1100 REM . MARK END OF DISC FILE.
1110 REM . IF DISCON NOT STATED, NOTHING
1120 REM . WILL HAPPEN.
1130 REM .** DISC ****IF DATA IS RECORDED ON DISC****
1140 REM .** CAUTION ****AND A PROGRAM IS SUBSEQUENTLY**
1150 REM .** DISC ****SAVED ON THAT SAME DISC,THE****
1160 REM .** CAUTION ****DATA WILL BE OVERWRITTEN ****
1170 REM STATE NLOCK *-UNLOCKS THE KEYBOARD
1180 REM STATE LOCK -LOCKS THE KEYBOARD
1190 REM STATE NUPC *-ALLOWS KEYBOARD UPPER/LOWER CASE
1200 REM STATE UPC -ALLOWS ONLY UPPER CASE KEYBOARD
1210 REM . WHEN KBD GOTO NNNN
1220 REM STATE NFAST *-CANCELS FAST
1230 REM STATE FAST -ELIMINATES ATTRIBUTE TO CRT
1240 REM -----
1250 REM STOP - STOPS PROGRAM EXECUTION
1260 REM . PRINTS "STOP AT LINE NUMBER NNNN" CRT
1270 REM -----
1280 REM TIMER
1290 REM -----
1300 REM TIMER - THIS IS THE INTERRUPT TIMER
1310 REM . WHEN TIMER = 3000 GOTO 140.

```



COMBASIC SUMMARY

INSTRUCTIONS/ARGUMENTS (Cont'd)

```
1320 REM .           INTERRUPT TIMER SET TO VALUE
1330 REM .           (3000)AND DECREMENTS TO ZERO,
1340 REM .           PROGRAM EXECUTION VECTORED
1350 REM .           TO LINE NUMBER IN STATEMENT
1360 REM -----
1370 REM TRANSMISSION MODES
1380 REM SDLC - SET IO TO SDLC
1390 REM .           DIM STRINGS,SET VALUE OF STRINGS,DLC
1400 REM .           STRINGS, PRINT #1,RECEIVING STRING
1410 REM .           IN DLC INSTRUCTION EXAMPLE BELOW
1420 REM .           10 DIM A$=100,B$=10,C$=10,D$=10,E$=10
1430 REM .           15 LET B$="123",C$="45",D$="AB",E$="CD"
1440 REM .           20 DLC A$=B$,C$D$,.,E$
1450 REM .           30 PRINT #1,A$
1460 REM .           MESSAGE WILL LOOK LIKE THIS
1470 REM .           ~123BC~45ABBC~~~CDBC~ (BC=FCS CHAR)
1480 REM .           FOR'NRZI' INSERT LINE 25 NRZI A$
1490 REM .           THE RECEIVING STRING (A$) MUST BE
1500 REM .           LARGE ENOUGH TO RECEIVE DATA,FLAGS AND
1510 REM .           INSERTED ZEROS AND FCS CHARACTERS
1520 REM BINARY SYNCHRONOUS
1530 REM .           LET BCS ab =B$,C$,D$,E$
1540 REM .           BCC CHARACTERS ARE CALCULATED ON THE
1550 REM .           CONTENTS OF B$,C$,D$,E$ AND PUT INTO
1560 REM .           BYTE VARIABLES ab
1570 REM .           10 LET BCS ab = B$,C$,D$,E$
1580 REM .           20 PRINT #1,"\\XX\\XX\\",A$,[a],[b]
1590 REM .           MESSAGE LOOKS LIKE THIS
1600 REM .           \\XX\\XX\\12345ABCDBC
1610 REM -----
1620 REM TON/TOFF      (PINS)
1630 REM -----
1640 REM TON P2,P2D,P2B - TURNON P2,P2 DISPLAY,P2 CBUF
1650 REM TON P3,P3D,P3B - TURNON P3,P3 DISPLAY,P3 CBUF
1660 REM TON P4,P5,P6,P8,P11,PA,P18,PB,P20,P22,P24
1670 REM .           NOTE TON PINS 2,3 BEFORE ATTEMPTING TO TON
1680 REM .           ANY OTHER PINS
1690 REM TOFF         OPPOSITE OF TON
1700 REM -----
1710 REM VARIABLES
1720 REM -----
1730 REM INTEGER VARIABLES - UPPER CASE A THRU Z
1740 REM .           NUMERIC ONLY, MAX 32,767
1750 REM .           (+) ADD,(-) SUBTRACT ONLY
1760 REM BYTE VARIABLES - LOWER CASE a THRU z
```

## COMBASIC SUMMARY

## INSTRUCTIONS/ARGUMENTS (Cont'd)

```

1770 REM .                ALPHANUMERIC, MAX 255
1780 REM .                (+) ADD, (-) SUBTRACT ONLY
1790 REM FLOATING POINT VARIABLES - FA THRU FZ
1800 REM .                NUMERIC ONLY, USED FOR
1810 REM .                LARGE + OR - NUMBERS, + ADD
1820 REM .                (-)SUBTRACT, (*)MULT, (/)DIV
1830 REM STRING VARIABLES - A$ THRU Z$
1840 REM .                ALPHANUMERIC DATA STORAGE
1850 REM .                LENGTH MUST BE DIMENSIONED
1860 REM -----
1870 REM WAIT
1880 REM -----
1890 REM WAIT 1000  -FOR 1000 MILLESECONDS (1 SEC)
1900 REM WAIT A    -FOR A MILLESECONDS(1 SEC IF A=1000)
1910 REM WAIT A +B
1920 REM WAIT XDONE -WAIT UNTIL THE TRANSMITTER HAS
1930 REM .                BEEN LOADED WITH THE LAST TWO
1940 REM .                CHARACTERS BEFORE PROCEEDING
1950 REM .                TO THE NEXT INSTRUCTION. WAITS 2
1960 REM .                ADDITIONAL CHARACTER TIMES
1970 REM .                BEFORE TURNING OFF ANY PINS.
1980 REM -----
1990 REM WHEN
2000 REM -----
2010 REM WHEN IS TIME AND EVENT ORIENTED
2020 REM WHEN P2="ABC" GOTO 140
2030 REM WHEN P2="[7E]" GOTO 140 (HEX VAL)
2040 REM WHEN P3=1 GOTO 140 (ANY ONE CHARACTER)
2050 REM WHEN P4=1 GOTO 140 (PIN4 IS ON)
2060 REM WHEN P2=BCC,1 GOTO 140 (TWO BCC CHARACTERS)
2070 REM WHEN P2=BCC GOTO 140 (BLOCK CHECK CHAR)
2080 REM WHEN P2=BCG GOTO 140 (BLOCK CHECK CHAR GOOD)
2090 REM WHEN P2=BCB GOSUB 140 (BLOCK CHECK CHAR BAD)
2100 REM WHEN P2=PE GOSUB 140 (PARITY ERROR)
2110 REM WHEN P3=FLAG GOSUB 140 (SDLC FLAG)
2120 REM WHEN P3 TON P3D,P3="123"A$ GOTO 140
2130 REM .                WHEN P3 IS -TO THE FIRST 3 POSITIONS OF A$
2140 REM WHEN P2 TOFF P3 ,P2="123"A$ GOTO 140
2150 REM WHEN P2 a
2160 REM .                EXECUTION WILL NOT GO BEYOND THE ABOVE
2170 REM .                INSTRUCTION UNTIL A CHARACTER IS REC'VD
2180 REM .                ON P2, THEN EXECUTION GOES TO NEXT LINE
2190 REM .NOTE: THIS INSTRUCTION CAN BE USED FOR P3
2200 REM .                WHEN P2/P3 CAN BE INTERRUPTED BY
2210 REM .                ANOTHER WHEN. RETURN FROM WHEN/GOSUB

```

COMBASIC SUMMARY

INSTRUCTIONS/ARGUMENTS (Cont'd)

