HONEYWELL

MLC-16 COMMUNICATIONS HANDBOOK

HARDWARE

Honeywell

Dear Reader:

Certain Engineering terms that may appear in this publication have been replaced by new Marketing terminology. These changes are as follows:

Old Term	New Term
NMLCP	MLC-16
Flexible Line Adapter Package (FLAP)	Line Interface Unit (LIU)
System Console Facility (SCF), Virtual Control Panel (VCP)	Remote Support Facility (RSF)

Technical Publications
Billerica, MA

DPS 6 & LEVEL 6
NMLCP COMMUNICATIONS HANDBOOK

SUBJECT

New Multiline Communications Processor, Related Line Interfaces, and Reference Information for Programmer Developed Communications Applications

SPECIAL INSTRUCTIONS

For your convenience this manual contains a CCP programming instruction card located at the back of the manual.

The following notice is provided in accordance with the United States Federal Communications Commission's (FCC) regulations.

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation it has not been tested for compliance with limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

ORDER NUMBER	DOCUMENT NUMBER	
GA02-00	71017570-00	May 1983

Honeywell

PREFACE

This manual is intended for computer programmers who are already experienced in creating communications applications. The purpose of this manual is to assist these experienced communications programmers in creating applications for a Honeywell DPS 6/Level 6 hardware system, utilizing a New Multiline Communication Processor (NMLCP).

The manual discusses the NMLCP and its associated line interfaces. The reader's knowledge of data communications equipment, communications line conventions, and data terminal equipment is expected to be based on additional sources such as those listed at the end of this Preface.

Section 1 of this document is a general introduction to the NMLCP and compatibility with the Multiline Communications Processor (MLCP).

Section 2 provides a programming overview of the NMLCP.

Section 3 contains the channel control program instruction set.

Sections 4, 5, and 6 detail the communications control blocks, line control tables and the processor interfaces used in creating applications for the NMLCP.

Appendix A contains the programming guidelines for selected NMLCP features.

Appendixes B, C, D, and F describe the communications adapters supported by the NMLCP.

USER COMMENTS FORMS are included at the back of this manual. These forms are to be used to record any corrections, changes, or additions that will make this manual more useful.

This document is issued under authority of Honeywell Document Issue Notice No. BLCDD7341 (April 1983).

File No.: 1RO3, 1SO3 GA02-00

Appendix E presents the Flexible Line Adapter Package.

Appendix G contains a glossary of acronyms and their definitions.

Reference literature for data communications equipment includes the following Honeywell publications.

Order Number	Manual Title
CZ02	GCOS 6 MOD 400 System Building & Administration
CZ03	GCOS 6 MOD 400 System Concepts
CZ05	GCOS 6 MOD 400 System Programmer's Guide - Volume I
CZ06	GCOS 6 MOD 400 System Programmer's Guide - Volume II
CZ07	GCOS 6 MOD 400 Programmer's Pocket Guide
CZ16	GCOS 6 MOD 400 System Messages
CZ17	GCOS 6 MOD 400 Commands
CZ38	GCOS 6 Assembly Language (MAP) Reference

The following non-Honeywell publications should be referenced as appropriate.

- EIA (Electronic Industries Association) Standard RS-232C
- CCITT (International Consultive Committee for Telephony and Telegraphy) White Book, Volume VIII
- Data Set 103A Interface Specification - February 1967 (Bell System Technical Reference, PUB41101)
- Data Set 103A3/103E/103G/103H Interface Specification - October 1973 (Bell System Technical Reference, PUB41102)
- Data Set 103F Interface Specification - May 1964 (Bell System Technical Reference, PUB41103)
- Data Set 113A Interface Specification - August 1973 (Bell System Technical Reference, PUB41104)
- 113-Type Data Station Interface Specification - October 1971 (Bell System Technical Reference, PUB41105)
- Data Sets 201A & B - August 1969 (Bell System Technical Reference, PUB41201)
- Data Set 201C Interface Specification - April 1973 (Bell System Technical Reference, PUB41210)

- Data Sets 202C & D Interface Specification - May 1964 (Bell System Technical Reference, PUB41202)
- Data Sets 202S & T Interface Specification - August 1974 (Bell System Technical Reference, PUB41212)
- Data Set 203-Type - Revised April 1974 (Bell System Technical Reference, PUB41204)
- Data Set 208A Interface Specification - November 1973 (Bell System Technical Reference, PUB41209)
- Data Set 208B Interface Specification - August 1973 (Bell System Technical Reference, PUB41211)
- Data Set 301B Interface Specification - March 1967 (Bell System Technical Reference, PUB41301)
- Wideband Data Station 303 Type - December 1974 (Bell System Technical Reference, PUB41302)
- Digital Data System Data Service Unit Interface Specification - March 1973 (Bell System Technical Reference, PUB41450)
- Interface Between Data Terminal Equipment and Automatic Calling Equipment for Data Communication - August 1969 (Electronic Industries Association, RS-366)

CONTENTS

	Page
SECTION 1 INTRODUCTION	1-1
NMLCP Operation	1-2
NMLCP Hardware Overview	1-3
Communications Adapters (CAs)	1-4
Flexible Line Adapter Packages (FLAPs)	1-4
Random Access Memory (RAM) Layout	1-4
Compatibility with MLCP	1-9
New Functionality of the NMLCP	1-12
Communications Control Blocks	1-13
Interrupt Handling	1-13
Executive Channel	1-14
RAM Data Transfer	1-14
Block Mode Read/Write	1-15
Data Set Scan Firmware	1-16
Processor Load Monitor	1-16
Vectored Jump Mechanism	1-16
Channel Service Multiplexing	1-17
WAIT Functionality	1-17
Pause Functionality	1-17
Priority Scheme	1-18
Data Service and Event Service.Levels	1-19
Data Transfer	1-19
Receive	1-20
Transmit	1-20
Test Mode/Direct-Connect Clock	1-21
CCP Instructions	1-22
SECTION 2 PROGRAMMING OVERVIEW	2-1
Main Memory Program	2-2
Communications Control Blocks	2-2
Channel Control Program	2-3
Line Control Tables	2-4
Setting Up the Processor; Receiving and Transmitting Data	2-5
Initializing the Processor writing the LCT and CCP area	2-9

CONTENTS

	Page
Writing the CCBs	2-9
Loading Adapter Line Registers	2-10
Receiving Data	2-10
Transmitting Data	2-11
End of CDB Processing	2-11
End of Logical Message Processing	2-12
End of Channel Line Usage	2-12
Main Memory Program Input/Output Instructions	2-13
Control of Communications Data Blocks	2-18
Control of Communications Control Blocks	2-18
Control of Channel Control Programs	2-19
Detection of Errors and Status Changes Related to Data Communications Equipment and Data Terminal Equipment	2-19
Detailed Description of Main Memory Program Input/Output Instructions	2-20
IO (Input CCB Control) Instruction	2-21
IO (Input CCB Range) Instruction	2-21
IO (Input CCB Status) Instruction	2-21
IO (Input Configuration A) Instruction	2-22
IO (Input Line Status) Instruction	2-23
IO (Input Device Identification.Number) Instruction	2-23
IO (Input Extended Identification Number) Instruction	2-23
IO (Input Firmware Revision) Instruction	2-25
IO (Input Interrupt Control A) Instruction	2-25
IO (Input Interrupt Status B) Instruction	2-25
IO (Input LCT Byte) Instruction	2-26
IO (Input Load Monitor) Instruction	2-26
IO (Input Next CCB Status) Instruction	2-26
IO (Read and Clear LCT Byte) Instruction	2-27
IO (Input from Analyzer) Instruction	2-27
IO (Output Channel Control) Instruction	2-27
IO (Output Configuration A) Instruction	2-30
IO (Output CCB Control) Instruction	2-31
IO (Output Interrupt Control B) Instruction	2-32
IO (Output Interrupt Control A) Instruction	2-33
IO (Output NMLCP Control) Instruction	2-33
IO (Output RAM Control) Instruction	2-34
IO (OR to LCT Byte) Instruction	2-34
IO (Output LCT Byte) Instruction	2-35
IOLD (Load Data Area) Instruction	2-35
IOLD (RAM Data Transfer) Instruction	2-36
IO (Output to Analyzer) Instruction	2-36
IO (Output to Analyzer) Instruction	2-37
WAIT and NAK Responses	2-37

CONTENTS

	Page
Channel Numbers	2-37
Device Identification Numbers	2-38
Interrupts to the Central System	2-38
Timers	2-39
Restart/Downline Load/Downline Dump	2-39
SECTION 3 CHANNEL CONTROL PROGRAM INSTRUCTION SET	3-1
Initial CCP Setup	3-1
Starting a CCP	3-2
CCP Execution	3-3
NMLCP CCP Registers and Program Indicators	3-4
CCP Control and Executable Instructions	3-7
Program Development Tools	3-7
Programming Rules	3-7
Macropreprocessor Assembly Operation	3-7
Internal Formats	3-8
CCP Generation Control Statements	3-8
LOC Statement	3-10
ORG Statement	3-10
MORG Statement	3-11
DATA Statement	3-11
CCP Instruction Execution Timing	3-12
CCP Executable Instructions	3-19
Op Code Map	3-163
SECTION 4 COMMUNICATIONS CONTROL BLOCKS	4-1
Communications Control Blocks	4-1
CCBs and CCB Processing	4-3
CCB Setup	4-3
Characteristics of a CDB	4-4
CCB Execution	4-4
CCB List	4-6
CCB List Setup	4-6
CCB List Return Status	4-6
CCB List Initial Conditions	4-7
CCB List Execution	4-7
CCB List in RAM	4-7
CCB Field Format	4-8
The Address Field	4-9
CCB Range Field	4-9
CCB Control Field	4-10
CCB Status Field	4-11
Setting Up a CCB	4-14

CONTENTS

	Page
CCB Descriptions of a CDB	4-14
Processing an Current CCB	4-15
Mode Considerations for CCB Processing	4-16
CCB Processing in Extended Mode	4-16
CCB Processing in Compatable Mode	4-17
Instructions for CCB Processing in Both Modes	4-17
Completion of a CCB	4-18
SECTION 5 LINE CONTROL TABLES	5-1
LCT Description	5-1
Loading an LCT after an Initialization	5-4
LCT Locations To Be Set Up by CPU	5-4
LCT Locations for CCP Work Area	5-7
LCT Locations Maintained by NMLCP	5-7
LCT Locations Reserved for Firmware	5-10
Extended LCTs	5-10
LCT Stacks and Queues	5-10
Layout of LCT Bytes	5-10
SECTION 6 PROCESSOR INTERFACES	6-1
Processor Channel Number Addressing from Main Memory Program	6-3
Processor Interrupts to Main Memory Program	6-3
Processor Control of Data Sets and Line Adapters	6-5
Processor Monitoring of Data Set and Adapter Status	6-5
Processor Parity Checking and Generation	6-6
Processor Cyclic Redundancy Checking	6-7
Data Transfer Rates for Processor Communications Lines	6-9
APPENDIX A PROGRAMMING GUIDELINES FOR SELECTED NMLCP FEATURES	A-1
Initialization	A-1
Adapter Setup	A-2
Access to Line and Flap Registers	A-3
CCP Instructions	A-4
SEND Instruction	A-4
RECV Instruction	A-5
OUT LR1 Instruction	A-5
IN LR1 Instruction	A-5
SFS Instruction	A-5
BLBT, BLBF Instructions	A-5
LD and ST Instructions	A-6

CONTENTS

Page

WAIT Instruction..... A-6
BLCT, BLCF Instructions..... A-6
INTR Instruction..... A-6
Table Look-up Instruction Programming Example..... A-7
Valid CCBs..... A-12
Data Set Scan..... A-13
CPU Interrupts..... A-14
CCBs as Cause of CPU Interrupts..... A-15
Data Set Scan as Source of CPU Interrupts..... A-16
Combination of CCP Interrupts and INTR in Debug..... A-16
Deferred Interrupt Queue..... A-16
Addressing Limits..... A-17
CCB Area Only Implicitly Accessibly..... A-17
Inability of One Line to Access Another..... A-17
Need for Pad Characters..... A-17
Two-Way Alternate Operation..... A-18
Error Handling..... A-19
Conditions Under Which Processor will Issue a NAK..... A-19
LCT Status Bytes..... A-20
CCB Status Bytes..... A-21

APPENDIX B NEW MULTILINE CONTROLLER SYNCHRONOUS/ ASYNCHRONOUS ADAPTER..... B-1

Introduction..... B-1
Software Overview..... B-1
Data Formats..... B-3
Synchronous Format..... B-3
Asynchronous Format..... B-4
Isochronous Format..... B-5
Hardware Overview..... B-5
ACLA/SCLA Functionality..... B-5
New Functionality..... B-7
Line Registers..... B-7
Line Registers Bit Assignment (ACLA/SCLA Mode)..... B-8
Line Register 0 (LR0)..... B-8
Line Register 1 (LR1)..... B-8
Line Register 2 (LR2)..... B-10
Line Register 3 (LR3)..... B-12
Line Register 4 (LR4)..... B-12
Line Register 5 (LR5)..... B-12
Line Register 6 (LR6)..... B-13
Line Register 7 (LR7)..... B-14
Line Registers Bit Assignment (Non-ACLA/SCLA Mode)..... B-14

CONTENTS

	Page
Transmitter Operation	B-14
Receiver Operation	B-16
Composite Line Register Bit Assignments	B-16
Additional Functionality	B-16
FLAP Registers	B-18
Channel Request Interrupt (CRI)	B-18
Configuration	B-18
Channel Number Assignment	B-19
Device Identification Number	B-20
Data Clock	B-20
Receive Synchronization	B-21
Asynchronous Channel	B-21
Synchronous Channel	B-21
Receive Overrun	B-21
Transmit Underrun/Transmit Fill	B-21
Asynchronous Channel	B-21
Synchronous Channel	B-22
Initialization	B-22
Auto Call Attachment	B-22
Mode Control	B-22
APPENDIX C AUTO CALL UNIT FLAP	C-1
Configuration Information	C-1
ACUF Configuration	C-1
Device Identification Number	C-2
Automatic Calling Device Configuration Options	C-2
FLAP Registers	C-3
FLAP Register 1 - Output Data	C-4
FLAP Register 2 - Output Control	C-5
FLAP Register 5 - Input Status	C-7
FLAP Register 7 - Input Status	C-9
Programming Considerations	C-10
Timing Sequence of Call Placement	C-10
Data Transfers	C-11
Call Termination	C-11
Summary; Examples	C-12
Example 1: General Sequence of CCP and MMP Interface.	C-13
Example 2: Transfer of each Individual Digit	C-13
Status of Automatic Calling Device	C-15
Avoiding Possible Race Condition During Call Abort	C-15
Physical Interface to Data Communication Equipment and Data Terminal Equipment	C-15

CONTENTS

Page

APPENDIX D NEW MULTILINE CONTROLLER BROADBAND HDLC/ SYNCHRONOUS ADAPTER..... D-1

Introduction..... D-1
Software Overview..... D-1
Synchronous Data Format..... D-3
HDLC/SDLC Frame Structure..... D-4
  General Format of HDLC/SDLC Frames..... D-4
  Fields of the Frame..... D-6
    Flag Sequence..... D-6
    Address Field..... D-7
    Control Field..... D-7
    Information Field..... D-7
    Frame Check Sequence..... D-8
  Order of Bit Transmissions..... D-9
  Abbort..... D-9
  Transparency..... D-9
  Interframe Time Fill..... D-9
  Intraframe Time Fill..... D-9
  Receive Idle Link State..... D-9
Basic Functions..... D-10
Registers..... D-10
  Register Format in HDLC Mode..... D-10
    Receive Channel Register Definitions..... D-13
    Transmit Channel Register Definitions..... D-17
  Register Format in Synchronous Mode..... D-21
    Receive Channel Register Definitions..... D-21
    Transmit Channel Register Definitions..... D-30
Register Transfer..... D-30
Data Transfer..... D-30
Data Elements..... D-31
Redundancy and Parity..... D-31
Sync Mode Operation..... D-32
  Receive Synchronization..... D-32
  Receive Overrun..... D-32
  Transmit Underrun/Transmit Fill..... D-32
HDLC Mode Operation..... D-33
  Receive Startup and Frame Synchronization..... D-33
  Receive Data Transfers and Receive Overrun..... D-33
  Receive End of Frame..... D-34
  Receive Idle Link State..... D-34
  Receive Missing Frame State..... D-34
  Minimum Receive CCP Structure..... D-35
  Transmit Initialization..... D-36
  Transmit Startup..... D-36
  Transmit Data Transfers and Transmit Underrun..... D-36

Page

Transmit End of Frame..... D-36
Minimum Transmit CCP Structure..... D-37
Status..... D-38
Data Clock..... D-38
Channel Request Interrupt Priority..... D-38
Channel Number Assignment..... D-39
Direct Connect Clock..... D-39
Mode Control..... D-39
Device Identification Number..... D-40

APPENDIX E FLEXIBLE LINE ADAPTER PACKAGE..... E-1

Introduction..... E-1
FLAP-Common Equipment..... E-3
FLAP-Unique Functionality..... E-5
DSA-47 (RS-232C/V.24) FLAP with NRZI Support..... E-5
FLAP Registers..... E-5
DSA-47(RS-232C/V.24) Interface.Pin Assignments..... E-11
DSA-47(RS-232C/V.24) FLAP Driver and Receiver Characteristics..... E-11
DSA-47(RS-232C/V.24) Cable and Connector Characteristics..... E-11
FLAP Line Switching Functionality..... E-11
DSA-47(RS-232C/V.24) FLAP Without NRZI Support..... E-14
DSA-46 Direct-Connect FLAP (RS-422A)..... E-14
FLAP Registers..... E-14
DSA-46 Direct-Connect FLAP Interface Pin Assignments. E-18
DSA-46 Direct-Connect FLAP Driver and Receiver Characteristics..... E-18
DSA-46 Direct-Connect FLAP Cable and Connector Characteristcs..... E-18
DSA-46 Direct-Connect FLAP Interface Terminations, Fail Safing, Filters, and Grounding Options..... E-18
Effect of Synchronous Adapter Direct Connect Bit..... E-19
Current Loop FLAP..... E-19
FLAP Registers..... E-19
Current Loop FLAP Interface Pin Assignments..... E-22
Current Loop FLAP Electrical Characteristics..... E-22
X.21(DSA-49) FLAP..... E-23
FLAP Registers..... E-23
X.21(DSA-49) FLAP Interface Pin Assignments..... E-26

CONTENTS

Page

X.21(DSA-49) FLAP Driver and Receiver Characteristics..... E-26
X.21(DSA-49) FLAP Cable and Connector Characteristics..... E-26
X.21(DSA-49) FLAP Synchronization Procedure..... E-27
Automatic Call Unit FLAP..... E-27
FLAP Registers..... E-27
ACUF Interface Pin Assignments..... E-31
ACUF Driver and Receiver Characteristics..... E-32
ACUF Cable and Connector Characteristics..... E-32
Wideband Data Stations 303-Type FLAP..... E-32
FLAP Registers..... E-33
303 FLAP Interface Pin Assignments..... E-35
303 FLAP Driver and Receiver Electrical Characteristics..... E-36
303 FLAP Cable and Connector Characteristics..... E-36
V.35 FLAP..... E-36
FLAP Registers..... E-36
V.35 FLAP Interface Pin Assignments..... E-38
V.35 FLAP Electrical Characteristics..... E-39
V.35 FLAP Cable and Connector Characteristics..... E-39
MIL-STD-188-114 FLAP..... E-39
FLAP Registers..... E-39
MIL-STD-188-114 FLAP Interface Pin Assignments..... E-39
MIL-STD-188-114 FLAP Driver and Receiver Characteristics..... E-39
MIL-STD-188C Compatibility..... E-41
FLAP Bus Interface..... E-42

APPENDIX F ASYNCHRONOUS DIRECT CONNECT ADAPTER..... F-1

Introduction..... F-1
Software Overview..... F-1
Data Formats..... F-3
Basic Functions..... F-3
ACLA Functionality..... F-3
New Functionality..... F-5
Line Registers..... F-6
Line Register Bit Assignments (ACLA Mode)..... F-6
Line Register 0 (LR0)..... F-6
Line Register 1 (LR1)..... F-6
Line Register 2 (LR2)..... F-7
Line Register 3 (LR3)..... F-8

Page

Line Register 4 (LR4)..... F-9
Line Register 5 (LR5)..... F-9
Line Register 6 (LR6)..... F-10
Line Register 7 (LR7)..... F-10
Line Register Bit Assignments (Non-ACLA Mode)..... F-10
Composit Line Register Bit Assignmments..... F-11
FLAP Register Bit Assignments..... F-12
Miscellaneous Additional Functionality..... F-13
Channel Request Interrupt (CRI)..... F-13
Configuration..... F-14
Channel Number Assignment..... F-14
Device Identification Number..... F-15
Data Clock..... F-15
Receive Synchronization..... F-15
Receive Overrun..... F-15
Transmit Underrun/Transmit Fill..... F-16
Initialization..... F-16
Special Functionality for the SCF..... F-16

APPENDIX G GLOSSARY..... G-1

ILLUSTRATIONS

Figure		Page
1-1	NMLCP Communications Subsystem.....	1-3
1-2	NMLCP-Compatible Memory Map (Block Mode Access)....	1-6
1-3	NMLCP Memory Map (RAM Data Transfer Access).....	1-7
1-4	NMLCP Block Diagram.....	1-8
1-5	NMLCP Priority Scheme.....	1-18
2-1	Setting up the NMLCP.....	2-6
2-2	Receiving Data.....	2-7
2-3	Transmitting Data.....	2-8
4-1	Communications Control Block Flow.....	4-2
4-2	CCB General Format.....	4-3
4-3	Format of CCB Fields in the NMLCP.....	4-8
4-4	Three Types of Data Transfer.....	4-9
4-5	NMLCP CCB Status Bytes 1 and 2.....	4-12
5-1	Line Control Table Layout.....	5-13
A-1	Sample Table Look-Up Program.....	A-8
B-1	NSAA Register Access.....	B-2
B-2	Synchronous Data Format.....	B-3
B-3	General Synchronous Line Format.....	B-4
B-4	Asynchronous Data Format.....	B-4
B-5	NMLCP Communications Subsystem.....	B-6
B-6	NSAA Block Diagram.....	B-7
B-7	NSAA Line Registers - Asynchronous Bit Assignments Compatable Mode.....	B-9
B-8	NSAA Line Registers - Synchronous Bit Assignments Compatable Mode.....	B-9
B-9	LR2-to-FLAP Registers Bit Mapping.....	B-11
B-10	NSAA Line Registers - Asynchronous Bit Assignments New Functionality.....	B-15
B-11	NSAA Line Registers - Synchronous Bit Assignments New Functionallity.....	B-15
B-12	NSAA Line Registers - Composite Asynchronous Bit Assignments.....	B-17
B-13	NSAA Line Registers - Composite Synchronous Bit Assignments.....	B-17

ILLUSTRATIONS

Figure		Page
C-1	ACUF Environment	C-2
C-2	FLAP Register 1 Output Data	C-4
C-3	FLAP Register 2 Output Control	C-5
C-4	FLAP Register 5 Input Status	C-7
C-5	FLAP Register 7 Input Status	C-9
C-6	Typical Automatic Calling Equipment/ACUF/NMLCP Timing Sequence	C-11
D-1	NBHSA Register Access	D-3
D-2	Synchronous Data Format	D-4
D-3	General Synchronous Line Format	D-4
D-4	General Format of HDLC/SDLC Frame	D-5
D-5	NBHSA Registers (Receive Channel - HDLC Mode)	D-11
D-6	NBHSA Registers (Transmit Channel - HDLC Mode)	D-12
D-7	NBHSA Registers (Synchronous Mode)	D-22
E-1	FLAP Module Chassis	E-2
E-2	DSA-47(RS-232C/V.24) FLAP Register Definition	E-6
E-3	DSA-47(RS-232C/V.24) Direct-Connect FLAP Timing Configurations	E-9
E-4	DSA-47(RS-232C/V.24) FLAP Synchronous Direct-Connect Cable Connections	E-10
E-5	DSA-47(RS-232C/V.24) FLAP Pin Connectivity Facilities	E-13
E-6	DSA-46 Direct-Connect FLAP Register Definition	E-15
E-7	Control Signal Interface Configurations	E-17
E-8	DSA-46 Direct-Connect FLAP Timing Configurations	E-20
E-9	Current Loop FLAP Register Definition	E-21
E-10	X.21(DSA-49) FLAP Register Definition	E-24
E-11	Automatic Calling Unit FLAP Register Definition	E-28
E-12	Connector Wraparound Loop-Back	E-30
E-13	Loop-Back of Digit Signal Bits	E-31
E-14	303 FLAP Register Definition (Current Mode)	E-33
E-15	V.35 FLAP Register Definition (Balanced Line)	E-36
E-16	MIL-STD-188-114 FLAP Register Definition	E-41
F-1	NACLA/DC Register Access	F-2
F-2	Asynchronous Data Format	F-3
F-3	NMLCP Communications Subsystem	F-4
F-4	NACLA/DC Block Diagram	F-5
F-5	NACLA/DC Line Registers - Compatible Functionality Only	F-7
F-6	NACLA/DC Line Registers - New Functionality Only	F-11
F-7	NACLA/DC Line Registers - Composite Bit Assignments	F-12

TABLES

Table		Page
1-1	Comparison of Characteristics	1-9
1-2	CCP Instructions	1-22
2-1	Summary of Main Memory Program Input/Output Instructions	2-14
2-2	Channel Number Assignments (Decimal)	2-37
3-1	Instruction Symbols	3-6
3-2	Format of Macrocalls for CCP Generation Control Statements	3-9
3-3	Instruction Timing and Location	3-13
3-4	Op Code Map	3-164
4-1	CCB Completion Conditions	4-19
5-1	Line Control Table	5-2
6-1	NMLCP Channel Number Addressing	6-2
6-2	Cyclic Redundancy Check Information	6-8
6-3	Data Transfer Related to Adapter Type and Operation Mode	6-9
6-4	Possible Settings for NMLCP Fixed-Rate Clock	6-11
B-1	Channel Numbering	B-19
C-1	Digital Signal Character Set	C-5
C-2	Automatic Calling Equipment/ACUF Interface Signals.	C-16
E-1	DSA-47(RS-232C/V.24) FLAP Interchange Circuits Supported	E-12
E-2	DSA-46 FLAP Interchange Circuits	E-18
E-3	Current Loop FLAP Interchange Circuits	E-22
E-4	X.21(DSA-49) Interface	E-26
E-5	Automatic Call Unit FLAP Interface	E-31
E-6	303 FLAP Interchange Circuits	E-35
E-7	V.35 FLAP Interchange Circuits	E-38
E-8	MIL-STD-188-114 FLAP Interchange Circuits	E-40
E-9	Voltage Polarity of DTE/DCE Unbalanced Voltage Interchange Circuits	E-42
F-1	Channel Numbering	F-15

Section 1
INTRODUCTION

The New Multiline Communications Processor (NMLCP) is a primary hardware product that will support the communication requirements of networking. This includes both primary and secondary networks, front end and satellite processors, in nodes of DPS 6/Level 6-based networks and Binary Synchronous Communication (BSC) and Systems Network Architecture (SNA) compatible nodes, and general communications facilities. It supports local and remote terminals in a multifunction environment including transaction processing, data entry, and word processing. The NMLCP also supports the requirements of public Value Added Network (VAN), and Public Data Network (PDN).

The NMLCP is the successor to the Multiline Communications Processor (MLCP) and, as such, provides Compatible MLCP functionality and additional Extended functionality. The Compatibility and Extended modes of operation are described in this section.

The NMLCP is a programmable, firmware-controlled communications processor that provides an interface with the DPS 6/Level 6 Megabus Network and up to 16 full-duplex communications lines. Low-speed lines (up to 300 bits per second), medium-speed lines (from 600 to 20,000 bits per second), and high-speed lines (broadband) (up to 100,000 bits per second) are supported by the system.

NMLCP OPERATION

The NMLCP performs data transfer between main memory and the Communications Adapter (CA) of each line; each character or byte is handled individually by the NMLCP, hereafter referred to as the Processor. This Processor uses a combination of firmware, software-generated parameters and software-generated Channel Control Programs (CCPs) to produce the appropriate user-defined action for each data element.

The CCP controls the communications equipment interface, connects, disconnects, and monitors the state of the line, and transmits and receives data. This data is then processed one character at a time, with the ability of the CCP to transfer, translate, edit, delete, add elements, and recognize special characters; the protocol of the lines can also be supported.

The software components required to support the Processor are the main memory resident driver program and the local store (CCP RAM) resident Channel Control Programs (CCPs).

The interface between main memory and the CCPs is the Line Control Tables (LCTs) and Communications Control Blocks (CCBs). Each channel has a unique and independent LCT and CCB area, thus making the CCPs reentrant. The LCT contains configuration, status, and control information while the CCBs contain address, range, and the control and status required to execute direct memory access transfers between the Processor and main memory.

The main memory program is used to control, by the execution of I/O orders, the operations of the controller. The main memory program has the responsibility to initialize, load the CCP, set up the line and control information, set up CCBs, start and stop CCP execution, and process the interrupts from the controller.

The Processor also provides other services for all channels such as parity and Cyclic Redundancy Check (CRC) checking, timers, and the means by which the channels are controlled by and report to the CPU software.

The Processor includes special functionality to support online testing of individual channels, to provide for a remotely initiated initialization/restart, and to support polling of multidrop lines.

NMLCP HARDWARE OVERVIEW

The Processor is a single primary circuit controller board using a single interface slot of the DPS 6/Level 6 Megabus Network. The Processor provides the common elements shared by all communications lines. It is one component of a set of communications components consisting of communications controllers, Communications Adapters (CAs), and Flexible Line Adapter Packages (FLAPs). These components provide flexibility of choice as to number of lines, maximum line speed, data format/protocol, and Dataset/DCE interface. A typical multiline communications subsystem consists of a communications controller (i.e., the NMLCP), up to 4 CAs and up to 16 FLAPs. The CAs mount on and connect to the Processor via a special adapter interface, the FLAPs mount in a separate chassis and connect to the CA via a FLAP bus. Figure 1-1 shows a subsystem with an NMLCP, 4 CAs, and 16 FLAPs.

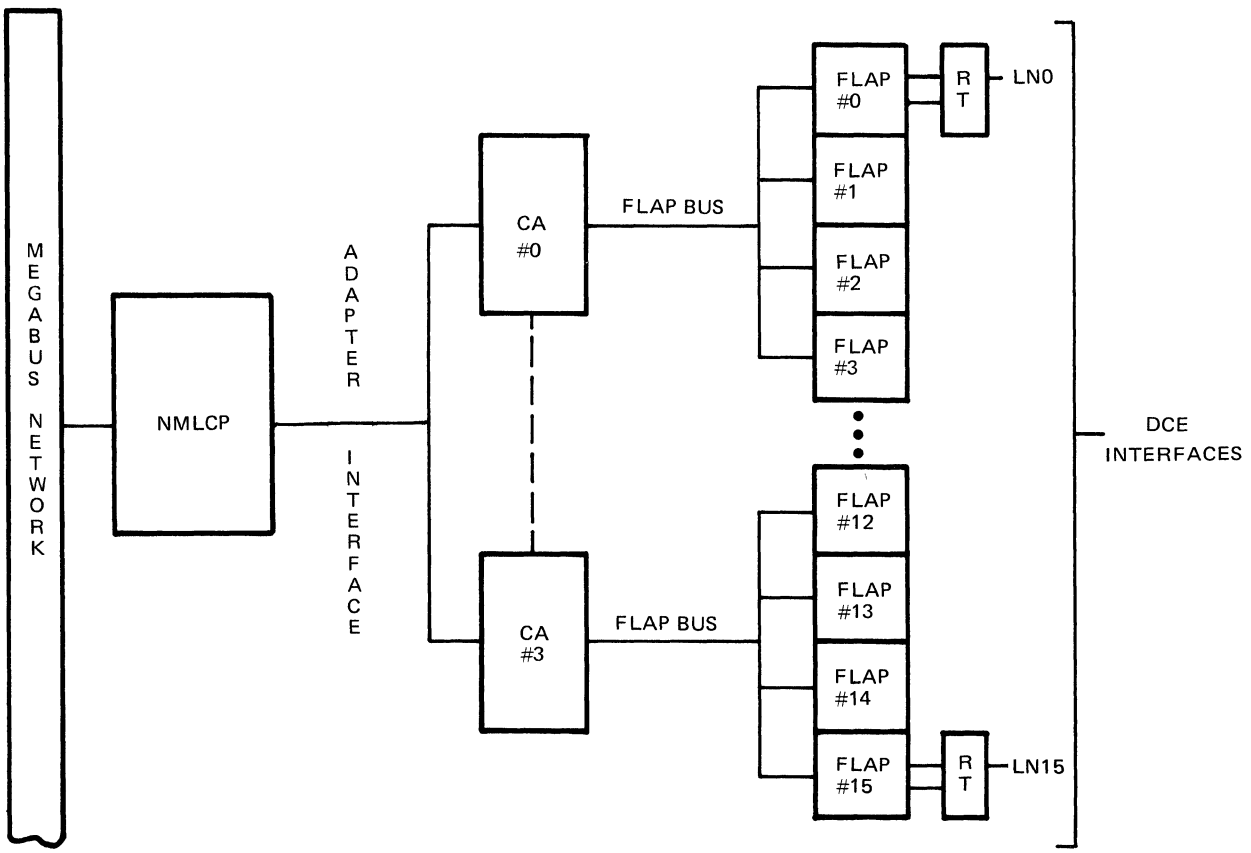


Figure 1-1. NMLCP Communication Subsystem

Communications Adapters (CAs)

The CA is a quarter-sized adapter board which attaches to the controller board via a special adapter interface. The CA contains line data format hardware, serial-parallel conversion, line buffering, and line protocol generation/recognition functions such as Flags, zero bit insertion Cyclic Redundancy Check (CRC), zero bit insertion and deletion, etc. CA boards of various types will support up to four lines, speeds up to 100K bps, and synchronous/asynchronous or High-Level Data Link Control (HDLC) protocols.

The three Communications Adapters availiable are:

1. New Multiline Controller Synchronous/Asynchronous Adapter (NSAA)

2. NMLCP Broadband HDLC/Synchronous Adapter (NBHSA)

3. NMLCP Asynchronous, Integrated RS-422 Direct Connect Adapter (NACLA/DC).

Item 3 is the integrated RS-422 direct connect interface and is considered to be a FLAPless CA. The functionality normally supported by this FLAP has been integrated on the adapter board.

Flexible Line Adapter Packages (FLAPs)

The FLAPs are mounted in a separate chassis and connect to the CA via a bus cable. The FLAPs contain the electrical and mechanical characteristics of the interface and the line interface-specific hardware to the DCE. The FLAPs provide the following interfaces: RS-232C/V.24, MIL-188-114, RS-422A Direct Connect, 20-mA current loop, Autocall, X.21, V.35, and Bell 303. Refer to Appendix E for a detailed FLAP description.

Random-Access Memory (RAM) Layout

The Processor contains four random-access memories. The first is the 16K byte Local Store (CCP RAM) which holds CCPs for all channels. Software determines which size local store is present by writing and then reading the RAM using the RAM Data Transfer operation described in the subsection entitled "RAM Data Transfer." The second is the 2K byte CCB RAM providing eight CCBs per channel. The third RAM is 4K bytes in size and contains the LCTs. The fourth RAM is the bus processor RAM containing various protected registers; these are not accessible to CCPs. All RAM locations consist of eight bits of information plus one parity bit. The CCP RAM also has a parity bit per byte location. If a RAM parity error is detected, the RAMP bit in interrupt B status (FC=06) is set and a stop I/O is performed on the channel.

For compatibility with the MLCP, the LCT, CCB, and CCP areas shown in Figure 1-2 (equivalent to those of the MLCP) are visible to the CPU via Block Read/Write orders. The set of LCTs and CCBs that are visible depends on the line number used for the order. (Lines 0 through 7 obtain LCTs and CCBs for lines 0 through 7 and lines 8 through 15 obtain LCTs and CCBs for lines 8 through 15.) The same CCP area is accessed for all lines.

Note that this memory map is created by the Processor specifically to resemble that of the MLCP; it should also be noted that CCP memory locations 3584 through 4095 are not accessible via Block Mode Access and that only the first four CCBs of each channel are accessible.

All of the information required by the Processor to perform its processing is stored locally in its RAMs. The RAMs may be written or read by the CPU as desired. RAM contents include the following general classes of information:

1. One or more CCPs

2. Configuration and control information for all lines

3. Processor context for each channel

4. DMA range and address for each channel.

The nature of the Processor makes CCPs inherently reentrant so that one CCP can service a number of lines. The maximum processing capacity of the Processor is over 30,000 characters per second, using minimum loop CCPs. The existence of an upper limit of capacity places practical limits on the configuration and also affects the amount of off-loading that the Processor can give the CPU.

Block mode access to NMLCP memory is expected to be used only on a line that is operating with software originally written for the MLCP.

The entire LCT, local store CCP, CCB store and bus processor areas of the Processor are accessible to the CPU via the RAM Data Transfer operations. This type of access can be used regardless of the mode (Compatibility/Extended) of the line that is used. Note that CCP area locations 512 to 3583 and 4096 to 22143 are the same physical locations whether accessed via Block mode or RAM Data Transfer operations.

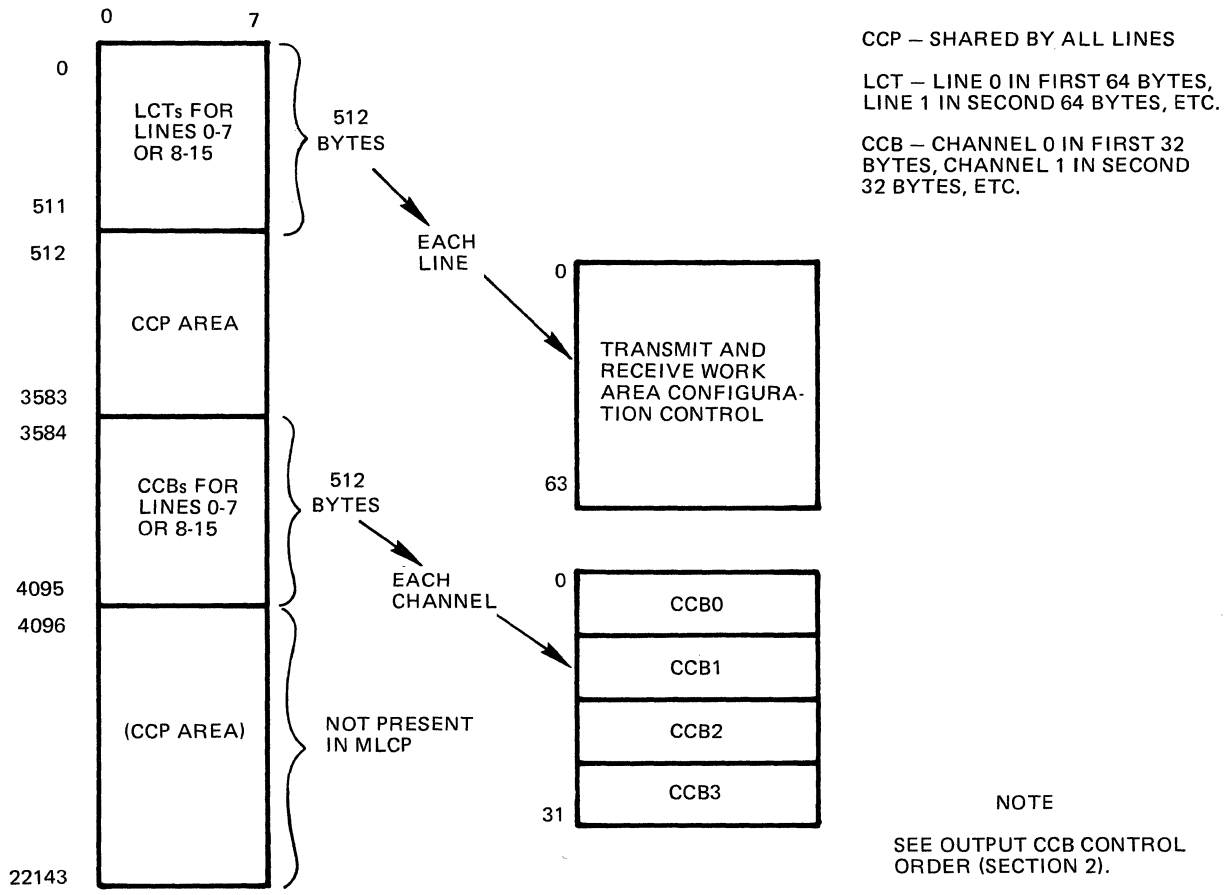


Figure 1-2. NMLCP-Compatible Memory Map (Block Mode Access)

The RAM memories that are accessed are shown in Figure 1-3. Access to the CCB and bus processor RAMs is intended for T&V use only.

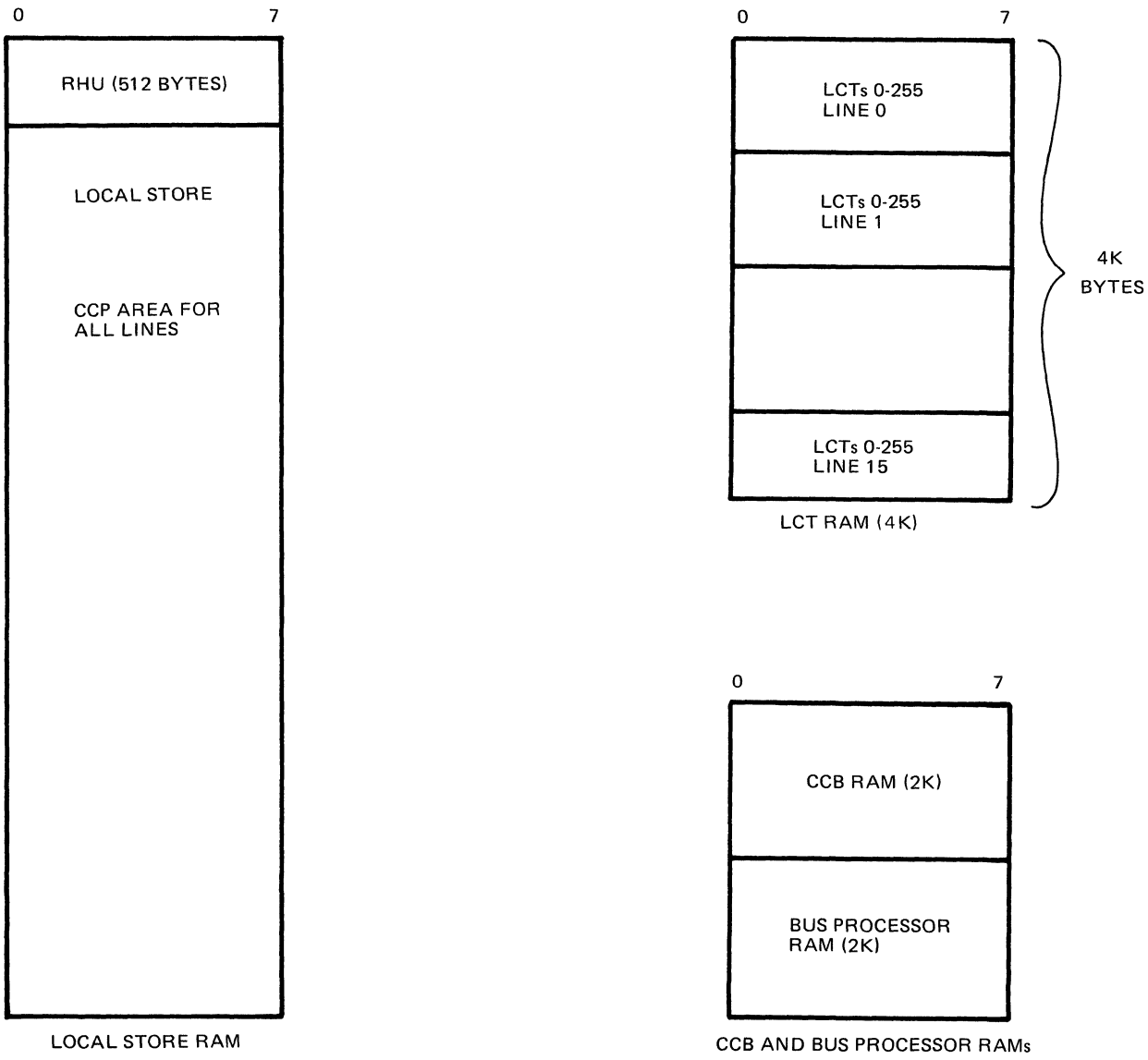


Figure 1-3. NMLCP Memory Map (RAM Data Transfer Access)

A block diagram of the Processor is shown in Figure 1-4. The figure contains:

1. Bus service processor which executes I/O commands issued by the CPU and performs other operations on the DPS 6/Level 6 bus such as channel buffer to memory DMA.

2. Channel program processor which executes CCPs on behalf of the data channels.

3. Random Access Memories (RAMs) for CCBs, LCTs, and Scratchpad, shared by the bus and CCP processors.

4. Local Store (CCP RAM) which contains the CCPs for all channels.

5. Personality PROM (located on the CA) which contains the code that adapts MLCP compatible instructions to the USARTS contained on the CA.

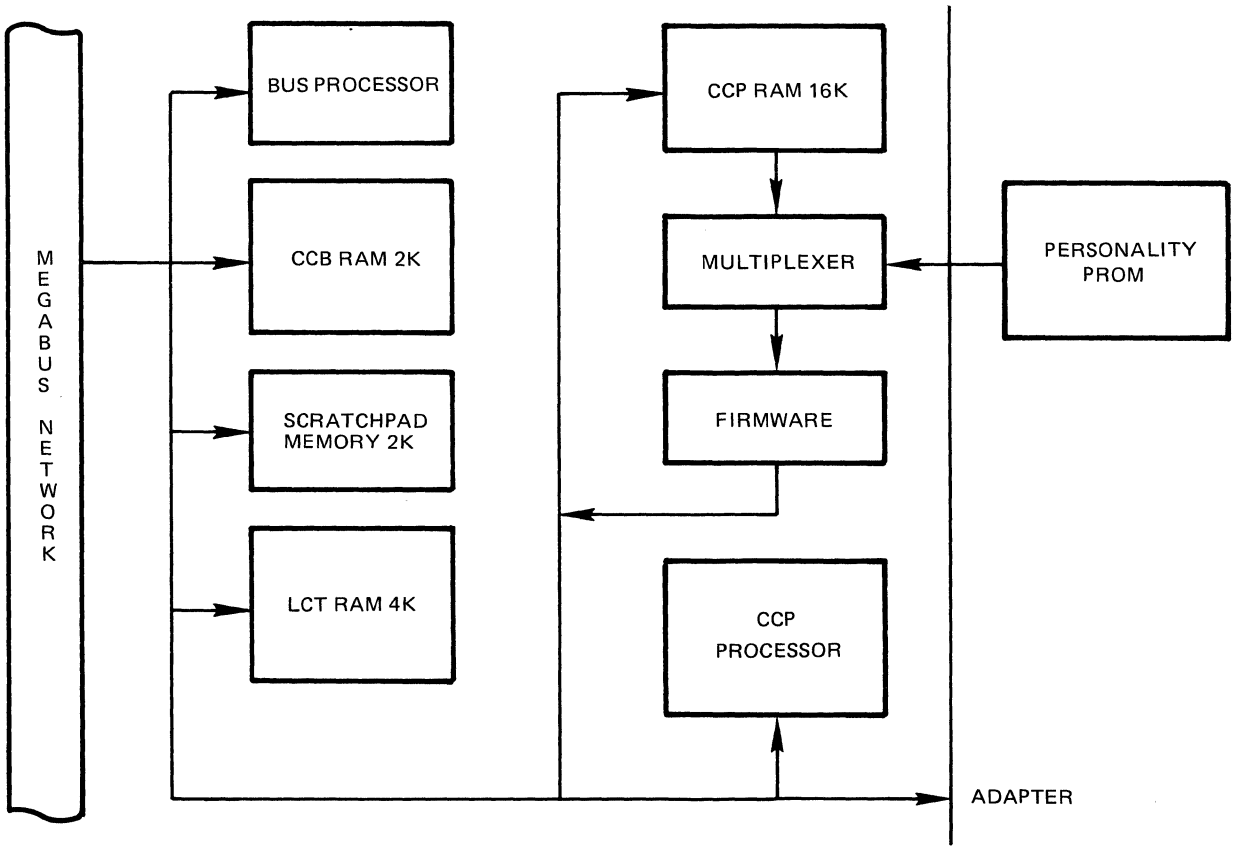


Figure 1-4. NMLCP Block Diagram

COMPATIBILITY WITH MLCP

The NMLCP provides a Compatibility mode of operation so that the existing CPU driver software written for the MLCP will operate the NMLCP. CCP compatibility with the MLCP applies at the object level; CCPs originally written for the MLCP in MLCP CCP source language and processed through the MLCP macroprocessor and the DPS 6/Level 6 assembler, produce an object code CCP which can then be loaded and executed. Certain practices in such source programs may, however, cause compatibility problems; i.e., time-dependent programming, and use of undefined bits in MLCP firmware working registers of the LCT.

Table 1-1 provides a summary of compatible characteristics of the MLCP and NMLCP.

Table 1-1. Comparison of Characteristics

Component	MLCP	NMLCP
Random Access Memory (RAM)[a]		
Dedicated to LCTs	512	4,096
Usable for CCPs	3,072	15,872
Dedicated to CCBs	512	2,048
Total Size	4,096	22,016
Registers/Indicators[b]	12-bit P-register 8-bit R-register E-indicator V-indicator LC-indicator LB-indicator	16-bit P-register 8-bit R-register E-indicator V-indicator LC-indicator LB-indicator 8-bit B-register AR-indicator C-indicator Z-indicator SB-indicator
CCP Instruction Set	43 Instructions: 15 Branch 14 Double Operand 2 Input/Output 2 Send/Receive 10 Generic	140 Instructions: 22 Branch 91 Double Operand 6 Input/Output 2 Send/Receive 19 Generic

Table 1-1 (Cont). Comparison of Characteristics

Component	MLCP	NMLCP
Line Control Tables (LCTs)	8 LCTs, each 64 bytes: 32 bytes for Receive 32 bytes for Transmit	16 LCTs, each 256 bytes: 128 bytes for Receive 128 bytes for Transmit
Communications Control Blocks (CCBs)	64 CCBs, each 8 bytes: 4 CCBs per channel	256 CCBs, each 8 8 CCBs per channel
Communications-Pacs, Adpaters Communications Lines & Channels	From one to four with 2 lines each From one to eight with 1 receive/1 transmit each	From one to four with 4 lines each From one to sixteen with 1 receive/1 transmit each
Line Speed	From 45 to 72,000 bps	From 45 to 100,000 bps
Flexible Line Adapter Packages (FLAPs)	--	From one to sixteen

[a] All figures are in bytes.
[b] See "NMLCP CCP Registers and Program Indicators" in Section 3 for functional descriptions.

Areas where the NMLCP differs from the MLCP to provide improved functionality, which may cause a need for software modification in some cases, include the following:

1. The device ID and extended ID of the NMLCP differs from the MLCP.

2. The Input Line Status order delivers eight bits of information to the CPU; bits 0 through 7 are obtained from the attached FLAP (register FR5).

3. The NMLCP will accept up to eight CCBs per channel as compared to four CCBs per channel in the MLCP.

4. The NMLCP will not permit DMA transfer to occur beyond the boundaries of the data area defined in a CCB.

5. The upper two bits in instructions and I/O orders that address the LCT are used for addressing; in the MLCP these were ignored. In the NMLCP LCT 70, 71, 102, and 103 directly affect operation.

6. The NMLCP does not duplicate the functions of the firmware work registers of the MLCP LCT in all cases.

7. The upper four bits in effective CCP RAM addresses are used for addressing; in the MLCP these were ignored.

8. Execution times of CCP instructions and I/O orders differ from that of the MLCP.

9. The "Clear to Send" signal from the modem is not directly connected to the adapter logic. Thus there may be a delay in the adapter becoming aware of a change to a low state of this signal.

10. The RAM Data Transfer operation, described later in this section, must be used to access the full CCP RAM of the NMLCP.

11. The NMLCP provides specific fault indications for illegal orders and instructions and for RAM parity errors and interrupts the CPU when they occur (see Appendix A).

12. The Search for Synchronization instruction requires two sync characters. A CRI will be generated on the third character. Refer to Appendix A for details.

13. On the SYNC/ASYNC (NSAA) adapter, the first sync character is swallowed by the hardware; all others are passed on to the CCP. On the HDLC/SYNC (NBHSA) adapter, the first two characters are swallowed before the CCP receives any.

In most respects, the Processor may be treated as two MLCPs with adjacent channel group addresses on the Megabus network; however, the Hard Initialize I/O order affects 16 lines instead of 8. Furthermore, the memory map (Figure 1-2) differs. Full compatibility with existing software (both CPU and CCP) applies only to synchronous/asynchronous operation. For HDLC operation and Autocall, some modification of existing software will be necessary.

It should be noted that some of the new I/O and CCP instructions and features can be used with the Compatibility mode while others cannot. Details can be found in Section 3. The mode (Compatibility/Extended) can be set up on a per-line basis; all lines are initialized to the Compatibility mode. Any mix of Compatibility and Extended mode lines can be used.

NEW FUNCTIONALITY OF THE NMLCP

The NMLCP has a number of new features and concepts over and above those of the MLCP. Among these are the use of lists for processing of CCBs and an improved way of handling the queuing of interrupts. Other new features are:

1. Larger LCT space per line (256 bytes/line)
2. Larger CCP memory (16K bytes)
3. Channel timers
4. Two-level CCP processing for each channel
5. Remote restart and bootload
6. Polling can be handled within CCP by using timers and FIFOs
7. Load monitoring
8. A major expansion of the instruction set with about three times as many instructions as the MLCP
9. Program fault detection
10. 16-line capacity
11. An executive channel that provides access to NMLCP memory areas while data channels are operating.
12. Functionality to support PARS (CRC6)
13. Programmed stacks for data/parameter/subroutine nesting
14. Arithmetic and shift instructions
15. Bit manipulation instructions
16. Improved channel priority scheme
17. Two interrupt level parameters per channel (A/B).

Communications Control Blocks

Communications Control Blocks (CCBs) are used by the Processor to define the Address, Range, Control, and Status for each data transfer between the Processor and main memory.

The main memory program dynamically sets up the Address, Range, and Control Field for a CCB by issuing an I/O order. At the completion of a data transfer, the CCB Status Field is updated to reflect the final status of the CCB.

The data buffers in main memory to or from which the data channels request DMA service are described by Communications Control Blocks (CCBs).

Each CCB contains an address and range field that defines a data buffer, and also includes other control and status information related to the handling of the particular message. This information permits a message, as transmitted over the line, to be placed or obtained from more than one data buffer. A detailed discription of CCBs is contained in Section 4.

Interrupt Handling

The scheme of handling interrupts in Extended mode is designed to reduce the amount of level changing required by the CPU.

Each channel of the Processor maintains a Deferred Interrupt Count register in its LCT area (LCT 77/109).

Whenever an event occurs on a channel that requires an interrupt to the CPU, an interrupt is attempted to the CPU according to the contents of the appropriate Interrupt Control register. If the interrupt is not accepted by the CPU, the channel's Deferred Interrupt Count register is incremented. When an interrupt is accepted by the CPU, the register is decremented if non-zero. Deferred interrupts posted in the register will be retried when the CPU sends a Resume Interrupt signal on the bus (i.e., a level change). However, the CPU can alternatively read each channel's Deferred Interrupt Count via a Read and Clear LCT order (FC=0E) (see Section 2). Thus, by reading the register, the CPU program can be made aware of the number of interrupts outstanding on a channel without the necessity of fielding each one separately via a level change and context switch.

In addition, to simplify the resolution of interrupts by CPU software, the Processor provides for the use of two interrupt levels by each channel. The prime level is designated as Interrupt A and is used to report the completion of channel data transfers on the communication line. The second level (Interrupt B) is used to report various other occurrences such as Data Set Scan events (see Appendix A), RAM Data Transfer completions (see "RAM Data Transfer" subsection) the occurrence of illegal I/O

orders or illegal instructions, and software-generated interrupts via the INTR instruction. For compatibility reasons, unless Interrupt B is specifically set up on a channel, only one interrupt level (Level A) will be used for all occurrences on the channel; in this case occurrences such as Data Set Scan and INTR instructions are reported in CCB status and not in Interrupt B status.

The deferred interrupt count does not include an Interrupt B which may have been deferred.

Executive Channel

The Processor provides Executive Channel firmware that is not associated with a communication line. One of its functions is to perform RAM Data Transfer orders for the CPU; this allows the CPU to load CCPs into the Processor's Local Store (RAM) or to load or dump LCTs without interfering with operation of the 16 lines. Other executive firmware performs background chores such as data set scan, timer update, and load monitoring.

An executive channel software aspect can also be created by implementing the concept in software. Executive channel software would exist as a set of routines that could be invoked by any data channel via a Vectored Jump Mechanism. Since these jump calls are intended to be performed at the channel's event service level (see "Data Service and Event Service Levels" in this section, Executive routines would run at lower priority than data service. Through suitable software and programmed gates, the CPU could cause any data channel to call into execution an Executive routine on its event level of service.

RAM DATA TRANSFER

RAM Data Transfer can be used for dynamic loading or dumping of the Local Store (CCP RAM) or the LCT RAM.

RAM Data Transfer operations require the execution of three I/O orders in a specific sequence. The first order to be issued is the RAM Data Transfer IOLD (FC=29/2D). This order specifies: (1) the main memory area into which or from which the transfer is to be made, (2) the direction of transfer, and (3) which Processor channel will signal completion of the transfer. The second order to be issued is the Output RAM Control IO (FC=15); this order specifies the starting address of an area in a Processor RAM into which or from which the transfer is to occur. The third order to be issued is the Output Processor Control IO (FC=01); this order specifies which Processor RAM is to be accessed (Bit 14 or 15 of control). The RAM Data Transfer starts upon receipt of this third order.

Ram Data Transfer is performed as a background function during intervals when no data channel needs service. On completion or termination of the operation, the RTT bit is set in Interrupt status B (FC=06) of the channel to which the RAM transfer IOLD (FC=29/2D) was issued. If any errors occur during the operation, the operation is immediately terminated and the error is flagged in Interrupt status B. An interrupt is sent on completion or termination using Interrupt Register B.

CAUTION

The CPU software must not attempt to start or set up a RAM Data Transfer while the Processor is still performing a previously initiated RAM Data Transfer. This will have unspecified effects on the transfer being performed.

Data transfer in the Processor occurs in two directions independently controlled for each channel with receive channels putting data into main memory CDBs, and transmit channels taking data out of main memory CDBs. The interface for receive and transmit is basically the same and each is discussed below. It is assumed here that a Start I/O was issued to initiate data transfer, and that the CAs are enabled for receiving and transmitting data.

BLOCK MODE READ/WRITE

Block mode access to the NMLCP random access memories has been implemented for MLCP compatibility. Block mode access is a logical access. The set of LCTs and CCBs that are visible depend on the line number used for the order. Lines 0-7 obtain LCTs and CCBs for lines 0-7. Lines 8-15 obtain LCTs and CCBs for lines 8-15. The same CCP area is accessed for all lines. The RAM address determines what logical area will be accessed. An address of 0-511 will access LCT RAM. An address of 3584-4095 will access CCB RAM (only the first four CCBs of each channel are accessible). An address of 512-3583 and 4096-16383 will access CCP RAM. CCP RAM locations 3584-4095 are not accessible via Block mode address. The RAM data transfer operation must be used to access the full CCP RAM.

When a Block mode write is completed, the transmit channel will be returned to a stop input/output condition. The main memory program will be automatically interrupted at the interrupt level previously established for the transmit channel. (If no interrupt level has been established for this channel, the interrupt will not occur.)

DATA SET SCAN FIRMWARE

The Data Set Scan firmware will run after all channel data service needs are met. This firmware reads LR5 or FR5 of each data channel and records it in one of the channel's LCT locations. Further, if Data Set Scan is specified for that channel, the firmware scans the specified bits, in status, for any change from previous value, and takes the action specified in LCT 8 if such change has occurred.

PROCESSOR LOAD MONITOR

The Executive channel maintains a 16-bit Load Monitor register that provides an approximate indication of the current amount of spare load capability (i.e., processing cycles) of the Processor. The Load Monitor register is incremented by a background firmware routine. This firmware runs once after each completed round-robin scan of Event service. If there are no demands on the Processor (no data CRI and no Event service requests), a complete scan takes about 200 microseconds; as the Processor becomes busier, this register is tallied less frequently. The CPU may read this register at appropriate intervals and compute an average load factor.

The Load Monitor register is read via the Input Load Monitor order (FC=14).

VECTORED JUMP MECHANISM

A mechanism can be programmed to provide a means whereby a CCP can call for execution of a routine whose location in Local Store (CCP RAM) is not known to the CCP. The mechanism assumes that transfer vector(s) will be loaded into some preagreed location(s) in the LCT of the channel. To perform the Vectored Jump, the channel CCP executes a JSRV instruction; this causes a jump via the transfer vector and saves a return point (next instruction in the calling CCP) in the channel's stack.

The set of Executive routines can serve to gather various statistics (retransmissions, overruns, etc.) for any channel (perhaps keeping them in the Extended LCT where they can easily be retrieved by the CPU). The data channel CCP thus will have JSRV instructions embedded in its code at suitable points and, depending upon the transfer vector loaded into the channel's LCT, appropriate Executive routines (or a Null routine) can be invoked when needed on a channel basis without affecting the data channel CCP itself (which may be shared by many channels).

Return from an Executive routine is accomplished by the RETJSR instruction.

Generally, Executive routines should be invoked only from the event level of a channel (see "Data Service and Event Service Levels" subsection) since data service routines should be self-contained for performance reasons.

Channel Service Multiplexing

The general channel servicing scheme of the NMLCP is similar to that used in the MLCP, wherein, when a channel requires service of the controller (for example to store or fetch a character), it sets a Channel Request Interrupt (CRI) condition. The controller will, at some point, service this CRI by commencing execution of the CCP associated with this channel. Typically, the CCP will, after handling the character, relinquish control by executing a WAIT instruction and the controller then proceeds to choose another requesting channel to service next. A priority scheme is implemented in the Processor to select which channel to service next.

WAIT FUNCTIONALITY

The WAIT instruction is provided as the means by which a CCP can relinquish control to another channel. If the CCP is performing data service (via SEND and RECV instructions), the channel will not relinquish control on a WAIT until further data service is needed on that channel and the channel is the highest priority channel needing service. (It is expected that channel adapters may have built-in data FIFOs to ease service timing constraints.) If the CCP is performing Event service, its WAIT instruction will always relinquish control. When control is relinquished, the channel's context is saved.

PAUSE FUNCTIONALITY

As a safeguard against one channel monopolizing the controller, due to execution of a CCP with a long string of instructions, the NMLCP provides a hardware-enforced Pause function compatible with that of the MLCP.

When the Pause function is invoked, the channel's context is saved, its Pause count in LCT 78/110 is incremented, and service is granted to the currently highest priority outstanding service request (the Paused channel itself is included in this evaluation).

The RAM Data Transfer function of the Executive channel has a built-in Pause which occurs after each byte transfer. These pauses are not tallied.

The Processor Pause function allows up to 32 CCP instructions to be executed by a channel before a Pause operation is forced on the channel (see Note). If a channel attempts to execute more than that number, the firmware will cause a pseudo-WAIT function to occur, transparent to the CCP. As a result, higher priority

lines will have a chance to gain service. Pause occurrences are tallied in LCT 78/110. The Pause function can be disabled on data service on a per channel basis via bit 7 of LCT 5/37. The Pause function cannot be disabled for Event service.

NOTE

Under certain conditions a forced Pause may be initiated. Any line register instruction; i.e., IN, OUT, will be paused with the execption of IN5 and OUT2. The forced pause is also disabled by bit 7 of LCT 5/37.

PRIORITY SCHEME

The Processor channel priority scheme is based partly on channel number and partly on the reason for the channel request for service. Requests for data service are given highest priority. Within this group, service is granted on a round-robin or fixed basis as between the attached adapters as specified in the Processor control order (FC=01). Adapters have an internal priority for data service scheme between channels and can signal their data service requests as being high or low priority. Next priority is given to Executive channel firmware. Lowest priority is given to servicing of events for all of the data channels (see "Data Service and Event Service Levels" subsection below); within this group, service is granted on a round-robin basis. The scheme is shown in Figure 1-5. When the Processor is set up for a fixed priority basis, it gives the group of adapter high-priority data service requests a higher priority than the group of adapter low-priority data service requests. Within each group, priority is determined by the line number position of the adapter (lowest line position is highest priority).

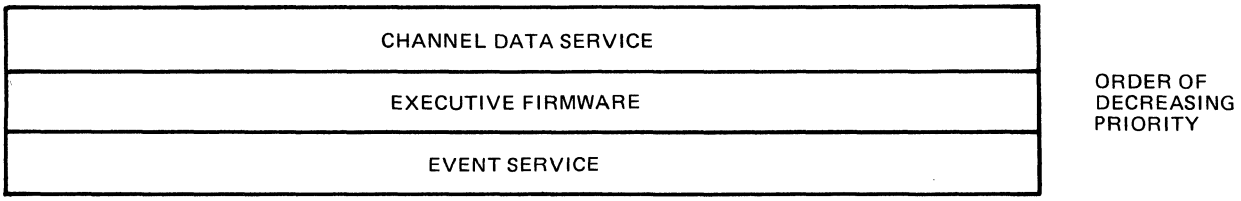


Figure 1-5. NMLCP Priority Scheme

It is envisioned that the Processor will normally be set up for a fixed adapter priority basis.

DATA SERVICE AND EVENT SERVICE LEVELS

Each data channel can execute on either of two priority levels, depending on the reason for servicing it. The higher level is used exclusively for data service while the lower level is used for servicing various Events related to the channel such as Timer Runout, Data Set Scan status change, Start I/O, and various software-defined tasks such as message level functions.

If LCT 70 and 71 contain zeros (as for executing CCPs originally developed for the MLCP), both levels for each channel will use the same context save area (e.g., LCT 6,7 or 38,39 for the P-counter).

If LCT 70 and 71 are non-zero each of the two levels for each channel will have its own context save area (e.g., LCT 6,7 or 38,39 for data service P-counter and LCT 70,71 or 102,103 for Event service P-counter); thus each channel can execute two independent tasks concurrently. The event service CCP will determine which type of event it is being called on to service by reading LCT 75 or 107 and testing it. On completion of handling of an event, the CCP routine must clear the particular bit(s) in LCT 75 or 107; this must be done in one indivisible operation using the Load Bit indicator and set False (LBF) instruction. Note that CCP software must use the LBF instruction as the means for setting a code into the Software Interrupt (SWI) field of LCT 75 or 107.

The CPU software may also invoke Event service to perform such functions as channel status or channel statistic gathering. This presupposes the existence of a CCP to perform these functions being resident in the Processor.

By definition, an event service request is outstanding for a channel whenever bits 2 through 7 of that channel's Activity Flags (LCT 75 or 107) are non-zero.

Data Transfer

Data transfer in the Processor occurs in two directions independently controlled for each channel with receive channels putting data into main memory CDBs, and transmit channels taking data out of main memory CDBs. The interface for receive and transmit is basically the same, and each is discussed below. It is assumed here that a Start I/O was issued to initiate data transfer, and that the CAs are enabled for receiving and transmitting data.

RECEIVE

Incoming data from DCEs (usually bit serial) is converted by the configured USART of the CA into data elements of eight bits or less. Each time a data element is accumulated, the CA issues a CRI to the Processor. This tells the Processor that the CA has a data element accumulated for transfer to the R-register of the Processor where the data element can be analyzed if that is desirable.

When the Processor services a CRI, it must first load a CCP pointer and the program indicators into the Processor (it does a context restore for this channel). The Processor then, using the CCP pointer as the next instruction location, executes that CCP instruction on behalf of the channel. Since CCPs are loaded by the user, the Processor channel is under complete user control at this point.

Typically, the CCP will load the R-register with the accumulated CA data element, analyze it, and then take some specific action such as deleting it, converting it to some other data pattern, adding the data element to the CRC residue, or relinquishing it to the main memory.

Storing a data element into the Processor data buffer is essentially a request to transfer the data to the proper CDB via a CCB. The mechanics of this transfer is accomplished by Processor hardware and is not visible to the CCP.

While a CCP is executing, the entire contents of the LCT and the LRs may be read or written for that channel. When the channel request interrupt has been serviced, the CCP relinquishes control for the next Processor function by executing a WAIT instruction.

Under control of the CCP, CCBs are fetched as needed to accommodate the incoming data with CCBs being completed by range exhaust or end-of-block indications in the data itself.

Error conditions that may occur are reported in CCB Status.

TRANSMIT

Transmit is a duplicate of the receive with the data flowing out instead of in. Configured USARTs convert data elements of eight bits or less into DCE-compatible data (usually bit serial). Each time the CA requires another data element, the CA issues a CRI to the Processor. This tells it that the CLA needs a data element for transmission.

When the Processor services this CRI, it must first load the CCP pointer and the program indicators into the Processor similar to the receive case.

Typically, the CCP will load the R-register with the next data element. This data element is transferred from CDB in main memory by a DMA operation.

The CCP also has the option of loading the R-register with CCP-generated data, and of extracting constants or data elements out of the LCT configuration data and LCT work area. In this way, special data elements such as SYN, STX, ETX, HDLC address and control bytes, can be inserted into the data stream by a CCP.

When the CCP has completed the operations for this CRI, the CCP relinquishes control for the next Processor function by executing a WAIT instruction.

The CA will accomplish the data transmission of any data elements that it is given without any other Processor action.

Under control of the CCP, CCBs are fetched as needed to supply outgoing data with CCBs being completed by range exhaust.

Error conditions that may occur are reported in CCB Status.

TEST MODE/DIRECT-CONNECT CLOCK

Synchronous channels generally use the clock supplied by the DCE (modem). The Processor has an internal clock that is used when a line is running in Internal Loop Test mode. Clock speed is selected by switches on the Processor; one switch selects the speed for lines 0 through 7 and the other for lines 8 through 15. Clock speeds that can be selected are:

0.8K bps	28.7K bps
1.2K bps	31.9K bps
1.75K bps	38.0K bps
2.15K bps	57.8K bps
2.4K bps	76.8K bps
4.8K bps	114.3K bps
9.6K bps	153.6K bps
19.2K bps	307.2K bps

The clock can also be used in synchronous Direct-Connect applications; this ability is invoked by setting the clock source bit in the adapter's LR2. Note that, as indicated above, lines 0 through 7 can operate at some one speed and lines 8 through 15 at another speed when in Direct-Connect mode.

NOTE

When directly connecting an NMLCP to an NMLCP in synchronous applications (including HDLC) only one end should source the clock. To ensure that this occurs, the clock source bit (LR2, bit 4) should be enabled for one end only.

CCP INSTRUCTIONS

Table 1-2 provides an alphabetical mnemonic list of the NMLCP instruction set. The mnemonic in parenthesis is the hardware equivalent mnemonic where it differs.

Table 1-2. CCP Instructions

Instruction	Function
ADD	Add R-Register with LCT
ADD (ADDB)	Add B-Register with LCT
ADD (ADDRN)	Add R-Register with B-Referenced LCT
ADD (ADIR)	Add R-Register with IMO
ADD (ADIB)	Add B-Register with IMO
AND	And R-Register with LCT
AND (ANDB)	And B-Register with LCT
AND (ANDRN)	And R-Register with B-Referenced LCT
AND	And R-Register with IMO
AND (ANIB)	And B-Register with IMO
B	Unconditional Branch
BS	Branch to Subroutine
BARF	Branch on Adapter Ready False
BART	Branch on Adapter Ready True
BCF	Branch on Carry False
BCT	Branch on Carry True
BEF	Branch on Equal False
BET	Branch on Equal True
BLBF	Branch on Last Block False
BLBT	Branch on Last Block True
BLCF	Branch on Last Character False
BLCT	Branch on Last Character True
BSBF	Branch on Most Significant Bit False
BSBT	Branch on Most Significant Bit True
BSF	Branch on Status False
BST	Branch on Status True
BVBF	Branch on Valid Block False
BVBT	Branch on Valid Block True
BZF	Branch on Zero False
BZT	Branch on Zero True
C	Compare R-Register with LCT
C (CB)	Compare B-Register with LCT
C (CRN)	Compare R-Register with B-Referenced LCT
C	Compare R-Register with IMO
C (CBI)	Compare B-Register with IMO
CADD	Add R-Register with LCT Plus Carry
CADD (CADB)	Add B-Register with LCT Plus Carry
CADD (CADRN)	Add R-Register with B-Referenced LCT Plus Carry
CADD (CADIR)	Add R-Register with IMO Plus Carry
CADD (CADIB)	Add B-Register with IMO Plus Carry

Table 1-2 (Cont). CCP Instructions

Instruction	Function
CBNB	Cancel Block
CANC	Cancel Character
CBC	Calculate Block Check
CL (CLL)	Clear Contents of LCT
CL (CLN)	Clear Contents of B-Referenced LCT
CL (CLR)	Clear Contents of R-Register
CB (CLB)	Clear Contents of B-Register
DADD	Decimal Add R-Register with LCT
DADD (DADI)	Decimal Add R-Register with IMO
DEC	Decrement Contents of R-Register
DEC (DECL)	Decrement Contents of LCT
DEC (DECN)	Decrement Contents of B-Referenced LCT
DEC (DECB)	Decrement Contents of B-Register
DEC (DECX)	Decrement Local Store Location (Extended)
DEQ (PULLF)	Dequeue
ENQ (PUSHF)	Enqueue
FIN	Input FLAP Register
FOUT	Output FLAP Register
GIVE	Give a CCB back to CPU
GNB	Get Next Block (CCB)
IAS	Input Adapter Status to R-Register
ILP	Initialize LCT Pointer
IN	Input LR to R-Register
INC (INCL)	Increment Contents of LCT
INC (INCN)	Increment Contents of B-Referenced LCT
INC	Increment Contents of R-Register
INC (INCB)	Increment Contents of B-Register
INC (INCX)	Increment Local Store Location (Extended)
INTR	Interrupt CPU
INZ	Initialize (All Channels)
JS (JSR)	Jump to Subroutine via Displacement
JS (JSRV)	Jump to Subroutine via Vector
JUMP	Jump via Displacement
JUMP (JV)	Jump via Vector
LB (LBL)	Load Bit Indicator from LCT with IMO
LB (LBN)	Load Bit Indicator from B-Referenced LCT with IMO
LB (LBR)	Load Bit Indicator from R-Register with IMO
LB (LBB)	Load Bit Indicator from B-Register with IMO
LB (LBX)	Load Bit Indicator from Local Store with IMO
LBF (LBFL)	Load Bit Indicator from LCT with IMO and Set False
LBF (LBFN)	Load Bit Indicator from B-Referenced LCT with IMO and Set False
LBF (LBFX)	Load Bit Indicator from Local Store with IMO and Set False
LBT (LBTL)	Load Bit Indicator from LCT with IMO and Set True
LBT (LBTN)	Load Bit Indicator from B-Referenced LCT with IMO and Set True

Table 1-2 (Cont). CCP Instructions

Instruction	Function
LBT (LBTX)	Load Bit Indicator from Local Store with IMO and Set True
LD	Load R-Register from Main Memory
LD	Load R-Register from LCT
LD (LDB)	Load B-Register from LCT
LD (LDRN)	Load R-Register from B-Referenced LCT
LD	Load R-Register with IMO
LD (LDRB)	Load B-Register from R-Register
LD (LDIB)	Load B-Register with IMO
LD (LDBR)	Load R-Register from B-Register
LD (LDX)	Load R-Register from Local Store (Extended)
MC	Master Clear
NEG (NEGR)	Negate R-Register
NOP	No Operation
OAC	Output Adapter Control
OR	Or R-Register with LCT
OR (ORB)	Or B-Register with LCT
OR (ORRN)	Or R-Register with B-Referenced LCT
OR (OR)	Or R-Register with IMO
OR (ORIB)	Or B-Register with IMO
OUT	Output LR from R-Register
PULL (POPB)	Pull Value from Stack into B-Register
PULL (POPR)	Pull Value from Stack into R-Register
PUSH (PUSHB)	Push Value from B-Register onto Stack
PUSH (PUSHR)	Push Value from R-Register onto Stack
RECV	Receive Data
RETB (RET)	Return from Branch Subroutine
RETJ (RETJSR)	Return from Jump Subroutine
RHB	Return Held Block
SCF	Set Carry False
SCL (SCLR)	Shift Closed Left R-Register
SCL (SCLB)	Shift Closed Left B-Register
SCR (SCRR)	Shift Closed Right R-Register
SCR (SCRB)	Shift Closed Right B-Register
SCT	Set Carry True
SEND	Send Data
SFS	Search for Synchronization
SOL (SOLR)	Shift Open Left R-Register
SOL (SOLB)	Shift Open Left B-Register
SOR (SORR)	Shift Open Right R-Register
SOR (SORB)	Shift Open Right B-Register
ST	Store R-Register into Main Memory
ST	Store R-Register into LCT
ST (STB)	Store B-Register into LCT
ST (STRN)	Store R-Register into B-Referenced LCT
ST (STX)	Store R-Register into Local Store (Extended)
SUB	Subtract R-Register with LCT

Table 1-2 (Cont). CCP Instructions

Instruction	Function
SUB (SUBB)	Subtract B-Register with LCT
SUB (SUBRN)	Subtract R-Register with B-Referenced LCT
SUB (SBIR)	Subtract R-Register with IMO
SUB (SBIB)	Subtract B-Register with IMO
TLU	Table Look-Up
TQE (TSTFE)	Test Queue Empty
TQF (TSTFF)	Test Queue Full
WAIT	Wait (Suspend Channel)
XOR	Exclusive Or R-Register with LCT
XOR (XORB)	Exclusive Or B-Register with LCT
XOR (XORN)	Exclusive Or R-Register with B-Referenced LCT
XOR	Exclusive Or R-Register with IMO
XOR (XRIB)	Exclusive Or B-Register with IMO

Section 2
PROGRAMMING OVERVIEW

The New Multiline Communications Processor (NMLCP) is a programable, interrupt driven, firmware controlled communications controller. The controller's primary function is to transfer data between DPS 6/Level 6 main memory and a number of communication channels.

The CPU program, via input and output orders, defines, and controls the operation of the controller.

The programmer of a communications controller has the responsibility of creating three interactive components; the main memory program, controller memory program, and the definition and usage of Line Control Table work locations.

In addition to preparing a main memory program, which operates in the central processor, the programmer is responsible for creating the following software and writing it to the Processor.

1. Communications Control Blocks (CCBs)
2. Channel Control Program(s) (CCPs)
3. Line Control Tables (LCTs)

Before communications processing begins, the CCPs and LCTs must be prepared and then transferred to the Processor by means of a RAM Data Transfer or Block Mode Write. Loading methods are described in Section 1 under "RAM Data Transfer" and "Block Mode Read/Write." CCBs are dynamically supplied by the main memory program during communications processing.

MAIN MEMORY PROGRAM

One or more programs must reside in main memory to interact with the Processor. A main memory program interfaces with one or more communications channels. The general responsibilities of a main memory program are as follows:

1. It writes the user-defined CCP to the controller memory.

2. It sets up status and process control information.

3. It performs Processor and channel control functions such as initialization and starting/stopping channel operations when errors are detected.

4. It sets up the required CCBs and maintains them throughout execution of the application.

5. It maintains CDBs in main memory. This activity includes (1) handling the CDBs as they are completed, (2) supplying pointers to CDBs (for use by the CCBs) when required, and (3) monitoring the status and error conditions for each CDB and reacting appropriately.

6. It monitors the status of the data sets and adapters and takes appropriate action when certain changes take place.

Communications Control Blocks

For each channel, space exists in the upper Processor RAM for a list of eight consecutive 8-byte CCBs. Each CCB is used to store main memory address information that indicates the area to which communications data is to be delivered (receive operation) or from which communications data is to be obtained (transmit operation). The main memory area is called a Communications Data Block (CDB). Processor firmware uses the programmer-supplied information in the CCB when transferring data to or from the main memory CDB. The CCB also contains a control field and a firmware storage area for status and error indicators relating to the data transfer to or from the CDB.

Setup and maintenance of eight CCBs dedicated to each channel must be performed from the main memory program associated with that channel. A detailed description appears in Section 4.

Channel Control Program

A Channel Control Program (CCP) directs the movement of each data character through the Processor. The CCP can cause a data character to be processed in a simple, straightforward manner requiring a minimum of time. At the discretion of the programmer, the CCP can conduct more extensive checking and editing functions that require more NMLCP processing time. If the CCP is programmed to perform data character processing, keyboard basic message delimiting, and block-checking functions, this processing will be performed at the expense of the throughput speed.

Because of the nature of the instruction set and the design of the Processor, each CCP is reentrant and therefore usable for more than one channel. A major factor permitting reentrant CCPs is that the control information necessary to operate a channel is stored in the LCT and the CCB associated with only that one channel.

The following functions can be performed by a CCP:

1. Direct memory access to and from CPU memory
2. Processing of Communications Control Blocks (CCBs)
3. Error detection and handling
4. Parity and/or Cyclic Redundancy Check (CRC) generation and verification
5. Recognition of special characters and message delimiters
6. Data set and Adapter/Communications-Pac control, transfer, translation, editing, deletion, or addition of data elements.