```
2220 REM .           WILL BE SET TO WHEN P2/P3 LOCATION.
2230 REM WHEN TIMER = 3000 GOTO 140 (3 SECONDS)
2240 REM WHEN KBD a GOTO 140
2250 REM .           WHEN THE KEYBOARD IS STRUCK PUT CHAR IN a
2260 REM .           IN NO PARITY ASCII - USE DECIMAL TO
2270 REM .           INTERROGATE a , IF a=97 GOTO 140
2280 REM .           (97 = LOWER CASE NO PARITY ASCII A)
2290 REM WHEN XIOIN a GOTO 140 (XIO PORT P2)
2300 REM WHEN FULL GOTO 140 (CBUF)
2310 REM WHEN P4=1 GOTO 140 (RTS)
2320 REM WHEN P5=1 GOTO 140 (CTS)
2330 REM WHEN P6=1 GOTO 140 (DSR)
2340 REM WHEN P8=1 GOTO 140 (COD)
2350 REM WHEN P11=1 GOTO 140
2360 REM WHEN PA=1 GOTO 140
2370 REM WHEN P18=1 GOTO 140
2380 REM WHEN PB =1 GOTO 140
2390 REM WHEN P20=1 GOTO 140 (DTR)
2400 REM WHEN P22=1 GOTO 140 (RI)
2410 REM WHEN ERROR GOTO 140
2420 REM .           LINE NUMBER GOES INTO E
2430 REM .           ERROR NUMBER GOES INTO e
2440 REM WHEN RTC=A$ GOTO 140 (REAL TIME CLOCK)
2450 REM .           LET A$="123 10:20:00"
2460 REM WHEN P2= LIMIT OF 4 WHENS IN EFFECT AT ONE TIME
2470 REM .           FOR PIN 2
2480 REM WHEN P3= LIMIT OF 4 WHENS IN EFFECT AT ONE TIME
2490 REM .           FOR PIN 3
2500 REM WHEN/GOSUB THE RETURN ADDRESS AT THE END OF
2510 REM .           THE SUBROUTINE WILL BE SET TO THE
2520 REM .           INSTRUCTION AFTER THE INSTRUCTION
2530 REM .           COMPLETED BEFORE THE WHEN VECTOR;
2540 REM .           THEREFORE NO INSTRUCTIONS WILL BE
2550 REM .           MISSED BECAUSE OF THE VECTOR.
2560 REM -----
2570 REM XDONE - REFER TO WAIT INSTRUCTION
2580 REM XIOIN
2590 REM .           XIOIN IS ALWAYS ASYNC, MODEM
2600 REM XIO PORT - REFER TO PRINT#3
```

## ENCORE INSTRUCTION EXECUTION TIMES

INSTRUCTION	OFF LINE (MS)	SYNC 9600FDUX, D & B ON (MS)
ABORT ALL	1.15	2.50
ABORT FULL	0.11	0.11
ABORT KBD	0.11	0.11
ABORT PIN	0.11	0.11
ABORT P2	0.45	0.92
ABORT P3	0.45	0.92
ABORT RTC	0.11	0.11
ABORT TIMER	0.11	0.11
ABORT XIOIN	0.11	0.11
BOOT	-	-
CALL	-	-
CHAIN	-	-
CLEAR CBUF	0.08	0.08
CLEAR MSTM	0.12	0.12
CLEAR RETURN	0.04	0.04
DIM	0.04	-
DISP B	0.37	-
DISP C	0.27	-
DISP D	0.30	-
DISP E	0.40	-
DISP F	0.26	-
DISP G	0.44	-
DISP H	0.31	-
DISP L	46. -	-
DISP P	0.32	-
DISP R	46. -	-
DISP SIX\$	0.90	-
DISP SX\$ (X\$="12345")	0.90	-
FESYN	-	1.8-
FOR/NEXT (T=1 TO 1)	0.08	0.08
GOTO	-.40	-.40
IF A = B	1.9	4.25
IF A = > B	1.9	4.25
IF A = < B	1.9	4.25
IF A < > B	1.9	4.25
IF A + 2 - C = B + D - 3	4.5	13. -
IF A\$ = "ABCDE"	1.09	2.5

ENCORE INSTRUCTION EXECUTION TIMES (Cont'd)

INSTRUCTION	OFF LINE (MS)	SYNC 9600FDUX, D & B ON (MS)
IF FA = -123	1.42	3.25
IF "PROGRAM -1"	78.-	187.-
INPUT	-	-
LET A = 0	-.05	-.05
LET A\$ = "12345"	1.1-	2.65
LET A\$ (1,5) = a TO e	1.2	2.6-
LET A\$ (2,2) = b TO b	-.7	1.65
LET A\$ (4,5) = "45"	-.82	1.76
LET a = 0	-.05	-.05
LET a = 00"176"	-.06	-.06
LET a = 126	-.06	-.06
LET a = "(A)"	-.06	-.06
LET a = a HEX	-.08	-.08
LET a = a LANG	-.11	-.11
LET a = a LPAR	-.12	-.12
LET a = b	-.06	-.06
LET a = b AND c	-.11	-.11
LET a = b OR c	-.11	-.11
LET a = b RTL 3	-.13	-.13
LET a = b RTR 3	-.13	-.13
LET a = bx OR c	-.11	-.11
LET a = "[FF]"	-.06	-.06
LET a = 0B "01111110"	-.06	-.06
LET a = 0H "7E"	-.06	-.06
LET a = DSTAT	-.06	-.06
LET FA = 0	-.17	-.17
LET FA = 2400	.17	-
LET FA = FB * FC/FD + FE - FF	15.-	38.-
LET FA = MSTM	-.96	1.6
LET LANG =	-.03	-
LET MODE =	-.03	-
LET NRZI =	-.03	-
LET PARITY =	-.03	-
LET SPEED = FA	35.-	-
LET STOPS =	-.03	-
LET SYNC = "[1616]"	-.03	-
LET SYNC = "S <sub>Y</sub> S <sub>Y</sub> "	-.03	-
PRINT A	12.25	-
PRINT A = B * FA/FB - a + b	45.0	-
PRINT A\$	1.4	-
PRINT "A\$, B\$, ABC (FF) (A) XYZ"	3.2	-
PRINT A\$ (1-15), A\$ (16,20);	2.0	-
PRINT a	15.-	-

## ENCORE INSTRUCTION EXECUTION TIMES (Cont'd)






INSTRUCTION	OFF LINE (MS)	SYNC 9600FDUX, D & B ON (MS)
PRINT [a]	.47	-
PRINT a, a, %7.7, FA	27.5	-
PRINT FA	12.25	-
PRINT "FF (A)"	.6	-
PRINT "YOUR MESSAGE WITHIN QUOTES"	3.0	-
PRINT #1, "S <sub>Y</sub> S <sub>Y</sub> AA E <sub>Q</sub> "	16.5	-
PRINT #5	-.04	-
READ A	14.0	-
READ a	14.0	-
READ FA	-.26	-
RESTORE	-.05	-
SCREEN B	-.09	-
SCREEN C	16.0	-
SCREEN E	-.55	-
SCREEN G <sub>a</sub>	-.13	-
SCREEN H	-.05	-
SCREEN I	-.05	-
SCREEN L	-.14	-
SCREEN N	-.05	-
SCREEN P = 0-0	-.10	-
SCREEN R	-.12	-
SCREEN S = 64-7	16.0	-
SCREEN SA	-.10	-
SCREEN SZ	16.0	-
SCREEN U	-.05	-
SCREEN "YOUR MESSAGE WITHIN QUOTES"	2.3	-
STATE BRG	16.0	-
STATE FAST	6.0	-
STATE FCS = OR CRC =	-.04	-
STATE FDUX	16.0	-
STATE FLTR = FF	3.75	-
STATE HDUX	16.0	-
STATE LEAD = Y	3.75	-
STATE LOCK	-.04	-
STATE MODEM	-.05	-
STATE NFAST	6.0	-
STATE NFLTR	1.85	-
STATE NLEAD8	3.75	-
STATE NLOCK	-.04	-
STATE NP3ONLY	-.43	-
STATE NSTAT	06.0	-
STATE NSYNC	1.85	-
STATE NUPC	.04	-

ENCORE INSTRUCTION EXECUTION TIMES (Cont'd)

INSTRUCTION	OFF LINE (MS)	SYNC 9600FDUX, D & B ON (MS)
STATE P3ONLY	-.43	-
STATE STAT	06.0	-
STATE SYNC	1.85	-
STATE TERM	-.05	-
STATE UPC	.04	-
STATE XIDL = FF	3.75	-
TON P2, P2D, P2B	-.06	-
TON P3, P3D, P3B	-.06	-
TON P4, P5, P6, P8, P11, P18, P20, P22, P24	-.18	-
WHEN	0.12	-.12

**APPENDIX D**  
**CONTROL GRAPHICS AND CODE CHARTS**

**\*\*\*NOTE\*\*\***

When using case shift codes, remember to precede data with the proper case shift character. For example, a typical fox message in BAUDOT would include the following: CR CR LF ^ THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG \ 1234567890. The caret (^) tells the receiving device that the data to follow is comprised of capital letters. The backward slash (\) tells the receiving device that numbers or symbols will follow. When entering data from the ENCORE keyboard, these case shift characters are not automatically entered by simply using the  and  keys. Use  ,  or  .



**CONTROL GRAPHICS**

All control symbols displayed on the CRT appear as two character mnemonics as shown below.

**CONTROL GRAPHICS**

KEYBOARD DESIGNATION	VIDEO DISPLAY	DEFINITION	KEYBOARD DESIGNATION	VIDEO DISPLAY	DEFINITION
NUL	N <sub>U</sub>	Null	DC1	D <sub>1</sub>	Device Control 1
SOH	S <sub>H</sub>	Start of Heading	DC2	D <sub>2</sub>	Device Control 2
STX	S <sub>X</sub>	Start of Text	DC3	D <sub>3</sub>	Device Control 3
ETX	E <sub>X</sub>	End of Text	DC4	D <sub>4</sub>	Device Control 4
EOT	E <sub>T</sub>	End of Transmission	NAK	N <sub>K</sub>	Negative Acknowledgement
ENQ	E <sub>Q</sub>	Enquire	SYN	S <sub>Y</sub>	Synchronous Idle
ACK	A <sub>K</sub>	Affirmative Acknowledgement	ETB	E <sub>B</sub>	End of Transmission Block
BEL	B <sub>L</sub>	Bell	CAN	C <sub>N</sub>	Cancel
BS	B <sub>S</sub>	Backspace	EM	E <sub>M</sub>	End of Medium
HT	H <sub>T</sub>	Horizontal Tab	SUB	S <sub>B</sub>	Substitute
LF	L <sub>F</sub>	Line Feed	ESC	E <sub>C</sub>	Escape
VT	V <sub>T</sub>	Vertical Tab	FS	F <sub>S</sub>	File Separator
FF	F <sub>F</sub>	Form Feed	GS	G <sub>S</sub>	Group Separator
CR	C <sub>R</sub>	Carriage Return	RS	R <sub>S</sub>	Record Separator
SO	S <sub>O</sub>	Shift Out	US	U <sub>S</sub>	Unit Separator
SI	S <sub>I</sub>	Shift In	DEL	⋮	Delete
DLE	D <sub>L</sub>	Data Link Escape			

## ASCII CONVERSION

<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>	<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>
NUL	00000000	000	000	00	-	00101101	055	045	2D
SOH	00000001	001	001	01	.	00101110	056	046	2E
STX	00000010	002	002	02	/	00101111	057	047	2F
ETX	00000011	003	003	03	0	00110000	060	048	30
EOT	00000100	004	004	04	1	00110001	061	049	31
ENQ	00000101	005	005	05	2	00110010	062	050	32
ACK	00000110	006	006	06	3	00110011	063	051	33
BEL	00000111	007	007	07	4	00110100	064	052	34
BS	00001000	010	008	08	5	00110101	065	053	35
HT	00001001	011	009	09	6	00110110	066	054	36
LF	00001010	012	010	0A	7	00110111	067	055	37
VT	00001011	013	011	0B	8	00111000	070	056	38
FF	00001100	014	012	0C	9	00111001	071	057	39
CR	00001101	015	013	0D	:	00111010	072	058	3A
SO	00001110	016	014	0E	;	00111011	073	059	3B
SI	00001111	017	015	0F	<	00111100	074	060	3C
DLE	00010000	020	016	10	=	00111101	075	061	3D
DC1	00010001	021	017	11	>	00111110	076	062	3E
DC2	00010010	022	018	12	?	00111111	077	063	3F
DC3	00010011	023	019	13	@	01000000	100	064	40
DC4	00010100	024	020	14	A	01000001	101	065	41
NAK	00010101	025	021	15	B	01000010	102	066	42
SYN	00010110	026	022	16	C	01000011	103	067	43
ETB	00010111	027	023	17	D	01000100	104	068	44
CAN	00011000	030	024	18	E	01000101	105	069	45
EM	00011001	031	025	19	F	01000110	106	070	46
SUB	00011010	032	026	1A	G	01000111	107	071	47
ESC	00011011	033	027	1B	H	01001000	110	072	48
FS	00011100	034	028	1C	I	01001001	111	073	49
GS	00011101	035	029	1D	J	01001010	112	074	4A
RS	00011110	036	030	1E	K	01001011	113	075	4B
US	00011111	037	031	1F	L	01001100	114	076	4C
SP	00100000	040	032	20	M	01001101	115	077	4D
!	00100001	041	033	21	N	01001110	116	078	4E
"	00100010	042	034	22	O	01001111	117	079	4F
#	00100011	043	035	23	P	01010000	120	080	50
\$	00100100	044	036	24	Q	01010001	121	081	51
%	00100101	045	037	25	R	01010010	122	082	52
&	00100110	046	038	26	S	01010011	123	083	53
'	00100111	047	039	27	T	01010100	124	084	54
(	00101000	050	040	28	U	01010101	125	085	55
)	00101001	051	041	29	V	01010110	126	086	56
*	00101010	052	042	2A	W	01010111	127	087	57
+	00101011	053	043	2B	X	01011000	130	088	58
,	00101100	054	044	2C	Y	01011001	131	089	59

**ASCII CONVERSION (Cont'd)**

<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>	<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>
Z	01011010	132	090	5A	v	01110110	166	118	76
(	01011011	133	091	5B	w	01110111	167	119	77
\	01011100	134	092	5C	x	01111000	170	120	78
)	01011101	135	093	5D	y	01111001	171	121	79
^	01011110	136	094	5E	z	01111010	172	122	7A
-	01011111	137	095	5F	{	01111011	173	123	7B
↖	01100000	140	096	60		01111100	174	124	7C
a	01100001	141	097	61	}	01111101	175	125	7D
b	01100010	142	098	62	~	01111110	176	126	7E
c	01100011	143	099	63	DEL	01111111	177	127	7F
d	01100100	144	100	64	UC/LC	10000000	200	128	80
e	01100101	145	101	65	HEX	10000001	201	129	81
f	01100110	146	102	66	RWD	10000010	202	130	82
g	01100111	147	103	67	STOP	10000011	203	131	83
h	01101000	150	104	68	↑	10000100	204	132	84
i	01101001	151	105	69	→	10000101	205	133	85
j	01101010	152	106	6A	UDF	10000110	206	134	86
k	01101011	153	107	6B	CMD	10000111	207	135	87
l	01101100	154	108	6C	CONT	10001000	210	136	88
m	01101101	155	109	6D	↓	10001001	211	137	89
n	01101110	156	110	6E	←	10001010	212	138	8A
o	01101111	157	111	6F	BRG	10001011	213	139	8B
p	01110000	160	112	70	VE/STAT	10001100	214	140	8C
q	01110001	161	113	71	CLEAR	10001101	215	141	8D
r	01110010	162	114	72	IC	10001110	216	142	8E
s	01110011	163	115	73	DC	10001111	217	143	8F
t	01110100	164	116	74	UC/ONLY	10010000	220	144	90
u	01110101	165	117	75	P	10010001	221	145	91

**CONTROL CHARACTER ABBREVIATIONS**

NUL	-	NULL OR ALL ZEROS	DC1	-	DEVICE CONTROL 1
SOH	-	START OF HEADING	DC2	-	DEVICE CONTROL 2
STX	-	START OF TEXT	DC3	-	DEVICE CONTROL 3
ETX	-	END OF TEXT	DC4	-	DEVICE CONTROL 4
EOT	-	END OF TRANSMISSION	NAK	-	NEGATIVE ACKNOWLEDGE
ENQ	-	ENQUIRY	SYN	-	SYNCHRONOUS IDLE
ACK	-	ACKNOWLEDGE	ETB	-	END OF TRANSMISSION BLOCK
BEL	-	BELL	CAN	-	CANCEL
BS	-	BACK SPACE	EM	-	END OF MEDIUM
HT	-	HORIZONTAL TAB	SUB	-	SUBSTITUTE
LF	-	LINE FEED	ESC	-	ESCAPE
VT	-	VERTICAL TAB	FS	-	FILE SEPARATOR
CR	-	CARRIAGE RETURN	GS	-	GROUP SEPARATOR
SO	-	SHIFT OUT	RS	-	RECORD SEPARATOR
FF	-	FORM FEED	US	-	UNIT SEPARATOR
SI	-	SHIFT IN	SP	-	SPACE
DLE	-	DATA LINK ESCAPE	DEL	-	DELETE

## ASCII

b8 b7 b6 b5					1	0	1	0	1	0	1	0	1	0	1	0	b8=1 b8=0	
					0	0	0	1	0	1	0	1	0	1	0	1		
B I T S	b4	b3	b2	b1 LSB	1st HEX DIGIT →	8	9	A	B	C	D	E	F					
						2nd HEX DIGIT ↓	0	1	2	3	4	5	6	7				
						0	NUL	DLE	SP	Ø	@	P	▾	p				
						1	SOH	DC1	!	1	A	Q	a	q				
						2	STX	DC2	"	2	B	R	b	r				
						3	ETX	DC3	#	3	C	S	c	s				
						4	EOT	DC4	\$	4	D	T	d	t				
						5	ENQ	NAK	%	5	E	U	e	u				
						6	ACK	SYN	&	6	F	V	f	v				
						7	BEL	ETB	'	7	G	W	g	w				
						8	BS	CAN	(	8	H	X	h	x				
						9	HT	EM	)	9	I	Y	i	y				
						A	LF	SUB	*	:	J	Z	j	z				
					B	VT	ESC	+	;	K	[	k	{					
					C	FF	FS	,	<	L	\	l						
					D	CR	GS	-	=	M	]	m	}					
					E	SO	RS	.	>	N	^	n	~					
					F	SI	US	/	?	O	_	o	DEL					

### Legend

ENQ-Enquiry	ETX-End of text	SI-Shift in	RS-Record separator
ACK-Acknowledge	EOT-End of transmission	DLE-Data link escape	US-Unit separator
BEL-Bell	ETB-End of transmission block	DC1-DC4 - Device controls	LF-Line feed
BS-Back space	CAN-Cancel	NAK-Negative acknowledge	VT-Vertical tab
HT-Horizontal tab	EM-End of media	SYN-Synchronize	FF-Form feed
NUL-Null	SUB-Substitute	FS-File separator	CR-Carriage return
SOH-Start of heading	ESC-Escape	GS-Group separator	SO-Shift out
STX-Start of text			



BAUDOT

CHARACTERS

CCITT NO. 2  
UPPER CASE \*

LOWER CASE	UPPER CASE	CASE COMM WEATHER	START	1	2	3	4	5	STOP	
A	-	↑		MARKING PULSE	MARKING PULSE				MARKING PULSE	
B	?	⊕		MARKING PULSE			MARKING PULSE	MARKING PULSE	MARKING PULSE	
C	:	○			MARKING PULSE	MARKING PULSE	MARKING PULSE		MARKING PULSE	
D	\$	↗		MARKING PULSE			MARKING PULSE		MARKING PULSE	WRU
E	3	3		MARKING PULSE			MARKING PULSE		MARKING PULSE	
F	!	→		MARKING PULSE			MARKING PULSE		MARKING PULSE	UNASSIGNED
G	&	↘			MARKING PULSE		MARKING PULSE	MARKING PULSE	MARKING PULSE	UNASSIGNED
H	STOP	↓					MARKING PULSE		MARKING PULSE	UNASSIGNED
I	8	8			MARKING PULSE	MARKING PULSE			MARKING PULSE	
J	'	↙		MARKING PULSE	MARKING PULSE		MARKING PULSE		MARKING PULSE	AUDIBLE SIGNAL
K	(	←			MARKING PULSE	MARKING PULSE	MARKING PULSE		MARKING PULSE	
L	)	↖			MARKING PULSE		MARKING PULSE		MARKING PULSE	
M	.	.				MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	
N	,	⊖				MARKING PULSE	MARKING PULSE		MARKING PULSE	
O	9	9					MARKING PULSE	MARKING PULSE	MARKING PULSE	
P	∅	∅			MARKING PULSE	MARKING PULSE			MARKING PULSE	
Q	1	1		MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE		MARKING PULSE	
R	4	4			MARKING PULSE		MARKING PULSE		MARKING PULSE	
S	BELL	BELL		MARKING PULSE					MARKING PULSE	' (APOSTROPHE)
T	5	5						MARKING PULSE	MARKING PULSE	
U	7	7		MARKING PULSE	MARKING PULSE	MARKING PULSE			MARKING PULSE	
V	;	⊙			MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	=
W	2	2		MARKING PULSE	MARKING PULSE				MARKING PULSE	
X	/	/				MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	
Y	6	6		MARKING PULSE		MARKING PULSE		MARKING PULSE	MARKING PULSE	
Z	"	+		MARKING PULSE				MARKING PULSE	MARKING PULSE	+
BLANK		-							MARKING PULSE	
SPACE						MARKING PULSE			MARKING PULSE	
CAR. RET.							MARKING PULSE		MARKING PULSE	
LINE FEED					MARKING PULSE				MARKING PULSE	
FIGURES				MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	
LETTERS				MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	MARKING PULSE	

NOTE: UPPER CASE H (COMM) MAY BE STOP OR #

 MARKING PULSE  
 SPACING PULSE

\*This Column Shows Only Those Characters Which Differ From The U.S.A. Variation

## EBCDIC CONVERSION

<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>	<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>
NUL	00000000	000	000	00	NOTE	00101001	051	041	29
SOH	00000001	001	001	01	SM	00101010	052	042	2A
STX	00000010	002	002	02	CU2	00101011	053	043	2B
ETX	00000011	003	003	03	NOTE	00101100	054	044	2C
PF	00000100	004	004	04	ENQ	00101101	055	045	2D
HT	00000101	005	005	05	ACK	00101110	056	044	2E
LC	00000110	006	006	06	BEL	00101111	057	045	2F
DEL	00000111	007	007	07	NOTE	00110000	060	048	30
NOTE	00001000	010	008	08	NOTE	00110001	061	049	31
RLF	00001001	011	009	09	SYN	00110010	062	050	32
SMM	00001010	012	010	0A	NOTE	00110011	063	051	33
VT	00001011	013	011	0B	PN	00110100	064	052	34
FF	00001100	014	012	0C	RS	00110101	065	053	35
CR	00001101	015	013	0D	UC	00110110	066	054	34
SO	00001110	016	014	0E	EOT	00110111	067	055	37
SI	00001111	017	015	0F	NOTE	00111000	070	056	38
DLE	00010000	020	016	10	NOTE	00111001	071	057	39
DC1	00010001	021	017	11	NOTE	00111010	072	058	3A
DC2	00010010	022	018	12	CU3	00111011	073	059	3B
DC3	00010011	023	019	13	DC4	00111100	074	060	3C
RES	00010100	024	020	14	NAK	00111101	075	061	3D
NL	00010101	025	021	15	SUB	00111110	076	062	3E
BS	00010110	026	022	16	SB	00111111	077	063	3F
IL	00010111	027	023	17	NOTE	01000000	100	064	40
CAN	00011000	030	024	18	NOTE	01000001	101	065	41
EM	00011001	031	025	19	NOTE	01000010	102	066	42
CC	00011010	032	026	1A	NOTE	01000011	103	067	43
CU1	00011011	033	027	1B	NOTE	01000100	104	068	44
IFS	00011100	034	028	1C	NOTE	01000101	105	069	45
IGS	00011101	035	029	1D	NOTE	01000110	106	070	46
IRS	00011110	036	030	1E	NOTE	01000111	107	071	47
IUS	00011111	037	031	1F	NOTE	01001000	110	072	48
DS	00100000	040	032	20	NOTE	01001001	111	073	49
SOS	00100001	041	033	21	⊕	01001010	112	074	4A
FS	00100010	042	034	22	.	01001011	113	075	4B
NOTE	00100011	043	035	23	<	01001100	114	076	4C
BYP	00100100	044	036	24	(	01001101	115	077	4D
LF	00100101	045	037	25	+	01001110	116	078	4E
EOB/ETB	00100110	046	038	26		01001111	117	079	4F
ESC/PRE	00100111	047	039	27	&	01010000	120	080	50
NOTE	00101000	050	040	28	NOTE	01010001	121	081	51

EBCDIC CONVERSION (Cont'd)

<u>KEY</u>	<u>BINARY</u> 87654321	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>	<u>KEY</u>	<u>BINARY</u> 87654321	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>
NOTE	01010010	122	082	52	NOTE	10000000	200	128	80
NOTE	01010011	123	083	53	a	10000001	201	129	81
NOTE	01010100	124	084	54	b	10000010	202	130	82
NOTE	01010101	125	085	55	c	10000011	203	131	83
NOTE	01010110	126	086	56	d	10000100	204	132	84
NOTE	01010111	127	087	57	e	10000101	205	133	85
NOTE	01011000	130	088	58	f	10000110	206	134	86
NOTE	01011001	131	089	59	g	10000111	207	135	87
!	01011010	132	090	5A	h	10001000	210	136	88
\$	01011011	133	091	5B	i	10001001	211	137	89
*	01011100	134	092	5C	NOTE	10001010	212	138	8A
)	01011101	135	093	5D	NOTE	10001011	213	139	8B
;	01011110	136	094	5E	NOTE	10001100	214	140	8C
┌	01011111	137	095	5F	NOTE	10001101	215	141	8D
-	01100000	140	096	60	NOTE	10001110	216	142	8E
/	01100001	141	097	61	NOTE	10001111	217	143	8F
NOTE	01100010	142	098	62	NOTE	10010000	220	144	90
NOTE	01100011	143	099	63	j	10010001	221	145	91
NOTE	01100100	144	100	64	k	10010010	222	146	92
NOTE	01100101	145	101	65	l	10010011	223	147	93
NOTE	01100110	146	102	66	m	10010100	224	148	94
NOTE	01100111	147	103	67	n	10010101	225	149	95
NOTE	01101000	150	104	68	o	10010110	226	150	96
NOTE	01101001	151	105	69	p	10010111	227	151	97
!	01101010	152	106	6A	q	10011000	230	152	98
,	01101011	153	107	6B	r	10011001	231	153	99
%	01101100	154	108	6C	NOTE	10011010	232	154	9A
	01101101	155	109	6D	NOTE	10011011	233	155	9B
>	01101110	156	110	6E	NOTE	10011100	234	156	9C
?	01101111	157	111	6F	NOTE	10011101	235	157	9D
NOTE	01110000	160	112	70	NOTE	10011110	236	158	9E
NOTE	01110001	161	113	71	NOTE	10011111	237	159	9F
NOTE	01110010	162	114	72	NOTE	10100000	240	160	A0
NOTE	01110011	163	115	73	~	10100001	241	161	A1
NOTE	01110100	164	116	74	s	10100010	242	162	A2
NOTE	01110101	165	117	75	t	10100011	243	163	A3
NOTE	01110110	166	118	76	u	10100100	244	164	A4
NOTE	01110111	167	119	77	v	10100101	245	165	A5
NOTE	01111000	170	120	78	w	10100110	246	164	A6
▲	01111001	171	121	79	x	10100111	247	167	A7
:	01111010	172	122	7A	y	10101000	250	168	A8
#	01111011	173	123	7B	z	10101001	251	169	A9
@	01111100	174	124	7C	NOTE	10101010	252	170	AA