All CCPs concurrently resident in the Processor share 16K bytes of RAM allocated for CCP usage.

The CCP is prepared by use of the appropriate program development facilities of the operating system.

Line Control Tables

For each line, space exists in the lower Processor RAM for one 256-byte Line Control Table (LCT). Each LCT is generally divided between the channels, with 128 bytes allocated to the receive channel of the line and 128 bytes allocated to the transmit channel. Each channel-related portion of an LCT comprises the following elements:

1. Data set status monitor control
2. LCT status
3. Cycle Redundancy Residue
4. Channel interrupt level
5. Data Set Control
6. Stack and Queue pointers
7. CCP execution pointer
8. Work locations for tempory storage and special character definition
9. Transmit/receive character configuration
10. Program-supplied input data
11. Programming work bytes
12. Programming information supplied by firmware
13. Bytes reserved for firmware use.

Note that a CCP of either channel can access all 256 bytes of its LCT.

The program-supplied input data bytes provide information required for character configuration, CCP control, interrupt control, firmware control relative to status and error conditions, and data set and adapter control.

The programming work bytes can be used in any way needed by the main memory program or CCPs.

Programming information supplied by firmware consists of status and error information related to the data set or adapter as well as to data transfer operations.

A number of bytes are reserved for firmware use. During Processor setup, these bytes may be overwritten with zeros. During subsequent communications processing, these bytes must not be modified by software.

Setting Up the Processor: Receiving and Transmitting Data

The following sequence of events provides a description of one possible way to set up the Processor before communications processing begins. Figure 2-1 describes this overall process, while Figure 2-2 indicates the general order of events as data is received over a channel, and Figure 2-3 indicates the general order of events as data is transmitted from a main memory program.

Refer to Figure 2-1 and perform the following steps from the main memory program:

1. Initialize the Processor and then use the RAM Data Transfer method to write a user-created block to the LCT area and the CCP area of the Processor RAM. (This action writes the user-desired values into each LCT to be used and writes one or more user-created CCPs into the CCP area.) Refer to "RAM Data Transfer" in Section 1 for a description.

CAUTION

A Block Mode Write should not be used to write into LCT areas other than the currently addressed channel when any other channels may be active. In addition, care should be taken when other channels are active not to write into their active LCT, CCB area.

NOTE

As an alternative, you can use a Block Mode Write (a sequence of input/output instructions in the main memory program) to write a user-created block to the LCT area and the CCP area of Processor RAM. (See "Block Mode Read/Write" in Section 1.)

2. Write the desired CCBs for initial communications data transfers.

3. For each CCP, issue an appropriate IO instruction to start the CCP, allowing it to load registers 6, 4, and 2 of the related adapter.

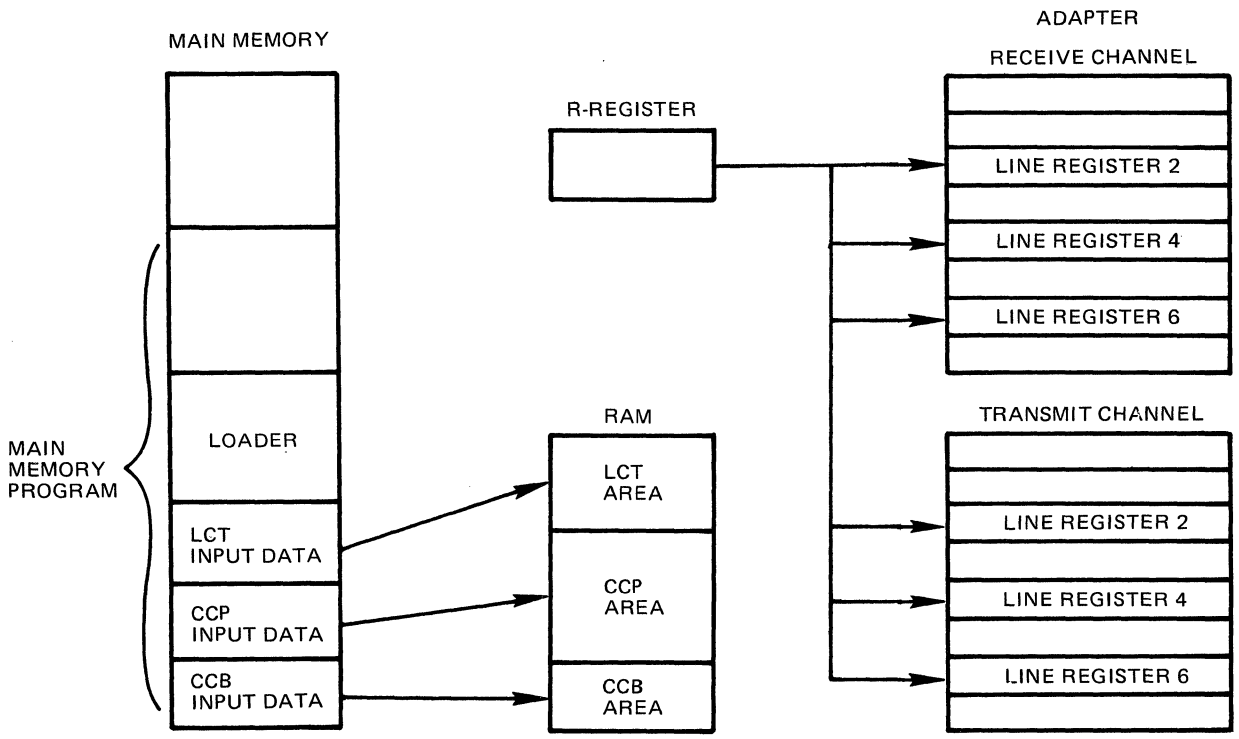


Figure 2-1. Setting Up the NMLCP

The sequence of events during a channel receive is depicted in Figure 2-2 and is listed as follows:

1. A data character received in a receive channel's line register 1 causes the Communications-Pac/Adapter to generate a CRI to the Processor.

2. The CCP is started. It loads the received data character into the Processor's R-register.

3. The CCP edits/modifies the data character in the R-register as required.

4. The CCP transfers the data character from the R-register to a CDB in main memory.

5. The CCP assumes a wait mode.

6. The Processor starts processing the next pending function.

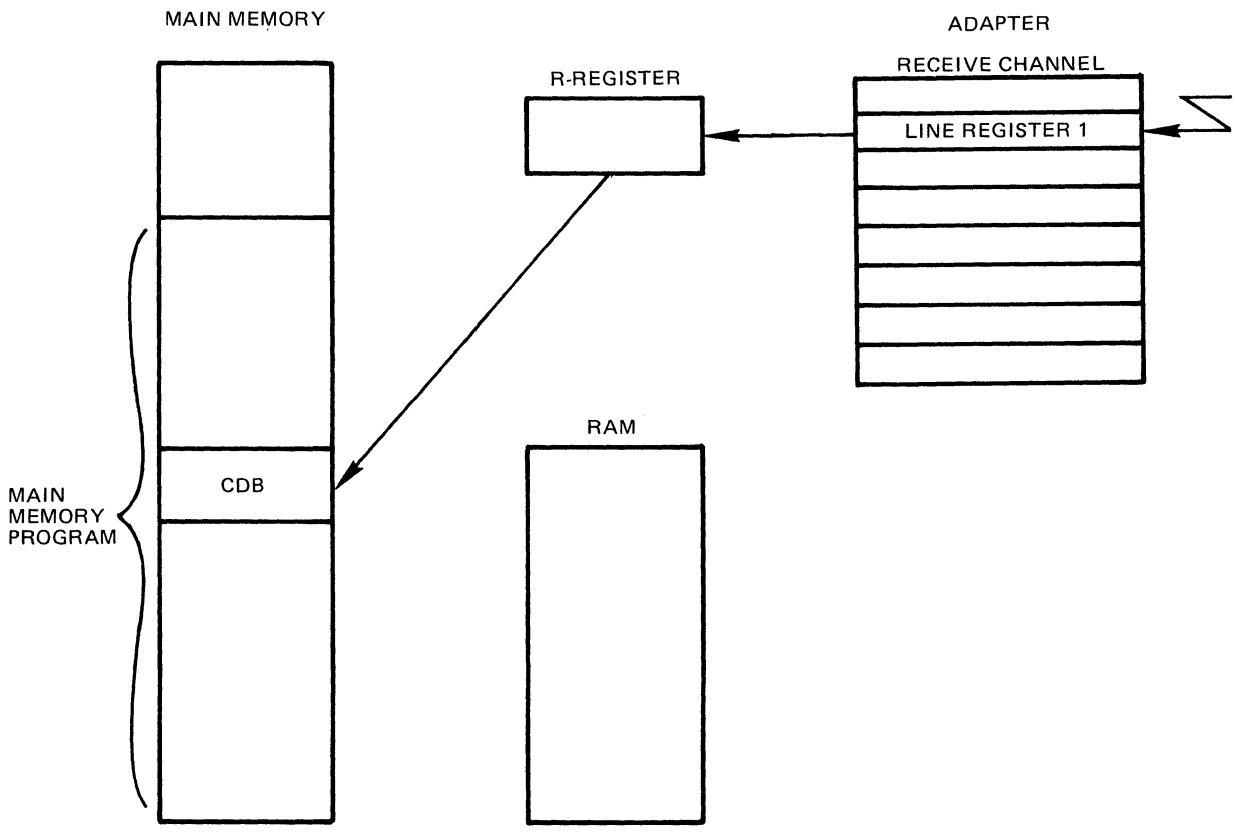


Figure 2-2. Receiving Data

The sequence of events during a transmission of data is shown in Figure 2-3 and is listed as follows:

1. The adapter generates a CRI, signifying it can accept a data character for transmission.

2. The CCP loads a data character from the main memory CDB into the Processor's R-register.

3. The CCP edits/modifies the data character in the R-register as required.

4. The CCP sends the data character to the transmit channel's line register 1 of the adapter. From there, the data character is automatically transmitted.

5. The CCP assumes a wait mode.

6. The Processor starts processing the next pending function.

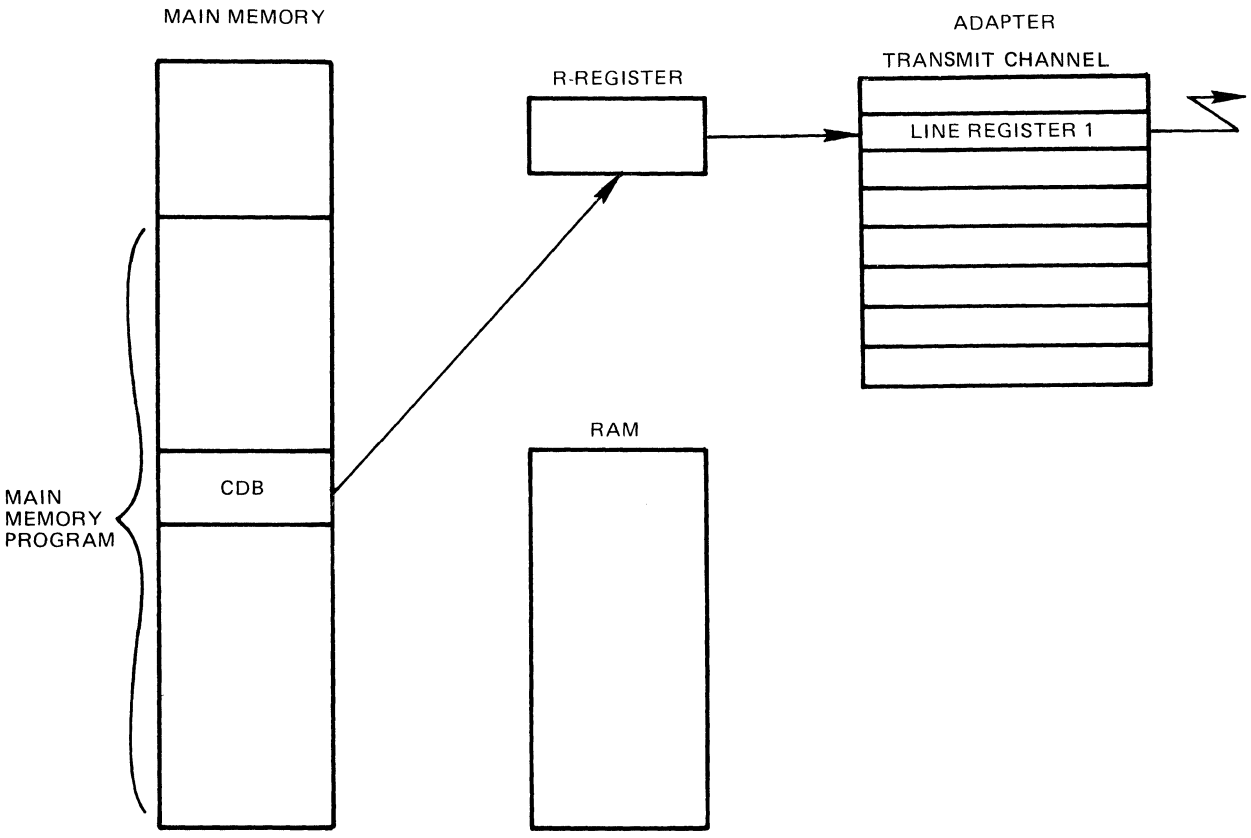


Figure 2-3. Transmitting Data

The following sequence of events provides a more detailed description of one way to perform Processor setup and subsequent data communications operations.

INITIALIZING THE PROCESSOR; WRITING THE LCT AND CCP AREA

NOTE

Prior to this step, you may wish to have a main memory program issue a separate IO (Input Device Identification Number) instruction for each channel to be used. For each channel specified, this action returns an identification number that indicates what type of adapter is in use.

Special care must be taken when creating the image that is to be transferred to the LCT area of the Processor RAM. Certain LCT bytes must be set up with appropriate values to control hardware/firmware operations. Other LCT bytes can be set up with application-specific values, as desired. Still other LCT bytes must contain zero when communications processing begins. Section 5 describes the program-visible LCT bytes and provides a detailed layout of each LCT byte, including information about the initial settings and subsequent modifiability of all bit positions.

NOTE

A Block Mode Write can be used, for instance, when you wish to initialize and set up only a subset of the Processor's channels, while previously initiated communications processing continues concurrently over other channels.

The RAM Data Transfer method can be used to initialize the entire Processor and to write a user-created block to the LCT area and the CCP area of Processor RAM. Details regarding the Transfer appear in Section 1 under "RAM Data Transfer."

WRITING THE CCBs

For each channel, one or more CCBs can now be written to define the starting location and length of one or more CDBs in the main memory program. Each CCB must be set up by following instructions from the main memory program:

1. IOLD (Output CCB Address and Range)
2. IO (Output CCB Control - format 1).

Thereafter, throughout execution of the application, the main memory program is responsible for supplying CCBs as needed for the CCB list of each channel being used.

LOADING ADAPTER LINE REGISTERS

The Line Registers (LRs) of each adapter are loaded next. This action is performed by the appropriate CCP, and the main memory program must issue an IO (Output Channel Control - start input/output) instruction to start the CCP. Startup procedures are unique to the individual line adapter. This loading is accomplished by a separate OUT (Output) instruction for each LR to be loaded.

LR6 (character configuration) is loaded from LCT byte 2 or from LCT byte 34. LR6 is shared by both channels of one line.

LR4 (line speed or synchronization/transmit-fill character) can be loaded from a byte in the programming work area of an LCT. LR4 may have a different function, depending upon the type of adapter used. For asynchronous line adapters, each register 4 must be loaded with a value indicating the speed at which each channel of the line will operate. For synchronous line adapters, a synchronization character must be loaded into LR4 for the receive channel. For the Processor, a transmit fill character must be loaded into LR4 for the transmit channel. LR2 (data set and adapter control) is loaded last from LCT byte 20. LR2 is shared by both channels of one line.

Once LR2 is loaded, the adapter will be enabled to generate CRIs if the transmit on bit of LR2 is set, according to the configuration of the adapter and the data communications equipment. The CCP should execute a WAIT instruction at this point. In the CCP, the WAIT instruction is followed by a data character processing loop, which usually terminates a branch back to the WAIT. The CCP startup coding preceding the WAIT is normally executed only the first time the CCP is started.

Other line registers or FLAP registers may be loaded as appropriate.

RECEIVING DATA

A CRI from the adapter to the Processor indicates that an input (receive) data character is available in a receive channel's data register of the adapter. The Processor performs a context restore for the channel (preparing the appropriate CCP for execution). The CCP is turned on at the instruction just after the previous WAIT instruction executed by this CCP; the CCP uses a RECV (Receive) instruction to load the Processor's R-register with the data character from the data register. The CCP edits and manipulates the data character in the R-register as required by the application. The CCP then transfers the data

character from the R-register to the CDB by means of an ST (Store) instruction. The CCP then branches back to the WAIT instruction.

This is the basic CCP receive processing loop for each data character of a communications message. The loop can also contain branch and/or TLU (Table Look-Up) instructions for other checks relative to the data character. A CCP subroutine could also be used.

TRANSMITTING DATA

The adapter issues a CRI to the Processor, indicating that it is ready to accept a data character for transmission. The CCP is turned on after the context restore. The CCP then either loads a data character into the Processor's R-register or uses the character reloaded into the R-register during the context restore.

Next, the CCP can edit and manipulate the data character as required before transferring it to the transmit channel's data register of the adapter by means of a SEND instruction. The data character is then automatically transmitted from the adapter.

If desired, after issuing the SEND instruction, the CCP can immediately issue a WAIT instruction. In this case, when the CCP is next activated, it will have to load the R-register with the data character to be transmitted next (editing and manipulating it as necessary) before issuing a SEND and a WAIT instruction. Alternatively, after issuing the SEND instruction, the CCP can load the R-register with the data character to be transmitted next (editing and manipulating it as necessary) before issuing the WAIT instruction. In this case, the data character will be reloaded into the R-register during the context restore that accompanies reactivation of the CCP, and the SEND can be done immediately.

END OF CDB PROCESSING

The relationship between physical CDBs and logical communications messages is completely under programmer control.

In Receive mode, when the CCP executes an ST (Store) instruction for the last character in a CDB, the range in the CCB decreases to zero and the Last Character (LC) indicator is set to 1. To check for the end of receive data before a CDB becomes full, the CCP can search for a specific control character in the input data stream. The CCP can use a TLU or a C (Compare) instruction to check for this condition. Whenever processing ends relative to a CDB, the CCP can obtain the next CDB by issuing a GNB (Get Next Block) instruction.

In Transmit mode, termination of CDB processing normally occurs when CCB range decreases to zero and the Last Character (LC) indicator is set. In some cases, earlier termination may be necessary because of some other condition discovered by the CCP. In any case, to continue transmission with another CDB and CCB, the CCP must issue a GNB instruction.

END OF LOGICAL MESSAGE PROCESSING

As mentioned above, the relationship between physical CDBs and logical communications messages is completely under programmer control.

If a logical communications message uses only one CDB, processing for that CDB is basically as described in the subparagraph above; however, instead of the CCP routinely proceeding from one CDB to another, CDB processing should continue as required by the application.

If logical communications messages comprise more than one CDB, individual messages may use either a variable or fixed number of CDBs. In any case, the last CDB in a message can be identified to the CCP if the main memory program has set the Last Block (LB) indicator in the CCB control byte. The LB indicator can be set by an IO (Output CCB Control) instruction from the main memory program. In Receive mode, the last CDB can be indicated by a control character in the incoming data stream. Alternatively, the CCB valid indicator can be tested by the CCP (refer to Appendix A).

END OF CHANNEL LINE USAGE

When processing relative to a channel or line is finished, you can indicate this fact to the adapter by resetting to 0 the receive on and/or the transmit on bit in LR2 of the adapter.

Subsequently, communications processing should be resumed over the channel or line, by reloading LR2 (as a minimum) with appropriate values. A more extensive restart would involve initializing the channel by means of an IO (Output Channel Control - channel initialize) instruction, performing one or more Block Mode Write operations to set up the LCT and CCP, and then reloading the appropriate adapter LRs from the new CCP.

The main memory program has the following responsibilities:

1. Control of the Processor by means of input/output instructions issued to it

2. Control of Communications Data Blocks (CDBs)

3. Control of Communications Control Blocks (CCBs)

4. Control of Channel Control Programs (CCPs)

5. Detection of errors and status changes related to data communications equipment and data terminal equipment.

(All terms related to CCBs are defined in Section 4.)

Main Memory Program Input/Output Instructions

The following is a list of I/O orders which are provided for control of the Processor. An Illegal Function Code (IFC) will set the IFC bit in Interrupt B status (FC=06), and a Stop I/O is performed on the channel.

Input Orders

1. IO (FC=0E) Input CCB Control
2. IO (FC=0C) Input Range
3. IO (FC=18) Input Status
4. IO (FC=10) Input Configuration A
5. IO (FC=1C) Input Line Status
6. IO (FC=26) Input Device ID
7. IO (FC=08) Input Extended ID
8. IO (FC=04) Input Firmware Revision
9. IO (FC=02) Input Interrupt Control A
10. IO (FC=06) Input Interrupt B Status
11. IO (FC=1E) Input LCT Byte
12. IO (FC=14) Input Load Monitor
13. IO (FC=1A) Input Next Status
14. IO (FC=12) Read and Clear LCT Byte
15. IO (FC=3E) Input from Analyzer

Output Orders

1. IO (FC=05) Output Channel Control
2. IO (FC-11) Output Configuration A
3. IO (FC=0F) Output CCB Control
4. IO (FC=07) Output Interrupt Control B
5. IO (FC=03) Output Interrupt Control A
6. IO (FC=01) Output NMLCP Control
7. IO (FC=15) Output RAM Control
8. IO (FC=13) OR to LCT Byte
9. IO (FC=0B) Output LCT Byte
10. IOLD (FC=09/0D) Load Data Area
11. IOLD (FC=29/2D) RAM Data Transfer
12. IO (FC=3D) Output to Analyzer
13. IO (FC=3F) Output to Analyzer

The conditions under which the above orders receive ACK, NAK, or WAIT response are given in a later subsection entitled, "WAIT and NAK Responses."

Table 2-1 provides a summary description of the input/output instructions available to the main memory program for controlling the Processor, and appear in alphabetical order. For detailed I/O descriptions see a later subsection entitled, "Detailed Description of Main Memory Input/Output Instructions."

Table 2-1. Summary of Main Memory Program Input/Output Instructions

Instruction	Function Code	Description
IO (Input CCB Control)	0E	Transfers the control byte of the current status CCB (addressed channel) to the main memory program (ML).
IO (Input CCB Range)	0C	Transfers the two range bytes of the status CCB to the main memory program (ML).
IO (Input CCB Status)	18	Transfers the two CCB status bytes to the main memory program (ML).
IO (Input Configuration A)	10	Transfers the contents of configuration A (addressed channel) to the main memory program (ML).
IO (Input Line Status)	1C	Transfers the contents of line adapter LR 5 for a specified Processor channel to the main memory program (ML).
IO (Input Device Identification Number)	26	Transfers the identification number that indicates the type of device associated with a specified Processor channel to the main memory program (ML). The Processor response is 2978 .
IO (Input Extended Device Identification Number)	08	Transfers the identification number indicating the type of adapter and FLAP associated with a specified Processor channel to the main memory program (ML).

Table 2-1 (cont). Summary of Main Memory Program Input/Output Instructions

Instruction	Function Code	Description
IO (Input Firmware Revision)	04	Transfers the firmware revision numbers of the Bus and CCP processors to the main memory program (ML).
IO (Input Interrupt Control A)	02	Transfers the contents of interrupt control A (designated channel) to the main memory program (ML).
IO (Input Interrupt B Status)	06	Transfers the contents of interrupt B status register (addressed channel) to the main memory program (ML), and then clears the channel.
IO (Input LCT byte)	1E	Transfers the LCT byte addressed by the contents of LCT byte 55 (specified processor channel) to the main memory program (ML). It must also be preloaded by the Output LCT Byte instruction.
IO (Input Load Monitor)	14	Transfers the contents of the processor load monitor to the main memory program (ML).
IO (Input Next CCB Status)	1A	Moves the CCB status pointer to the following CCB in the CCB list; transfers the two status bytes of that CCB (which is now the status CCB) to the main memory program (ML).
IO (Read and Clear LCT Byte)	12	Uninterruptable operation to read and clear the contents of the LCT location, which is indicated by the contents of LCT 55 (addressed channel).

Table 2-1 (cont). Summary of Main Memory Program Input/Output Instructions

Instruction	Function Code	Description
IO (Input from Analyzer)	3E	Used to detect various activities within the Processor.
IO (Output Channel Control)	05	Causes the Processor to perform one of the following actions: 1. Channel initialization 2. Start or stop input/output 3. Start "Block mode read" or "Block mode write" 4. CCB list reset.
IO (Output Configuration A)	11	Transfers a 16-bit word to the addressed channel, defining the following operating modes: 1. Compatibility 2. Extended 3. Restart Privilaged 4. Extended LCT Write Permit
IO (Output CCB Control)	0F	Format 1: Causes load pointer to advance. Transfers control information from the main memmory program (ML) to the control byte of the current CCB. Moves the load CCB pointer to the following CCB in the CCB list. Format 2: Sets up CCB Control byte. Transfers the starting Processor RAM address for Block mode input/output, to the current CCB. Moves the load CCB pointer to the following CCB in the CCB list.

Table 2-1 (cont). Summary of Main Memory Program Input/Output Instructions

Instruction	Function Code	Description
		Format 3: Clears the two status bytes of the new CCB.
IO (Output Interrupt Interrupt Control B	07	Transfers a 16-bit control word to the designated channel.
IO (Output Interrupt Control	03	Transfers a 16-bit Interrupt control word to the designated processor channel.
IO (Output NMLCP Control)	01	Causes the processor to perform one of the following actions: 1. NMLCP hard initialize 2. Set RAM transfer bits 3. Set priority scheme of controller.
IO (Output RAM Control)	15	Transfers a 16-bit control word to the Processor which defines the starting address in Processor RAM. This also applies to the next RAM data transfer.
IO (OR to LCT byte)	13	Uninterruptable operation which logically ORs the LCT data byte given on the Bus with the current contents of the LCT location (specified addressed channel).
IO (Output LCT Byte)	0B	Transfers one byte from the main memory program to a specified byte in a specified LCT.
IOLD (Output CCB Address and Range)	09/0D	Transfers the starting address and range of a CDB in main memory to the load CCB.

Table 2-1 (cont). Summary of Main Memory Program Input/Output Instructions

Instruction	Function Code	Description
IOLD (RAM Data Transfer)	29/2D	Delivers address and range to the Processor which is used for loading/storing RAM area information to/from the main memory.
IO (Output to Analyzer)	3D	Used to detect various activities within the Processor.
IO (Output to Analyzer)	3F	Used to detect various activities within the Processor.

Control of Communications Data Blocks

The main memory program has total responsibility for the control of CDBs. This responsibility includes (1) supplying new CDBs, as needed, for use in communications data transfers and (2) servicing CDBs after they have been used for communications data transfers. Since communications data transfers to and from main memory are controlled by a fixed number of CCBs, the main memory program must know the completion status of CCBs in order to coordinate its control of them. If desired, the main memory program can arrange for the Processor to generate an interrupt as soon as a CCB is marked completed; this technique is described in Section 6. Alternatively, the main memory program can perform its own checking of CCB completion status; the format of the two CCB status bytes is described in Section 4.

Control of Communications Control Blocks

The main memory program completely controls additions to and deletions from the list of CCBs available to each channel of the Processor. This control is achieved by use of the Processor-related input/output instructions described in this section.

Control of Channel Control Programs

The main memory program is responsible for loading and starting CCPs. The main memory program can load CCPs by use of the RAM Data Transfer method or it can use one or more Block Mode Writes for this purpose; both techniques are described in Section 1. The main memory program starts initial execution of each CCP by issuing an IO (Output Channel Control) instruction.

Detection of Errors and Status Changes Related to Data Communications Equipment and Data Terminal Equipment

The main memory program can detect and respond to errors and status changes related to Data Communications Equipment (DCE) and Data Terminal Equipment (DTE). Relevant information is available in the two status bytes of a CCB and in LR5 of a line adapter. All of this information is also available to CCPs, allowing them to detect and respond to errors and status changes, if desired.

Note that if a CCP is designed to perform extensive responses to errors and status changes, its execution time will be increased accordingly. (This increase in execution time may be perfectly acceptable in some applications - e.g., those using low-speed communications lines.)

One approach to detecting and responding to errors and status changes related to data communications equipment and data terminal equipment would be to have the main memory program and the CCP share this responsibility. The CCP could be designed to react to errors and status changes as individual characters in a data stream are processed; the main memory program could be designed to react to these errors and status changes as they affect an entire CDB.

Another approach would be to use the foreground-background functionality of the Processor (see subsection entitled "Data Service and Event Service Levels" in Section 1). This is accomplished by using the foreground level for character processing and the background level for CDB (message) level error control and recovery.

At any rate, the designer of a communications application must decide how much handling of errors and status changes will be performed by the main memory program and how much (if any) will be performed by CCPs. Section 6 and Appendix A provide more information on detecting errors and status changes related to DCE and DTE.

Detailed Description of Main Memory Program Input/Output Instructions

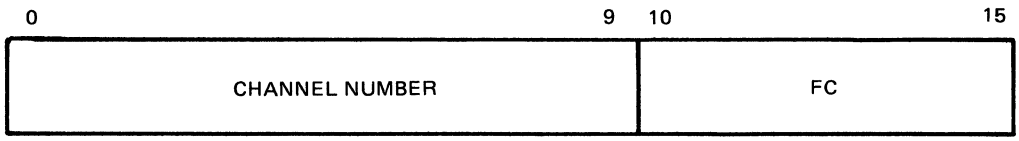
The following subsections describe the input/output instructions usable in a main memory operation interfacing with a Processor. See the appropriate Assembly Language manual for details about coding these and other instructions used in the main memory program.

All but two of these input/output instructions are IO instructions. (The two others are IOLD instructions.) The format of these IO instructions is shown below.

```
IO ML,CF
```

ML is an address expression identifying a memory location or register to which or from which information is to be transferred.

CF is an address expression identifying a memory location or register that contains a channel number and a function code. The format of this information is shown below:



The channel number comprises two parts: bits 0 through 5 contain the six bits of the fixed (switch-selectable) channel number on the Processor on the Megabus network; bits 6 through 9 identify one of the communications channels of the Processor. (The designated communications channel must be serviced by an adapter.)

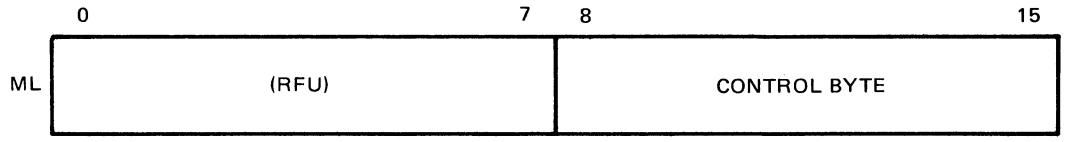
FC indicates the function code, which specifies the exact input/output operation to be performed. An odd function code signifies an output instruction; the contents of ML will be transferred to the Processor. An even function code signifies an input instruction; data will be transferred from the Processor to ML.

The format of the IOLD instruction is shown in a later subsection entitled "IOLD (Load Data Area) Instruction."

Input/output instructions to the Processor are generally executed immediately without causing a NAK. However, you may wish to code a BIOF (Branch if Input/Output Indicator False) instruction after any input/output instruction which might be NAKed (see Appendix A).

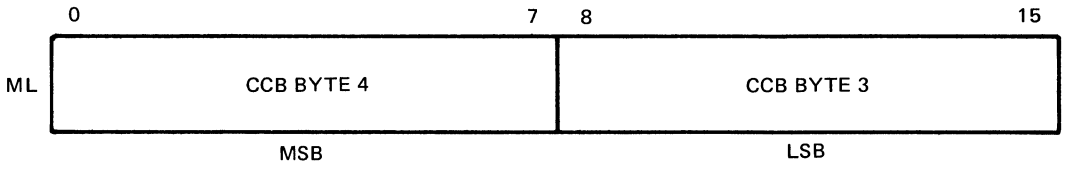
IO (INPUT CCB CONTROL) INSTRUCTION

This instruction (function code 0E) transfers, to ML, the CCB control byte of the current CCB. If the GIVE instruction was executed for this CCB, bits 3-7 will have been reloaded from the LCT (see the subsection in Section 4 entitled "CCB Control Field"). The Input Next Status order (FC=1A) is used to advance to the next CCB (see Section 4). The format of the word transfered is shown below.



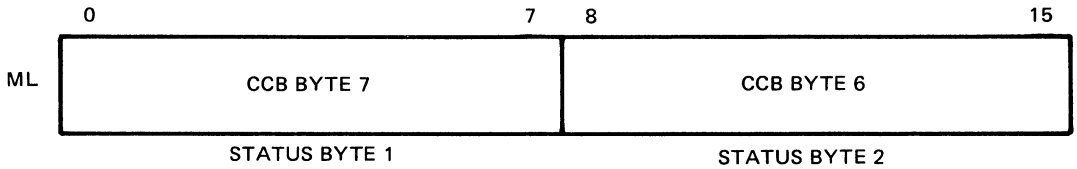
IO (INPUT CCB RANGE) INSTRUCTION

This instruction (function code 0C) transfers, to ML, the two range bytes in the CCB at the top of the CCB list for the Processor channel specified in CF. As shown below, the range byte from CCB byte 4 is transferred to the left byte of ML and the range byte from CCB byte 3 is transferred to the right byte of ML.

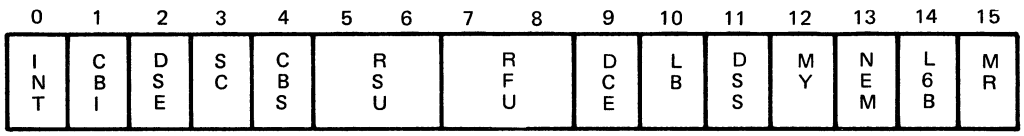


IO (INPUT CCB STATUS) INSTRUCTION

This instruction (function code 18) transfers, to ML, the two status bytes in the CCB at the top of the CCB list for the Processor channel specified in CF. As shown below, the status byte from CCB byte 7 is transferred to the left byte of ML and the status byte from CCB byte 6 is transferred to the right byte of ML.



The CCB status bytes will contain invalid information if the CCB has not yet been marked as completed. The contents of this word are shown below:



INT = CCP executed INTR instruction
CBI = CCB interrupt flag was set
DSE = Data Service Error (Underrun/Overrun)
SC = Status Complete (Compatibility mode GNB; Extended mode GIVE)
CBS = CCB Service Error (Next Block fault)
DCE = Data Check Error (Parity/CRC)
LB = Last Block flag/residue range
DSS = Data Set Scan (Appendix A)
MY = Memory Yellow (CCB execution)
NEM = Nonexistent Memory (CCB execution)
L6B = Level 6 Bus parity (CCB execution)
MR = Memory Red (CCB execution)

CAUTION

Input CCP Status does not advance the CCB status pointer. When a status incomplete is detected, after an Input Next CCB Status, then an Input CCB Status would be issued to detect the status complete bit. This technique is used for non-CPU Interrupt mode where the main memory program is waiting for CCP completion.

IO (INPUT CONFIGURATION A) INSTRUCTION

This instruction (function code 10) transfers, to ML, the contents of Configuration A register. The bit format of this register is shown below:

Bit 0 - Extended Mode. - When this bit is one, the channel operates in Extended mode. When this bit is Zero, the channel operates in Compatibility mode.

Bits 1 and 2 - RFU

Bit 3 - Restart Privileged - When this bit is One, a Master Clear instruction is permited to be executed by the channel.

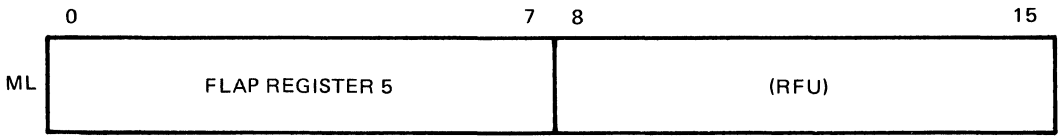
Bit 4 - Extended LCT Write Permit

0 = Off
1 = On

Bits 5-15 - RFU

IO (INPUT LINE STATUS) INSTRUCTION

This instruction (function code 1C) causes a direct transfer, to ML, of the contents of the FLAP FR5 for the Processor channel specified in CF. The format of the word transferred to ML is shown below:



The contents of FR5 vary according to the type of FLAP; refer to the appropriate appendix.

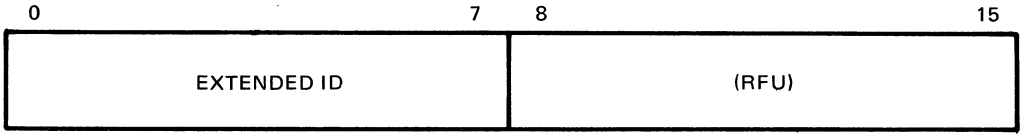
IO (INPUT DEVICE IDENTIFICATION NUMBER) INSTRUCTION

This instruction (function code 26) transfers, to ML, the Processor device identification number. The device identification number for the Processor is (2978) .

IO (INPUT EXTENDED IDENTIFICATION NUMBER) INSTRUCTION

This instruction (function code 08) transfers, to ML, the identification number that indicates the type of adapter and FLAP associated with a specified Processor channel. When this command (Input Extended Identification Number) is issued, the Processor will return the contents of LR0 of the adapter, and FR0 of the FLAP as shown in a later subsection entitled "Device Identification Number."

The format of this instruction is shown below:



The extended identification codes are listed below:

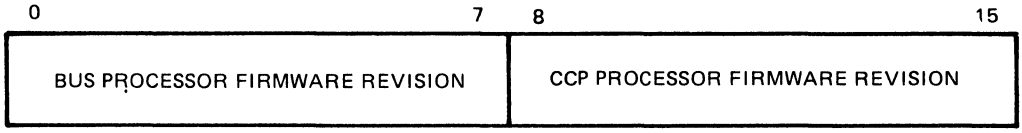
ID CODES

	FC26	FC08		
NMLCP	2978			
		0100		HDLC/SYNC ADAPTER
ADAPTER		78XX	SYNC	
		79XX	SYNC	
		7AXX	ASYNC	SYNC/ASYNC ADAPTER
		7BXX	ISOCH	
		48XX	SYNC	
		49XX	HDLC	HDLC/SYNC ADAPTER
		3A00	422DC	ASYNC/DC ADAPTER

00	No FLAP
01	RS232
02	RFU
03	RS422
04	Current Loop
05	X.21
06	Auto Call
07	301/303
08	V.35
09	RFU
0A	Mil-188-114
0B	RFU

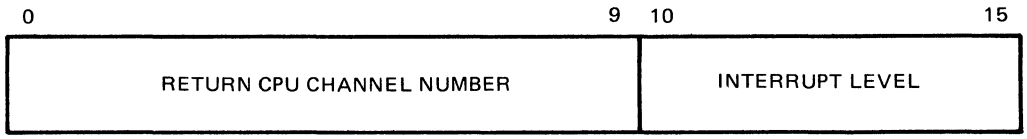
IO (INPUT FIRMWARE REVISION) INSTRUCTION

This instruction (function code 04) transfers, to ML, the firmware revision numbers for the Bus and CCP processors. The format of the information transferred to ML is shown below:



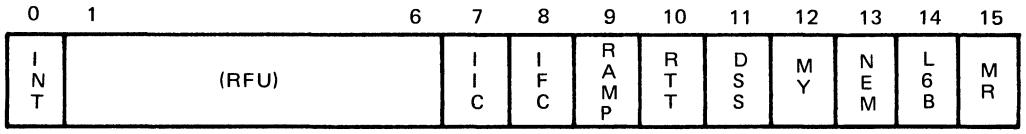
IO (INPUT INTERRUPT CONTROL A) INSTRUCTION

This instruction (function code 02) transfers, to ML, the Interrupt Control A register contents for the addressed channel. The format of the register contents is shown below:



IO (INPUT INTERRUPT STATUS B) INSTRUCTION

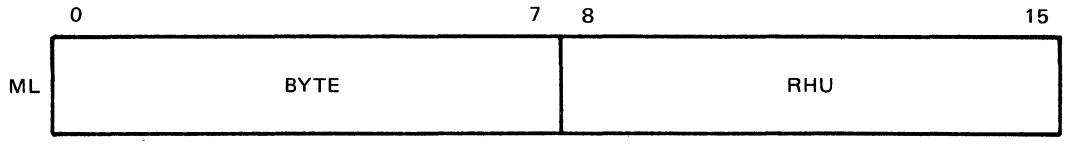
This instruction (function code 06) transfers, to ML, the status associated with an Interrupt B occurrence on the addressed channel and then the status is cleared. The contents of the word is shown below:



INT = CCP Executed INTR Instruction (Section 3)
IIC = Illegal Instruction Code
IFC = Illegal Function Code
RAMP = RAM Parity error
RTT = RAM data Transfer Termination
DSS = Data Set Scan (Appendix A)
MY = Memory Yellow (RAM Data Transfer)
NEM = Nonexistent Memory (RAM Data Transfer)
L6B = Level 6 Bus parity (RAM Data Transfer)
MR = Memory Red (RAM Data Transfer)

IO (INPUT LCT BYTE) INSTRUCTION

This instruction (function code 1E) transfers, to ML, the LCT byte addressed by the contents of LCT byte 55. The contents of LCT byte 55 is loaded by the Output LCT Byte instruction (function code: 0B) or by a Block Mode Write instruction. Refer to Section 5 for an additional definition of LCT byte 55. The format of the instruction is shown below:

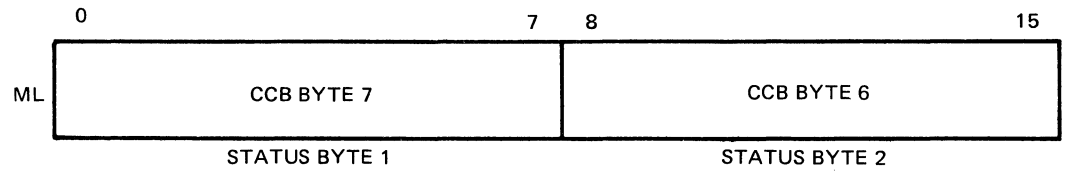


IO (INPUT LOAD MONITOR) INSTRUCTION

This instruction (function code 14) transfers, to ML, the current contents of the processor load monitor. Refer to the subsection entitled "Processor Load Monitor" in Section 1.

IO (INPUT NEXT CCB STATUS) INSTRUCTION

This instruction (function code 1A) causes the CCB status pointer to be moved to the following CCB in the CCB list for the Processor channel specified in the CF. The two status bytes of that CCB (which is now at the top of the CCB list) are then transfered to ML. As shown below, the status byte from CCB byte 7 is transferred to the left byte of ML and the status byte from CCB byte 6 is transfered to the right byte for ML.



The CCB status bytes should be considered to contain invalid information if the CCB has not yet been marked as completed (i.e., Status complete bit is One). If the Status complete bit is Zero, then only the Input CCB Status instruction should be used if waiting for completion. If an IO (Input Next CCB Status) information is issued, then the CCB pointer will move to the next CCB and the previous status will be lost. If the CCB does not complete, the LCT status bytes 16/48 and 17/49 should be read by using the Input LCT Byte instruction. A decision based on the LCT status can then be made.

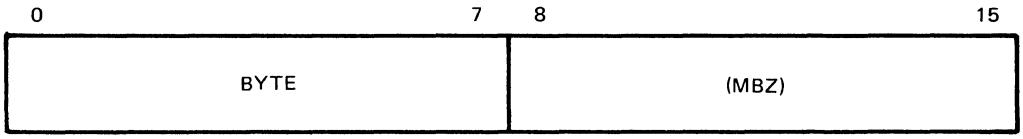
In the CCB list, the CCB that was formerly at the top (before this instruction was executed) is now available for reuse.

Note that the IO (Input Next CCB Status) instruction must be used the first time status is obtained from a CCB list. The use of the instruction moves the CCB status pointer to CCB 1, which is the first CCB used after the CCB list is initialized (as described in Section 3).

Under certain circumstances, an attempt to execute an IO (Input Next CCB Status) instruction will cause the Processor to issue a NAK; see "Conditions Under which Processor Will Issue a NAK" in Appendix A.

IO (READ AND CLEAR LCT BYTE) INSTRUCTION

This instruction (function code 12) causes the Processor, in a single uninterruptable operation, to read and clear the byte contents of the LCT location. The LCT location of the addressed channel is indicated by the current contents of LCT 55. The instruction is in the following format:



This instruction is used so the CPU can retrieve flag bits, on a race-free basis, from an LCT location used for communications from a CCP.

IO (INPUT FROM ANALYZER) INSTRUCTION

This instruction (function code 3E) is used to record various activities within the Processor.

IO (OUTPUT CHANNEL CONTROL) INSTRUCTION

This instruction (function code 05) transfers a control word from ML to the Processor. Each execution of this instruction affects only one channel. Only one bit in the control word can be set to One.

The operations achieved by the bits of the control word (if set to One) are summarized below:

Bit 0 - Channel Initialize
Bit 1 - Start Input/Output
Bit 2 - Stop Input/Output
Bit 3 - Reserved
Bit 4 - Start Block Mode Read
Bit 5 - Start Block Mode Write
Bit 6 - Reserved
Bit 7 - CCB List Reset
Bits 8 through 15 - Reserved

The following actions are performed by the Processor when it receives a control word with a bit set to One.

Bit 0 - Channel Initialize

1. Resets to Zero LR2 of the adapter.

2. Halts execution of CCP.

3. Stops all activity for this channel.

4. Resets to Zero the entire LCT area for the line associated with this channel.

5. Resets CCB list (see bit 7, below, for a description of this operation).

Bit 1 - Start Input/Output

1. Starts input/output. Start I/O causes the CCP to begin execution. The starting address of the CCP is contained in LCT bytes 6/7 or 70/71 for receive, and LCT bytes 38/39 or 102/103 for transmit (see "Data Service and Event Service Levels" in Section 1). The main memory program must have previously set these locations to the CCP starting address at least once during the program.

2. Starts execution of the CCP. If the CCP is already running and a Start I/O is issued, a loss of data characters may result.

Bit 2 - Stop Input/Output

1. Resets to Zero Receive On (bit 6) or Transmit On (bit 7) in LR2 of the adapter. This action prevents subsequent data-generated CRIs.

2. Resets to Zero Receive On (bit 6) or Transmit On (bit 7) in the LCT byte 20 copy of LR2.

3. Halts execution of CCP.

4. Resets to Zero the LCT status bytes (LCT bytes 16 and 17 for receive channel, LCT bytes 48 and 49 for transmit channel) - after they have been transferred to the appropriate CCB.

5. Terminates active CCB (with meaningful status information) and stops CCB list processing.

6. Inhibits interrupts to the main memory program (but does not change channel's interrupt level).

7. Stops all activity for this channel.

Bit 3 - Reserved

Bit 4 - Start Block Mode Read

1. Used to read any portion of the Processor RAM. A receive (even-numbered) channel must be designated in CF when this operation is performed. Block Mode Reads should not be used on channels operating in Extended mode.

2. Uses the CCB next in line to be active to read a block of consecutive RAM locations into the main memory program. (Details appear in Appendix A.) When the read is completed (CCB range equals Zero), the CCB will be marked as completed (with meaningful status information). The main memory program will be interrupted at the interrupt level of the receive channel used for the Block Mode Read (unless this channel's interrupt level is Zero).

Bit 5 - Start Block Mode Write

1. Used to write a block of consecutive RAM locations. It is a convenient way to write the LCT area and the CCP area of the addressed Processor channel. Block Mode Writes should not be used on channels operating in Extended mode.

CAUTION

A Block Mode Write should not be used to write into LCT areas other than the currently addressed channel when any other channels may be active. In the addressed channel, care should be taken not to overwrite reserved firmware areas (see Section 5). In addition, care should be taken when other channels are active not to write into their active LCT, CCB area.

2. A transmit (odd-numbered) channel must be designated in CF when this operation is performed. Note that the LCT bytes for the Processor channel used for a Block Mode Write cannot themselves be written into at this time because these bytes are used by firmware during the course of this operation. Also, any firmware-reserved LCT bytes written into during a Block Mode Write must be written with Zeros.

3. Uses the CCB next in line to be active to write a block of consecutive RAM locations from the main memory program. (Details appear in Section 7.) When the write is completed (CCB range equals Zero), the CCB will be marked as completed (with meaningful status information). The main memory program will be interrupted at the interrupt level of the transmit channel used for the Block Mode Write (unless this channel's interrupt level is Zero).

Bit 6 - Reserved

Bit 7 - CCB List Reset

1. Resets the channel's CCB pointers to their initialized state (i.e., the CCB status pointer is set to point to CCB Zero of the CCB list and the load CCB pointer and active CCB pointer are set to point to CCB One of the CCB list).

2. Resets the control byte of each of the eight CCBs of the channel. This action resets the valid bits.

IO (OUTPUT CONFIGURATION A) INSTRUCTION

This instruction (function code 11) transfers a 16-bit word to the addressed channel to define the operating mode of the line. Initialization (see subsection in Section 3 entitled "Op Code Map") sets these mode bits to an OFF condition.

Bit 0 - Extended Mode. - When this bit is One, the channel operates in Extended mode. When this bit is Zero, the channel operates in Compatibility mode.

Bits 1 and 2 - RFU

Bit 3 - Restart Privileged - When this bit is One, a Master Clear instruction is permitted to be executed by the channel.

Bit 4 - Extended LCT Write Permit

0 = Off
1 = On

Bits 5-15 - RFU

IO (OUTPUT CCB CONTROL) INSTRUCTION

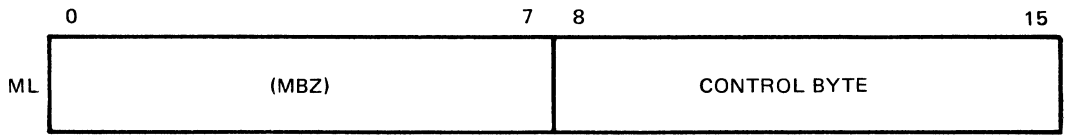
This instruction (function code 0F) is used for either of two purposes:

1. It transfers, from the right byte of ML, a control byte to byte 5 of a CCB, resetting bytes 6 and 7 (the status bytes) of the CCB to Zero.

2. It transfers, from ML, a RAM address (where a Block Mode Read or a Block Mode Write will begin) to bytes 5 and 6 of a CCB, resetting byte 7 to Zero.

In both cases, execution of this instruction completes CCB setup initiated by an IOLD (Output CCB Address and Range) instruction and moves the load CCB pointer to the following CCB in the CCB list.

If a control byte is to be transferred to byte 5 of a CCB, ML must be formatted as shown below.



The bits in the control byte have the following significance:

Bit 0 - Interrupt Control

0 - No action.

1 - Interrupt the main memory program when this CCB is marked as completed. (The interrupt will occur at the interrupt level assigned to the related channel.)

Bit 1 - Valid CCB

0 - Firmware sets this bit to Zero when this CCB has been marked as completed and is therefore no longer valid (i.e., usable as an active CCB).

1 - This CCB is valid (i.e., usable as an active CCB). This bit must be set to One to complete setup of the CCB.

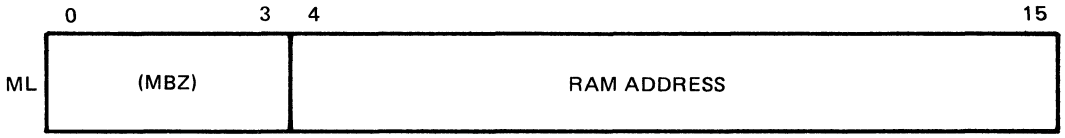
Bit 2 - last CDB

0 - No action.

1 - This CCB pertains to the last CDB in a message. If this bit is set to One, it serves as a flag that can be used by the CCP for special processing of the last CDB in a message. The Processor's LB (Last Block) indicator will be set to One when this CCB is active.

Bits 3 through 7 - RSU

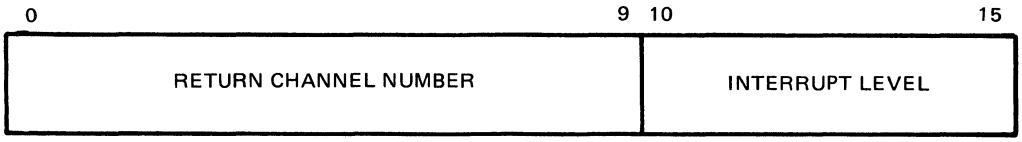
If a RAM address is to be transferred to bytes 5 and 6 of a CCB, ML must be formatted as shown below.



The right byte of this word is transferred to CCB byte 5; the left byte is transferred to CCB byte 6. The 12-bit RAM address indicates the RAM byte at which the Block Mode Read or Block Mode Write will begin.

IO (OUTPUT INTERRUPT CONTROL B) INSTRUCTION

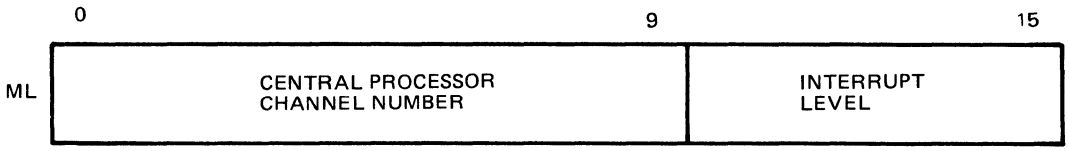
This instruction (function code 07) transfers a 16-bit interrupt control word to the designated data channel in the format shown below:



Both of these fields are stored in a firmware register for the appropriate data channel for reference when generating an interrupt for one or more of the events reported in Interrupt B status. When this interrupt level field is all zeros, the contents of Interrupt Control A register are used instead, and Data Set Scan and INTR instruction occurrences are reported through CCB status.

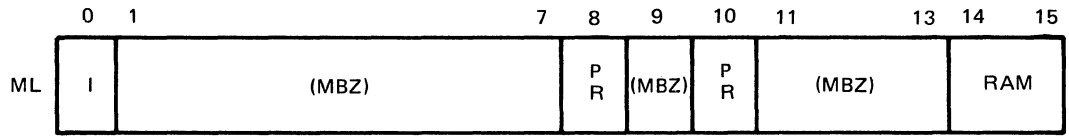
IO (OUTPUT INTERRUPT CONTROL A) INSTRUCTION

This instruction (function code 03) transfers, from ML, a return channel number (the central processor's channel number) and an interrupt level to be used during interrupts from the Processor channel specified in CF. For a receive channel, the left byte of ML is transferred to LCT byte 12 and the right byte is transferred to LCT byte 13. For a transmit channel, the left byte of ML is transferred to LCT byte 44 and the right byte is transferred to LCT byte 45. The format of ML is shown below:



IO (OUTPUT NMLCP CONTROL) INSTRUCTION

This instruction (function code 01) transfers, from ML, a 16-bit control word to the Processor. All Processor channels are affected by this control word. Any channel can be specified in CF, provided that channel is serviced by an adapter. The format of ML is shown below:



Bit 0	If set to One, NMLCP Hard Initialize
Bits 1-7, 9, 11-13	MBZ (Must Be Zero)
Bit 8	Priority Scheme (0=fixed, 1=round robin) is discussed in Section 1
Bit 10	If set to One, set Priority Scheme as specified in bit 8.
Bits 14, 15	RAM Transfer Map (see subsection entitled "RAM Data Transfer" in Section 1 00 = Null 01 = Local Store 10 = LCT RAM 11 = CCB and Bus Processor RAM

If I is set to One, Processor initialization will be performed; otherwise no action is taken. Initialization comprises the following actions:

1. The Processor executes its basic test.

2. Each LR2 of each adapter is reset to Zero; CRIs are thus inhibited.

3. All of RAM is reset to Zero.

4. LCT byte 1 of channel 0 contains the hexadecimal number of the firmware revision.

5. All channels are initialized. (This operation is described under the IO (Output Channel Control) Instruction, earlier in this section.)

6. The Processor is placed in a quiescent state; no interrupts or data transfers can occur.

7. Clears the adapters.

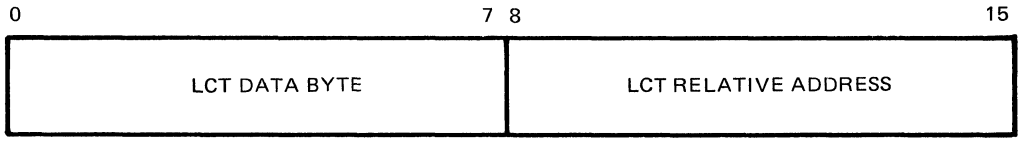
Hard initialize causes the actions described below. It requires hundreds of milliseconds to accomplish and is intended to be used only at startup or by T&V software.

IO (OUTPUT RAM CONTROL) INSTRUCTION

This instruction (function code 15) transfers a 16-bit control word to the Processor, which defines the starting address in a Processor RAM. This is to be applied on the next RAM Data Transfer operation (see Section 1).

IO (OR TO LCT BYTE) INSTRUCTION

This instruction (function code 13) causes the Processor, in a single uninterruptible operation, to logically OR the LCT data byte given on the bus into the current contents of the specified LCT location of the addressed channel. The format on the bus is as follows:



LCT Data Byte is an 8-bit byte to be logically ORed into the specified LCT location

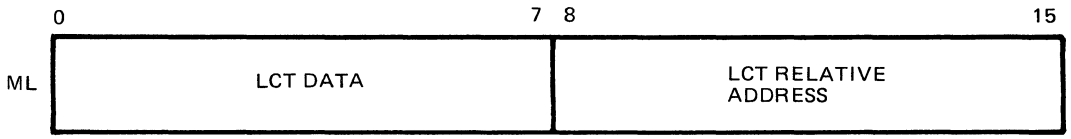
LCT Relative Address specifies which LCT location of the addressed channel is to be operated upon.

This order is used so the CPU may set flag bits, on a race-free basis, into an LCT location for communication with a CCP; the CCP may access these flag bits via LBFL instructions.

IO (OUTPUT LCT BYTE) INSTRUCTION

This instruction (function code 0B) transfers, from ML, a byte of information to a specific LCT byte address. Bits 0 through 8 of the Processor channel number specified in CF establish the LCT to which the information is transferred; bit 9 of CF is not meaningful in this case because one LCT applies to both channels of a line. (The base from which LCT address is an offset is byte 0 of the LCT that applies to the line indicated by bits 0 through 8 of CF.)

The format of ML is shown below:



LCT Data Byte indicates an 8-bit value to be transferred to the LCT address indicated in bits 8 through 15.

LCT Relative Address is relative to byte 0 of the LCT for the line indicated by bits 0 through 8 of CF. The eight bits of the LCT address permit any of the 256 LCT bytes to be designated.

IOLD (LOAD DATA AREA) INSTRUCTION

This instruction (function code 09/0D) transfers, from the address and range (see format below) the starting address and range of a CCB. The starting address is the starting byte address and the range in bytes. The transfer is made to the load CCB in the CCB list for the Processor channel specified in CF.

This is an IOLD instruction, the format of which is shown below.

IOLD address,CF,range

Address is an address expression identifying a memory location or register (the latter for use of the indirect addressing technique) that indicates the starting byte address of the CCB in main memory.

CF is an address expression identifying a memory location or register that contains a channel number and a function code. The format of CF is the same as that described under CF for an IO instruction, earlier in this section.

Range is an address expression identifying a memory location or register that indicates the number of bytes in the CDB. The range must be an integer from 1 to 65535.

The starting address of the CDB is stored in bytes 0, 1, and 2 of the load CCB. The LSBs of the address are stored in byte 0. The range for the CDB is stored in bytes 3 and 4 of the load CCB. The LSBs of the range are stored in byte 3.

Under certain circumstances, an attempt to execute an IOLD (Output CCB address and Range) instruction will cause the Processor to issue a NAK; see "Conditions Under Which the Processor will Issue a NAK" in Appendix A.

IOLD (RAM DATA TRANSFER) INSTRUCTION

This instruction (function code 29/2D) delivers address and range information to the Processor that is used for loading/storing of RAM area information from/to main memory (see "RAM Data Transfer" in Section 1).

If the order is addressed to an even-numbered channel, Executive channel firmware will store RAM area information into memory.

If the order is addressed to an odd-numbered channel, Executive channel firmware will load RAM area information from memory. The data address and range must both be wordbound.

If an error occurs during a RAM Data Transfer operation, the operation is terminated and the type error is set into Interrupt B Status (FC=06).

IO (OUTPUT TO ANALYZER) INSTRUCTION

This instruction (function code 3D) is used to control various activities within the Processor.

IO (OUTPUT TO ANALYZER) INSTRUCTION

This instruction (function code 3F) is used to control various activities within the Processor.

WAIT and NAK Responses

The Processor will issue a NAK response to an I/O order (except a Hard Initialize) in the following situations:

1. An initialize is in progress (but not yet complete)

2. In response to a Data Area IOLD (FC=09/0D) or Input Next Status IO (FC=1A) if the pointers cannot advance without causing an error (Section 3).

Normally the Processor will respond to a legal I/O order with an ACK response. If an order arrives before the Processor has completed the preceding one (except for initialize as noted above), a WAIT response will be given.

Note that while an Initialize is in progress, all orders are NAKed.

CHANNEL NUMBERS

The Processor uses a set of 32 channel numbers. The channel group is switch settable on the Processor and constitute the high-order five bits of the Processor channel number. The low-order five bits of the channel number are usually used to specify one of the 32 data channels. Channel number assignments are given in Table 2-2.

The Executive channel has no channel number as such; orders that are addressed to it are defined by function code.

Table 2-2. Channel Number Assignments (Decimal)

Channel Number	Line Number	Transmit/Receive
0	0	R
1	0	T
2	1	R
3	1	T
4	2	R
5	2	T
.	.	.
.	.	.
.	.	.
.	.	.
30	15	R
31	15	T

DEVICE IDENTIFICATION NUMBER

The Basic Device Identification Number (see IO Function Code 26) of the Processor is $(2978)_{16}$.

The Extended Device Identification Number (see IO function code 08) of the Processor is (N n n n) where N N where N N is provided by the attached adapter and N N is provided by the FLAP attached to the adapter. If there is no adapter on the channel, N N $=(00)_{16}$. If the channel can only be used for parallel output to a FLAP (e.g., an ACU FLAP), N N $=(01)_{16}$. If the adapter firmware is inoperable, the highest order bit of N will be a One.

INTERRUPTS TO THE CENTRAL SYSTEM

The Processor will generate interrupts to the Central Processor in accordance with the mode (Compatability/Extended) of the Processor and subject to the interrupt level specifications maintained in the interrupt level registers of the channel (see IO function code 03 and 07).

When a condition requiring an interrupt occurs, the Processor will address the CPU with an interrupt format bus transfer. In accordance with the operation of the Level 6 interrupt structure, the interrupt may be accepted (ACK) or rejected (NAK) by the CPU. If the interrupt is accepted, the function is complete; if rejected, the Processor will increment its deferred interrupt queue and will retry the interrupt when the CPU issues a RINT (retry interrupts) signal on the Megabus network. The handling of the deferred interrupt queue is entirely a background firmware function of the Processor. The deferred interrupt queue is decremented when the interrupt is accepted.

The conditions that may cause an interrupt to be generated by the Processor are:

1. Block Read/Write completion

2. A data set status change is detected by firmware (see "Executive Channel" in Section 1) and the channel command byte (LCT8/40) specifies interrupt as the action to be taken.

3. Execution of an INTR instruction by a CCP.

4. In Compatibility mode, completion of a CCB occurs (GNB) and the Interrupt bit of the CCB control byte (see "Setting Up a CCB" in Section 4) is set.

5. In Extended mode, execution of a GIVE on a CCB in which the interrupt was set.

6. Via Interrupt B, completion of a RAM data transfer or occurrence of an illegal I/O order or illegal CCP instruction, or a RAM parity error in the Processor.

TIMERS

A timer is provided for each data channel. The timers are countdown interval timers that are decremented by firmware at 40 counts per second. The value in the timer is an 8-bit unsigned integer, providing a maximum interval of about 6 seconds. When a timer decrements from $(01)_{16}$ to $(00)_{16}$ a CRI and timer Activity Flag are set for the channel. If a timer contains $(00)_{16}$, it is not decremented. A Stop I/O sets the timer to $(00)_{16}$. The timer is started by loading a non-zero value into LCT 74/106.

RESTART/DOWNLINE LOAD/DOWNLINE DUMP

A priviledged instruction, MC, is provided by which Processor software can cause a Master Clear of the system. The instruction causes the Processor to pull the BSMCLR line on the Level 6 bus to a true state for at least 250 microseconds. This causes all units on the bus to initialize and run their QLTs.

The Restart, Downline Load, and Downline Dump functions are provided by software as described in the following paragraph.

The system would need to have some kind of local bootload source for cold start of the system (e.g., diskette, main memory PROM, or host system coupler). This source would be used to get the system underway after power-up and would result in the loading of suitable programs and CCBs into the Processor and CPU to support the Restart and Downline functions desired. One or more lines of the Processor could be assigned to the Restart and Downline function when these are to be supported. The line(s) would probably be connected to the switched network or to a (Local) System Control Facility (SCF). The receive channel CCP would provide for automatic answer of the line and password-command checking. Among the commands to be implemented would be: Restart with bootload device, Downline Load and Execute, and Downline Dump of Memory. A channel timer could be assigned by software to be a local dead-man timer which, through the CCP, would initiate a RESTART if the CPU fails to perform some defined operation on the channel within the timeout period.

Section 3
CHANNEL CONTROL PROGRAM INSTRUCTION SET

A Channel Control Program (CCP) is created and then stored in the Processor RAM, where it serves as the interface between one or more processor communications channels and Communications Data Blocks (CDBs) in the main memory program. Each CCP handles a communications data stream being received by, or transmitted from, these CDBs. The data stream is handled one character at a time, and the CCP can modify or delete an individual character in the data stream or it can transfer the character unchanged. The CCP can also manipulate certain bytes in the Line Control Table (LCT) pertaining to the channel serviced by the CCP. This LCT information relates to CCBs, the line interface (adapter or Communications-Pac), and Data Communications Equipment (DCE).

INITIAL CCP SETUP

A CCP is written in CCP source assembly language and processed through the macro preprocessor and DPS 6/Level 6 assembler. This produces an object code CCP which is linked and then loaded into the Processor local store RAM for execution.

All data stored in Processor memory is based on 8-bit memory bytes. The format of each byte is designed from left to right, with the first bit numbered zero and the last seven. The right-most Bit (i.e., bit 7) is the least significant bit. All data is unsigned integer bytes.

Every CCP in the Processor must reside in consecutive locations of the CCP area of RAM. Multiple CCPs can coexist in RAM provided they do not overlap.

A CCP can service more than one communications channel, but each channel's LCT and CCBs exist in channel-specific RAM locations outside the CCP area, regardless of whether that channel is serviced by a dedicated CCP or by a CCP that services multiple channels. The channel-specific LCT and CCB storage areas permit CCPs to be reentrant and therefore able to service more than one channel.

A CCP is stored in the Processor RAM by a main memory program's use of a RAM Data Transfer or Block Mode Write operation (Section 1). In either case, the CCP's initial starting address must be written into the appropriate bytes of the LCT for the channel to be serviced by this CCP. (The format of LCTs is described in Section 5.) When the CCP is started for the first time, its initial starting address, stored in the LCT, will be loaded into the Processor's P-register (program counter) by firmware.

STARTING A CCP

Once all desired CCPs have been stored in RAM and all setup activity has occurred, the CCP can be started by the main memory program's execution of an IO (Output Channel Control - start input/output) instruction. For example, the first action of the CCP may be to load line registers 6, 4, and 2 of the appropriate channel of the adapter; line register 2 must be loaded last. The startup procedure is adapter-dependent; refer to the appropriate adapter in question. The CCP may then execute a WAIT instruction, pending the first communications message activity.

During processing, a CCP can be started by any of the following means:

1. A Channel Request Interrupt (CRI) from the appropriate adapter (a request for CCP service). This starts the receive or transmit CCP.

2. Execution of an IO (Output Channel Control - start input/output) instruction in the main memory program. This could start either a receive or transmit CCP.

3. A change in data set status (provided the firmware scan bit and the start CCP bit are both set to one in the appropriate LCT byte). This would start the receive CCP as a result of processing defined LCT 8 and would start the transmit CCP dependent on LCT 40.

4. Either channel of a line pair can turn on the other channel by setting the appropriate bit (6, 7) of the LCT byte 20 and the Line Register 2 (LR2). This technique is used for echoplexing the received data back to the terminal in full-duplex mode, or for managing line turnaround in two-way alternate mode. A change in transmit bit 7 in LR2 from zero to one will cause the transmit CCP to be turned on.

5. Channel Timer Exhaustion - When a timer (LCT 74/106) decrements from $(01)_{16}$ to $(00)_{16}$ an event-level service request is generated.

6. A Software-Created Service Request. The software interrupt field bits 5, 6, 7 of LCT 75/107 can be set by CCP to create an event-level service request.

Each time a CCP is started, the Processor restores its channel-specific context from the appropriate LCT. This context includes the proper settings of the P-register (program counter), R-register (general register), B-register (base register), and program indicators.

CCP EXECUTION

When the CCP is started, the Processor allows it to execute, without interruption, as many as 32 instructions. (Communications-Pac buffering is provided to ensure that consecutive execution of 31 instructions in one CCP does not cause an error - receive overrun or transmit underrun - on another communications channel.) After 31 instructions have been executed, a firmware pause occurs and the CCP is interrupted. The CCP's context is stored in firmware-reserved bytes of the appropriate LCT. The firmware pause allows background firmware scanning to occur and channel request interrupts to be serviced. When the CCP is resumed following the pause, its saved context is automatically restored by firmware. The pause is not apparent to the CCP unless it contains a timing loop.

The data character processing loop of a CCP typically handles a single character of the data stream and terminates with a WAIT instruction. When the CCP's WAIT instruction is executed, Processor firmware stores, in the appropriate LCT, the current contents of the P-register and program indicators. This context will be restored by the Processor when this CCP is started again.

NOTE

If, desired, the main memory program can alter the stored value of the P-register by issuing IO (Output LCT Byte) instructions only when the CCP is not executing. This action will cause the CCP to resume at a different RAM address when it is started again.

NMLCP CCP REGISTERS AND PROGRAM INDICATORS

The Processor CCP has two visible hardware registers, the R-register and B-register. The R-register is an 8-bit general purpose accumulator. All data and control sent to and received from the control adapters must be written from or read into this register. The B-register is an 8-bit limited purposes accumulator, which is primarily used as an LCT location pointer, or it can be used as an index to the extended local store (RAM) work area.

The Processor registers and program indicators of particular significance to the CCP are as follows:

1. P (Program counter) - a 16-bit pointer that defines the current location of CCP execution.

2. R (general-Register) - an 8-bit general purpose accumulator.

3. B (Base-register) - an 8-bit general purpose accumulator which also serves as an LCT pointer and local store index.

4. W (Work-register) - an 8-bit general purpose accumulator not visible to software.

5. LR (Line-Register) - an 8-bit register used to access the Communications Line Adapter.

6. FR (FLAP-Register) - an 8-bit register used to access the Flexible Line Adapter.

7. SOQ (Start of Queue pointer) - an 8-bit LCT value which identifies the current start of channel queue.

8. EOQ (End Of Queue pointer) - an 8-bit LCT value which identifies the current end of channel queue.

9. SP (Stack Pointer) - an 8-bit LCT value which identifies the current position of channel stack.

10. E (Equal indicator) - a status bit which reflects the result of the most recently executed Compare (C) instruction.

11. LC (Last Character indicator) - a status bit which reflects the result of the most recently executed Direct Memory Access Load or Store instruction.

12. LB (Last Block indicator) - a status bit which reflects the state of bit 2 of the control field of the current CCB.

13. VB (Valid Block indicator) - a status bit which reflects the state of bit 1 of the control field of the current CCB.

14. AR (Adapter Ready indicator) - a status bit which reflects the state of the adapter buffer.

15. C (Carry indicator) - a status bit which reflects the result of the most recently executed instruction which affects a carry.

16. Z (Zero indicator) - a status bit which reflects the result of the most recently executed instruction which affects a zero result.

17. SB (most Significant Bit indicator) - a status bit which reflects the state of bit 0 of the result of the most recently executed instruction which affects the most significant bit.

The following table (Table 3-1) defines the symbols used in describing the instruction set.

All communications data is stored right-justified in each Communications-Pac line register one and the Processors R-register; the same format is used in both registers. Parity, if used, is stored in the leftmost bit of the communications character.

The CCPs send/receive instructions and the Calculate Block Check (CCH) instruction can use the Processor's block-check hardware to support a Cyclic Redundancy Check (CRC).

Table 3-1. Instruction Symbols

Convention	Definition
( )	Contents of; e.g., (R) = contents of R
<--	Is replaced by; e.g., (R) <-- (LR1) = contents of R is replaced by contents of LR1
+	Addition operator
-	Subtraction operator
.	Multiplication operator
/	Division operator
^	AND
v	Inclusive OR
⊕	Exclusive OR
::	Compare
=	Equal to
<	Less than
>	Greater than
≤	Less than or equal to
≥	Greater than or equal to
EA	Effective address
[ ]	Optional field; e.g., [comments] = optional field for comments
(( ))	Indirection; e.g., ((B)) = contents of location pointed to by contents of B
:	Concatenation; e.g., (LCT):(B) = 16-bit quantity with (LCT) most significant

CCP CONTROL AND EXECUTABLE INSTRUCTIONS

CCP control statements and executable instructions are used to create the CCP. They are coded as macrocalls and constitute the source of the CCP. This source is processed by the macropreprocessor and then the expanded source module produced by the macropreprocessor is used as input to the assembler.

Program Development Tools

Before attempting to create a CCP, you should be thoroughly familiar with the description of the macrofacility in the appropriate assembly language manual. Refer also to the documentation listed in the appropriate software overview manual. You should be familiar with the program development tools of the operating system, and with the operating system macropreprocessor and assembler, and the library of Processor macroroutines. The library of macroroutines is used whenever the macropreprocessor executes.

Programming Rules

The CCP instructions that are described in the remainder of this section are those control and executable instructions that are used to create CCPs.

In a few instances, assembly language instructions have mnemonic op codes that are identical to the macronames of CCP generation control statements and CCP executable instructions (e.g., ORG, NOP, AND, OR, XOR, B, DEC). If any of these assembly language instructions appear in a source module that includes a CCP, they must be [protected] from being misinterpreted as macrocalls during execution of the macropreprocessor. See the appropriate assembly language manual for a definition.

Note also that the global macrovariables GX, GY, and GZ are reserved for use by the macroroutines and should not appear anywhere in input to the macropreprocessor.

The use of assembler control and executable instructions in the CCP is not permitted. A single exception exists to this rule, namely the assembler instruction EQU (Equate), which may be used within the CCP. The EQU has no effect during the macropreprocessor phase, but is passed along to be assembled during assembly of the CCP. The operand of the EQU statement must be a value; i.e., $ cannot be used as an operand. Refer also to the LOC statement defined later.

Macropreprocessor and Assembly Operation

Refer to the macropreprocessor section of the assembly language manual for the correct control information that must be provided to the macropreprocessor itself and to the assembler portion for proper operation.

Internal Formats

The assembler manual also includes a discussion of the terminal data formats and hardware registers in the DPS 6/Level 6 system. These should be thoroughly understood before attempting to program the Processor. To assist the user, the CCP instructions in this manual contain the ranges for internal value expressions that are generated; the definitions of internal value expressions, constants, arithmetic expressions, etc., are found in the assembler manual itself.

In the coding examples used for the CCP instructions, representative type of coding usage are shown.

CCP GENERATION CONTROL STATEMENTS

There are four CCP generation control statements (see Table 3-2) available to assist in the creation of a CCP:

- LOC
- ORG
- MORG
- DATA.

These statements, which are analogous to Assembler control statements, are described below.

Table 3-2. Format of Macrocalls for CCP Generation Control Statements

Instruction	Macrocall Format	Comments
CCP Generation	LOC operand [comments]	Operand is a user-supplied label
Control Statements	ORG operand [comments]	Operand is a decimal or hexadecimal constant.
	MORG [comments]	A modulo 2 value is assigned to the Macropreprocessor's byte allocation counter.
	DATA operand [,operand] [comments]	One to 35 operands are possible. Each operand is an internal value expression. $0 \leq IVE \leq 255$

LOC Statement

Format of Macrocall:

LOC operand [comments]

operand

A user-supplied label.

Description of Statement:

The LOC statement assigns a user-supplied label to the immediately following CCP byte location. The LOC label statement is comparable to the Assembler control statement label EQU $.

Example:

LOC START

ORG Statement

Format of Macrocall:

ORG operand [comments]

operand

A decimal or hexadecimal integer constant, where

$0 \leq \text{constant} \leq 65535$.

Description of Statement:

The ORG statement assigns a user-supplied value to the Macro-preprocessor's byte allocation counter. The CCP locations following the ORG statement will have byte addresses based on the value supplied in the ORG statement.

NOTE

The addresses established by the ORG statement do not necessarily indicate the RAM locations into which the generated CCP will eventually be loaded. This decision need not be made until the Processor is loaded.

If the first CCP generation control statement is not an ORG statement, the Macropreprocessor assumes an implied ORG statement whose operand is zero.

Example:

```
ORG 256
ORG X'0100'
```

MORG Statement

Format of Macrocall:

```
MORG [comments]
```

Description of Statement:

The MORG statement assigns, to the Macropreprocessor's byte allocation counter, a modulo 2 value relative to the address of the most recent Table Look-Up (TLU) instruction. If no TLU instruction has yet appeared in the CCP, the assigned modulo 2 value is relative to the CCP's most recent ORG statement (either explicit or implied; see preceding subsection).

This statement is used to ensure that a branch resulting from a TLU instruction will reach the proper memory address relative to the address of the TLU instruction.

Example:

```
MORG
```

DATA Statement

Format of Macrocall:

```
DATA operand [,operand...] [comments]
```

operand

An internal value expression that, when resolved, is a data constant modulo 256 (value from 0 to 255, inclusive); this data constant will be included in the CCP. From 1 to 35 operands can be included in a DATA statement.

Generated Code:

Description of Statement:

The DATA statement creates a string of 1 to 35 data constant bytes in the CCP. One use of this statement is to create a CCP table that can be used in conjunction with the TLU instruction.

Example:

```
DATA 0,1,2,3
DATA X'01',X'02',X'03'
DATA NUL,SOH,STX,ETX
DATA (START-BASE)/2+X'80'
```

INSTRUCTION EXECUTION TIMING

Table 3-3 provides a quick reference to the CCP instruction times in microseconds and the page number where each instruction is located. The following abbreviations explain the terms used in Table 3-3.

label= Internal Value Label

disp = 8-bit signed displacement between the current byte address in P and the operative byte address, where $-128 \leq disp \leq +127$.

Disp = 16-bit signed displacement between the current byte address in P and the operative byte address, where $-32768 \leq Disp \leq +32767$.

lct = Internal Value Expression which identifies the operative LCT entry, where $0 \leq lct \leq 255$.

imo = Internal Value Expression which identifies the operative IMO value, where $0 \leq imo \leq 255$.

lr = Internal Value Expression which identifies the operative LR, where $0 \leq lr \leq 7$.

fr = Internal Value Expression which identifies the operative FR, where $0 \leq fr \leq 7$.

Table 3-3. Instruction Timing and Location

Instruction	Functional Description	Time	Page
ADD R,lct	(R) <-- (R) + (LCT)	0.75	3-20
ADD B,lct	(B) <-- (B) + (LCT)	0.75	3-21
ADD R,B	(R) <-- (R) + ((B))	0.75	3-22
ADD R,=imo	(R) <-- (R) + IMO	0.75	3-23
ADD B,=imo	(B) <-- (B) + IMO	0.75	3-24
AND R,lct	(R) <-- (R) ^ (LCT)	0.75	3-25
AND B,lct	(B) <-- (B) ^ (LCT)	0.75	3-26
AND R,B	(R) <-- (R) ^ ((B))	0.75	3-27
AND R,=imo	(R) <-- (R) ^ IMO	0.75	3-28
AND B,=imo	(B) <-- (B) ^ IMO	0.75	3-29
B label	(P) <-- (P) + disp	1.5	3-30
BS label	(LCT 18) <-- (P) + 1 (P) <-- (P) + disp	3.75	3-31
BARF label	If AR=0; then (P) <-- (P) + disp	3.0	
	else (P) <-- (P) + 1	4.5	3-32
BART label	If AR=1; then (P) <-- (P) + disp	3.0	
	else (P) <-- (P) + 1	4.5	3-33
BCF label	If C=0; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-34
BCT label	If C=1; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-35
BEF label	If E=0; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-36
BET label	If E=1; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-37
BLBF label	If LB=0; then (P) <-- (P) + disp	1.5	
	else (P) <-- (P) + 1	2.8	3-38
BLBT label	If LB=1; then (P) <-- (P) + disp	1.5	
	else (P) <-- (P) + 1	2.8	3-39
BLCF label	If LC=0; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-40
BLCT label	If LC=1; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-41
BSBF label	If SB=0; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-42
BSBT label	If SB=1; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-43
BSF label	If Z=0; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-44
BST label	If Z=1; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-45
BVBF label	If VB=0; then (P) <-- (P) + disp	1.8	
	else (P) <-- (P) + 1	2.8	3-46

Table 3-3 (cont). Instruction Timing and Location

Instruction	Functional Description	Time	Page
BVBT label	If VB=1; then (P) <-- (P) + disp	1.8	
	else (P) <-- (P) + 1	2.8	3-47
BZF label	If Z=0; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-48
BZT label	If Z=1; then (P) <-- (P) + disp	0.75	
	else (P) <-- (P) + 1	2.0	3-49
C R,lct	(W) <-- (R) - (LCT)	0.75	3-50
C B,lct	(W) <-- (B) - (LCT)	0.75	3-51
C R,B	(W) <-- (R) - ((B))	0.75	3-52
C R,=imo	(W) <-- (R) - IMO	0.75	3-53
C B,=imo	(W) <-- (B) - IMO	0.75	3-54
CADD R,lct	(R) <-- (R) + C + (LCT)	3.25	3-55
CADD B,lct	(B) <-- (B) + C + (LCT)	3.25	3-56
CADD R,B	(R) <-- (R) + C + ((B))	3.4	3-57
CADD R,=imo	(R) <-- (R) + C + IMO	3.2	3-58
CADD B,=imo	(B) <-- (B) + C + IMO	3.2	3-59
CANB	Curr Address <-- Init Address		
	Curr Range <-- Init Range	6.75	3-60
CANC	(W) <-- Curr Range - Init Range		
	If Z = 0; then		
	Curr Address <-- Curr Address - 1		
	Curr Range <-- Curr Range + 1	TBD	3-61
CCH	(CRC) <-- (CRC) + (R)	3.9-9.5	3-62
CL lct	(LCT) <-- 0	0.75	3-63
CL B	((B)) <-- 0	0.75	3-64
CL =R	(R) <-- 0	0.38	3-65
CL =B	(B) <-- 0	0.38	3-66
DADD R,lct	(R) <-- (R) + (LCT)	5.88	3-67
DADD R,=imo	(R) <-- (R) + IMO	5.75	3-68
DEC =R	(R) <-- (R) - 1	0.38	3-69
DEC lct	(LCT) <-- (LCT) - 1	0.38	3-70
DEC B	((B)) <-- ((B)) - 1	0.38	3-71
DEC =B	(B) <-- (B) - 1	1.0	3-72
DEC *lct[a]	((LCT):(B)) <-- ((LCT):(B)) - 1	7.0	3-73
DEQ[b]	(R) <-- ((SOQ))		
	(SOQ) <-- (SOQ) + 1	1.2	3-74
ENQ[b]	((EOQ)) <-- (R)		
	(EOQ) <-- (EOQ) + 1	1.2	3-75

Table 3-3 (cont). Instruction Timing and Location

Instruction	Functional Description	Time	Page
FIN fr	(R) <-- (FR)	2.6	3-76
FOUT fr	(FR) <-- (R)	4.4	3-77
GIVE	Post Current CCB - Extended	TBD	3-78
GNB	Post Current CCB - Compatible	10.0	3-79
IAS	(R) <-- CA Status	3.4	3-80
ILP lct,label	(LCT) <-- (P) + Disp	1.2	3-81
IN lr	(R) <-- (LR)	TBD	3-82
INC lct	(LCT) <-- (LCT) + 1	0.38	3-83
INC B	((B)) <-- ((B)) + 1	0.38	3-84
INC =R	(R) <-- (R) + 1	0.38	3-85
INC =B	(B) <-- (B) + 1	1.0	3-86
INC *lct[a]	((LCT):(B)) <-- ((LCT):(B)) + 1	7.0	3-87
INTR	Interrupt CPU	TBD	3-88
INZ	Stop I/O	TBD	3-89
JS label[b]	(SP) <-- (SP) - 1 ((SP)) <-- (P) + 1 (SP) <-- (SP) - 1 (P) <-- (P) + Disp	6.2	3-90
JS *lct[b]	(SP) <-- (SP) - 1 ((SP)) <-- (P) + 1 (SP) <-- (SP) - 1 (P) <-- (LCT)	TBD	3-91
JUMP label	(P) <-- (P) + Disp	3.0	3-92
JUMP *lct[a]	(P) <-- (LCT)	TBD	3-93
LB lct,=imo	(W) <-- (LCT) ^ IMO	1.25	3-94
LB B,=imo	(W) <-- ((B)) ^ IMO	0.25	3-95
LB =R,=imo	(W) <-- (R) ^ IMO	0.75	3-96
LB =B,=imo	(W) <-- (B) ^ IMO	0.75	3-97
LB *lct,=imo[a]	(W) <-- ((LCT):(B)) ^ $\overline{\text{IMO}}$	6.1	3-98
LBF lct,=imo	(W) <-- (LCT) ^ $\underline{\text{IMO}}$ (LCT) <-- (LCT) ^ IMO	2.3	3-99

Table 3-3 (cont). Instruction Timing and Location

Instruction	Functional Description	Time	Page
LBF B,=imo	(W) <-- ((B)) ^ IMO ((B)) <-- ((B)) ^ IMO	2.3	3-100
LBF *lct,=imo[a]	(W) <-- ((LCT):(B)) ^ IMO ((LCT):(B)) <-- ((LCT):(B)) ^ IMO	7.9	3-101
LBT lct,=imo	(W) <-- (LCT) ^ IMO (LCT) <-- (LCT) v IMO	2.3	3-102
LBT B,=imo	(W) <-- ((B)) ^ IMO ((B)) <-- ((B)) v IMO	2.3	3-103
LBT *lct,=imo[a]	(W) <-- ((LCT):(B)) ^ IMO ((LCT):(B)) <-- ((LCT):(B)) v IMO	2.3	3-104
LD,	(R) <-- (CDB)	0.75	3-105
LD R,lct	(R) <-- (LCT)	0.75	3-106
LD B,lct	(B) <-- (LCT)	0.75	3-107
LD R,B	(R) <-- ((B))	0.75	3-108
LD R,=imo	(R) <-- IMO	0.75	3-109
LD B,=R	(B) <-- (R)	0.75	3-110
LD B,=imo	(B) <-- IMO	0.75	3-111
LD R,=B	(R) <-- (B)	0.75	3-112
LD R,*lct[a]	(R) <-- ((LCT):(B))	6.1	3-113
MC	Master Clear	TBD	3-114
NEG[c]	(R) <-- ($\overline{R}$) + 1	TBD	3-115
NOP	(P) <-- (P) + 1	0.38	3-116
OAC[d]	CA Control <-- (R)	TBD	3-117
OR R,lct	(R) <-- (R) v (LCT)	0.75	3-118
OR B,lct	(B) <-- (B) v (LCT)	0.75	3-119
OR R,B	(R) <-- (R) v ((B))	0.75	3-120
OR R,=imo	(R) <-- (R) v IMO	0.75	3-121
OR B,=imo	(B) <-- (B) v IMO	0.75	3-122
OUT lr	(LR) <-- (R)	TBD	3-123
PULL B[b]	(SP) <-- (SP) + 1 (B) <-- ((SP))	1.1	3-124
PULL R[b]	(SP) <-- (SP) + 1 (R) <-- ((SP))	1.1	3-125

Table 3-3 (cont). Instruction Timing and Location

Instruction	Functional Description	Time	Page
PUSH B[b]	((SP)) <-- (B) (SP) <-- (SP) - 1	1.1	3-126
PUSH R[b]	((SP)) <-- (R) (SP) <-- (SP) - 1	1.1	3-127
RECV 0	(R) <-- (LR1) $\overline{par}$;$\overline{crc}$	5.4	3-128
RECV 1	(R) <-- (LR1) $\overline{par}$;crc	11.1	3-128
RECV 2	(R) <-- (LR1) par;$\overline{crc}$	8.5	3-128
RECV 3	(R) <-- (LR1) par;crc	13.8	3-128
RETB	(P) <-- (LCT 18)	1.5	3-129
RETJ[b]	(SP) <-- (SP) + 1 (P) <-- ((SP)) (SP) <-- (SP) + 1	2.5	3-130
RHB	Return Held Block	TBD	3-131
SCF	C <-- 0	1.2	3-132
SCL R	(R) <-- (R) . 2 + C	1.3	3-133
SCL B	(B) <-- (B) . 2 + C	1.3	3-134
SCR R	(R) <-- (R) / 2 + C . X'80'	1.3	3-135
SCR B	(B) <-- (B) / 2 + C . X'80'	1.3	3-136
SCT	C <-- 1	1.2	3-137
SEND 0	(LR1) <-- (R) $\overline{par}$;$\overline{crc}$	5.8	3-138
SEND 1	(LR1) <-- (R) $\overline{par}$;crc	10.9	3-138
SEND 2	(LR1) <-- (R) par;$\overline{crc}$	7.9	3-138
SEND 3	(LR1) <-- (R) par;crc	12.9	3-138
SFS	Search For Synchronization	23.7	3-139
SOL R	(R) <-- (R) . 2	0.38	3-140
SOL B	(B) <-- (B) . 2	0.38	3-141
SOR R	(R) <-- (R) / 2	1.5	3-142
SOR B	(B) <-- (B) / 2	1.3	3-143
ST,	(CDB) <-- (R)	7.3	3-144
ST R,lct	(LCT) <-- (R)	0.75	3-145
ST B,lct	(LCT) <-- (B)	0.75	3-146
ST R,B	((B)) <-- (R)	TBD	3-147
ST R,*lct	((LCT):(B)) <-- (R)	6.8	3-148

Table 3-3 (cont). Instruction Timing and Location

Instruction	Functional Description	Time	Page
SUB R,lct	(R) <-- (R) - (LCT)	0.75	3-149
SUB B,lct	(B) <-- (B) - (LCT)	0.75	3-150
SUB R,B	(R) <-- (R) - ((B))	TBD	3-151
SUB R,=imo	(R) <-- (R) - IMO	0.75	3-152
SUB B,=imo	(B) <-- (B) - IMO	0.75	3-153
TLU *lct	(W) <-- ((LCT) + (R)) If SB=0, then (R) <-- (W) (P) <-- (P) + 1 else (W) <-- (W) ^ X'7F' (W) <-- (W) . 2 (P) <-- (P) + 3 + (W)	6.0 7.5	3-154
TQE[b]	(W) <-- (SOQ) - (EOQ)	0.75	3-155
TQF[b]	(W) <-- (SOQ) - 15 - (EOQ)	1.1	3-156
WAIT	Suspend CCB execution	13.25	3-157
XOR R,lct	(R) <-- (R) ⊕ (LCT)	0.75	3-158
XOR B,lct	(B) <-- (B) ⊕ (LCT)	0.75	3-159
XOR R,B	(R) <-- (R) ⊕ ((B))	0.75	3-160
XOR R,=imo	(R) <-- (R) ⊕ IMO	0.75	3-161
XOR B,=imo	(B) <-- (B) ⊕ IMO	0.75	3-162

[a]Extended LCTs are not channel related and must be handled on a controller basis.
[b]Stack operation pointers must be set prior to issuing these instructions.
[c]Exclusive OR.
[d]Plus Personality PROM, if applicable.

CCP EXECUTABLE INSTRUCTIONS

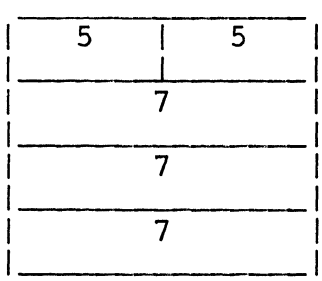
The following is a sample of the instruction form used in this section. A key to each function is listed below.

1

2 _____

Format: 3 (4)

Time: 6 ____ microseconds



Summary: 8

Usage: 9

Indicators: 10

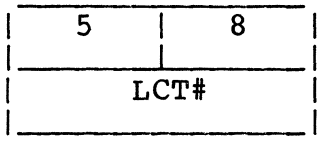
Example: 11

Abbreviations In This Instruction: 12

1 - Instruction mnemonic
2 - Instruction mnemonic name
3 - Operand
4 - Hardware equivalent mnemonic
5 - Operation code
6 - Instruction time
7 - Additional bytes as required
8 - Summary of instruction
9 - Instruction usage
10 - Indicators affected
11 - Programming example
12 - Abbreviations used within the instruction

ADD R-REGISTER WITH LCT

Format: ADD R,lct (ADD)



Time: 0.75 microseconds

Summary: ADD the contents of LCT to the contents of R-register. The result is saved in R-register.

(R) <-- (R) + (LCT)

Usage: The ADD instruction is used to calculate the sum of two values.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: ADD R,200

```
        R-REG = |0 0 1 0 0 1 0 1
    LCT 200 = |1 1 1 1 1 0 0 1
Result R-REG = |0 0 0 1 1 1 1 0
```

C = 1, SB = 0, Z = 0

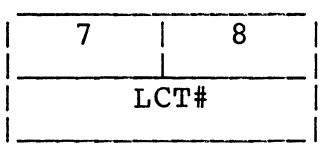
Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD B-REGISTER WITH LCT

Format: ADD B,lct (ADDB)

Time: 0.75 microseconds



Summary: ADD the contents of LCT to the contents of B-register. The result is saved in B-register.

(B) <-- (B) + (LCT)

Usage: The ADD instruction is used to calculate the sum of two values.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: ADD B,210

```
        B-REG = |1 1 0 0 0 0 1 1
    LCT 210 = |1 0 0 0 0 1 0 1
Result B-REG = |0 1 0 0 1 0 0 0
```

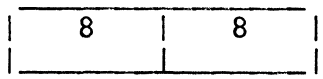
C = 1, SB = 0, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD R-REGISTER WITH B-REFERENCED LCT

Format: ADD R,B (ADDRN)



Time: 0.75 microseconds

Summary: ADD the contents of LCT pointed to by B-register to the contents of R-register. The result is saved in R-register.

(R) <-- (R) + ((B))

Usage: The ADD instruction is used to calculate the sum of two values.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: ADD R,B

```
                R-REG = |1 0 1 0 0 0 0 1
B=C8       LCT 200 = |1 1 1 1 0 1 0 0
     Result R-REG = |1 0 0 1 0 1 0 1
```

C = 1, SB = 1, Z = 0

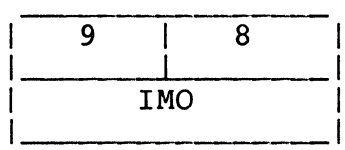
Before Execution	After Execution
R = A1	R = 95
B = C8	B = C8
LCT 200 = F4	LCT 200 = F4

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD R-REGISTER WITH IMO

Format: ADD R,=imo (ADIR)



Time: 0.75 microseconds

Summary: ADD IMO to the contents of R-register. The result is saved in R-register.

(R) <-- (R) + IMO

Usage: The ADD instruction is used to calculate the sum of two values.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: ADD R,=X'69'

```
        R-REG = |1 0 0 1 0 0 0 1
          IMO = |0 1 1 0 1 0 0 1
Result R-REG = |1 1 1 1 1 0 1 0
```

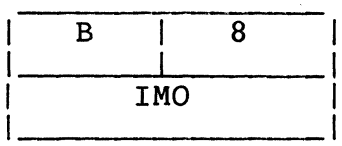
C = 0, SB = 1, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD B-REGISTER WITH IMO

Format: ADD B,=imo (ADIB)



Time: 0.75 microseconds

Summary: ADD IMO to the contents of B-register. The result is saved in B-register.

(B) <-- (B) + IMO

Usage: The ADD instruction is used to calculate the sum of two values.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: ADD B,=X'A5'

```
          IMO = |1 0 1 0 0 1 0 1
        B-REG = |0 1 0 1 1 0 1 1
Result B-REG = |0 0 0 0 0 0 0 0
```

C = 1, SB = 0, Z = 1

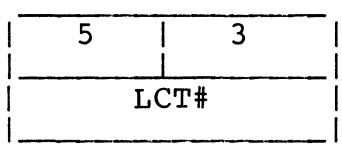
Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

AND R-REGISTER WITH LCT

Format: AND R,lct (AND)

Time: 0.75 microseconds



Summary: Logically AND the contents of LCT with the contents of R-register. The result is saved in R-register.

(R) <-- (R) ^ (LCT)

Usage: The AND instruction is used to isolate a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: AND R,16

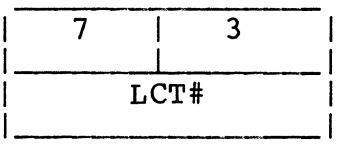
```
        R-REG = |0 1 1 1 0 0 1 0
          LCT = |0 0 1 1 0 1 1 1
Result R-REG = |0 0 1 1 0 0 1 0

SB = 0,    Z = 0
```

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

AND B-REGISTER WITH LCT

Format: AND B,lct (ANDB)



Time: 0.75 microseconds

Summary: Logically AND the contents of LCT with the contents of B-register. The result is saved in B-register.

(B) <-- (B) ^ (LCT)

Usage: The AND instruction is used to isolate a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

Z - Set if results equal Zero; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.

Example: AND B,210

```
          B-REG = |0 1 1 1 0 0 1 0
            LCT = |0 0 1 1 0 1 1 1
Result B-REG = |0 0 1 1 0 0 1 0

SB = 0,    Z = 0
```

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

AND R-REGISTER WITH B-REFERENCED LCT

Format: AND R,B (ANDRN)

8	3

Time: 0.75 microseconds

Summary: Logically AND the contents of LCT pointed to by B-register with the contents of R-register. The result is saved in R-register.

(R) <-- (R) ^ ((B))

Usage: The AND instruction is used to isolate a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

Z - Set if results equal Zero; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.

Example: AND R,B

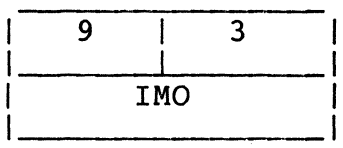
```
                R-REG = |0 1 1 1 0 0 1 0
B=D2       LCT 210 = |0 0 1 1 0 1 1 1
Result R-REG = |0 0 1 1 0 0 1 0

SB = 0,    Z = 0
```

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

AND R-REGISTER WITH IMO

Format: AND R,=imo (AND)



Time: 0.75 microseconds

Summary: Logically AND IMO with the contents of R-register. The result is saved in R-register.

(R) <-- (R) ^ IMO

Usage: The AND instruction is used to isolate a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

Z - Set if results equal Zero; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.

Example: AND R,=X'37'

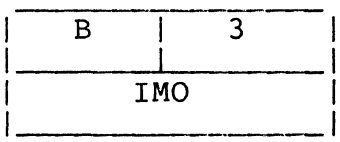
```
        R-REG = |0 1 1 1 0 0 1 0
          IMO = |0 0 1 1 0 1 1 1
Result R-REG = |0 0 1 1 0 0 1 0

SB = 0,    Z = 0
```

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

AND B-REGISTER WITH IMO

Format: AND B,=imo (ANIB)



Time: 0.75 microseconds

Summary: Logically AND IMO with the contents of B-register. The result is saved in B-register.

(B) <-- (B) ^ IMO

Usage: The AND instruction is used to isolate a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

Z - Set if results equal Zero; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.

Example: AND B,=X'37'

```
        B-REG = |0 1 1 1 0 0 1 0
          IMO = |0 0 1 1 0 1 1 1
Result B-REG = |0 0 1 1 0 0 1 0

SB = 0,    Z = 0
```

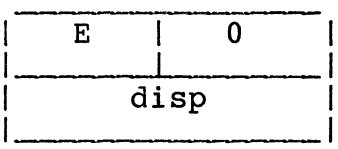
Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

B

UNCONDITIONAL BRANCH

Format: B label (B)



Time: 1.5 microseconds

Summary: Branch to the routine which is identified by label.

(P) <-- (P) + disp

Usage: The Branch (B) instruction is used to unconditionally transfer control to a routine within the range of a short displacement, where -128 ≤ disp ≤ +127.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

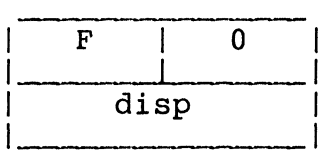
Example:

Abbreviations In This Instruction:

None

BRANCH TO SUBROUTINE

Format: BS label (BS)



Time: 3.75 microseconds

Summary: Branch to the subroutine which is identified by label.

```
LCT 18 <-- (P) + 1
   (P) <-- (P) + disp
```

Usage: The Branch to Subroutine (BS) instruction is used to unconditionaly transfer control to a subroutine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

Example:

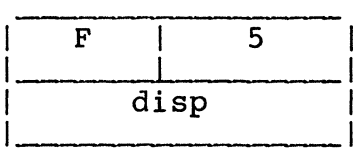
Abbreviations In This Instruction:

None

BARF

BRANCH ON ADAPTER READY FALSE

Format: BARF label (BARF)



Time: 3.0/4.5 microseconds

Summary: Branch to the routine which is identified by label if the Adapter Ready indicator is false.

```
If AR=0; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on Adapter Ready False (BARF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

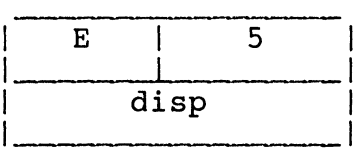
Example:

Abbreviations In This Instruction:

None

BRANCH ON ADAPTER READY TRUE

Format: BART label (BART)



Time: 3.0/4.5 microseconds

Summary: Branch to the routine which is identified by label if the Adapter Ready indicator is true.

```
If AR=1; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on Adapter Ready True (BART) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-byte field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

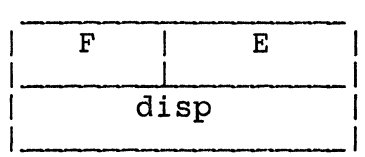
Example:

Abbreviations In This Instruction:

None

BRANCH ON CARRY FALSE

Format: BCF label (BCF)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Carry indicator is false.

If C=0; then (P) <-- (P) + disp
else (P) <-- (P) + 1

Usage: The Branch on Carry False (BCF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where -128 ≤ disp ≤ +127.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

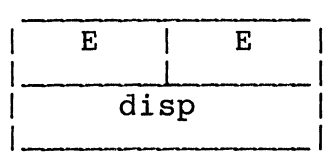
Example:

Abbreviations In This Instruction:

None

BANCH ON CARRY TRUE

Format: BCT label (BCT)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Carry indicator is true.

```
If C=1; then (P) <-- (P) + disp
        else (P) <-- (P) + 1
```

Usage: The Branch on Carry True (BCT) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where -128 ≤ disp ≤ +127.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

The Carry indicator is affected by the following instructions; ADD, CADD, DADD, DEC, INC, SCF, SCL, SCR, SCT, SOL, SOR, or SUB.

Indicators:

None affected.

Example:

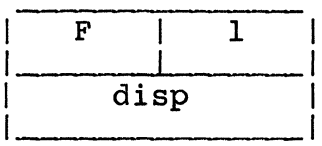
Abbreviations In This Instruction:

None

BEF

BRANCH ON EQUAL FALSE

Format: BEF label (BEF)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Equal indicator is false.

```
If E=0; then (P) <-- (P) + disp
        else (P) <-- (P) + 1
```

Usage: The Branch on Equal False (BEF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

Example:

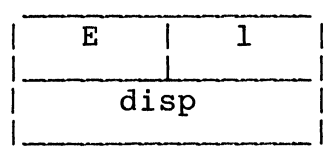
Abbreviations In This Instruction:

None

BRANCH ON EQUAL TRUE

Format: BET label (BET)

Time: 0.75/2.0 microseconds



Summary: Branch to the routine which is identified by label if the Equal indicator is true.

If E=1; then (P) <-- (P) + disp
else (P) <-- (P) + 1

Usage: The Branch on Equal True (BET) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

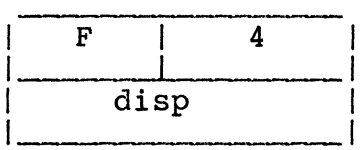
Example:

Abbreviations In This Instruction:

None

BRANCH ON LAST BLOCK FALSE

Format: BLBF label (BLBF)



Time: 1.5/2.8 microseconds

Summary: Branch to the routine which is identified by label if the Last Block indicator is false.

If LB=0; then (P) <-- (P) + disp
else (P) <-- (P) + 1

Usage: The Branch on Last Block False (BLBF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where -128 ≤ disp ≤ +127.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

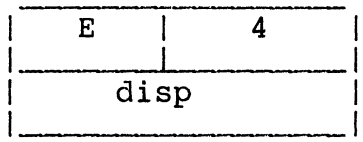
Example:

Abbreviations In This Instruction:

None

BRANCH ON LAST BLOCK TRUE

Format: BLBT label (BLBT)



Time: 1.5/2.8 microseconds

Summary: Branch to the routine which is identified by label if the Last Block indicator is true.

If LB=1; then (P) <-- (P) + disp
else (P) <-- (P) + 1

Usage: The Branch on Last Block True (BLBT) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

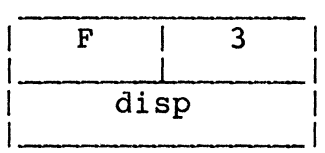
Example:

Abbreviations In This Instruction:

None

BRANCH ON LAST CHARACTER FALSE

Format: BLCF label (BLCF)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Last Character indicator is false.

```
If LC=0; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on Last Character False (BLCF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

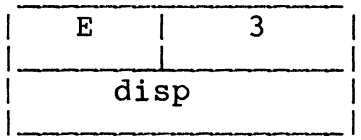
Example:

Abbreviations In This Instruction:

None

BRANCH ON LAST CHARACTER TRUE

Format: BLCT label (BLCT)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Last Character indicator is true.

```
If LC=1; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on Last Character True (BLCT) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

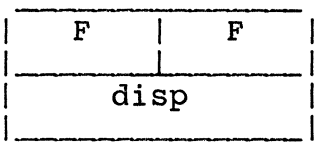
Example:

Abbreviations In This Instruction:

None

BRANCH ON MOST SIGNIFICANT BIT FALSE

Format: BSBF label (BPL)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the MSB indicator is false.

```
If SB=0; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on most Significant Bit False (BSBF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

The MSB indicator is affected by the following instructions; ADD, AND, C, CADD, CL, DADD, DEC, DEQ, FIN, IAS, IN, INC, LB, LBF, LBT, LD, OR, PULL, RECV, SCL, SCR, SOL, SOR, ST, SUB, TLU, or XOR.

Indicators:

None affected.

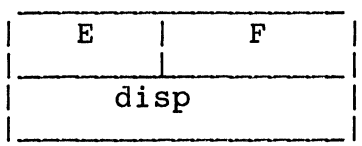
Example:

Abbreviations In This Instruction:

None

BRANCH ON MOST SIGNIFICANT BIT TRUE

Format: BSBT label (BMI)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the MSB indicator is true.

```
If SB=1; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on most Significant Bit True (BSBT) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

The MSB indicator is affected by the following instructions; ADD, AND, C, CADD, CL, DADD, DEC, DEQ, FIN, IAS, IN, INC, LB, LBF, LBT, LD, OR, PULL, RECV, SCL, SCR, SOL, SOR, ST, SUB, TLU, or XOR.

Indicators:

None affected.

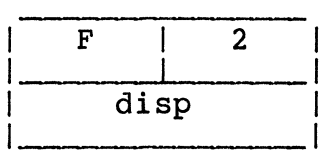
Example:

Abbreviations In This Instruction:

None

BRANCH ON STATUS FALSE

Format: BSF label (BZF)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Zero indicator is false.

```
If Z=0; then (P) <-- (P) + disp
        else (P) <-- (P) + 1
```

Usage: The Branch on Status False (BSF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTES

1. The displacement is represented as a signa 8-bit field which is the 2's complement difference (label byte address - operand byte address).

2. In the NMLCP the BSF instruction is functionally equivalent to the BZF instruction. In other controllers they are unique instructions.

Indicators:

None affected.

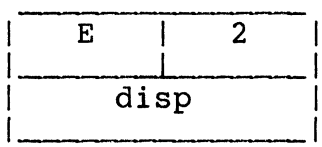
Example:

Abbreviations In This Instruction:

None

BRANCH ON STATUS TRUE

Format: BST label (BZT)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Zero indicator is true.

```
If Z=1; then (P) <-- (P) + disp
        else (P) <-- (P) + 1
```

Usage: The Branch on Status True (BST) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where -128 ≤ disp ≤ +127.

NOTES

1. The displacement is represented as a signal 8-bit field which is thD 2's complement difference (label byte address - operand byte address).

2. In the NMLCP the BST instruction is functionally equivalent to the BZT Dnstruction. other controllers they are unique instructions.

Indicators:

None affected.

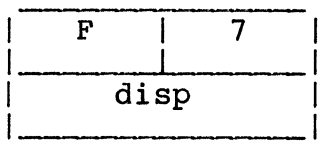
Example:

Abbreviations In This Instruction:

None

BRANCH ON VALID BLOCK FALSE

Format: BVBF label (BVBF)



Time: 1.8/2.8 microseconds

Summary: Branch to the routine which is identified by label if the Valid Block indicator is false.

```
If VB=0; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on Valid Block False (BVBF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

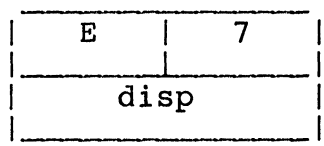
Example:

Abbreviations In This Instruction:

None

BRANCH ON VALID BLOCK TRUE

Format: BVBT label (BVBT)



Time: 1.8/2.8 microseconds

Summary: Branch to the routine which is identified by label if the Valid Block indicator is true.

```
If VB=1; then (P) <-- (P) + disp
         else (P) <-- (P) + 1
```

Usage: The Branch on Valid Block True (BVBT) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

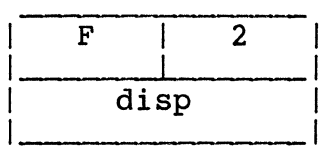
Example:

Abbreviations In This Instruction:

None

BRANCH ON ZERO FALSE

Format: BZF label (BZF)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Zero indicator is false.

```
If Z=0; then (P) <-- (P) + disp
        else (P) <-- (P) + 1
```

Usage: The Branch on Zero False (BZF) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

The Zero indicator is affected by the following instructions; ADD, AND, C, CADD, CL, DADD, DEC, FIN, IAS, IN, INC, LB, LBF, LBT, LD, OR, PULL, RECV, SCL, SCR, SOL, SOR, ST, SUB, TLU, or XOR.

CANC - The Zero indicator will be set false if a character is canceled.

GIVE - The Zero indicator will be set false when a block is given.

Indicators:

None affected.

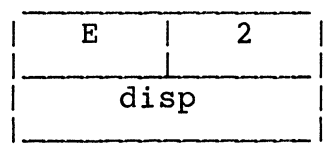
Example:

Abbreviations In This Instruction:

None

BRANCH ON ZERO TRUE

Format: BZT label (BZT)



Time: 0.75/2.0 microseconds

Summary: Branch to the routine which is identified by label if the Zero indicator is true.

If Z=1; then (P) <-- (P) + disp
else (P) <-- (P) + 1

Usage: The Branch on Zero True (BZT) instruction is used to conditionally transfer control to a routine within the range of a short displacement, where $-128 \leq disp \leq +127$.

NOTE

The displacement is represented as a signed 8-bit field which is the 2's complement difference (label byte address - operand byte address).

The Zero indicator is affected by the following instructions; ADD, AND, C, CADD, CL, DADD, DEC, FIN, IAS, IN, INC, LB, LBF, LBT, LD, OR, PULL, RECV, SCL, SCR, SOL, SOR, ST, SUB, TLU, or XOR.

CANC - The Zero indicator will be set true if there is no character to cancel.

GIVE - The Zero indicator will be set true if there is no block to give.

Indicators:

None affected.

Example:

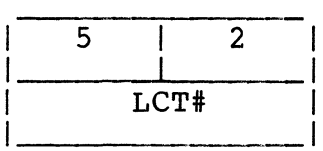
Abbreviations In This Instruction:

None

COMPARE R-REGISTER WITH LCT

Format: C R,lct (C)

Time: 0.75 microseconds



Summary: Subtract the contents of LCT from the contents of R-register. The result is not saved.

(W) <-- (R) - (LCT)

Usage: The Compare (C) instruction is used to test the equality of two values without modifing either value.

Indicators:

C - Set if carry out; otherwise reset.
E - Set if the two values are equal; otherwise reset.

Example:

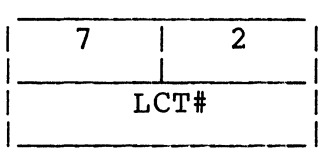
Abbreviations In This Instruction:

C = Carry Bit
E = Equal Indicator
LCT = Line Control Table

COMPARE B-REGISTER WITH LCT

Format: C B,lct (CB)

Time: 0.75 microseconds



Summary: Subtract the contents of LCT from the contents of B-register. The result is not saved.

(W) <-- (B) - (LCT)

Usage: The Compare (C) instruction is used to test the equality of two values without modifing either value.

Indicators:

C - Set if carry out; otherwise reset.
E - Set if the two values are equal; otherwise reset.

Example: C B,30 = Compare B-register to LCT 30.

Abbreviations In This Instruction:

C = Carry Bit
E = Equal Indicator
LCT = Line Control Table

C

COMPARE R-REGISTER WITH B-REFERENCED LCT

Format: C R,B (CRN)

Time: 0.75 microseconds

Summary: Subtract the contents of LCT pointed to by B-register from the contents of R-register. The result is not saved.

(W) <-- (R) - ((B))

Usage: The Compare (C) instruction is used to test the equality of two values without modifing either value.

Indicators:

C - Set if carry out; otherwise reset.
E - Set if the two values are equal; otherwise reset.

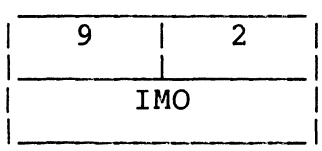
Example: C R,B = Compare R-register with LCT pointed to by B-register.

Abbreviations In This Instruction:

C = Carry Bit
E = Equal Indicator
LCT = Line Control Table

COMPARE R-REGISTER WITH IMO

Format: C R,=imo (C)



Time: 0.75 microseconds

Summary: Subtract IMO from the contents of R-register. The result is not saved.

(W) <-- (R) - IMO

Usage: The Compare (C) instruction is used to test the equality of two values without modifing either value.

Indicators:

C - Set if carry out; otherwise reset.
E - Set if the two values are equal; otherwise reset.

Example: C R,=X'20' Compare R-register with IMO X'20'.

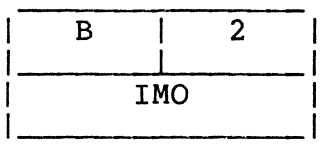
Abbreviations In This Instruction:

C = Carry Bit
E = Equal Indicator
IMO = Immediate Memory Operand
LCT = Line Control Table

C

COMPARE B-REGISTER WITH IMO

Format: C B,=imo (CBI)



Time: 0.75 microseconds

Summary: Subtract IMO from the contents of B-register. The result is not saved.

(W) <-- (B) - IMO

Usage: The Compare (C) instruction is used to test the equality of two values without modifing either value.

Indicators:

C - Set if carry out; otherwise reset.
E - Set if the two values are equal; otherwise reset.

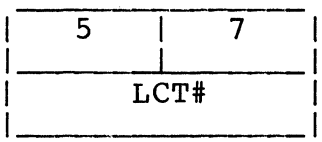
Example:

Abbreviations In This Instruction:

C = Carry Bit
E = Equal Indicator
IMO = Immediate Memory Operand
LCT = Line Control Table

ADD R-REGISTER WITH LCT PLUS CARRY

Format: CADD R,lct (CADD)



Time: 3.25 microseconds

Summary: ADD the carry plus the contents of the LCT to the contents of R-register. The result is saved in R-register.

(R) <-- (R) + C + (LCT)

Usage: The Carry ADD (CADD) instruction is used to add a number that is more than one byte wide (a number greater than 255). The CADD is used on the more significant bytes to propagate the carry bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: CADD R,58

```
      R-REG = |0 0 1 1 1 0 1 1
          C = |               1
  LCT 58 = |0 1 0 0 0 1 0 0
Result R-REG = |1 0 0 0 0 0 0 0
```

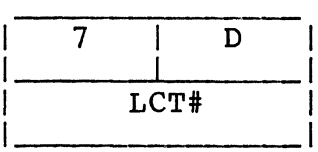
C = 0, SB = 1, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD B-REGISTER WITH LCT PLUS CARRY

Format: CADD B,lct (CADB)



Time: 3.25 microseconds

Summary: ADD the carry plus the contents of the LCT to the contents of B-register. The result is saved in B-register.

(B) <-- (B) + C + (LCT)

Usage: The Carry ADD (CADD) instruction is used to add a number that is more than one byte wide (a number greater than 255). The CADD is used on the more significant bytes to propagate the carry bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: CADD B,60

```
        B-REG = |0 0 1 1 1 0 1 1
            C = |               1
   LCT 60 = |0 1 0 0 0 1 0 0
Result B-REG = |1 0 0 0 0 0 0 0
```

C = 0, SB = 1, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD R-REGISTER WITH B-REFERENCED LCT PLUS CARRY

Format: CADD R,B (CADRN)

8	C

Time: 3.4 microseconds

Summary: ADD the carry plus the contents of the LCT pointed to by B-register to the contents of R-register. The result is saved in R-register.

(R) <-- (R) + C + ((B))

Usage: The Carry ADD (CADD) instruction is used to add a number that is more than one byte wide (a number greater than 255). The CADD is used on the more significant bytes to propagate the carry bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: CADD R,B

```
                R-REG = |0 0 1 1 1 0 1 1
                    C = |               1
B=C8     LCT 200 = |0 1 0 0 0 1 0 0
   Result R-REG = |1 0 0 0 0 0 0 0
```

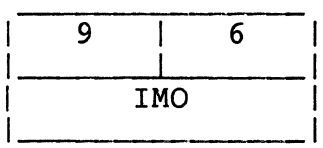
C = 0, SB = 1, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD R-REGISTER WITH IMO PLUS CARRY

Format: CADD R,=imo (CADIR)



Time: 3.2 microseconds

Summary: ADD the carry plus IMO to the contents of R-register. The result is saved in R-register.

(R) <-- (R) + C + IMO

Usage: The Carry ADD (CADD) instruction is used to add a number that is more than one byte wide (a number greater than 255). The CADD is used on the more significant bytes to propagate the carry Bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: CADD R,=68

```
        R-REG = |0 0 1 1 1 0 1 1
            C = |               1
          IMO = |0 1 1 1 1 0 1 0
Result R-REG = |0 0 1 1 0 1 1 0
```

C = 1, SB = 1, Z = 0

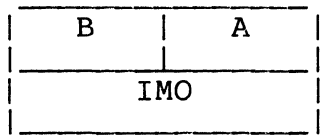
Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

ADD B-REGISTER WITH IMO PLUS CARRY

Format: CADD B,=imo (CADIB)

Time: 3.2 microseconds



Summary: ADD the carry plus IMO to the contents of B-register. The result is saved in B-register.

(B) <-- (B) + C + IMO

Usage: The Carry ADD (CADD) instruction is used to add a number that is more than one byte wide (a number greater than 255). The CADD is used on the more significant bytes to propagate the carry bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: CADD B,=68

```
          R-REG = |0 0 1 1 1 0 1 1
              C = |               1
            IMO = |0 1 0 0 0 1 0 0
Result R-REG = |1 0 0 0 0 0 0 0
```

C = 0, SB = 1, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

CANCEL BLOCK

Format: CANB (CANB)

Time: 6.75 microseconds

Summary: The Cancel Block (CANB) instruction causes the current CCB to be reset to its initial address and range.

Current Address <-- Initial Address
Current Range <-- Initial Range

Usage: The CANB instruction is used to attempt recovery after an abort or special character detection.

Indicators:

None affected.

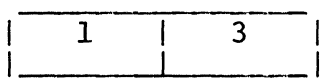
Example:

Abbreviations In This Instruction:

CCB = Communications Control Block
LCT = Line Control Table
MSB = Most Significant Bit

CANCEL CHARACTER

Format: CANC (CANC)



Time: TBD microseconds

Summary: The Cancel Character (CANC) instruction increments the current CCB range by one, and then decrements the current CCB address by one.

```
(W) <-- Current Range - Initial Range
            If Z = 0; then
Current Address <-- Current Address - 1
 Current Range  <-- Current Range + 1
```

Usage: The CANC is used in receive CCB to delete previous characters received. When special characters are detected, executing a CANC in place of Store (ST) will discard the special character and overwrite previous characters on the next ST.

NOTE

If the initial CCB range equals the current CCB range, the instruction is executed as NOP.

Indicators:

Z - Set if CCB initial range minus CCB current range equals Zero; otherwise reset.

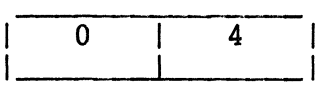
Example:

Abbreviations In This Instruction:

CCB = Communications Control Block
LCT = Line Control Table
MSB = Most Significant Bit
NOP = No Operation
Z = Zero Indicator

CALCULATE BLOCK CHECK

Format: CCH (CCH)



Time: 3.9-9.5 microseconds

Summary: The Calculate Block Check (CCH) instruction performs a cycle redundancy check on the contents of R-Register, and updates the cycle redundancy check residue.

(CRC) <-- (CRC) + (R)

Usage: The CCH is used after an Input (IN) one instruction when it is determined that the character is part of the message.

Indicators:

None affected.

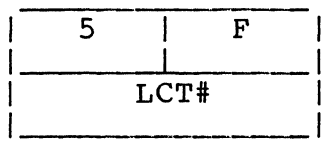
Example:

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit

CLEAR CONTENTS OF LCT

Format: CL lct (CLL)



Time: 0.75 microseconds

Summary: Clear to Zero the contents of LCT.

(LCT) <-- 0

Usage: The Clear (CL) instruction is used to create a value of Zero.

Indicators:

None affected.

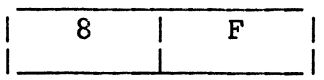
Example:

Abbreviations In This Instruction:

LCT = Line Control Table

CLEAR CONTENTS OF B-REFERENCED LCT

Format: CL B (CLN)



Time: 0.75 microseconds

Summary: Clear to Zero the contents of LCT pointed to by B-register.

((B)) <-- 0

Usage: The Clear (CL) instruction is used to create a value of Zero.

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

LCT = Line Control Table

CLEAR CONTENTS OF R-REGISTER

Format: CL =R (CLR)

9	F

Time: 0.38 microseconds

Summary: Clear to Zero the contents of R-register.

(R) <-- 0

Usage: The Clear (CL) instruction is used to create a value of Zero.

Indicators:

None affected.

Example:

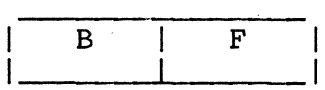
Abbreviations In This Instruction:

None

CL

CLEAR CONTENTS OF B-REGISTER

Format: CL =B (CLB)



Time: 0.38 microseconds

Summary: Clear to Zero the contents of B-register.

(B) <-- 0

Usage: The Clear (CL) instruction is used to create a value of Zero.

Indicators:

None affected.

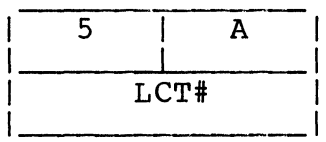
Example:

Abbreviations In This Instruction:

None

DECIMAL ADD R-REGISTER WITH LCT

Format: DADD R,lct (DADD)



Time: 5.88 microseconds

Summary: ADD the packed binary coded decimal value in R-register to the unsigned decimal contents of LCT. The result is saved in R-register.

(R) <-- (R) + (LCT)

Usage: The Decimal ADD (DADD) instruction is used to add two unsigned decimal numbers.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: DADD R,200

```
      REG-R = |1 0 0 1 0 1 0 0
  LCT 200 = |0 0 0 0 0 1 1 0
Result REG-R = |0 0 0 0 0 1 0 0
```

C = 1, SB = 0, Z = 1

CAUTION

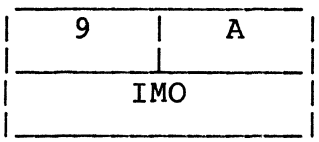
Validity of data is not checked by the assembler for this instruction.

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DECIMAL ADD R-REGISTER WITH IMO

Format: DADD R,=imo (DADDI)



Time: 5.75 microseconds

Summary: ADD the packed binary coded decimal value in R-register, to the unsigned decimal IMO. The result is saved in R-register.

(R) <-- (R) + IMO

Usage: The Decimal ADD (DADD) instruction is used to add two unsigned decimal numbers.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: DADD R,=imo

```
        REG-R = |1 0 0 1 0 1 0 0
    LCT 200 = |0 0 0 0 0 1 1 0
Result REG-R = |0 0 0 0 0 1 0 0
```

C = 1, SB = 0, Z = 1

CAUTION

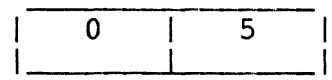
Validity of data is not checked by the assembler for this instruction.

Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DECREMENT CONTENTS OF R-REGISTER

Format: DEC =R (DEC)



Time: 0.38 microseconds

Summary: Decrement by one the contents of R-register.

(R) <-- (R) - 1

Usage: The Decrement (DEC) instruction is used to decrease a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

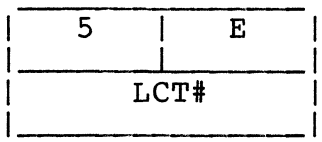
Example:

Abbreviations In This Instruction:

C = Carry Bit
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DECREMENT CONTENTS OF LCT

Format: DEC lct (DECL)



Time: 0.38 microseconds

Summary: Decrement by one the contents of LCT.

(LCT) <-- (LCT) - 1

Usage: The Decrement (DEC) instruction is used to decrease a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DECREMENT CONTENTS OF B-REFERENCED LCT

Format: DEC B (DECN)

8	E

Time: 1.0 microseconds

Summary: Decrement by one the contents of LCT pointed to by B-register.

((B)) <-- ((B)) - 1

Usage: The Decrement (DEC) instruction is used to decrease a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

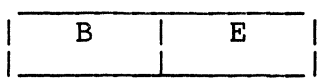
Example:

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DECREMENT CONTENTS OF B-REGISTER

Format: DEC =B (DECB)



Time: 0.38 microseconds

Summary: Decrement by one the contents of B-register.

(B) <-- (B) - 1

Usage: The Decrement (DEC) instruction is used to decrease a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

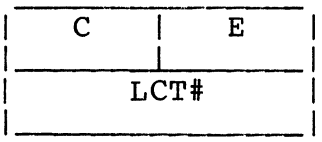
Example:

Abbreviations In This Instruction:

C = Carry Bit
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DECREMENT LOCAL STORE LOCATION (EXTENDED)

Format: DEC *lct (DECX)



Time: 7.0 microseconds

Summary: Decrement by one the contents of the local store location whose address is defined by the conjunction of the contents of LCT and the contents of B-register.

((LCT):(B)) <-- ((LCT):(B)) - 1

NOTE

Prior to the execution of this format of the DEC instruction, it is necessary to execute an output configuration A (FC-11) with a bit four set. Otherwize it will be treated as an illegal instruction.

Usage: The Decrement (DEC) instruction is used to decrease a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

DEQ DEQUEUE

Format: DEQ (PULLF)

Time: 1.2 microseconds

Summary: Dequeue the first item on the channel queue (FIFO) into R-register.

```
(R) <-- ((SOQ))
(SOQ) <--  (SOQ) + 1
```

Usage: The Dequeue (DEQ) instruction is used to retrieve data which has previously been Enqueued (ENQ) on the channel queue. Each channel queue has a maximum size of 16 items, wherein the address of the base item must be modulo-16. The queue is filled in the direction of an increasing LCT number.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: Given a queue with the two items X'7F' preceding X'00', the following instructions will alter the queue as shown.

```
DEQ
ST,
DEQ
ST,
```

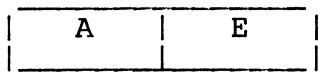
```
Queue          Queue          Queue
Before         After          After
1st DEQ        1st DEQ        2nd DEQ

SOQ->7F           ..             ..
     00        SOQ->00           ..
     ..<-EOQ        ..<-EOQ   SOQ->..<-EOQ
```

Abbreviations In This Instruction:

EOQ = End Of Queue
FIFO = First In First Out
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
SOQ = Start Of Queue
Z = Zero Indicator.

ENQUEUE

Format: ENQ (PUSHF)



Time: 1.2 microseconds

Summary: Enqueue the contents of R-register on the channel queue (FIFO).

```
((EOQ)) <-- (R)
 (EOQ)  <-- (EOQ) + 1
```

Usage: The Enqueue (ENQ) instruction is used to store data in the channel queue. Each channel queue has a maximum size of 16 items, wherein the address of the base item must be modulo-16. The queue is filled in the direction of an increasing LCT number.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: Given an empty queue, the following instructions will alter the queue as shown.

```
        LD  R,=X'7F'
        ENQ
        LD  R,=X'00'
        ENQ
```

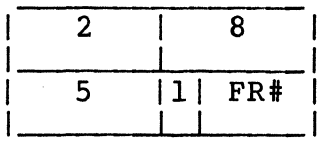
```
Queue                Queue                Queue
Before               After                After
1st ENQ              1st ENQ              2nd ENQ

SOQ->..<-EOQ         SOQ->7F              SOQ->7F
     ..                   ..<-EOQ              00
     ..                   ..                   ..<-EOQ
```

Abbreviations In This Instruction:

EOQ = End Of Queue
FIFO = First In First Out
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
SOQ = Start Of Queue
Z = Zero Indicator

INPUT FLAP REGISTER

Format: FIN fr (FIN)



Time: 2.6 microseconds

Summary: Input the contents of FLAP register to R-register.

(R) <-- (FR)

Usage: The FIN instruction is used to access specific FLAP registers 0, 1, 5, 6, and 7.

NOTE

FR1 should only be accessed by the receive channel.

Indicators:

SB - Set if MSB is one; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

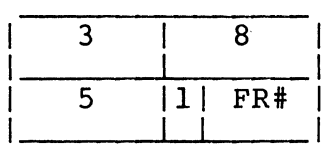
Example:

Abbreviations In This Instruction:

FLAP = Flexible Line Adapter Package
FR = FLAP Register
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OUTPUT FLAP REGISTER

Format: FOUT fr (FOUT)



Time: 4.4 microseconds

Summary: Output the contents of R-register to the FLAP register.

(FR) <-- (R)

Usage: The FOUT instruction is used to access specific FLAP registers 2, 3, and 4.

NOTE

FR3 should only be accessed by the transmit channel.

Indicators:

None affected.

Example:

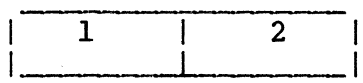
Abbreviations In This Instruction:

FLAP = Flexible Line Adapter Package
FR = FLAP Register
MSB = Most Significant Bit

GIVE

GIVE A CCB BACK TO CPU

Format: GIVE (GIVE)



Time: TBD microseconds

Summary: The GIVE instruction is used to return a completed CCB to the CPU and set the CCB status complete bit field to one. The CCB residual range is loaded in the range field. Bits 11 through 15 of the CCB control field are then loaded from the software defined LCT, pointed to by B-register. If the CCB control field has bit 8 set, an interrupt is sent to the CPU.

Post Current CCB - Extended

Usage: The GIVE instruction is used after a GNB instruction to return completed CCBs to the CPU. The CPU can then, by the execution of the input next status order, release the CCB for reuse by the channel.

NOTE

(Extended Mode Only)

Prior to the execution of the GIVE instruction, configuration A (FC-11) must be executed to set the channel in Extended mode (Bit 0=1). If not in Extended mode, this instruction will be executed as NOP.

Indicators:

Z - Set if the GNB instruction has not been executed prior to the GIVE instruction; otherwise reset.

Example:

Abbreviations In This Instruction:

CCB = Communications Control Blocks
GNB = Get Next Block
MSB = Most Significant Bit
NOP = No Operation
Z = Zero Indicator

GET NEXT BLOCK (CCB)

Format: GNB (GNB)

Time: 10.0 microseconds

Summary: The GNB instruction is used to terminate the current CCB, and return to the CPU. The receive/transmit status LCT 16,17/48,49 is loaded into the CCB status. In the Compatibility mode, the Status Complete bit in the CCB status field is set to one. If the CCB control field has bit 8 set, an interrupt is sent to the CPU. In the Extended mode, the status complete is not set and the interrupt is not sent. The initial CCB range is restored in the CCB range field. The next CCB is fetched, the internal valid last block indicators are set to CCB control field values, and the last character indicator is reset. After performing a GNB the Valid indicator should always be tested before using the new CCB.

Post Current CCB - Compatible

Usage: The GNB instruction is used to terminate and return CCB to the CPU. The CPU can then, by the execution of the input next status, order release of the CCB for reuse by the channel.

Indicators:

V - Set if CCB control field bit is set.
LB - Set if CCB control field bit is set.
LC - Reset.

Example:

Abbreviations In This Instruction:

CCB = Communications Control Blocks
LB = Last Block Indicator
LC = Last Character Indicator
LCT = Line Control Table
MSB = Most Significant Bit
V = Valid Indicator

INPUT ADAPTER STATUS TO R-REGISTER

Format: IAS (AST)

2	9

Time: 3.4 microseconds

Summary: Input Adapter Status into R-register.

(R) <-- CA Status

Bit Definitions:

Bit 0-3 - RFU
4 - Adapter Ready
5 - 0
6 - Receive overrun
7 - Transmit underrun

Usage: The Input Adapter Status (IAS) instruction is used to test adapter status for receive overrun or transmit underrun. The IAS instruction should be used for its performance advantage over the IN LR5 instruction.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

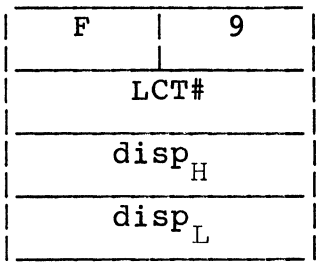
Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
RFU = Reserved for Future Use
Z = Zero Indicator

INITIALIZE LCT POINTER

Format: ILP lct,label (ILP)

Time: 1.2 microseconds



Summary: Initialize LCT with an IMA pointer to the location defined by label.

(LCT) <-- (P) + Disp

NOTE

In this case LCT identifies a 16-bit value.

Usage: The Initialize LCT Pointer (ILP) instruction is used to dynamically create IMA pointers for TLU, JUMP, and JS usage.

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

disp = Displacement
LCT = Line Control Table

INPUT LR TO R-REGISTER

Format: IN fr (IN)

Time: TBD microseconds

Summary: Input the contents of LR in the communications adapter into the R-register.

(R) <-- (LR)

Usage: The IN instruction is used to access specific adapter line registers 0, 1, and 5.. The bit definition is adapter specific.

NOTE

IN 7 is a special case using the NBHSA (Broadband Adapter) in HDLC mode. IN 7 inputs two characters from the receive FIFO. The first is the data character loaded into the R-register. The second is the status character loaded into LCT 11. All indicators are set on the results of the status character (LCT 11).

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

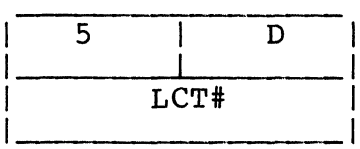
Abbreviations In This Instruction:

FIFO = First In First Out
LCT = Line Control Table
LR = Line Register
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

INCREMENT CONTENTS OF LCT

Format: INC lct (INCL)

Time: 0.38 microseconds



Summary: Increment by one the contents of LCT.

(LCT) <-- (LCT) + 1

Usage: The Increment (INC) instruction is used to increase a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

INCREMENT CONTENTS OF B-REFERENCED LCT

Format: INC B (INCN)

Time: 1.0 microseconds

Summary: Increment by one the contents of LCT pointed to by B-register.

((B)) <-- ((B)) + 1

Usage: The Increment (INC) instruction is used to increase a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

INCREMENT CONTENTS OF R-REGISTER

Format: INC =R (INC)

Time: 0.38 microseconds

Summary: Increment by one the contents of R-register.

(R) <-- (R) + 1

Usage: The Increment (INC) instruction is used to increase a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Bit
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

INCREMENT CONTENTS OF B-REGISTER

Format: INC =B (INCB)

Time: 0.38 microseconds

Summary: Increment by one the contents of B-register.

(B) <-- (B) + 1

Usage: The Increment (INC) instruction is used to increase a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

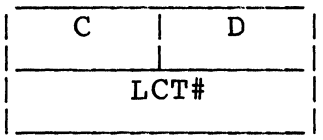
Example:

Abbreviations In This Instruction:

C = Carry Bit
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator.

INCREMENT LOCAL STORE LOCATION (EXTENDED)

Format: INC *lct (INCX)



Time: 7.0 microseconds

Summary: Increment by one the contents of the local store location whose address is resolved by the conjunction of the contents of LCT and the contents of B-register.

((LCT):(B)) <-- ((LCT):(B)) + 1

NOTE

Prior to the execution of this format of the INC instruction, it is necessary to execute an output configuration A (FC-11) with bit 4 set. Otherwize it will be treated as an illegal instruction.

Usage: The Increment (INC) instruction is used to increase a value by one.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Bit
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

INTR

INTERRUPT CPU

Format: INTR (INTR)

Time: TBD microseconds

Summary: Interrupt the CPU unless the interrupt level is zero.

Interrupt CPU

Usage: The Interrupt (INTR) instruction is used to interrupt the main memory program when a CCP detected condition occurs.

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

CCP = Channel Control Program
MSB = Most Significant Bit

INITIALIZE (ALL CHANNELS)

Format: INZ (INZ)

Time: TBD microseconds

Summary: The Initialize (INZ) instruction causes a soft initialize of all channels. It cleares adapters, LCT 8 (Dataset Scan Control), LCT 9 (Receive Firmware Control), LCT 40 (Dataset Scan Control), LCT 41 (Transmit Firmware Control), and clears all internal registers.

Stop I/O

Usage: The INZ instruction is used as a debug aid. This instruction is used after an interrupt instruction to cause all processor lines to cease operation. After an INZ command, meaningfull dumps can be taken to indicate the processor condition at the time of the occurrence.

Indicators:

None affected.

Example:

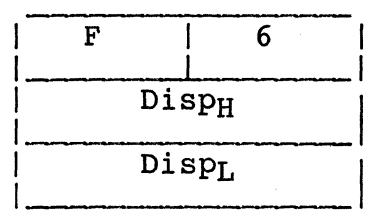
Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit

JS

JUMP TO SUBROUTINE VIA DISPLACEMENT

Format: JS label (JSR)



Time: 6.2 microseconds

Summary: Jump to the subroutine which is identified by label after saving the return address pointer on the channel stack.

```
 (SP)  <-- (SP) - 1
((SP)) <-- (P)  + 1
 (SP)  <-- (SP) - 1
  (P)  <-- (P) + Disp
```

Usage: This format of the JS instruction is used to transfer control to a subroutine within the range of a long Displacement, where $-32768 \leq Disp \leq +32767$.

NOTE

The Displacement is represented as a signed 16-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected

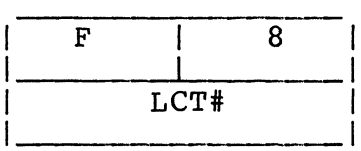
Example:

Abbreviations In This Instruction:

SP = Stack Pointer

JUMP TO SUBROUTINE VIA VECTOR

Format: JS *lct (JSRV)



Time: TBD microseconds

Summary: Jump to the subroutine which is identified by the IMA pointer contained in LCT after saving the return address pointer on the channel stack.

```
  (SP)  <-- (SP) - 1
((SP))  <-- (P)  + 1
  (SP)  <-- (SP) - 1
   (P)  <-- (LCT)
```

NOTE

In this case LCT identifies a 16-bit value.

Usage: This format of the JS instruction is used to transfer control to a subroutine within the range of an IMA pointer, where $0 \leq IMA \leq 65535$.

Indicators:

None affected.

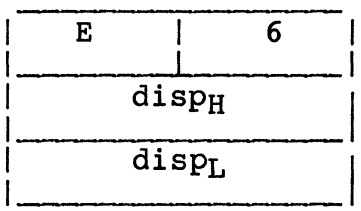
Example:

Abbreviations In This Instruction:

IMA = Immediate Memory Address
LCT = Line Control Table
SP = Stack Pointer

JUMP VIA DISPLACEMENT

Format: JUMP label (JUMP)



Time: 3.0 microseconds

Summary: Jump to the routine which is identified by label.

(P) <-- (P) + Disp

Usage: This format of the JUMP instruction is used to transfer control to a routine within the range of a long Displacement, where $-32768 \leq Disp \leq +32767$.

NOTE

The Displacement is represented as a signed 16-bit field which is the 2's complement difference (label byte address - operand byte address).

Indicators:

None affected.

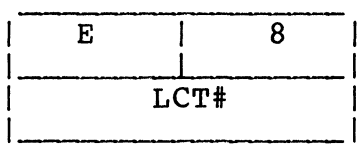
Example:

Abbreviations In This Instruction:

None

JUMP VIA VECTOR

Format: JUMP *lct (JV)



Time: TBD microseconds

Summary: Jump to the routine which is identified by the IMA pointer contained in LCT.

(P) <-- (LCT)

NOTE

In this case LCT identifies a 16-bit value.

Usage: This format of the JUMP instruction is used to transfer control to a routine within the range of an IMA pointer, where $0 \leq IMA \leq 65535$.

Indicators:

None affected.

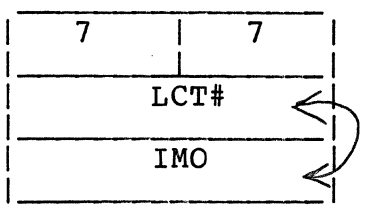
Example:

Abbreviations In This Instruction:

IMA = Immediate Memory Address
LCT = Line Control Table

LOAD BIT INDICATOR FROM LCT WITH IMO

Format: LB lct,=imo (LBL)



Time: 1.2 microseconds

Summary: AND IMO with the contents of the LCT and set indicators. The result is not saved.

(W) <-- (LCT) ^ IMO

Usage: The Load Bit indicator (LB) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: LB 60,=X'96'

```
      IMO = |1 0 0 1 0 1 1 0
   LCT 60 = |1 1 0 0 0 1 0 1
Result W = |1 0 0 0 0 1 0 0
```

SB = 1, Z = 0

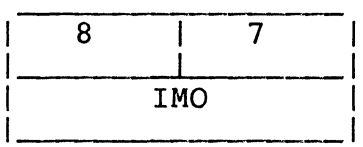
Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM B-REFERENCED LCT WITH IMO

Format: LB B,=imo (LBN)

Time: 0.25 microseconds



Summary: AND IMO with the contents of the LCT pointed to by B-register and set indicators. The result is not saved.

(W) <-- ((B)) ^ IMO

Usage: The Load Bit indicator (LB) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: LB B,=X'A5'

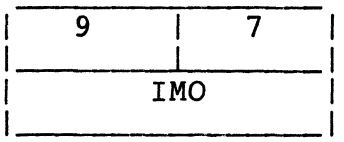
```
                    IMO = |1 0 1 0 0 1 0 1
B-REG=10        LCT 16 = |1 1 0 0 1 0 1 0
                Result W = |1 0 0 0 0 0 0 0

                SB = 1,   Z = 0
```

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM R-REGISTER WITH IMO

Format: LB =R,=imo (LBR)



Time: 0.75 microseconds

Summary: AND IMO with the contents of R-register and set indicators. The result is not saved.

(W) <-- (R) ^ IMO

Usage: The Load Bit indicator (LB) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: LB =R,=X'96'

```
     IMO = |1 0 0 1 0 1 1 0
   R-REG = |1 1 0 0 0 1 0 1
Result W = |1 0 0 0 0 1 0 0
```

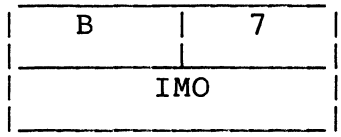
SB = 1, Z = 0

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM B-REGISTER WITH IMO

Format: LB =B,=imo (LBB)



Time: 0.75 microseconds

Summary: AND IMO with the contents of B-register and set indicators. The result is not saved.

(W) <-- (B) ^ IMO

Usage: The Load Bit indicator (LB) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: LB =B,=X'96'

```
      IMO = |1 0 0 1 0 1 1 0
  B-REG = |1 1 0 0 0 1 0 1
Result W = |1 0 0 0 0 1 0 0
```

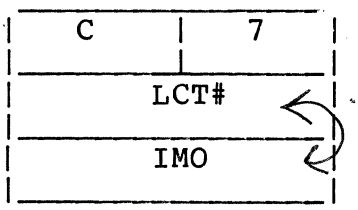
SB = 1, Z = 0

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM LOCAL STORE WITH IMO

Format: LB *lct,=imo (LBX)



Time: 6.1 microseconds

Summary: AND IMO with the contents of the local store location whose address is defined by the conjunction of the contents of LCT and the contents of B-register and set indicators. The result is not saved.

(W) <-- ((LCT):(B)) ^ IMO

NOTE

Prior to the execution of this format of the LB instruction, it is necessary to execute an output configuration A (FC-11) with bit 4 set. Otherwise it will be treated as an illegal instruction.

Usage: The Load Bit indicator (LB) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: LB *60,=X'96'

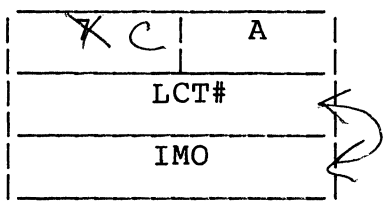
```
                    IMO = |1 0 0 1 0 1 1 0
Local Store Location = |1 1 0 0 0 1 0 1
            Result W = |1 0 0 0 0 1 0 0

            SB = 1,   Z = 0
```

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM LCT WITH IMO AND SET FALSE

Format: LBF lct,=imo (LBFL)



Time: 2.3 microseconds

Summary: AND IMO with the contents of the LCT and set indicators. The result is not saved. Then AND the 1's complement of IMO with the contents of LCT. The result is saved in LCT.

(W) <-- (LCT) ^ IMO
(LCT) <-- (LCT) ^ ~IMO

Usage: The Load Bit indicator and set False (LBF) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

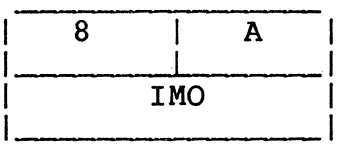
Example:

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM B-REFERENCED LCT WITH IMO AND SET FALSE

Format: LBF B,=imo (LBFN)



Time: 2.3 microseconds

Summary: AND IMO with the contents of the LCT pointed to by B-register and set indicators. The result is not saved. Then AND the 1's complement of IMO with the contents of LCT pointed to by B-register. The result is saved in LCT.

$$(W) \leftarrow\!- ((B)) \wedge IMO$$
$$((B)) \leftarrow\!- ((B)) \wedge \overline{IMO}$$

Usage: The Load Bit indicator and set False (LBF) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

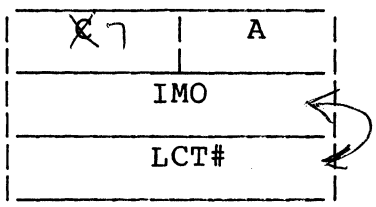
Example:

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM LOCAL STORE WITH IMO AND SET FALSE

Format: LBF *lct,=imo (LBFX)



Time: 7.9 microseconds

Summary: AND IMO with the contents of the local store location whose address is defined by the conjunction of the contents of LCT and the contents of B-register and set indicators. The result is not saved. Then AND the 1's complement of IMO with the contents of the same local store location. The result is saved in local store.

```
          (W) <-- ((LCT):(B)) ^ IMO
((LCT):(B)) <-- ((LCT):(B)) ^ IMO
```

NOTE

Prior to the execution of this format of the LBF instruction, it is necessary to execute an output configuration A (FC-11) with bit 4 set. Otherwise it will be treated as an illegal instruction.

Usage: The Load Bit indicator and set False (LBF) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

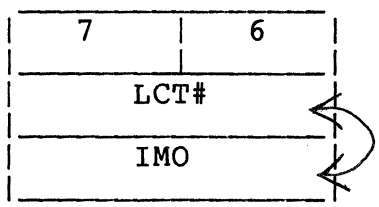
Example:

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM LCT WITH IMO AND SET TRUE

Format: LBT lct,=imo (LBTL)



Time: 2.3 microseconds

Summary: AND IMO with the contents of the LCT and set indicators. The result is not saved. Then OR the 1's complement of IMO with the contents of LCT. The result is saved in LCT.

$$(W) \leftarrow (LCT) \wedge \overline{IMO}$$
$$(LCT) \leftarrow (LCT) \vee IMO$$

Usage: The Load Bit indicator and set True (LBT) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

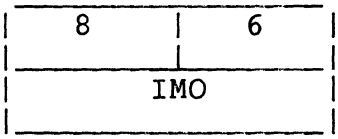
Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM B-REFERENCED LCT WITH IMO AND SET TRUE

Format: LBT B,=imo (LBTN)

Time: 2.3 microseconds



Summary: AND IMO with the contents of the LCT pointed to by B-register and set indicators. The result is not saved. Then OR the 1's complement of IMO with the contents of LCT pointed to by B-register. The result is saved in LCT.

(W) <-- ((B)) ^ IMO
((B)) <-- ((B)) v IMO

Usage: The Load Bit indicator and set True (LBT) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: LBT B,=X'96'

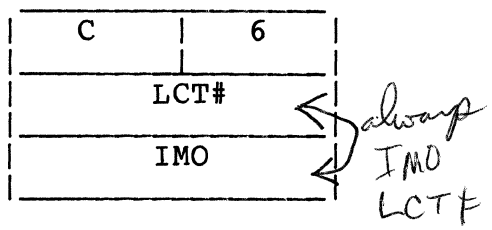
```
                IMO = |1 0 0 1 0 1 1 0
B-REG = 20 LCT 32 = |1 1 0 0 0 1 0 1
      Result W = |1 0 0 0 0 1 0 0
           LCT = |1 1 0 1 0 1 1 0

SB = 1,   Z = 0
```

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD BIT INDICATOR FROM LOCAL STORE WITH IMO AND SET TRUE

Format: LBT *lct,=imo (LBTX)



Time: 7.9 microseconds

Summary: AND IMO with the contents of the local store location whose address is defined by the conjunction of the contents of LCT and the contents of B-register and set indicators. The result is not saved. Then OR the 1's complement of IMO with the contents of the same local store location. The result is saved in local store.

$$(W) \leftarrow (((LCT):(B)) \wedge IMO$$
$$((LCT):(B)) \leftarrow (((LCT):(B)) \vee IMO$$

NOTE

Prior to the execution of this format of the LBT instruction, it is necessary to execute an output configuration A (FC-11) with bit 4 set. Otherwise it will be treated as an illegal instruction.

Usage: The Load Bit indicator and set True (LBT) instruction is used to test the state of a particular bit or group of bits.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD R-REGISTER FROM MAIN MEMORY

Format: LD, (LD)

Time: 0.75 microseconds

Summary: Load R-register with the contents of the main memory byte pointed to by the active CCB.

(R) <-- (CDB)

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

LC - Set if Current Range - 1 = Zero; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

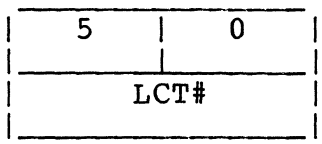
CCB = Communications Control Block
CDB = Communications Data Block
LC = Last Character Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LD

LOAD R-REGISTER FROM LCT

Format: LD R,lct (LD)

Time: 0.75 microseconds



Summary: Load R-register with the contents of LCT.

(R) <-- (LCT)

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

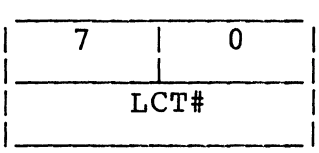
Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD B-REGISTER FROM LCT

Format: LD B,lct (LDB)

Time: 0.75 microseconds



Summary: Load B-register with the contents of LCT.

(B) <-- (LCT)

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

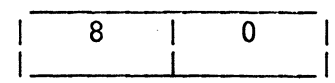
Example:

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD R-REGISTER FROM B-REFERENCED LCT

Format: LD R,B (LDRN)



Time: 0.75 microseconds

Summary: Load R-register with the contents of the LCT pointed to by B-register.

(R) <-- ((B))

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

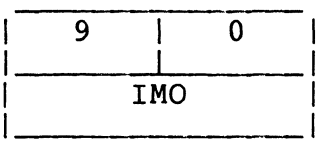
Example:

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD R-REGISTER WITH IMO

Format: LD R,=imo (LD)



Time: 0.75 microseconds

Summary: Load R-register with IMO.

(R) <-- IMO

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

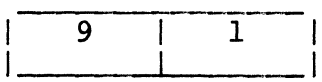
Example:

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD B-REGISTER FROM R-REGISTER

Format: LD B,=R (LDRB)



Time: 0.75 microseconds

Summary: Load B-register with the contents of R-register.

(B) <-- (R)

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

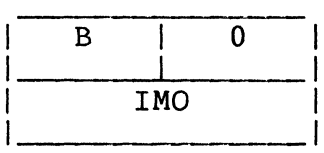
Example:

Abbreviations In This Instruction:

MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD B-REGISTER WITH IMO

Format: LD B,=imo (LDIB)



Time: 0.75 microseconds

Summary: Load B-register with IMO.

(B) <-- IMO

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

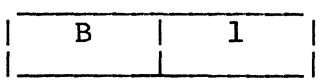
Example:

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD R-REGISTER FROM B-REGISTER

Format: LD R,=B (LDBR)



Time: 0.75 microseconds

Summary: Load R-register with the contents of B-register.

(R) <-- (B)

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

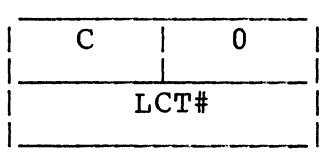
Example:

Abbreviations In This Instruction:

MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

LOAD R-REGISTER FROM LOCAL STORE (EXTENDED)

Format: LD R,*lct (LDX)



Time: 6.1 microseconds

Summary: Load R-register with the contents of the local store location whose address is defined by the conjunction of the contents of LCT and the contents of B-register.

(R) <-- ((LCT):(B))

NOTE

Prior to the execution of this format of the LD instruction, it is necessary to execute an output configuration A (FC-11) with bit 4 set. Otherwize it will be treated as an illegal instruction.

Usage: The Load (LD) instruction is used to update the contents of a register.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

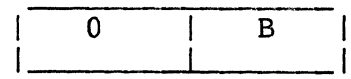
Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

MC

MASTER CLEAR

Format: MC (MC)



Time: TBD microseconds

Summary: Master Clear causes the Processor to pull the BSMCLR line on the DPS 6/Level 6 Bus to a true state for the last 258 microseconds. This causes all units on the Bus to initialize and run their QLTs. The CPU must have some kind of local bootload source for a cold start of the system.

Usage: The Master Clear (MC) instruction can be used with the Processor transfer function to restart the system, if the timer is not reset in the assigned time.

NOTE

Prior to the execution of the MC instruction, it is necessary to execute an output configuration A (FC-11) with bit 8 set. Otherwise it will be treated as an illegal instruction.

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

QLT = Quality Logic Test

NEGATE R-REGISTER

Format: NEG (NEGR)

Time: 0.75 microseconds

Summary: Negate the contents of R-register and add one. The result is saved in R-register.

$$(R) \leftarrow\!- \overline{(R)} + 1$$

Usage: The Negate (NEG) instruction is used to generate the 2's complement of a value. NEG is useful in generating the absolute difference after a subtract when the C-bit equals zero.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

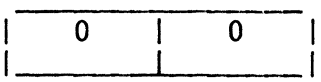
Abbreviations In This Instruction:

C = Carry Bit
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

NOP

NO OPERATION

Format: NOP (NOP)



Time: 0.38 microseconds

Summary: Perform No Operation.

(P) <-- (P) + 1

Usage: The No Operation (NOP) instruction can be used to reserve space within CCPs for patches or software probe points.

Indicators:

None Affected.

Example:

Abbreviations In This Instruction:

None

OUTPUT ADAPTER CONTROL

Format: OAC (ACTL)

Time: TBD microseconds

Summary: The Output Adapter Control (OAC) instruction moves the contents of R-register into the adapter control register.

CA Control <-- R

Bit Definitions:

Bit 0-4 - Must be zero
5 - Test
6 - Receive on
7 - Transmit on

Usage: The OAC instruction is used to change the state of the Test, Receive on, and Transmit on without affecting the data set control.

Indicators:

None affected.

Example:

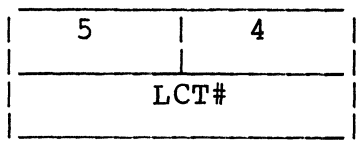
Abbreviations In This Instruction:

None

OR R-REGISTER WITH LCT

Format: OR R,lct (OR)

Time: 0.75 microseconds



Summary: Logically OR the contents of LCT with the contents of R-register. The result is saved in R-register.

(R) <-- (R) v (LCT)

Usage: The inclusive OR (OR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: OR R,63

```
        R-REG = |0 0 1 1 1 0 1 0
      LCT 63  = |0 0 1 1 0 1 1 0
Result R-REG  = |0 0 0 0 1 1 0 0
```

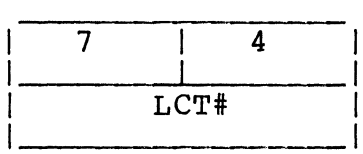
SB = 0, Z = 0

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OR B-REGISTER WITH LCT

Format: OR B,lct (ORB)



Time: 0.75 microseconds

Summary: Logically OR the contents of LCT with the contents of B-register. The result is saved in B-register.

(B) <-- (B) v (LCT)

Usage: The inclusive OR (OR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: OR B,63

```
        B-REG = |0 0 1 1 1 0 1 0
      LCT 63 = |0 0 1 1 0 1 1 0
Result B-REG = |0 0 1 1 1 1 1 0
```

SB = 0, Z = 0

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OR R-REGISTER WITH B-REFERENCED LCT

Format: OR R,B (ORRN)

8	4

Time: 0.75 microseconds

Summary: Logically OR the contents of LCT pointed to by B-register with the contents of R-register. The result is saved in R-register.

(R) <-- (R) v ((B))

Usage: The inclusive OR (OR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: OR R,B

```
        R-REG = |0 0 1 1 1 0 1 0
B=3F  LCT 63 = |0 0 1 1 0 1 1 0
Result R-REG = |0 0 1 1 1 1 1 0
```

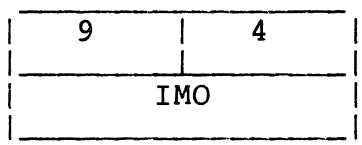
SB = 0, Z = 0

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OR R-REGISTER WITH IMO

Format: OR R,=imo (OR)



Time: 0.75 microseconds

Summary: Logically OR IMO with the contents of R-register. The result is saved in R-register.

(R) <-- (R) v IMO

Usage: The inclusive OR (OR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: OR R,=X'36'

```
        R-REG = |0 0 1 1 1 0 1 0
          IMO = |0 0 1 1 0 1 1 0
Result R-REG = |0 0 1 1 1 1 1 0
```

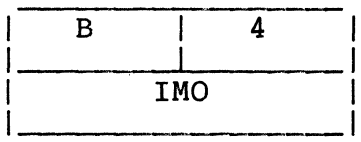
SB = 0, Z = 0

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OR B-REGISTER WITH IMO

Format: OR B,=imo (ORIB)



Time: 0.75 microseconds

Summary: Logically OR IMO with the contents of B-register. The result is saved in B-register.

(B) <-- (B) v IMO

Usage: The inclusive OR (OR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: OR B,=X'36'

```
        B-REG = |0 0 1 1 1 0 1 0
          IMO = |0 0 1 1 0 1 1 0
Result B-REG = |0 0 1 1 1 1 1 0
```

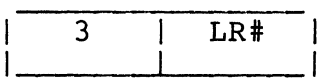
SB = 0, Z = 0

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OUTPUT LR FROM R-REGISTER

Format: OUT lr (OUT)



Time: TBD microseconds

Summary: Transfer the contents of R-register to the indicated line register.

(LR) <-- (R)

Usage: The OUT instruction is used to load the contents of the adapter specific line registers 0 - 7.

NOTE

OUT 7 is a special case using the NBHSA (Broadband Adapter) in HDLC mode. Two bytes are transfered to the transmit FIFO. The first is the data character loaded into the R-register. The second is the control character loaded from LCT 43.

Indicators:

None affected

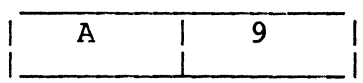
Example:

Abbreviations In This Instruction:

FIFO = First In First Out
LCT = Line Control Table
LR = Line Register

PULL VALUE FROM STACK INTO B-REGISTER

Format: PULL B (POPB)



Time: 1.1 microseconds

Summary: Pull the current entry from the channel stack (LIFO) into B-register.

```
(SP) <--  (SP) + 1
 (B) <-- ((SP))
```

Usage: The PULL instruction is used to restore the contents of a register from the channel stack.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

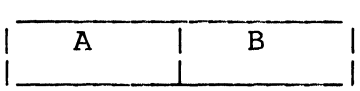
Example:

Abbreviations In This Instruction:

MSB = Most Significant Bit
SB = MSB Indicator
SP = Stack Pointer
Z = Zero Indicator

PULL VALUE FROM STACK INTO R-REGISTER

Format: PULL R (POPR)



Time: 1.1 microseconds

Summary: Pull the current entry from the channel stack (LIFO) into R-register.

```
(SP) <--  (SP) + 1
 (R) <-- ((SP))
```

Usage: The PULL instruction is used to restore the contents of a register from the channel stack.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

MSB = Most Significant Bit
SB = MSB Indicator
SP = Stack Pointer
Z = Zero Indicator

PUSH VALUE FROM B-REGISTER ONTO STACK

Format: PUSH B (PUSHB)

Time: 1.1 microseconds

Summary: PUSH the contents of B-register onto the channel stack (LIFO).

```
((SP)) <-- (B)
 (SP)  <-- (SP) - 1
```

Usage: The PUSH instruction is used to store a value on the channel stack.

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

SP = Stack Pointer

PUSH VALUE FROM R-REGISTER ONTO STACK

Format: PUSH R (PUSHR)

Time: 1.1 microseconds

Summary: PUSH the contents of R-register onto the channel stack (LIFO).