▲	01111101	175	125	7D	NOTE	10101011	253	171	AB
=	01111110	176	126	7E	NOTE	10101100	254	172	AC
"	01111111	177	127	7F	NOTE	10101101	255	173	AD

## EBCDIC CONVERSION (Cont'd)

<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>	<u>KEY</u>	<u>BINARY</u> <u>87654321</u>	<u>OCT</u>	<u>DEC</u>	<u>HEX</u>
NOTE	10101110	256	174	AE	P	11010111	327	215	D7
NOTE	10101111	257	175	AF	Q	11011000	330	216	D8
NOTE	10110000	260	176	B0	R	11011001	331	217	D9
NOTE	10110001	261	177	B1	NOTE	11011010	332	218	DA
NOTE	10110010	262	178	B2	NOTE	11011011	333	219	DB
NOTE	10110011	263	179	B3	NOTE	11011100	334	220	DC
NOTE	10110100	264	180	B4	NOTE	11011101	335	221	DD
NOTE	10110101	265	181	B5	NOTE	11011110	336	222	DE
NOTE	10110110	266	182	B6	NOTE	11011111	337	223	DF
NOTE	10110111	267	183	B7	\	11100000	340	224	E0
NOTE	10111000	270	184	B8	NOTE	11100001	341	225	E1
NOTE	10111001	271	185	B9	S	11100010	342	226	E2
NOTE	10111010	272	186	BA	T	11100011	343	227	E3
NOTE	10111011	273	187	BB	U	11100100	344	228	E4
NOTE	10111100	274	188	BC	V	11100101	345	229	E5
NOTE	10111101	275	189	BD	W	11100110	346	230	E6
NOTE	10111110	276	190	BE	X	11100111	347	231	E7
NOTE	10111111	277	191	BF	Y	11101000	350	232	E8
{	11000000	300	192	C0	Z	11101001	351	233	E9
A	11000001	301	193	C1	NOTE	11101010	352	234	EA
B	11000010	302	194	C2	NOTE	11101011	353	235	EB
C	11000011	303	195	C3	┐	11101100	354	236	EC
D	11000100	304	196	C4	NOTE	11101101	355	237	ED
E	11000101	305	197	C5	NOTE	11101110	356	238	EE
F	11000110	306	198	C6	NOTE	11101111	357	239	EF
G	11000111	307	199	C7	0	11110000	360	240	F0
H	11001000	310	200	C8	1	11110001	361	241	F1
I	11001001	311	201	C9	2	11110010	362	242	F2
NOTE	11001010	312	202	CA	3	11110011	363	243	F3
NOTE	11001011	313	203	CB	4	11110100	364	244	F4
┌	11001100	314	204	CC	5	11110101	365	245	F5
NOTE	11001101	315	205	CD	6	11110110	366	246	F6
└	11001110	316	206	CE	7	11110111	367	247	F7
NOTE	11001111	317	207	CF	8	11111000	370	248	F8
}	11010000	320	208	D0	9	11111001	371	249	F9
J	11010001	321	209	D1	NOTE	11111010	372	250	FA
K	11010010	322	210	D2	NOTE	11111011	373	251	FB
L	11010011	323	211	D3	NOTE	11111100	374	252	FC
M	11010100	324	212	D4	NOTE	11111101	375	253	FD
N	11010101	325	213	D5	NOTE	11111110	376	254	FE
O	11010110	326	214	D6	NOTE	11111111	377	255	FF



**CONTROL CHARACTER ABBREVIATIONS**

NUL	-	NULL	IFS	-	INFORMATION FIELD SEPARATOR
SOH	-	START OF HEADING	IGS	-	INFORMATION GROUP SEPARATOR
STX	-	START OF TEXT	IRS	-	INFORMATION RECORD SEPARATOR
ETX	-	END OF TEXT	IUS	-	INFORMATION UNIT SEPARATOR
PF	-	PUNCH OFF	DS	-	DIGIT SELECT
HT	-	HORIZONTAL TAB	SOS	-	START OF SIGNIFICANCE
LC	-	LOWER CASE	FS	-	FIELD SEPARATOR
DEL	-	DELETE	BYP	-	BY PASS
RLF	-	REVERSE LINE FEED	LF	-	LINE FEED
SMM	-	START OF MANUAL MESSAGE	EOB/	-	END OF BLOCK/
VT	-	VERTICAL TAB	ETB	-	END OF TRANSMISSION BLOCK
FF	-	FORM FEED	ESC/	-	ESCAPE/
CR	-	CARRIAGE RETURN	PRE	-	PREFIX
SO	-	SHIFT OUT	SM	-	SET MODE
SI	-	SHIFT IN	CU2	-	CUSTOMER USE 2
DLE	-	DATA LINK ESCAPE	ENQ	-	ENQUIRY
DC1	-	DEVICE CONTROL 1	ACK	-	ACKNOWLEDGE
DC2	-	DEVICE CONTROL 2	BEL	-	BELL
DC3	-	DEVICE CONTROL 3	SYN	-	SYNCHRONOUS
RES	-	RESTORE	PN	-	PUNCH OUT
NL	-	NEW LINE	RS	-	READER STOP
BS	-	BACKSPACE	UC	-	UPPER CASE
IL	-	IDLE	EOT	-	END OF TRANSMISSION
CAN	-	CANCEL	CU3	-	CUSTOMER USE 3
EM	-	END OF MEDIUM	DC4	-	DEVICE CONTROL 4
CC	-	CURSOR CONTROL	NAK	-	NEGATIVE ACKNOWLEDGE
CU1	-	CUSTOMER USE 1	SUB	-	SUBSTITUTE
			SP	-	SPACE
			NOTE	-	UNASSIGNED

### EXTERNAL BCD

				<table style="display: inline-table; border-collapse: collapse; margin-right: 10px;"> <tr><td style="border: none;">b8</td><td style="border: none;">→</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td></tr> <tr><td style="border: none;">b7</td><td style="border: none;">→</td><td style="border: none;">0</td><td style="border: none;">0</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">1</td><td style="border: none;">0</td></tr> <tr><td style="border: none;">b6</td><td style="border: none;">→</td><td style="border: none;">0</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">1</td></tr> <tr><td style="border: none;">b5</td><td style="border: none;">→</td><td style="border: none;">0</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">0</td><td style="border: none;">1</td><td style="border: none;">1</td></tr> </table>										b8	→	1	0	1	0	1	0	1	0	1	0	1	0	1	0	b7	→	0	0	0	1	1	0	1	0	1	1	0	1	1	0	b6	→	0	0	1	0	1	0	1	0	1	0	1	0	1	1	b5	→	0	0	1	0	1	0	1	0	1	0	1	0	1	1	b <sub>8</sub> =1		b <sub>8</sub> =0	
b8	→	1	0	1	0	1	0	1	0	1	0	1	0	1	0																																																																		
b7	→	0	0	0	1	1	0	1	0	1	1	0	1	1	0																																																																		
b6	→	0	0	1	0	1	0	1	0	1	0	1	0	1	1																																																																		
b5	→	0	0	1	0	1	0	1	0	1	0	1	0	1	1																																																																		
b4	b3	b2	b1	1st HEX DIGIT → 2nd HEX DIGIT ↓	8	9	A	B	C	D	E	F	7																																																																				
LBS					0	1	2	3	4	5	6	7																																																																					
0	0	0	0	0	NUL	:	-	+	@	SP		p																																																																					
0	0	0	1	1	SOH	DC1	J	A	1	/	a	q																																																																					
0	0	1	0	2	STX	DC2	K	B	2	S	b	r																																																																					
0	0	1	1	3	ETX	DC3	L	C	3	T	c	s																																																																					
0	1	0	0	4	EOT	DC4	M	D	4	U	d	t																																																																					
0	1	0	1	5	ENQ	NAK	N	E	5	V	e	u																																																																					
0	1	1	0	6	ACK	SYN	O	F	6	W	f	v																																																																					
0	1	1	1	7	BEL	ETB	P	G	7	X	g	w																																																																					
1	0	0	0	8	BS	CAN	Q	H	8	Y	h	x																																																																					
1	0	0	1	9	HT	EM	R	I	9	Z	i	y																																																																					
1	0	1	0	A	LF	SUB	!	<	∅	]	j	z																																																																					
1	0	1	1	B	VT	ESC	\$	.	=	,	k	{																																																																					
1	1	0	0	C	FF	FS	*	)	"	(	l																																																																						
1	1	0	1	D	CR	GS	'	\	@	-	m	}																																																																					
1	1	1	0	E	SO	RS	?	ESC	%	#	n	~																																																																					
1	1	1	1	F	SI	US	>	;	[	&	o	DEL																																																																					

#### Legend

ENQ - Enquiry  
 ACK - Acknowledge  
 BEL - Bell  
 BS - Back space  
 HT - Horizontal tab  
 NUL - Null  
 SOH - Start of heading  
 STX - Start of test

ETX - End of text  
 EOT - End of transmission  
 ETB - End of transmission block  
 CAN - Cancel  
 EM - End of media  
 SUB - Substitute  
 ESC - Escape

SI - Shift in  
 DLE - Data link escape  
 DC1-DC4 - Device controls  
 NAK - Negative acknowledge  
 SYN - Synchronize  
 FS - File separator  
 GS - Group separator

RS - Record separator  
 US - Unit separator  
 LF - Line feed  
 VT - Vertical tab  
 FF - Form feed  
 CR - Carriage return  
 SO - Shift out

### EBCD CORRESPONDENCE (SELECTRIC)

REVERSE HEX				b7-c →	0	0	0	0	1	1	1	1
				b6-1 →	0	0	1	1	0	0	1	1
				b5-2 →	0	1	0	1	0	1	0	1
				1st HEX DIGIT →								
b4-4	b3-8	b2-A	b1-B	2nd HEX DIGIT ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	SP/SP	\$/4	T/+	L/l	0/!	O/o	J/j	?//
0	0	0	1	1	SP/SP	\$/4	T/t	L/l	0/!	O/o	J/j	?//
0	0	1	0	2	+/1	)/0	X/x	H/h	M/m	S/s	G/g	Y/y
0	0	1	1	3	+/1	)/0	X/x	H/h	M/m	S/s	G/g	Y/y
0	1	0	0	4	@/2	Z/z	N/n	VT/FF	./.	:/<	+/=	~/>
0	1	0	1	5	@/2	Z/z	N/n	VT/FF	./.	:/<	+/=	~/>
0	1	1	0	6	#/3	(/9	U/u	B/b	V/v	W/w	F/f	-/-
0	1	1	1	7	#/3	(/9	U/u	B/b	V/v	W/w	F/f	-/-
1	0	0	0	8	%/5	PN/PN	E/e	BY/BY	"/'	RES/RES	P/P	PF/PF
1	0	0	1	9	%/5	PN/PN	E/e	BY/BY	"/'	RES/RES	P/p	PF/PF
1	0	1	0	A	&/7	RS/RS	D/d	LF/LF	R/r	NL/NL	;/;	HT/HT
1	0	1	1	B	&/7	RS/RS	D/d	LF/LF	R/r	NL/NL	;/;	HT/HT
1	1	0	0	C	¢/6	UC/UC	K/k	EOB/EOB	I/i	BS/BS	Q/q	LC/LC
1	1	0	1	D	¢/6	UC/UC	K/k	EOB/EOB	I/i	BS/BS	Q/q	LC/LC
1	1	1	0	E	*/8	EOT/EOT	C/c	PRE/PRE	A/a	IL/IL	,/,	DEL/DEL
1	1	1	1	F	*/8	EOT/EOT	C/c	PRE/PRE	A/a	IL/IL	,/,	DEL/DEL

#### LEGEND

UPPER CASE / LOWER CASE

1=MARK  
0=SPACE

b7=PARITY CHECK BIT  
SHADED BLOCKS=ODD PARITY

PN ○ PUNCH ON  
 BY ⊗ BYPASS  
 RES ⊙ RESTORE  
 PF ⊗ PUNCH OFF  
 RS □ READER STOP  
 LF ≡ LINE FEED  
 VT ) VERTICAL TAB

NL ⇐ NEW LINE  
 (Carrier Return and line feed)  
 HT ⇒ HORIZONTAL TAB  
 UC ^ UPPER CASE  
 EOB ⊥ END OF BLOCK  
 FF ≡ FORM FEED

BS ↵ BACK SPACE  
 LC // LOWER CASE  
 EOT ⏏ END OF TRANSMISSION  
 PRE ⏏ PREFIX  
 IL ⏏ IDLE  
 DEL ⏏ DELETE

### EBCD 1050 (STANDARD ENCORE BCD)

REVERSE HEX				b7-c →	0	0	0	0	1	1	1	1
				b6-1 →	0	0	1	1	0	0	1	1
				b5-2 →	0	1	0	1	0	1	0	1
				1st HEX DIGIT →								
b4-4	b3-8	b2-A	b1-B	2nd HEX DIGIT ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	SP/SP	* / 8	¢ / @	Y / y	— / —	Q / q	+ / &	H / h
0	0	0	1	1	SP/SP	* / 8	¢ / @	Y / y	— / —	Q / q	+ / &	H / h
0	0	1	0	2	= / 1	( / 9	? / /	Z / z	J / j	R / r	A / a	I / i
0	0	1	1	3	= / 1	( / 9	? / /	Z / z	J / j	R / r	A / a	I / i
0	1	0	0	4	¿ / 2	) / 0	S / s	VT / FF	K / k	¡ / <	B / b	~ / >
0	1	0	1	5	¿ / 2	) / 0	S / s	VT / FF	K / k	¡ / <	B / b	~ / >
0	1	1	0	6	; / 3	+ / #	T / t	, / ,	L / l	! / \$	C / c	./ / .