```
((SP)) <-- (R)
 (SP)  <-- (SP) - 1
```

Usage: The PUSH instruction is used to store a value on the channel stack.

Indicators:

None affected.

Example:

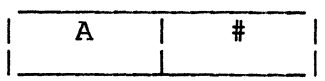
Abbreviations In This Instruction:

SP = Stack Pointer

RECV

RECEIVE DATA

Format: RECV # (RECV)



Time: 5.4/11.1/8.5/13.8 microseconds

Summary: The Receive data instruction transfers the contents of the receive channels line register one to the R-register. It then applies the parity and/or CRC characteristics indicated by the operand.

(R) <-- (LR1)

0 = No Parity, No CRC
1 = Parity, No CRC
2 = No Parity, CRC
3 = Parity, CRC

Usage: The RECV instruction is used to input data characters with parity and/or CRC calculations. If parity is checked, the parity bit is the leftmost bit of the character length, and includes all bits of the defined character length. If a parity error is detected, the firmware sets bit one (Data check error) of LCT 17. If both a parity check and a CRC is performed, we should note that the CRC characters do not include parity. Therefore, the checks should not be performed but should be input with a one instruction. If the firmware detects receive overrun, bit 2 (data services error) of LCT 16 is set to One.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

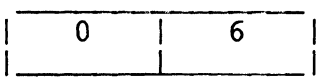
Example:

Abbreviations In This Instruction:

CRC = Cyclic Redundancy Check
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

RETURN FROM BRANCH SUBROUTINE

Format: RETB (RET)



Time: 1.5 microseconds

Summary: Return from branch subroutine.

(P) <-- (LCT 18)

Usage: The Return from Branch subroutine (RETB) instruction is used at the end of a Branched subroutine to return to the instruction following the subroutine call (BS instruction).

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

None

RETJ

RETURN FROM JUMP SUBROUTINE

Format: RETJ (RETJSR)

1	6

Time: 2.5 microseconds

Summary: Return from jump subroutine.

```
(SP) <--  (SP) + 1
 (P) <-- ((SP))
(SP) <--  (SP) + 1
```

Usage: The Return from Jump subroutine (RETJ) instruction is used at the end of a Jumped subroutine to return to the instruction following the subroutine call (JS instruction).

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

SP = Stack Pointer

RETURN HELD BLOCK

Format: RHB (RHB)

Time: TBD microseconds

Summary: The Return Held Block (RHB) instruction causes completed CCBs, that have not been released to the CPU by GIVE, to be placed at the top of the next CCB queue.

Usage: The Return Held Block (RHB) instruction is used to retransmit CCBs rejected by the remote station. Always test for a valid CCB indicator after execution of this instruction.

NOTE

Prior to the execution of the RHB instruction, it is necessary to execute an output configuration A (FC-11) with bit Zero set. Otherwise it will be treated as an illegal instruction.

Indicators:

V - Set if CCB Control Field Bit is set.
LB - Set if CCB Control Field Bit is set.
LC - Reset.

Example:

Abbreviations In This Instruction:

CCB = Communications Control Blocks
LB = Last Block Indicator
LC = Last Character Indicator
V = Valid Indicator

SCF

SET CARRY FALSE

Format: SCF (SCF)

Time: 1.2 microseconds

Summary: Set the Carry indicator false.

(C) <-- 0

Usage: The Set Carry False (SCF) instruction is used to reset the Carry indicator.

Indicators:

C - Reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator

SHIFT CLOSED LEFT R-REGISTER

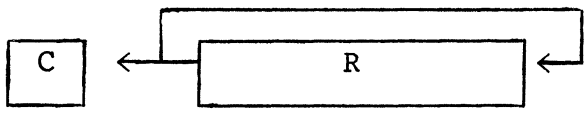
Format: SCL R (SCLR)

9	B

Time: 1.3 microseconds

Summary: Shift left one bit position the contents of R-register. Bit Zero moves into Carry indicator and bit position 7.

(R) <-- (R) . 2 + C



Usage: The Shift Closed Left (SCL) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SCL

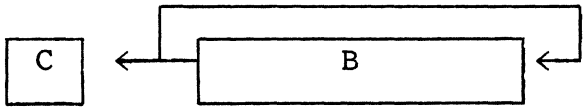
SHIFT CLOSED LEFT B-REGISTER

Format: SCL B (SCLB)

Time: 1.3 microseconds

Summary: Shift left one bit position the contents of B-register. Bit Zero moves into Carry indicator and bit position 7.

(B) <-- (B) . 2 + C



Usage: The Shift Closed Left (SCL) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

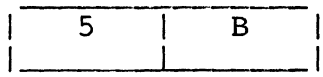
Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SHIFT CLOSED RIGHT R-REGISTER

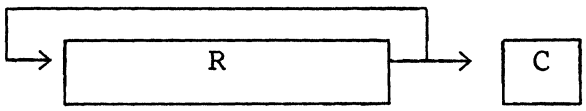
Format: SCR R (SCRR)



Time: 1.3 microseconds

Summary: Shift right one bit position the contents of R-register. Bit 7 moves into Carry indicator and bit position 0.

(R) <-- (R) / 2 + C . X'80'



Usage: The Shift Closed Right (SCR) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

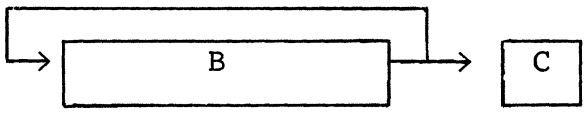
SHIFT CLOSED RIGHT B-REGISTER

Format: SCR B (SCRB)

Time: 1.3 microseconds

Summary: Shift right one bit position the contents of B-register. Bit 7 moves into Carry indicator and bit position 0.

(B) <-- (B) / 2 + C . X'80'



Usage: The Shift Closed Right (SCR) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SET CARRY TRUE

Format: SCT (SCT)

Time: 1.2 microseconds

Summary: Set the Carry indicator true.

C <-- 1

Usage: The Set Carry True (SCT) instruction is used to set the Carry indicator.

Indicators:

C - Set.

Example:

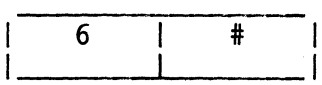
Abbreviations In This Instruction:

C = Carry Indicator

SEND

SEND DATA

Format: SEND # (SEND)



Time: 5.8/10.9/7.9/12.9 microseconds

Summary: The Send Data (SEND) instruction trannsfers the data character in R-Register to the transmit channel line register of the adapter. Apply the parity and/or cyclic redundancy check characteractics indicated by operand.

(LRl) <-- (R)

0 = No Parity, No CRC
1 = Parity, No CRC
2 = No Parity, CRC
3 = Parity, CRC

Usage: The SEND is used to output data character with parity and/or CRC calculations. If parity is to be generated, the leftmost bit of the character must be Zero. If both parity generation and cyclic redundancy checks are performed, correct parity is generated before the cyclic redundancy. Cyclic redundancy is performed on the data character including generated parity. (Note that parity generation must not be performed on the cyclic redundancy check characters). If the firmware detects transmit underrun, bit 2 (data services error) of LCT 48 is set to One.

Indicators:

None affected.

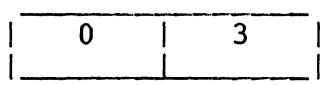
Example:

Abbreviations In This Instruction:

CRC = Cyclic Redundancy Check
IMO = Immediate Memory Operand
LCT = Line Control Table
MSB = Most Significant Bit

SEARCH FOR SYNCHRONIZATION

Format: SFS (SFS)



Time: 23.7 microseconds

Summary: The Search for Synchronization (SFS) instruction causes the current Communication Adapter (CA) channel to search for a synchronization character. SFS effects only the receiver control. The receive control is first set to off and then to on. Two sync characters are required for synchronization.

Usage: The SFS instruction is used to resynchronize the CA after a lost or false synchronization.

Indicators:

None affected.

Example:

Abbreviations In This Instruction:

CA = Communications Adapter
LCT = Line Control Table
LR = Line Register

SOL

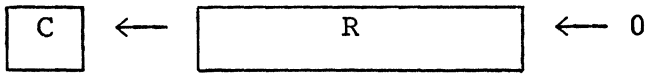
SHIFT OPEN LEFT R-REGISTER

Format: SOL R (SOLR)

Time: 0.38 microseconds

Summary: Shift left one bit position the contents of R-register. Bit 0 moves into Carry indicator. Zero moves into bit position 7.

(R) <-- (R) . 2



Usage: The Shift Open Left (SOL) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

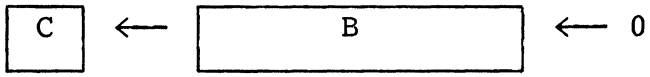
SHIFT OPEN LEFT B-REGISTER

Format: SOL B (SOLB)

Time: 0.38 microseconds

Summary: Shift left one bit position the contents of B-register. Bit Zero moves into Carry indicator. Zero moves into bit position 7.

(B) <-- (B) . 2



Usage: The Shift Open Left (SOL) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

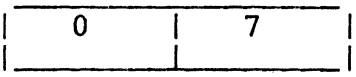
Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SOR

SHIFT OPEN RIGHT R-REGISTER

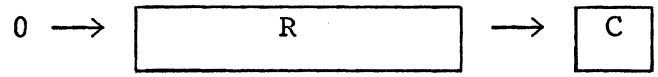
Format: SOR R (SR)



Time: 1.5 microseconds

Summary: Shift right one bit position the contents of R-register. Bit 7 moves into Carry indicator. Zero moves into bit position 0.

(R) <-- (R) / 2



Usage: The Shift Open Right (SOR) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SHIFT OPEN RIGHT B-REGISTER

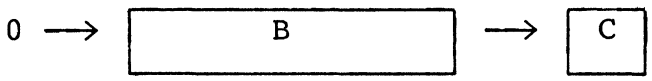
Format: SOR B (SORB)

B	C

Time: 1.3 microseconds

Summary: Shift right one bit position the contents of B-register. Bit 7 moves into Carry indicator. Zero moves into bit position 0.

(B) <-- (B) / 2



Usage: The Shift Open Right (SOR) instruction is used to isolate an individual bit.

Indicators:

C - Set if carry out; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

C = Carry Indicator
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

STORE R-REGISTER INTO MAIN MEMORY

Format: ST, (ST)

Time: 7.3 microseconds

Summary: Store the contents of R-register into the main memory byte pointed to by the current CCB.

(CDB) <-- (R)

Usage: The Store (ST) instruction is used to unload a register.

Indicators:

LC - Set when CCB range is decremented to zero.

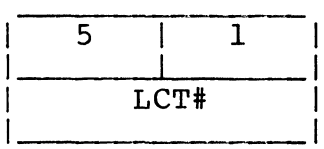
Example:

Abbreviations In This Instruction:

CCB = Communications Control Block
CDB = Communications Data Block
LC = Last Character Indicator

STORE R-REGISTER INTO LCT

Format: ST R,lct (ST)



Time: 0.75 microseconds

Summary: Store the contents of R-register into LCT.

(LCT) <-- (R)

Usage: The Store (ST) instruction is used to unload a register.

Indicators:

None Affected.

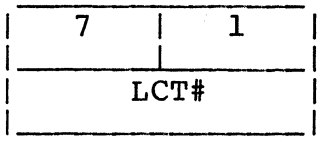
Example:

Abbreviations In This Instruction:

LCT = Line Control Table

STORE B-REGISTER INTO LCT

Format: ST B,lct (STB)



Time: 0.75 microseconds

Summary: Store the contents of B-register into LCT.

(LCT) <-- (B)

Usage: The Store (ST) instruction is used to unload a register.

Indicators:

None Affected.

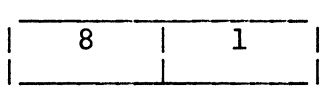
Example:

Abbreviations In This Instruction:

LCT = Line Control Table

STORE R-REGISTER INTO B-REFERENCED LCT

Format: ST R,B (STRN)



Time: TBD microseconds

Summary: Store the contents of R-register into the LCT pointed to by B-register.

((B)) <-- (R)

Usage: The Store (ST) instruction is used to unload a register.

Indicators:

None Affected.

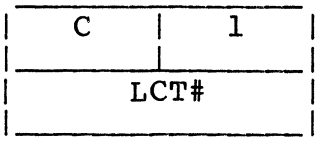
Example:

Abbreviations In This Instruction:

LCT = Line Control Table

STORE R-REGISTER INTO LOCAL STORE (EXTENDED)

Format: ST R,*lct (STX)



Time: 6.8 microseconds

Summary: Store the contents of R-register into the local store location whose address is defined by the conjunction of the contents of LCT and the contents of B-register.

((LCT):(B)) <-- (R)

NOTE

Prior to the execution of this format of the ST instruction, it is necessary to execute an output configuration A (FC-11) with bit 4 set. Otherwise it will be treated as an illegal instruction.

Usage: The Store (ST) instruction is used to unload a register.

Indicators:

None Affected.

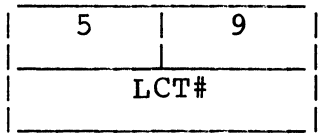
Example:

Abbreviations In This Instruction:

LCT = Line Control Table

SUBTRACT R-REGISTER WITH LCT

Format: SUB R,lct (SUB)



Time: 0.75 microseconds

Summary: Subtract the contents of LCT from the contents of R-register. The result is saved in R-register.

(R) <-- (R) - (LCT)

Usage: The Subtract (SUB) instruction is used to calculate the difference between two values.

Indicators:

C - Set to Zero if borrow required; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: SUB R,200

```
          R-REG = |0 0 1 0 0 1 0 1
      LCT 200 = |1 1 1 1 1 0 0 1
Results R-REG = |0 0 1 0 1 1 0 0
```

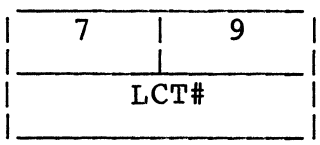
C = 0, SB= 0, Z = 0

Abbreviations In This Instruction:

C = Carry Indicator
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SUBTRACT B-REGISTER WITH LCT

Format: SUB B,lct (SUBB)



Time: 0.75 microseconds

Summary: Subtract the contents of LCT from the contents of B-register. The result is saved in B-register.

(B) <-- (B) - (LCT)

Usage: The Subtract (SUB) instruction is used to calculate the difference between two values.

Indicators:

C - Set to Zero if borrow required; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: SUB B,210

```
          B-REG = |1 1 0 0 0 0 1 1
      LCT 210 = |1 0 0 0 0 1 0 1
Results B-REG = |0 0 1 1 1 1 1 0
```

C = 1, SB= 0, Z = 0

Abbreviations In This Instruction:

C = Carry Indicator
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SUBTRACT R-REGISTER WITH B-REFERENCED LCT

Format: SUB R,B (SUBRN)

8	9

Time: TBD microseconds

Summary: Subtract the contents of LCT pointed to by B-register from the contents of R-register. The result is saved in R-register.

(R) <-- (R) - ((B))

Usage: The Subtract (SUB) instruction is used to calculate the difference between two values.

Indicators:

C - Set to Zero if borrow required; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: SUB R,B

```
                R-REG = |1 0 1 0 0 0 0 1
  B=C8        LCT 200 = |1 1 1 1 0 1 0 0
       Result R-REG = |1 1 0 0 1 1 0 1

          C = 0,  SB= 1,  Z = 0
```

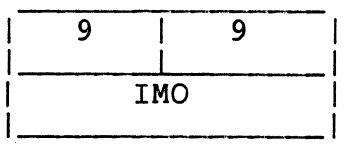
Before Execution	After Execution
R = A1	R = CD
B = C8	B = C8
LCT 200 = F4	LCT 200 = F4

Abbreviations In This Instruction:

C = Carry Indicator
LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SUBTRACT R-REGISTER WITH IMO

Format: SUB R,=imo (SBIR)



Time: 0.75 microseconds

Summary: Subtract IMO from the contents of R-register. The result is saved in R-register.

(R) <-- (R) - IMO

Usage: The Subtract (SUB) instruction is used to calculate the difference between two values.

Indicators:

C - Set to Zero if borrow required; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: SUB R,=X'69'

```
        R-REG = |1 0 0 1 0 0 0 1
          IMO = |0 1 1 0 1 0 0 1
Result R-REG = |0 0 1 0 1 0 0 0
```

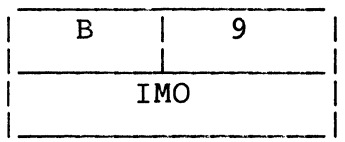
C = 1, SB= 1, Z = 0

Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

SUBTRACT B-REGISTER WITH IMO

Format: SUB B,=imo (SBIB)



Time: 0.75 microseconds

Summary: Subtract IMO from the contents of B-register. The result is saved in B-register.

(B) <-- (B) - IMO

Usage: The Subtract (SUB) instruction is used to calculate the difference between two values.

Indicators:

C - Set to Zero if borrow required; otherwise reset.
SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: SUB B,=X'A5'

```
        B-REG = |0 1 0 1 1 0 1 1
          IMO = |1 0 1 0 0 1 0 1
Result B-REG = |1 0 1 1 0 1 1 0
```

C = 0, SB= 0, Z = 1

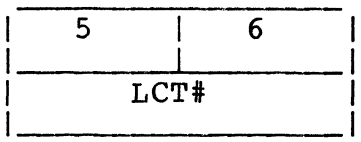
Abbreviations In This Instruction:

C = Carry Bit
IMO = Immediate Memory Operand
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

TABLE LOOK-UP

Format: TLU *lct (TLU)

Time: 6.0/7.5 microseconds



Summary: The Table Look-Up (TLU) instruction specifies the first LCT location of a pair of consecutive LCT work locations. These two locations contain a 12-Bit RAM address pointer to the begining of the TLU data table.

```
(W) <-- ((LCT0 + (R))
If SB=0;then (R) <-- (W)
             (P) <-- (P) + 1
        else (W) <-- (W) ^ X'7F'
             (W) <-- (W) . 2
             (P) <-- (P) + 3 + (W)
```

Usage: The TLU instruction is used to add the contents of the R-register previously loaded by the CCP. The resulting address now points to a specific byte in a TLU data table.

Indicators:

B - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example:

Abbreviations In This Instruction:

CCP = Channel Control Program
LCT = Line Control Table
MSB = Most Significant Bit

TEST QUEUE EMPTY

Format: TQE (TSTFE)

Time: 0.75 microseconds

Summary: Test channel queue empty by subtracting the EOQ pointer from the SOQ pointer to set indicators. The result is not saved.

(W) <-- (SOQ) - (EOQ)

Usage: The Test Queue Empty (TQE) instruction is used to test the state of the channel queue.

Indicators:

Z - Set if results equal Zero; otherwise reset.

Example:

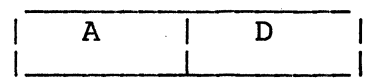
Abbreviations In This Instruction:

EOQ = End Of Queue
SOQ = Start Of Queue
Z = Zero Indicator

TQF

TEST QUEUE FULL

Format: TQF (TSTFF)



Time: 1.1 microseconds

Summary: Test channel queue full by subtracting the EOQ pointer from the SOQ pointer minus 15 to set indicators. The result is not saved.

(W) <-- (SOQ) - 15 - (EOQ)

Usage: The Test Queue Full (TQF) instruction is used to test the state of the channel queue.

Indicators:

Z - Set if result equal Zero; otherwise reset.

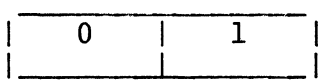
Example:

Abbreviations In This Instruction:

EOQ = End Of Queue
SOQ = Start Of Queue
Z = Zero Indicator

WAIT (SUSPEND CHANNEL)

Format: WAIT (WAIT)



Time: 13.25 microseconds

Summary: The WAIT instruction suspends channel activity and saves the content of the active CCP. A service request scan is initiated and service granted to the highest priority channel. Reactivation of the channel issuing the WAIT instruction restores the channel content and executes the next instruction.

Usage: The WAIT instruction is used after data character processing to allow other channels CCP execution time. It also suspends data block execution waiting for the CPU to start the next data block.

Indicators:

None affected.

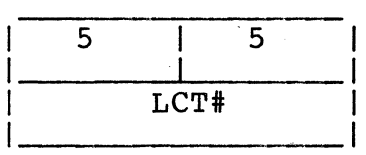
Example:

Abbreviations In This Instruction:

CCP = Channel Control Program
CPU = Central Processing Unit

EXCLUSIVE OR R-REGISTER WITH LCT

Format: XOR R,lct (XOR)



Time: 0.75 microseconds

Summary: Logically XOR the contents of LCT with the contents of R-register. The result is saved in R-register.

(R) <-- (R) + (LCT)

Usage: The exclusive OR (XOR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: XOR R,63

```
          R-REG = |0 0 1 1 1 0 1 0
       LCT 63 = |0 0 1 1 0 1 1 0
Result R-REG = |0 0 0 0 1 1 0 0
```

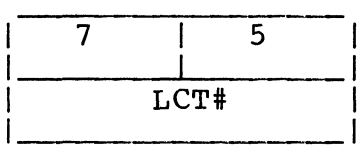
SB = 0, Z = 0

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

EXCLUSIVE OR B-REGISTER WITH LCT

Format: XOR B,lct (XORB)



Time: 0.75 microseconds

Summary: Logically XOR the contents of LCT with the contents of B-register. The result is saved in B-register.

(B) <-- (B) + (LCT)

Usage: The exclusive OR (XOR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: XOR B,63

```
        R-REG = |0 0 1 1 1 0 1 0
LCT 63 =        |0 0 1 1 0 1 1 0
Result R-REG = |0 0 0 0 1 1 0 0
```

SB = 0, Z = 0

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

EXCLUSIVE OR R-REGISTER WITH B-REFERENCED LCT

Format: XOR R,B (XORN)

8	5

Time: 0.75 microseconds

Summary: Logically XOR the contents of LCT pointed to by B-register with the contents of R-register. The result is saved in R-register.

(R) <-- (R) + ((B))

Usage: The exclusive OR (XOR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: XOR R,B

```
          R-REG = |0 0 1 1 1 0 1 0
     B-REG 63 = |0 0 1 1 0 1 1 0
Result R-REG = |0 0 0 0 1 1 0 0
```

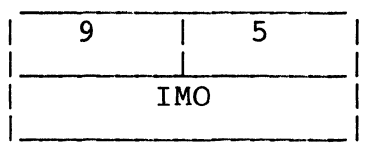
SB = 0, Z = 0

Abbreviations In This Instruction:

LCT = Line Control Table
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

EXCLUSIVE OR R-REGISTER WITH IMO

Format: XOR R,=imo (XOR)



Time: 0.75 microseconds

Summary: Logically XOR IMO with the contents of R-register. The result is saved in R-register.

(R) <-- (R) + IMO

Usage: The exclusive OR (XOR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: XOR R,=54

```
        R-REG = |0 0 1 1 1 0 1 0
          IMO = |0 0 1 1 0 1 1 0
Result R-REG = |0 0 0 0 1 1 0 0
```

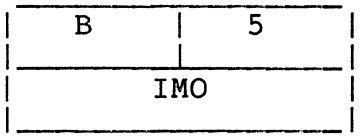
SB = 0, Z = 0

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

EXCLUSIVE OR B-REGISTER WITH IMO

Format: XOR B,=imo (XRIB)



Time: 0.75 microseconds

Summary: Logically XOR IMO with the contents of B-register. The result is saved in B-register.

(B) <-- (B) + IMO

Usage: The exclusive OR (XOR) instruction is used to modify a particular bit or group of bits. Decisions can be made by testing the indicators with branch instructions.

Indicators:

SB - Set if MSB of results is One; otherwise reset.
Z - Set if results equal Zero; otherwise reset.

Example: XOR B,=54

```
        B-REG = |0 0 1 1 1 0 1 0
          IMO = |0 0 1 1 0 1 1 0
Result B-REG = |0 0 0 0 1 1 0 0
```

SB = 0, Z = 0

Abbreviations In This Instruction:

IMO = Immediate Memory Operand
MSB = Most Significant Bit
SB = MSB Indicator
Z = Zero Indicator

OP CODE MAP

Table 3-4 contains the op code map for the NMLCP. Op codes that are not assigned in the table are treated as illegal instructions.

Table 3-4. Op Code Map

Bits 0-3	Format Type	Bits 4-7															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Generic	NOP	WAIT	GNB	SFS	CCH	DEC	RET	SOR	INTR	INZ	NEG	MC	SCT	SCF	---	---
1	Next Character	LD	ST	GIVE	CANC	CANB	RHB	RET	---	---	---	---	---	---	---	---	---
2	IN	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7	FIN	IAS	---	---	---	---	---	---
3	OUT	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7	FOUT	OAC	---	---	---	---	---	---
4	RFU	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
5	LCT+d Addressing	LD	ST	C	AND	OR	XOR	TLU	CADD	ADD	SUB	DADD	SCR	SOL	INC	DEC	CL
6	SEND (See Note)	NP/NC	NP/C	P/NC	P/C	---	---	---	---	---	---	---	---	---	---	---	---
7	LCT - B	LD	ST	C	AND	OR	XOR	LBT	LB	ADD	SUB	LBF	SCR	SOL	CADD	---	---
8	LCT - (B)	LD	ST	C	AND	OR	XOR	LBT	LB	ADD	SUB	LBF	---	CADD	INC	DEC	CL
9	Immediate Operand	LD	LD	C	AND	OR	XOR	CADD	LB	ADD	SUB	DADD	SCL	---	INC	---	CL
A	RECV (See Note)	NP/NC	NP/C	P/NC	P/C	---	---	---	---	PUSH	PULL	PUSH	PULL	TQE	TQF	ENQ	DEQ
B	B Immediate	LD	LD	C	AND	OR	XOR	---	LB	ADD	SUB	CADD	SCL	SOR	INC	DEC	CL
C	Extended	LD	ST	---	---	---	---	LBT	LB	---	---	LBF	---	---	INC	DEC	---
D	RFU	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
E	Branch True	B	BET	*	BLCT	BLBT	BART	JUMP	BVBT	JUMP	---	---	---	---	---	BCT	BSBT
F	Branch False	BS	BEF	#	BLCF	BLBF	BARF	JS	BVBF	JS	ILP	---	---	---	---	BCF	BSBF

NOTES:
- NP = No Parity
- NC = No CRC
- P = Parity
- C = CRC
- * = BZT/BST
- # = BZF/BSF

Section 4
COMMUNICATIONS CONTROL BLOCKS

COMMUNICATIONS CONTROL BLOCKS

Communications Control Blocks (CCBs) are used by the Processor to define the address, range, control, and status for each data transfer between the Processor and main memory.

The main memory program dynamically sets up the address, range, and control field for a CCB by issuing an I/O order. At the completion of a data transfer, the CCB status field is updated to reflect the final status of the CCB.

CCBs are data buffers in main memory to or from which the data channels request DMA service. Each CCB contains an address and range field that defines a data buffer, and also includes other control and status information related to the handling of the particular message. This information permits a message as transmitted over the line to be placed or obtained from more than one data buffer.

The CPU issues each CCB to the channel by executing a Load Data Area IOLD order and an output CCB control I/O order. The channel signals completion of a CCB to the CPU by executing a GIVE/GNB instruction; this also frees a CCB space for the channel. Each channel has space for eight CCBs and treats the space as a closed list. By means of firmware pointers and channel program instructions, the channel is able to manage its CCB list as consisting of three queues; a Next queue, a Processed queue (P-queue), and a current CCB (a queue of one CCB). Figure 4-1 shows these queues and the manipulations possible.

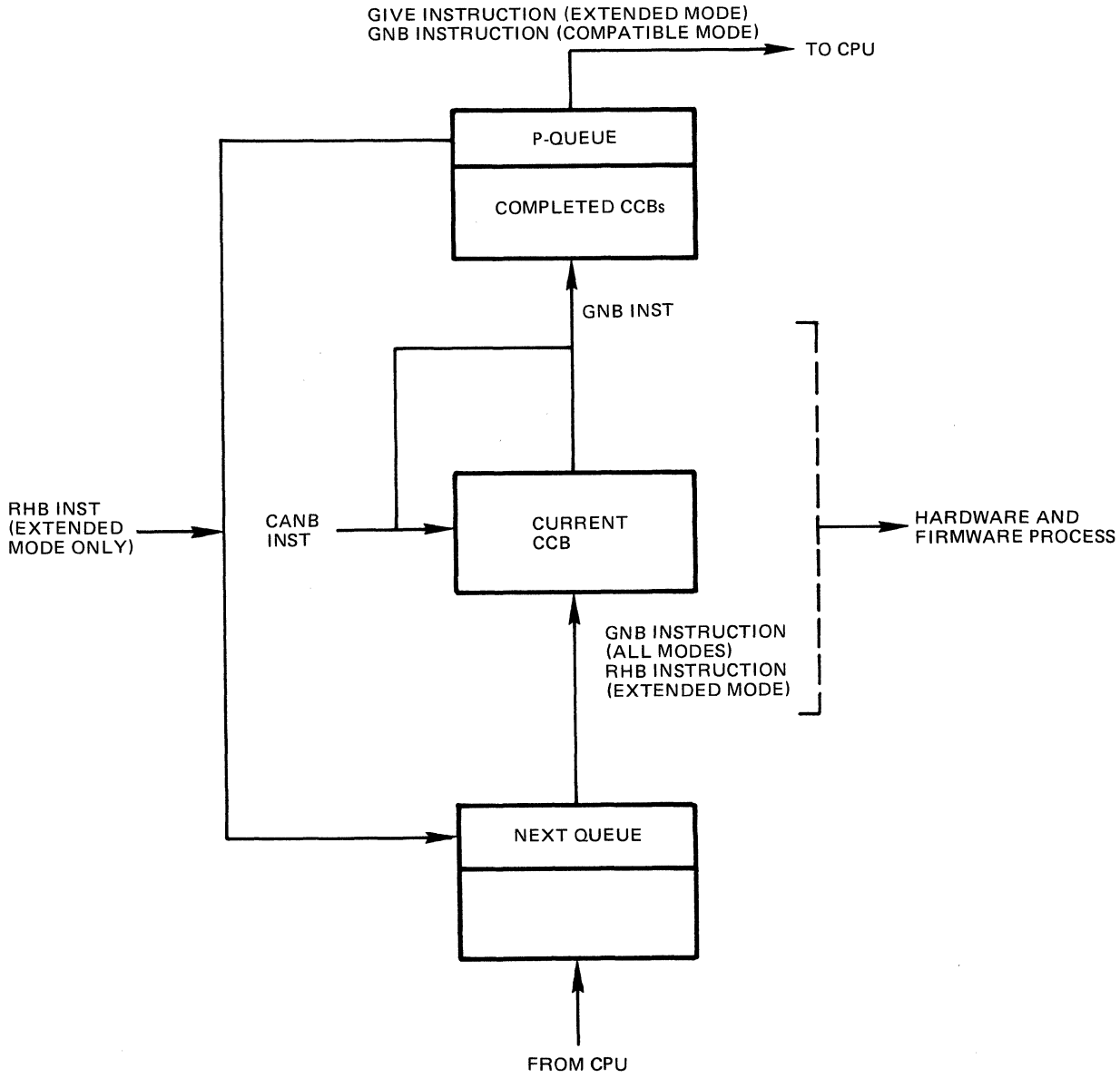


Figure 4-1. Communications Control Block Flow

CCBs AND CCB PROCESSING

CCBs are given to the NMLCP for processing via I/O orders. The CCBs reside in the Processor's Random Access Memory (RAM) and each channel is capable of storing up to eight CCBs.

CCB Setup

Each CCB is set up in the Processor RAM by the use of a Data Area IOLD (FC=09/0D) order and an Output CCB Control IO (FC=0F) order. The general format of a CCB is shown in Figure 4-2. A CCB occupies eight consecutive bytes. A description of CCB contents follows.

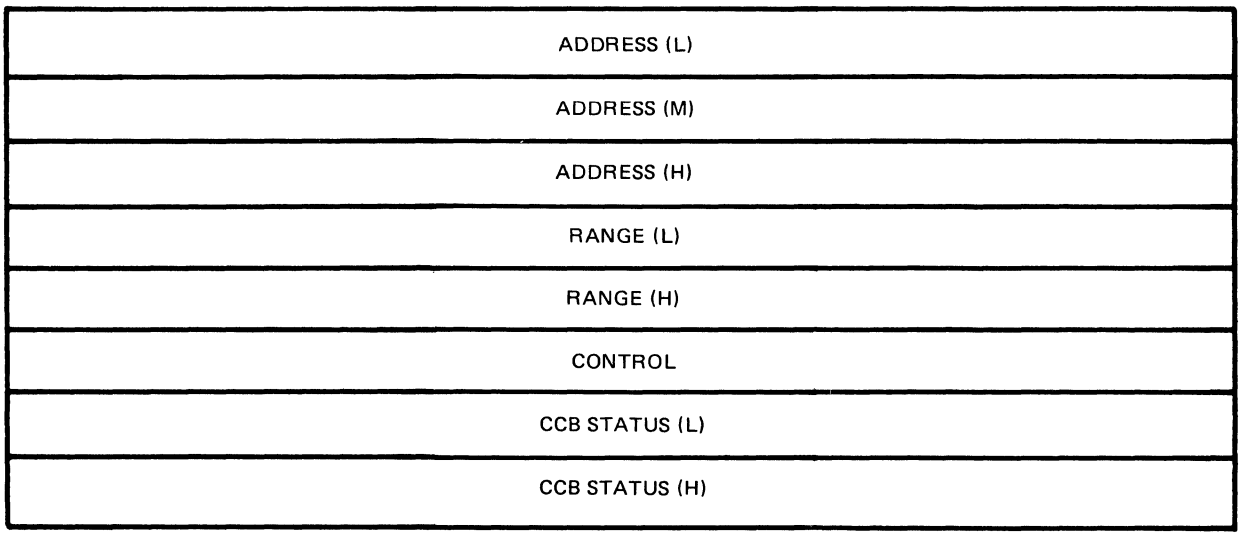


Figure 4-2. CCB General Format

1. ADDRESS - The initial address is loaded into a CCB by the IOLD order. This 24-bit address is the starting address of a Communications Data Block (CDB) in main memory to be used in an Processor/system data transfer.

2. RANGE - The initial range is also loaded into a CCB by the IOLD order. This 16-bit byte range defines the range count in bytes of the extent of a block or CDB which is to be read or written during the execution of this CCB.

3. CONTROL - The control field is used to contain CCB-specific control information.

4. CCB STATUS - The status area is used to store the return status as accumulated in LCT 16,17/48,49. The status word is set to Zeros by the execution of the Output CCB Control order for this CCB during CCB setup. Return status from LCT 16,17/48,49 is set in this area on completion of the CCB (GIVE/GNB instruction). Refer to Section 2 function code 18 for a listing of the status bits.

5. RESIDUE - The residue of the range count is loaded into the range area of the CCB by the receive channel on completion of the CCB (GIVE/GNB instruction).

When a CCB is loaded, the address and range combined describe a Communications Data Block (CDB) in main memory to be read or written via the DMA capabilities of the Processor. CDBs are used as containers for both input-receive data and output-transmit data by the Processor and the system.

Characteristics of a CDB

Specific characteristics of a CDB are:

1. The CDB address is the byte address of the first byte in the CDB. As such, the first byte may be located in either the left or right byte of a main memory word.

2. Data in CDBs is arranged in increasing order within the block.

3. The order of transmit out of a CDB or receive into a CDB is the same.

4. The CDB range is the count of the number of bytes in the CDB. This number, added to the CDB address, specifies the last byte location in the CDB which may be on an odd or even byte boundary. The CDB range is specified in the IOLD range.

CCB Execution

CCBs are accessed by the Processor firmware during their execution and are effectively moved from one queue to another by updating of queue pointers in the Processor. The general flow and controlling of CP orders and CCP instructions are shown in Figure 4-1. The CCBs themselves are not accessible to CCPs.

The starting point for CCB execution is the setup established in the "CCB Setup" subsection. As data is transferred, the address is incremented and the range is decremented by 2 (1 for every byte that is processed by DMA), as long as the range is not Zero. When the range is Zero, no further change is made to the address and range on DMA transfers performed under control of this CCB, and the Last Character indicator that can be tested by the CCP is also set. The primary method of transfer is Word mode. Byte mode is used at the beginning or end of a CDB as required when start or end is on an odd byte boundary. The Processor post-increments the address and post-decrements the range so that after each transfer the address points to the location for the next transfer.

A Get Next Block (GNB) instruction is used to cause the current CCB to be placed in the P-queue and the CCB at the top of the Next queue to be fetched to become the new current CCB. In Extended mode, the original address and range is retained in the just-completed CCB so that the CCB could be reused. If desired, a Return Held Block (RHB) instruction could be used. CCB status is stored in the CCB as described in the "CCB Setup" subsection in Extended mode. The final (residual) range of the CCB is held in a firmware working location. Only one such location is provided for each receive channel, thus it contains the residual of the most recently completed CCB. (The contents of this location are used during the GIVE instruction by receive channels.)

In Compatible mode, the GNB instruction places the final address and range of the just completed CCB into the CCB. The contents of the control field of the new CCB are loaded into the indicators and LCT described in Section 2. After performing a GNB, the CCP should always test the Valid indicator for a True condition before attempting further data transfer.

A Return Held Block (RHB) instruction causes the CCBs in the P-queue and the current CCB to be returned to the top of the Next queue and then the CCB at the top of the Next queue is fetched to become the new current CCB. The indicators are set to appropriate values for the new CCB. After performing an RHB, the CCP should always test the Valid indicator for a True condition before attempting further data transfer.

The Cancel Character (CANC) instruction decrements the address in the Processor and increments the range in the Processor of the current CCB by 1 for each execution. However, the canceling cannot go beyond the initial starting point of the current CCB. The Zero (Z) indicator will be set by the CANC instruction if the current CCB is at its initial starting point.

The Cancel Block (CANB) instruction sets the address and range in the Processor of the current CCB back to their initial values.

In Extended mode, the GIVE instruction is used to return a completed CCB from the P-queue to the jurisdiction of the CPU software. The control field of the CCB currently at the top of the P-queue is set as described in the "CCB Setup" subsection and if the interrupt bit of the field was a One, an Interrupt A is sent to the CPU. The status complete bit in the CCB's status field is set to One. The firmware's top P-queue pointer is updated. On receive channels this instruction also causes the contents of the channel's firmware residual range register to be set into the range field of the CCB. If the P-queue was empty when this instruction was attempted, the Zero (Z) indicator will be set (no operation is performed).

CCB List

Each channel is capable of storing a maximum of eight CCBs. The CPU interface to this list is via I/O orders. While the Processor assists the CCB management, control of the CCB list resides in the CPU. The CPU adds CCBs to the bottom of the list and fetches completed CCBs from the top of the list.

All output orders that reference a CCB operate on the bottom of the CCB list. The Data Area IOLD order adds to the bottom of the list.

All input orders that reference a CCB operate on the top of the CCB list. The Input Next Status order deletes entries at the top of the list, making the next CCB the top of the list.

CCB LIST SETUP

The Output CCB Control order always references the present output CCB in the list of CCBs; this CCB is the last one for which a Data Area IOLD was issued, making the next CCB the bottom of the list.

The Data Area IOLD order always references the CCB that follows the present output CCB in the list. If this command is accepted, this CCB becomes the new present output CCB. The CCB list setup can progress until eight CCBs have been set up, or until the number of CCBs set up is eight ahead of the current CCB. An attempt to add another CCB to the setup list while the above conditions exist will result in a NAK response from the Processor.

A CCB is deleted from the top of the setup list by execution of the Input Next Status order.

CCB LIST RETURN STATUS

Input Status, Input CCB Control, and Input Range orders always reference the CCB for which return status is required in the list of CCBs. The CCB in this case is that currently on the top of the CCB list.

Execution of the Input Next Status order causes the CCB on the top of the CCB list to be deleted, thus releasing Processor channel storage for CCBs.

If the CCB list is empty, the Input Next Status order will be NAKed by the Processor.

CCB LIST INITIAL CONDITIONS

The initial condition of the CCB list for each channel is that the list is empty. The following actions will cause the CCB list initial conditions to exist:

1. Processor Hard Initialize (all channels are affected)
2. Channel Initialize
3. CCB List Reset.

Note that the CCB list may be empty without being reset. When the CCB list is in the reset state, the initial entry to the list will always be placed in the RAM location designated CCB1 for the channel.

CCB LIST EXECUTION

The Processor executes CCBs in the same order that CCBs were added to the CCB list. Execution of a CCB is complete as defined in the subsection entitled "Characteristics of a CDB."

Execution of the CCB list may be stopped or started by the Output Channel Control order. Setting bit 2 will stop I/O. Stop I/O will terminate the CCB being executed. During a stop condition, status is not accumulated. Setting bit 1 will start CCP execution, thereby enabling execution of the next CCB to be started.

CCB LIST IN RAM

For test purposes the CCB storage in the Processor is accessible through the RAM Data Transfer functions (see the subsection in Section 1 entitled "RAM Data Transfer").

For each channel, storage is provided in the Processor for a maximum of eight CCBs. The CPU-visible CCB list is stored in this storage area. The CCB list logically rotates through the storage area as CCBs are added to the bottom and deleted from the top of the CCB list. Whenever the channel CCB list is reset or initialized, the CCB list is emptied.

CCB FIELD FORMAT

Figure 4-3 shows the format of fields within a CCB that control a communications data transfer. Each field is described in the following subsections.

NOTE

A specially formatted CCB is used to control a Block mode input/output operation from/to the Processor RAM. For more information about this special type of CCB, see the "Block Mode Read/Write" subsection in Section 1.

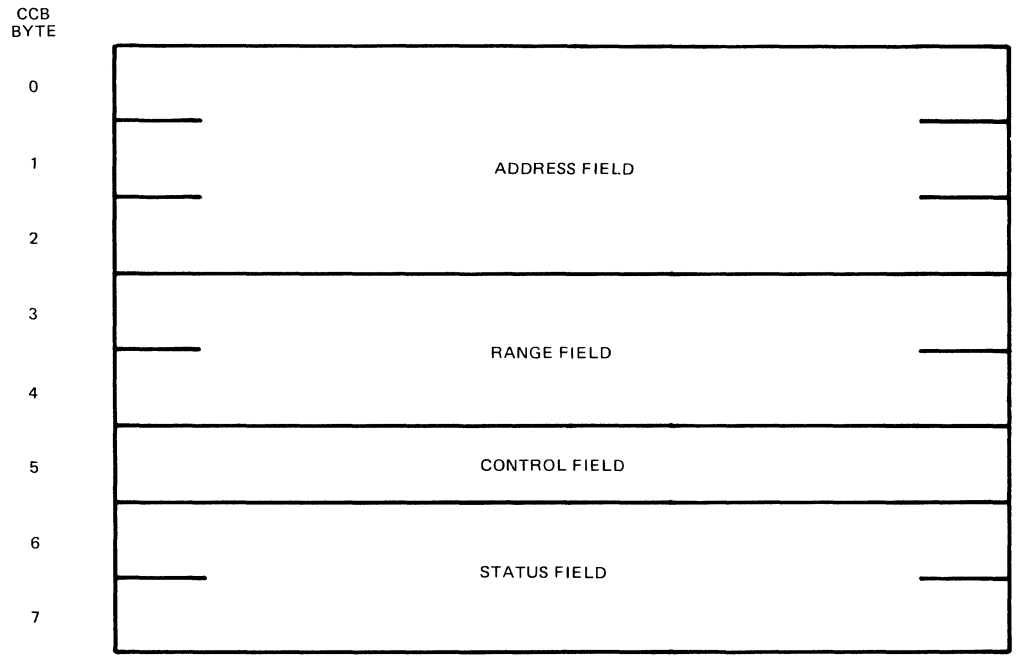


Figure 4-3. Format of CCB Fields in the NMLCP

CCB Address Field

This field, occupying bytes 0, 1, and 2 of the CCB, is written from the main memory program by an IOLD (Output CCB Address and Range) instruction. When first written, the address field contains the starting byte address of a CDB in the main memory program. The low-order end of the starting byte address is contained in byte 0 of the CCB. During processing, it is always increased in increments of one.

The address field value is increased as data bytes are physically transferred between the CDB and the Processor, based on the CCP's execution of format 1 Load (LD) or Store (ST) instructions. Executing these instructions may cause three different types of data transfer, depending upon the position of the byte (or bytes) within the 16-bit CDB word being affected (see Figure 4-4).

Normally, the address field value is increased by 1 for every two executions of a format 1 LD or ST instruction. This causes two data bytes to be physically transferred between the CDB and the Processor.

Two exceptions occur when the address field will be increased by 1 for the physical transfer of only one data byte. This occurs when the instruction pertains to (1) a CDB that begins at an odd-byte boundary or (2) a CDB that ends at an even-byte boundary. In no case will the address field be incremented if the current range value is Zero.

	16-BIT WORDS FOR CDBs	
EXCEPTION ONE	(EMPTY)	FIRST BYTE
NORMAL DATA TRANSFER	BYTE	BYTE
NORMAL DATA TRANSFER	BYTE	BYTE
NORMAL DATA TRANSFER	BYTE	BYTE
NORMAL DATA TRANSFER	BYTE	BYTE
EXCEPTION TWO	LAST BYTE	(EMPTY)

Figure 4-4. Three Types of Data Transfer

CCB Range Field

The range field occupies bytes 3 and 4 of the CCB. This field is written from the main memory program by an IOLD (Output CCB Address and Range) instruction. When first written, the range field indicates the number of bytes in the CDB. The low-order end of the range value is contained in byte 3 of the CCB.

The range field value is decreased by 1 each time a format 1 LD or ST instruction is executed in the CCP unless the current range value is Zero.

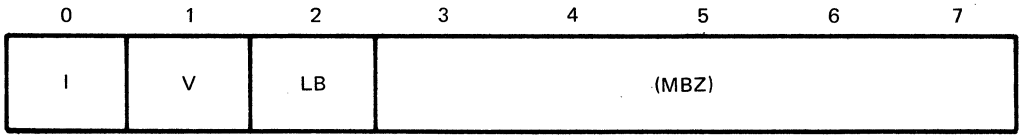
CCB Control Field

The control field occupies byte 5 of the CCB. The control field is written from the main memory program by an IO (Output CCB Control) instruction.

Only the first three bits of the control field are functional. These bits indicate the presence or absence of three conditions: Interrupt control, valid CCB, and last CDB. The format and significance of byte 5 is described in the following paragraphs:

Bits 0-3 are delivered to Processor indicators and bits 0-7 are also delivered to LCT 76/108 when the CCB is loaded by a GNB instruction.

When the CCB is completed bits 0-3 will be cleared to Zeros (see Table 4-1).



Bit 0 - Interrupt Control

0 - No action.
1 - Interrupt the main memory program when this CCB is marked as completed (CCB byte 7, bit 3).

The interrupt will occur at the interrupt level assigned to this channel. If no interrupt level has been assigned, no interrupt will occur.

Bit 1 - Valid CCB

0 - This is not a valid CCB; it cannot be used as an current CCB. Such a condition exists until this bit is set to 1 by an IO (Output CCB Control) instruction.

This bit is reset to Zero by firmware when this CCB is marked as completed (bit 3 of CCB byte 7) after it has been used during processing of a CDB.

1 - This is a valid CCB; it is usable as an current CCB.

This bit must be set to One so as to complete the setup of the CCB.

Bit 2 - Last CDB

0 - No action.
1 - This CCB pertains to the last CDB in a message.

This is a flag that can be used by the CCP for special processing of the last CDB in a message. If this bit is set to One, the Processor's LB-indicator will be set to One when this CCB is current. The CCP can test the LB-indicator by means of BLBT (Branch if Last Block True) and BLBF (Branch if Last Block False) instructions.

Bits 3 through 7 - Reserved for Software Use

In the Extended mode Bits 3 - 7 are loaded from the LCT location defined by B-register, when the GIVE instruction is executed.

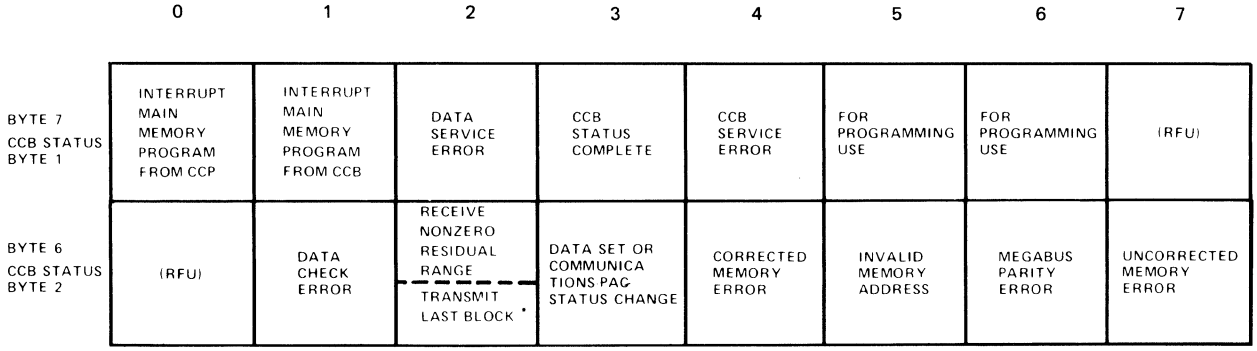
CCB Status Field

The status field comprises bytes 6 and 7 of the CCB. The CCB status field is reset to Zero as setup of the CCB is completed by execution of an IO (Output CCB Control) instruction. Later, as processing ends relative to a CCB, its status field is updated by firmware and the CCB status complete bit (CCB byte 6, bit 2) is set to One. (Table 4-1 indicates the conditions under which processing relative to a CCB can end.)

The CCB status field is updated with information from the two LCT status bytes combined with other information. The LCT status bytes are bytes 16 and 17 for a receive channel and bytes 48 and 49 for a transmit channel. Once the status field of the CCB has been updated, and the status complete bit set, the CCB's status is said to be meaningful.

The status bytes of the status CCB can be read from the main memory program, whenever appropriate, by an IO (Input CCB Status) instruction. An IO (Input Next CCB Status) instruction moves the status CCB pointer to the following CCB (which then becomes the status CCB) and reads the status field of this new status CCB.

Figure 4-5 shows the format of the two bytes of the CCB status field. The entire word is passed from the LCT status bytes. Note that, in the CCB status field, status byte 1 is stored above status byte 2. This order is the opposite of the order of the status byte in the LCT and will be presented that way.



* For transmit, bit 2 equals last CCB block.

Figure 4-5. NMLCP CCB Status Bytes 1 and 2

BYTE 7 - CCB STATUS BYTE 1

Bit 0 - Interrupt Main Memory Program from CCP

0 - No action.
1 - The interrupt instruction has been executed.

Bit 1 - Interrupt Main Memory Program from CCP

0 - No action.
1 - The main memory program has been interrupted when processing end relative to this CCB.

This bit is set to One in either of two cases: (1) if bit 0 of CCB byte 5 has been set to One by an IO (Output CCB Control) instruction in the main memory program or (2) if bits 0 and 2 of LCT byte 8/40 have been set to One and a data set or adapter status change has been recorded in LCT byte 14/46 (Data Set Scan).

Bit 2 - Data Service Error

0 - No data service error has occurred.
1 - A data timing window has been missed.

On receive, the adapter has detected a receive overrun (see bit 6 of LCT byte 14/46 in Section 5). On transmit, the adapter has detected a transmit underrun (see bit 7 of LCT byte 14/46 in Section 5).

Bit 3 - CCB Status Complete

This bit is set to One after the CCB status field is written by Processor firmware to indicate that processing relative to this CCB has ended and the contents of its status field are

meaningful. Table 4-1 indicates the conditions under which processing relative to a CCB can end.

Bit 4 - CCB Service Error

0 - No CCB service error has occurred.
1 - An error that occurred before this CCB became valid. (Error occurred in either receive or transmit, each explained below)

On receive, an ST instruction was attempted when there was no valid CCB. The instruction was not executed. Instead, Processor firmware set this bit to One (in LCT status byte 1) and proceeded to the next sequential instruction inthe CCP.

On transmit, an LD instruction was attempted when there was no valid CCB. The instruction was not executed. Instead, Processor firmware set this bit to One (in LCT status byte 1), returned the CCP pointer to the address of this LD instruction, and executed a WAIT (Wait) instruction. At the next channel request for this channel, this instruction was attempted again.

Bits 5 and 6 - For Programming Use

Within LCT status byte 1, these two bits can be used by the CCP for application-specific purposes. Later, when the contents of the LCT status bytes are transferred to the CCB status field, these two bit positions become avaialbe for security by the main memory program as it issues an IO (Input CCB Status) or IO (Input Next CCB Status) instruction. Thus, these two bit positions can be used as a means for the CCP to pass application-specific status information to the main memory program.

Bit 7 - Reserved for Future Use (RFU)

BYTE 6 - CCB STATUS BYTE 2

Bit 0 - Reserved for Future Use (RFU)

Bit 1 - Data Check Error

0 - No data check error has occurred.
1 - A data parity error has been detected by firmware, or the CCP has set this bit after detecting a cyclic redundancy check error.

In both cases, this bit setting is relevant only for receive operations.

Bit 2 - (For Receive channels) Nonzero Residual Range

0 - No CCB residual range exists.
1 - The CCB has been terminated before its range field value decreased to Zero.

Bit 2 - (For Transmit channels) Last Block

0 - Not last block
1 - Last block

Bit 3 - Data Set or Communications-Pac Status Change

0 - No data set or adapter status change has been recorded.
1 - Bits 0 and 1 of LCT byte 8/40 were set to One and a data set or adapter status change was recorded in LCT byte 14/46.

Bit 4 - Corrected Memory Error

0 - No corrected memory error has occurred.
1 - One or more hardware-corrected memory errors occurred in the CDB related to this CCB.

Bit 5 - Invalid Memory Address

0 - No invalid memory address has occurred.
1 - A reference to a CDB has resulted in an invalid memory address on the Megabus Network; main memory has issued a NAK. This condition has caused the CCB to be terminated.

Bit 6 - Megabus Parity Error

0 - No Megabus parity error has occurred.
1 - Incorrect parity existed on the Megabus network as a data character was transferred to the Processor. This condition has caused the CCB to be terminated.

Bit 7 - Uncorrected Memory Error (NMLCP)

0 - No uncorrected memory error has occurred.
1 - An uncorrected memory error occurred in the CDB related to this CCB. This condition has caused the CCB to be terminated.

SETTING UP A CCB

A CCB is normally set up by the execution of two instructions by the main memory program. The first instruction is an IOLD (Output CCB Address and Range) which fills in the CCB's address and range fields with appropriate values. The second instruction is an IO (Output CCB control) which writes an appropriate value into the CCB's control field and resets the CCB's status field to Zero. A CCB is not available for use until it has been set up by these two instructions.

NOTE

For a description of how to set up a CCB to control a Block mode input/output operation, see subsection entitled "Block Mode Read/Write" in Section 1.

CCB DESCRIPTIONS OF A CDB

When a CCB is first written by an IOLD (Output CCB Address and Range) instruction, it identifies the starting byte address of a CDB in the main memory program. The starting byte address can be either the first (left) or second (right) byte of a word in main memory. (Likewise, the CDB can end with the first or second byte of a main memory word.)

A CDB is a consecutive series of bytes in main memory. It can be used as either an input (receive) buffer or an output (transmit) buffer during data transfers to or from the Processor.

Data in a CDB is arranged in increasing order from the lowest byte address towards the highest byte address in the CDB. The first character transmitted or received is contained in the lowest byte address of the CDB.

Note that the CCB's address field value increases only as data characters are physically transferred between the CDB and the Processor; see "CCB Address Field" earlier in this section. The contents of the CCB's Range field decrease by 1 each time a format 1 LD or ST instruction is executed in the CCP unless the current range value is Zero.

PROCESSING A CURRENT CCB

A CCB is current when it is availiable for use by Processor firmware to control the flow of data to or from a CDB. A CCB becomes current when the CCP issues a GNB (Get Next Block) instruction that moves the current CCB pointer to it; if the CCB list was empty, the act of setting up the first CCB will also make that CCB current. The fields of a CCB are not accessible to the CCP. A copy of the control field of a current CCB is loaded into LCT 76/108 and the Valid and Last Block bits of the control field are set into indicators where they may be tested via the BVBF, BVBT, BLBF, and BLBT instructions.

As data characters are physically transferred to or from the CDB by way of the Processor, the contents of the CCB address field are increased appropriately; see "CCB Address Field" earlier in this section.

If the CCB describes a CDB input (receive) buffer, the CCB address field points to the location that will be used to store the next data character to be received from the Processor.

If the CCB describes a CDB output (transmit) buffer, the CCB address field points to the next data character to be transferred from main memory to the Processor.

The contents of the CCB range field are decreased by 1 as each LD or ST instruction is executed by the CCP unless the current range is already Zero. This process continues until a termination condition (EOT, EOB or range runout) establishes the end of a data block. Range residue in the range field indicates the number of bytes in the CDB to or from which direct memory access data transfers had not been made when the CCB was marked as completed.

When processing of a current CCB is completed, the status of the related data transfer is recorded in the two status bytes of the CCB. (Some of the status information comes from the LCT status bytes; see "CCB Status Field" earlier in this section.) At this point, the CCB status complete bit (CCB byte 7, bit 3) is set to One if compatible mode CCB processing is being used.

MODE CONSIDERATIONS FOR CCB PROCESSING

The Processor has two modes of operation - Compatible and Extended - which can be selected on a channel basis. The mode selected determines how CCP instructions that control the CCB processing are executed.

The instructions affected are:

- CANB
- CANC
- GIVE
- GNB
- RHB

CCB Processing in Extended Mode

In Extended mode, two CCP instructions to control CCB processing should be used at the completion of a CCB; namely, the GNB and the GIVE instructions. This two-step process enables the Controller to hold completed CCBs.

When processing receive CCBs, the GIVE instruction should normally be issued immediately after the GNB instruction to release the CCB to CPU control (if receive error recovery procedures are not being implemented in the Processor CCP). The contents of the channel's firmware Residual Range register is loaded into the range field of the most recent held CCB.

When processing transmit CCBs, the two steps should be separated if transmit error recovery procedures are not being implemented in the Processor CCP. At the completion of a CCB, the GNB instruction is issued first. In this mode the GNB will:

1. Terminate the current CCB.

2. Transfer the contents of the LCT Status bytes to the status field of the current CCB.

3. Reset the address and range to their initial values.

4. Advance the current CCB pointer.

5. Load the control field of the new CCB into the Processor indicators.

A CCB in this state is called a Held CCB.

The channel can reuse all of the Held CCBs by issuing a Return Held Block (RHB) Instruction. The RHB will:

1. Set the address and range of the current CCB back to their initial values.

2. Fetch the first Held CCB (one beyond the status CCB) to become the current CCB.

3. Load the control field of the new current CCB into the Processor indicators.

The channel can release a CCB to the CPU by issuing a GIVE instruction for each Held CCB. The GIVE Instruction will:

1. Set the status complete bit.

2. Attempt an interrupt, if the interrupt bit of the CCB Control Field is set.

3. Reset the CCB control field.

4. Load the contents of the LCT pointed to by the B-register into the software bits of the CCB control field.

5. Load the channel's Residual Range Register into the CCB.

GIVEN CCBs are under the control of the CPU which can check the Status and the residual range, and can also issue an Input Next Status instruction to release the CCB for reuse.

CCB Processing in Compatible Mode

In Compatible mode, only one CCP instruction to control CCB processing should be executed at the completion of a CCB; i.e., the Get Next Block (GNB) instruction.

The GNB will:

1. Terminate the current CCB.

2. Transfer the contents of the LCT status bytes to the status field of the current CCB.

3. Set the status complete bit.

4. Attempt an interrupt if the interrupt bit of the CCB control field is set.

5. Reset the control field of the current CCB

6. Advance the current CCB pointer.

7. Load the control field of the new CCB into the Processor indicators and LCT.

The GIVE and the RHB instructions should not be used in Compatible mode.

Instructions for CCB Processing in Both Modes

The Cancel Character (CANC) and the Cancel Block (CANB) instructions execute in both Compatible and Extended mode.

The CANC is used in receive CCP to delete the previous character received. When a special character is detected, executing a CANC in place of the ST will discard the special character and overwrite this previous character on the next ST. The CANC instruction decrements the address field and increments the range field of the current CCB. (this can not be continued beyond the initial starting point of the current CCB).

The CANB instruction is used to attempt recovery after an abort. The CANB sets the address and range of the current CCB back to their initial values.

COMPLETION OF A CCB

Table 4-1 describes various conditions that cause a CCB to be marked as completed (i.e., the CCB status complete bit CCB byte 7, bit 3 is set to One). The table also describes the means by which each condition can be ascertained by the main memory program.

Table 4-1. CCB Completion Conditions

CCB Completion Condition	Action by Which Main Memory Program Can Ascertain Condition
In Compatible mode, the CCP has issued a GNB instruction, causing the current CCB to be marked as completed. In Extended mode, the CCP has issued a GIVE instruction for the CCB at Held list.	CCP could indicate this action by setting bit 5 or 6 in LCT byte 16 (for receive channel) or LCT byte 48 (for transmit channel) before issuing the GNB instruction. This information would then be available to the main memory program through bit 5 or 6 of CCB status byte 7.
Main memory progam has issued an IO (Output Channel Control - Stop Input/Output) instruction	Self-evident
Invalid memory address	IO (Input CCB Status) or IO (Input Next CCB Status). CCB byte 6, bit 5 is set to One.
Megabus parity error	IO (Input CCB Status) or IO (Input Next CCB Status). CCB byte 6, bit 6 is set to One.
Uncorrected memory error	IO (Input CCB Status) or IO (Input Next CCB Status). CCB byte 6, bit 7 is set to One.

Section 5
LINE CONTROL TABLES

LCT DESCRIPTION

Each of the 16 data channel lines in the Processor has its own unique and independent Line Control Table (LCT) which is located within the Processor memory. All line configuration, setup, status, and control information appears in the LCT during Processor operation.

The LCT is important to Processor operations because it is visible to the CPU programmer via I/O commands, to the Processor CCP and firmware.

Each LCT consists of a block of 256 bytes. Refer to Table 5-1 for a summary of those functions. The locations within the first 64 LCTs that have software-processing functions are the same for the NMLCP as for the MLCP. Note that those LCTs (and bit positions within LCTs) designated for hardware use do not have identical purposes for the two processors. The LCTs are completely independent on a line basis, such that a Processor CCP can access only the LCT of the line on whose behalf it is executing.

LCTs can be accessed by the CPU via the Input, Output, and Read and Clear LCT byte IO orders, or by executing a Compatible Mode Block Read or Block Write or RAM Data Transfer operation.

Table 5-1. Line Control Table

LCT Byte	Contents
0	RHU (Reserved for Hardware Use)
1	Firmware revision number of adapter PROM
2	Receive configuration
3[a]	Receive CRC residue - byte 1
4[a]	Receive CRC residue - byte 2
5	Receive context
6	Foreground receive CCP pointer - MSB
7	Foreground receive CCP pointer - LSB
8	Data set scan control
9	Receive firmware control
10	RHU
11	Adapter specific
12	Receive interrupt control A - byte 1
13	Receive interrupt control A - byte 2
14	Receive CA status
15[a]	Receive CA status mask
16[a]	Receive status - byte 1
17[a]	Receive status - byte 2
18	Receive CCP subroutine pointer - MSB
19	Receive CCP subroutine pointer - LSB
20	Receive/transmit CA control
21	RHU
22	RHU
23-31[a]	CCP work area
32	RHU
33	RHU
34	Transmit configuration
35[a]	Transmit CRC residue - byte 1
36[a]	Transmit CRC residue - byte 2
37	Transmit context
38	Foreground transmit CCP pointer - MSB
39	Foreground transmit CCP pointer - LSB
40	Data set scan control
41	Transmit firmware control
42	RHU
43	Adapter specific
44	Transmit interrupt control A - byte 1
45	Transmit interrupt control A - byte 2
46	Transmit CA status
47[a]	Transmit CA status mask

Table 5-1 (cont). Line Control Table

LCT Byte	Contents
48[a]	Transmit status - byte 1
49[a]	Transmit status - byte 2
50	Transmit CCP subroutine pointer - MSB
51	Transmit CCP subroutine pointer - LSB
52	CCP work area
53	RHU
54	RHU
55[a]	Address for input LCT byte command
56-63[a]	CCP work area
64	Receive stack pointer
65	Receive start of queue pointer
66	Receive end of queue pointer
67	Receive CRC residue byte 3
68	Receive CRC residue byte 4
69	RHU
70	Background receive CCP pointer MSB
71	Background receive CCP pointer LSB
72	RHU
73	RHU
74[a]	Receive timer
75[a]	Receive activity flags
76[b]	Receive CCB control field
77	Receive deferred interrupt count
78	Receive pause count
79-95	RHU
96	Transmit stack pointer
97	Transmit start of queue pointer
98	Transmit end of queue pointer
99[a]	Transmit CRC residue byte 3
100[a]	Transmit CRC residue byte 4
101	RHU
102	Background transmit CCP pointer MSB
103	Background transmit CCP pointer LSB
104	RHU
105	RHU
106[a]	Transmit timer
107[a]	Transmit activity flags
108[b]	Transmit CCB control field
109	Transmit deferred interrupt count
110	Transmit pause count
111-127	RHU
128-255[a]	CCP work area

[a] May be modified by CCP.
[b] Extended mode only.

The contents of each LCT must be preconditioned by either a load or a channel initialize before the start-up of any data transfer by a channel. The Hard Initialize and Channel Initialize provide an all Zero starting value for each LCT location.

The contents of an LCT location can be specified in the DPS 6 system by a data declaration to the DPS 6 assembler. Construction of an entire LCT will be assisted by macrofacilities, and should be used wherever possible because of the error checking features of macrofacilities.

LOADING AN LCT AFTER AN INITIALIZATION

Hard Initialize or Channel Initialize clears all LCT locations to zero. CPU software need only load certain specific locations, as described below, to permit a specific line to be used.

LCT Locations to be Set Up by CPU

1. LCT 2 Receive Configuration. This location is used to store the Communications Adapter (CA) configuration data. The bit definitions are CA specific and the appropriate CA specification should be consulted; generally the following format applies:

   Bits 0 & 1 - Character Size:

   00 = 5 bits
   01 = 6 bits
   10 = 7 bits
   11 = 8 bits

   Bit 2 - MBZ

   Bit 3 - Parity:

   0 = Odd
   1 = Even

   Bit 4 - Stop Bits:

   0 = 1 Stop bit
   1 = 1.5 or 2.0 Stop bits (1.5 for 5-level code only)

   Bits 5-7 - CRC:

000 = CRC-16	001 = Federal Standard
010 = CCITT	011 = CRC-6 (PARS)
100 = CRC-12	101 = RFU
110 = LRC	111 = RFU

   LCT 2 should be set up before a START I/O is issued for any CCP segment that is to involve a data transfer.

2. LCT 6 and 7 Foreground Receive CCP Pointer. If LCT 70 and 71 contain all zeros, the receive channel CCP P-counter used during both data service and events is kept in these two bytes and, when the channel is serviced, the contents of these locations are placed in the P-register. If LCT 70 and 71 do not contain all zeros, the receive channel CCP P-counter kept in these two bytes is used only for data service, and LCT 70 and 71 are used for the CCP P-counter on event service.

3. LCT 8 Data Set Scan Control. This byte is used by the Data Set Scan firmware to determine what action it should take on its Data Set Scan of this channel. The bits are defined as follows:

   Bit 0 - Data Set Scan Control:

   0 = Do not scan (and take no action on bits 1, 2, and 3 below)
   1 = Scan

   Bit 1 - Set Data Set Status Change:

   0 = No action
   1 = Set status bit

   If a data set status change is detected, bit 3 of LCT 17 (LCT status byte 2) is set.

   Bit 2 - Interrupt:

   0 = No action
   1 = Interrupt

   If a status change is detected, an interrupt is generated on this channel. Bit 1 of LCT 16 (LCT status)is set together with bit 3 of LCT 17.

   Bit 3 - Start CCP:

   0 = No action
   1 = Start CCP

   If a status change is detected, the CCP program is started for the channel.

   NOTE

   Bits 2 and 3 are mutually exclusive; only one of them should be set.

Bit 4 - Register Control:

0 = Scan on LR5 contents as assembled by the adapter
1 = Scan on FR5 contents from the FLAP

Bits 5-7 - MBZ

4. LCT 15 Receive CA Status Mask. This mask is used by the hardware for data set status change detection on receive. All bits are used. The bits that are set to One in this mask select the bits in the CA status which may be monitored for status changes.

5. LCT 20 Receive/Transmit CA Control. The receive/transmit CA control byte is used to control the data set control (bits 0 through 4) and the CA state (bits 5 through 7).

   Refer to the adapter specification for the definition of bits 0 through 4.

   Bit 5 is defined as the Test Mode control, bit 6 the receive on bit, and bit 7 the transmit on bit. The LR2 register of the adapter specification defines the precise usage of these bits in the CA. Bits 5, 6, and 7 are used by hardware for reference and control.

6. LCT 55 Address for Input LCT Byte. This location will be interpreted by firmware as containing the address from which the LCT byte will be delivered in response to the Input LCT Byte (FC=1E) comand. If that command is not used, LCT 55 may be considered to be a working location. The address shall be an eight-bit quantity located in bits 0 through 7.

7. LCT 70 and 71 Background Receive CCP Pointer. If this pointer is nonzero, then when the channel is given Event service, the contents of these locations are placed in the P-register and when the channel relinquishes control, the P-counter is stored back in these two bytes. If these two bytes are zero, see LCT 6 and 7.

8. LCT 76 Receive CCB Control Field. The low-order eight bits of the Control field of a CCB is placed here when the CCB is loaded into the channel.

   The following LCT locations act for transmit functions, but they have the same definitions as their corresponding LCT locations governing receive functions. Refer to those LCTs for definitions.

9. LCT 34 Transmit Configuration. See LCT 2.

10. LCT 38 and 39 Foreground Transmit CCP Pointer. See LCT 6 and 7.

11. LCT 40 Data Set Scan Control. See LCT 8.

12. LCT 47 Transmit CA Status Mask. See LCT 15.

13. LCT 102 and 103 Background Transmit CCP pointer. See LCT 70 and 71.

14. LCT 108 Transmit CCB Control Field. See LCT 76.

LCT Locations for CCP Work Area

LCTs specifically indicated as such may be used by the CCP as working locations (see Section 3). Alternately, they can be used as a means by which to pass special information between the CCP and the CPU software or vice versa.

LCT Locations Maintained By NMLCP

Certain LCT locations contain significant information relative to management of the communications path. This information is generally maintained by the firmware for use by the CCP during character processing. It is dynamic and tends to change at a character rate. The information may be useful to the T&V or during software or CCP debugging.

1. LCT 1 Firmware Revision Number. This contains the revision number of the firmware specifically associated with the line. The revision number will be $(10)_{16}$ or greater.

2. LCT 3,4,67,68 Receive CRC Residue. Intermediate Block Check results are stored in these locations for reference by both hardware and the Processor CCP procedure on receive.

3. LCT 5 Receive Context. This location is used to save certain context information during times when the channel CCP is not being executed.

   Bits 0-6 - RHU

   Bit 7 - Pause Bypass

   0 = Pause ON
   1 = Pause OFF

4. LCT 9 Receive Firmware Control (for firmware use only)

   Bits 0-4 - RHU

   Bit 5 - Start I/O See Output Channel Control (FC=05)

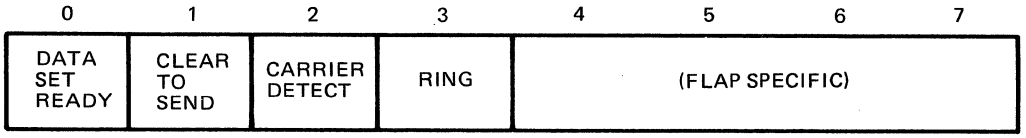
   Bits 6 and 7 - RHU

5. LCT 12 and 13 Receive Interrupt Control A. This contains information delivered to the channel via the Output Interrupt Control A order (FC=03).

   LCT 12 - Bits 0-7 CPU Channel Number (MSB)

   LCT 13 - Bits 0 and 1 CPU Channel Number (LSB) Bits 2-7 Level

6. LCT 14 Receive CA Status. This location is periodically updated by the firmware when a data set status scan occurs. At that time the contents of LR5 or FR5 will be copied into LCT 14. The bit definitions are FLAP register specific but generally meet the following format:



7. LCT 16 and 17 Receive Status - Bytes 1 and 2. Status is accumulated here during data transfer and transferred to the appropriate CCB status area during GNB execution. See I/O Orders Input Next Status (FC=1A) and Input Status (FC=18) for further treatment.

8. LCT 18 and 19 Receive CCP Subroutine Pointer. The receive channel CCP subroutine pointer locations in the LCT are used to store the subroutine return pointer used by the BS and RET instructions (see Table 3-3). LCT 19 contains the eight least significant bits and LCT 18 contains the most significant bits of the pointer.

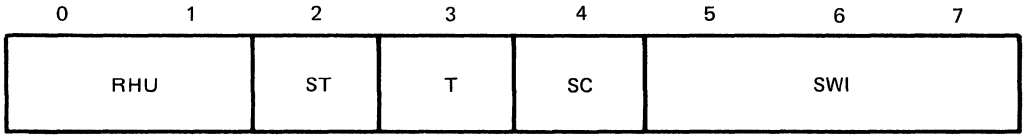
9. LCT 64 Receive LCT Stack Pointer. This points to the current top of the receive channel's LCT LIFO stack. See Table 3-3 concerning PUSH, PULL, JS, and RETJ instructions. The pointer must be set prior to executing these instructions.

10. LCT 65 Receive LCT Start of Queue Pointer. This points to the current input location of the receive channel's LCT end of queue stack. See Table 3-3 concerning DEQ, ENQ, TQE, TQF instructions.

11. LCT 66 Receive End of Queue Pointer. This points to the current output location of the receive channel's LCT End of Queue stack. See Table 3-3 concerning DEQ, ENQ, TQE, TQF instructions.

12. LCT 74 Receive Timer. This contains the current time value of the receive channel's timer.

13. LCT 75 Receive Activity Flags. A receive channel Activity Flag code is accumulated here where it can be consulted by the CCP. The SWI field can be set by CCP to create an event-level service request. The event-level CCP should clear these bits as it services the request(s). Format of the byte is as follows:



where ST = Start I/O
T = Timer
SC = Data Set Scan
SWI = Software-Created Start Interrupt code

14. LCT 77 Receive Deferred Interrupt Count. This register contains a count of the number of deferred interrupts outstanding on the channel.

15. LCT 78 Receive Pause Count. A running count of the number of Pause operations forced on the channel is maintained here by incrementing this location each time a Pause occurs.

The following LCT locations act for transmit functions, but they have the same definitions as their corresponding LCT locations governing receive functions. Refer to those LCTs for definitions.

16. LCT 35,36,99,100 Transmit CRC Residue. See LCT 3,4,67,68.

17. LCT 37 Transmit Context. See LCT 5.

18. LCT 41 Transmit Firmware Control. See LCT 9.

19. LCT 44 and 45 Transmit Interrupt Control A. See LCT 12 and 13.

20. LCT 46 Transmit CA Status. See LCT 14.

21. LCT 48 and 49 Transmit Status - Bytes 1 and 2. See LCT 16 and 17.

22. LCT 50 and 51 Transmit CCP Subroutine Pointer. See LCT 18 and 19.

23. LCT 96 Transmit Stack Pointer. See LCT 64.

24. LCT 97 Transmit SOQ Pointer. See LCT 65.

25. LCT 98 Transmit EOQ Pointer. See LCT 66.

26. LCT 106 Transmit Timer. See LCT 74.

27. LCT 107 Transmit Activity Flags. See LCT 75.

28. LCT 109 Transmit Deferred Interrupt Count. See LCT 77.

29. LCT 110 Transmit Pause Count. See LCT 78.

LCT Locations Reserved for Firmware

Certain LCT locations are used by firmware as work areas. While they are equal in significance to other LCT locations, detailed knowledge of their contents is not necessary to an understanding of the Processor. Software should not access locations designated RHU or Firmware Work Area.

Extended LCTs

The Local Store can be accessed by the CCP as Extended (additional) LCTs. The type of access can be limited to Read only, or can permit writing as well, as defined on a channel basis via the Output Configuration A order. The Extended LCTs are addressed via an indirect indexed technique using specific CCP instructions (see Table 3-3 for all instructions with the hardware suffix -X, e.g., DECX, INCX, etc.) The instruction specifies an LCT in the channel's basic LCT area. The 8-bit contents of this LCT is then concatenated with eight lower order bits from the contents of the B-register to obtain the 16-bit Local Store Address.

The access time to Extended LCTs is considerably longer than to the basic LCTs.

If a channel attempts to Write into Extended LCTs and is not permitted to Write (Write Permit is off), the instruction is treated as illegal.

LCT Stacks and Queues

Each channel has the ability to use a portion of its basic LCT area as a Stack and/or a Queue. Refer to the treatment of LCT 64-66 and 96-98 where the names of the instructions that operate such storage arrays are given along with an explanation of the LCTs containing pointers to the arrays.

Stacks fill in the direction of decreasing LCT locations and are confined to the top 128 LCT locations of the line's LCT area and will go end-around from LCT 128 to LCT 255. A postdecrement/ preincrement scheme is used. Prevention of stack overflow/ underflow is the responsibility of the CCP.

A Queue is limited to 16 LCT locations and must begin on a modulo-16 LCT location. The Queue must be initialized by setting the start of queue and end of queue pointers equal to a modulo-16 location. Test Queue Full and Test Queue Empty instructions are provided. Queues fill in the direction of increasing LCT numbers and will go end-around at the maximum LCT entry.

Layout of LCT Bytes

Figure 5-1 presents a detailed layout of each byte in an LCT. The purpose is to indicate how each bit position of an LCT should be set up before communication processing begins and how each bit position of each LCT byte is used after communication processing has begun.

The following key applies to the LCT layout. One of the terms appears in the lower right hand corner of each bit position.

1(A/B/C) - This bit position must be written with an appropriate value by software before communications processing begins; the value of this bit position affects hardware/firmware operations. Differences between 1A, 1B, and 1C are described below:

1A - This bit position may be modified by software at appropriate times after communications processing has begun.

1B - This bit position may be modified by the main memory program, but not by the CCP, at appropriate times after communications processing has begun.

1C - This bit position normally should not be modified by software after communications processing has begun.

2(A/B/C) - This bit position may be written with an appropriate value by software before communications processing begins. The value of this bit position affects hardware/firmware operations. Differences between 2A, 2B, and 2C are described below:

2A - This bit position may be modified by software at appropriate times after communications processing has begun.

2B - This bit position may be modified by the main memory program, but not by the CCP, at appropriate times after communications processing has begun.

2C - This bit position normally should not be modified by software after communications processing has begun.

3 - This bit position may be used by software as required. The value in this bit position does not directly affect hardware/firmware operations. This bit position may be written with an application-specific value before communications processing begins and/or it may be modified at appropriate times thereafter.

4(A/B/C) - This bit position must be reset to Zero before communications processing begins. Thereafter it is maintained by firmware. Differences between 4A, 4B, and 4C are described below:

4A - This bit position may be read by the CCP at appropriate times after communications processing has begun. The CCP may reset this bit position to Zero, when appropriate.

4B - This bit position may be read by the CCP at appropriate times after communications processing has begun. The CCP should not modify this bit position.

4C - The contents of the bit position should not be consulted by software.

5 - This bit position must be reset to Zero before communications processing begins. Thereafter it is not used.

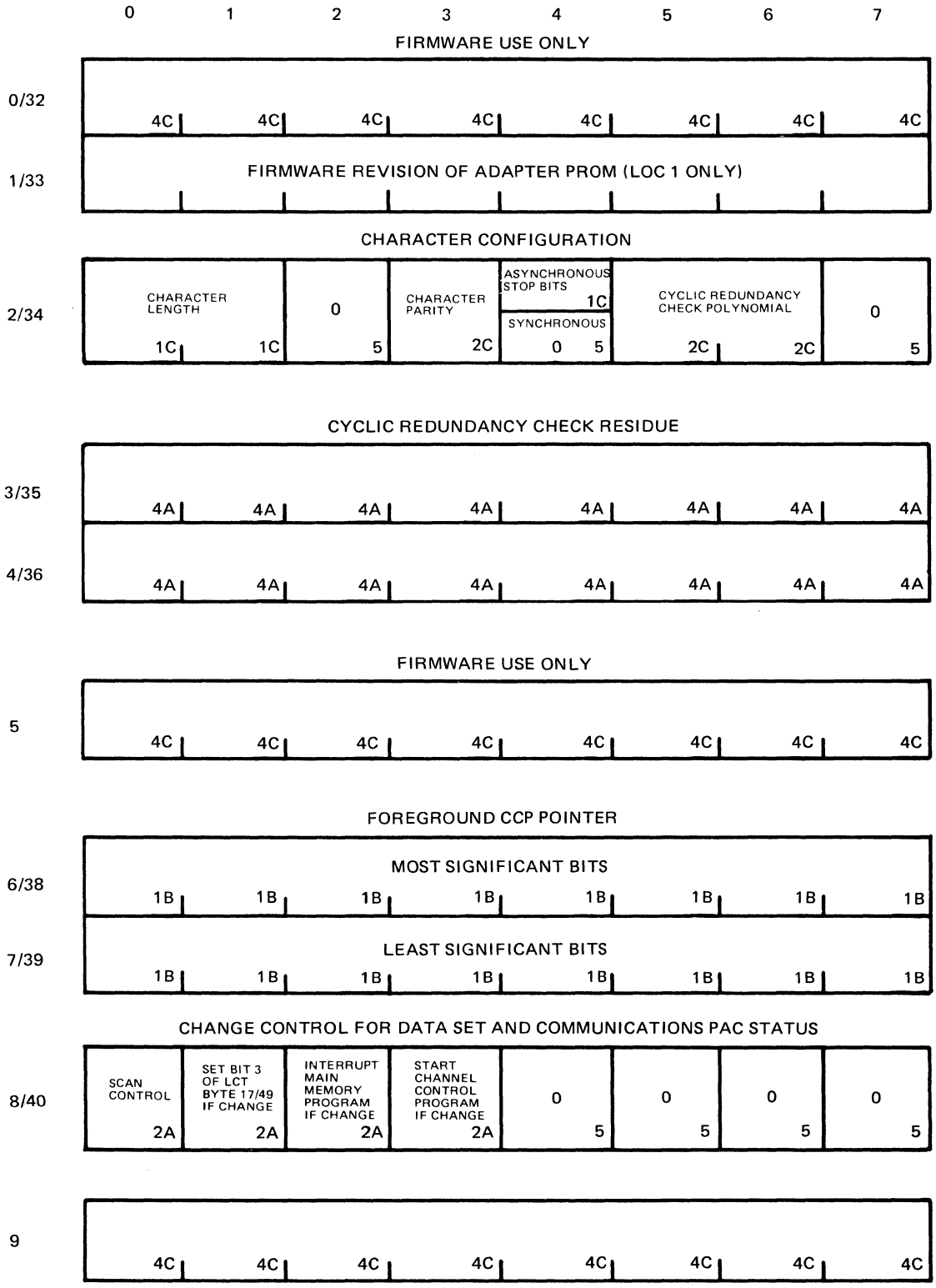


Figure 5-1. Line Control Table Layout

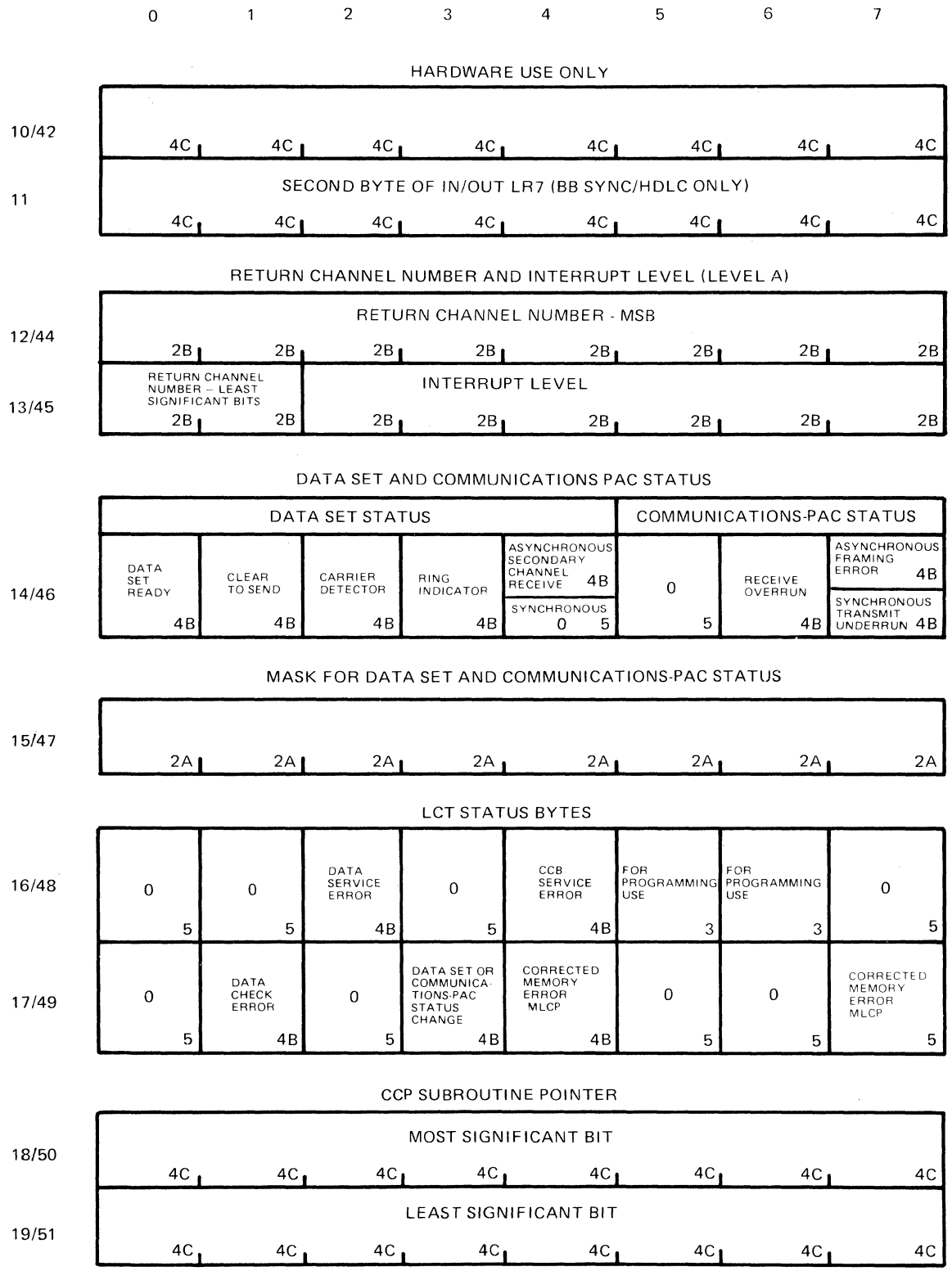


Figure 5-1 (cont). Line Control Table Layout

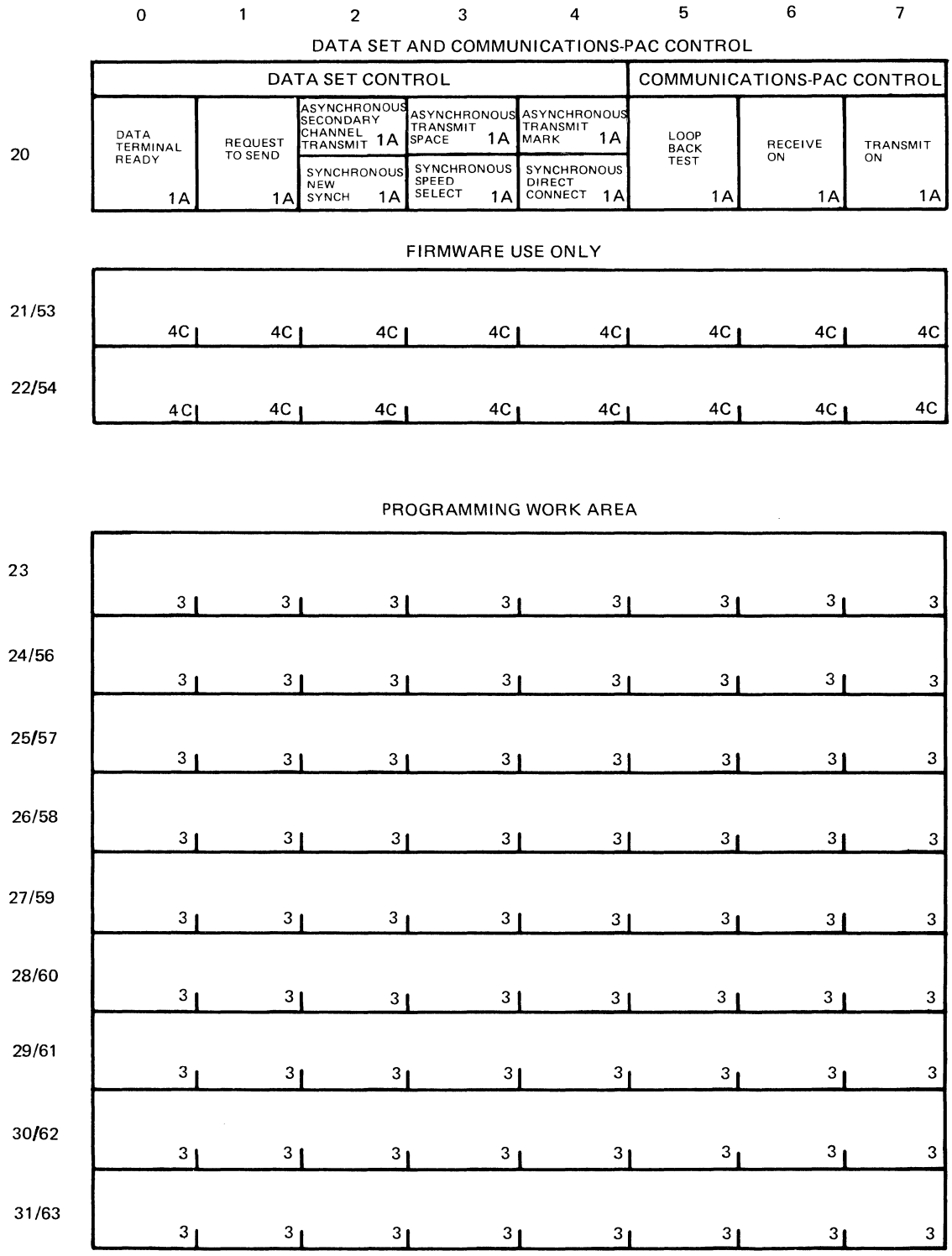


Figure 5-1 (cont). Line Control Table Layout

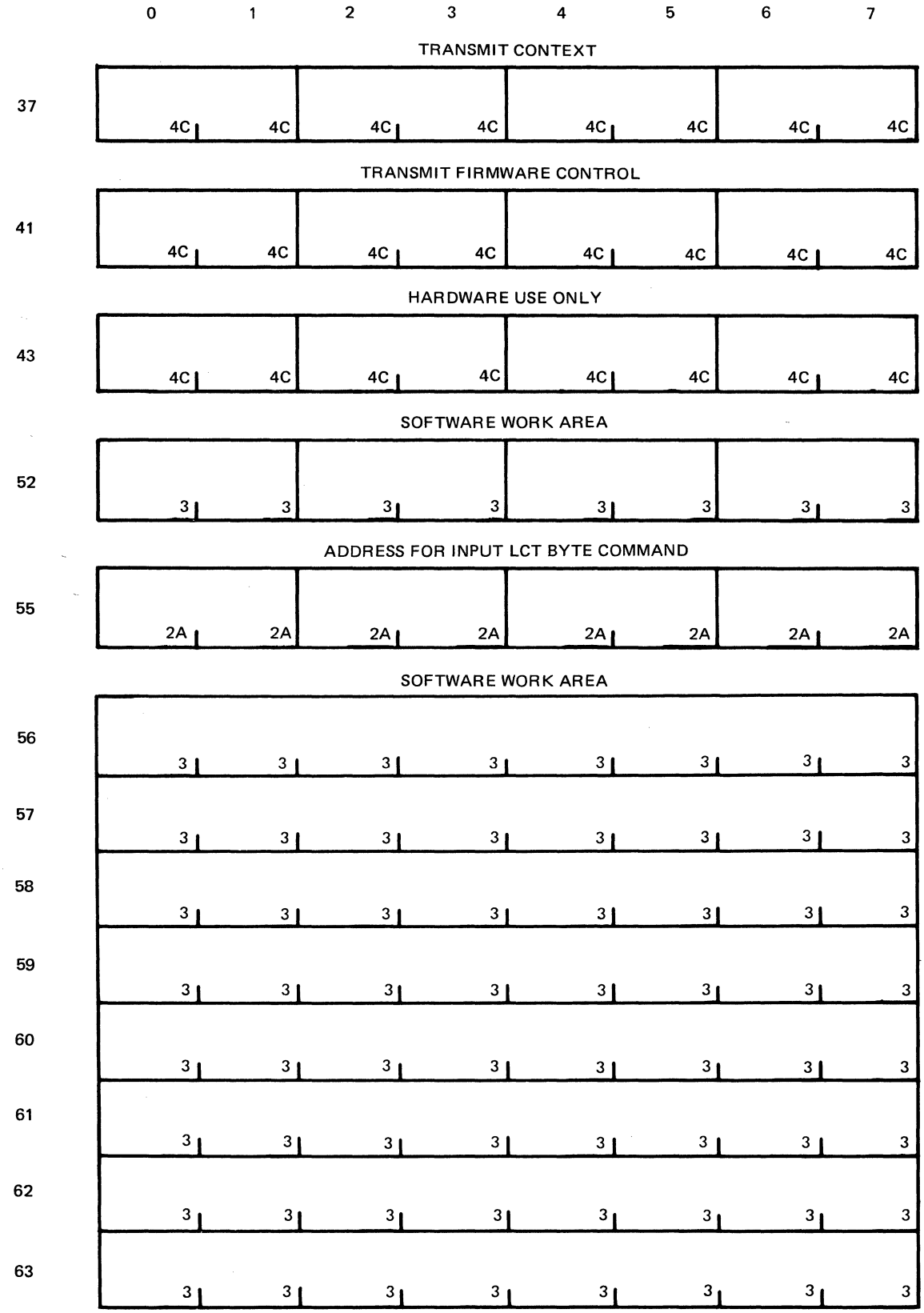


Figure 5-1 (cont). Line Control Table Layout

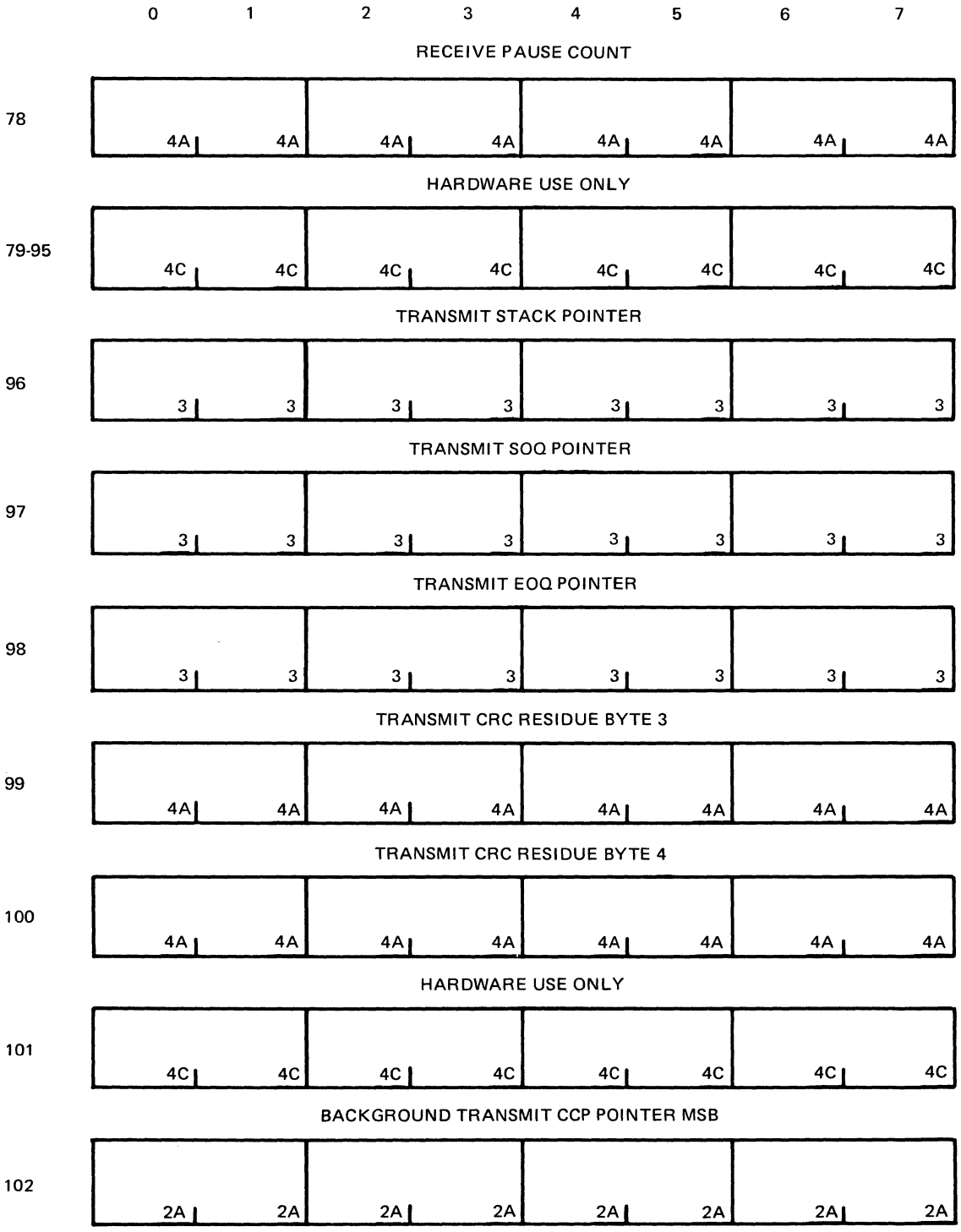


Figure 5-1 (cont). Line Control Table Layout

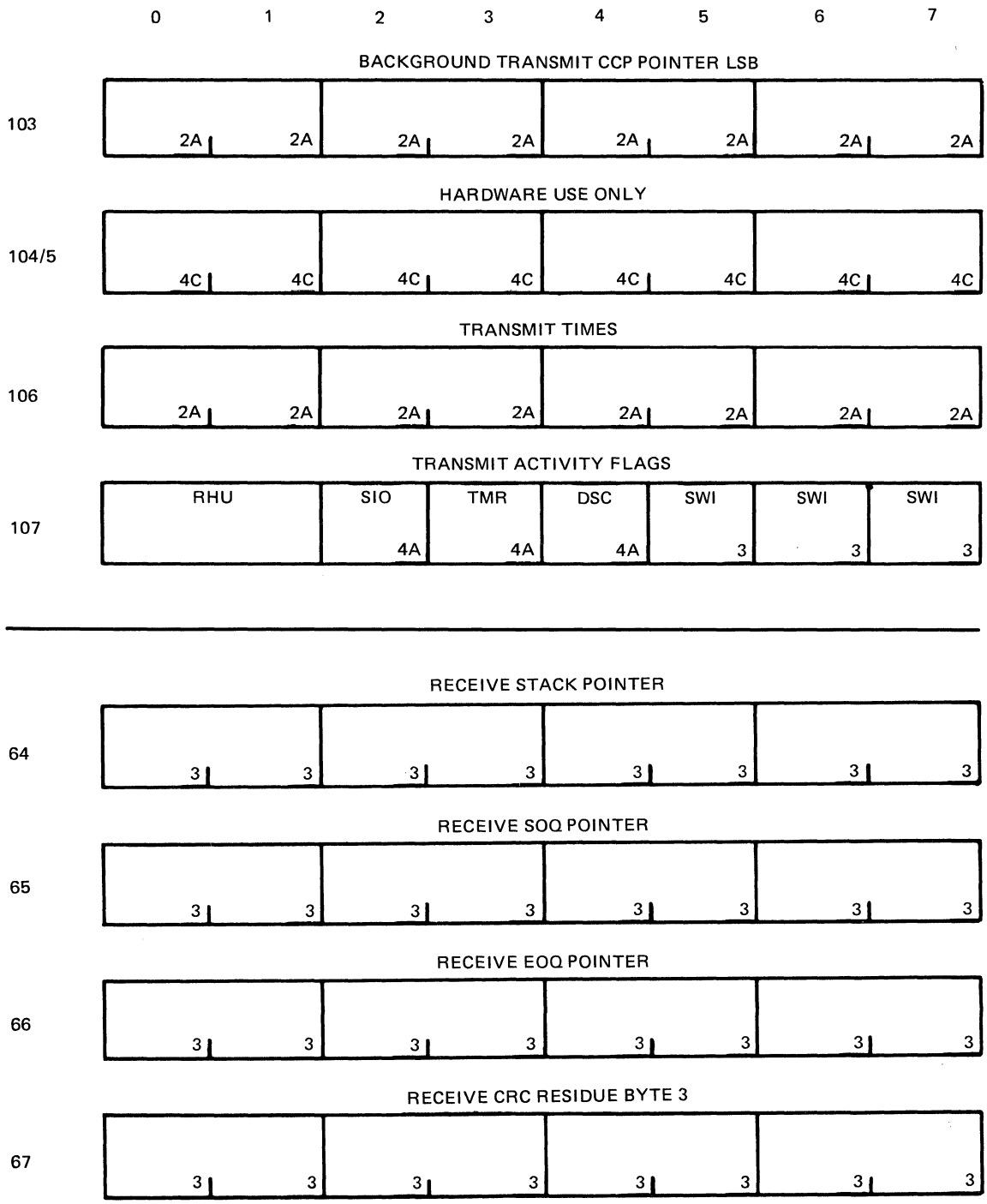


Figure 5-1 (cont). Line Control Table Layout

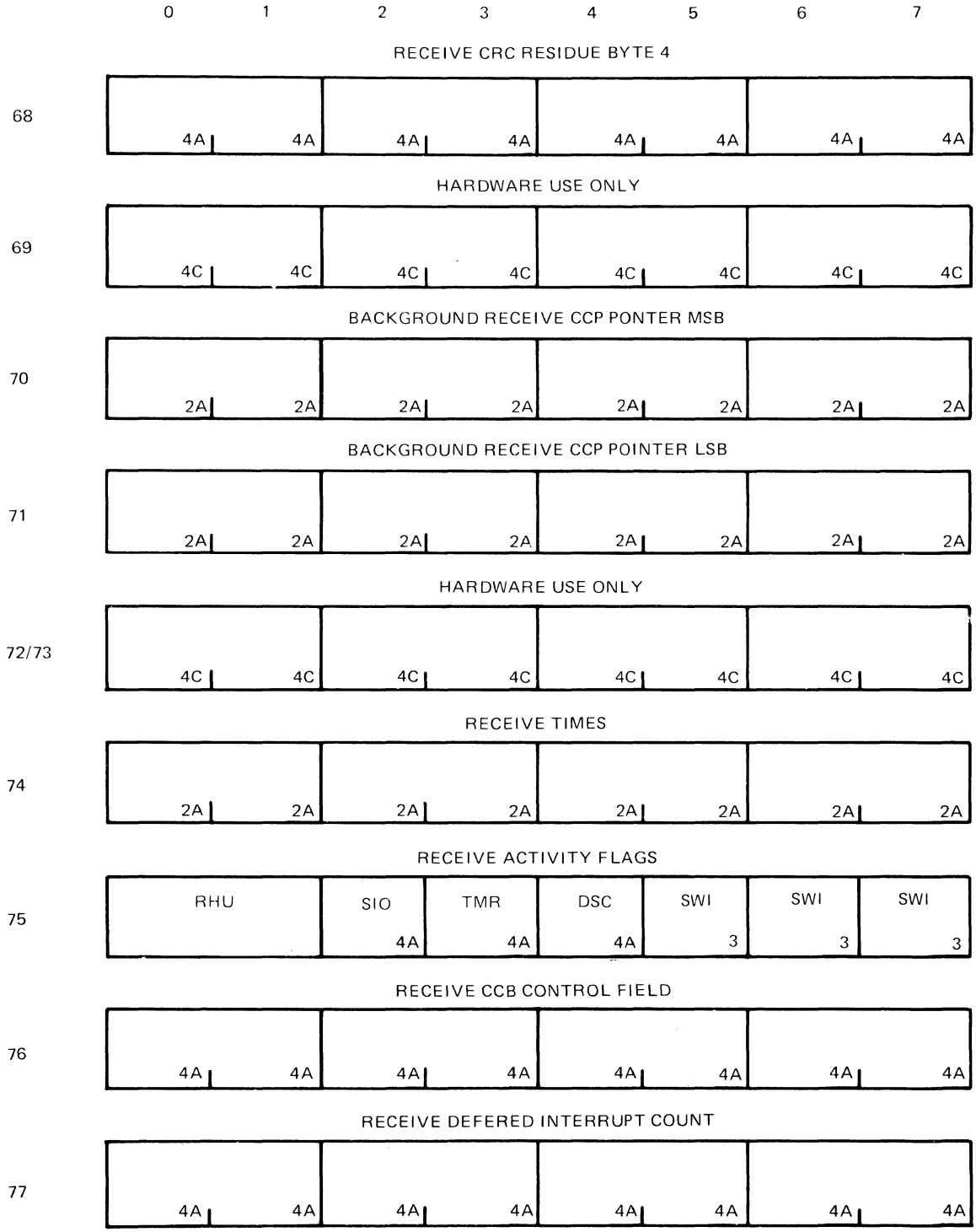


Figure 5-1 (cont). Line Control Table Layout

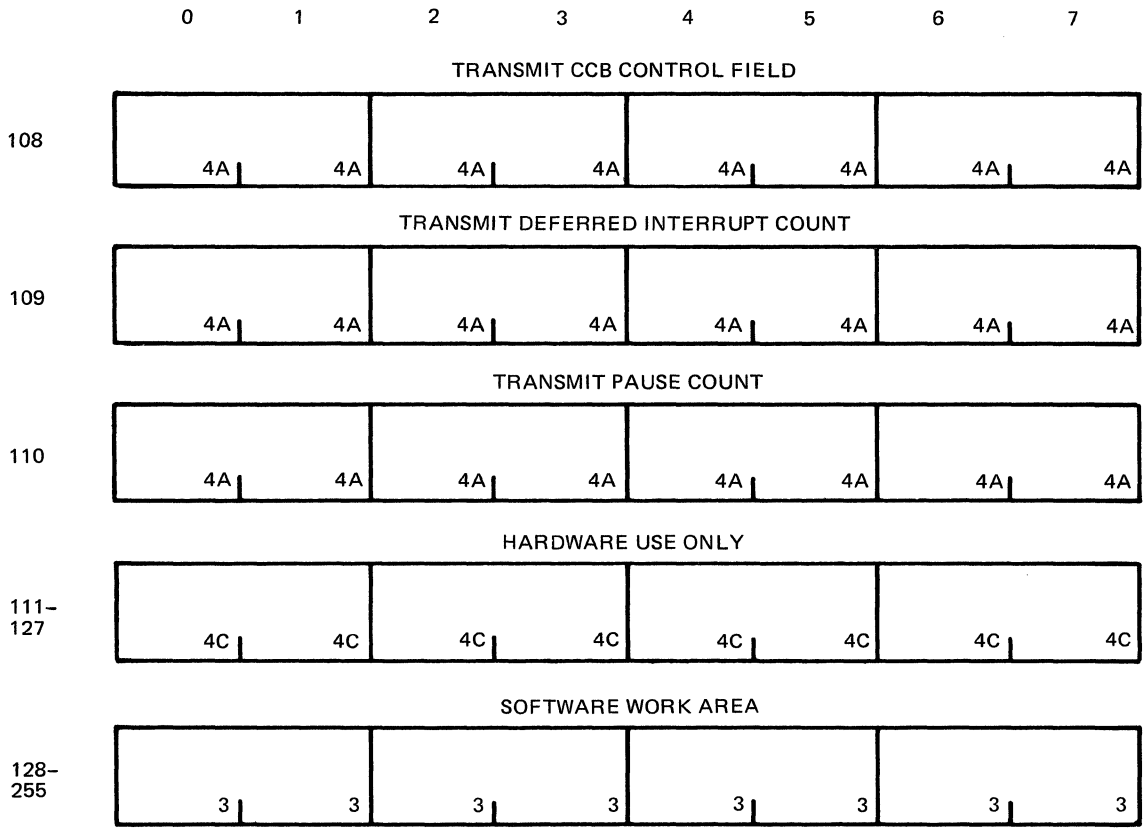


Figure 5-1 (cont). Line Control Table Layout

Section 6
PROCESSOR INTERFACES

This section describes the following subjects, all of which relate to interfaces between the Processor and either the main memory program or data communications equipment/lines:

1. Processor channel number addressing from main memory program
2. Processor interrupts to main memory program
3. Processor control of data sets and line adapters
4. Processor monitoring of data set and adapter status
5. Processor parity checking and generation
6. Processor cyclic redundancy checking
7. Data transfer rates for Processor communications lines
8. FLAP bus interface.

Table 6-1 shows the format of the 10-bit channel number and its relationship to Processor channels and line numbers.

Table 6-1. NMLCP Channel Number Addressing

10-Bit Channel Number in Memory and On Megabus Network	Channel Number	Line Number	Channel Type
xxxxx00000	0	0	Receive
xxxxx00001	1	0	Transmit
xxxxx00010	2	1	Receive
xxxxx00011	3	1	Transmit
xxxxx00100	4	2	Receive
xxxxx00101	5	2	Transmit
xxxxx00110	6	3	Receive
xxxxx00111	7	3	Transmit
xxxxx01000	8	4	Receive
xxxxx01001	9	4	Transmit
xxxxx01010	10	5	Receive
xxxxx01011	11	5	Transmit
xxxxx01100	12	6	Receive
xxxxx01101	13	6	Transmit
xxxxx01110	14	7	Receive
xxxxx01111	15	7	Transmit
xxxxx10000	16	8	Receive
xxxxx10001	17	8	Transmit
xxxxx10010	18	9	Receive
xxxxx10011	19	9	Transmit
xxxxx10100	20	10	Receive
xxxxx10101	21	10	Transmit
xxxxx10110	22	11	Receive
xxxxx10111	23	11	Transmit
xxxxx11000	24	12	Receive
xxxxx11001	25	12	Transmit
xxxxx11010	26	13	Receive
xxxxx11011	27	13	Transmit
xxxxx11100	28	14	Receive
xxxxx11101	29	14	Transmit
xxxxx11110	30	15	Receive
xxxxx11111	31	15	Transmit

xxxxx represents the five bits of the Processor's fixed channel number, and is set by a switch on the Processor. The value is the same for each communications channel of a given Processor.

PROCESSOR CHANNEL NUMBER ADDRESSING FROM MAIN MEMORY PROGRAM

As described in Section 2, whenever the main memory program issues an Input/Output instruction to the Processor it must provide a 10-bit channel number (along with an appropriate 6-bit function code). This information is placed on the Megabus network during execution of the Input/Output instruction. The 10-bit channel number identifies a specific Processor (as distinguished from any other physical unit on the Megabus network) and one of the Processor's 32 communications channels.

The 10 bits of the channel number are divided into two parts: the five high-order bits identifying the Processor's unique fixed channel number (Megabus address), which is set by a switch on the Processor, and the five low-order bits indicating one of the 32 communications channels.

The 10-bit channel number is stored in bits 0 through 9 of a word in main memory or a register (with the function code stored in bits 10 through 15). On the Megabus network, the 10-bit bus channel number occupies bits 8 through 17 of the address portion of the Megabus network (with the function code occupying bits 18 through 23).

PROCESSOR INTERRUPTS TO MAIN MEMORY PROGRAM

Under certain conditions (described below), the Processor can interrupt execution of the main memory program. The interrupt is generated on behalf of one of the Processor's communications channels, each of which can have an interrupt level assigned to it. (The interrupt level for a channel can be unique or it can be shared with one or more other channels.)

A channel can be assigned an interrupt level when its LCT area is set up by a RAM Data Transfer or by a Block Mode Write; alternatively, a channel can be assigned an interrupt level by execution of an IO (Output Interrupt Control) instruction in the main memory program. A channel's interrupt level (together with the return channel number - i.e., the central processor's channel number) is stored in LCT bytes 12 and 13 (for a receive channel) or in LCT bytes 44 and 45 (for a transmit channel). The format of the LCT bytes is described in Section 5.

After a channel has been assigned a nonzero interrupt level as described above, the Processor will interrupt the main memory program on behalf of this channel under any of the circumstances described below. (Refer also to the discussion of CPU interrupts in Appendix A.)

1. At the end of processing relative to a CCB whose control byte (byte 5) has a 1 in bit 0 (interrupt control). (The bit can be set to 1 by an IO (Output CCB Control) instruction in the main memory program.)

2. When a data set or adapter status change is recorded in LCT byte 14/46 and bit 2 of LCT byte 8/40 is set to 1.

3. Upon completion of a Block Mode Read or Block Mode Write.

4. Upon execution of an INTR instruction by a CCP.

5. Upon completion of a RAM data transfer or the occurance of an illegal I/O or CCP instruction.

6. In Extended mode, upon execution of a GIVE instruction on a CCB that had an interrupt set.

7. In Compatibility mode, upon completion of a CCB GNB instruction.

Remember that an interrupt will occur under the above circumstances only if the channel has a nonzero interrupt level when the circumstance occurs.

Note that any of the following actions causes a channel's interrupt level to be reset to Zero:

1. Execution, in the main memory program, of an IO (Output NMLCP Control) instruction that initializes the Processor.

2. Execution, in the main memory program, of an IO (Output Channel Control-Channel Initialize) instruction.

3. Execution, in the main memory program, of an IO (Output Interrupt Control) instruction that specifies a zero value for the interrupt level.

If a CCB is active when the channel's interrupt level is reset to Zero, an interrupt of the main memory program will not occur at the end of processing relative to the CCB, even if bit 0 (interrupt control) of the CCB control byte (byte 5) is set to One.

In those cases where a channel's interrupt level is lower (higher-numbered) than the interrupt level at which the main memory program is currently running, an attempt to interrupt the main memory program will result in a NAK from the central processor. The interrupt will be deferred and the queue for that channel is incremented.

Following a change in the interrupt level at which the main memory program is running, the Processor will retry deferred interrupts during the background firmware scanning periods mentioned in Section 1. Beginning with the lowest numbered channel that has a deferred interrupt queued, the Processor will attempt one interrupt for each such channel before cycling back to the queue for the lowest numbered channel that has a deferred interrupt queued.

PROCESSOR CONTROL OF DATA SETS AND LINE ADAPTERS

For each communications line, the Processor controls the data set (if any) and adapter by means of an adapter control register LR2 (Line Register 2) for that line. (An adapter has a separate LR2 for each line it accommodates.) The value in LR2 affects the operations of both channels of one line.

LR2 of an adapter is loaded from a CCP that services one or both channels of the related communications line. The CCP must first ensure that the desired control value exists in LCT byte 20. Then, at an appropriate time, the CCP places this control value in the R-register and loads it into LR2 of the adapter. The control value is effective immediately.

Refer to the appropriate appendixes for programming considerations for the adapters available.

PROCESSOR MONITORING OF DATA SET AND ADAPTER STATUS

If so directed by the CCP, the Processor will periodically scan data set and adapter status relative to a given communications line, as refelcted in an adapter status register (LR5/FR5) for that line. Whenever the Processor firmware scan detects certain types of status changes (as predefined by the CCP), it will store the entire contents of LR5/FR5 in LCT byte 14 or LCT byte 46. Subsequent actions to be taken are also predefined by the CCP.

Processor scanning of LR5/FR5 takes place by setting to One bit 0 (scan control) of LCT byte 8 or LCT byte 40. By setting appropriate bits in a mask contained in LCT byte 15 or LCT byte 47, the programmer defines types of data set or adapter status changes that cause the entire contents of LR5/FR5 to be copied into LCT byte 14 or LCT byte 46. Appropriate settings of bits 1, 2, and 3 in LCT byte 8 or LCT byte 40 define the subsequent action(s) to be taken, as follows:

1. If bit 1 is set to One, a detected status change will cause bit 3 of LCT byte 17 or LCT byte 49 to be set to One.

2. If bit 2 is set to One, a detected status change will cause the active CCB to be terminated and the main memory program to be interrupted. If the CCP is currently active, the main memory program will continue to run.

3. If bit 3 is set to One, a detected status change will cause the CCP to be started. (This action will be taken only if bit 2 is reset to Zero.)

Each time a firmware scan occurs, the following sequence of actions is performed:

1. An exclusive OR operation is performed on (1) the contents of LCT byte 14 or LCT byte 46 and (2) the contents of LR5/FR5.

2. A logical AND operation is performed on the result from step 1 and the contents of the mask in LCT byte 15 or LCT byte 47.

If a nonzero result is produced by the logical AND operation in step 2, a data set or adapter status change has occurred. The contents of the LR5/FR5 are copied into LCT byte 14 or LCT byte 46 and subsequent action(s) are based on the settings of bits 1, 2, and 3 of LCT byte 8 or LCT byte 40, as described above. (If the CCP is started in response to a status change, it can read LCT byte 14 or 46 to ascertain what type of status change has occurred.) This can only take place for a receive CCP on an incoming call to wake up the receive CCP.

The Processor firmware scan is a low-priority activity that is independent of output operations from the main memory program and CCP processing. The firmware scan will occur at least every one-half second.

PROCESSOR PARITY CHECKING AND GENERATION

A CCP can direct the Processor to check parity during receive operations and to generate parity during transmit operations.

For receive operations, the CCP must first ensure that bit 3 of LCT byte 2 indicates the desired type of parity. Then, as data characters are transferred from the receive channel's LR1 to the Processor's R-register, the operand of each RECV instruction can indicate that a parity check is to be performed. The parity check will include all bits of the character length specified in bits 0 and 1 of LCT byte 2. (In this case, the leftmost bit of each data character is a parity bit; the parity bit may be either 0 or 1 as the data character arrives in the R-register.) If the Processor detects a parity error, it will set to One bit 1 (data check error) of LCT byte 17 (LCT status byte 2).

For transmit operations, the CCP must first ensure that bit 3 of LCT byte 34 indicates the desired type of parity. Then, as data characters are transferred from the R-register to the transmit channel's LR1, the operand of each SEND instruction can indicate that parity is to be generated. (In this case, the leftmost bit of each data character, whose length is specified in bits 0 and 1 of LCT byte 34, is a parity bit; the parity bit must be 0 as the SEND instruction is issued.) The Processor generates the proper value for the parity bit as the data character is transferred to the transmit channel's LR1.

Remember that both channels of one line use the same character length. Both channels of a line share one LR6, which defines character configuration in an adapter.

PROCESSOR CYCLIC REDUNDANCY CHECKING

A CCP can direct the Processor to use its block-check capability to perform cyclic redundancy checking during receive or transmit operations. Normally, CRC is required only for higher speed data transfers.

For receive operations, the CCP must first ensure that bits 5 and 6 of LCT byte 2 indicate the desired cyclic redundancy checking polynomial (see Table 6-2). Then, as data characters are received, the CCP can obtain either unconditional or conditional cyclic redundancy checking:

1. Unconditional cyclic redundancy checking is obtained if the operand of each RECV type 1 or 3 instruction indicates that a check is to be performed. The CRC residue (in LCT bytes 3 and 4) will be updated as each data character is loaded into the Processor's R-register.

2. Conditional cyclic redundance checking is obtained if RECV instructions type 0 or 2 do not indicate that a check is to be performed. Each received data character is analyzed in the R-register and a check is performed as a separate operation. This is accomplished by means of a CCH (Calculate Block Check) instruction only when desired. CRC residue will be updated each time a CCH instruction is executed.

At the end of a received data block, a CRC must be performed on the block-check character(s). Afterwards, the CRC residue should have the value expected. If 12-bit CRC residue has been accumulated, the contents of LCT byte 3 may need to be shifted two bit positions to the right. The SR (Shift R-Register Right) instruction is available for this purpose.

For transmit operations, the CCP must first ensure that bits 5 and 6 of LCT byte 34 indicate the desired CRC polynomial. See Table 6-2. Then, as each data character is transferred from the R-register to the transmit channel's LR1, the operand of each SEND (Send) instruction can indicate that a check is to be performed. The CRC residue (in LCT bytes 35, 36, 99, and 100) will be updated as each data character is transferred into the transmit channel's LR1. At the end of each block, the CCP must transmit the CRC residue accumulated in LCT bytes 35, 36, 99, and 100. If 12-bit CRC residue has been accumulated, the contents of LCT byte 35 may need to be shifted two bit positions to the right before being transmitted. The SR instruction is available for this purpose.

Table 6-2. Cyclic Redundancy Check Information

Bits 5,6,&7 of LCT Byte 2 or Byte 34	CRC Polynomial Used	CRC Residue Bit Length	Configuration of CRC Residue in LCT Bytes 3/35 and 4/36	Transmission Modes
000	$x^{16}+x^{15}+x^2+1$	16	0 7 3/35 [MSB] Must be zero at start. 4/36 [LSB]	IBM BSC CRC-16
010	$x^{16}+x^{12}+x^5+1$	16	0 7 3/35 [MSB] LCT 3/35, 4/36 must be all ones at start. 4/36 [LSB]	HDLC, SDLC, ADCCP, CCITT Recommendation
100	$x^{12}+x^{11}+x^3+$ X^2+X+1	12	012 567 3/35 [MSB ////] Must be zero at start. 4/36 [//// LSB]	IBM Transcode CRC-12
110	x^8+1	8	0 7 3/35 [LRC] Must be zero at start.	LRC-8, Basic Mode ASCII
001	$x^{32}+x^{26}+x^{23}+$ $x^{22}+x^{16}+x^{12}+$ $x^{11}+x^{10}+x^8+x^7$ $+x^5+x^4+x^2+f+1$	32	0 7 Note 4 [ ]	Federal Standard
011	x^6+x^5+1	6	0 7 3/35 [CRC]	CRC-6 (PARS)
101	RFU			
111	RFU			

NOTES

1. On transmit, LCT byte 35 should be sent first.
2. On a receive, LCT byte 3 should be compared with the first CRC byte received.
3. For code 011, only LCT byte 3/35 is applicable. LCT byte 35 must be sent with odd parity.
4. LCT 3, 4, 67, 68/35, 36, 99, 100.

When using the NMLC CRC capability, residue information is maintained in the LCT associated with each channel. The LCR and CRC-6 residues are maintained in LCT 3/35. The CRC-16, CRC-12, and CCITT residues are maintained in LCT 3,4/35,36. The Federal Standard residues are maintained in LCT 3,467,68/35,36,99,100. Preset of the residue, addition to the residue and checking, or transmission are under control of CCP instructions issued at the appropriate time.

CRC residue can be reset to Zero at appropriate times by the CCP or by the main memory program. The main memory program can use the IO (Output LCT Byte) instruction for this purpose.

Whenever a CRC error is detected, it is the channel program responsibility to set LCT 17 bit 1 data check error.

DATA TRANSFER RATES FOR PROCESSOR COMMUNICATIONS LINES

The data transfer rate for a given communications line depends on two factors:

1. The type of adapter with which the line is associated.

2. The mode (normal, direct-connect, or loop-back test) in which the line is being used.

NOTE

Normal mode signifies that the line is connected to the adapter by means of data communications equipment (i.e., the line is not direct-connected) and no loop-back of transmitted data is taking place.

Each channel of the communications line uses the line's data transfer rate. A channel's bit transfer rate (together with the defined length of the data characters) establishes the interval between channel request interrupts for that channel.

Table 6-3 summarizes the relationship between data transfer rate, type of adapter, and mode of line operation. Note that for synchronous direct-connect or loop-back, the line speed is governed in the Processor by the Processor fixed-rate clock and the selected rate applies to all lines that use this clock.

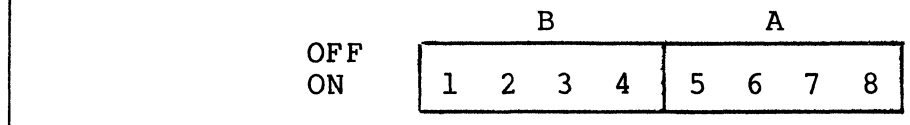
Table 6-3. Data Transfer Related to Adapter Type and Operation Mode

Type of Adapter	Mode of Line Operation	Data Transfer Rate
Asynchronous	Normal Direct-Connect Loop-Back Test	Goverened by clock in line adapter. Speed indicated by settings of bits 4 through 7 of LR4 of adapter.
Synchronous	Normal	Goverened by clock in data set. If the data set clock permits either of two data transfer rates to be used, bit 3 of LR2 can be used to select the desired rate.
	Direct-Connect Loop-Back Test	Goverened by Processor's fixed-rate clock. This clock is set to two rates by hardware configuration; the possible settings are shown in Table 6-4. Table 6-4 shows the rates selected by switch positions 1-4 which apply to lines 0-7. The rates selected by switch positions 5-8 apply to lines 8-15 for the lines that use the Processor's fixed-rate clock.

Table 6-4. Possible Settings for NMLCP's Fixed-Rate Clock

Clock Rate (in Bits per Second)	Switch Position	X = Up 0 = Down
	1234/5678	
800	XXXX	
1,200	XXX0	
1,759	XX0X	
2,152	XX00	
2,400	X0XX	
4,800	X0X0	
9,600	X00X	
19,200	X000	
28,743[a]	0XXX	
31,917[a]	0XX0	
38,400[a]	0X0X	
57,826[a]	0X00	
76,600[a]	00XX	
114,306[a]	00X0	
153,600[a]	000X	
307,200[a]	0000	

A = 5,6,7,and 8 - Lines 8 - 15
B = 1,2,3,and 4 - Lines 0 - 7



[a]Maximum rate depends on adapter type; refer to the appendixes for specific adapters.

Appendix A
PROGRAMMING GUIDELINES FOR SELECTED NMLCP FEATURES

This appendix contains supplementary information on programming certain Processor features or operations, such as initialization, set up, line registers, certain CCP instructions, CPU interrupts, and error recovery. Cross references are made to the text where appropriate.

Special programming techniques specific to adapter type are discussed in the separate appendixes on the adapters. References in this appendix are made to those later appendixes where appropriate.

INITIALIZATION

A complete Processor initialization occurs only on the issuance of an Output NMLCP Control (FC=01). The complete initialization is not affected by a Power On which performs a soft initialization only.

Because the hard initialize affects all channels, it normally occurs only at startup. In normal operation, the Output Channel Control instruction (FC=05) with bit 0 set (channel initialize) is more appropriate because it allows other lines to continue operation. Channel initialize kills both the receive and transmit channels of the communications line because it causes Line Register 2 (LR2) to be cleared. In that sense it can always be assumed that channel initialize on one channel will cause the paired channel to cease operation. The paired channel is not completely initialized, however, because its CCB list has

not been reset. In general, it is best to treat channel initialize as line initialize and issue commands in pairs, one to each channel pertaining to a particular line.

Channel initialize results in the clearing of the LSI chip which performs the serial/parallel conversion at the data set interface, and clears the adapter and FLAP registers for the channel. In order to completely clear the Processor and all adapter and FLAPs a hard initialized command must be issued.

Soft initialize is accomplished by pressing the CLR (Clear) pushbutton on the central processor control panel. This causes the Processor to perform a soft initialize which does the following:

1. Clears the adapters
2. Clears LCT bytes 8, 9, 40 and 41 of each channel
3. Clears the Processor internal registers.

The soft initialize differs from the hard initialize caused by the command Output Processor Control (FC=01) in that the hard initialize, in addition to performing the soft initialize functions defined above, also clears the RAM (software-visible).

When in a software debug mode, it is preferable that the RAM not be cleared so that meaningful dumps can be taken to indicate the Processor condition.

It is preferable to use the Output Channel Control instruction (FC=05) with bit 2 set (stop I/O) or to use the CCP to control LR2 directly in order to cease communications on a line. This causes minimum change to the setup of the Processor relative to the line and simplifies restart of communication.

ADAPTER SETUP

Adapter (Communications-Pac) setup is performed by the CCP between the Output Channel Control instruction (FC=05) with bit 1 set (start I/O) and the first WAIT instruction. In that interval, the following actions must be performed as a minimum:

1. Load LR0 if applicable
2. Load LR6
3. Load LR4 if applicable
4. Load LR2

The information to be loaded in LR6 will be found in the LCT byte 2 for receive and in LCT byte 34 for transmit. The information for LR4 has no specific LCT location assigned and could come from an LCT work location or from the CCP itself as a result of a Load (LD) instruction, Format 3 (load immediate). The information for LR2 will be found in LCT byte 20 for both receive and transmit.

The LCT area must already have been set up by the main memory program. It is the task of the CCP to get that information to the adapter where it will be stored. The specific information to be output to the adapter is unique to each type. There is, however, a sequence in which it must be done. This sequence is configuration-word first and data set control-word last.

LCT byte 20 is one case where the same byte applies to both sides of the line. In most other cases, separate LCT locations exist for transmit and receive. In the case of LCT byte 20 which contains such indicators as request to send, data terminal ready, transmitter on, and receiver on, there can be only one such set of indicators since there is only one LR2 and only one modem.

The number of adapter registers which must be reloaded to set up the adapter depends on what has previously been programmed. At the very first usage of the adapter after power on, all the registers (LR6, LR4, LR2) need to be set up. If a line has been in use and a stop I/O issued, only LR2 need be reloaded to resume communications.

Refer to the later appendixes for the specific information on each adapter.

ACCESS TO LINE AND FLAP REGISTERS

The visibility of the adapters to the Channel Control Program is for control, data access, and status-sensing purposes. This is accomplished through a set of eight line registers (LRs) within the adapter. It is the responsibility of the firmware to map the LR content either into or from the appropriate USART and FLAP registers.

In order to implement more complex DCE interfaces than are capable through LRs, the CCP can directly access the eight FLAP registers (FRs) in the FLAP. These registers implement Data Set Control and Data Set Status functions associated with Data Communication Equipment. With the mapping of LRs and the direct access to the FRs the Processor is capable of supporting existing and new functionality of Data Communication Equipment. The FLAP register bit definitions are also FLAP specific.

An adapter can have as many as eight line registers on the transmit channel and an additional eight on the receive channel; in many, the number is more like six or seven. Some registers are accessible on either channel (e.g., LR2) while other registers are specific to the transmit or the receive channel (e.g., LR1). This type of information is adapter-specific.

In practice, it should be noted that certain LR and FR registers can be written only while others can only be read. (If a line register is used incorrectly or does not exist, the OUT instruction will act as a NOP and the IN instruction will result in invalid information.) In the case of two important line

registers this means that, once they are loaded,the CCP cannot directly determine their contents. The two most critical of these are LR2 and LR6. The programming solution is to make use of the association of LCT location and LR as follows:

LR2 = LCT 20
LR6 = LCT 2 (receive channel)
LR6 = LCT 34 (transmit channel)

LR2, which controls the data set, should be kept identical to LCT byte 20 at all times. If, for example, the contents of LR2 and LCT byte 20 show the transmit on set and the receive on reset and it is desired to reverse this, the proper technique would be:

1. Load LCT 20 into R
2. Set receive on bit
3. Reset transmit on bit
4. Output to LR2
5. Store R into LCT 20.

Within this approach, the state of the data set interface can always be determined by reading LCT byte 20. This is particularly useful in case of error recovery activity.

Concerning LR6, which contains configuration information, the same kind of situation exists. In some instances there is a single LR6 while in other instances there are two. The configuration information does not have to be identical on the receive and transmit channels. In any case, it is important to use the same technique to be sure that LR6 is always identical to the related LCT locations.

CCP INSTRUCTIONS

The following paragraphs are intended as an aid in programming the referenced instructions. This information supplements Section 3.

SEND Instruction

In addition to outputting data and performing checking generation, SEND causes the firmware to OR the contents of bit 7 of LR5 into LCT status, LCT byte 48 bit 2. This status bit is used to designate different things in different Communications-Pacs. For example, in the synchronous line adapter, it is a transmit underrun while in the asynchronous line adapter it is a framing error. The important point here is that LCT byte 48 bit 2 is not necessarily a transmit error but is only a copy of LR5 bit 7. In the case of the asynchronous line adapter, that represents a framing error so that a receive error is being reported on a transmit channel. The transmit channel for that Communications-Pac should reset the status bit before returning to the main memory program. There is no meaningful asynchronous line adapter transmit error. Refer also to Appendix C.

RECV Instruction

In addition to outputting data and performing checking generation, RECV causes the firmware to OR the contents of bit 6 of LR5 into LCT byte 16 bit 2. This status bit may be used to designate different things in different adapters. In most cases it means receive overrun. This status bit is on the receive channel.

OUT LR1 Instruction

OUT LR1 has the same function as SEND except that error status is not ORed from LR5 into the LCT area. Also, CRC and/or parity generation are not possible on OUT instructions. Because of the fact that error status is ignored, OUT LR1 is a means of avoiding conditions where error flags exist relative to the LSI chip performing the serial/parallel conversion.

NOTE

The initial character of a message must be sent with OUT LR1 due to the false underrun reported by the USART at some speeds.

IN LR1 Instruction

IN LR1 has the same function as RECV except that CRC and parity checking are not possible and the error status bit is not copied from LR5 into the LCT area. The fact that the error status is ignored makes IN LR1 a means of avoiding conditions where error flags may be set relative to the LSI chip performing the serial/parallel conversion (see above).

SFS Instruction

SFS causes the adapter to search for one or two synchronization characters by manipulating the receive on bit in LR2. Synchronization is established with the reception of two consecutive sync characters. All succeeding characters are transferred to the Processor. When SFS is executed, the firmware takes the following actions:

1. Turns off the receive in the adapter
2. Turns on the receive in the adapter
3. ORs the receive bit On in LCT 20.

Note that the states of other bits in LR2/FR2 are not affected.

BLBT, BLBF Instructions

The last block indicator in the CCB control byte is designed so that the main memory program can notify the CCP when the last block of a multiblock message has arrived so that some

terminating activity can take place. With this purpose, the BLBT or BLBF may be used at or near the end of the last block, possibly after deciding to end the block but before doing a GNB.

For the Processor, the last block indicator in the CCB control byte can be tested by the CCP with the BLBT and BLBF instructions. The last block indicator in the CCB is copied by firmware, during a GNB instruction, into a firmware use only location in the LCT area. BLBT and BLBF operate on that bit, not on the bit in CCB.

LD and ST Instructions

In order for an LD/ST to execute properly, there mus tbe a valid CCB. If this is not the case and an LD/ST is attempted the instruction is executed as a WAIT or NOP respectively. Refer to Valid CCBs for details on avoiding this problem. In addition, if there was no valid CCB, when a CCB is finally set up, a CCB service error will be indicated in bit 4 (LCT byte 48 bit 4) CCB status.

WAIT Instruction

The WAIT instruction, which has an execution time of 9.4 microseconds, actually consists of two parts. One part is a context save for the CCP which issued the WAIT. When a channel request interrupt next occurs and the CCP is reactivated, the remainder of the WAIT (a context restore) occurs. (Refer to Section 1 for a definition of channel request interrupt.)

For the Processor, one of the functions performed by the firmware during the context restore is to read the contents of LCT byte 2 (LCT byte 34) and load the character size and checking specification into the CRC hardware in the Processor.

BLCT, BLCF Instructions

The use of the BLCT or BLCF instructions to test for end-of-range is mandatory. The Processor will not permit DMA transfer to occur beyond the boundaries of the data area defined in a CCB.

INTR Instruction

INTR clears LCT 16 and then sets only bit 0 of LCT 16. The following sequence may be used to preserve the contents of LCT 16.

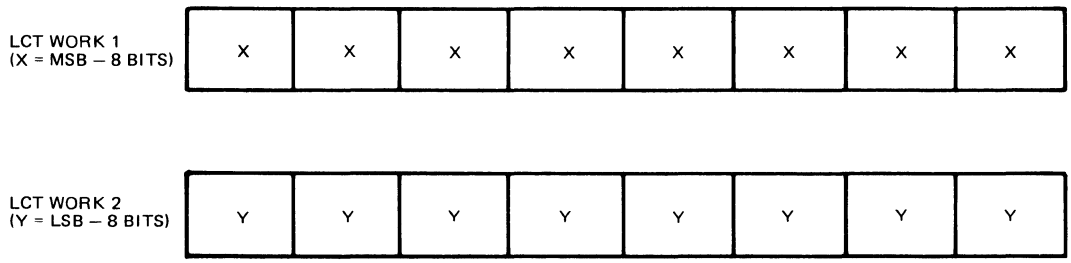
```
|
|
|
LD 16
 INTR
OR 16
|
|
|
```

TABLE LOOK-UP INSTRUCTION PROGRAMMING EXAMPLE

NOTE

This program is a sample of one of several techniques that can be used for Table Look-Up (TLU). Note that the MORG techniques used in the sample program can be used only if the TLU instruction starts on an even boundary.

The TLU operand specifies the first LCT location of a pair of consecutive LCT work locations. These two locations contain a 16-bit RAM address pointer to the beginning of the TLU data table.



The 16-bit address must be within the channel program portion of the RAM,. It is the responsibility of the main memory program to load the correct address of the TLU data table into the corresponding LCT locations specified by the TLU instruction. This can be done by either a block write or an output LCT byte instruction to each of the LCT WORK locations.

Having resolved the location of the TLU data table, the TLU now adds the contents of the Processor R-register previously loaded by the channel program. The resulting address now points to a specific byte in the TLU data table.

Bit 0 of the specific TLU data table byte is tested to determine if action requested is a data translation (bit 0 = 0) or an indexed branch (bit 0 = 1).

1. Data Translation (Bit 0 = 0)

   The contents of the specific TLU table byte is loaded into the R-register overwriting the previous value. The control returns to the instruction immediately following the TLU.

   The maximum table size for the TLU is 2^8 bytes (256); however, only a 7-bit translation is possible because of the use of bit 0 as an indicator. Refer to the sample program shown in Figure A-1.