0	1	1	1	7	; / 3	+ / #	T / t	, / ,	L / l	! / \$	C / c	./ / .
1	0	0	0	8	¡ / 4	PN / PN	U / u	BY / BY	M / m	RES / RES	D / d	PF / PF
1	0	0	1	9	¡ / 4	PN / PN	U / u	BY / BY	M / m	RES / RES	D / d	PF / PF
1	0	1	0	A	% / 5	RS / RS	V / v	LF / LF	N / n	NL / NL	E / e	HT / HT
1	0	1	1	B	% / 5	RS / RS	V / v	LF / LF	N / n	NL / NL	E / e	HT / HT
1	1	0	0	C	' / 6	UC / UC	W / w	EOB / EOB	O / o	BS / BS	F / f	LC / LC
1	1	0	1	D	' / 6	UC / UC	W / w	EOB / EOB	O / o	BS / BS	F / f	LC / LC
1	1	1	0	E	" / 7	EOT / EOT	X / x	PRE / PRE	P / p	IL / IL	G / g	DEL / DEL
1	1	1	1	F	" / 7	EOT / EOT	X / x	PRE / PRE	P / p	IL / IL	G / g	DEL / DEL

**NOTE:** The correct case shift character must be entered for proper display of data. \ = lower case, ^ = upper case.

#### LEGEND

UPPER  
CASE / LOWER  
CASE

1=MARK  
0=SPACE

b7=PARITY CHECK BIT  
SHADED BLOCKS=ODD PARITY

PN ○ PUNCH ON  
 BY ∞ BYPASS  
 RES ⊙ RESTORE  
 PF ⊗ PUNCH OFF  
 RS □ READER STOP  
 LF ≡ LINE FEED  
 VT ⊓ VERTICAL TAB

NL ⇐ NEW LINE  
 HT ⇨ HORIZONTAL TAB  
 UC ^ UPPER CASE  
 EOB ⊥ END OF BLOCK  
 FF ⊓ FORM FEED

BS ↶ BACK SPACE  
 LC ↷ LOWER CASE  
 EOT ↘ END OF TRANSMISSION  
 PRE ⊥ PREFIX  
 IL ⊥ IDLE  
 DEL ⊘ DELETE

**IPARS (SABRE)**

				b6 →	0	0	1	1
				b5 →	0	1	0	1
				1st HEX →				
b4	b3	b2	b1	2nd HEX ↓	0	1	2	3
0	0	0	0	0	NUL	NOP	@	'
0	0	0	1	1	1	/	J	A
0	0	1	0	2	2	S	K	B
0	0	1	1	3	3	T	L	C
0	1	0	0	4	4	U	M	D
0	1	0	1	5	5	V	N	E
0	1	1	0	6	6	W	O	F
0	1	1	1	7	7	X	P	G
1	0	0	0	8	8	Y	Q	H
1	0	0	1	9	9	Z	R	I
1	0	1	0	A	∅	-	LIGHTS	?
1	0	1	1	B	*	,	\$	.
1	1	0	0	C	CR	SP	HT	BS
1	1	0	1	D	EOM1	EOMC	EOMV	EOM
1	1	1	0	E	CR OUT	SEGID	TIC !	SYN 2
1	1	1	1	F	GA	RGA	TIC 2	SYN 1

**NOTE:**

1. Use CRC-12 with IPARS.
2. Typically, runs in idle space condition i.e., HEX ∅∅.
3. Requires special front end program, when used with ENCORE.

**XS-3**

**(NORMALLY ODD PARITY)**

				PARITY	0	0	0	0	1	1	1	1
				b6 →	0	0	1	1	0	0	1	1
				b5 →	0	1	0	1	0	1	0	1
				1st HEX →								
b4	b3	b2	b1	2nd HEX ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	SOM	&	'					∅
0	0	0	1	1				%	□	:	*	∅
0	0	1	0	2	-							
0	0	1	1	3		?	!		∅	.	\$	
0	1	0	0	4	1			/		A	J	
0	1	0	1	5		B	K	SYN	2	EOB/EOM		S
0	1	1	0	6		C	L		3			T
0	1	1	1	7	4			U		D	M	
1	0	0	0	8	5			V		E	N	
1	0	0	1	9		F	O		6			W
1	0	1	0	A		G	P		7			X
1	0	1	1	B	8			Y		H	Q	
1	1	0	0	C		I	R		9			Z
1	1	0	1	D	/					#	(	
1	1	1	0	E	;			>		<	@	
1	1	1	1	F		=	BEL		⌂			)

- NOTES:** All characters are odd parity except:  
 SOM (Start of Message)  
 SYN (SYNC)  
 EOB/EOM (End of Block/End of Message)

REVERSE ASCII  
(GRTS/355)

				bit 7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
				bit 6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				bit 5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				bit 4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
bit 3	bit 2	bit 1	Parity	1st HEX →	∅	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
				2nd HEX ↓					STX				SOH							
0	0	0	0	∅					STX				SOH							
0	0	0	1	1													ETX			
0	0	1	0	2	@			L		J	F			I	E		C			O
0	0	1	1	3		H	D		B		*	&		N	A		M		K	G
0	1	0	0	4	Blank			,		*	&			)	%		#			/
0	1	0	1	5		(	\$		"				.	!			-		+	'
0	1	1	0	6																
0	1	1	1	7																
1	0	0	0	8									SYN							
1	0	0	1	9																
1	0	1	0	A		X	T		R			ç	Q			]		[		W
1	0	1	1	B		P		\			Z	V			Y	U		S		-
1	1	0	0	C		8	4		2				>	1			=		;	7
1	1	0	1	D		∅		<			:	6			9	5		3		?
1	1	1	0	E																
1	1	1	1	F																

LEGEND

SOH - Start of heading  
STX - Start of text

ETX - End of text  
SYN - Synchronizing character

UNIVAC FIELDATA

				b7 →	0	0	0	0
				b6 →	0	0	1	1
				b5 →	0	1	0	1
				1st HEX DIGIT →				
b4	b3	b2	b1	2nd HEX DIGIT ↓	0	1	2	3
0	0	0	0	0	@	K	)	0
0	0	0	1	1	[	L	-	1
0	0	1	0	2	J	M	+	2
0	0	1	1	3	#	N	<	3
0	1	0	0	4	△	O	=	4
0	1	0	1	5	SP	P	>	5
0	1	1	0	6	A	Q	&	6
0	1	1	1	7	B	R	\$	7
1	0	0	0	8	C	S	*	8
1	0	0	1	9	D	T	(	9
1	0	1	0	A	E	U	%	'
1	0	1	1	B	F	V	:	;
1	1	0	0	C	G	W	?	/
1	1	0	1	D	H	X	!	.
1	1	1	0	E	I	Y	,	□
1	1	1	1	F	J	Z	\	≠

1 = MARK

0 = SPACE

RMS

(NORMALLY ODD PARITY)

				PARITY	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1	1st HEX								
				2nd HEX	0	1	2	3	4	5	6	7
0	0	0	0	0	SPACE	&	'	%	SPACE	:	*	NK
0	0	0	1	1				,	□	.	\$	
0	0	1	0	2	-							
0	0	1	1	3		?	!		0			+
0	1	0	0	4	1			/		A	J	
0	1	0	1	5		B	K	S	2	B		S
0	1	1	0	6		C	L		3		T	
0	1	1	1	7	4			U		D	M	
1	0	0	0	8	5			V		E	N	
1	0	0	1	9		F	O		6			W
1	0	1	0	A		G	P		7			X
1	0	1	1	B	8			Y		H	Q	
1	1	0	0	C		I	R		9			Z
1	1	0	1	D				-		#	(	
1	1	1	0	E				<		>	@	
1	1	1	1	F		=	BL			□	)	

NOTES: 1. Control characters are normally even parity.  
2. Data characters are odd parity.

**APPENDIX E**

**X25/X75**

**SCENARIO AND LINK TEST INSTRUCTIONS**



**X25/X75 SCENARIO INSTRUCTION SET**

The instructions that follow are designed for use in the interactive simulate and scenario driven monitor modes of operation. They allow the user to write scenarios for the operation of the ENCORE as DCE or DTE in the X25/X75 environment. Additional information needed to write and execute these scenarios is found in Chapter 8.

**INDEX**

<b>NO.</b>	<b>INSTRUCTION</b>	<b>PAGE NO.</b>	<b>NO.</b>	<b>INSTRUCTION</b>	<b>PAGE NO.</b>
1	FOR	8-65	**17	[PR]-1	8-67
2	NEXT	8-65	**18	[PS]+1	8-67
3	SET-TIMER	8-65	**19	[PS]-1	8-67
4	CLEAR-TIMER	8-66	**20	[PR]-CLEAR	8-68
5	WAIT-FOR	8-66	**21	[PS]-CLEAR	8-68
6	RECEIVE-ONLY	8-66	22	HALT	8-68
**7	TRANSMIT	8-66	23	SET WINDOW	8-68
8	SET-Q-BIT	8-66	**24	BAD-FCS-NEXT	8-68
9	SET-D-BIT	8-66	25	MARKER	8-68
*7	DTE ENABLE	8-66	26	GOSUB-MARKER-#	8-68
*8	DCE ENABLE	8-66	27	ERROR ENTRY	8-69
*9	BOTH ENABLE	8-66	28	ALARM SOUND	8-69
10	SET-LCGN	8-66	29	CLEAR-COUNT-#	8-69
11	SET-LCN	8-67	30	COUNT-COUNTER-#	8-69
**12	SET-M-BIT	8-67	31	DISPLAY-COUNT-#	8-69
13	PAUSE	8-67	32	CLEAR-SECONDS	8-69
**14	SET-N2	8-67	33	SECONDS > COUNT	8-69
**15	SET-T1	8-67	34	LCN-1	8-69
**16	[PR]-1	8-67	35	LCN+1	8-69

## INDEX (Cont'd)

NO.	INSTRUCTION	PAGE NO.	NO.	INSTRUCTION	PAGE NO.
36	LOG-COUNT-#	8-70	49	SKIP-UNLESS-LAST =	8-71
37	BUFFER-CLEAR/OPEN	8-70	50	LOG-COMMENT	8-71
38	TURN-OFF-LAW	8-70	51	!CHEAT-LINK-UP!	8-71
39	TURN-ON-LAW	8-70	**52	DCE-LAP-MOD	8-71
40	TRANSFER-TO-SCENARIO-0	8-70	**53	AUTO RESPONSE	8-71
41	LAPB	8-70	54	SKIP-IF-LINK-DOWN	8-71
**42	LINK-DISC	8-70	55	SKIP-IF-LCN-MATCH	8-71
43	WAIT-FOR-LINK-UP	8-70	56	SET-[STAT] =	8-72
44	ALL LCN'S	8-70	57	SKIP-UNLESS-[STAT] =	8-72
45	RETURN	8-70	58	NOT-ALL-LCN'S	8-72
46	GOTO-MARKER-#	8-70	59	CLEAR STATS	8-72
47	COPY-LAST-LCN	8-70	60	MODULO-8	8-72
48	RECEIVE-ANY	8-70	61	MODULO-128	8-72

**NOTES:** \* Used in Scenario Driven Monitor Mode only  
 \*\* Not used in Scenario Driven Monitor Mode



## LINK TEST INSTRUCTIONS

These instructions are designed for use in the LINK-TESTER Mode. There are 29 instructions, each represented by a single alpha-numeric keystroke. The instructions are executed immediately, as entered, or sequentially in a string of up to 62 characters or instructions. During auto-run, or sequential execution, the left-most character in the string is executed first. When the last instruction is executed, the sequence is repeated. While in the Link Test Mode, the complete instruction set is displayed on the CRT by striking the 7 key. Detailed information for each instruction can be found in Chapter 8.

## INDEX

INSTRUCTION	PAGE NO.	INSTRUCTION	PAGE NO.
Address invert	8-79	Set N(R) = 0	8-79
SABM transmit	8-79	Set N(R) = 1	8-80
CMDR transmit	8-79	Set N(R) = 2	8-80
DISC transmit	8-79	Set N(R) = 3	8-80
INFO transmit	8-79	Set N(R) = 4	8-80
REJ transmit	8-79	Set N(R) = 5	8-80
SARM transmit	8-79	Set N(R) = 6	8-80
DM transmit	8-79	Set N(R) = 7	8-80
RNR transmit	8-79	N(S)+1	8-80
P/F toggle	8-79	N(S)-1	8-80
RR transmit	8-79	Open string	8-80
UA transmit	8-79	Character delete	8-80
Wait $\frac{1}{4}$ sec	8-79	Auto-run	8-80
DTE/DCE/BRG	8-79	Auto-run-stop	8-80
N(S) + N(R) = 0	8-79		



**APPENDIX F**  
**FORMS AND RECORDS**

**PROGRAM WORKSHEETS**

In this appendix you will find several program worksheets that can be easily removed from the manual. These are provided for your convenience when writing new programs. You are invited to submit copies of these worksheets to DIGITECH so that we may make them available to other users.









## ALPHABETICAL INDEX

<u>SUBJECT</u>	<u>Page</u>
ABORT. . . . .	7-61
ABORT ALL. . . . .	7-61
ABORT FULL . . . . .	7-61
ABORT KBD. . . . .	7-61
ABORT PIN . . . . .	7-61
ABORT P2. . . . .	7-61
ABORT P3. . . . .	7-61
ABORT RTC. . . . .	7-61
ABORT TIMER. . . . .	7-61
ABORT XIOIN. . . . .	7-61
ADD ONE COUNT TO COUNTER. . . . .	6-21
Addresses . . . . .	8-35, 8-75
Address Field . . . . .	8-4
ALARM SOUND . . . . .	8-70
ALL-LCN'S . . . . .	8-71
AND . . . . .	7-21
Applications Assistance. . . . .	9-7
ARGUMENT CHAIN . . . . .	6-19
ASGN. . . . .	7-2
Asynchronous Library. . . . .	5-3
Attribute Byte. . . . .	7-41
Audible Alarm . . . . .	8-30
Audio Output . . . . .	3-19
AUTO RESPONSE . . . . .	8-72
Auxiliary Packet Information. . . . .	8-76

# INDEX

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
BAD-FCS-NEXT . . . . .	8-69
Bert . . . . .	5-24
Bit Oriented Framing (HDLC/SDLC) . . . . .	8-4
Bit Oriented Library . . . . .	5-13
Block Check Calculations . . . . .	7-46
Block Check Sequence. . . . .	8-3
BLOCK ERROR RATE TEST, SKIP ON ERROR. . . . .	6-15
BOOT. . . . .	7-62
BOTH-ENABLE . . . . .	8-67
BUFFER-CLEAR/OPEN. . . . .	8-71
Byte Variables. . . . .	7-32
Calculate . . . . .	7-55
CAL?. . . . .	8-11
CAL!. . . . .	8-12
CALL. . . . .	7-63
Called Address Length . . . . .	8-76
Called DTE Address. . . . .	8-10
Called DTE Address Length . . . . .	8-9
Calling Address Length . . . . .	8-76
Calling DTE Address . . . . .	8-10
Calling DTE Address Length. . . . .	8-9
Call Set-Up and Clearing Packets. . . . .	8-11
Capture Data . . . . .	8-31, 8-49
Capture Data Control. . . . .	7-43
Cert . . . . .	5-25

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
CHAIN . . . . .	7-64
CHARACTER ERROR RATE TEST, SKIP ON ERROR. . . . .	6-15
Character Oriented Framing (BISYNC, BSC). . . . .	8-2
CHAR IN . . . . .	5-11
CHAR OUT . . . . .	5-11
!CHEAT-LINK-UP!. . . . .	8-72
Classification, Packet. . . . .	8-75
Cleaning . . . . .	2-4
CLEAR. . . . .	7-65
CLEAR CBUF . . . . .	7-65
CLEAR CHAIN. . . . .	7-65
CLEAR-COUNT-# . . . . .	8-70
CLEAR DISPLAY . . . . .	6-18
CLEAR MSTM. . . . .	7-65
CLEAR PARITY ERROR . . . . .	6-18
CLEAR RETURN. . . . .	7-65
CLEAR SECONDS . . . . .	8-70
CLEAR-STATS. . . . .	8-73
CLEAR TIMER. . . . .	6-22
CLEAR-TIMER. . . . .	8-67
Clock, IO . . . . .	7-9, 8-39
CLR . . . . .	7-107
CLR?. . . . .	8-12
CLR!. . . . .	8-13
CMA . . . . .	7-107

# INDEX

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
CMDR . . . . .	8-7
COMBASIC Instruction Set . . . . .	7-61
COMBASIC Programming . . . . .	7-20
Control Cluster Keys . . . . .	3-16
Control Field . . . . .	8-4
Control Lead and Count Instructions . . . . .	6-19
Controls, Displays, and Connectors . . . . .	3-11
Control Signal Assignment. . . . .	3-7
Control Signal Attributes . . . . .	3-5
COPY LAST LCN . . . . .	8-71
Count ACK/NAK's - Async . . . . .	5-27
COUNT-COUNTER-#. . . . .	8-70
Count Polls Per Minute . . . . .	5-28
CRC COMPUTATION IN TRANSPARENT BCS. . . . .	6-12
CRT Controller . . . . .	3-7
D* . . . . .	7-3
Data and Interrupt . . . . .	8-13
Data Packet. . . . .	8-13
DATA/READ/RESTORE. . . . .	7-65
Date and Time Entry . . . . .	8-82
D-Bit. . . . .	8-11, 8-36
DCE-ENABLE . . . . .	8-67
DCE-LAP-MOD . . . . .	8-72
D EDIT <i>program name</i> . . . . .	7-4
D EDIT <i>program name!</i> . . . . .	7-4

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
D FORMAT . . . . .	7-3
DICOL Instruction Set. . . . .	6-8
DIM. . . . .	7-66
DISC . . . . .	8-7
Disc Cleaning . . . . .	9-2
Disc Controller. . . . .	3-7
Disc Drive. . . . .	3-3
Disc Handling . . . . .	9-2
Diskette. . . . .	9-1
DISP . . . . .	7-66
DISP B . . . . .	7-67
DISP C . . . . .	7-67
DISP D . . . . .	7-67
DISP dBG . . . . .	7-70
DISP dBS . . . . .	7-70
DISP dE. . . . .	7-70
DISP dR. . . . .	7-71
DISP E . . . . .	7-68
DISP F . . . . .	7-68
DISP G . . . . .	7-68
DISP H . . . . .	7-68
DISP L . . . . .	7-69
Display Capture Buffer and EIA Status . . . . .	5-20
Display Capture Buffer FDUX . . . . .	5-18
Display Capture Buffer HDUX . . . . .	5-17

**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
Display Capture Buffer . . . Mode "D". . . . .	6-24
Display Captured Data . . . . .	8-41, 8-49, 8-50
DISPLAY CHARACTERS . . . . .	6-18
DISPLAY CONTENTS OF BLOCK COUNTER. . . . .	6-16
DISPLAY CONTENTS OF COUNTER . . . . .	6-21
DISPLAY-COUNT-# . . . . .	8-70
Display Data Disc . . . . .	5-19
Display Disc. . . . .	7-57
Display Disc and EIA Status . . . . .	5-21
DISPLAY FULL DUPLEX (OPTION). . . . .	6-20
DISPLAY HALF DUPLEX (OPTION). . . . .	6-20
Display Mode Commands . . . . .	6-26
DISPLAY TIMER. . . . .	6-23
DISP P . . . . .	7-69
DISP R . . . . .	7-69
DISP SA\$-SZ\$ . . . . .	7-69
DISP SIA\$-SIZ\$ . . . . .	7-70
DISP T1. . . . .	7-71
DISP T2. . . . .	7-71
DISP T4. . . . .	7-71
DISP T8. . . . .	7-71
D KILL <i>program name</i> . . . . .	7-11
DLC . . . . .	7-72
DLC A\$-Z\$ . . . . .	7-72

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
DMA Control . . . . .	3-6
D MENU . . . . .	7-13
DNZ . . . . .	7-107
D <i>program name</i> . . . . .	7-14
D <i>program name!</i> . . . . .	7-14
D SAVE. . . . .	7-15
D SAVE <i>program name</i> . . . . .	7-16
D SAVE <i>program name!</i> . . . . .	7-16
DSZ . . . . .	7-107
DTE-ENABLE . . . . .	8-67
Dump Capture Buffer to XIO Port . . . . .	5-22
Dump Current Menu to Diskette . . . . .	6-38
Dump Current Menu to PACER. . . . .	6-36
DXFR . . . . .	7-3
Echo as DCE. . . . .	5-6
Echo as DTE. . . . .	5-6
EDIT . . . . .	7-4
EDIT/ENTER Program . . . . . Mode "P" . . . . .	6-3
EDIT <i>program name</i> . . . . .	7-4
EDIT <i>program name!</i> . . . . .	7-4
EIA Timing Monitor. . . . .	5-33
ENCORE 200 Architecture. . . . .	3-5
ENCORE 200 Description . . . . .	3-2
ENCORE 200 Design Philosophy . . . . .	3-5
End Flag . . . . .	8-3



**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
ENTER CONVERSATION MODE, EXIT ON ARGUMENT. . . . .	6-15
Environmental Considerations . . . . .	2-1
Erase Capture Buffer . . . . .	8-43
ERROR-ENTRY . . . . .	8-70
Error Messages. . . . .	9-3
Errors . . . . .	8-37
Example Programs . . . . .	7-51
Execute Program . . . Mode "E". . . . .	6-3
Execution Times . . . . .	7-50
Facility Field . . . . .	8-10
Facility Field Length . . . . .	8-10
Fast Capture and Display . . . . .	8-48
FEM . . . . .	7-108
FESYN . . . . .	7-73
FLAGS . . . . .	8-35
Flag Sequence Field . . . . .	8-4
Floating Point Variables. . . . .	7-28
Flow Control and Reset. . . . .	8-15
FOR . . . . .	8-66
FOR/NEXT . . . . .	7-73
Frame Address. . . . .	8-35
Frame and Packet Definitions . . . . .	8-1
Frame Check Sequence (FCS) . . . . .	8-7, 8-36
Frame Delay. . . . .	8-31
Frame Level Header . . . . .	8-35

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
Frame Types. . . . .	8-36
Front End Programming. . . . .	7-107
Front Ends . . . . .	7-48
Front End 2/3 . . . . .	3-8
General Format Identifier. . . . .	8-8
GOSUB-MARKER-# . . . . .	8-66
GOSUB/RETURN. . . . .	7-73
GOTO <i>line number</i> . . . . .	7-74
GOTO-MARKER-#. . . . .	8-68
HALT. . . . .	8-66
HELP. . . . .	7-8
HEX . . . . .	7-108
HEX ID. . . . .	8-72
How to Write a Scenario. . . . .	8-55
IF . . . . .	7-74
IF A-Z . . . . .	7-74
IF A\$-Z\$ . . . . .	7-75
IF a-z. . . . .	7-75
IF FA-FZ . . . . .	7-75
IF " <i>program name</i> ". . . . .	7-75
Information Field. . . . .	8-7
Information Frames. . . . .	8-5
Information Transfer (LAP and LAPB). . . . .	8-21
Initialize Instructions. . . . .	6-8
Initial Power-Up Procedure . . . . .	4-3

**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
Initial Operation (Power-Up) . . . . .	8-23
Initial Set-Up . . . . .	4-1
INP. . . . .	7-108
INPUT . . . . .	7-76
INPUT A-Z . . . . .	7-76
INPUT A\$-Z\$ . . . . .	7-77
INPUT a-z. . . . .	7-76
INPUT FA-FZ . . . . .	7-76
Inspection. . . . .	2-4
Installation . . . . .	2-4
INT? . . . . .	8-14
INT! . . . . .	8-14
INT (Interrupt) Control . . . . .	3-7
Integer Variables. . . . .	7-25
Interactive Displays . . . . .	8-59
Interface . . . . .	3-3
INU. . . . .	7-108
IO Control . . . . .	3-6
IO Mode. . . . .	7-8, 8-37, 8-54, 8-81
IO or XIO Set-Up. . . . .	8-54
IO Parameters in COMBASIC . . . . .	7-40
IXO. . . . .	7-108
IXU. . . . .	7-108
JMP . . . . .	7-108

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
JUMP. . . . .	6-10
Keyboard . . . . .	3-3