```
      *TLU TABLE FOLLOWS
      *CHARACTERS 00-0F
200           DATA      X'00',X'82',X'82',X'8B',X'82',X'05',X'06',X'82',X'82',X'09',X'0A', X'82',X'0C',X'82',X'82',X'0F'
      *CHARACTERS 10-1F
210           DATA      X'82',X'11',X'12',X'82',X'14',X'82',X'82',X'17',X'18',X'82',X'82', X'1B',X'82',X'1D',X'1E',X'82'
      *CHARACTERS 20-2F
220           DATA      X'82',X'21',X'22',X'82',X'24',X'82',X'82',X'27',X'28',X'82',X'82', X'2B',X'82',X'2D',X'2E',X'82'
      *CHARACTERS 30-3F
230           DATA      X'30',X'82',X'82',X'33',X'82',X'35',X'36',X'82',X'82',X'39',X'3A', X'82',X'3C',X'82',X'82',X'3F'
      *CHARACTERS 40-4F
240           DATA      X'82',X'41',X'42',X'82',X'44',X'82',X'82',X'47',X'48',X'82',X'82', X'4B',X'82',X'4D',X'4E',X'82'
      *CHARACTERS 50-5F
250           DATA      X'50',X'82',X'82',X'53',X'82',X'55',X'56',X'82',X'82',X'59',X'5A', X'82',X'5C',X'82',X'82',X'5F'
      *CHARACTERS 60-6F
260           DATA      X'40',X'82',X'82',X'43',X'82',X'45',X'46',X'82',X'82',X'49',X'4A', X'82',X'4C',X'82',X'82',X'4F'
      *CHARACTERS 70-7F
270           DATA      X'82',X'51',X'52',X'82',X'54',X'82',X'82',X'57',X'58',X'82',X'82', X'5B',X'82',X'5D',X'5E',X'82'
      *CHARACTERS 80-8F
280           DATA      X'82',X'86',X'89',X'82',X'8D',X'82',X'82',X'07',X'08',X'82',X'82', X'0B',X'82',X'0D',X'0E',X'82'
      *CHARACTERS 90-9F
290           DATA      X'10',X'82',X'82',X'13',X'82',X'15',X'90',X'82',X'82',X'19',X'1A', X'82',X'1C',X'82',X'82',X'1F'
      *CHARACTERS A0-AF
2A0           DATA      X'20',X'82',X'82',X'23',X'82',X'25',X'26',X'82',X'82',X'29',X'2A', X'82',X'2C',X'82',X'82',X'2F'
      *CHARACTERS B0-BF
2B0           DATA      X'82',X'31',X'32',X'82',X'34',X'82',X'82',X'37',X'38',X'82',X'82', X'3B',X'82',X'3D',X'3E',X'82'
      *CHARACTERS C0-CF
2C0           DATA      X'40',X'82',X'82',X'43',X'82',X'45',X'46',X'82',X'82',X'49',X'4A', X'82',X'4C',X'82',X'82',X'4F'
      *CHARACTERS D0-DF
2D0           DATA      X'82',X'51',X'52',X'82',X'54',X'82',X'82',X'57',X'58',X'82',X'82', X'5B',X'82',X'5D',X'5E',X'82'
      *CHARACTERS E0-EF
2E0           DATA      X'82',X'41',X'42',X'82',X'44',X'82',X'82',X'47',X'48',X'82',X'82', X'4B',X'82',X'4D',X'4E',X'82'
      *CHARACTERS F0-FF
2F0           DATA      X'50',X'82',X'82',X'53',X'82',X'55',X'56',X'82',X'82',X'59',X'5A', X'82',X'5C',X'82',X'82',X'5F'
      :::::::::
      :::::::
      :::::
      ::::

      *EXAMPLE OF THE USE OF THE TLU INSTRUCTION FOR A RECEIVE CCP
      *ALL DATA RECEIVED IS EVEN PARITY
              LOC       NEXT      INPUT NEXT CHARACTER ENTRY POINT
300           WAIT                WAIT FOR NEXT CHARACTER TO COME IN
301           RECV      2         CHAR AVAILABLE,CHECK EVEN PARITY

      *ADDRESS IN LCT 23,24=TLU DATA TABLE,WAS LOADED BY MAIN PROGRAM
              LOC       TLU1      ADDRESS OF TLU INSTRUCTION
302,303       TLU       23        TABLE LOOK-UP
304           ST        ,         TRANSLATE COMES HERE SHIP R TO MAIN MEMORY
305,306       BLCF      NEXT      IF NOT LAST CHARACTER GET NEXT ONE

      *LAST CHARACTER, GO TO MESSAGE FOR LONG ERROR
307,308,309   JUMP      LONG      3 BYTE JUMP
              MORG                INSURE THAT NEXT ADDRESS IS EVEN DISPLACEMENT
      *ENTRY FOR BAD PARITY DETECTED
              LOC       TARG1     ENTRY POINT
30A           ST        ,         STORE IT AS IS
30B,30C       LD        25        GET PARITY ERROR COUNT
30D           DEC                 COUNT 2S COMPLEMENT
30E,30F       ST        25        PUT PARITY COUNT BACK
310,311       B         NEXT      GO TO NEXT CHARACTER
              MORG                INSURE EVEN DISPLACEMENT
      *ENTRY FOR SOH =X'81'
              LOC       TARG2     ENTRY POINT
312,313       AND       =X'7F'    STRIP PARITY BIT
314,315,316   JUMP      SOH1      GO TO PROCESS SOH
              MORG                INSURE EVEN ENTRY POINT
      *ENTRY FOR STX X'82'
              LOC       TARG3     ENTRY POINT
318,319       AND       =X'7F'    STRIP PARITY
31A,31B       B         STX1      GO TO PROCESS SIX
              MORG                INSURE EVEN DISPLACEMENT
      *ENTRY FOR ETX X'03'
              LOC       TARG4     ENTRY POINT
31C,31D,31E   JUMP      ETX1      GO TO PROCESS ETX
              MORG                INSURE EVEN DISPLACEMENT
      *ENTRY FOR EOT X'84'
              LOC       TARG 5    ENTRY POINT
320,321       AND       =X'7F'    STRIP PARITY BIT
322,323,324   JUMP      EOT1      GO TO PROCESS END OF TRANSMISSION
              MORG                INSURE EVEN DISPLACEMENT
      *ENTRY FOR SYN X'96' EVEN PARITY SYNC
              LOC       TARG6     ENTRY POINT
326,327       LD        26        GET SYNC COUNT LOCATION
328           DEC                 COUNT TWOS COMPLEMENT
329,32A       ST        26        PUT COUNT BACK
32B,32C       B         NEXT      IGNORE SYNC GET NEXT CHARACTER
              MORG                INSURE EVEN DISPLACEMENT
```

TLU TABLE FOR SAMPLE PROGRAM OF TLU USAGE

1. UNCONDITIONAL BRANCH ON THE FOLLOWING CONDITIONS
   - A. ODD PARITY IS PARITY ERROR =X'82'
   - B. SOH =X'81' EVEN PARITY START OF HEADER
   - C. STX =X'82' EVEN PARITY START OF TEXT
   - D. ETX =X'03' EVEN PARITY END OF TEXT
   - E. EOT =X'84' EVEN PARITY END OF TRANSMISSION
   - F. SYN =X'96' EVEN PARITY SYNC CHARACTER
2. TRANSLATE LOWER CASE 60-7F TO UPPER CASE 40-5F
3. STRIP PARITY BIT

Figure A-1. Sample Table Look-Up Program

2. Indexed Branch (Bit 0 =1)

The contents of the specific TLU table byte is not loaded into the Processor R-register. The R-register retains the initial contents through this mode of the TLU.

The address of the indexed branch is generated as follows:

a. The channel program sequence counter currently pointing to the LCT displacement byte of the TLU instruction is incremented by 3 bytes. This spaces the sequence counter over the 2-byte translation branch following the TLU. This allows a minimum of one branch instruction for processing translated characters.

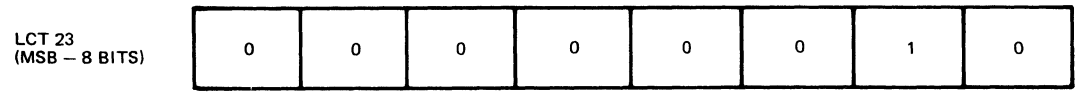
b. The specific TLU table byte, bits 1-7, are multiplied by a factor of 2 because a short displacement branch requires 2 bytes. This allows a table of branches to direct action to the proper routines. The result is then added to the sequence counter causing it to point to a specific branch or macro which is coded inline and after the TLU instruction. The processing continues from this point under control of the channel program.

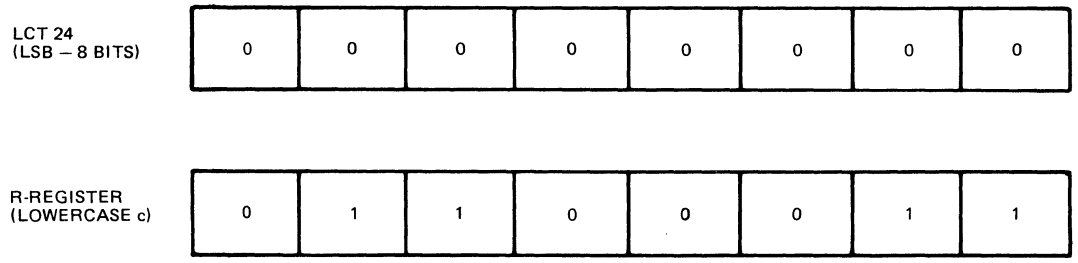
The TLU data table can contain a mixture of translation data and indexed branch data. The following example illustrates how the indexed branch option can be used to recognize escape characters and bad parity while translating to other values including stripping the parity bit.

The TLU instruction makes it very easy for the channel program to convert from or to the native ASCII code to any other 7-bit or less code. Thus the Processor can be fluent in any language without affecting system performance.

Example 1: Translate lowercase c X'63' to uppercase C X'43'. (Refer to Figure A-1).

Instruction: Convert TLU 23 lowercase ASCII to uppercase ASCII





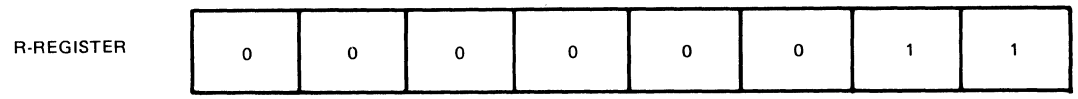
Explanation

The TLU table base in LCT 23, 24 (X'0200') is added to the contents of the R-register (X'63') which generated a TLU data table address of X'0263'.

The contents of location X'0263' is X'43' and since the value X'43' has bit 0 = 0, a translation is required. The value X'43' is loaded into the R-register and control returns to X'0304' the instruction immediately following the TLU.

The original value of X'63' has been translated to X'43'. The channel program then transfers the X'43' to main memory, branches if last character is false and waits for the next character to come in.

Example 2: Indexed branch on ETX (refer to Figure A-1). LCT 23, 24 Equal X'200' (as in Example 1)



Explanation

The TLU table base in LCT 23, 24 (X'200') is added to the contents of the R-register (X'03') which generates a TLU data table address of X'203'.

The contents of this location is X'8B' and bit 0 = 1. The TLU switches into indexed branch mode and the CCP takes the following action.

a. The R-register is preserved and remains at X'03'.

b. The address of the TLU instruction itself is already in the P-counter (X'303') and points to the LCT displacement byte of the TLU instruction being executed. The value 3 is added to the P-counter making P = X'306'. This constant of 3 is added to allow a branch out to process a translation as in Example 1.

c. The value of bits 1-7 (X'0B') of the data extracted from the TLU table is then multiplied by 2. This is done to allow a table of branch instructions which require 2 bytes minimum per branch (2(X'0B')=X'16').

d. The previous value X'16' is added to the contents of the P-counter which is X'306'. The new P-counter value is now the target address of the branch (X'31C') and processing continues at this location.

e. The first instruction at X'31C' is a long displacement jump to process the ETX1 (P-counter +3 +2* (TLU data bits 1-7)=Target).

NOTE

The programmer must generate the branch information contained in the TLU table to match the indexed branch entry points coded after the TLU instruction. In the example, the table value was calculated as follows:

$$\frac{\text{TARGET ADDRESS} - \text{ADDRESS OF SECOND BYTE OF TLU INSTRUCTION} - 3}{2} = \frac{(X'31C' - X'303') - 3 = X'B'}{2}$$

Bit 0 was set to One to indicate unconditional branch and the value placed in the table of position X'03' = X'8B'.

Each target area must have an even number of bytes. This is why the MORG instruction is inserted after each target. This instruction will fill with an NOP to make an even number of bytes. The programmer can also directly insert an NOP. If changes were made in the target area, the target area indexes affected have to be recalculated.

The DATA statement can be used to program the target area indexes to adjust automatically to any changes. The target index used in this example could have been calculated by the assembler if the following DATA statement had been substituted for the (X'8C') at location X'203'.

```
DATA(TARG4-TLU1-X'04)/X'02'+X'80'
```

The assembler would evaluate this expression as follows:

$$\frac{(X'31C'-X'302'-X('04')}{2} +X'80'=X'8B'$$

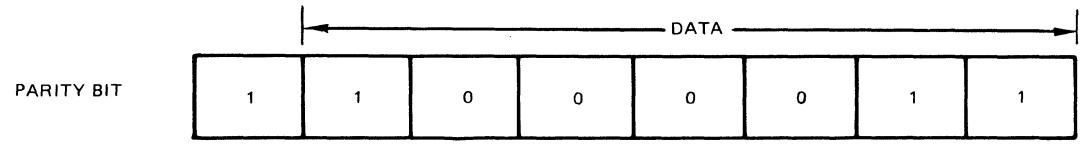
NOTE

Refer to the DATA statement for rules governing internal value expressions.

Example 3: Branch on odd parity.

Same as Example 2. Unconditional branch, except all cases of odd parity in the table have been replaced by X'82' which causes a branch to the TARG1 for bad parity at X'30A'. The bad parity condition can then be programmed as desired.

Example 4: Strip parity bit (R-register = X'C3').



The above value is added to the table base of X'200' for a result of X'23C'.

The contents of X'2C3'=X'43' bit 0=0, therefore a translation is required, the X'43' is loaded into the R-register, and processing continues at location X'304'. The parity bit has, in effect, been stripped off.

NOTE

The previous examples illustrate the potential of the TLU instruction for normal synchronous data received processing. There are many other uses including user status evaluation, error code processing, decoding action codes passed by the main memory program, etc.

VALID CCBs

When a CCB is set up by the main memory program, the Output CCB Control instruction (FC=0F) must have bit 1 set (valid bit). When a start I/O is issued for a channel or an LCT byte 8/40 (data set scan) starts the CCP, the firmware takes no specific notice of the valid bit. A CCB service error is deemed to exist if either an LD or an ST is executed and there is no valid CCB. After using the CCB, the firmware resets the valid bit when a GIVE/GNB instruction is executed.

The CCP instructions BVBT and BVBF enable the CCP to branch on the valid bit condition. Whether or not this instruction is necessary in a specific application depends on the way the program is established.

In one case, a number of CCBs are set up and the last is marked with a last block indicator. The program does BLBT or BLBF at the end of each block and if that is the end, the call is terminated.

In a more sophisticated case the software sets up a number of CCBs and does not mark any of them as last block. The software intent is to set up new CCBs as a response to the CCB interrupts and always keep ahead of the ability of the Processor to use up CCBs. The advantage of this mode of operation is that it can lead to very high line utilization. A danger is that the main memory program may not keep up. In order to avoid a fatal error, the first instruction following the GNB should be a BVBT or BVBF so that the lack of a valid block can be detected before any damage is done. If no valid block exists, special action could be taken.

DATA SET SCAN

The data set scan can be activated by setting bit 0 of LCT byte 8 (LCT byte 40) to One. The data set scan will scan either the contents of LR5 or FR5, depending on the state of LCT 8 bit 4. When a scan occurs, the firmware ANDs the contents of LR5/FR5. With the contents of LCT byte 15 (LCT byte 47), ANDs the contents of LCT byte 14 (LCT byte 46), and compares the results for any change. As a result of the scan, the contents of the LR5/FR5 replace the previous contents of LCT byte 14 (LCT byte 46). If a change occurred in the designated data set status bit(s), the change is indicated by the setting of bit 3 of LCT byte 17 (LCT byte 49) which is designated as data set status change. In addition, several other actions could take place:

1. The CCP may be started if bit 3 of LCT 8 (LCT 40) is ON.

2. The CPU may be interrupted if bit 2 of LCT 8 (LCT 40) is ON. If bits 2 and 3 are both ON, bit 2 will take precedence.

If the data set scan is being used when a CCP is not active, then it is reasonable to either start a CCP or interrupt the CPU. If a CCP is active, there is no obvious advantage to the data set scan. It may be more convenient to enable the data set scan so the CCP can check data set status in the LCT area as compared to checking LR5/FR5 directly. It is not clearly any faster one way or the other. Usually, data set status need only be checked at end of message to ensure that nothng has changed.

It is possible to use the data set scan to start the CCP even if the CCP is active. This is because the scan only occurs when the CCP is not running so no conflict is possible. When the CCP starts, however, there is a problem because the CCP would have to determine why it had been activated (channel request interrupt or data set scan). This decision would slow down the normal character processing loop enough to make this mode undesirable in most cases. This problem does not occur if forground-background CCP processing is being used since Data Set Scan starts always invoke Background mode.

It is also possible to use the data set scan to interrupt the CPU while a CCP is active. This, however, can lead to confusion concerning input status (see CPU Interrupts).

CPU INTERRUPTS

The Processor provides two interrupt levels for use by each channel. The prime level is designated as interrupt A, and is used to report the completion of data channel transfers on the communication line. The second level is designated interrupt B, and is used to report various other occurrences such as data set scan, RAM data transfer completions, illegal I/O orders, illegal instructions, and software generated interrupts via the INTR instruction. For compatability, unless interrupt B is specifically set on A channel, only one interrupt level (level A) will be used for all occurrences on the channel. In this case occurrences such as data set scan and INTR instructions, are reported in CCB status and not interrupt B status.

CPU interrupts are caused by the following three actions in the Processor:

1. Block termination (GNB with I bit set) (see Note below)
2. Data set scan
3. INTR instruction.

NOTE

The I bit is bit Zero of the CCB control byte.

A data set scan interrupt (status change) sets bit 11 of LCT status (specifically, bit 3 of LCT byte 17/49) (see Note below).

NOTE

If data set status changes and bit 2 of LCT byte 8/40 is set, bit 1 of LCT byte 16/48 will be set. There will be no termination of the current CCB. When the main memory program executes an Input Status (code 1C) or Input Next CCB Status (code 1A) instruction, the CCB status will be zero. The LCT status (LCT byte 16/48) should then be read. If bit 1 is set, a data set scan interrupt has occurred.

An INTR instruction sets bit 0 of LCT status (bit 0 of LCT byte 16).

As a response to either of these interrupts, the software could read LCT status directly via the Input LCT byte instruction (FC=1E). When a GNB occurs, LCT status is copied into the CCB status and LCT status is cleared. When a CCB completes, status bit 3 of CCB status is set. If an interrupt was generated on that CCB, bit 1 of CCB status is set. If only CCBs and related interrupts are being used, the Input Next CCB Status instruction (FC=1A) and Input CCB Status instruction (FC=18) are used.

It becomes difficult for the software when CCBs are used in conjunction with data set scan interrupts or INTR instructions because of the three choices of input status commands available. In practice, rarely would the various causes of interrupts be used together so that it is generally obvious which I/O command to issue in response to an interrupt. Some of the more typical cases will be described.

CCBs as Cause of CPU Interrupts

If CCBs are the only cause of interrupts, the software response to an interrupt is Input Next CCB Status instruction (FC=1A). If that status is complete and bit 1 is set, the cause of the interrupt has been detected. If bit 1 is not set, the operation can be repeated until a CCB is found with bit 1 set. This is the most common usage of interrupts. In this case the assumption is that some CCBs were output by the software with I bits and some without (i.e., in a message spanning several blocks an interrupt would not be required on every block). Refer to the previous discussion of the GNB cause of interrupt.

One usage of the INTR instruction in conjunction with CCBs is in the case of a receipt of a long message of unknown length. In this case it is not efficient for the main memory program to set I bits in CCBs, not knowing which was to be the end. For this situation, an alternate method is to output CCBs without I bits and use the INTR instruction followed by a GNB to cause an interrupt based on a CCP-detected event (e.g., EOM).

Data Set Scan as Source of CPU Interrupts

If a data set scan is the only source of interrupts, the software response to an interrupt would be to check LCT status. This would be the case, for example, in an auto answer mode of operation.

Combination of CCP Interrupts and INTR in Debug

Another usage is modifying CCP by the addition of INTR instructions to create breakpoints. In this case, at certain points of significance in the CP, a branch(es) would be added, pointing to an INTR instruction. Following the INTR, an INZ would be used to freeze activity for software analysis via a dump routine. The software response to an interrupt would be as in the case of a CCB causing a CPU interrupt (see previous discussion) except when the Input Next CCB Status indicated an incomplete CCB. That response would be a signal to the software that this interrupt was not CCB-related but INTR-related. A check of LCT Status would then show bit 1 set.

More complex usage of the three types of interrupts are possible. The major point to be made is that it can be difficult to determine the cause of the interrupt if they are intermixed too freely in one application.

DEFERRED INTERRUPT QUEUE

When the Processor has reached an event which requires a CPU interrupt, that interrupt is sent via the Megabus network regardless of other prior responses of the CPU to interrupts. This particular interrupt may be accepted (ACK) or rejected (NAK), depending on the CPU level at that time. If an interrupt is rejected, that event is noted by the Processor and the interrupt is retried when the retry interrupt Megabus signal occurs. The Processor maintains a count of the number of deferred interrupts on a per-channel basis in LCT 77/109. When the retry interrupt Megabus signal occurs, that event is noted and deferred interrupts are resubmitted by the firmware in Background mode. Firmware in Background mode scans the channels in turn and sends an interrupt for each channel that has a non-zero count for deferred interrupts. If the interrupt succeeds, the count is decremented by one. Only a single pass through the channel is made for each retry interrupt Megabus signal.

The presumption in the Processor is that the major source of deferred interrupts is CCBs which have completed, but for which the CPU has not yet taken the interrupts.

The CPU can alternatively read a channels deferred interrupt count via a read and clear LCT order (FC=0E). By reading the register, the CPU program can be made aware of interrupts outstanding on a channel without the necessity of waiting for each interrupt.

ADDRESSING LIMITS

Implicit in the Processor architecture are certain addressing limits which subject the CCP to definite restrictions.

CCB Area Only Implicitly Accessible

The CCP cannot access the CCB area. All the required functions which are required are embodied in the Processor instruction set. In particular, these are the total complement of instructions relating to the CCB area:

LD (Next Character)	BLBT
ST (Next Character)	BLBF
BVBT	GNB
BVBF	GIVE
BLCT	RHB
BLCF	CANC
	CANB

Inability of One Line to Access Another

The CCP may access the entire LCT area for a line. In that sense, the two CCPs of a line may communicate with each other via an LCT location. There is, however, no addressing mode which permits a CCP for one line to access another line in any way. Any line-to-line communication must be by way of the main memory program or through a common extended LCT area in Local Store.

NEED FOR PAD CHARACTERS

On the transmit side of the line, the adapter has an 8-bit register (LR1) which receives the character from the Processor and an additional 8-bit shift register between LR1 and the line. The shift register is loaded from LR1 and shifted serially out to the line. There is, therefore, a delay between the loading of LR1 and the time the last bit of that character clears the adapter and gets physically onto the line. In the synchronous adapter, that delay can be as much as 2-1/8 characters. In the asynchronous adapter,the delay can be a maximum of two characters.

In contrast, the setting or clearing of a bit in LR2 causes that function to occur immediately. One can, for example, output the last character of a message to LR1 and then turn off the transmitter before that character clears the adapter, thereby truncating the message. The bits in LR2 which cause this kind of problem are adapter-specific:

1. Asynchronous Adapter

   Request to Send
   Transmitter on
   Transmit Mark
   Transmit Space

2. Synchronous Adapter

   Request to Send
   Transmitter on

In order to avoid problems when making a change in one of the above at the end of a transmission, the general rule is to follow the last character with pad characters. For the synchronous adapter, three pad characters are recommended. For the asynchronous adapter, pad characters are only required if Request to Send will be turned off at the end of the message (half-duplex operation). In this case, use two pad characters. The pad characters provide sufficient delay before the output to LR2 so that the real end-of-message may get onto the line before the command to LR2 takes effect. An all ones pad character is recommended. The actual number of pad characters which get onto the line will vary, depending on the load on the Processor at the time.

(Other adapters tend to have the same general characteristics.)

TWO-WAY ALTERNATE OPERATION

When operating in Two-Way Alternate mode, the transmitter must be turned off and a request to send dropped at the end of transmission to condition the modem for reception of the message from the other end of the line. The use of pad characters is required in this case as discussed in the preceding paragraphs. In addition, another difficulty relates to some of the earlier modems (201, 202) which tie the transmit and receive lines together so that everything that is being transmitted is being received.

Another difficulty is the fact that the received message may include some of the pad characters output at the end of the transmit operation. Defining a SOM character and having the CCP discard everything until that point is one solution. The other solution is to have the CCP check all received characters and discard all pads up to the first non-pad character.

Refer to Appendix B for further detail on the adapter receive overrun condition.

ERROR HANDLING

The following subjects, all of which relate directly or indirectly to error handling, are described:

1. Conditions under which the Processor will issue a NAK
2. LCT status bytes
3. CCB status bytes
4. Block mode read
5. Abnormal CCP Termination.

Conditions Under Which Processor Will Issue a NAK

Under the following conditions, the Processor will issue a NAK in response to an Input/Output instruction from the main memory program:

1. An Input/Output instruction is issued before Processor initialization of a recent IO (Output NMLCP Control) instruction.

NOTE

Exception: If a second IO (Output NMLCP Control) instruction is issued to initialize the Processor, the Processor will not issue a NAK. Instead, execution of the first instruction is terminated and execution of the second instruction begins.

2. An IO (Output CCB Control) instruction has moved the load CCB pointer to the CCB one beyond the status CCB and an IOLD (Output CCB Address and Range) instruction is attempted before an IO (Input Next CCB Status) instruction is executed. The Processor will issue a NAK in response to the IOLD (Output CCB Address and Range) instruction.

3. After CCB list initialization, an IO (Input Next CCB Status) instruction is attempted before the first CCB is set up.

4. An IO (Input Next CCB Status) instruction has moved the status CCB pointer to the CCB one behind the load CCB and another IO (Input Next CCB Status) instruction is attempted before the load CCB is set up.

After the Processor issues a NAK, processing in the main memory program continues with the next sequential instruction. The main memory program can use a BIOF (Branch if Input/Output Indicator False) instruction to branch back to the Input/Output instruction that caused the NAK. However, only a limited number of attempts should be made to reissue this Input/Output instruction since a closed (two-instruction) loop between the Input/Output instruction and a BIOF instruction would be endless if a NAK were always returned.

LCT Status Bytes

While a given CCB is active, Processor firmware uses two bytes of the related channel's LCT to accumulate certain status and error information. LCT bytes 16 and 17 are used for a receive channel and LCT bytes 48 and 49 are used for a transmit channel. The format of these bytes is shown in the diagram below. These bit positions that are written with information from the LCT status bytes are shown shaded. A detailed discussion of each significant bit position appears under CCB status field in Section 4.

	0	1	2	3	4	5	6	7
LCT BYTE 16/48 STATUS BYTE 1	0 SEE NOTE 1	0 SEE NOTE 2	DATA SERVICE ERROR	0	CCB SERVICE ERROR	FOR PROGRAMMING USE	FOR PROGRAMMING USE	0
LCT BYTE 17/49 STATUS BYTE 2	0	DATA CHECK ERROR	0	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE	CORRECTED MEMORY ERROR	0	0	CORRECTED MEMORY ERROR (MLCP = 0)

NOTES

1. This bit is set by the firmware when the CCP issues the INTR instruction.

2. This bit is set when the data set status change causes an interrupt.

Processor firmware can update the LCT status bytes throughout the time a given CCB is active. (The firmware sets a bit when the related condition is detected; while the CCB is active. The firmware never resets to Zero any bit of the LCT status bytes.)

As soon as processing ends relative to the given CCB, the contents of the LCT status bytes are combined with other information and transferred to the CCB status field (described in the following subsection). The contents of LCT byte 16/48 (LCT status byte 1) is part of the information moved to byte 7 of the CCB. The contents of LCT byte 17/49 (LCT status byte 2) is part of the information moved to byte 6 of the CCB. Immediately after the LCT status bytes are used to update the CCB status field, they are reset to Zero, pending activation of the next CCB for the same channel.

While a given CCB is still active, the LCT status bytes can be read by the main memory program through the use of the IO (Input LCT Byte) instruction. During this time, the CCP can read the LCT status bytes through use of the format 2 LD instruction. The CCP can manipulate bits 5 and 6 of LCT byte 16/48 for application-specific purposes; this technique can be used for passing information from the CCP to the main memory program (which can later read these bit positions from the CCB status field). These bit positions are shown shaded in the diagram.

CCB Status Bytes

As soon as processing ends relative to a CCB, its status field (CCB bytes 6 and 7) is updated by Processor firmware and is said to be meaningful. The CCB status bytes are written with information transferred from the LCT status bytes combined with other information. Bit 3 of CCB byte 7 is set to One to indicate that the CCB status bytes are meaningful. Bit 0 of CCB byte 7 will be set to One if the CCP issued the INTR instruction. (Refer to the LCT byte 16/48 diagram above.)

The CCB's status field can be read from the main memory program by means of an IO (Input CCB Status) or IO (Input Next CCB Status) instruction.

The format of the CCB status bytes is shown in Section 4 of this manual. Those bit positions that are written with information from the LCT status bytes are shown shaded. Note that CCB status byte 1 is stored above CCB status byte 2; this order is the opposite of the order of the status bytes in the LCT. A detailed description of each significant bit position appears under "CCB Status Field" in Section 4.

	0	1	2	3	4	5	6	7
CCB BYTE 6 BYTE 7 STATUS BYTE 2	0	DATA CHECK ERROR	CCB NONZERO RANGE RESIDUE	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE	CORRECTED MEMORY ERROR	INVALID MEMORY ADDRESS	MEGABUS PARITY ERROR	UNCORRECTED MEMORY ERROR
CCB BYTE 7 BYTE 6 STATUS BYTE 1	0 SEE NOTE	INTERRUPT MAIN MEMORY PROGRAM	DATA SERVICE ERROR	CCB STATUS COMPLETE	CCB SERVICE ERROR	FOR PROGRAMMING USE	FOR PROGRAMMING USE	0

NOTE

This bit is set by the firmware when the CCP issues the INTR instruction. (Refer to the LCT byte 16/48 description above.)

Appendix B
NEW MULTILINE CONTROLLER SYNCHRONOUS/ ASYNCHRONOUS ADAPTER

INTRODUCTION

This appendix defines the New Multiline Controller Synchronous/Asynchronous Adapter (NSAA). The adapter is packaged on a quarter-size board which attaches to the New Multiline Communications Processor (NMLCP). The NSAA supports up to four communication lines operating in any mixture of full or half-duplex communications. The protocol on any line can be synchronous, asynchronous, or isochronous.

Maximum line speed is 19.2K bps. Synchronous communication at data rates in excess of 19.2K bps, or operations in accordance with HDLC protocol, is supported by other adapters in the Processor family.

Connections to line equipment (DCE or DTE) are accomplished via Flexible Line Adapter Packages (FLAPs) and FLAP bus cables. The NSAA has a connector for one FLAP bus cable.

SOFTWARE OVERVIEW

A requirement in the development of the NMLCP is that existing CCPs written for use with the ACLA or SCLA on the MLCP are, on an object code basis and without need for change, usable for operating the NSAA in asynchronous or synchronous modes. New CCPs written for the NSAA for asynchronous or synchronous operation should operate without change if and when future NSAA versions using different hardware are implemented.

The prime visibility of adapters on the MLCP to the CCP for control, data access, and status-sensing purposes is that of a set of Line Registers (LRs). In the NSAA, this same visibility and assignment of LRs and their contents will be maintained and will continue to be maintained in future NSAA versions. Firmware will map the LR content into/from the appropriate spots in the NSAA and FLAP.

Newly visible to the CCP is a set of FLAP Registers (FRs) located in the FLAP associated with each line. These registers implement data set control and data set status functions associated with the Data Communications Equipment (DCE). Adapters used on the MLCP did not use FLAPs but instead implemented the data set control and status functions associated with EIA RS-232C type DCEs through bit positions in the LRs.

In the NMLCP/NSAA configurations, firmware will automatically map LR bits to FR register bits on OUT CCP instructions, and will assemble FR bits on IN CCP instructions as appropriate.

Associated with each Universal Synchronous/Asynchronous Receiver/Transmitter (USART) in the NSAA is a set of registers for control and status indication of the USART itself. Adapters on the MLCP used USARTs which had limited capability and flexibility. All required control and status was provided by bit positions in the LRs. Firmware automatically provides mapping of these LR bits into and from the appropriate USART registers. In order that existing CCPs may be used with the NSAA, necessary access to USART registers not represented in the LRs must not require CCP instructions. This also allows future replacement of the USART component with a newer type without impacting the viability of existing CCPs. USART control and status sensing required to be done on a dynamic basis (e.g., error sensing, error reset, etc.) will be performed automatically by firmware. Thus, replacement of the USART with a newer type would impact firmware, but would not affect the CCP.

Figure B-1 shows the register sets associated with the NSAA and their means of access by software.

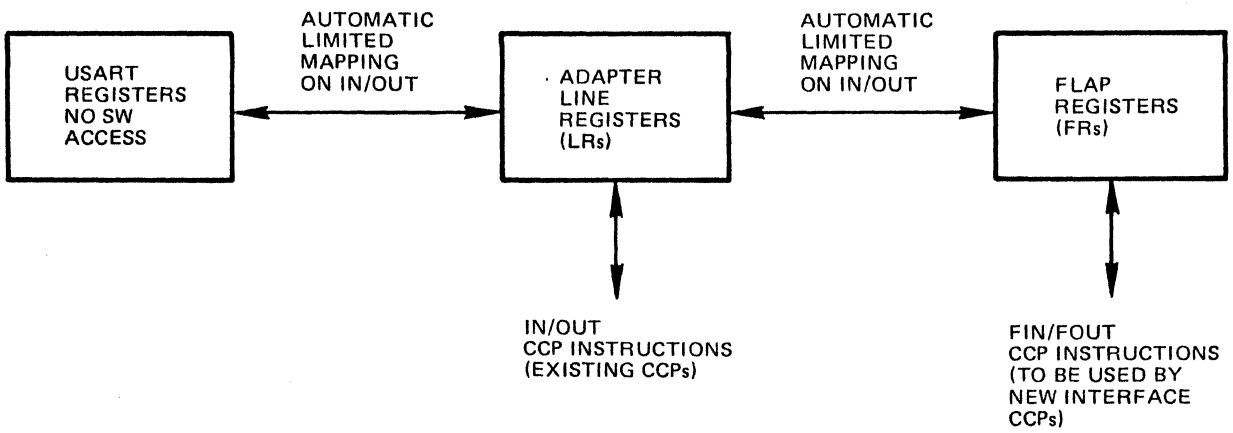


Figure B-1. NSAA Register Access

In order to implement more complex DCE interfaces (e.g., MIL-STD-188C) than were supported on the MLCP, new CCPs must be written directly addressing the FLAP registers. Since the FLAP concept is expected to be retained in future designs, these new CCPs should continue to be usable.

To summarize, software visibility to the NSAA in the case of existing CCPs is exclusively through the LRs. Firmware loads certain default values into the USART registers upon initialization. These are values needed for USART operation which are not supplied via the LRs. Additionally, firmware maps LR bits to appropriate bit positions in the USART registers and vice versa. With the initial values plus subsequent LR-to/from-USART register mapping, existing CCPs and drivers will operate without modification. It will be necessary to add new device IDs to the CLM since the IDs returned by the NSAA, synchronous and asynchronous, will differ from those returned by the SCLA and ACLA.

New functionality over and above that of the MLCP is implemented by the utilization of LR bits hitherto unused. As mentioned above, new CCPs written to capitalize on this functionality will continue to be visible since FLAPs will be retained in future designs and the new functionality specified in Section 3 will survive any USART changes.

DATA FORMATS

Synchronous Format

Any line on the NSAA may be configured for character-oriented synchronous format including bisynchronous or Binary Synchronous Communication (BSC).

All data is transmitted as a serial stream of binary digits. The receiving station operates in step with the transmitting station through the recognition of a specific bit pattern (sync pattern) at the beginning of each tranmsission. The character consists of 5, 6, 7, or 8 data bits plus parity, if used. The data is transmitted least significant bit first (see Figure B-2).

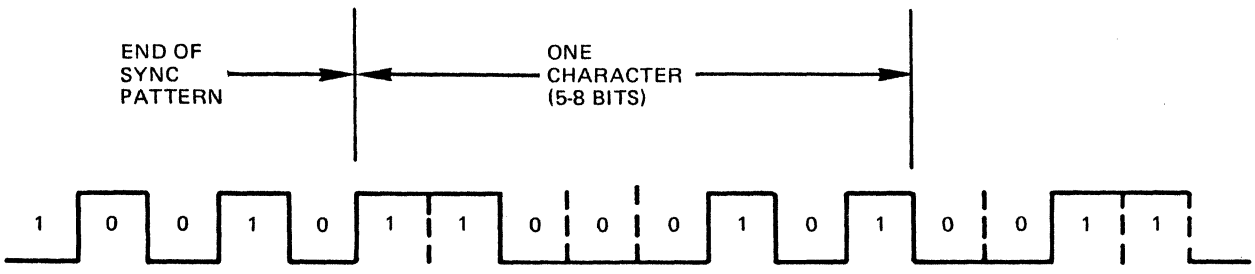


Figure B-2. Synchronous Data Format

The general line format for a block of data is as shown in Figure B-3. During idle periods in transmission within or between blocks, synchronization characters are transmitted as time fill. The Block Check Character (BCC) may be based on either a Longitudinal Redundancy Check (LRC) or Cyclic Redundancy Check (CRC). The Block Check character is provided by, or used by, the Processor.

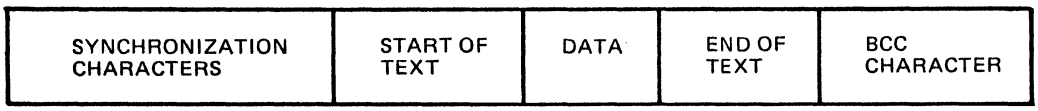


Figure B-3. General Synchronous Line Format

The format used in the IBM Programmed Airlines Reservation System (PARS) is similar to the synchronous format shown above. Character size is 6 bits. Characters are transmitted in complemented form, most significant bit first. The sync pattern consists of two specific characters. Block checking uses a special CRC polynomial. The Channel Control Program (CCP) of the Processor used for PARS support obviously will differ from that of other synchronous protocols. However, the NSAA will synchronize this PARS sync pattern on Receive, and provide the pattern on Transmit.

Asynchronous Format

The asynchronous data format of the NSAA is basically asynchronous bit serial with character synchronization using framing (start/ stop) bits. The character consists of 5, 6, 7 or 8 data bits plus parity, if used, plus a start bit and one or two stop bits. The data is transmitted least significant bit first. During idle periods in transmission, a continuous stop bit (marking) is transmitted. The data format is shown in Figure B-4.

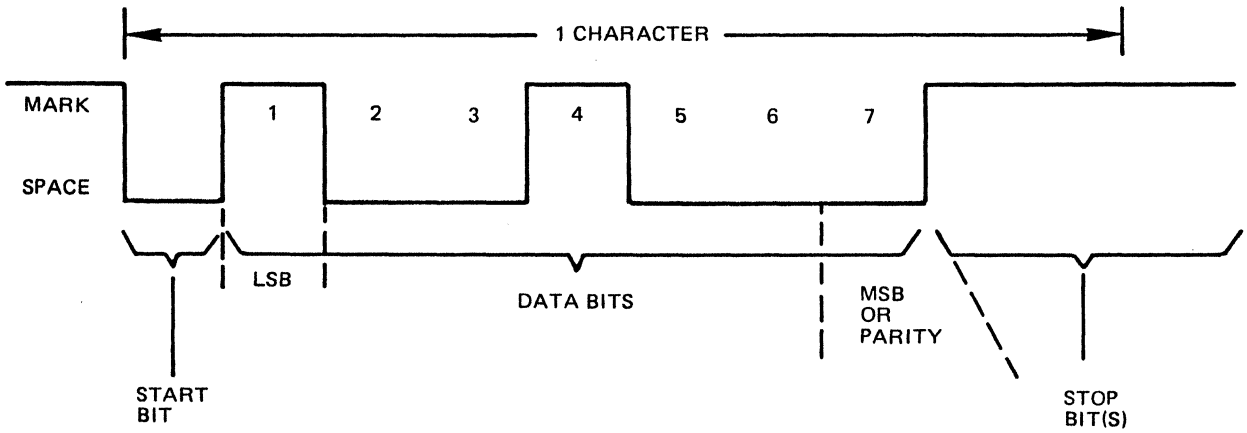


Figure B-4. Asynchronous Data Format

Isochronous Format

Isochronous format is, within a character, identical to asynchronous format. Bit transitions are synched to a clock provided by the DCE rather than the NSAA baud rate generator as in the asynchronous case. Further, transmission characters can be contiguous; i.e., no gaps between characters. To software, isochronous operation is identical to asynchronous.

HARDWARE OVERVIEW

A typical communications subsystem using NMLCP-family products consists of an assemblage of components connected through a set of interfaces as shown in Figure B-5. This appendix addresses one of a set of adapters which can be connected to the Processor, namely, the NSAA. Refer to the appropriate appendix for information pertaining to the Megabus network, Processor, FLAP bus and FLAPs. The primary NSAA functional components are illustrated in Figure B-6.

The NSAA supports four lines, full or half duplex, at rates up to 19.2K bps. Each line may be separately configured as to speed, data format, character size, error control, etc.

ACLA/SCLA Functionality

The NSAA provides the total functionality of the ACLA (asynchronous operation) or the SCLA (synchronous operation). Visibility to software (via Line Registers) is identical to the ACLA or SCLA, as applicable.

Since the NSAA must be capable of operation in either asynchronous or synchronous mode, a switch is associated with each line on the adapter, one position indicating asynchronous mode, the other synchronous. The position of the switch defines the Default mode upon initialization, and thereby allows firmware to provide a unique response to software's Device ID command - a requirement for compatibility with existing software.

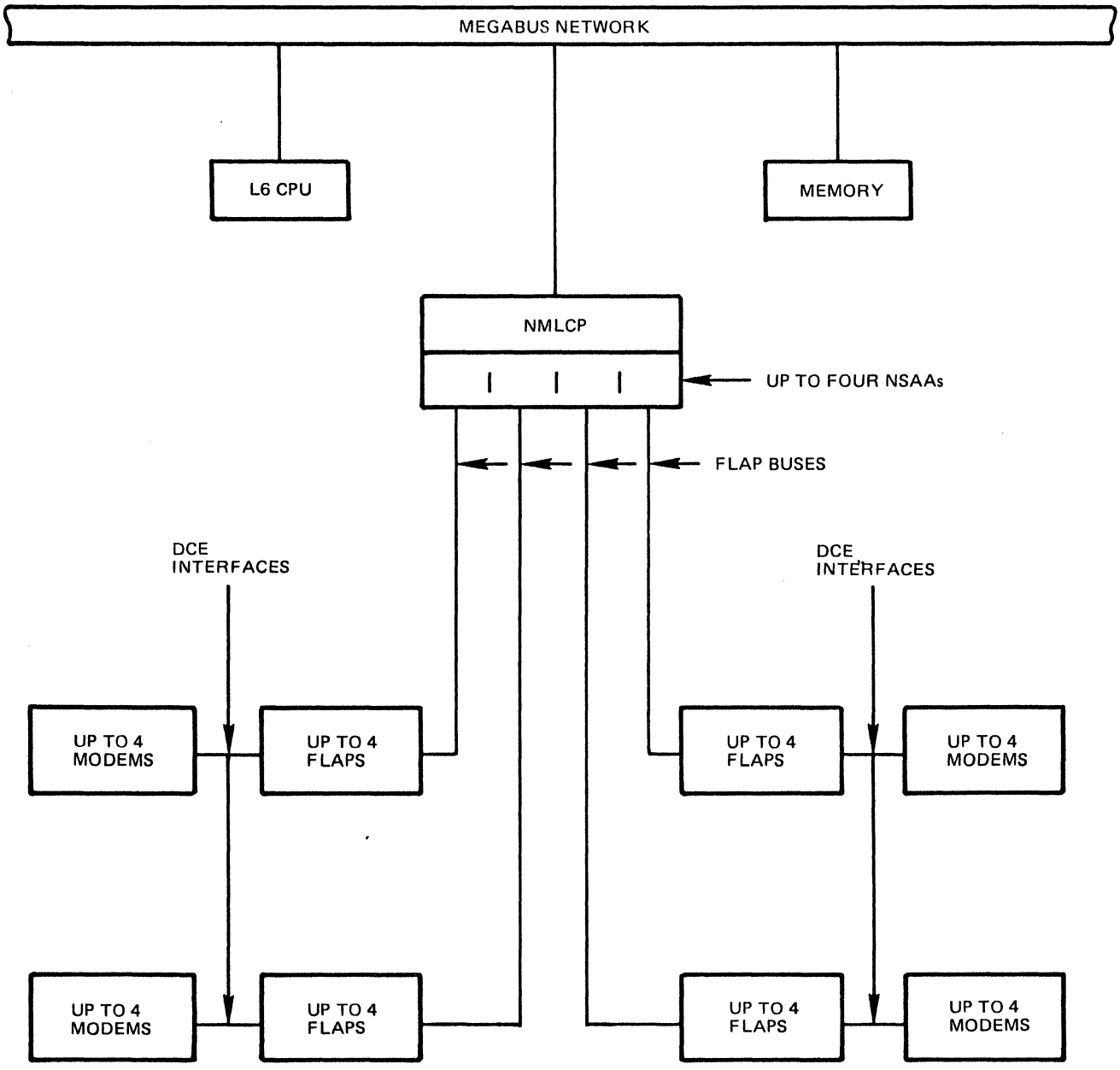


Figure B-5. NMLCP Communications Subsystem

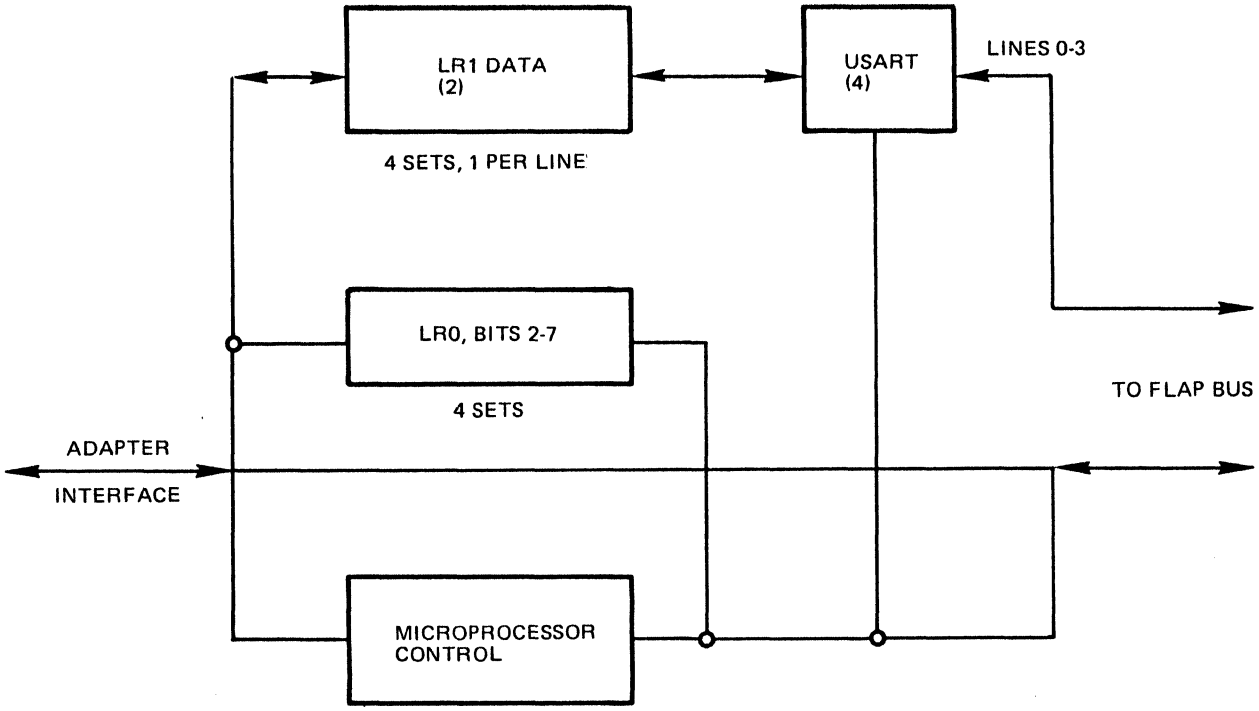


Figure B-6. NSAA Block Diagram

New Functionality

The NSAA provides functionality over and above that required to emulate the ACLA or the SCLA. This new functionality consists of:

1. Parity Generation/Detection
2. Isochronous mode (Via Adapter Switch Control)
3. Double Sync Operation
4. Enhanced Break Detection (in conjunction with FLAPs)
5. Eight-bit word plus parity support
6. Interfaces to FLAPs.

The usage of this new functionality is optional; it can be incorporated with new CCPs. In no way does it impair the operation of existing CCPs written to be run on the MLCP/ACLA or MLCP/SCLA.

The NSAA functionality, including new functionality as well as the ACLA/SCLA subset, is described in the following subsections.

LINE REGISTERS

Software visibility to the NSAA is via a set of Line Registers (LRs).

The NSAA and Processor exchange program visible data, configuration, and control information and status by means of these LRs. The registers are accessed by the Processor through the NMLCP/NSAA interface (see appropriate subsection).

Register information is passed between the Processor and NSAA by the Processor specifying to the NSAA the address and the direction of transfer. The transfer is 8 bits in parallel.

The following subsections detail the bit assignments of the LRs. Reference is made to various CCP instructions (IN, OUT, FIN, FOUT, SEND, RECEIVE, etc.). These instructions are decoded by the Processor and converted by the Processor to commands to the NSAA on the NMLCP/NSAA interface, to which the NSAA responds by either sending an LR or presenting the contents of an LR to the Processor as appropriate.

It should also be noted that if a register identified as READ is written or a WRITE channel is read, if SEND/RECEIVE instructions are misused, or if the Transmit channel accesses a Receive/Transmit Channel register, the results are unpredictable.

Line Register Bit Assignments (ACLA/SCLA Mode)

The LR bit assignments are shown in Figures B-7 and B-8. These assignments are identical to those used in the ACLA and SCLA.

The format of the LRs varies somewhat, depending upon whether the line has been set by switches to operate with an asynchronous or synchronous data format. The NSAA firmware interprets and maps LR contents to USART and FLAP registers according to the line's configuration.

LINE REGISTER 0 (LR0)

LR0 contains the two low-order hex digits of the NSAA extended ID number as given in the subsection entitled, "Device Identification Number."

LINE REGISTER 1 (LR1)

The NSAA and the Processor exchange communication line data in parallel data elements on a channel basis through register LR1. The NSAA's USARTs convert parallel data to bit serial data for transmission and convert bit serial data to parallel data on receive. Each channel contains a serial input or serial output buffer.

A Channel Request Interrupt (CRI) function (see later subsection) is provided for each channel to coordinate data transfer needs with the Processor.

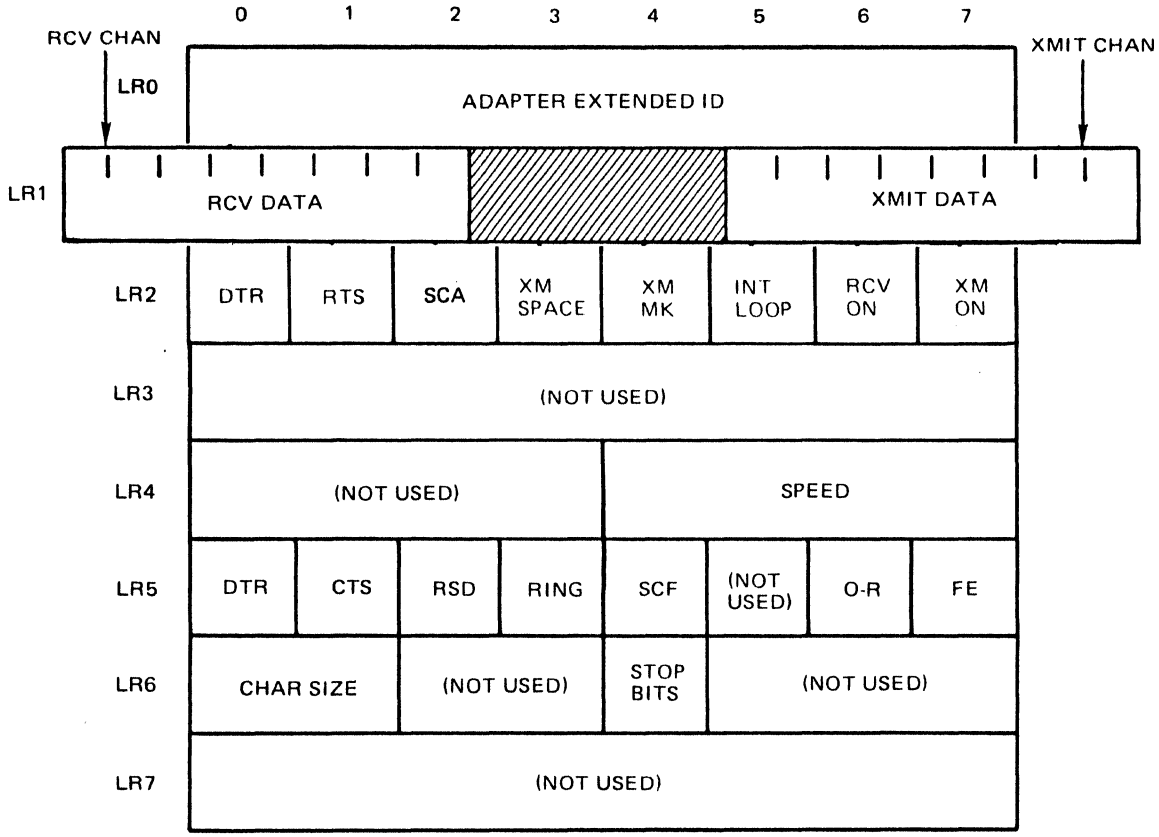


Figure B-7. NSAA Line Registers - Asynchronous Bit Assignments Compatible Mode

A separate LR1 is provided for each channel. The receive channel, via RECV or IN1 instructions, may obtain data characters from this channel. The CCP servicing the receive channel should not use SEND or OUT1 instructions.

The transmit channel, via SEND or OUT1 instructions, delivers data characters to the register. The CCP servicing the transmit channel should not use RCV or IN1 instructions. Bit 7 (least significant bit) is the first bit transmitted or received. If the character size, including parity, is less than 8 bits, the most significant bits will be Zeros.

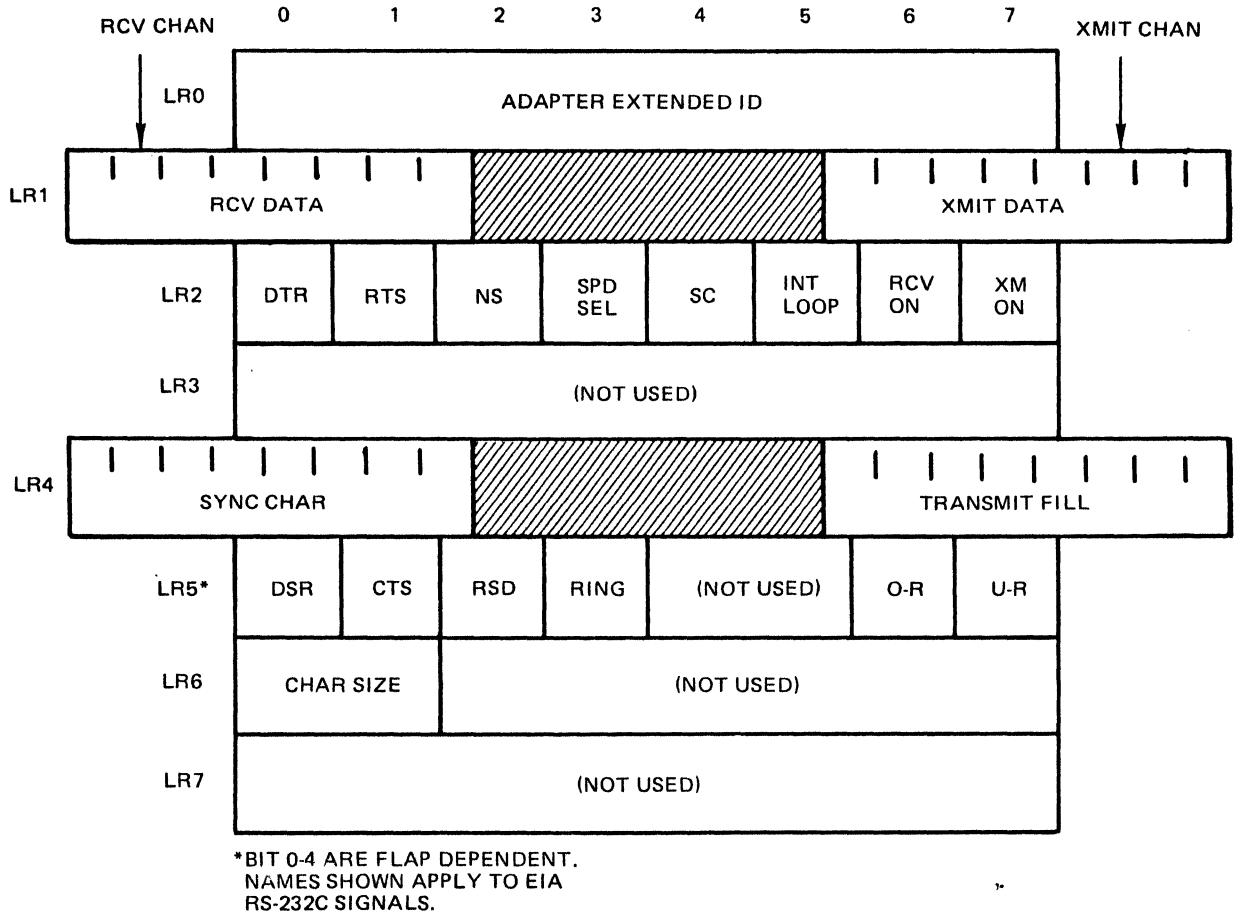


Figure B-8. NSAA Line Registers - Synchronous Bit Assignments Compatible Mode

LINE REGISTER 2 (LR2)

LR2 is used for control purposes by both channels and may be written. An OUT2 instruction delivers bits 0 through 7 to LR2.

In Asynchronous mode, an OUT2 instruction delivers bits 0 through 7 to LR2 and also delivers bits 0 and 1 of LR2 into bit positions 0 and 1 of FLAP register FR2, and bit 2 of LR2 into bit 4 of FLAP register FR2.

In Synchronous mode, an OUT2 instruction delivers bits 0 through 7 to LR2 and also delivers bits 0 through 3 into bit positions 0 through 3 of FLAP register FR2, and bit 4 of LR2 into bit 1 of FLAP register FR4.

Bit names are as follows:

```
Bit 0, DTR       - Data Terminal Ready
Bit 1, RTS       - Request to Send
Bit 2, SCA       - Secondary Request to Send (Asynchronous)
Bit 3, XM SPACE  - Transmit Space (Break) (Asynchronous)
Bit 4, XM MK     - Transmit Mark (Mark Line) (Asynchronous)
Bit 5, INT LOOP  - Internal Loop (Test)
Bit 6, RCV ON    - Receive ON
Bit 7, XM ON     - Transmit ON
Bit 2, NS        - New Sync (Synchronous)
Bit 3, SPD SEL   - Speed Select (Synchronous)
Bit 4, CS        - Clock Source (Synchronous)
```

Bits 0, 1 and 2 (Asynchronous) and 3 are DTE to DCE control signals and are defined in EIA RS-232C.

Other bit meanings are:

Bit 3 - Asynchronous - Transmit Space:
0 - Transmit Data supplied by CCP
1 - Hold Data Output in SPACE (logical 0) condition

Bit 4 - Asynchronous - Transmit Mark:
0 - Transmit Data supplied by CCP
1 - Hold Data Output in MARK (logical 1) condition

NOTE

Bits 3 and 4 are mutually exclusive and therefore must not both be set to 1 in Asynchronous mode.

Bit 4 - Synchronous - Direct Connect: This bit controls the adapter routing of the Processor Direct Connect Clock to the FLAP (see Appendix E).

Bit 5 - Internal Loop (Test): Each line in the NSAA has a software-controlled Test mode which internally loops back data from the transmit channel to the receive channel. The NSAA transmit channel will be disconnected from the FLAP bus interface lines during this mode and these lines held in a mark state.

Setting the INT LOOP bit on the transmit or receive channel will invoke the Test mode. During Test mode, the data rate will be determined by the Processor.

Bit 6 - Receiver ON:
0 - Set the Receiver OFF for this line
1 - Set the Receiver ON for this line

Bit 7 - Transmitter ON:
0 - Set the Transmitter OFF for this line
1 - Set the Transmitter ON for this line

The mapping of LR bits to FR bits is illustrated in Figure B-9.

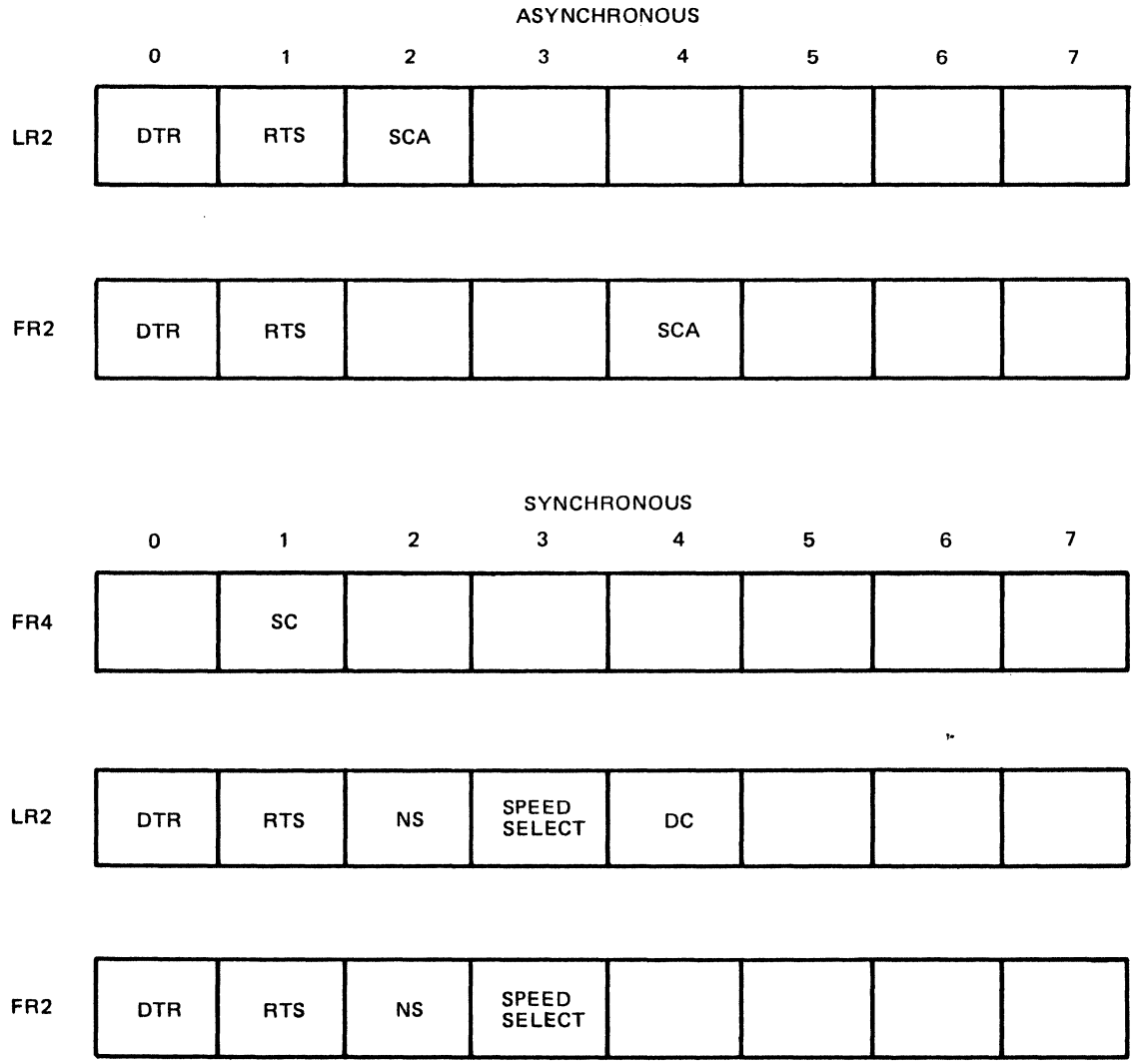


Figure B-9. LR2-To-FLAP Registers Bit Mapping

LINE REGISTER 3 (LR3) - NOT USED

LINE REGISTER 4 (LR4)

In Asynchronous mode, LR4 contains a code in bits 4 through 7 to define the line speed to be used. Speeds are:

Bits 4-7	Rate (bps)
0000	50
0001	75
0010	100
0011	134.5
0100	150
0101	200
0110	300
0111	600
1000	1050
1001	1200
1010	1800
1011	2000
1100	2400
1101	4800
1110	9600
1111	19200

In Synchronous mode, LR4 on the receive side contains a sync character to be used by the receiver for establishing character synchronization. On the transmit side, it contains the fill character used by the transmitter when there is no data to send. LR4 may be written or read. These characters must be the same.

LINE REGISTER 5 (LR5)

LR5 is used for status purposes by both channels.

Whenever LR5 is read via the IN5 instruction, the NSAA copies bits 0 through 4 of FLAP register FR5 into bit positions 0 through 4 of LR5 and then delivers the entire contents of LR5 to the Processor.

Bit names are as follows:

Bit 0, DSR - Data Set Ready
Bit 1, CTS - Clear to Send
Bit 2, RSD - Receive Signal Detect
Bit 3, RING - Ring Indicator
Bit 4, SCF - Secondary Carrier Detect (Asynchronous)
Bit 5, -- - Not Used
Bit 6, O-R - Overrun (Receive)
Bit 7, U-R - Underrun (Xmit) (Synchronous)
Bit 7, FE - Framing Error (Receive) (Asynchronous)

Note these bits are FLAP dependent (see appropriate appendix).

Bits 0 through 4 are DCE to DTE Control Signals, and are defined in Appendix E. Other bit meanings are:

Bit 5 - Parity Error (Asynchronous):
0 - No error detected.
1 - Parity error detected (8 bits plus the parity bit was enabled by the PE bit is LR5).

Bit 6 - Overrun:
0 - No Receive overrun has occurred.
1 - A Receive overrun has occurred. The NSAA was not answered fast enough by the receive CCP and one or more data characters were overwritten (and thus lost) in the receive channel's LR1.

Bit 7 - Asynchronous - Framing Error:
0 - No framing error.
1 - A framing error has occurred. The NSAA has detected a missing stop bit.

Bit 7 - Synchronous - Transmit Underrun:
0 - No transmit underrun.
1 - Transmit underrun has occurred. The NSAA was not serviced fast enough by the transmit CCP and the transmit fill character (transmit channel LR4 contents) was transmitted.

LINE REGISTER 6 (LR6)

LR6 can be written or read, and is used to provide configuration data to the NSAA. Bit meanings are as follows:

Bits 0 and 1, Character Size:
00 - 5 bits
01 - 6 bits
10 - 7 bits
11 - 8 bits

Bit 4, Asynchronous - Stop bits:
0 - 1 stop bit per 5-, 6-, -7 or 8-bit data character.
1 - 1.5 stop bits per 5-bit data character.
1 - 2 stop bits per 6-, 7- or 8-bit data character.

LINE REGISTER 7 (LR7) - Not Used.

Line Register Bit Assignments (Non-ACLA/SCLA)

Non-ACLA/SCLA mode functionality is provided by the NMLCP beyond that of the MLCP. The register bits involved are illustrated in Figures B-10 and B-11. In these figures, bits not shown are as in Figures B-7 and B-8, respectively. The adapter control and adapter status registers are accessed via the ACTL and AST instructions, respectively.

Line Register 0, refer to subsections entitled "Device Identification Number" and "Mode Control."

Line Register 5, bits 2 and 3, asynchronous when written via OUT5 instruction:

Bit 3 (PE) - Parity Enable (asynchronous only)
Bit 2 (PT) - Parity Type (asynchronous only)

If bit 3 is set, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming 8-bit data:

Bit 2 = 0 selects odd parity
Bit 2 = 1 selects even parity.

Bit 2 is ignored unless bit 3 = 1.

Eight bits plus parity can therefore be supported in the adapter by software setting LR6 bits 0 and 1 to 11 (8-bit) and setting LR5 bits 2 and 3 as appropriate. For other character sizes, parity is supported in the NMLCP analogous to the MLCP.

Line Register 5, bit 5, asynchronous when read is parity error.

Double Synchronous and Transparent mode (Synchronous)

Line Registers 4, 3 and 7 contain the SYNC1, SYNC2 and DLE characters. Their utilization is controlled by LR5 bit 0, DSM (Double Synchronous mode), and LR5 bit 1, TRM (Transparent mode), when LR5 is written via the OUT5 instruction in Synchronous mode.

TRANSMITTER OPERATION

When DSM = 0 and TRM = 0, SYNC1 alone is used to establish synchronization and for character fill. When DSM = 1 and TRM = 0, SYN1-SYN2 is used.

In Transparent mode (LR6 bit 5 = 1), DLE-SYN1 is used for character fill and the synchronization sequence used to establish character synchronization is controlled by DSM; i.e., SYN1 when DSM = 0, or SYN1-SYN2 when DSM = 1.

RECEIVER OPERATION

In Single Sync, Nontransparent mode (DSM, TRM = 00), the first character in the data stream matching SYN1 is not transferred to Receive LR1.

Note that SYN characters used to establish initial synchronization are not transferred to Receive LR1 in any case.

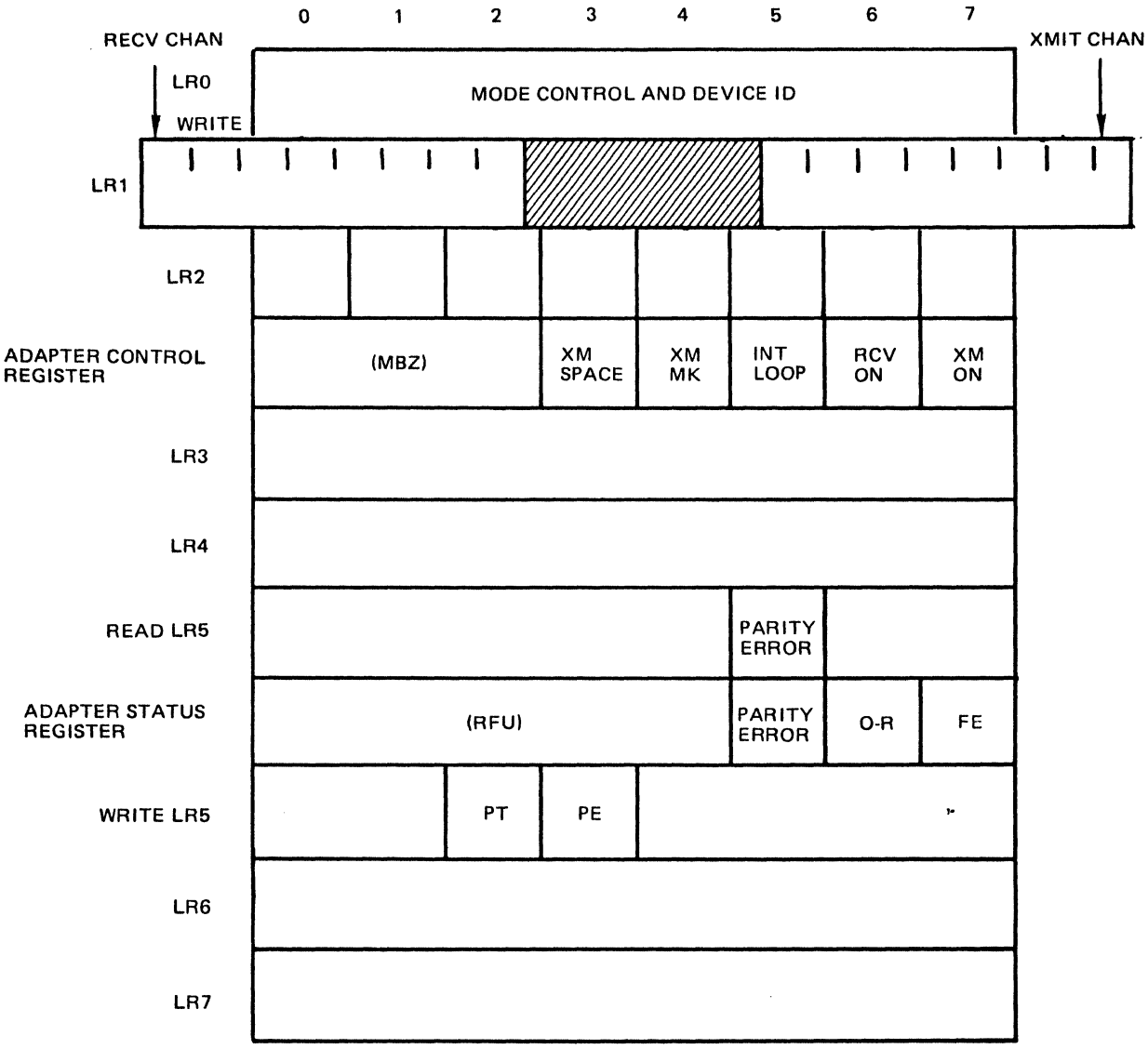


Figure B-10. NSAA Line Registers - Asynchronous Bit Assignments - New Functionality

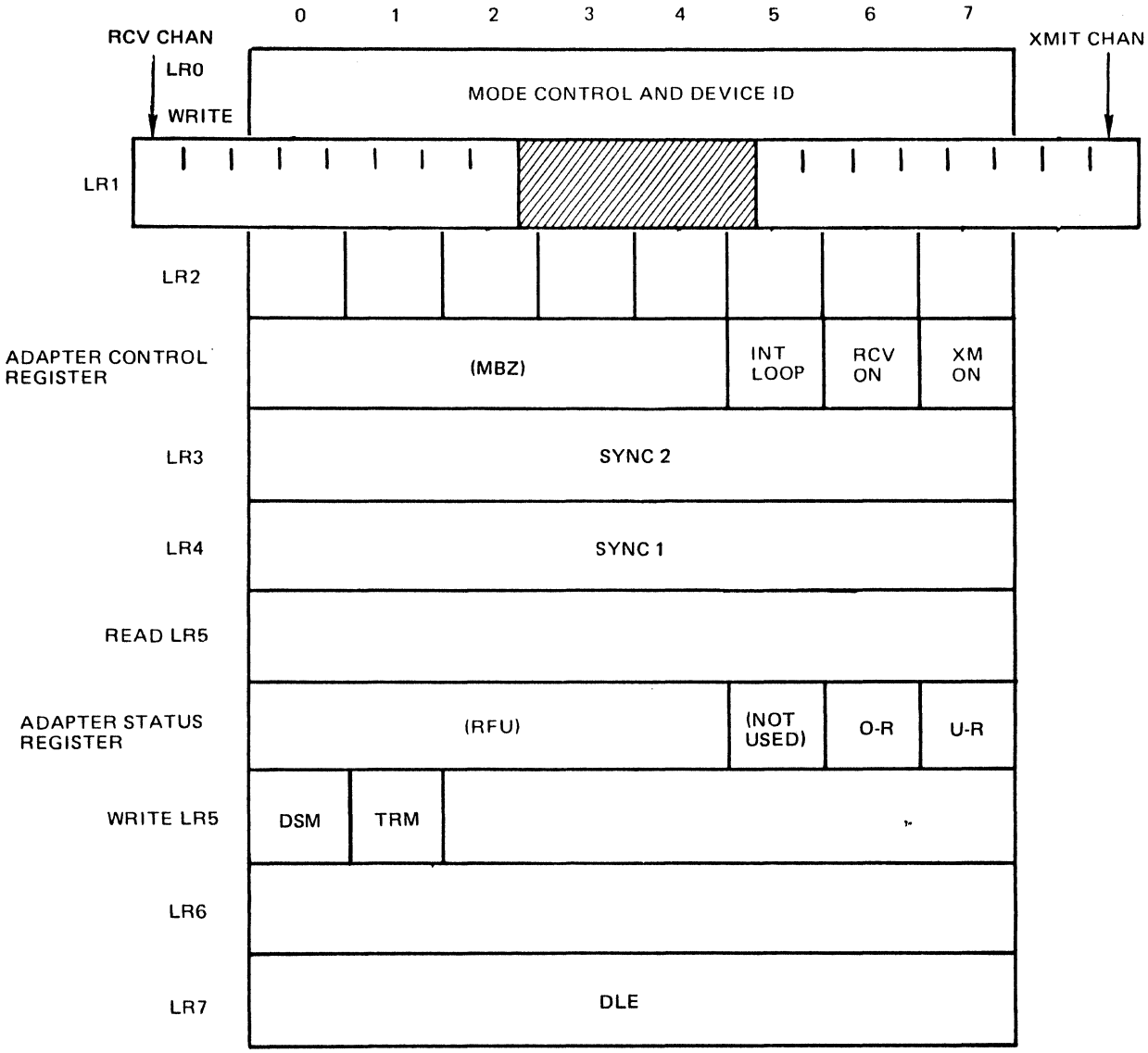


Figure B-11. NSAA Line Registers - Synchronous Bit Assignments - New Functionality

Composite Line Register Bit Assignments

The complete line register bit assignments are shown in Figures 3-12 and B-13.

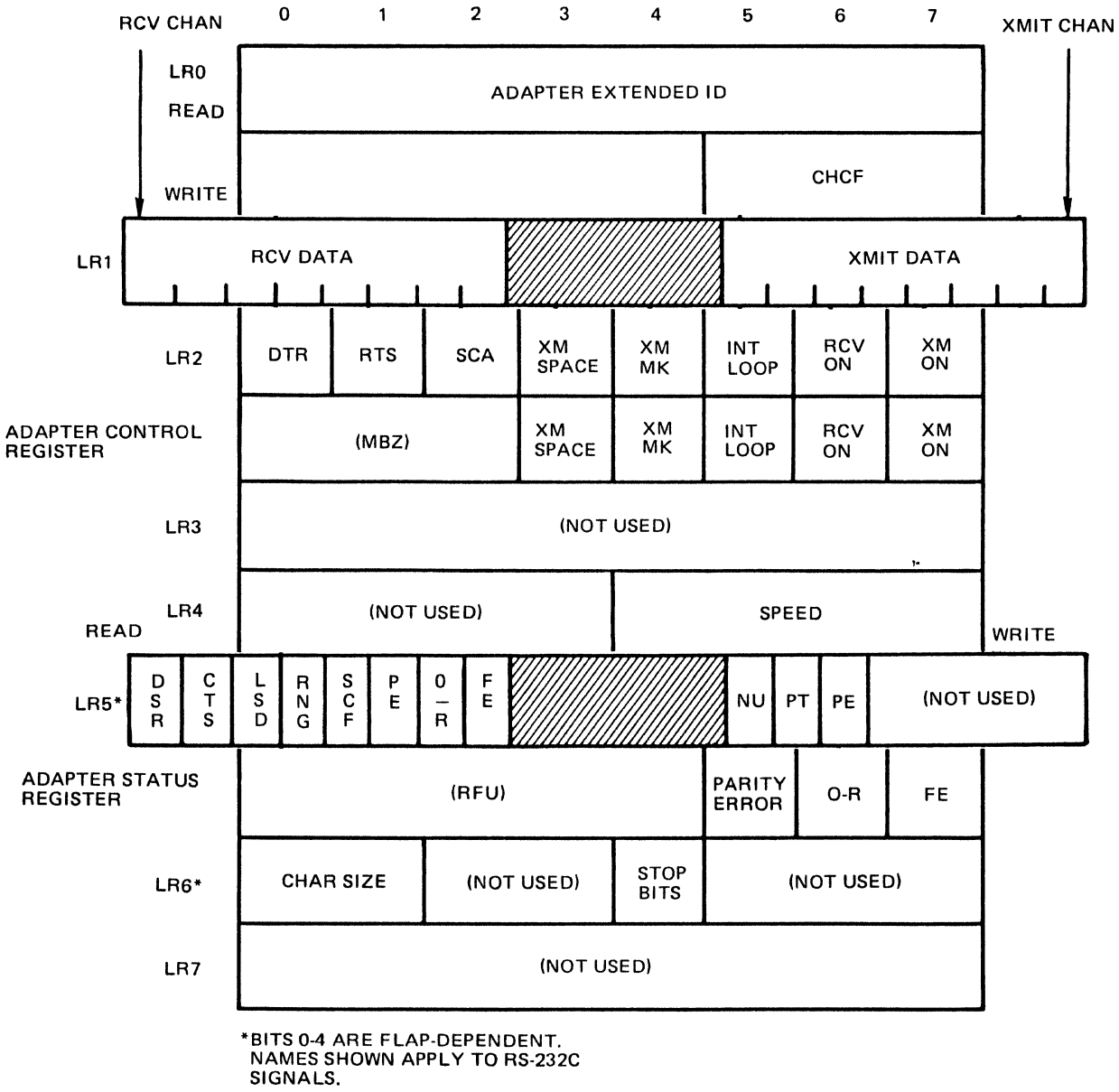


Figure B-12. NSAA Line Registers - Composite Asynchronous Bit Assignments

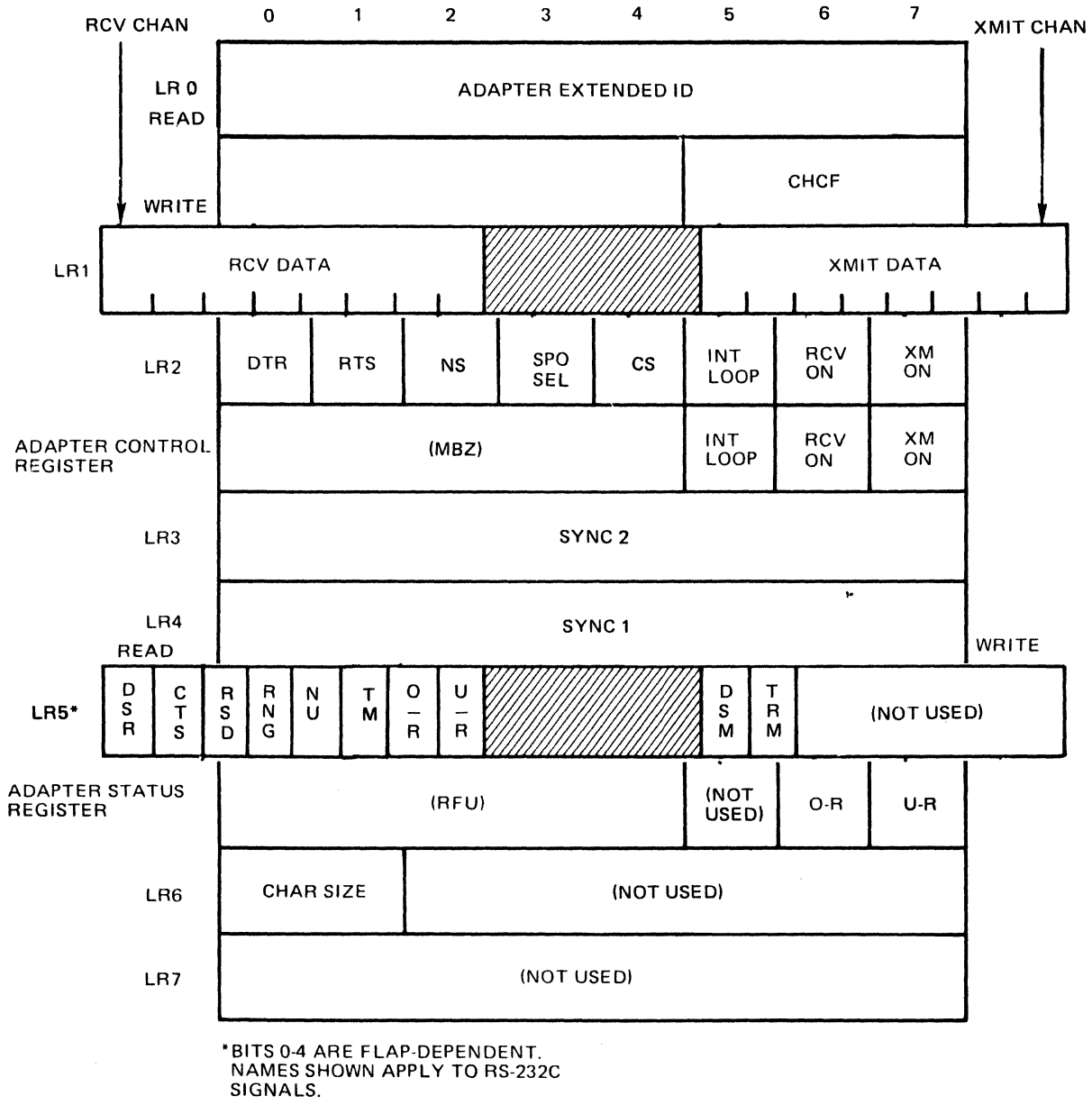


Figure B-13. NSAA Line Registers - Composite Synchronous Bit Assignments

ADDITIONAL FUNCTIONALITY

Other new functionality is available with the NSAA and with the NSAA plus the associated FLAPs. This functionality includes:

1. Isochronous Mode Support

   A switch position will be associated with each line of the NSAA. One position of this switch will be inactive and the NSAA line will assume either Asynchronous or Synchronous modes as determined by the setting of the Asynchronous/Synchronous switch discussed previously in this appendix. In the other position of the Isochronous switch, the NSAA will accept from the associated DCE and sync its data transitions to this clock.

2. Enhanced Break Detection

   Framing errors, as reported by LR5 bit 7 in Asynchronous operation is retained from previous designs. This status indication can be used by CCPs in conjunction with associated FLAP FR bit 6. This FLAP register bit tracks Receive Data, and can therefore be used to indicate the cessation of a BREAK.

3. FLAP Interface

   FLAPs provide additional control and status features including improved testing capability via local and remote loop-back. Refer to Appendix E for further details.

FLAP REGISTERS

Each NSAA line has a set of eight addressable FLAP registers associated with it. These registers provide FLAP ID, DCE control, and DCE status. Those registers are transparent to existing software. They can, however, be addressed directly via Processor FIN and FOUT CCP instructions.

CHANNEL REQUEST INTERRUPT (CRI)

Each channel is capable of generating its own CRI to signal a need for data service.

On receive, when a previously empty receive holding register is loaded with a data character by the USART, a CRI is generated by that channel.

If the RCV ON bit is turned off, channel CRIs are disabled and any characters in the USART are discarded.

On transmit, a CRI is generated by the channel when the last character loaded has been serialized. If the XM ON bit is turned off, all characters in the USART are transmitted completely and channel CRIs are disabled.

CRI generation conditions mentioned above are those in which the channel would have previously entered a WAIT condition.

If the RCV ON bit or the XM ON bit is on, the USART channel is enabled and the CRI function is enabled. Turning these bits will eventually result in a CRI since the channel would previously have been in a WAIT condition.

Each NSAA attached to the parent Processor via the adapter Bnterface has an Interrupt High Priority and an Interrupt Low Priority line on the bus. Receive channels are high priority and transmit channels are low priority and, within each set, the lower the channel number, the higher the priority. The Processor responds to the interrupt by requesting from the adapter the channel requiring service. The NSAA then provides the channel number on the bus. Thus, the Processor decides which adapter is to be serviced, and the NSAA determines which channel.

CONFIGURATION

Upon initialization, firmware will load the USART registers with those default values which are not provided by current CCPs via the LRs. Firmware does this by first determining whether a line is to operate asynchronously or synchronously via a read of the Asynchronous/Synchronous switch. Firmware then sets up the following USART default values:

Asynchronous Default Loads:

1. Asynchronous mode
2. Parity Generation/Detection disabled

Synchronous Default Loads:

1. Synchronous mode
2. Parity Generation/Detection disabled
3. Single Sync Operation
4. Nontransparent mode

These initial values, plus those subsequently mapped from the LRs, enable existing software designed for the MLCP and its synchronous and asynchronous adapters to operate on the NMLCP/NSAA without change.

New CCPs can override these initial values by writes to LR0, LR3, LR4, LR5, LR6 and LR7, and in so doing capitalize on the new functionality discussed previously in this appendix. The CCPs should ensure that the transmitter and receiver are OFF by a write to LR2 with bits 6 and 7 equal to Zero. Line Registers 0, 3, 4, 5, 6 and 7 should thus be configured and finally the transmitter and receiver should be enabled.

CHANNEL NUMBER ASSIGNMENT

The channel numbering for a fully configured Processor with four NSAAs is given in Table B-1. Channels 0 through 7 apply to one NSAA, channels 8 through 15 apply to the second NSAA, etc.

Table B-1. Channel Numbering

Channel	Line	Direction
0	0	R
1	0	T
2	1	R
3	1	T
4	2	R
5	2	T
6	3	R
7	3	T
.	.	.
.	.	.
.	.	.
30	15	R
31	15	T

DEVICE IDENTIFICATION NUMBER

The basic identification number is furnished by the Processor without reference to the presence or type of NSAA attached to it (refer to Section 2).

The extended device identification is provided by the NSAA and its associated FLAP. This number is $(N1\ N2\ N3\ N4)_{16}$ where N1 N2 is provided by LR0 of the adapter and is 78-7A in the case of the NSAA, and N3 N4 is provided by FR0 of the attached FLAP. N3 N4 = $(00)_{16}$ indicates that no FLAP Is present.

N1, N2 values are as follows:

78 = Synchronous operation
79 = Isochronous operation
7A = Asynchronous operation

In performing the Input Extended ID order, the Processor reads LR0 and FR0, and assembles their contents into the L6 bit numbers given above. Since the reading of FR0 fetches bits 0-7, and bits 0 and 1 of FR0 are not meaningful in this context, the Processor masks FR0 bits 0 and 1 to 00_2 prior to the assembly and presentation of the extended ID to the Megabus network.

DATA CLOCK

For NSAA lines configured for Asynchronous mode, a baud rate generator in the NSAA frequency divides a clock source provided by the Processor to produce a line speed bit rate clock at the rate specified by the SPEED field in LR4. The USART uses this line speed clock to time data serialization. This mode is used only with asynchronous modems and asynchronous DTE.

For NSAA lines configured for Synchronous mode (except for direct connect), the data clock source is always in the DCE (modem). Both a receive and a transmit clock are provided and are fed through the DCE interface to the FLAP, through the FLAP bus to the NSAA, and finally to the USART.

For NSAA lines configured for Asynchronous mode and which are set by switch to operate isochronously, the data clock source is always in the DCE (modem). Both a receive and a transmit clock are provided and are fed through the DCE interface to the FLAP, through the FLAP bus to the NSAA, and finally to the USART.

RECEIVE SYNCHRONIZATION

Asynchronous Channel

Synchronization is established for each character received by the presence of framing (start and stop) bits associated with each character. On a Break, as evidenced by the NSAA detecting a message stop bit, the framing error bit is set and remains set until reset by software which can be accomplished by turning the Receiver off then on.

Synchronous Channel

The receive channel has the capability to scan the incoming serial bit stream for a particular pattern (search for synchronization) to achieve byte level synchronization.

In ACLA/SCLA Compatibility mode, the receive channel compares the contents of LR4 (the sync character) with the most recently received bits, at each bit time, until the patterns match. The number of bits compared is in accordance with the character size specified in LR6. No data transfer is initiated on the channel until the patterns match. Synchronization is established upon detection of the synchronization character which is not transferred to the Processor. Succeeding characters of the

specified character size are thereafter transferred to the Processor. If double sync is being used, synchronization is achieved when the NSAA detects SYN1 immediately followed by SYN2. Note that the sequence SYN1-SYN1-SYN2 will not achieve synchronization. Note also that neither SYNC character is transferred to the Processor.

RECEIVE OVERRUN

If a receive channel overflows because the Processor did not service the channel CRI soon enough, receive data is lost and the receive overrun bit (LR5, bit 6) is set. This bit will be sensed by the next RCV instruction. Clearing of the error and resync of the receiver is the responsibility of the CCP which can be performed by turning the receiver off and then on.

TRANSMIT UNDERRUN/TRANSMIT FILL

If a transmit channel character is lost because the Processor did not service the channel CRI soon enough, a transmit underrun condition exists.

Asynchronous Channel

On an asynchronous channel, the transmitter will simply leave the line in a continuous marking (stop bit) condition when there are no characters to send and no underrun indication is reported.

Synchronous Channel

On a Synchronous mode channel, a transmit underrun condition occurs when the USART becomes empty, causing the transmit underrun bit (LR6, bit 7) to be set.

In ACLA/SCLA Compatibility mode, when a transmit underrun condition occurs, the transmit fill character (the contents of LR4) is taken as the next character.

In Double Sync mode, SYN1-SYN1 is used for transmit fill.

In Transparent mode, SYN1-DLE is used for transmit fill.

INITIALIZATION

Processor Hard Initialize clears all USARTs and all line registers. Each line must be configured before it can be operated.

The channel Initialize clears interrupts and errors and turns off the receiver and transmitter.

AUTO CALL ATTACHMENT

Up to two lines on the NSAA can be configured with auto-call capability, using the Auto-Call FLAP in conjunction with the applicable communication interface FLAP, e.g., EIA RS-232C. In other words, each auto-call FLAP must be paired with another FLAP.

The Auto-Call FLAP is accessed directly via FIN and FOUT instructions, rather than indirectly via IN and OUT instructions which are mapped to FLAP registers. As such, existing Auto-Call CCPs are not compatible.

MODE CONTROL

The NASS supports three modes of operation, Asynchronous, Isochronous, and Synchronous.

A DIP switch is provided on the adapter to specify which mode the NSAA shall enter, and set up initial conditions for when the adapter is initialized. The adapter is initialized by a Channel Initialize, an Processor Soft Initialize or an Processor Hard Initialize as described in Appendix A

A means is provided whereby the mode of the NSAA can be switched under software control.

The NSAA mode can be switched by executing an OUT instruction to LR0 with a value corresponding to that in the subsection in this appendix entitled, "Device Identification Number," that is:

A value of $(78)_{16}$ causes NSAA to enter SYNC mode
A value of $(79)_{16}$ causes NSAA to enter ISOC mode
A value of $(7A)_{16}$ causes NSAA to enter ASYNC mode.

It should be noted that switching of the mode by software will affect the contents of LR0 and the adapter's ID. Mode switching automatically turns off both Transmitter and Receiver but does not otherwise alter the adapter LRs or the FLAP FRs. It is the responsibility of the CCP to reload these in accordance with the new mode.

Appendix C
AUTO CALL UNIT FLAP

The Auto Call Unit FLAP (ACUF) makes possible the use of the automatic calling facility on DPS 6/Level 6 systems, eliminating the need for manual dial-up procedures in communications networks where switched telecommunication lines are involved. The ACUF is attached to the DPS 6/Level 6 New Multiline Communications Processor (NMLCP) and is also physically connected to the automatic calling device.

CONFIGURATION INFORMATION

ACUF Configuration

Figure C-1 illustrates the attachment of an NMLCP to a remote terminal via a switched autocall line. The data connection is made by any of the supported modems (e.g., Bell Type 103, 201, 202, 203, 208, 209 or equivalent) which necessitates an appropriate NMLCP adapter. The attachment to the automatic calling device (e.g., Bell System 801A, 801C or equivalent) is made via the ACUF. The connection between the data modem and the automatic calling equipment is made by the supplier of the Data Communication Equipment.

The ACUF attaches to the NMLCP adapter via the FLAP bus. Each adapter is capable of supporting four automatic calling devices.

NOTE

Certain information presented in this appendix is reprinted with permission of the Electronic Industries Association (EIA) specification, Interface Between Data Terminal Equipment and Automatic Calling Equipment for Data Communication, RS-366, August 1969.

Automatic calling devices may be any devices meeting the EIA RS-366 specification, for example, Bell System 801A and 801C units. The 801A unit or equivalent is used where the existing data set mode of communication uses rotary dialing. The 801C or equivalent is used where the data set mode of communication is parallel binary signals. Two different types of automatic calling equipment transfer to Data Set mode are supported by the Communications-Pac; the use of the End-of-Number code and the receipt of answer code.

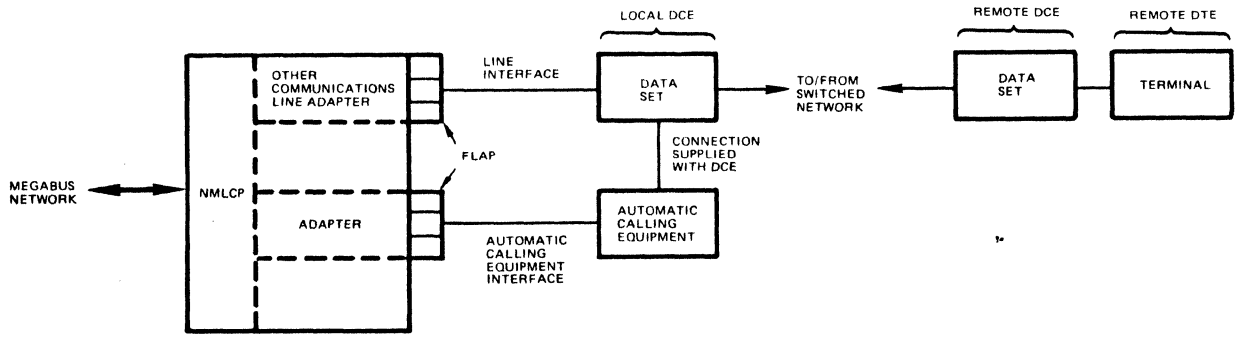


Figure C-1. ACUF Environment

Device Identification Number

The device identification number of the ACUF is 0106 (when attached to an HDLC, or 7X06 if attached to a SYNC/ASYNC adapter). The device identification number is returned in response to an Input Extended Device Identification (function code 08) issued to either of the Processor channels containing the ACUF.

Automatic Calling Device Configuration Options

When the automatic calling equipment is ordered from the communciations carrier, the following device type is required:

1. Bell System 801A Automatic Calling Unit with rotary dialing or equivalent unit, or

2. Bell System 801C Automatic Calling Unit with the American Telephone and Telegraph Company's TOUCH-TONE dialing or equivalent unit (preferred because of faster dialing capability).

Either type 801A or 801C may be configured with the following:

Call Termination:

1. By the ACUF turns off CRQ.

2. By associated communications line adapter sets the data set ready indicator (i.e., dropping DTR) in the appropriate line register.

NOTE

Refer to the appropriate appendix of this manual for the Communications-Pac in question.

Transfer to Data Mode:

1. Automatic calling device detects an answer signal from the called station and returns that line to the associated data set (Recommended).

2. Automatic calling device returns that line to the associated data set upon detection of End-of-Number code, used where the associated data set detects the answer signal (e.g., Bell System, 100 Series of data sets and some European models).

Abandon Call and Retry (ACR) Timer Turn-Off:

1. When the associated data set goes into Data Set mode (used with automatic calling device detection of answer signal).

2. Do not turn off but rather set ACR when preset time elapses (used with data set detection of answer signal).

Interval of Abandon Call and Retry Timer: User-selected Options are 7, 10, 15, 25, or 40 seconds. (25 seconds is recommended interval for domestic (U.S.) application.)

FLAP REGISTERS

The CCP program to the Processor from the ACUF is via a set of registers. The CCP can access these registers via the FIN/FOUT instructions. By appropriate CCP programming, a dialog can be established with an automatic calling device which calls the designated number and then transfers control to an associated data set transmission/reception.

The ACUF has six visible registers. A one bit in any register corresponds to a one bit for a digit, or for a control or status signal at the automatic calling device interface. These registers are defined below.

FLAP Register 1 -- Output Data

This register (Figure C-2) is accessible on the odd channel via a FOUT instruction and may not be read by the CCP.

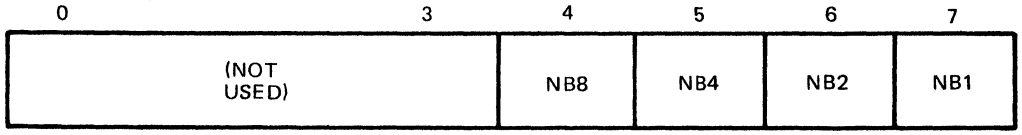


Figure C-2. FLAP Register 1 Output Data

Bit definitions are as follows:

Bits 0-3 - Not Used
Bit 4 NB8 - Digit Signal Circuit, high order bit
Bit 5 NB4 - Digit Signal Circuit, third order bit
Bit 6 NB2 - Digit Signal Circuit, second order bit
Bit 7 NB1 - Digit Signal Circuit, low order bit.

The information presented on these interchange circuits may be either transmitted (e.g., digits of the called number) or used locally as a control signal. An important use of these interchange circuits for control purposes is the passing of the End-of-Number code combination to the automatic calling equipment after the last digit of the number to be called has been passed. In response to End-of-Number, the automatic calling equipment immediately transfers the communication channel to the data set without waiting for an answer signal from the called data set. Table C-1 defines the digit signal character set.

Table C-1. Digital Signal Character Set

Digital Signal Circuit States				
Digit	NB8	NB4	NB2	NB
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
*	1	0	1	0
#	1	0	1	1
EON	1	1	0	0
Unassigned	1	1	0	1 (See Note)
Unassigned	1	1	1	0
Unassigned	1	1	1	1

NOTE

Used as the Separation control character (SEP) in CCITT Recommendation V-24.

FLAP Register 2 -- Output Control

This register (Figure C-3) may be accessed on both channels of the ACUF via the FOUT instruction and may only be written.

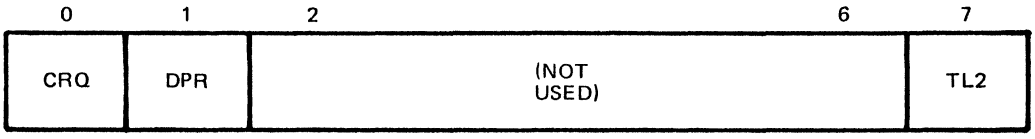


Figure C-3. FLAP Register 2 Output Control

Bit definitions are as follows:

Bit 0. CRQ - Call Request to automatic calling device.

Signals on this circuit are generated by the data terminal equipment to request the automatic calling equipment to originate a call.

The on condition indicates a request to originate a call and must be maintained during call origination, until Circuit Call Origination Status (bit 0 of FR5) is turned on, in order to hold the connection to the communication channel (remains off hook). The call is aborted if Call Request to ACU (bit 1 of FR2) is turned off prior to turning on Call Origination Status.

The off condition indicates that the data terminal equipment is not using or has completed a prior use of the automatic calling equipment.

Call Request must be turned off between calls or call attempts and turned on unless Data Line Occupied (bit 2 of LR5) is in the off condition.

Bit 1. DPR - Digit Present to automatic calling device.

Signals on this ciruit are generated by the data terminal equipment to indicate that the automatic calling equipment may read the code combination presented on the Digit Signal Circuits (NB1, NB2, NB4, and NB8 (see LR1 description). The off to on transition indicates that the data terminal equipment has set the status of the Digit Signal Circuits for the next digit.

Digit Present must not be turned on before a Present Next Digit signal from the ACU has occurred, that is, a channel request interrupt is generated by the ACUF. Digit Present must not be turned off until after the CCP outputs the next digit. Refer to "Example 2, Transfer of Each Individual Digit," in the portion of this appendix entitled "Programming Considerations."

The status of the Digit Signal Circuits must not change when the Digit Present bit is in the on condition.

Bits 2-6 Not Used

Bit 7 TL2 - Test Loop 2 sets the FLAP in register loop-back mode, when TL2 is set.

FLAP Register 5 - Input Status

This register (Figure C-4) may be accessed on both channels of the ACUF via the IN instruction and may only be read.

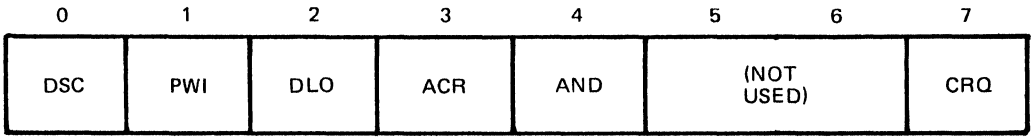


Figure C-4. FLAP Register 5 Input Status

Bit definitions are as follows:

Bit 0. DSC - Distant Station Connected from automatic calling device.

NOTE

In the Bell System 801A and 801C manuals, this is called DSS (Data Set Status). Signals on this circuit are generated by the automatic calling equipment to indicate the status of automatic call origination procedures.

The on condition presented during a call originated by the automatic calling equipment indicates that the automatic calling equipment has completed its call origination functions and that the control of the communication channel has been transferred from Call Request (bit 1 of LR2) to Circuit CD (Data Terminal Ready) in the data set interface (see the following Note). When Call Originating Status is turned on, the data terminal equipment may turn Call Request off without causing a communication channel disconnect. Disconnection of the channel by the data terminal equipment is then possible only through the associated data set interface.

NOTE

This indication is present in the appropriate line register of the associated communications line adapter and must be examined by the communication (line) CCP. Refer also to the EIA RS-232C Specification.

Once Call Originating Status is turned on, it shall remain on at least until Call Request is turned off by the data terminal equipment. Call Originating Status may come on at other times; e.g., during an incoming call or a manually originated call. Any on condition appearing at a time other than during automatic call origination by the automatic calling equipment should be disregarded.

This circuit should not be interpreted to convey information regarding the operational status or state of preparedness of the associated data set (see Note).

NOTE

If call termination is by the associated communications line adapter setting data, set the ready indicator (i.e., dropping DTR) in the appropriate line register. Otherwise, the line is dropped.

Bit 1. PWI - Power Indicator from ACU.

Signals on this circuit are generated by the automatic calling equipment to indicate whether power is available within the automatic calling equipment.

The on condition indicates that power is available in the automatic calling equipment. The off condition indicates a loss of power in the automatic calling equipment. This circuit should not be interpreted to indicate the power status in any other equipment.

Bit 2. DLO - Data Line Occupied from automatic calling device.

Signals on this circuit are used to indicate when the communication channel is in use for automatic calling, data communication, voice communication or for testing of the automatic calling or data communication equipment.

The on condition indicates that the communicator is in use.

The off condition indicates that the data terminal equipment may originate a call provided that Circuit PWI (Power Indication bit 1 of LR5) is on.

The off condition of Data Line Occupied (DLO) shall not be presented until all of the other interchange circuits from the automatic calling equipment are returned to their proper idle condition.

Bit 3. ACR - Abandon Call and Retry from automatic calling device.

Signals on this circuit are used to indicate the probability of successful completion of the call attempt.

The on condition, when presented during the process of call origination, indicates that there is a high probability that the connection to a remote data station cannot be successfully established and is a suggestion to the data terminal equipment to abandon the call and to reinitiate the call at a later time. The automatic calling equipment does not determine that the call is to be abandoned. Action required to abandon the call must be initiated by the data terminal equipment.

When the Answer Signal mode of operation is used, Abandon Call and Retry remains in the off condition after Call Origination Status (bit 0 of LR5) is turned on. When the End-Of-Number (EON) mode is used, ACR continues to function (i.e., the bit continues to be set) after Call Origination Status is turned on.

Bit 5. PND - Present Next Digit from ACU.

Bit 6. CRQ - Call Request Signal externally looped back when wrap is used at the connector.

FLAP Register 7 - Input Status

This register (Figure C-5) may be accessed on both channels of the ACUF via the FIN instruction and may only be read.

These bits contain the contents of FR2 bits 4-7 looped back when TL2 is set.

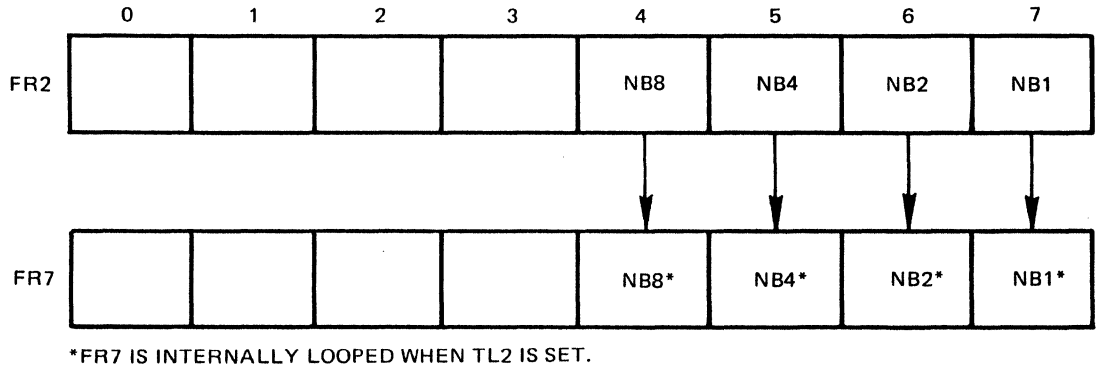
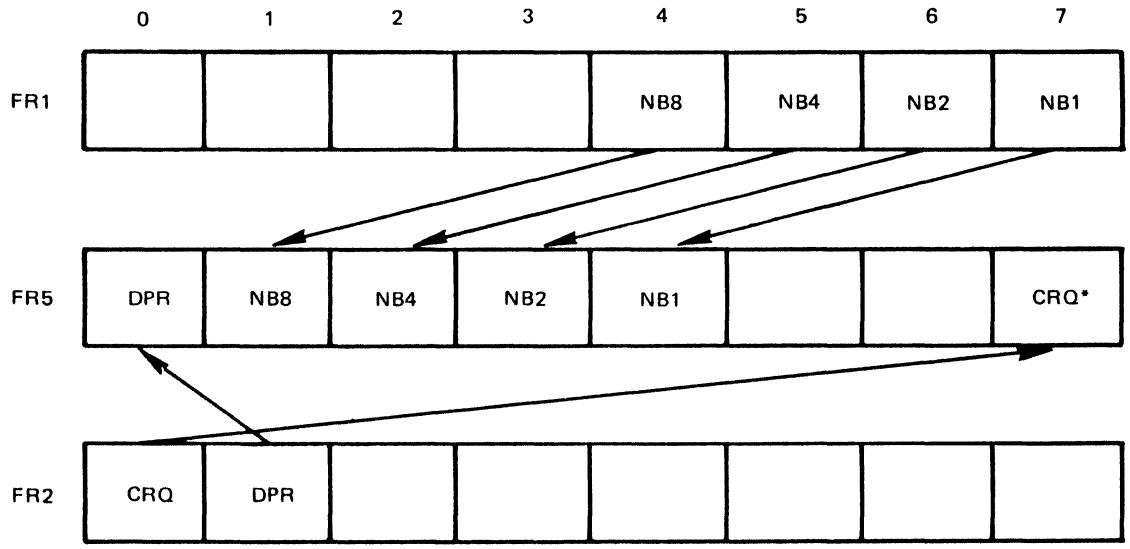


Figure C-5. Line Register 7 Input Status

The following is an example of loop-back conditions when a wrap-around plug is used at the connector.



*EXTERNALLY LOOPED.

PROGRAMMING CONSIDERATIONS

The topics that follow discuss various programming aspects of the ACUF. The reader should be familiar with the FLAP register fields and their functions, discussed previously.

The timing sequence of call placement, how data transfers occur (i.e., how the digits of the call are transferred from the ACUF to the automatic calling device), and call termination of an ACUF CCP (and its relationship to a main memory program) are discussed below. It is important to note here that all communication between the ACUF and a communications line adapter (such as a NSAA) takes place via the main memory program. The ACUF and the communications line adapter never communicate directly. Three examples of a program sequence are included.

Timing Sequence of Call Placement

Figure C-6 is a typical diagram of the sequence which would occur during call placement. The occurrence of various signals all come about either because of specific CCP action or because of a response from the automatic calling device. To that extent, the (automatic calling) CCP has complete control over the situation. Data Set Scan is used to detect transitions of the Present Next Digit signal from the automatic calling device.

Refer to Examples 1 and 3 following. Note that the automatic calling CCP is differentiated from the communciations line adatper CCP.

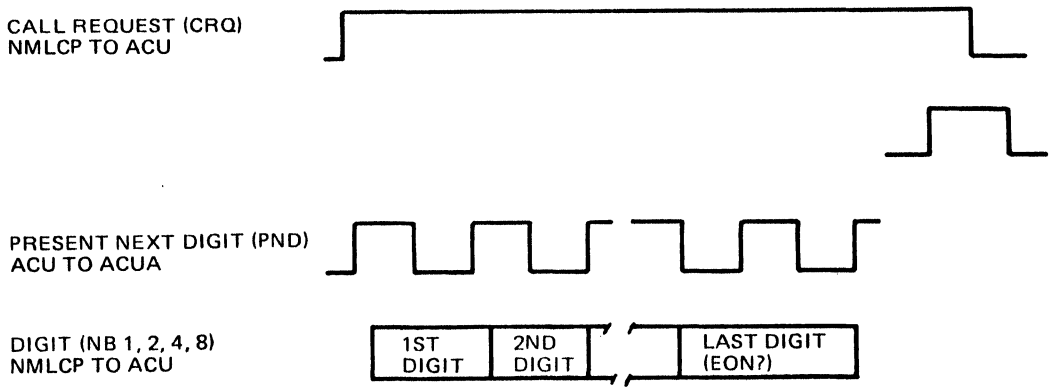


Figure C-6. Typical Automatic Calling Equipment/ACUF/NMLCP Timing Sequence

Data Transfers

The data interface from the ACUF to the automatic calling device is a 4-bit wide path. Each transfer represents a single digit to the called station. In addition to the data, several control lines are supplied to the automatic calling device from the ACUF. Each of the output signals (data and control) are buffered by a flip/flop in the ACUF so that the CCP need only set them in the correct condition and they will remain in that condition until specifically changed by the CCP or until the channel is initialized.

Signals from the automatic calling device to the ACUF are visible to the CCP as specific bits in specific registers, as previously detailed in the register descriptions. It is useful to repeat here that the ACUF contains no storage on these signals, so that the condition of the bit read by the CCP always represents the condition of that specific line at the time the IN instruction (or SEND) is issued.

Call Termination

The automatic calling devices offer two methods for terminating a call. In the first case, after the transfer to the data set, the call is terminated by the automatic calling devices's dropping the Call Request (resulting in bit 0 of FLAP register 2 being turned off automatically by the ACUF hardware) at the end of the call.

In the second case, the call is terminated via the data set's dropping the Data Terminal Ready at the end of the call. The latter option is preferred because it makes possible the elimination of one CCP interrupt. Refer to Examples 1 and 3 following.

Summary and Examples

The following examples illustrate the general sequence of a CCP for automatic calling. Example 1 illustrates a possible sequence for a CCP. Example 2 details the sequence of the data transfer (dial) procedure. Note that Example 1 provides a clue where the main memory program is involved. Example 2 is concerned strictly with the transfer of data between ACUF and automatic calling device.

Individual programs must take into consideration the type of automatic calling unit and its interface. It is suggested that readers also obtain a copy of the Electronic Industries Association (EIA) RS-366 document for specifications of the signals that have been defined in this appendix.

In the examples, the terms automatic calling CCP (present in the ACUF) and communcations line CCP (present in the communications line adapter) are used to reference the CCP in question. Abbreviations for the fields in the FLAP register are used. Refer to the list below for appropriate meaning and bit position.

Abbreviation	Meaning	FLAP Register and Bit Position
ACR	Abandon Call and Retry	Bit 3 of FR5
CRQ	Call Request to Automatic Calling Device	Bit 0 of FR2
DLO	Data Line Occupied	Bit 2 of FR5
DPR	Digit Present	Bit 1 of FR2
DSC	Distant Station Connected	Bit 0 of FR5
DSR	Data Set Ready	In communications line adapter
DTR	Data Terminal Ready	In communications line adapter
PND	Present Next Digit	Bit 4 of FR5

EXAMPLE 1: GENERAL SEQUENCE OF CCP AND MMP INTERFACE

1. Start the automatic calling CCP to place a call. When all dial digits have been sent, the automatic calling CCP then starts a Data Set Scan (defined in Section 5) to restart itself upon detecting the turn-on of either DSC or ACR. The automatic calling CCP finally issues a WAIT.

2. The main memory program sets up CCBs for the associated communications line adapter. It starts the communications line CCP which then turns on DTR (bit 0 of LR 2, typically) of the communications line adapter. After turning on DTR, the communications line CCP starts a Data Set Scan to restart itself upon detecting the turn-on of DSR (Data Set Ready). The communications line CCP finally issues a WAIT.

3. As provided for in step 1 above, the automatic calling CCP is restarted upon detecting the turn-on of DSC or ACR. If ACR is on, then the automatic calling CCP interrupts the main memory program, turns off CRQ, and issues a WAIT. If DSC is on, and the termination method is by the automatic calling device dropping the CRQ at the end of the call, the automatic calling CCP issues a WAIT. If DSC is on and the termination is by the associated communications line adapter dropping DTR at the end of the call, then the automatic calling CCP turns off CRQ and issues a WAIT.

4. As provided for in step 2 above, the associated communications line adapter CCP is restarted upon detecting the turn-on of DSR. Data transfer may now be started by the communications line CCP. If the termination method is by the first method (automatic calling unit dropping CRQ) the associated communications line CCP must exit by issuing a WAIT and the main memory program must restart the automatic calling CCP to turn off CRQ. If the termination is by the second method (data set dropping DTR), the communications line CCP turns off DTR and exits with a WAIT.

EXAMPLE 2: TRANSFER OF EACH INDIVIDUAL DIGIT

The transfer of each individual digit involves the following sequence of events:

1. Automatic calling device - Turns on PND (hardware)

2. Automatic calling CCP - Detects PND via Data Set Scan

3. Automatic calling CCP - Outputs digit to LR1

4. Automatic calling CCP - Outputs LR2 with DPR on (see Note)

5. Automatic calling device - Turns off PND

6. Automatic calling CCP - Detects PND off via Data Set Scan

7. Automatic calling CCP - Outputs FLAP register 2 with DPR off (see Note)

Repeat this sequence of events for each digit.

NOTE

In order to maintain the connection, CRQ and DIP must be output in the on condition each time an output command to FR2 is issued.

EXAMPLE 3: CCP AND CHANNEL REQUEST INTERUPT SEQUENCE

1. Call Initiation

   Set CRQ to One
   Set SIP to One

   This signals the automatic calling device that a call is to be originated.

2. First Channel Request Interrupt (CRI) and every other Odd-Numbered CRI (e.g., 1, 3, 5, 7, 9, etc.)

   Output Digit
   Set DPR to One

   This signals the automatic calling device that a digit is available on the output lines.

3. Second Channel Request Interrupt and every Odd-Numbered Channel Request Interrupt except last (2, 4, 6, 8, 10, etc.)

   Set DPR to Zero

   This completes the handshaking with the automatic calling device for each digit.

4. Last Channel Request Interrupt

   Set DPR to Zero
   Set DIP to Zero

   Setting DIP to zero prevents the DCM9110 from generating further channel request interrupts.

Status of Automatic Calling Device

Status of the automatic calling device is available in FR5. The main memory program may read this status at any time via an Input Data Set Status (function code: 1C) command to either Processor channel containing the ACUF. Assuming that conditions were such that a call can be placed (Distant Station Connected off, Power Indicator on, Data Line Occupied off), an automatic calling CCP can access FR5 directly as the call proceeds. The Data Set Scan capability of the Processor can also be used to either start the automatic calling CCP or interrupt the main memory program upon transition of any specific bit in FR5. This facility may be used to reactivate the automatic calling CCP when Call Originating Status=1 (is on), indicating that the connection to the called station has been made.

Avoiding Possible Race Condition During Call Abort

If Call Request has been turned off prior to Distant Station Connected coming on (for example, during the abortion of a call attempt) a possible race condition could occur. To prevent this, Data Terminal Ready in the associated communications line adapter should be turned off.

PHYSICAL INTERFACE TO DATA COMMUNICATIONS EQUIPMENT AND DATA TERMINAL EQUIPMENT

The circuit interface of the ACUF to the automatic calling device is in compliance with EIA RS-366. Data communications equipment using the CCITT V24 (line) and V25 (automatic calling) functional interfaces which are electrically compatible with EIA RS-366 (e.g., CCITT V28) may also attach. The signal interface and connector pin assignments of the ACUF cable at the connector which connects to the automatic calling equipment are defined in Table C-2.

Table C-2. Automatic Calling Equipment/ACUF Interface Signals

EIA Pin No.	To/From DCE	Function	EIA RS-366	CCITT Equivalent
2	T	Digit Present	DPR	211
3	F	Abandon Call and Retry	ACR	205
4	T	Call Request	CRQ	202
5	F	Present Next Digit	PND	210
6	F	Power Indication	PWI	213
7	-	Signal Ground	SG	201
13	F	Distant Station Connected	DSC	204
14	T	Digit Lead (LR1 bit 7)	NB1	206
15	T	Digit Lead (LR1 bit 6)	NB2	207
16	T	Digit Lead (LR1 bit 5)	NB4	208
17	T	Digit Lead (LR1 bit 4)	NB8	209
22	F	Data Line Occupied	DLO	203
1	-	Protective Ground	AA	212

Appendix D
NEW MULTILINE CONTROLLER BROADBAND HDLC/SYNCHRONOUS ADAPTER

INTRODUCTION

The New Multiline Controller Broadband HDLC/Synchronous Adapter (NBHSA) is a quarter-sized board which attaches to the New Multiline Communications Processor (NMLCP). The NBHSA supports one communication line operating in full or half duplex communications. The protocol on the line can be synchronous or High level Data Link/Synchronous Data Line Control (HDLC/SDLC). The maximum line speed is 100K bps (8-bit characters). The support of SDLC NRZI bit coding is provided by the EIA RS-232C/V.24 FLAP.

The NBHSA, in conjunction with an X.21 FLAP, provides for full X.21 support including Call Establishment with the Byte Timing option.

Connections to line equipment (DCE or DTE) are accomplished via Flexible Line Adapter Packages (FLAPs) and FLAP bus cables. The NBHSA has a connector for one FLAP bus cable.

SOFTWARE OVERVIEW

A requirement in the development of the NMLCP is that existing CCPs written for use with the Wideband Communications Line Adapter (WCLA) on the MLCP are, on an object code basis and without need for change, usable for operating the NBHSA in

Synchronous mode. New CCPs written for the NBHSA for HDLC or synchronous operation operate without change if and when future NBHSA versions using different hardware are implemented.

The prime visibility of adapters on the MCLP to the CCP for control, data access, and status-sensing purposes is that of a set of Line Registers (LRs). In the NBHSA, this same visibility and assignment of LRs and their contents will be maintained and will continue to be maintained in future NBHSA versions. Firmware will map the LR content into/from the appropriate spots in the NBHSA and FLAP.

Newly visible to the CCP is a set of FLAP Registers (FRs) located in the FLAP associated with the line. These registers implement data set control and data set status functions associated with the DCE. Adapters used on the MLCP did not use FLAPs but instead implemented the data set control and status functions associated with EIA RS-232C type DCEs through bit positions in the LRs.

Two other new registers are also visible: an Adapter Control register and an Adapter Status register. These contain the adapter-related bits of LR2 and LR5, respectively. There are performance advantages to accessing either these registers or FR2 and FR5 instead of LR2 and LR5.

In the NMLCP/NBHSA configurations, firmware will automatically map LR bits to FR register bits on OUT type CCP instructions, and will assemble FR bits on IN type CCP instructions as appropriate.

Associated with the USRT in the NBHSA is a set of registers for control and status indication of the USRT itself. Adapters on the MLCP used USRTs which had limited capability and flexibility. All required control and status was provided by bit positions in the LRs. Firmware automatically provides mapping of these LR bits into and from the appropriate USRT registers. In order that existing CCPs may be used with the NBHSA, necessary access to USRT registers not represented in the LRs must not require CCP instructions. This also allows future replacement of the USRT component with a newer type without impacting the viability of existing CCPs. USRT control and status sensing required to be done on a dynamic basis (e.g., error sensing, error reset, etc.) will be performed automatically by firmware. Thus, replacement of the USRT with a newer type would impact firmware, but would not affect the CCP.

Figure D-1 shows the register sets associated with the NBHSA and their means of access by software.

In order to implement more complex DCE interfaces (e.g., CCITT X.21) than were supported on the MLCP, new CCPs must be written directly addressing the FLAP registers.

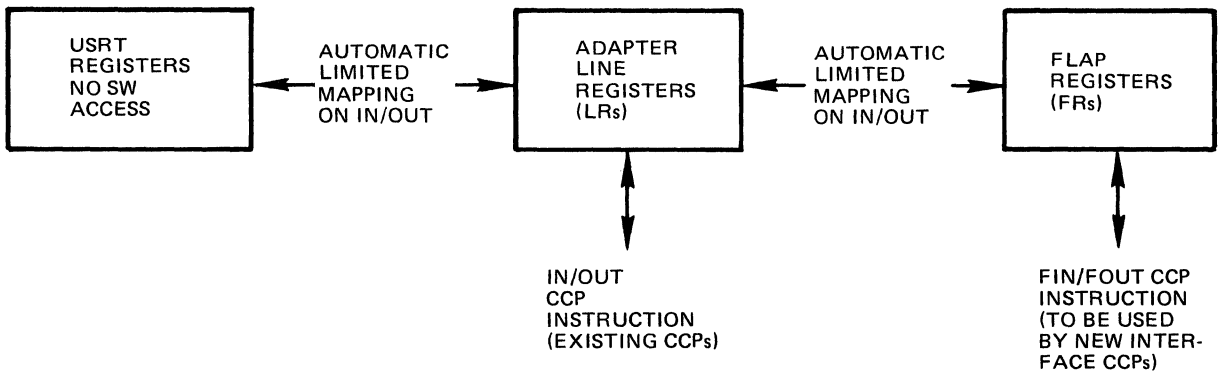


Figure D-1. NBHSA Register Access

To summarize, software visibility to the NBHSA in the case of existing CCPs is exclusively through the LRs. Firmware loads certain default values into the USRT registers upon initialization. These are values needed for USRT operation which are not supplied via the LRs. Additionally, firmware maps LR bits to appropriate bit positions in the USRT registers. With the initial values plus subsequent LR-to/from-USRT register mapping, existing CCPs and drivers will operate without modification. It wasnecessary to add new device IDs to the CLM since the ID returned by the NBHSA Synchronous mode will differ from those returned by the WCLA.

New functionality over and above that of the MLCP is implemented by the utilization of LR bits which were previously unused. As mentioned above, new CCPs written to capitalize on this functionality will continue to be usable since FLAPs will be retained in future designs and the new functionality specified in the subsection entitled "Line Registers" in Appendix B will survive any USRT changes.

Existing CCPs written for the WCLA on the MLCP will run without change on the NBHSA. New CCPs must be written to operate the NBHSA in HDLC mode.

SYNCHRONOUS DATA FORMAT

The NBHSA may be configured for character-oriented synchronous format including bisynchronous or Binary Synchronous Communication (BSC).

All data is transmitted as a serial stream of binary digits. The receiving station operates in step with the transmitting station through the recognition of a specific bit pattern (synchronous pattern) at the beginning of each transmission. The character consists of 5, 6, 7, or 8 data bits plus parity if used. The data is transmitted least significant bit first (see Figure D-2).

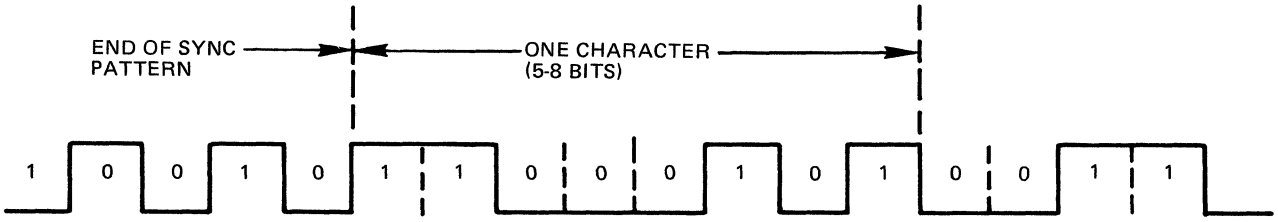


Figure D-2. Synchronous Data Format

The general line format for a block of data is as shown in Figure D-3. During idle periods in transmission within or between blocks, synchronization characters are transmitted as time fill. The Block Check Character (BCC) may be based on either a Longitudinal Redundancy Check (LRC) or Cyclic Redundancy Check (CRC). The Block Check character is provided by or used by the Processor.

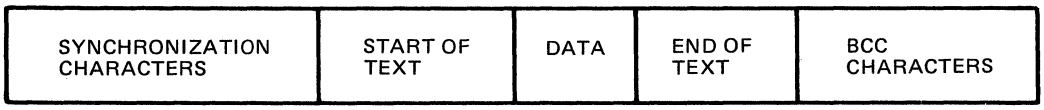


Figure D-3. General Synchronous Line Format

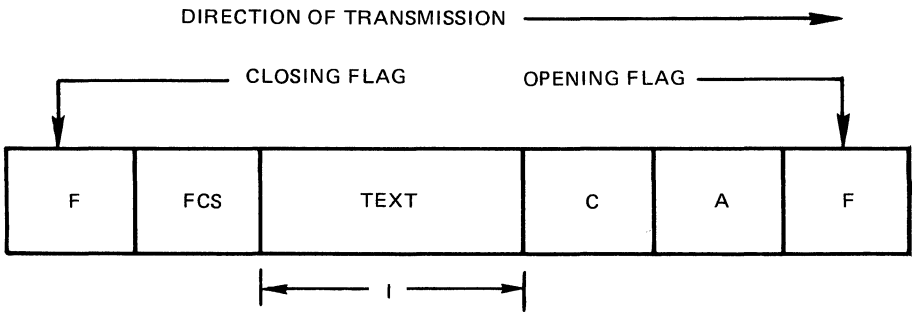
HDLC/SDLC FRAME STRUCTURE

The format of all data transferred between the communication equipment conforms to the High-Level Data Link Control (HDLC) standards. The fields of the frame and other HDLC/SDLC attributes are described here for information purposes, along with the relationship of hardware and software regarding implementation responsibility.

General Format of HDLC/SDLC Frames

In the HDLC/SDLC communications link control procedure, all transmissions are in frames. The format of a typical frame is shown in Figure D-4.

The Flag sequence and the A, C and FCS fields each consist of a multiple of 8 bits. For contiguous frames, a single flag sequence may serve as the closing flag for the first frame and the opening flag for the next.



WHERE THE FIELDS ARE:

F	–	FLAG SEQUENCE (01111110)
A	–	ADDRESS FIELD (8 BITS; OPTIONALLY n X 8 BITS)
C	–	CONTROL FIELD (8 BITS; OPTIONALLY 16 BITS)
I	–	INFORMATION FIELD (ANY NUMBER OF 8-BIT CHARACTERS, OPTIONAL)
TEXT	–	TEXT FIELD (ANY NUMBER OF 8-BIT CHARACTERS, IF PRESENT)
FCS	–	FRAME CHECK SEQUENCE (16/32 BITS)

Figure D-4. General Format of HDLC/SDLC Frame

If the next frame is not ready after transmission of a frame is complete, the NBHSA transmitter will send interframe time fill until data for the next frame is available. The receiver will discard interframe time fill. Interframe time fill consists of continuous Flags or Idle pattern. When interframe fill is being transmitted, the link is still considered to be in an active state; i.e., the transmitter still has the right to recommence transmission of frames.

There is no provision for intraframe time fill. When such a situation arises, Frame Abort procedure is followed. Abort is the procedure by which a transmitter may terminate the current frame in an unusual manner such that the receiver will ignore the frame. Abort sequences are generated by the transmitter on an output error condition (underrun). A Frame Abort consists of the transmission of at least seven but less than fifteen contiguous Ones. When the receiver detects an Abort Sequence, it notifies the receiving CCP program through status. Transmission of fifteen or more Ones is defined to put the link into an IDLE state; this may occur following an Abort or following a completed frame. When entry into Idle state is detected, the receiver reports it to the receiving program via status.

The minimum number of bits between the Flags in a valid frame is 32 (consisting of 8 Address, 8 Control, and 16 FCS). A frame of less than 32 bits should be discarded by the receiving CCP.

In order to prevent accidental and unwanted Flag or Abort sequences from occurring between opening and closing Flags, a Zero bit insertion procedure is implemented by the NBHSA whereby a Zero is added after every five contiguous One bits during transmit and removed by the receiver.

While sending a frame, the transmitter examines the bit stream between the beginning and ending Flags. Whenever a sequence of five contiguous Ones is detected, a Zero is inserted into the data stream after the fifth One. (Note that this procedure applies to the contents of all fields between Flags, which includes the last five bits of the FCS.)

The receiver continuously monitors the received bit stream between Flags. When five contiguous Ones are detected followed by a Zero, the Zero is removed. (If a One follows the five contiguous Ones, then the receiver is in the process of receiving either a Flag or Abort sequence, but which of these it is cannot be determined until the next bit is received.)

Fields of the Frame

FLAG SEQUENCE

The Flag sequence is an 8-bit sequence (01111110) which delimits the beginning and end of each frame, and is used as a synchronization character. When contiguous frames are transmitted, a single Flag may serve as both the closing Flag for the first frame and the opening Flag for the second. Flags may also be used as interframe time fill.

The transmitter generates opening and closing Flags at the appropriate time. During idle periods between frames, Flag, Idle or Abort sequences (as determined by the TIFM bit in the NBHSA configuration register) are generated and transmitted. The receiver recognizes Flag sequences as opening or closing Flags or as interframe time fill but does not transfer them to the CPU memory.

ADDRESS FIELD

The address field (containing the secondary station link address) directly follows the opening Flag and normally consists of one 8-bit byte. Optionally, for extended addressing, this field may contain multiple octets. In this case all octets in the field except the last will have a Zero in their Continuation bit position.

The specification as to whether or not Extended mode applies is under program control and must be set up by prior agreement between the transmitter and receiver.

Transmitted address field bytes must be program generated and transferred like data to the NBHSA for transmission. Received address field bytes may be input to the main frame's memory as part of the data block or may be discarded by the receiver CCP.

CONTROL FIELD

The control field (containing commands or responses, and sequence numbers) directly follows the A field, and normally consists of one octet. In the optional Extended Control Field mode, a second octet follows the first. The first bit of the octet (i.e., the low order bit, bit 7) (first octet if in Extended mode) provides information about the format of the remainder of the frame. When a Zero, it indicates the frame is in Information Transfer format. When a One, it indicates the frame is in Supervisory format or Nonsequenced format. All data in the frame is in 8-bit bytes.

The specification as to whether or not Extended mode applies is under program control and must be set up by prior agreement between the transmitter and receiver.

Output control field bytes must be program generated and transferred as data to the NBHSA for transmission. Received Control bytes may be input to the main frame memory as part of the data block or may be handled by the receiver CCP.

INFORMATION FIELD

The optional information field directly follows the C field. This field may contain any number of text characters (including none at all). The text field ends with the start of the FCS field which occupies the 16 or 32 bits prior to the next Flag.

None of the data in this field is of significance to the NBHSA. The NBHSA packs/unpacks the 8 bits in the bytes transferred between it and main frame memory when receiving/transmitting.

Output text data must be program generated and transferred to the NBHSA for transmission. When the CCB range is exhausted, the CCP sets an indication in the NBHSA which causes it (after transmitting the last data character) and the CRC residue to generate and transmit the closing Flag.

Input text data is input to the main frame memory as part of the data block. Recognition of the closing Flag by the NBHSA causes it, after handling the last data character and the CRC residue, to signal the completion of the frame to the CCP. The FCS and closing Flag are not transferred to the main frame memory.

FRAME CHECK SEQUENCE

All frames include, for error detection purposes, a 16- or 32-bit frame check sequence just prior to the closing flag. An algebraic procedure based on a modulo 2 division process using a generation polynomial is used to generate and check the FCS.

At the transmitter, the initial remainder of the division is set to all Ones. This initial remainder is then modified by division by the generator polynomial. This division is performed on the contents of the address, control, and information fields, excluding Zero bits inserted for transparency. When these fields have completed the division process, the One-complement of the resulting remainder is transmitted (high order bit first) as the FCS.

At the receiver, the initial remainder is preset to all Ones, and the same division process takes place on the serial incoming bits. All bits between the opening and closing Flag are included, except Zero bits inserted for transparency. In the absence of transmission errors, the final remainder for the CCITT FCS is 1111000010111000 LSB to MSB in reading from left to right.

The CCITT generation polynomial is $X^{16} + X^{12} + X^5 + 1$. The adapter has the capability to generate and check this FCS. A 32-bit FCS may also be used; however, this FCS must be generated and checked by the Processor.

The checking polynomial for output (transmit) frames is automatically generated during transmission of the A, C, and I fields. When transmission of these fields is complete, the FCS is appended to the end of the frame before the closing Flag is transmitted.

Similarly, when receiving a frame, the necessary operations are automatically performed on the incoming data stream. When the closing Flag is received, the generated remainder is compared with the correct remainder for an errorless transmission. As a result of this comparison, an FCS error is made available to the programmer as part of the interrupt status.

Order of Bit Transmission

The flag, address, control and information fields are transmitted Least Significant Bit (LSB) first. The FCS is transmitted Most Significant Bit (MSB) first.

Abort

Abort is the procedure by which a station in the process of sending a frame ends the frame in an unusual manner such that the reveiving station will ignore the frame.

On transmit, the NBNHSA will automatically send an abort of eight Ones on underrun. On receive, a sequence of seven Ones will be detected as an abort.

Transparency

HDLC/SDLC provides transparency for data coded in the information field. The occurrence of the Flag sequence within the frame is prevented via a Zero bit insertion technique.

The transmitter inserts a Zero bit following five contiguous One bits anywhere between the opening and closing Flag of a frame. The receiver continuously monitors the received bit stream. Upon receiving a Zero bit followed by five contiguous One bits, the receiver inspects the following bit. If a Zero, the five One bits are passed and the Zero bit is deleted. If the sixth bit is a One, the receiver inspects the seventh bit. If the seventh bit is a Zero, a Flag has been received; if a One, an abort sequence has been received.

Interframe Time Fill

The NBHSA is capable of sending either Flag sequences or continuous Ones between frames under control of the CCP.

Intraframe Time Fill

The HDLC/SDLC protocol does not provide for intraframe time fill. All bytes within a frame are contiguous.

Receive Idle Link State

Receipt of fifteen (or more) contiguous One bits indicates the idle link state. The NBHSA reports this condition to the CCP via the status register (LR5).

BASIC FUNCTIONS

The new Multiline Controller Broadband HDLC/Synchronous Adapter (NBHSA) provides an interface for one synchronous clocked data communication line. This line supports either half or full duplex data transmission.

The NBHSA accommodates bit and character-synchronous protocols up to 100K bps (8-bit characters) via a bit serial DTE/DCE interface.

Character sizes of 5 to 8 bits are supported in Character-Synchronous mode. In Bit-Synchronous mode, all characters are 8 bits only.

All configurations fields are programmable via the Processor.

REGISTERS

Each channel of the NBHSA is controlled and programmed via a set of registers. The content of these registers depends upon whether the NBHSA is operating in HDLC (Bit-Synchronous) or synchronous (Character-Synchronous) mode.

Register Format in HDLC Mode

The format of these registers in HDLC mode is shown in Figures D-5 and D-6.

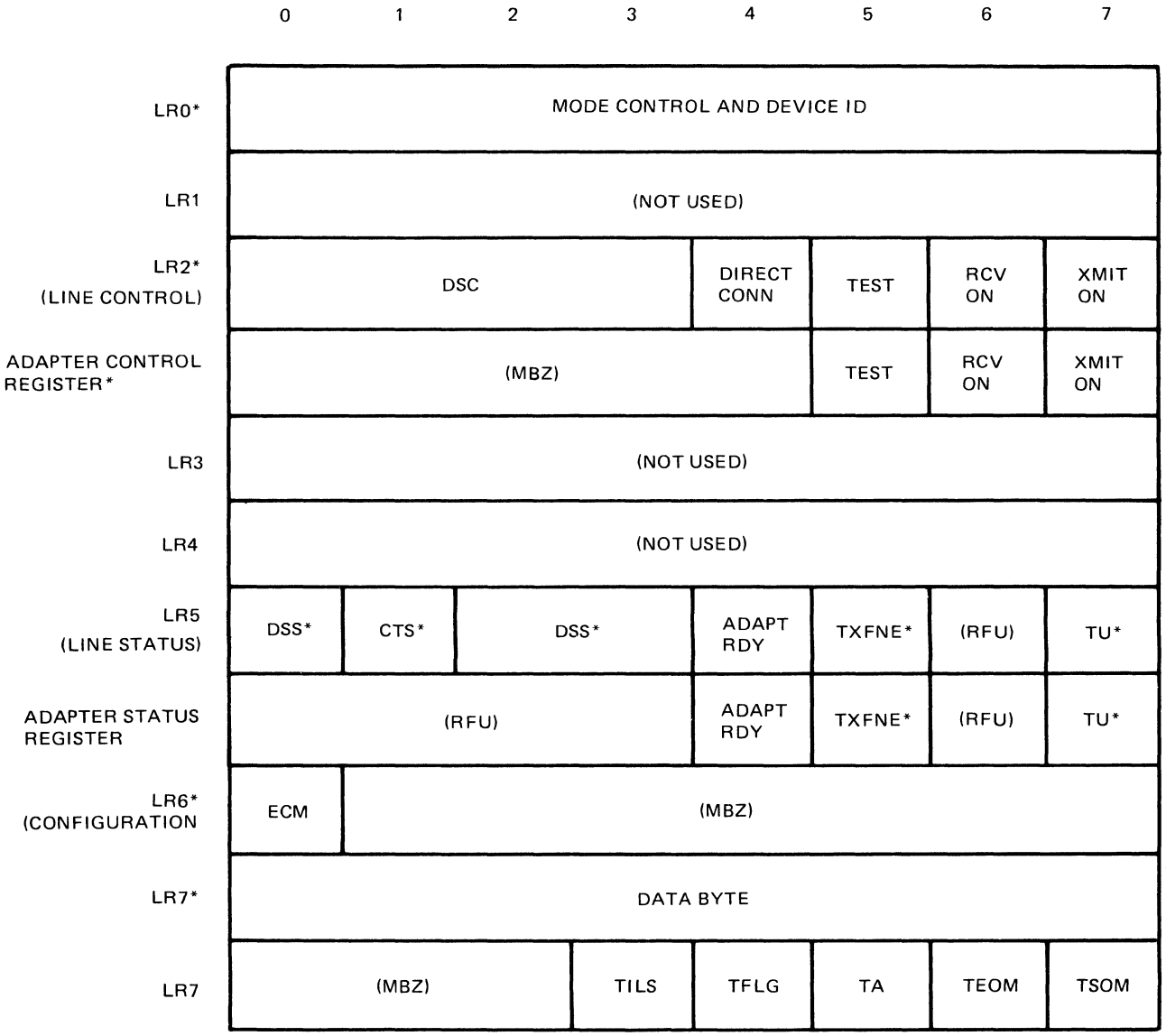


Figure D-5. NBHSA Registers (Receive Channel - HDLC Mode)

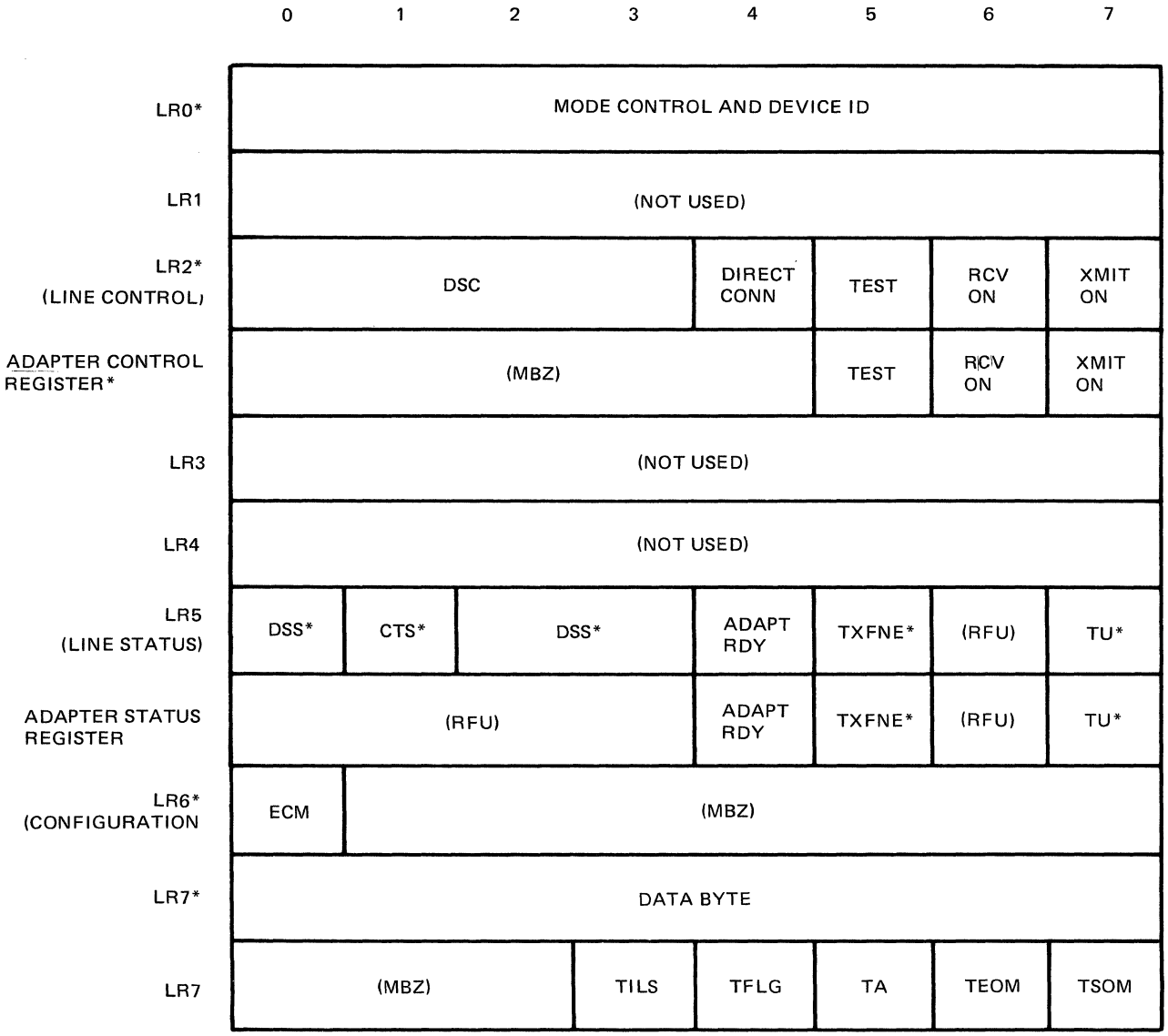


Figure D-6. NBHSA Registers (Transmit Channel - HDLC Mode)

RECEIVE CHANNEL REGISTER DEFINITIONS

All formats shown below follow the same data format and bit positions as the MLCP.

Receive

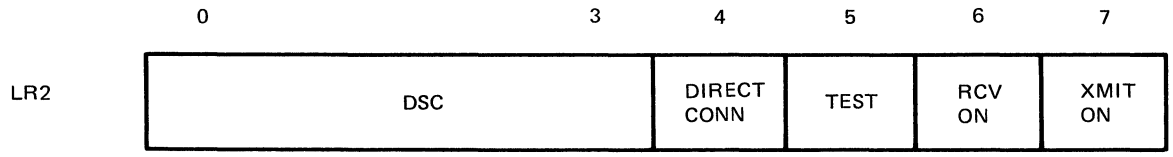
The receive channel includes a FIFO stack for message information buffering. The stack consists of 128 locations of 8 bits each. Every other location is used for data; the alternate locations are used for data status information. The topmost two locations of the stack are read by the Processor via an IN LR7 instruction; this also pops up the stack. The adapter loads the stack at the bottom with data and data status information as characters are received from the line.

RECV and IN LR1 instructions are illegal in HDLC mode.

LR0 - See the last two subsections in this appendix.

LR1 - Not Used.

LR2 - Line Control - This LR can be written into only by the Processor. LR2 for the even (receive) channel is the same physical register used by the odd (transmit) channel. Thus a transmit LR2 change also changes the receive LR2 and vice versa.



DSC = Data Set Control. These bits are used by the attached FLAP. On an OUT LR2 instruction, the adapter delivers bits 0-3 to FR2 bits 0-3. Usage of the bits is thus FLAP-dependent.

DIRECT CONN = Direct Connect. This bit is used by the attached FLAP to select whether an External (Modem) clock or an Internal (Processor provided) clock is to be used for the serial interface. On an OUT LR2 instruction, this bit is delivered by the adapter to FR4 bit 1.

TEST = This bit sets up loopback tests by enabling the test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Channel Request Interrupts will be sent to the Processor if the receive FIFO has information.

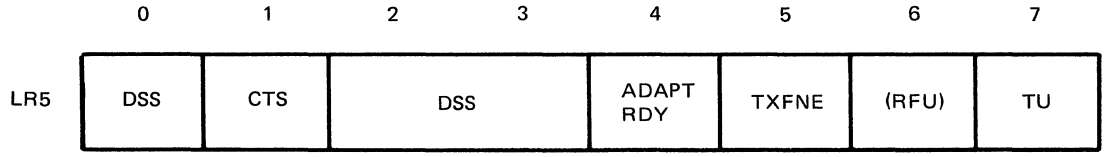
XMIT ON = Transmitter On for this line:

0 = Transmitter Off and inactive
1 = Transmitter On. Channel Request Interrupts will be sent to the Processor if the transmit FIFO has availiable space.

LR3 and LR4 - Not Used.

LR5 - Line Status - LR5 is used for status purposes by both channels and should only be read.

When an IN LR5 instruction is performed, the adapter copies bits 0 through 3 of FR5 into bits 0 through 3 of LR5 and then delivers the entire contents of LR5 to the Processor (bits 4-7 are provided by the adapter).



DSS = Data Set Status. The definition of these bits is FLAP dependent.

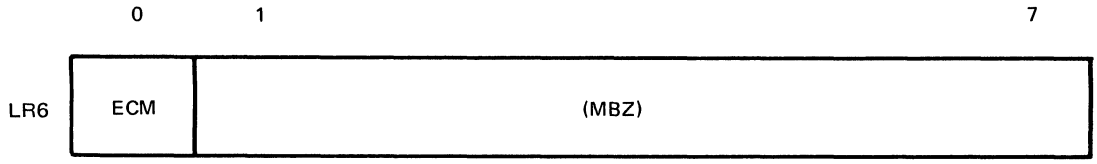
CTS = Clear To Send. Allows the transmit USRT to serialize the transmit data which is delivered to it. Transmit character requests require CTS as well as Transmitter On.

ADAPT RDY = Adapter Ready. When this bit is a One, it indicates that the Receive FIFO is not empty (i.e., contains information for the Processor).

TXFNE = Transmit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

TU = Transmitter Underrun. When this bit is a One, it indicates that the CCP did not service the transmitter fast enough and the NBHSA aborted the frame. The bit is reset by setting the transmit TSOM in LR7.

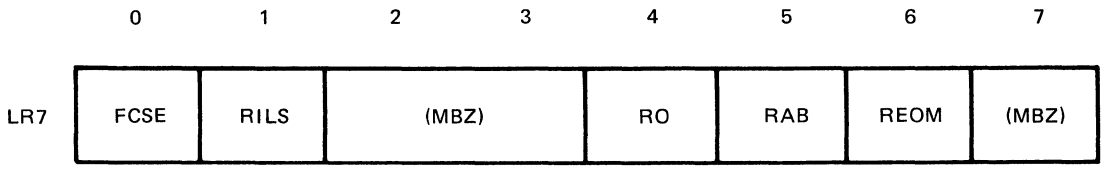
LR6 - Configuration - This LR can only be written into by the Processor.



ECM = Error Control Mode. Specifies the CRC checking facili- ties of the adapter to be used:

1 = OFF (No checking by the adapter)
0 = CCITT 16-bit CRC

LR7 - Data and Data Status - Reading LR7 accesses the receive FIFO stack. The stack consists of 128 locations of 8 bits each. Every other location contains data and the alternate locations contain data status information. The IN LR7 instruction causes the two topmost locations of the stack to be read into the Processor and pops up the stack. The data status byte is placed in LCTll and the data byte is delivered to R. The Processor indicators Zero and SB are set according to the contents of the data status which has just been set into LCT 11. Immediately after performing the IN LR7 instruction, a BSF and BST instruction may be used to test for a NULL state of the data status byte. (Note a BZF or BZT instruction should not be used here if the program capability with other Honeywell communication processors is to be maintained.) The format of the data status byte is as follows:



FCSE = Receive Frame Check Sequence Error. When this bit is a One it indicates that the frame was received in error (CRC check failed). This bit is normally Zero and has a valid meaning only when REOM is also set.

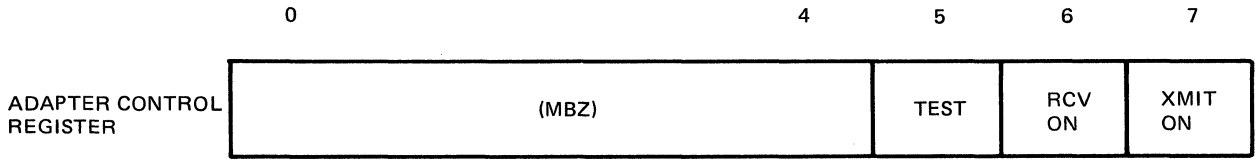
RILS = Receive Idle Link State. When this bit is a One, it indicates that fifteen or more Ones have been received and the input line is in an Idle Link state. Only one RILS indication will be indicated between frames.

RO = Receive Overrun. When this bit is a One it indicates that the NBHSA was not serviced fast enough (the FIFO is full) and one or more characters or frames in the NBHSA data path have been overwritten and lost. After the indication, the CCP should discard all data up to and including the data accompanied by REOM status. Data after REOM would be from a subsequent frame.

RAB = Receive Abort. When this bit is a One and REOM is a One it indicates that the frame was terminated with an Abort sequence of seven or more Ones.

REOM = Receive End of Message. When this bit is a One it indicates that the frame was terminated. The associated data byte is the last byte of the frame.

Adapter Control - This register is written into via the ACTL instruction.



TEST = This bit sets up loopback tests by enabling the test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

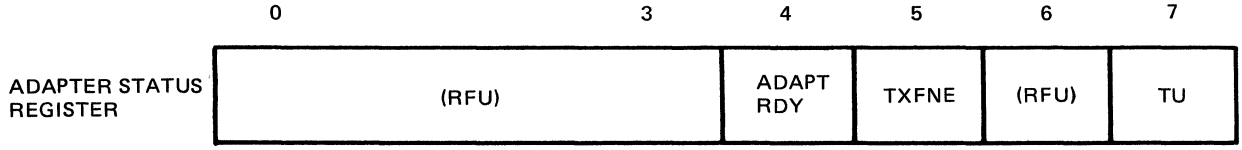
RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Channel Request Interrupts will be sent to the Processor if the receive FIFO has information.

XMIT ON = Transmitter On for this line:

0 = Transmitter Off and inactive
1 = Transmitter On. Channel Request Interrupts will be sent to the Processor if the transmit FIFO has availiable space.

Adapter Status - This register is read via the AST instruc-struction.



ADAPT RDY = Adapter Ready. When this bit is a One it indicates that the Receive FIFO is not empty (i.e., contains information for the Processor).

TXFNE = Transmit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

TU = Transmit Underrun. When this bit is a One, it indicates that the CCP did not service the transmitter fast enough and the NBHSA aborted the frame. The bit is reset by setting the transmit TSOM in LR7.

TRANSMIT CHANNEL REGISTER DEFINITIONS

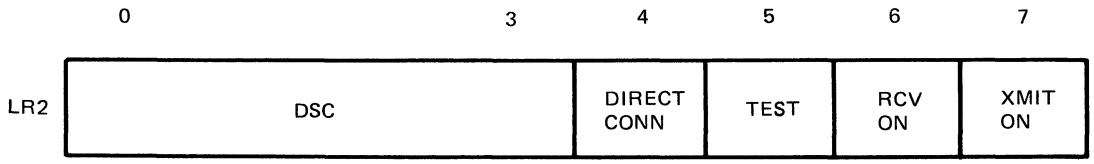
The transmit channel includes a FIFO stack for message information buffering. The stack consists of 128 locations of 8 bits each. Every other location is used for data; the alternate locations are used for data control information. The bottom two locations of the stack are written into by the Processor via an OUT LR7 instruction. The adapter reads out the stack at the topmost (output) location.

SEND and OUT LR1 instructions are illegal in HDLC mode.

LR0 - See the last two subsections in this appendix.

LR1 - Not Used.

LR2 - Line Control - This LR can only be written into by the Processor. This is the same physical register as used by the receive channel.



DSC = Data Set Control. These bits are used by the attached FLAP. On an OUT LR2 instruction, the adapter delivers bits 0-3 to FR2 bits 0-3. Usage of the bits is thus FLAP-dependent.

DIRECT CONN = Direct Connect. This bit is used by the attached FLAP to select whether an External (Modem) clock or an Internal (Processor provided) clock is to be used for the serial interface. On an OUT LR2 instruction, this bit is delivered by the adapter to FR4 bit 1.

TEST = This bit sets up loop-back tests by enabling the test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

RCV ON = Receiver On for this line:

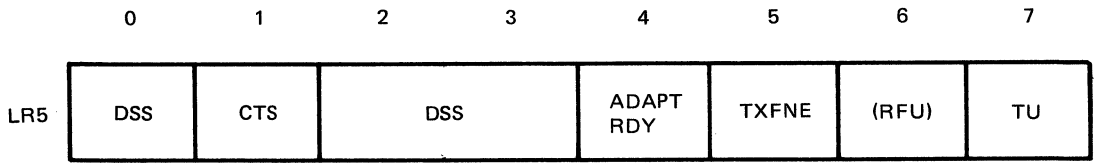
0 = Receiver Off and inactive
1 = Receiver On. Channel Request Interrupts will be sent to the Processor if the receive FIFO has information.

XMIT ON = Transmitter On for this line:

0 = Transmitter Off and inactive
1 = Transmitter On. Channel Request Interrupts will be sent to the Processor if the transmit FIFO has availiable space.

LR3 & LR4 - Not Used.

LR5 - Line Status - This LR can be read by but not written into by the Processor. The same register is accessed by both the receive and transmit channels.



DSS = Data Set Status. The definition of these bits is FLAP dependent.

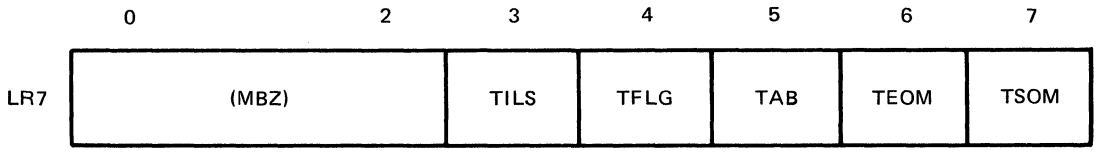
CTS = Clear To Send. Allows the transmit USRT to serialize the transmit data which is delivered to it.

ADAPT RDY = Adapter Ready. When this bit is a One, it indicates that the Transmit FIFO is not full (i.e., it can accept information from the Processor).

TXFNE = Transmit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters. See subsection entitled "Data Transfer").

TU = Transmitter Underrun. When this bit is a One, it indicates that the CCP did not service the transmitter fast enough and the NBHSA aborted the frame. The bit is reset by setting TSOM in LR7.

LR6 - Configuration - This LR can be written into by the Processor. The same register is accessed by both channels.



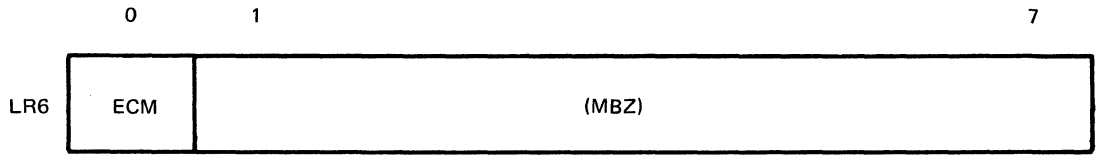
ECM = Error Control Mode. Specifies the CRC generation facilities of the adapter to be used:

1 = OFF (No CRC generation)
0 = CCITT 16-bit.

LR7 - Data and Data Control - Writing LR7 accesses the transmit FIFO stack. The stack consists of 128 locations of 8 bits each. Every other location contains data; alternate locations contain data control information. The OUT LR7 does not check the Transmit Underrun indicator, but may be checked by the CCP on a frame basis.

The OUT LR7 instruction causes the data byte in R and the Data Control byte in LCT 43 to be delivered to the stack.

The format of the data control byte is as follows (note that only one bit at a time can be set in this byte):



TILS = Transmit Idle Link State. When this bit is a One, it indicates that the line is to be placed in an idle link state by transmission of at least 15 Ones.

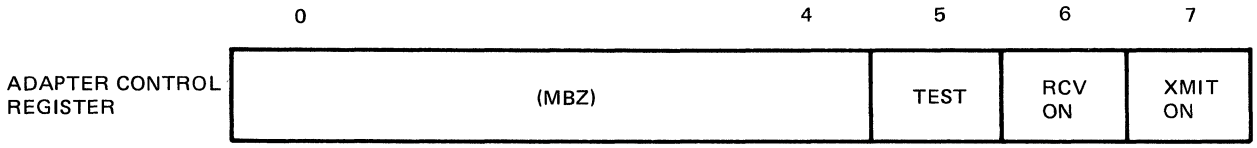
TFLG = Transmit Flag. This bit may be set in data control bytes which are output to the adapter between messages; the associated data byte (which is output from R) is ignored by the adapter. Each occurrence of a TFLG control byte will cause the adapter to generate and transmit one Flag charac- ter. This function may be used to provide a specific number of Flags between messages.

TAB = Transmit Abort. When this bit is a One it indicates that the frame is to be terminated immediately with an Abort sequence and the associated data byte and FCS are not transmitted.

TEOM = Transmit End Of Message. This bit will be set to One in LCT 43 by the CCP before it outputs the last data character of the frame to the adapter. The data byte position in the FIFO which contains the last data character of the frame will thus have this bit set in the associated data control byte location. When sensed by the adapter, this bit causes the transmitter to begin the End of Frame sequence (see subsection entitled "Receive End of Frame").

TSOM = Transmit Start of Message. This bit is to be set to One in LCT 43 by the CCP before it outputs the first character of the frame to the adapter. The data byte position in the FIFO which contains the first character of the frame (an address character) will thus have this bit set in the associated control byte location. When sensed by the adapter, this bit causes the TU bit in LR5 to be reset and the CRC facilities on the adapter to generate a new CRC.

Adapter Control - This register is written into via the ACTL instruction.



TEST = This bit sets up loop-back tests by enabling the test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

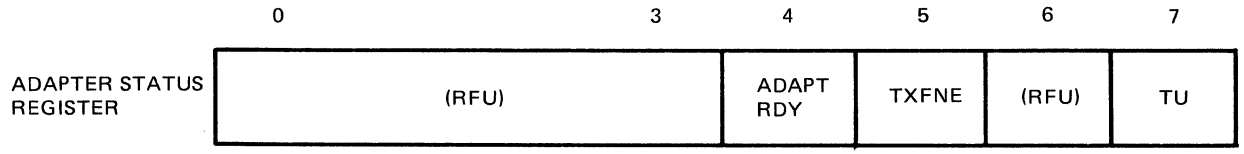
RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Channel Request Interrupts will be sent to the Processor if the receive FIFO has information.

XMIT ON = Transmitter On for this line:

0 = Transmitter Off and inactive
1 = Transmitter On. Channel Request Interrupts will be sent to the Processor if the transmit FIFO has availiable space.

Adapter Status - This register is read via the AST instruction.



ADAPT RDY = Adapter Ready. When this bit is a One it indicates that the Transmit FIFO is not full (i.e., it can accept information from the Processor).

TXFNE = Transmit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

TU = Transmitter Underrun. When this bit is a One, it indicates that the CCP did not service the transmitter fast enough and the NBHSA aborted the frame. The bit is reset by setting TSOM in LR7.

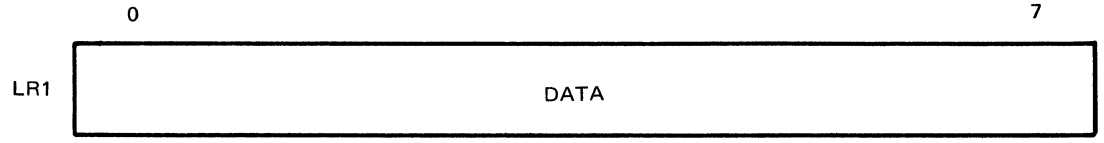
Register Format in Synchronous Mode

The format of the registers in Synchronous mode is shown in Figure D-7.

RECEIVE CHANNEL REGISTER DEFINITIONS

LR0 - See the last two subsections in this appendix.

LR1 - Data - This LR is constructed as a FIFO stack. The stack consists of 128 locations of 8 bits each. These bits are data and are read by the Processor through LR1. It is the topmost (output) location of the stack which is read by the Processor. The act of reading LR1 pops up the stack, thus a given data byte can only be read once by the Processor. The adapter loads the stack at the bottom (input) location with data, as characters are received from the line. Bit 7 is the first bit received. If the byte size is less than 8, the most significant bits will be Zero with the incoming data all right-justified. This register can only be read by the Processor. (An attempt to write it will cause unspecified results.)



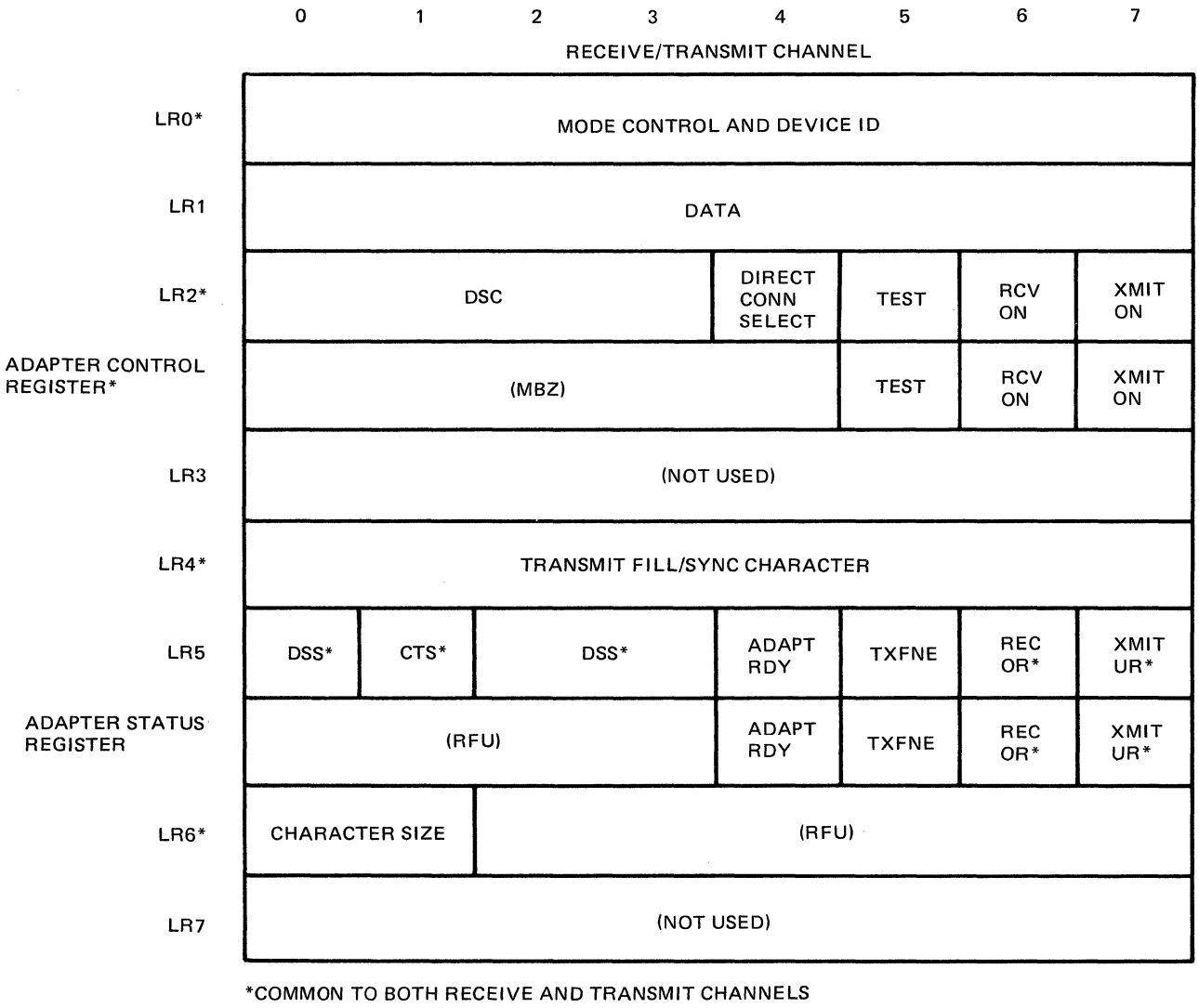
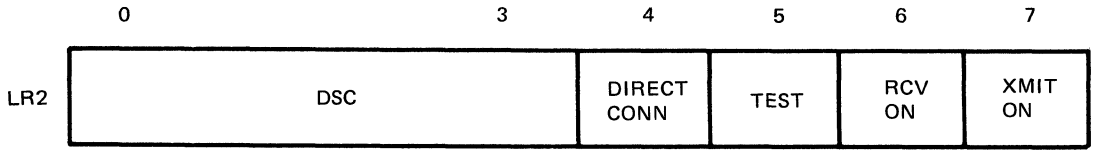


Figure D-7. NBHSA Registers (Synchronous Mode)

LR2 - Control - This LR can be written into by the Processor.



LR2 for the even (receive) channel is the same physical register used by the odd (transmit) channel. It follows that a transmit LR2 control change also changes the receive LR2 control and vice versa.

DSC = Data Set Control. These bits are used by the attached FLAP. On an OUT LR2 instruction, the adapter delivers bits 0-3 to FR2 bits 0-3. Usage of the bits is thus FLAP-dependent.

DIRECT CONN SELECT = This bit is used by the attached FLAP to select whether an External (Modem) clock or an Internal (Processor provided) clock is to be used for the serial interface. On an OUT LR2 instruction, this bit is delivered by the adapter to FR4 bit 1.

TEST = This test sets up the line for loop-back tests by enabling test clock, forcing CTS, and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Data will be transferred.

XMIT ON = Transmitter On for this line:

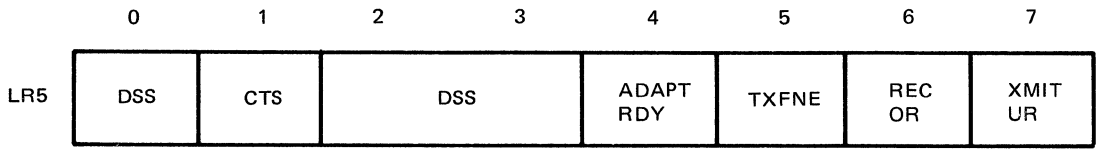
0 = Transmitter Off and inactive
1 = Transmitter On. Data request will be issued for data transfer.

LR3 - Not Used.

LR4 - Transmit Fill/Sync Character. LR4 contains a Transmit Fill/ Sync character code used by the receiver for establishing character synchronization. This register is the same as that used on the transmit side (see transmitter LR4). LR4 may be written only.

LR5 - Line Status. LR5 is used for status purposes by both channels and should only be read.

When an IN LR5 instruction is performed, the adapter copies bits 0 through 3 of FR5 into bits 0 through 3 of LR5 and then delivers the entire contents of LR5 to the Processor (bits 4-7 are provided by the adapter).



DSS = Data Set Status. The definition of these bits is FLAP- dependent.

CTS = Clear To Send. Allows the Transmit USRT to serialize and transmit data delivered to it. Transmit character requests required CTS as well as Transmitter On.

ADAPT RDY = Adapter Ready:

0 = Not ready
1 = On transmit, the internal buffer is not full. On Receive, the internal buffer is not empty.

TXFNE - Transit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

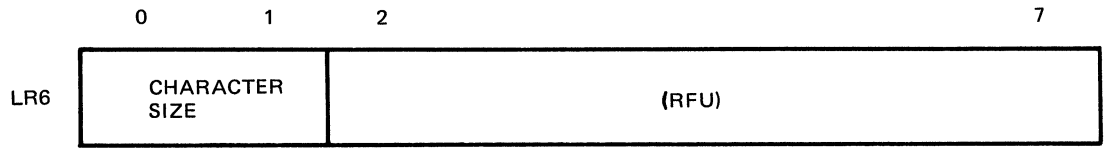
REC OR = Receive Overrun. The NBHSA was not serviced fast enough, and one or more characters have been overwritten and lost:

0 = Normal
1 = Overrun on Receive Channel.

XMIT UR = Transmit Underrun. The NBHSA was not serviced fast enough and the transmit fill character (usually SYN) has been sent in place of a data character. Normally this is not a fatal condition:

0 = Normal
1 = Underrun on Transmit Channel.

LR6 - Character Configuration - This LR can be written into by the Processor.



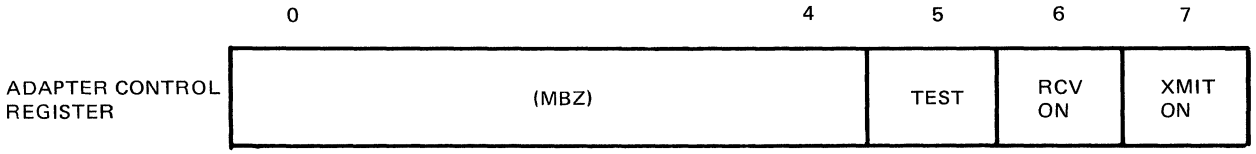
Character size = size of data characters:

00 = 5 bits
10 = 6 bits
01 = 7 bits
11 = 8 bits

The character size or data element size includes the parity bit, if any is used. Parity is checked and generated by the Processor under control of LCT 2 and/or LCT 34.

LR6 for Channel 0 (receive) is the same physical register used by Channel 1 (transmit). It follows that a transmit LR6 character configuration change also changes the receive LR6 character configuration and vice versa.

Adapter Control - This register is written into via the ACTL instruction.



TEST = This test sets up the line for loop-back tests by enabling test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

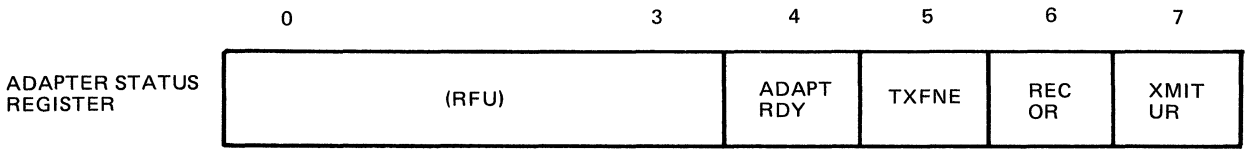
RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Data will be transferred.

XMIT ON = Transmitter On for this line:

0 = Transmitter Off and inactive
1 = Transmitter On. Data request will be issued for data transfer.

Adapter Status - This register is read via the AST instruction.



ADAPT RDY = Adapter Ready:

0 = Not ready
1 = On transmit, the internal buffer is not full. On Receive, the internal buffer is not empty.

TXFNE - Transit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

REC OR = Receive Overrun. The NBHSA was not serviced fast enough, and one or more characters have been overwritten and lost:

0 = Normal
1 = Overrun on Receive Channel.

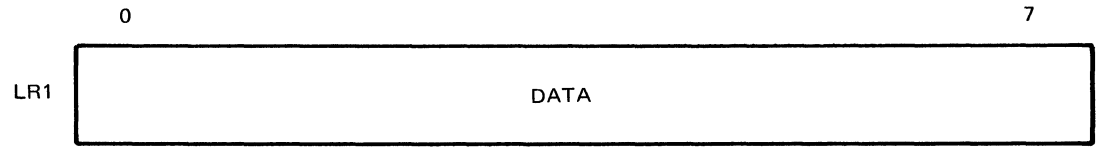
XMIT UR = Transmit Underrun. The NBHSA was not serviced fast enough and the transmit fill character (usually SYN) has been sent in place of a data character. Normally this is not a fatal condition:

0 = Normal
1 = Underrun on Transmit Channel.

TRANSMIT CHANNEL REGISTER DEFINITIONS

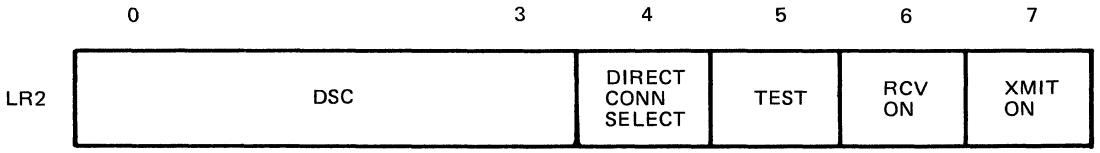
LR0 - See the last two subsections in this appendix.

LR1 - Data - This LR is constructed as a FIFO stack. The stack consists of 128 locations of 8 bits each. These bits are data and are written into by the Processor through LR1. It is the bottom (input) location of the stack which is written into by the Processor. The adapter reads out the stack at the topmost (output) location as characters are transmitted to the line and this pops up the stack.



Bit 7 (Least Significant Bit) is the first bit transmitted. If the character size, including parity, is less than 8 bits, the most significant bits will be Zero and will not be transmitted.

LR2 - Control - This LR can be written into by the Processor.



DSC = Data Set Control. These bits are used by the attached FLAP. On an OUT LR2 instruction, the adapter delivers bits 0-3 to FR2 bits 0-3. Usage of the bits is thus FLAP-dependent.

DIRECT CONN SELECT = This bit is used by the attached FLAP to select whether an External (Modem) clock or an Internal (Processor provided) clock is to be used for the serial interface. On an OUT LR2 instruction, this bit is delivered by the adapter to FR4 bit 1.

TEST = This bit sets up the line for loop-back tests by enabling the test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

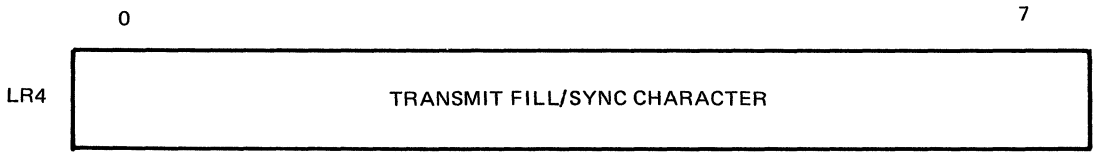
RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Channel Request Interrupts will be sent to the Processor if the receive FIFO has information.

XMIT ON = Transmitter On for this line:

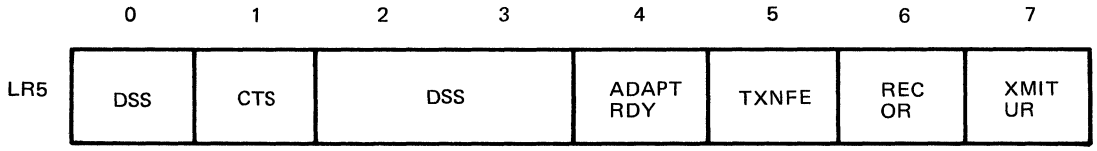
0 = Transmitter Off and inactive
1 = Transmitter On. Channel Request Interrupts will be sent to the Processor if the transmit FIFO has availiable space.

LR4 - Transmit Fill/Sync Character - This LR can be written into by the Processor.



Bit 7 is the least significant bit. In the case of a Transmit Underrun, this character code will be transmitted to the line.

LR5 - Line Status - This LR can be read by the Processor.



DSS = Data Set Status. The definition of these bits is FLAP dependent.

CTS = Clear To Send. Allows the Transmit USRT to serialize and transmit data delivered to it.

ADAPT RDY = Adapter Ready.

0 = Not ready
1 = On transmit, the internal buffer is not full. On Receive, the internal buffer is not empty.

TXFNE = Transit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

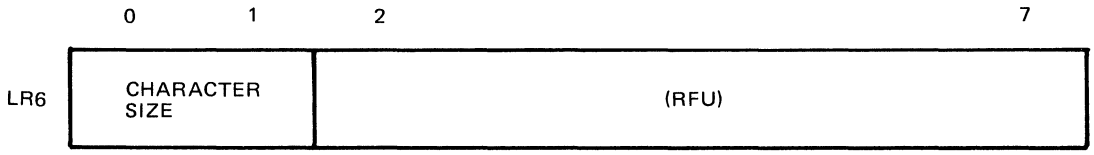
REC OR = Receive Overrun. The NBHSA was not serviced fast enough, and one or more characters have been overwritten and lost:

0 = Normal
1 = Overrun on Receive Channel.

XMIT UR = Transmit Underrun. The NBHSA was not serviced fast enough and the transmit fill character (usually SYN) has been sent in place of a data character. Normally this is not a fatal condition:

0 = Normal
1 = Underrun on Transmit Channel.

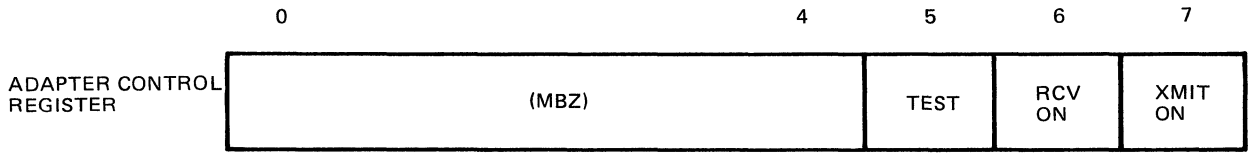
LR6 - Character Configuration - This LR may be written into by the Processor.



Same definition as LR6 receive.

LR6 for the odd (transmit) channel is the same register used by the even (receive) channel. It follows that the receive LR6 character configuration is changed by the transmit LR6 character configuration and vice versa.

Adapter Control - This register is written into via the ACTL instruction.



TEST = This bit sets up loop-back tests by enabling the test clock and looping transmit data back to the receive side (Channel 1 loops back to Channel 0):

0 = Normal operation
1 = Loop-back at test clock frequency.

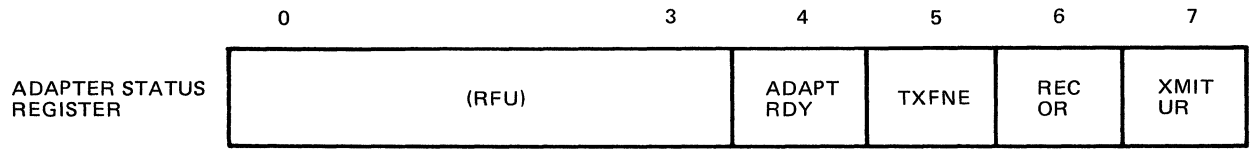
RCV ON = Receiver On for this line:

0 = Receiver Off and inactive
1 = Receiver On. Channel Request Interrupts will be sent to the Processor if the receive FIFO has information.

XMIT ON = Transmitter On for this line:

0 = Transmitter Off and inactive
1 = Transmitter On. Channel Request Interrupts will be sent to the Processor if the transmit FIFO has availiable space.

Adapter Status - This register is read via the AST instruction.



ADAPT RDY = Adapter Ready:

0 = Not ready
1 = On transmit, the internal buffer is not full. On Receive, the internal buffer is not empty.

TXFNE - Transit FIFO Not Empty. When this bit is a Zero, the transmit FIFO buffers have been emptied of all data characters (see subsection entitled "Data Transfer").

REC OR = Receive Overrun. The NBHSA was not serviced fast enough, and one or more characters have been overwritten and lost:

0 = Normal
1 = Overrun on Receive Channel.

XMIT UR = Transmit Underrun. The NBHSA was not serviced fast enough and the transmit fill character (usually SYN) has been sent in place of a data character. Normally this is not a fatal condition:

0 = Normal
1 = Underrun on Transmit Channel.

REGISTER TRANSFER

The NBHSA and the Processor exchange program visible data, configuration, and status via reference to the registers (see subsection entitled "Registers" at the beginning of this appendix.

The LR numbers correspond with the LR numbers of the MLCP instruction set.

DATA TRANSFER

The NBHSA and the Processor exchange parallel data elements on a channel basis. The NBHSA converts parallel data to bit serial data for transmission, and converts bit serial data to parallel data on receive. Each channel contains a parallel data FIFO buffer in addition to a serial input or serial output buffer. The Adapter Ready bit (LR5, bit 4) indicates the state of the channel parallel buffer. The receive buffer is cleared on

execution of the search for Sync instruction or transition of the REC (LR2, bit 6). Both receive and transmit are cleared with OUT LR0.

Prior to setting the Transmitter On bit, the NBHSA will allow characters to be output and stored in the transmit FIFO buffer until Adapter Ready is reset indicating the transmit FIFO is full). The NBHSA will begin transmission with the first character output to the transmit buffer when the Transmitter On bit (LR2, bit 7) is set. Once transmission has begun, the NBHSA will continue until the FIFO is empty, regardless of the Transmitter On bit. (The transmitter will most likely be turned off after the last character is output to the FIFO.)

In order to assure that all characters delivered to the adapter have been transmitted on the line before turning off Request to Send, the CCP should allow for at least three character transmission times after the TXFNE Flag (LR5, bit 5) becomes a Zero (The TXFNE Flag indicates that the transmit FIFO has been emptied.) It is set to One by the Transmitter On bit being turned on. The Flag is reset to Zero with the occurence of the following combination of events; the transmission of characters is talking place, the Transmitter On bit has been turned off, and the transmit FIFO has become empty.

The channel request interrupt (see later subsection in this appendix) provided for each channel is used to coordinate with the Processor when a character may be passed to or from the adapter.

The contents of R are the data element transferred. All data elements by convention are assumed to be right justified for use by the Processor.

DATA ELEMENTS

In Synchronous mode, data elements of 5, 6, 7 or 8 bits are defined for the NBHSA by the character configuration information in LR6. Each channel may load the data element size. However, the transmit and receive channels have the same data element size. Data element size includes all valid bits that are to be transmitted or received including any parity bits.

In HDLC mode, data elements are always 8 bits in size.

REDUNDANCY AND PARITY

Cyclic redundancy using the CCITT polynomial for the Frame Check Sequence may be generated and checked by the NBHSA on each channel. Character and block parity, if used, must be generated and checked using Processor facilities under CCP control. Other CCP polynomials (e.g., the 32-bit Federal Standard CRC) must be generated and checked using Processor facilities (i.e., CCH instruction).

SYNC MODE OPERATION

Receive Synchronization

The receive channel search for synchronization is controlled by the Receiver On bit (LR2, bit 6). Each transition of Receiver On from 0 (Off) to 1 (On) causes the receive channel to enter search for character synchronization.

Character synchronization is provided by the Transmit Fill/Sync Character code (see LR4) over the Transmit (T) and Receive (R) interfaces without Longitudinal Redundancy Check (LRC) and without ACK or NAK performed by the network.

The receive channel, in effect, compares the contents of LR1 (the data) to the contents of LR4 (the synchronization character) at each bit time until the patterns match. No data transfer is initiated on that channel by the NBHSA while this synchronization is taking place. Once character synchronization is established, characters are accumulated for the data element size as input characters.

Synchronization is established with two consecutive synchronization characters. All succeeding characters are transferred to the Processor.

Receive Overrun

If a receive channel accumulates more characters than the FIFO capacity because the Processor has not accepted the data by servicing the CRI, the overflow data is discarded and the Receive Overrun bit (LR5, bit 6) is set. It is reset when the Processor accepts a character.

Transmit Underrun/Transmit Fill

If the Processor does not respond to the channel request interrupt in time to provide sufficient data characters to the FIFO, a Transmit Underrun condition will exist. The status bit Transmit Underrun is set (LR5, bit 6) in the corresponding transmit channel for the notification of the Processor.

When a Transmit Underrun condition occurs, the Transmit Fill character (the contents of LR4) is taken as the next character. If more than one fill character is required for a specific fill sequence (e.g., SYN SYN or DLE SYN Transparent mode in BSC), the CCP in the Processor must provide the appropriate program control for the sequence. The transmit fill is a function provided primarily for idle time SYN fill, but may also assist operation in the Transmit Underrun case.

Treatment of the Transmit Underrun is under control of the CCP, and the NBHSA merely reports the occurrence. When a specific transmit fill sequence is required, it will not significantly affect performance since it is an exceptional condition.

HDLC MODE OPERATION

Receive Startup and Frame Synchronization

When receive operation from the DCE is desired, the CCP will load LR2 or the Adapter Control register and FLAP registers with appropriate information. This connects the DCE to the receive CLA and the receiver begins shifting in serial data bits from the DCE at a rate determined by the DCE receive clock.

During the startup period, an idle link state may be reported. As each bit is shifted in, the receiver examines that bit and the seven preceding bits in search of a Flag sequence (01111110). Once a Flag is detected, the receiver has achieved frame synchronization. Thereafter, the receiver inspects each received octet for a non-Flag or non-Abort sequence. If additional Flags are received, synchronization is maintained. If an Abort (a sequence of seven contiguous Ones) is found, synchronization is lost and must be reestablished.

When, in synchronization, a non-Flag, non-Abort octet is received, it is shifted further into the receiver as a byte.

Receive Data Transfers and Receive Overrun

After the Flag, each data byte received is loaded into the FIFO stack by the adapter. It also loads an associated Data Status byte into the stack. Since the FIFO may in some cases, be initially empty, ADAPT RDY could be OFF (Zero). The act of loading into the FIFO with ADAPT RDY a Zero would then cause the adapter to send a CRI to the Processor. The ADAPT RDY would then also become a One at this time, indicating the presence of information in the FIFO. The adapter continues to load data and data status information into the FIFO, maintaining ADAPT RDY to the proper state.

The receive loop CCP performed by the Processor removes data and data status from the top of the FIFO by reading LR7 as long as ADAPT RDY is a One (and the stack pops up for each read).

Receive overrun will occur if the stack overflows due to failure of the CCP to unload characters soon enough. If overrun in a frame occurs, RO is set in Data Status at the bottom of the stack and the remainder of the frame should be discarded by the CCP software up to and including the REOM. The next frame will be loaded in the usual manner provided the stack overflow has been alleviated. When the RO bit reaches the top of the stack, it results in the CCP becoming aware of the overrun at the point in the message stream at which it occurred. The condition is reported to software which takes further action.

Receive End of Frame

The data shifting, assembling, and FIFO load and unload process continues until a Flag or Abort sequence is detected by the adapter, denoting end of frame.

In the case of an Abort, the adapter sets ABORT and REOM in Data Status at the bottom of the stack. No further loading of the stack occurs until synchronization is reestablished and a new frame begins to be received. It will then be loaded into the stack in the usual manner. When the Abort bit reaches the top of the stack it results in the CCP becoming aware of the Abort at the point in the message stream at which it occurred. The condition is reported to software which takes further action.

In the case of a normal frame termination by a Flag sequence, the adapter sets REOM in Data Status at the bottom of the stack. It also sets the result of the FCS residue check into FCSE. The last character of the frame is also loaded. If another frame follows after the Flag, the adapter will proceed to load this into the bottom of the FIFO stack as previously described. When the REOM bit reaches the top of the stack, it results in the CCP becoming aware of the end of frame at the proper point. The CCP tests the FCSE bit, passing this information on to software. It then prepares to receive the next frame.

Receive Idle Link State

The input line is defined to be in the Idle state when a sequence of fifteen or more Ones are received. This event causes the adapter to report RILS into Data Status at the bottom of the stack. (Only one AILS indication will be posted between frames.) Succeeding action is similar to the Abort case except that, since there is no more data, the FIFO will become empty, ADAPT RDY will remain Zero, and the CCP will remain in a Wait condition.

Receive Missing Frame State

A Missing Frame State occurs if the overrun situation becomes so severe that entire frames are discarded in the adapter. This situation is detected by software as it performs sequence number checks on received frames.

Minimum Receive CCP Structure

A minimum receive loop CCP is shown below. The intraframe loop, being the most critical, is shown in the most detail (see Note).

NOTE

Any instruction that directly accesses the adapter (i.e., IN 7, OUT 7, BART, BARF, ACTL, or ACTS), has a degrading affect on the ability of the adapter to process bytes through the USRT. At high line speeds (greater than 72K bps) the total loop time should be at least two times greater than the time consumed by the adapter instructions.