Keyboard, External RS-232, Speaker, and Transmitter. . . . .	3-7
KILL . . . . .	7-11
KILL <i>program name</i> . . . . .	7-11
Language, IO . . . . .	7-10, 8-39
LAPB. . . . .	8-71
LCN+1 . . . . .	8-70
LCN-1 . . . . .	8-70
LET . . . . .	7-77
LET A-Z . . . . .	7-77
LET A\$-Z\$ . . . . .	7-78
LET A\$-Z\$=CBUFB. . . . .	7-78
LET A\$-Z\$=CBUFE. . . . .	7-79
LET A\$-Z\$=RTC. . . . .	7-79
LET a-z. . . . .	7-80
LET a-z=a-z LPAR. . . . .	7-80
LET a-z=DSTAT . . . . .	7-81
LET a-z=a-z HEX . . . . .	7-81
LET a-z=XIOP or LET XIOP=a-z . . . . .	7-82
LET a-z=0B " <i>binary number</i> ". . . . .	7-83
LET a-z=0H " <i>HEX pair</i> ". . . . .	7-83
LET a-z=0O " <i>octal number</i> ". . . . .	7-83
LET BCS . . . . .	7-83
LET CLOCK. . . . .	7-84

# INDEX

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
LET FA-FZ . . . . .	7-79
LET FA-FZ=MSTM . . . . .	7-80
LET LANG . . . . .	7-84
LET MODE . . . . .	7-85
LET NRZI. . . . .	7-85
LET PARITY . . . . .	7-85
LET SPEED . . . . .	7-85
LET STOPS . . . . .	7-85
LET SYNC . . . . .	7-86
Level-1 Directory . . . . .	5-1
Level-1, Physical Interface . . . . .	8-1
Level-2, Frame Level. . . . .	8-1
Level-3 Command Mode. . . . .	8-51, 8-83
Level-3 Commands. . . . .	7-1
Level-3, Packet Level. . . . .	8-8
Line Activity Buffer Display. . . . .	8-55
LINE ERROR PERFORMANCE. . . . .	5-12
LINE UTILIZATION. . . . .	5-11
Link Clear-Down (LAP) . . . . .	8-21
Link Clear-Down (LAPB) . . . . .	8-20
LINK-DISC . . . . .	8-71
Link Down State . . . . .	8-18
Link Set-Up (LAP) . . . . .	8-20
Link Set-Up (LAPB). . . . .	8-19
Link Tester . . . . .	8-79

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
Load New Menu from Diskette . . . . .	6-35
Load New Menu from PACER . . . . .	6-33
LOG-COMMENT . . . . .	8-72
LOG-COUNT-# . . . . .	8-71
Logical Channel Group Number, LG. . . . .	8-9, 8-36
Logical Channel Number, LCN . . . . .	8-9, 8-36
MARKER . . . . .	8-69
Master Directory. . . . .	5-33
Master/Slave Operation. . . . .	5-30
Maximum Octets. . . . .	8-75
M-Bit. . . . .	8-37
Measure CPU Response . . . . .	5-26
Measurement Library . . . . .	5-22
Measure RTS/CTS Delay. . . . .	5-30
Memories . . . . .	3-3
Memory. . . . .	3-8
Memory Allocation. . . . .	7-17
MENU . . . . .	7-12
MICRO . . . . .	7-86
Minimum Octets . . . . .	8-75
Miscellaneous Instructions. . . . .	6-18
Mode, IO . . . . .	7-8, 8-37, 8-54, 8-81
Mode, Transmission. . . . .	8-39
MODULO-8 . . . . .	8-73

**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
MODULO-128 . . . . .	8-73
Monitor Parameters . . . . .	8-30
Monitor RTS/CTS Delay. . . . .	5-29
Monitor/Trap . . . . .	5-4, 5-9, 5-15
MOV . . . . .	7-108
MSGS IN . . . . .	5-11
MSGS OUT . . . . .	5-11
MSK . . . . .	7-108
Network Analysis. . . . .	5-10
NEXT. . . . .	8-66
NOT ALL LCN'S . . . . .	8-73
NPE . . . . .	7-109
N(R) . . . . .	8-36
NRZI, IO . . . . .	7-8, 8-39
N(S) . . . . .	8-36
Number Facility Fields . . . . .	8-76
Off-Line Data Display . . . . .	5-6, 5-9, 5-15
Off-Line Data Display Library . . . . .	5-15
OFF-LINE PROGRAM LOAD (OPTION) . . . . .	6-23
ON. . . . .	7-86
Operation. . . . .	6-2
Operators. . . . .	7-21
Options. . . . .	3-10
OR. . . . .	7-22

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
OTC . . . . .	8-37
Other Variables . . . . .	7-37
Overall Line Performance . . . . .	5-11
Packet Classifications . . . . .	8-75
Packet Level Active . . . . .	8-53
Packet Level Header . . . . .	8-36
Packet Specifications . . . . .	8-73
Packet Type Identifier . . . . .	8-9
Packet Type Name . . . . .	8-68
Parity, IO . . . . .	7-9, 8-39
PAUSE . . . . .	8-68
P/F . . . . .	8-36
POLL CYCLE TIME . . . . .	5-11
Poll/Select Remote . . . . .	5-10
Power Requirements . . . . .	2-4
Power-Up . . . . .	4-2
P(R) . . . . .	8-37
[PR]+1 . . . . .	8-68
[PR]-1 . . . . .	8-68
[PR]-CLEAR . . . . .	8-69
PRE . . . . .	7-109
Primary Display . . . . .	3-2
PRINT . . . . .	7-87
PRINT A-Z . . . . .	7-88
PRINT A\$-Z\$ . . . . .	7-88



# INDEX

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
PRINT a-z . . . . .	7-88
Printer IO Mode . . . . .	8-41, 8-83
PRINT FA-FZ . . . . .	7-88
PRINT "[ <i>HEX number</i> ]" . . . . .	7-89
Print Level-2 Menu. . . . .	6-31
Print Scenario Buffer . . . . .	8-54
PRINT " <i>string constant</i> ". . . . .	7-89
PRINT "( <i>string constant</i> )" . . . . .	7-89
PRINT % . . . . .	7-89
PRINT #0. . . . .	7-89
PRINT #1. . . . .	7-89
PRINT #2. . . . .	7-89
PRINT #3. . . . .	7-90
PRINT #4. . . . .	7-90
PRINT #5. . . . .	7-90
PROGRAM IDENTIFIER. . . . .	6-23
Programming Hints. . . . .	6-6
Programming Rules and Aids. . . . .	7-20, 7-21
Program Mode Commands. . . . .	6-5
<i>Program name</i> . . . . .	7-14
<i>Program name!</i> . . . . .	7-14
P(S) . . . . .	8-37
[PS]+1 . . . . .	8-68
[PS]-1 . . . . .	8-68
[PS]-CLEAR. . . . .	8-69

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
Q-Bit . . . . .	8-13, 8-36
QPE . . . . .	7-109
READ . . . . .	7-65, 7-90
Real Time Clock . . . . .	7-37
Real Time Display . . . . .	3-3
Real Time Monitor . . . . .	8-32
Real Time Monitor and Display. . . . .	8-29
RECEIVE AND LOG ANY CHARACTER EXCEPT ARGUMENT. . . . .	6-14
RECEIVE AND LOG "n" CHARACTERS. . . . .	6-11
RECEIVE AND LOG THROUGH ANY ONE OF UP TO FIVE CHARACTERS . . . .	6-11
RECEIVE AND LOG THROUGH SEQUENCE. . . . .	6-11
RECEIVE-ANY. . . . .	8-71
Receive Instructions . . . . .	6-11
RECEIVE NEXT SEQUENCE AS BCS AND SKIP NEXT INSTRUCTION IF VALID . .	6-13
RECEIVE-ONLY . . . . .	8-66
Receiving Acknowledgement. . . . .	8-21
Receiving Reject. . . . .	8-21
REJ . . . . .	8-6, 8-16
REM . . . . .	7-90
Reserved Variables . . . . .	7-36
Reset (LAP). . . . .	8-22
Reset (LAPB) . . . . .	8-21
RESET LOGGING TO BEGINNING OF BUFFER. . . . .	6-10
Response . . . . .	8-75
RESP TIME . . . . .	5-11

# INDEX

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
Restart . . . . .	8-17
RESTORE. . . . .	7-65, 7-90
RETURN . . . . .	7-73, 7-90
RETURN . . . . .	8-71
RETURN FROM SUBROUTINE. . . . .	6-10
RETURN TO MODE SELECT. . . . .	6-10
RETURN TO MODE SELECT ON BUFFER FULL . . . . .	6-10
REXMIT IN . . . . .	5-11
REXMIT OUT . . . . .	5-11
RNR . . . . .	8-6, 8-15
RR. . . . .	8-6, 8-15
RSN . . . . .	7-109
RSR?. . . . .	8-16
RSR!. . . . .	8-17
RST?. . . . .	8-17
RST!. . . . .	8-18
RTL . . . . .	7-23
RTR . . . . .	7-23
RUN . . . . .	7-13
RUN!. . . . .	7-13
SABM. . . . .	8-7
Safety Precautions . . . . .	2-4
SARM/DM. . . . .	8-7
SAVE. . . . .	7-15
SAVE <i>program name</i> . . . . .	7-15

## ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
SAVE <i>program name!</i> . . . . .	7-15
Scenario Driven Monitor. . . . .	8-54
SCREEN . . . . .	7-90
SCREEN B . . . . .	7-90
SCREEN C . . . . .	7-91
Screen Control. . . . .	7-51
SCREEN E . . . . .	7-91
SCREEN G a-z. . . . .	7-91
SCREEN H . . . . .	7-91
SCREEN I. . . . .	7-91
SCREEN L . . . . .	7-92
SCREEN N . . . . .	7-92
SCREEN P = ____ . . . . .	7-92
SCREEN R . . . . .	7-92
SCREEN S . . . . .	7-92
SCREEN SA-SY . . . . .	7-93
Screen Size . . . . .	8-50
SCREEN " <i>string constant</i> ". . . . .	7-93
SCREEN SZ . . . . .	7-93
SCREEN U . . . . .	7-93
SCREEN Z . . . . .	7-93
SDLC FRAME CHAIN (OPTION) . . . . .	6-16
SDLC FRAME TRANSMIT CHARACTERS, RECEIVE AND LOG SIMUL- TANEOUSLY (OPTION). . . . .	6-18
SDLC FRAME TRANSMIT (OPTION) . . . . .	6-17

**INDEX**

**ALPHABETICAL INDEX (Cont'd)**


<u>SUBJECT</u>	<u>Page</u>
SECONDS > COUNTER . . . . .	8-70
SELECT BCS TYPE. . . . .	6-9
SELECT TEST MODE. . . . .	6-8
SET. . . . .	7-109
SET COUNTER A. . . . .	6-21
SET COUNTER B. . . . .	6-21
SET COUNTER "C" (OPTION) . . . . .	6-20
SET COUNTER "D" (OPTION) . . . . .	6-20
Set Date and Time . . . . .	8-81
SET-D-BIT . . . . .	8-67
Set IO Parameters . . . . . Mode "I" . . . . .	6-24
SET-LCGN . . . . .	8-67
SET-LCN . . . . .	8-68
Set Line IO . . . . .	8-50
SET-M-BIT . . . . .	8-68
Set Monitor Parameters. . . . .	8-49
SET-N2. . . . .	8-68
Set Printer IO (XIO) . . . . .	8-50
SET-Q-BIT . . . . .	8-67
SET-[STAT] . . . . .	8-73
SET-TIMER . . . . .	8-66
SET TIMER AND SKIP NEXT INSTRUCTION, ADVANCE TO NEXT INSTRUCTION WHEN TIMER RUNS OUT. . . . .	6-22
SET-T1 . . . . .	8-68
SET-WINDOW . . . . .	8-69

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
Simulate DCE . . . . .	8-54
Simulate DCE (Link Monitor). . . . .	8-54
Simulate DTE . . . . .	8-54
Simulate DTE (Link Monitor). . . . .	8-54
SKIP-IF-LCN-MATCH . . . . .	8-72
SKIP-IF-LINK-DOWN. . . . .	8-72
SKIP NEXT INSTRUCTION IF COUNTER EQUALS "00" . . . . .	6-21
SKIP NEXT INSTRUCTION IF RS-232 LEAD ___ IS OFF. . . . .	6-20
SKIP NEXT INSTRUCTION IF RS-232 LEAD ___ IS ON . . . . .	6-20
SKIP NEXT INSTRUCTION ON PARITY ERROR . . . . .	6-18
SKIP NEXT INSTRUCTION UNLESS ARGUMENT EQUALS LAST LOGGED CHARACTER(S). . . . .	6-14
SKIP-UNLESS-LAST = . . . . .	8-72
SKIP-UNLESS-[STAT]= . . . . .	8-73
Special Function Keys. . . . .	3-17
Speed, IO . . . . .	7-8, 8-37
Split Screen Format . . . . .	8-35
Standard Configuration . . . . .	3-8
START BCS . . . . .	6-10
Start Flag. . . . .	8-3
Start Monitor . . . . .	8-50
STATE . . . . .	7-94
STATE BRG. . . . .	7-94
STATE CRC = ___ . . . . .	7-94
STATE DISCON/DISCOFF. . . . .	7-95

**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
STATE FAST/NFAST . . . . .	7-95
STATE FCS = ____ . . . . .	7-95
STATE FDUX . . . . .	7-96
STATE FLTR/NFLTR. . . . .	7-96
STATE HDUX . . . . .	7-96
STATE LEAD=/NLEAD . . . . .	7-96
STATE LOCK/NLOCK . . . . .	7-96
STATE MODEM . . . . .	7-97
STATE NLEAD8 . . . . .	7-96
STATE NSTAT/FAST . . . . .	7-66
STATE NSTAT/NFAST . . . . .	7-67
STATE P3ONLY/NP3ONLY . . . . .	7-98
STATE PIPE. . . . .	7-98
STATE STAT/FAST. . . . .	7-67
STATE STAT/NSTAT . . . . .	7-98
STATE SYNC/NSYNC. . . . .	7-99
STATE TERM . . . . .	7-99
STATE UPC/NUPC. . . . .	7-99
STATE XIDL. . . . .	7-100
Station Performance . . . . .	5-12
Status Byte . . . . .	7-43
Status/Memory Allocation/  . . . . .	7-17
STOP. . . . .	7-100
Stop-on-Error . . . . .	8-31
Stops, IO . . . . .	7-9, 8-39

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
Storage . . . . .	2-4
STORE IO PARAMETERS . . . . .	6-9
String Variables . . . . .	7-29
SUBROUTINE CALL . . . . .	6-10
SUBTRACT ONE COUNT FROM COUNTER . . . . .	6-21
Supervisory Frames. . . . .	8-5
Synchronous Library . . . . .	5-7
Sync, IO. . . . .	7-8, 8-39
TAS . . . . .	7-109
TAS A . . . . .	7-110
T-Connector. . . . .	3-19
TDS . . . . .	7-110
TDS A . . . . .	7-110
Three Levels of Operation. . . . .	3-2
Time . . . . .	7-16
Timer Instructions . . . . .	6-22
Timers/Triggers . . . . .	7-38
Timing Chart . . . . .	7-50
TONP2 . . . . .	7-100
TONP2B . . . . .	7-101
TONP2D . . . . .	7-101
TONP3 . . . . .	7-101
TONP3B . . . . .	7-101
TONP3D . . . . .	7-101
TONP4 . . . . .	7-101



**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
TONP5 . . . . .	7-101
TONP6 . . . . .	7-101
TONP8 . . . . .	7-101
TONP11. . . . .	7-101
TONP18. . . . .	7-102
TONP20. . . . .	7-102
TONP22. . . . .	7-102
TONP24. . . . .	7-102
TON/TOFF . . . . .	7-100
Training, ENCORE and PACER-103. . . . .	9-7, 9-8
TRANSFER-TO-SCENARIO-# . . . . .	8-71
Transmission Modes. . . . .	8-39
Transmission Procedures. . . . .	8-18
TRANSMIT . . . . .	8-67
TRANSMIT BCS CHARACTER(S). . . . .	6-14
TRANSMIT CHARACTER(S). . . . .	6-14
TRANSMIT CHARACTERS, RECEIVE AND LOG SIMULTANEOUSLY . . . . .	6-14
Transparency . . . . .	8-3
Transparent Monitor . . . . .	5-9
TRAP SEQUENCE . . . . .	6-11
Troubleshooting Tips . . . . .	9-6
TTY Conversational as DCE . . . . .	5-6
TTY Conversational as DTE . . . . .	5-6
TURN OFF ALL CONTROLLED RS-232 LEADS (4, 5, 8, T, M, AND 21) EXCEPT DTE (20) AND DSR (6) . . . . .	6-19

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
TURN OFF AUDIO . . . . .	6-19
TURN OFF DATA DISPLAY . . . . .	6-19
TURN-OFF-LAW. . . . .	8-71
TURN OFF RECEIVER 2 OR 3. . . . .	6-11
TURN OFF RS-232 LEAD ____ . . . . .	6-20
TURN ON AUDIO . . . . .	6-18
TURN ON BOTH RECEIVERS . . . . .	6-11
TURN ON DATA DISPLAY . . . . .	6-19
TURN-ON-LAW . . . . .	8-71
TURN ON RS-232 LEAD ____ . . . . .	6-19
Typical Date and Time Operating Procedure . . . . .	8-82
Typical Display Mode Operation . . . . .	6-29
Typical ENCORE to PACER Menu Dump . . . . .	6-36
Typical Execute Procedure . . . . .	6-3
Typical Fast Capture and Display Operating Procedure . . . . .	8-51
Typical IO Entry Procedure . . . . .	7-10
Typical IO Operating Procedure . . . . .	8-39
Typical Level-1 Operating Procedure for Monitor and Storage. . . . .	4-5
Typical Level-2 Operating Procedure for Synchronous Monitor . . . . .	4-7
Typical Level-3 Operating Procedure for Synchronous Monitor . . . . .	4-9
Typical Menu Dump to Diskette . . . . .	6-38
Typical Menu Load from Diskette. . . . .	6-35
Typical Menu Print Operation . . . . .	6-32
Typical Operation in Level-1. . . . .	4-5
Typical Operation in Level-2. . . . .	4-7

**INDEX**

**ALPHABETICAL INDEX (Cont'd)**

<u>SUBJECT</u>	<u>Page</u>
Typical Operating Procedure, Level-3. . . . .	4-9
Typical PACER to ENCORE Menu Load. . . . .	6-34
Typical Packet Specification Operating Procedure . . . . .	8-76
Typical Program Entry Procedure. . . . .	6-6
Typical Real Time Monitor and Display Operating Procedure. . . . .	8-43
Typical Scenario Entry Procedure. . . . .	8-57
UA. . . . .	8-7
Unnumbered Frames . . . . .	8-6
Unpacking and Handling. . . . .	2-1
UPD . . . . .	7-110
User Data. . . . .	8-10
User Defined Library . . . . .	5-32
Variable Length Addresses. . . . .	8-72
Variables . . . . .	7-25
WAIT. . . . .	7-102
WAIT A-Z. . . . .	7-102
WAIT XDONE . . . . .	7-102
WHEN . . . . .	7-103
WHEN ERROR. . . . .	7-103
WHEN FULL . . . . .	7-103
WHEN KBD a-z . . . . .	7-103
WHEN P ___=0/1. . . . .	7-103
WHEN P2/P3 . . . . .	7-104
WHEN P2/P3 = A\$-Z\$. . . . .	7-104
WHEN P2/P3 a-z. . . . .	7-104

ALPHABETICAL INDEX (Cont'd)

<u>SUBJECT</u>	<u>Page</u>
WHEN P2/P3 = BCB . . . . .	7-105
WHEN P2/P3 = BCC . . . . .	7-105
WHEN P2/P3 = BCG . . . . .	7-105
WHEN P2/P3 = FLAG. . . . .	7-105
WHEN P2/P3 = "[HEX pair]" . . . . .	7-106
WHEN P2/P3 = PE . . . . .	7-106
WHEN P2/P3 = "string constant" . . . . .	7-106
WHEN RTC . . . . .	7-106
WHEN TIMER . . . . .	7-107
WHEN XIOIN . . . . .	7-107
WORST CASE RESPONSE TIME . . . . .	5-12
XBC . . . . .	7-110
XIO . . . . .	6-32, 7-18
XMT . . . . .	7-110
XOR . . . . .	7-24
X25/X75 Background . . . . .	8-1
X25/X75 Monitoring Speeds . . . . .	8-38
X25/X75 Menu. . . . .	8-27, 8-37, 8-50, 8-54
X25/X75 Operating System . . . . .	8-1
# D EDIT <i>program name!</i> . . . . .	7-4
# D <i>program name!</i> . . . . .	7-14
# EDIT <i>program name!</i> . . . . .	7-4
# <i>program name!</i> . . . . .	7-14



**Publication No. 810-00184A**

Your comments regarding this publication will help us to improve it for you. Please comment in the space provided giving specific page and paragraph numbers whenever possible. If you have any questions about DIGITECH products or services, please contact our Customer Service Department at (203) 438-3731.

---

Was your ENCORE received in perfect electrical order, complete with ordered options, manuals, diskettes? Yes  No  Comments: \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

---

What level of operation interests you the most? Resident Program Library  DICOL  COMBASIC  X25/X75  SNA  Other \_\_\_\_\_.

Have you had any previous programming experience? Yes  No . If yes, in what areas? \_\_\_\_\_

\_\_\_\_\_

---

Did you find any errors in this publication? (Please include page reference of error).

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

---

Do you have any recommendations to improve this publication? \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

---

**Optional:**

NAME: \_\_\_\_\_ TITLE: \_\_\_\_\_

COMPANY: \_\_\_\_\_

ADDRESS: \_\_\_\_\_

TELEPHONE NUMBER: \_\_\_\_\_ DATE: \_\_\_\_\_

SERIAL NUMBER OF ENCORE 200: \_\_\_\_\_

---

FOLD

FOLD

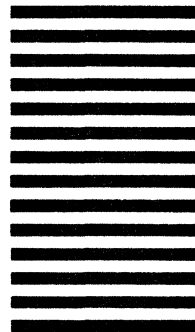


NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY CARD**  
FIRST CLASS      PERMIT NO. 7      RIDGEFIELD, CT

POSTAGE WILL BE PAID BY

**DIGITECH INDUSTRIES, INC.**  
POST OFFICE BOX 547  
RIDGEFIELD, CT 06877



FOLD

FOLD

# Warranty

DIGITECH Data Industries, Inc., herein referred to as DIGITECH (SELLER) warrants to the Buyer that all material of its manufacture purchased hereunder to be free from defects in materials and workmanship when used for the purpose for which it was designed.

In discharge of this warranty, DIGITECH (SELLER) agrees to replace or repair any material of its manufacture which when under proper and normal use shall, within 12 months after delivery to the Buyer, prove to be defective due to faulty material or poor workmanship provided, however:

1. That any defect is not the result of damage incurred in shipment of materials to Buyer.
2. That the equipment has not been altered in any way either as to design or use other than those approved by DIGITECH.
3. That any equipment or accessories furnished hereunder that are not of DIGITECH manufacture or design shall be limited to such adjustments as the manufacturer thereof makes with it.
4. That any claim of defect under this warranty is made immediately upon discovery thereof and that inspection by DIGITECH, if required, indicates the validity of such claim to DIGITECH's satisfaction.
5. That a return authorization is issued by DIGITECH to Buyer and the defective equipment is returned to the DIGITECH factory with transportation charges prepaid.

DIGITECH's obligation under this warranty is limited to the repair or replacement of defective equipment with the exceptions as noted above.

No warranty, expressed or implied, other than that specifically set forth herein shall be applicable to any equipment manufactured by DIGITECH and the foregoing warranty shall constitute the Buyer's sole right and remedy. In no event does DIGITECH assume any liability for general or consequential damages, loss or expense directly arising from the installation or use of DIGITECH products or any inability to use them either separately or in combination with other equipments or materials.

This warranty is invalid if any unauthorized changes have been made to the basic configuration of the equipment or if the internal workings have been tampered with in any way.

**DIGITECH**  
DATA INDUSTRIES, INC..

66 Grove St., Ridgefield, CT 06877 U.S.A.  
■ (203) 438-3731 ■ Telex No. 969623