```
LOC          OP
 .                                  }
 .                                  }->Start Up
 .                                  }
START    WAIT
 GO      IN (LR7)<-----------+      }
  -      BSF (STATUS)        |      }
  -      ST data       20 ms |      }->Intraframe Loop
  -      BLCT (EOB)          |      }
  -      BART (GO)<----------+      }
  -      B (START)                  }

 EOB     BLBT (EOF)                 }
  -      GNB                        }
  -      BVBT (XXX)                 }->End of Block
  -      BART (GO)                  }
  -      B (START)                  }

 EOF                                }
  -                                 }
  -                                 }
  -                                 }->Normal Interframe
  -                                 }
  -                                 }
  .                                 }
  .                                 }
STATUS                              }
  -                                 }
  -                                 }
  -                                 }
  .                                 }->Fault/Special Handling
  .                                 }  (RA, RO, FCSE, RILS, REOM)
  -                                 }
  -                                 }
  -                                 }
```

Transmit Initialization

Transmit operation is initialized by a CLA Master Clear from the Processor. Bits in LR2 are set to an OFF state effectively disconnecting the DCE interface.

Transmit Startup

When transmit operation to the DCE is desired, the Processor CCP loads LR2 or adapter control and FLAP registers with appropriate information. This connects the DCE to the transmit CLA and the transmitter begins marking the line.

The CCP may then prefill the FIFO to whatever extent desired. When XMIT ON is set and the transmitter senses TSOM in the FIFO, it will transmit a Flag and begin transmitting the contents of the FIFO.

Prefilling of the FIFO should not normally be necessary.

Transmit Data Transfers and Transmit Underrun

Each data byte sent by the Processor is loaded into the FIFO stack input along with a data control byte. Adapter Ready remains True as long as the stack is not full. As characters reach the top of the stack, the adapter continues to unload and transmit characters, popping up the stack as it goes.

The transmit loop CCP performed by the Processor loads the FIFO as long as Adapter Ready is True. The adapter will send a CRI to the Processor whenever Adapter Ready switches from False to True, and the Transmitter On is set.

Transmit Underrun will occur if the stack underflows due to failure of the CCP to load the FIFO soon enough. If underrun occurs, the transmitter sets TU and generates and sends an Abort sequence followed by Flags. It continues to request data from the Processor but does not transmit it to the line. When the TEOM bit is set into the stack, the CCP should read LR5 or Adapter Status; it tests TU and, finding it True, branches to its underrun handling routine. On completion of this routine, the CCP starts the next frame and sets TSOM. The adapter senses this and clears its internally stored underrun condition.

Underrun is a fatal error to the frame from an HDLC procedural aspect.

Transmit End of Frame

The data load and unload of the FIFO, disassembly, serial shifting and CRC calculation continues until a Last Character (and Last Block) condition is detected by the CCP. The CCP delivers the last character (data or CRC). For the last character of data it uses an OUT 7 in which the TEOM bit is set

in the data control byte (LCT 43). When the TEOM bit pops out of the top of the stack, the adapter transmits the character (followed by the CRC it has generated, if any) Transmits Flag, and proceeds with the next frame if available in the FIFO.

Minimum Transmit CCP Structure

A minimum transmit loop CCP is shown below. The intraframe loop, being the most critical, is shown in the most detail (See Note).

NOTE

Any instruction that directly accesses the adapter (i.e., IN 7, OUT 7, BART, BARF, ACTL, or ACTS), has a degrading affect on the ability of the adapter to process bytes through the USRT. At high line speeds (greater than 72K bps) the total loop time should be at least two times greater than the time consumed by the adapter instructions.

```
LOC          OP
 -                                  }-> Start Up
 -                                  }
 -                                  }

START     WAIT <-------------------+
 GO[a]    LD, LCTX                 |
 -        BZF Abort                |
 -        LD Data          19 ms   |  }->Intraframe Loop
 -        OUT LR7                  |  }
 -        BLCT EOB                 |  }
 -        BART GO <----------------+  }

          B START
             .
             .
EOB       BLBT EOF                   }
 -        GNB                        }
 -        BVBF SUSPEND               }->End of Block
 -        BART (GO)                  }
 -        B START                    }

             .                       }
             .                       }
             .                       }-> Interframe
             .                       }
             .                       }

             .                       }
             .                       }-> Fault Routine
```

[a]The utilization of LCTX provides for a software capability to abort a frame.

STATUS

The status for channels is provided in LR5, Adapter Status and in Data Status. Status includes Data Set Status, Closing Flag, Abort, Receive Overrun, Transmit Underrun, Receive FCS Error, Receive Idle Link and Adapter Ready.

DATA CLOCK

In most applications for the NBHSA, the data clock is provided through the DTE/DCE interface from the DCE. This is called the external clock. Each channel has independent clock control such that it may transfer data at a different rate than the other channel, as determined by the DCE.

In the case where a Direct Connect to a terminal is desired (no DCE and no external clock), a Direct Connect capability can be provided. The clock in this case is provided through the NMLCP/ NBHSA interface. The selection between external or internal clocking is made according to the clock source bit in LR2. Direct-Connect mode should be set up only at one end of a direct-connect link.

CHANNEL REQUEST INTERRUPT PRIORITY

Each channel is individually capable of generating a Channel Request Interrupt (CRI) which signals a requirement to the Processor that the channel be serviced. In the NBHSA, all CRIs are caused by a need for service.

Once it has begun servicing a FIFO as a result of a CRI, the CCP may sense the state of Adapter Ready and continues to send or accept characters as long as Adapter Ready is True.

In the receive case, Adapter Ready means that the FIFO has data not yet taken by the Processor.

In the transmit case, Adapter Ready means that the FIFO is not full with data.

CRIs are also controlled on a line basis through the RCV ON and XMIT ON bits in LR2. If RCV ON is reset, the receiver will not generate CRIs. If XMIT ON is reset, the transmitter will not generate CRIs.

The NBHSA signals a need for receive or transmit data service to the Processor as a Low Priority CRI for its adapter location. Within the adapter, the receive channel is given service preference over the transmit channel.

CHANNEL NUMBER ASSIGNMENT

The NBHSA uses the highest order line number of the group associated with its adapter board location as the port for synchronous serial data transmission:

	Adapter Board Location			
	0	1	2	3
Receive Channel	6	14	22	30
Transmit Channel	7	15	23	31

The three lower order line numbers of the group may be used for parallel output to a FLAP (e.g., an Auto Call Unit FLAP).

DIRECT CONNECT CLOCK

A clock for direct connect applications is provided by the Processor. The clock is also used during Loop-Back Test mode.

MODE CONTROL

The NBHSA supports two modes of operation, character-oriented (SYNC) and bit-oriented (HDLC).

A DIP switch is provided on the adapter to specify which mode the NBHSA shall enter, and set up initial conditions for when the adapter is initialized. The adapter is initialized by a Channel Initialize, a Processor Soft Initialize, or a Processor Hard Initialize as described in Appendix A.

In order to support X.21 Call Establishment procedures and for T&V purposes, a means is provided whereby the mode of the NBHSA can be switched under software control.

The NBHSA mode can be switched by executing an OUT instruction to LR0 with a value corresponding to that in the following subsection i.e.,:

A value of (48) causes NBHSA to enter SYNC mode.
A value of (49) causes NBHSA to enter HDLC mode.

It should be noted that switching of the mode by software will affect the contents of LR0 and the adapter's ID. Mode switching automatically turns off both transmitter and receiver but does not otherwise alter the adapter LRs or the FLAP FRs. It is the responsibility of the CCP to reload these in accordance with the new mode.

DEVICE IDENTIFICATION NUMBER

The basic ID number is furnished by the Processor without reference to the presence or type of adapter attached to it.

The Extended Device Identification Number is provided by the NBHSA and its associated FLAP. This number is (N1, N2, N3, and $N4)_{16}$ where N1, N2 is provided by the attached adapter and N3, N4 is provided by the FR0 of the attached FLAP. If there is no FLAP attached, N3, N4 $=(00)_{16}$.

In performing the Input Extended ID order, the Processor reads LR0 and FR0, and assembles their contents into the 16-bit numbers given above. Since the reading of FR0 fetches bits 0-7, and bits 0 and 1 of FR0 are not meaningful in this context, the Processor masks FR0 bits 0 and 1, to 00_2 prior to the assembly and presentation of the extended ID to the Megabus network. The value of N1, N2 in LR0 of the highest order line of the adapter depends upon the mode for which the NBHSA has been configured as follows:

Mode	N1 N2/LR0 Contents
SYNC	(48)
HDLC	(49)

The value of N1, N2 on the parellel output lines is $(01)_{16}$.

If the firmware on the adapter bopard is inoperable, the highest order bit of N1 will be forced to a One, resulting in a value of $(A)_{16}$ or C_{16} for N_1.

Appendix E
FLEXIBLE LINE ADAPTER PACKAGE

INTRODUCTION

This appendix describes the functions and interfaces of the Flexible Line Adapter Package (FLAP) used to connect a particular communication interface to a communications line adapter via the FLAP bus.

The FLAP types supported by this appendix are as follows:

1. DSA-47 (RS-232C/V.24)
2. DSA-46 Direct Connect
3. Current Loop
4. X.21 (DSA-49)
5. Autocall
6. Bell 303
7. V.35
8. MIL-188-114
9. MIL-188C.

In a DPS 6/Level 6 communications configuration, certain functionality exists between the Central Processing Unit (CPU) resident software package and the Data Communications Equipment (DCE). This functionality is provided by a combination of software/firmware/hardware and includes, but is not limited to, the following:

1. Transmit and receive control

2. Configuration control

3. Error detection and reporting

4. Line protocol handling

5. Data set control and status reporting

6. Provision of line drivers and receivers to match the electrical characteristics of the particular communications facility (RS-232C, V.24, etc.).

FLAPs are mounted physically separate from the adapter/controller to which they are connected via the FLAP bus. FLAPs provide the electrical interface circuitry needed to transmit/receive across any of a variety of standard communications interfaces. They consist of a set of FLAP registers (FRs) containing information and status of the FLAP and the communications interface. Included in the FLAP are an integral cable connector to the communication interface and a connection to the backplane.

The adapter/controller is mounted in a standard logic chassis. The FLAPs are mounted in a special module chassis as shown in Figure E-1.

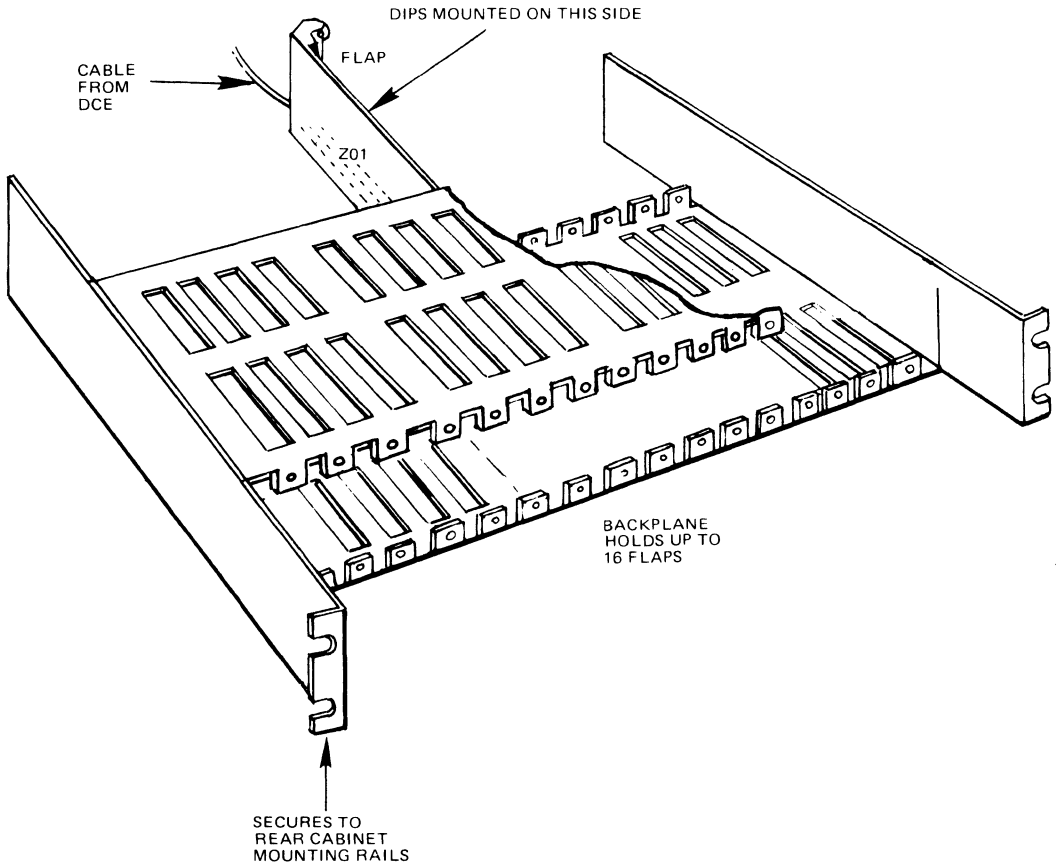


Figure E-1. FLAP Module Chassis

The use of FLAPs avoids the necessity of developing a unique adapter for each processor/interface combination, i.e., a single synchronous/asynchronous adapter can be used with an DSA-47(RS-232C), MIL-188C FLAP, or DSA-46 Direct-Connect FLAP as line electrical characteristics dictate.

FLAP-COMMON EQUIPMENT

The FLAP functionality defined herein applies to all FLAPs. FLAP-unique functionality is covered in the next subsection. FLAPs provide the following:

1. An interface to the FLAP bus. Up to four FLAPs can be connected to a FLAP bus.

2. The circuitry (line drivers and receivers, etc.) to interface with the data and control leads of a particular interface, with no tuning required over the operating frequency range.

3. A set of up to eight FLAP line registers which are written and read under the control of the associated Channel Control Program (CCP) and the controller/adapter.

Each FLAP on a bus is independent of other FLAPs on the bus; that is, there is no direct communication among FLAPs.

The configuration setup, control, data input and output, and status sensing functions for the communications path hardware, consisting of adapter/controller, FLAP, and data set, are accomplished via software-controlled loading and reading of LRs located in the adapter/controller and FLAP. In order that a common set of FLAPs can be used with various adapters/controllers, a set of conventions has been established with regard to register usage.

Software and/or firmware, via the IN, OUT, FIN, and FOUT instructions, has the ability to access up to 16 registers (8 in adapter, 8 in FLAP). Due to the need to maintain compatibility with Multiline Communications Processor (MLCP) usage, certain of these registers must be reserved for specific purposes.

The FLAP bus permits addressing up to eight registers on the FLAP. Registers with a software visibility of LR0 through LR7 will be used for line registers located in the adapter/controller. Those with a software visibility of FR0 through FR7 (except for FR3) will be used for registers located in the FLAP. The adapter/controller recognizes FIN and FOUT instructions as FLAP register "reads" and "writes," respectively, and controls the transfer of register contents over the FLAP bus and to/from the CCP.

Certain IN and OUT instructions (e.g., OUT 2, IN 5) will be acted on by the Processor and its associated synchronous/ asynchronous (see Appendix B) adapters in a manner different from a direct byte transfer. In the case of an OUT 2 instruction, the Processor/adapter will load part of the output byte into LR2 and part into FR2 and FR4. In the case of an IN 5 instruction, the Processor/adapter will return the concatenation of part of LR5 and part of FR5. The reason for this splitting is compatibility; it is a requirement that CCPs written to run on an MLCP/SCLA or an MLCP/ACLA combination will run without modification on the Processor, new adapter, and FLAP combination. For example, when an old CCP runs on the new configuration and executes an IN 5 instruction, it will be returned to a byte whose bit positions and meanings are identical to those of the MLCP configuration.

Note that FOUT 2 or FIN 5 will cause the complete contents of FR2 or FR5 to be loaded or delivered, respectively. To cover the future need of being able to load and access the total contents of LR2 and LR5 without any splitting, the new instructions ACTL and AST have been defined for the Processor.

The FLAP bus does not provide a signal to indicate whether an access to a given FR is to be a read or a write access. Instead, all controllers/adapters and FLAPs observe a convention whereby the type of access is implied by the FR being accessed. In order to perform needed functions, firmware in the controller/adapter must perform operations on certain FRs. Thus, some FRs can be used only for specific purposes in all FLAPs.

In general, the following are common to all FLAP types:

FR2 — Up to four bits, beginning with bit 0, are used for data set/FLAP control.

FR4 — Bit 1 is used for setting the FLAP for use of a direct-connect clock.

FR5 — Up to four bits, beginning with bit 0, are used for data set status.

FR5 — Bit 1 is used as a Clear To Send signal by the associated adapter.

FR5 — Bit 6 is used as a Receive Data signal.

FR2 — Bit 7 and FR4 bit 6 are used to provide loop-back capability for T&V purposes over and above any local or remote modem loop-back.

FLAP-UNIQUE FUNCTIONALITY

FLAP-unique functionality manifests itself in the electrical characteristics of the communications interfaces in the number of FRs utilized and the bit assignments of these registers (subject to conventions of the previous subsection). This subsection describes the FLAP-unique functionality on a FLAP-by-FLAP basis.

DSA-47(RS-232C/V.24) FLAP with NRZI Support

The DSA-47(RS-232C/V.24) FLAP is intended to support both U.S. and European applications. Specifically, it will support the majority of those modems identified in Honeywell Standard DSA-47.

Support of these modems and other U.S. and European modems is achieved by the provision of connector switching functionality. It should be noted, however, that not all modems are supported (e.g., the British BPO series). Consequently, a user of this FLAP should verify that the signals supported, either directly or by switching, provide the functionality required by the modem the user wishes to use.

FLAP REGISTERS

FLAP register bit assignments for the DSA-47(RS-232C/V.24) FLAP are shown in Figure E-2 and described below.

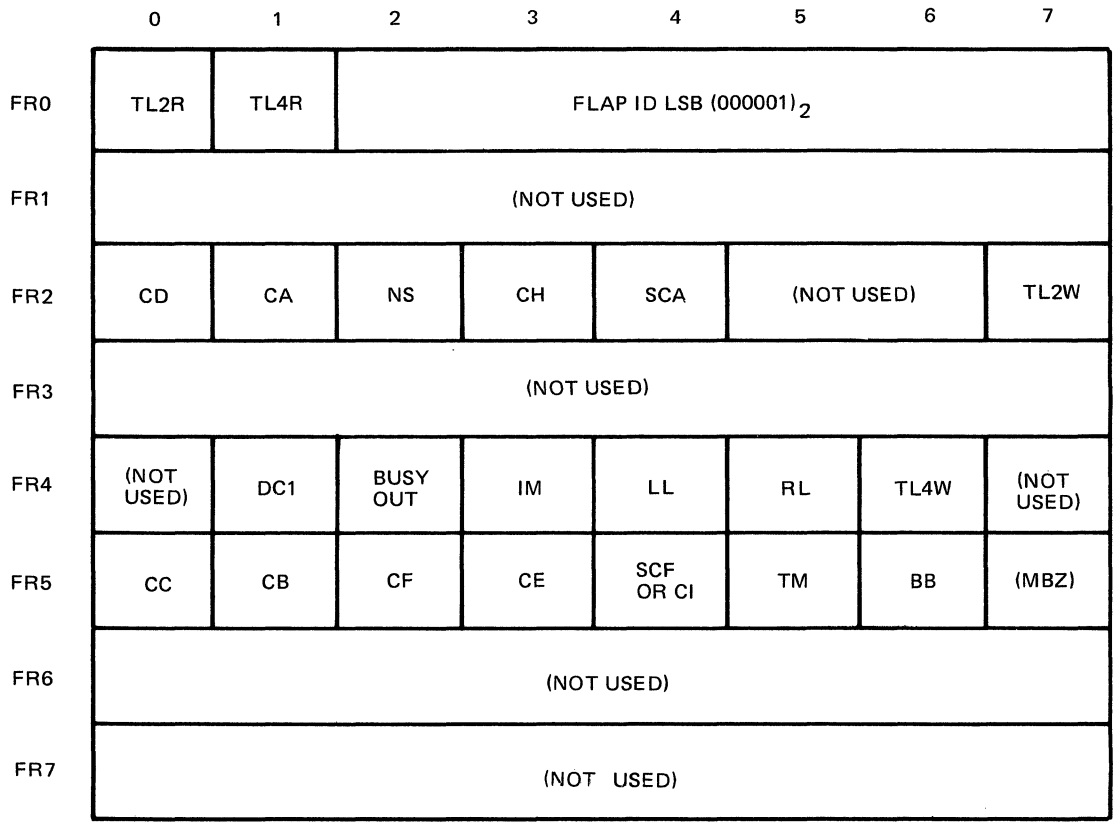


Figure E-2. DSA-47(RS-232C/V.24) FLAP Register Definition

FR0 - Read

Bits 0 and 1, T&D:

Used in conjunction with FR2 bit 7 and FR4 bit 6, these bits provide a loop-back capability for T&D purposes. Test loop 2 (FR2, bit 7) and test loop 4 (FR4, bit 6) are hardwired within the FLAP to FR0, bits 0 and 1, respectively. Thus, the FLAP has a built-in T&D capability in addition to any local and remote modem loop-back.

Additionally, when FR2 bit 7 is set, the transmit data lead is connected to the receive data. This connection is made from the input of the transmit driver to the output of the receive data receiver, with the Transmit line being held in the marking condition.

A full loop-back capability (including all control signals) requires a wrap-around plug at the connector.

Bits 2 through 7:

Least Significant FLAP Identification Bits -$(000001)_2$.

NOTE

When this register is read during an Input Extended ID order, bits 0 and 1 are masked by the adapter to produce a FLAP ID of $(01)_{16}$.

FR2 - Write

Bits used in this register are data set control signals with the exception of TL2W.

Bit 0, CD - Data Terminal Ready
Bit 1, CA - Request To Send
Bit 2, NS - New Sync
Bit 3, CH - Data Rate Selector
Bit 4, SCA - Request To Send On Backward Channel
Bit 7, TL2W - Test Loop 2 (Write). This bit is hard-wired to FR0 bit 0.

FR4 - Write

Bits used in this register control the direct-connect clock and provide a maintenance capability.

Bits 0 and 3 - Must be Zero.

Bit 1, DCl- Direct Connect: This bit controls the direct-connect clock.

When this bit is Zero, the adapter and FLAP are connected as illustrated in Figure E-3A. The DCE source signals DB and DD (Transmitter Signal Element Timing and Receiver Signal Element Timing) are connected to the transmit clock and the receive clock, respectively, on the adapter. The adapter direct-connect clock is not connected.

When this bit is One, the adapter direct-connect clock is connected to the transmit clock in the adapter and to the direct-connect clock lead in the FLAP bus. At the FLAP side, the direct-connect clock lead is connected to interface pin 14 (normally New Sync) and pin 23 (normally Data Signal Rate Selector). An existing direct connect cable external to the FLAP will loop pin 14 to DB and pin 23 to DD (refer to Figure E-3B). FR5 bit 3 (Ring Detector) is set to a logical One.

Complete direct-connect cable connections are shown in Figure E-4.

Bit 2, Busy Out:

This bit is used when supporting the BELL 212 modem

Bit 3, IM - NRZI Mode Bit:

0 = NRZ

1 = NRZI (SDLC support, synchronous modems only)

When this bit is One, the FLAP translates transmitted data from NRZ coding to NRZI and decodes received data from NRZI to NRZ coding.

When this bit is Zero, no coding translation of transmitted and received data is performed.

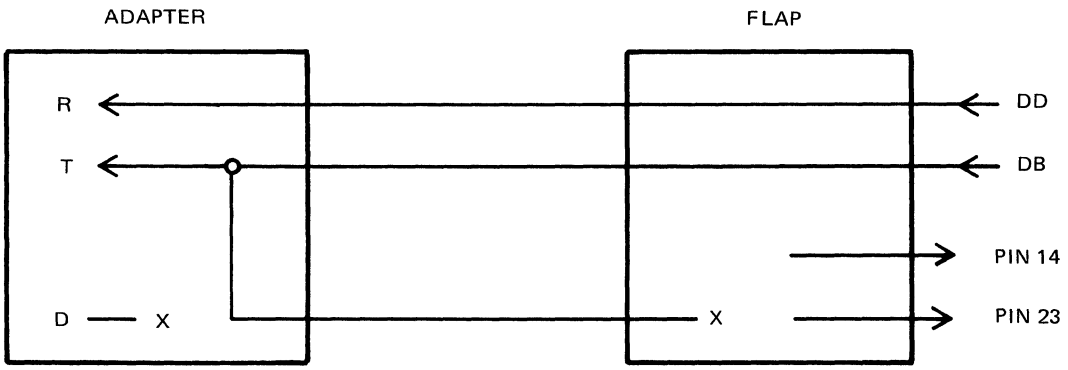
Bits 4 and 5, Local Loop and Remote Loop:

These signals are provided to permit fault isolation to be performed under control of the CCP when connected to a modem which offers this capability. These signals are:

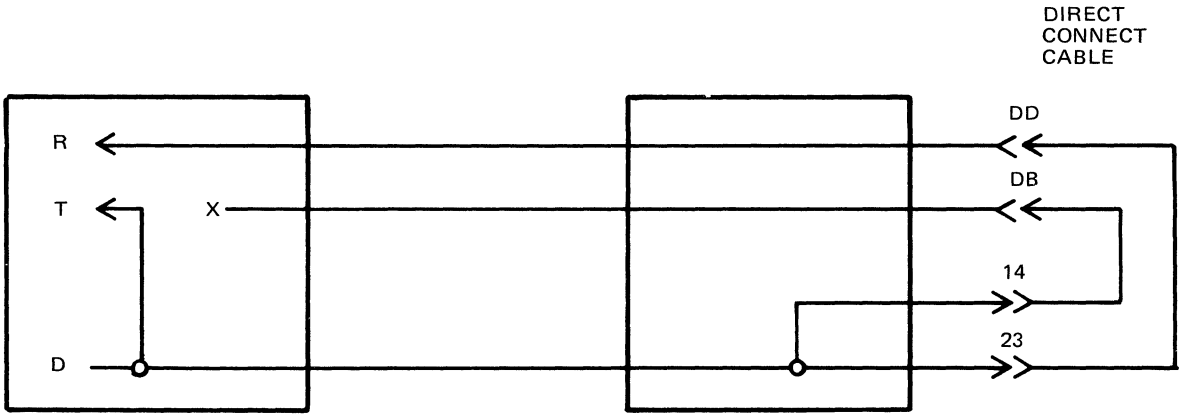
LL (Local Loop-Back)
RL (Remote Loop-Back)
TM (Test Mode).

The first two are signals from the Data Terminal Equipment (DTE) to the Data Communications Equipment (DCE). Setting the LL bit causes the local DCE to feed back its transmitted data to its receive line. Setting the RL bit produces the data crossover at the remote DCE. Note that LL and RL are mutually exclusive. The DCE sets the TM (FR5, bit 5) when either loop-back signal is active. This informs the local DTE that testing can proceed. Also, if the local DTE has initiated a remote loop-back, the remote DCE informs the remote DTE of this occurrence by setting the TM bit to the remote DTE.

Bit 6: TL4W - Test Loop 4(Write). This bit is hard-wired to FR0 bit 1.



A – FR4, BIT 1 = 0



B – FR4, BIT 1 = 1

X = NO CONNECTION
D = DIRECT CONNECT CLOCK

Figure E-3. DSA-47(RS-232C/V.24) Direct-Connect FLAP Timing Configurations

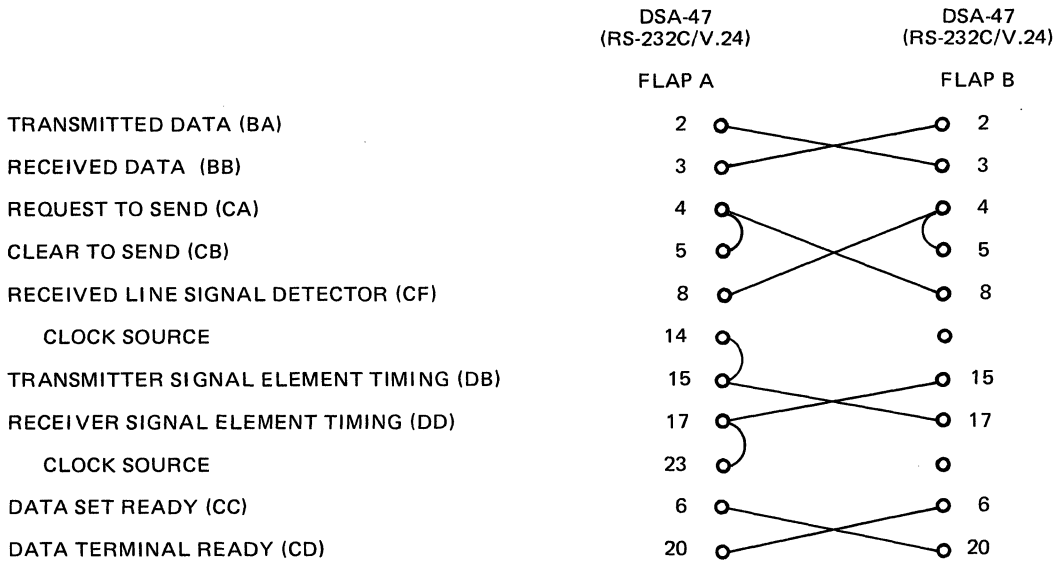


Figure E-4. DSA-47(RS-232C/V.24) FLAP Synchronous Direct-Connect Cable Connections

FR5 - Read

Bits used in this register provide data set status:

Bit 0, CC - Data Set Ready
Bit 1, CB - Clear to Send
Bit 2, CF - Data Carrier Detect
Bit 3, CE - Ring Detector
Bit 4, SCF - Carrier Detector on Backward Channel
Bit 5, TM - Test Mode (refer to FR4 preceding)
Bit 6, BB - Receive Data
Bit 7, MBZ.

DSA-47(RS-232C/V.24) INTERFACE PIN ASSIGNMENTS

The FLAP interface signals and their characteristics are shown in Table E-1.

DSA-47(RS-232C/V.24) FLAP DRIVER AND RECEIVER CHARACTERISTICS

The FLAP drivers and receivers will conform to the electrical characteristics specified in RS-232C.

DSA-47(RS-232C/V.24) CABLE AND CONNECTOR CHARACTERISTICS

The cable from the FLAP to the DCE will not exceed 50 feet in length. It will be furnished with a 25-pin connector (to the DCE) in accordance with ISO-1S2110, having male contacts and a female shell.

FLAP LINE SWITCHING FUNCTIONALITY

There is a nonconformity in pin assignments across the data sets listed in DSA-47. It is therefore necessary to provide on the FLAP a capability of switching or opening certain lines. This makes it possible to avoid certain undesirable conditions, such as connecting to pins prohibited by the DCE.

The required switching functionality is illustrated in Figure E-5. Examples of the necessity for each switch are:

- S1 - Needed to block NS to certain European modems which would interpret NS as Reverse Channel Transmit Data.
- S2 - Certain European modems require NS on pin 11.
- S3 - Needed to block incoming Signal Quality Detector provided by certain Bell modems (208A, 209A) on pin 21.
- S4 - Needed to block incoming Divided Clock Receiver provided by certain Bell modems (201C, 208A, 208B, 209A) on pin 19.
- S5 - Certain bell modems (208A, 208B, 209A) prohibit connection to pin 25.

Table E-1. DSA-47(RS-232C/V.24) FLAP Interchange Circuits Supported

Pin No.	Type	Interchange Circuit	Equivalent CCITT	Circuit Description	
				Name	Direction
1	Common	AA	101	Shield Ground	---
2	Data	BA	103	Transmitted Data	To DCE
3	Data	BB	104	Received Data	From DCE
4	Control	CA	105	Request to Send	To DCE
5	Control	CB	106	Clear to Send	From DCE
6	Control	CC	107	Data Set Ready	From DCE
7	Common	AB	102	Signal Ground/Common Return	---
8	Control	CF	109	Received Line Signal Detector	From DCE
9					
10					
11	Control	SCA	120	Request to Send (Secondary)	To DCE
		NS	136	or New Signal	
			--	or Unused	
12	Control	SCF	122	Received Line Signal Detector (Secondary) or Data Rate Selector (Indication)	From DCE
13					
14	Control	NS	136	New Sync or Unused	To DCE
15	Timing	DB	114	Transmitter Signal Element Timing (DCE)	From DCE
16					
17	Timing	DD	115	Receiver Signal Element Timing	From DCE
18	Control	LL	141	Local Loop-Back or Unused	To DCE
19	Control	SCA	120	Request to Send (Secondary) or Unused	
20	Control	CD	108	Data Terminal Ready	To DCE
21	Control	RL	140	Remote Loop-Back or Unused	To DCW
22	Control	CE	125	Ring Indicator	From DCE
23	Control	CH	111	Data Signal Rate Selector (DTE)	To DCE
24					
25	Control	TM	142	Test Mode, Unused, or Busy Out	From DCE To DCE

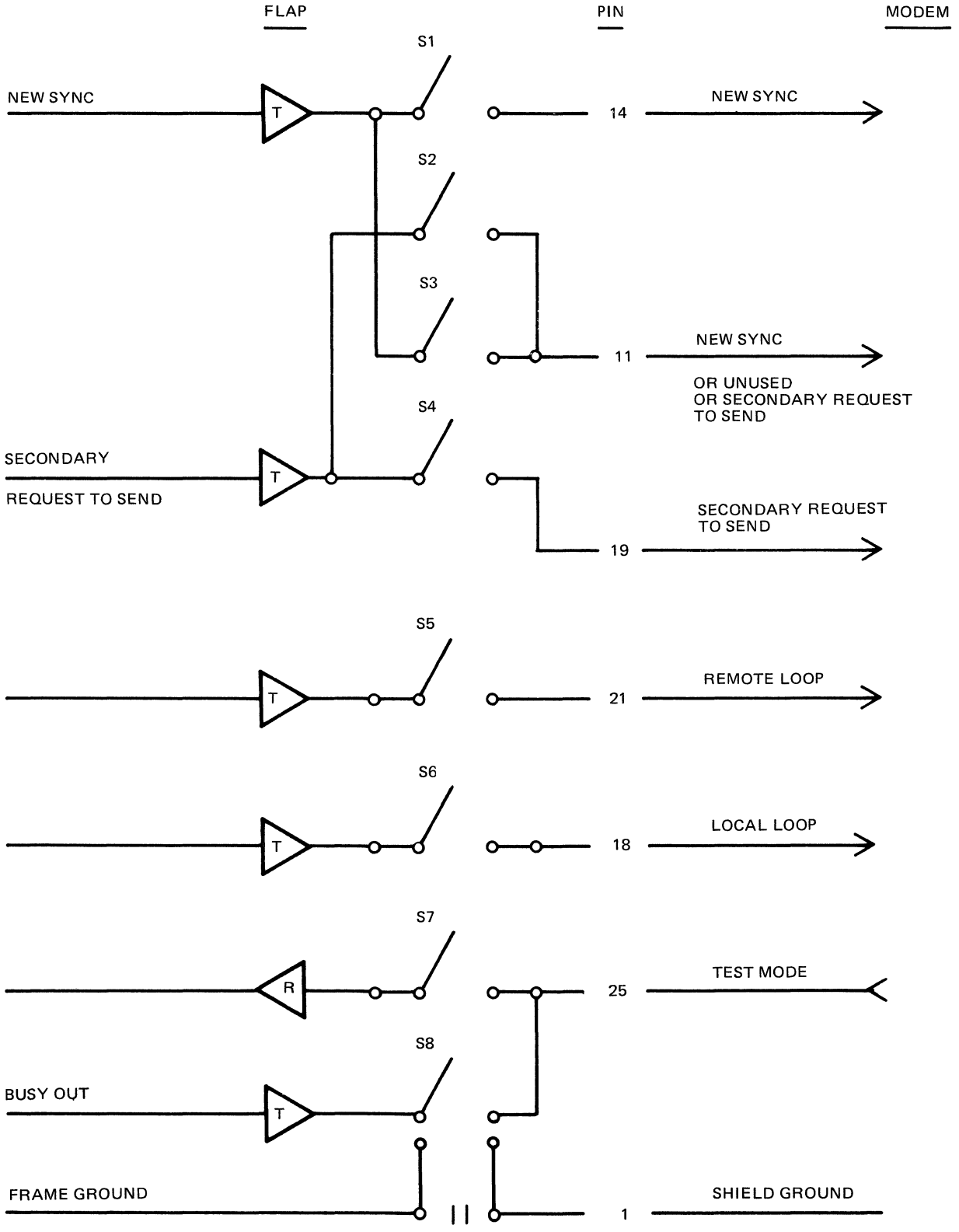


Figure E-5. DSA-47(RS-232C/V.24) FLAP Pin Connectivity Facilities

To satisfy various national requirements, it must be possible to:

- Connect shield ground (pin 1) to frame ground.
- Leave shield ground unconnected.
- Connect shield ground, through a capacitor, to frame ground.

DSA-47(RS-232C/V.24) FLAP Without NRZI Support

For cost reasons, a second version of the DSA-47 (RS-242/V.24) FLAP, described in the previous subsection, may be made available. This FLAP will differ in that support of NRZI mode is not provided.

The software visibility of this FLAP differs in that the FLAP ID from FR0 will be (09) , and bit 3 of FR4 is not used.

DSA-46 Direct Connect FLAP (RS 422A)

The DSA-46 direct connect FLAP provides an asynchronous/ synchronous full duplex communications interface, in accordance with DSA-46 for direct connection to the DTE over distances ranging up to 4000 feet at data rates up to 100K bps.

This interface provides for greater speed and greater distance direct-connect capability than can be achieved by either the RS232C/V.24 or current loop FLAP, and is a lower cost method than using an RS-449 direct-connect interface.

FLAP REGISTERS

Register bit assignments for the DSA-46 direct-connect FLAP are shown in Figure E-6.

In addition to providing for the requirements of DSA-46, register bits are provided for T&V loop-back purposes and other bits are included so that existing software, based upon an assumed RS-232C type interface, will operate successfully over this interface.

FR0 - Read

Bits 0 and 1, TL2R and TL4R:

Used in conjunction with TL2W and TL4W. These bits provide a loop-back capability for T&V purposes. TL2R and TL4R are hard-wired within the FLAP to bits TL2W and TL4W, respectively. Thus, the FLAP has a built-in T&V capability.

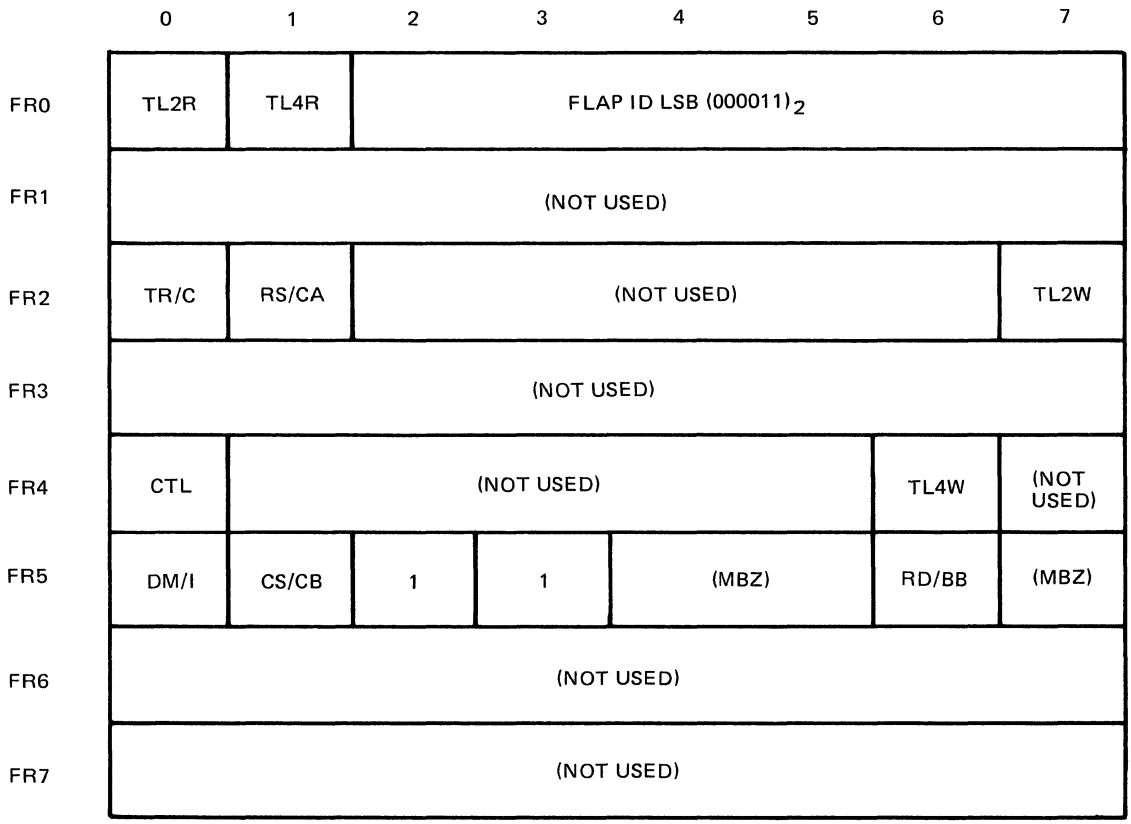


Figure E-6. DSA-46 Direct-Connect FLAP Register Definition

Bits 2 through 7:

Least Significant FLAP Identification Bits -$(000011)_2$. When this register is read as part of an Input Extended ID order, bits 0 and 1 are masked by the adapter to produce a FLAP ID of $(03)_{16}$.

FR2 - Write

Bit 0, TR/C - Control. See DSA-46.

Bit 1, RS/CA - Request to Send:

This bit is hard wired in the FLAP to FR5 bit 1 (CS/CB). The bit and connection are provided for the benefit of existing software which assumes an RS-232C type interface.

Bit 7, TL2W:

This bit is hard-wired to FR0 bit 0 (TL2W).

Additionally, when TL2W is set, transmit data is looped back to receive data. This connection is made from the input side of the transmit driver to the output side of the receive data receiver. The transmit line is held in a marking condition.

NOTE

A full loop-back capability (including all control signals) requires a wrap-around plug at the connector.

FR4 - Write

Bit 0, CTL:

This bit is used to control whether the FLAP makes use of the control signal interface, C and I signals, of DSA-46, or only the asynchronous subset of DSA-46. When this bit is One, bit 0 of FR5 (DM/I) is controlled by the interface signal line I; when this bit is Zero, the DM/I bit of FR5 directly follows the state of the TR/C bit in FR2. This is shown in Figure E-7.

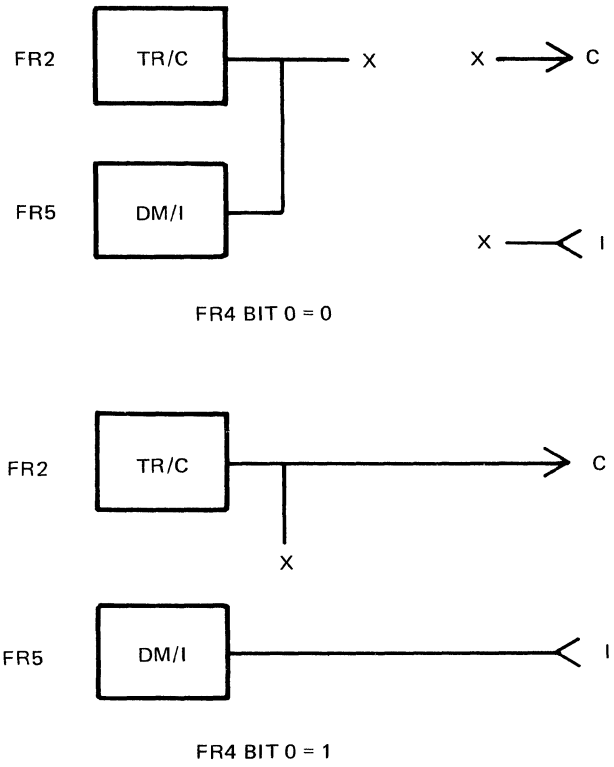


Figure E-7. Control Signal Interface Configurations

FR5 - Read

Bit 0, DM/I - Indicator. See DSA-46.

Bit 1, CS/CB:

This bit is hard-wired from FR2 bit 1 (RS/CA) for the benefit of existing software which assumes an RS-232C type interface.

Bits 2 and 3:

These bits are hard-wired to Ones.

Bit 4, 5 and 7: MBZ.

Bit 6, RD/BB:

This bit indicates the current state of the RD signal line from the interface.

DSA-46 DIRECT-CONNECT FLAP INTERFACE PIN ASSIGNMENTS

The DSA-46 direct-connect FLAP interface characteristics are shown in Table E-2.

Table E-2. DSA-46 FLAP Interchange Circuits[a]

Pin Number	Circuit Description			
	Type	Mnemonic	Name	Direction
1	Common	---	Shield	---
8	Common	SG	Signal Ground or Common Return	---
2-9	Data	SD	Send Data	Outgoing
3-10	Control	C	Control	Outgoing
4-11	Data	RD	Receive Data	Incoming
5-12	Control	I	Indicator	Incoming
6-13	Timing	RT	Receive Timing	Incoming
7-14	Timing	TT	Terminal Timing	Outgoing
15	Not Used			

[a]15-pin male D-shell.

DSA-46 DIRECT-CONNECT FLAP DRIVER AND RECEIVER CHARACTERISTICS

The FLAP drivers and receivers will conform to the electrical characteristics specified in RS-422A.

DSA-46 DIRECT-CONNECT FLAP CABLE AND CONNECTOR CHARACTERISTICS

The connector used on the FLAP is in accordance with ISO-4903, a 15-pin plug connector with male pins and a female shell.

DSA-46 DIRECT-CONNECT FLAP INTERFACE TERMINITIONS, FAIL SAFING, FILTERS, AND GROUNDING OPTIONS

Filters: All of the incoming signal lines (RD,I,RT) will be provided with filters suitable for operation with data speeds of up to 100K bps.

Fail-Safe: The I signal line will be provided with a fail-safe network which produces an OFF condition. The RD signal line is provided with the option of having either a fail-safe or no fail-safe network. The fail-safe on RD, when used, will provide fail-safe to a SPACE condition, but fail-safe to a MARK condition can be provided as an alternative by change of a jumper.

Termination: The RT line will be provided with the termination specified in DSA-46. By means of a DIP switch on the FLAP, the RD line may be set up either with or without a termination. It is intended that when the FLAP is used for the lower speed type asynchronous connection (C,I,RT,TT not used), the termination shall not be used on RD. When the FLAP is used for the higher speed type synchronous connection (all signal lines used), RD may be terminated. This scheme allows for the maximum cable distance, consistent with adequate fail-safing, over the full range of speeds that the FLAP accommodates. DSA-46 provides guidance in this area.

Grounding: Provision is made on the FLAP for connecting the SHIELD (pin 1), either directly or through a component (resistor or capacitor), to the Level 6 frame ground. Normally, the direct connection is provided.

Provision is also made on the FLAP for connecting the SIGNAL GROUND (pin 8), either directly or through a 100-ohm, 1/2-watt resistor, to the Level 6 frame ground. Normally, the direct connection is provided.

EFFECT OF SYNCHRONOUS ADAPTER DIRECT-CONNECT BIT

Synchronous adapters have a clock source bit in LR2 bit 4. On an OUT LR2 instruction this bit is loaded with a new value and is copied into FR4 bit 1 of the FLAP.

The DSA-46 Direct-Connect FLAP makes no use of FR4 bit 1, but the adapter uses LR2 bit 4 to select the source of the transmit clock. As shown in Figure E-8, when clock source LR2 bit 4 is One, the adapter direct-connect clock is used for the transmit clock; when LR2 bit 4 is Zero, the receive clock (from the other DTE) is used as the timing source for the transmit clock.

Current Loop FLAP

Each current loop FLAP provides an interface for one full-duplex (4-wire) asynchronous line. The line can operate up to 19.2K bps and at a distance of up to 1000 feet. It is designed to operate with Model 33/35 teleprinters, VIP7200 and VIP7800 series terminals, and PRU1003 and PRU1005 printers.

FLAP REGISTERS

FLAP register bit configurations are shown in Figure E-9.

This FLAP operates in conjunction with the NMLC and synchronous/ asynchronous adapter (NSAA) to provide a current loop interface. Bit configurations have been chosen to make the current loop FLAP/ adapter/Processor present a compatible interface to the existing driver and to maintain diagnostic uniformity (TL2, TL4).

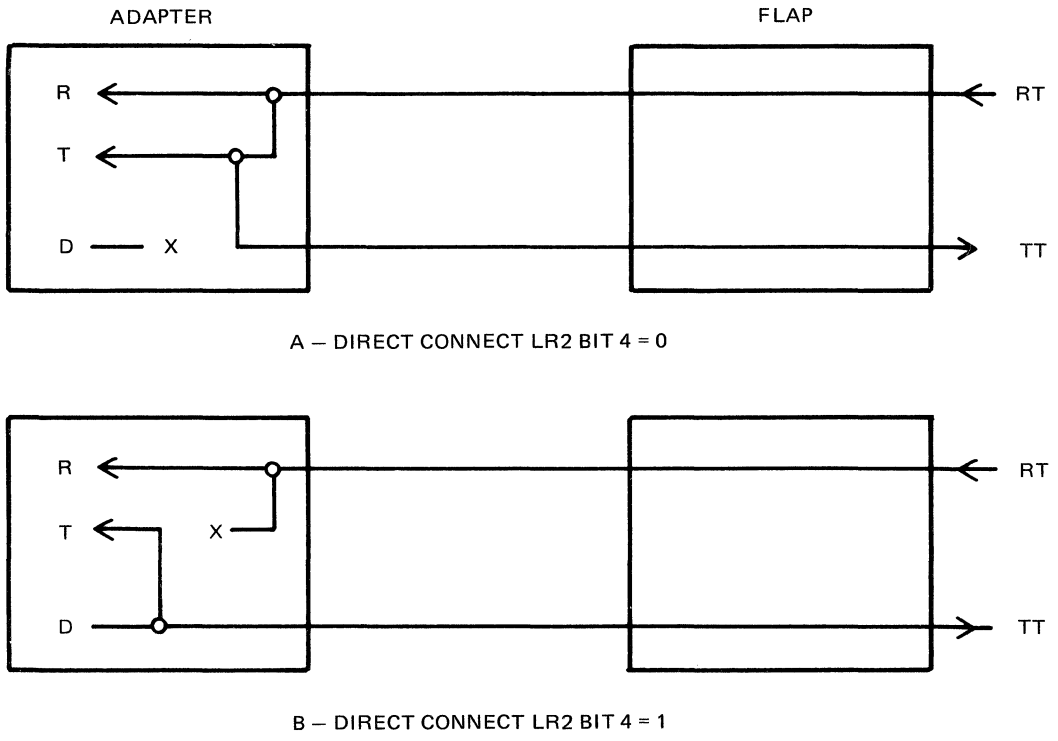


Figure E-8. DSA-46 Direct-Connect FLAP Timing Configurations

Bits 0 and 1, TL2R and TL4R:

Used in conjunction with FR2 bit 7 and FR4 bit 6, these bits provide a loop-back capability for T&D purposes. Test loop 2 (FR2, bit 7) and test loop 4 (FR4, bit 6) are hard-wired within the FLAP to FR0, bits 0 and 1, respectively. Thus, the FLAP has a limited T&D capability built-in which is in addition to any local and remote modem loop-back.

Additionally, when FR2 bit 7 is set, the transmit data lead is connected to the receive data. This connection is made from the input of the transmit driver to the output of the receive data receiver, with the Transmit line being held in the marking condition.

Bits 2 through 7:

Least Significant FLAP Identification Bits -$(000100)_2$. (The FLAP Extended ID will be $(04)_{16}$.)

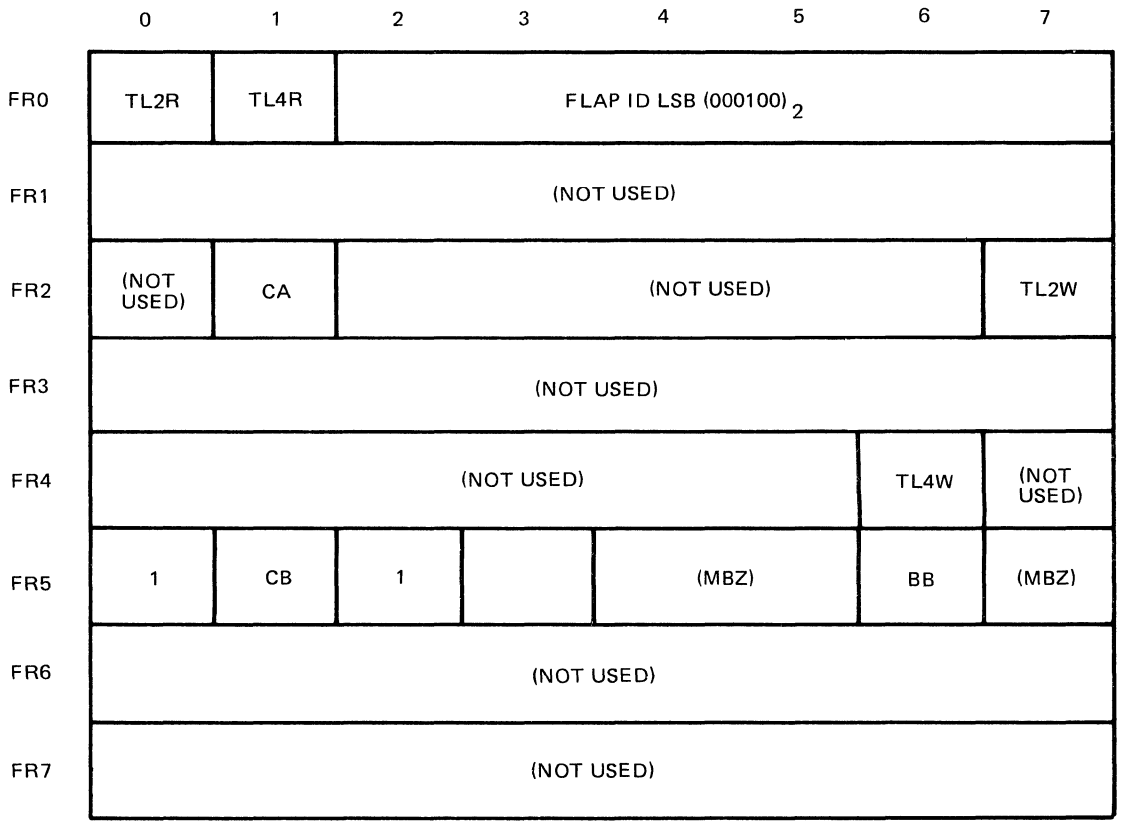


Figure E-9. Current Loop FLAP Register Definition

FR2 - Write

Bit 1, CA - Request to Send:

Set by the CCP to indicate that it is ready to send data. This bit is wired in the FLAP to FR5 bit 1 - Clear to Send.

Bit 7, TL2W - Test Loop 2 (Write):

This bit is hard-wired to FR0-bit 0.

FR4 - Write

Bit 6, TL4W - Test Loop 4 (Write):

This bit is hard-wired to FR0 bit 1.

FR5 - Read

Bits 0, 2, and 3:

These bits are hard-wired to Ones.

Bit 1, CB - Clear to Send:

This bit is hard-wired in the FLAP from FR2 bit 1 - Request to Send.

Bits 4, 5 and 7, MBZ.

Bit 6, BB - Receive Data:

This bit indicates the current state of the Receive Data signal line from the interface.

CURRENT LOOP FLAP INTERFACE PIN ASSIGNMENTS

The current loop FLAP interface characteristics are shown in Table E-3.

Table E-3. Current Loop FLAP Interchange Circuits

Pin Number	Circuit Description			
	Type	Mnemonic	Name	Direction
3-11	Data	--	Receive Data	Incoming
18-25	Data	--	Transmitted Data	Outgoing

CURRENT LOOP FLAP ELECTRICAL CHARACTERISTICS

The current loop interface will operate at 20 mA ± 10% signal current, with current flow indicating a marking condition and no current flow indicating a spacing condition. The FLAP will provide current sourcing for both the transmit loops and the receive loops.

The DTE will be driven over distances ranging from 30 feet to over 1000 feet at rates of up to 19.2K bps.

DTE Send: In spacing condition, the DTE output will appear as an open set of contacts, or their electrical equivalent, with a resistance greater than one megohm presented between the Received Data and Received Data Return lines. In the marking condition, the DTE output will appear as a closed set of contacts or, their electrical equivalent, with a resistance less than 150 ohms presented between the FLAP's Received Data and Received Data Return lines.

DTE Receive: The DTE will present a resistance of less than 150 ohms between the FLAPs Transmitted Data and Transmitted Data Return lines. The input circuitry will be isolated from ground (> 1.0 megohm).

X.21 (DSA-49) Flap

The X.21 (DSA-49) FLAP provides an interface to an X.21 DCE with rates up to 100K bytes per second using a sync/HDLC adapter such as the NBHSA.

FLAP REGISTERS

FLAP register definitions for the X.21 interface are shown in Figure E-10.

FR0 - Read

Bits 0 and 1, TL2R and TL4R:

Used in conjunction with FR2 bit 7 and FR4 bit 6, these bits provide a loop-back capability for T&D purposes. Test loop 2 (FR,2 bit 7) and test loop 4 (FR4 bit 6) are hard-wired within the FLAP to FR0, bits 0 and 1, respectively. Thus, the FLAP has a built-in T&D capability in addition to any local and remote modem loop-back.

Additionally, when FR2 bit 7 is set, the transmit data lead is connected to the receive data. This connection is made from the input of the transmit driver to the output of the receive data receiver, with the Transmit line being held in the marking condition.

A full loop-back capability independent of the modem requires a wrap-around plug at the connector.

Bits 2 through 7:

Least significant FLAP Identification Bits -$(000101)_2$. (The FLAP-extended ID will be $(05)_{16}$.)

FR2 - Write

Bit 0: C - Control Signal (to DCE).

Bit 1, BSM - Byte Sync Mode:

When this bit is One, the FLAP will utilize the Byte Timing Signal (B) from the DCE to align data characters on the Transmit Data line (T) and changes in state of the Control line (C) as presented to the DCE. (These alignments must be initialized using the SYM bit and a specific procedure as detailed below.)

Bit 2, SYM - Synchronize Mode:

When this bit is One, logic in the FLAP is enabled, which initializes and aligns the FLAP Byte Sync Mode shift registers to the Byte Timing signal (B) from the DCE.

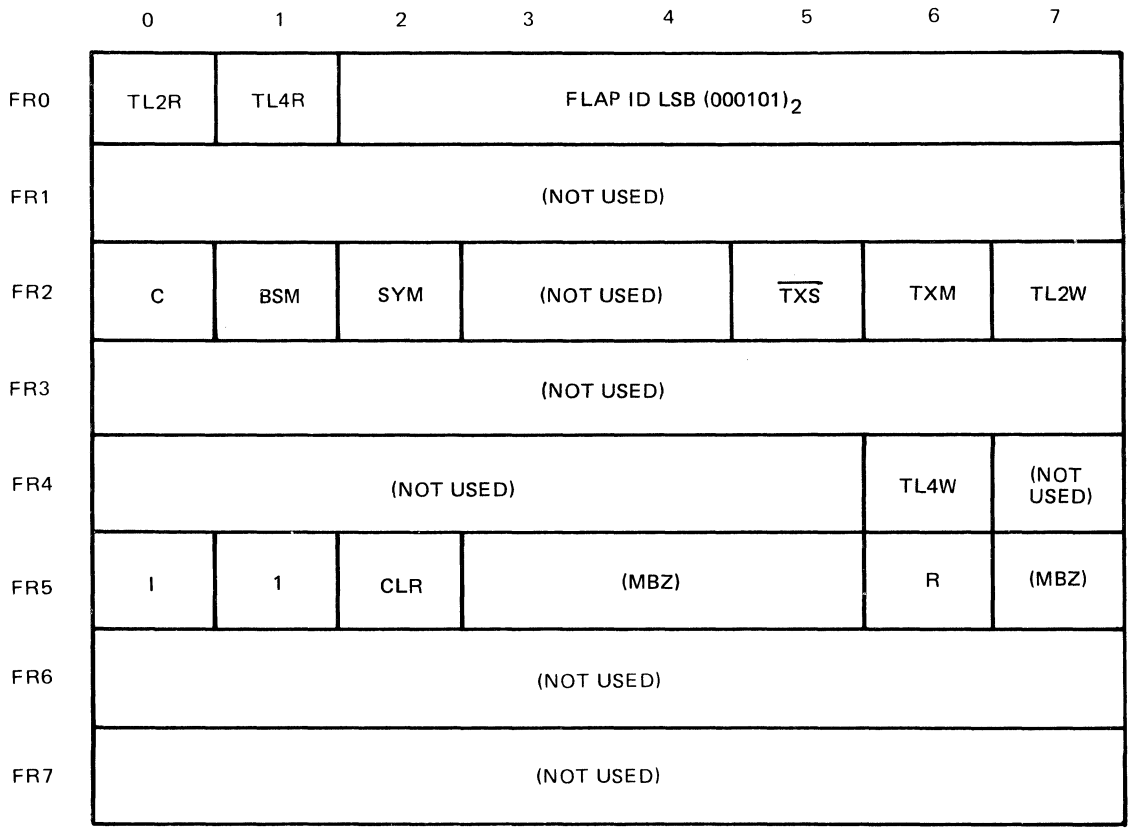


Figure E-10. X.21(DSA-49) FLAP Register Definition

Bit 5, TXS - Transmit Space:

When this bit is Zero, continuous spaces (Zeros) are transmitted to the DCE instead of data from the adapter. If BSM is One, the spaces will commence on a byte boundary. When this bit is One, the spaces are disabled.

Bit 6, TXM - Transmit Mark:

When this bit is One, continuous marks (Ones) are transmitted to the DCE instead of data from the adapter. If BSM is One, the marks commence on a byte boundary. When this bit is Zero, the marks are disabled.

Bit 7, TL2W - Test Loop 2 (Write):

This bit is hard-wired to FR0 bit 0.

FR4 - Write

Bit 6, TL4W - Test Loop 4 (Write):

This bit is hard-wired to FR0 bit 1.

FR5 - Read

Bit 0, I - Indicator Signal (from DCE):

This bit indicates the current state of the Indication signal line (I) from the DCE interface.

Bit 1:

This bit is hard-wired to a One.

Bit 2, CLR - Clear:

This bit is set when the Indicator signal is Zero and 16 consecutive Zeros are received from the DCE over the Received Data line (R). The bit is used to indicate a DCE-initiated clear request or to indicate a DCE clear confirmation in response to a DTE-initiated clear request. To reset this bit, a FIN or FOUT instruction must be performed on FR3.

Bits 3 to 5: MBZ.

Bit 6, R - Received Data signal from DCE:

This bit indicates the current state of the Received Data signal line from the DCE interface (see Note).

Bit 7: MBZ.

NOTE

To prevent false detection of call state changes by possible transients, it is recommended (when not in data transfer state) that a Data Set Scan (Appendix A) be used to detect the initial change of a signal. A second sensing of the signal state can then be performed as confirmation, after a time-out interval.

X.21 (DSA-49) FLAP INTERFACE PIN ASSIGNMENTS

The X.21 interface characteristics are given in Table E-4. For a description of the signals, refer to CCITT X.24.

X.21 (DSA-49) FLAP DRIVER AND RECEIVER CHARACTERISTICS

The X.21 FLAP drivers and receivers will conform to the electrical characteristics specified in CCITT V.11.

Table E-4. X.21 (DSA-49) Interface

Pin Number	CCITT X.21	Function	Direction
1	--	Shield	---
8	G	Signal Ground or Common Return	---
2/9[a]	T	Transmitted Data	To DCE
4/11[a]	R	Received Data	From DCE
3/10[a]	C	Control	To DCE
5/12[a]	I	Indication	From DCE
6/13[a]	S	Signal Element Timing	From DCE
7/14[a]	B	Byte Timing[b]	From DCE

[a] A/B = Signal/Return
[b] Used when BSM (FR2 bit 1) is set to One; application dependent.

Termination, Fail-Safe, and Filter recommendations are provided for as given in DSA-49.

X.21 (DSA-49) FLAP CABLE AND CONNECTOR CHARACTERISTICS

The connector used on the FLAP is in accordance with ISO-4903, a 15-pin plug connector with male pins and a female shell.

X.21 (DSA-49) FLAP SYNCHRONIZATION PROCEDURE

In order to support Byte Synchronization, the FLAP must be brought into synchronization with the Byte Timing Interface signal whenever the adapter is to operate in Character Synchronization mode over the X.21 interface.

There are two general occasions in which the adapter enters Character Synchronization mode on an X.21 interface and must therefore be synchronized if Byte Synchronization is being supported:

1. Before a Call Establishment procedure is begun.

2. After return from Data Transfer mode into Call Clearing procedure.

During the synchronization procedure, the Control signal (C) should be at a Zero state and the Transmit Data line (T) to the DCE should be set to send continuous Zeros by setting TXM and $\overline{\text{TXS}}$ to Zero.

The procedure used to synchronize the FLAP consists of setting up the adapter to transmit a continuous string of Ones to the FLAP and, after this has been established, turning on the SYM bit. The adapter is then set up to transmit a continuous string of Zeros to the FLAP; the first of these results in the correct alignments with the Byte Timing (B) signal being established within the FLAP. The SYM bit may then be rest, $\overline{\text{TXS}}$ returned to a One state, and byte-synchronized operations with the DCE may then be carried out.

In order to ensure that a continuous string of Ones or Zeros is transmitted by the adapter during the above sequence, the Transmit Fill character in LR4 should be set to all Ones or all Zeros (as appropriate), and at least two characters of all Ones or all Zeros should be delivered to LRI when the adapter requests service.

Automatic Call Unit FLAP

The Automatic Call Unit FLAP (ACUF) provides an interface to operate with an automatic calling unit which conforms to RS-366 or RS-366A. The ACUF occupies two NMLC channels.

The ACUF will operate with those models of the Bell 801A and 801C which provide RS-232C interface to the DTE.

FLAP REGISTERS

FLAP register definitions for the ACUF are shown in Figure E-11.

FR0 - Read

Bits 0 and 1, TL2R and TL4R:

Used in conjunction with FR2 bit 7 and FR4 bit 6, these bits provide a loop-back capability for T&D purposes. Test loop 2 (FR2, bit 7) and test loop 4 (FR4, bit 6) are hard-wired within the FLAP to FR0, bits 0 and 1, respectively. Thus, the FLAP has a built-in T&D capability in addition to any local and remote modem loop-back.

Additionally, when FR2 bit 7 is set, NB8, NB4, NB2, NB1 (FR1 bits 4 through 7) are looped back to FR7 bits 4 through 7. Thus, all output data signals can be looped back and read by FIN5 and FIN7 instructions. This loop-back checks all FLAP hardware other than the EIA drivers and receivers.

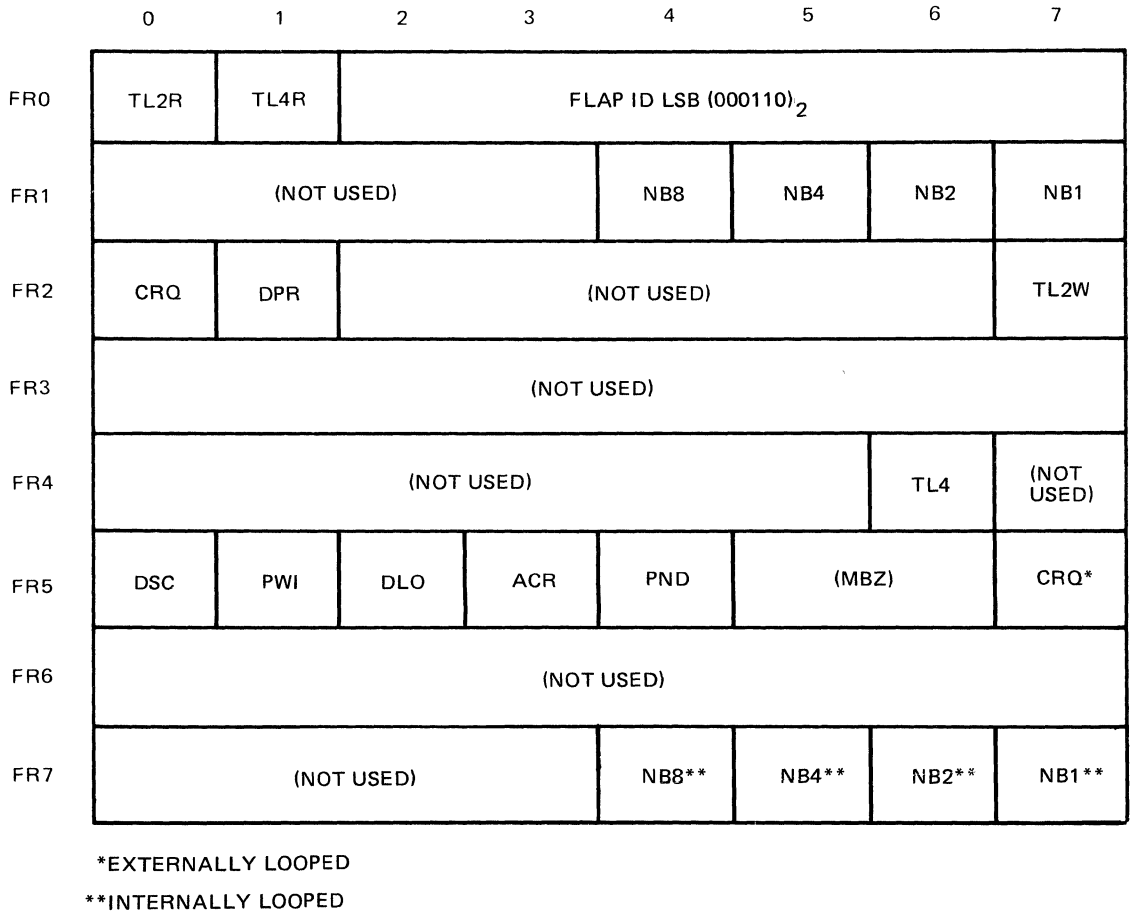


Figure E-11. Automatic Calling Unit FLAP Register Definition

A full loop-back capability (including all control signals) requires a wrap-around plug at the connector.

Bits 2 through 7:

Least Significant FLAP ID Bits - $(000110)_2$.
(The FLAP-extended ID will be $(06)_{16}$.)

FR1 - Write

Bits used in this register are Data Set control signals. Refer to RS-366 for definitions.

Bit 4, NB8 - Digit Signal

Bit 5, NB4 - Digit Signal

Bit 6, NB2 - Digit Signal

Bit 7, NB1 - Digit Signal

FR2 - Output Control

This register may be accessed on both channels of the ACUF via the FOUT instruction and may only be written. Refer to RS-366 for definitions of bits 0 and 1.

Bit 0, CRQ - Call Request to ACU

Bit 1, DPR - Digit Present to ACU

Bit 7, TL2W - Test Loop 2 (Write):

This bit is hard-wired to FR0 bit 0.

FR4 - Write

Bit 6, TL4W - Test Loop 4 (Write):

This bit is hard-wired to FR0 bit 1.

FR5 - Status Read

This register may be accessed on both channels of the ACUF via the FIN instruction and may only be read. Refer to RS-366 for definitions.

Bit 0, DSC - Distant Station Connected from ACU

Bit 1, PWI - Power Indicator from ACU

Bit 2, DLO - Data Line Occupied from ACU

Bit 3, ACR - Abandon Call and Retry from ACU

Bit 4, PND - Present Next Digit from ACU:

During an attempt to call, a Data Set Scan should be used to check this bit and generate CCP startup transitions.

Bits 5 and 6: MBZ

Bit 7, CRQ* - Call Request Signal externally looped back:

This bit is only meaningful when a wrap-around plug is used at the connector.

The complete loop-back under these conditions is shown in Figure E-12.

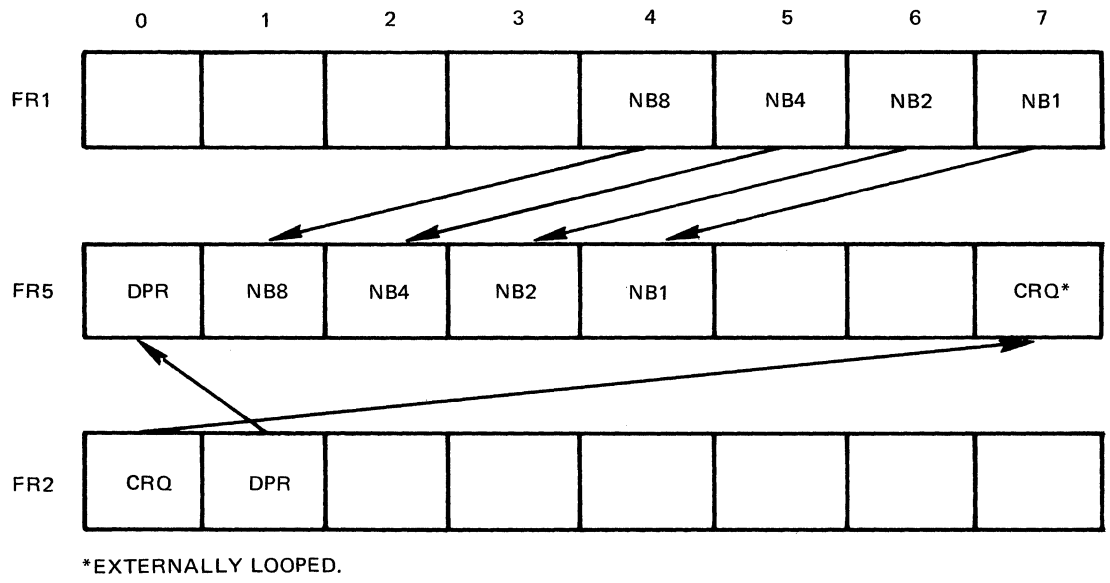


Figure E-12. Connector Wraparound Loop-Back

FR7 - Bits 4 Through 7

These bits contain the contents of FR1 bits 4 through 7. Looping is as illustrated in Figure E-13.

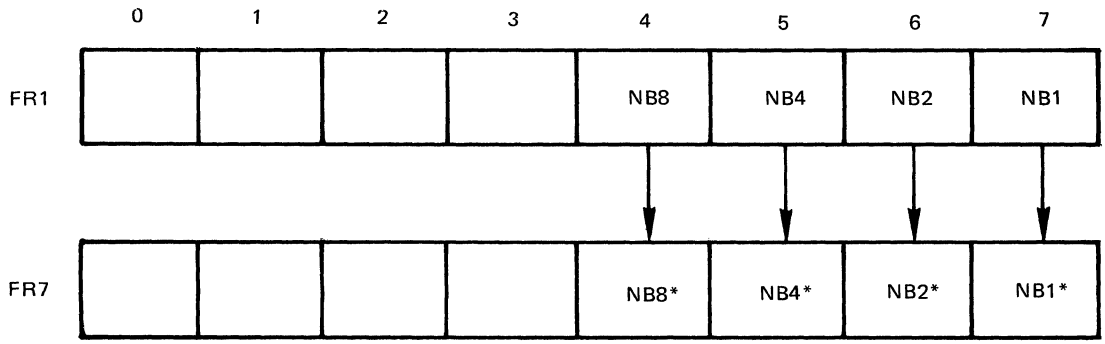


Figure E-13. Loop-Back of Digit Signal Bits

These bits can be read via a FIN7 instruction.

ACUF INTERFACE PIN ASSIGNMENTS

The automatic call unit FLAP interface characteristics are shown in Table E-5. For a description of the signals, refer to RS-366.

Table E-5. Automatic Call Unit FLAP Interface[a]

Pin No.	Inter-change Circuit	CCITT	Description	Signal Ground/ Common Return	Digit	Control	
					To ACE	To ACE	From ACE
7	SG	201	Signal Ground	X			
18	RC		Receive Common	X			
19	SC		Send Common	X			
25	CRQ[a]		Loop-back bit for external loop, delivered to FR5 bit 7				
4	CRQ	202	Call Request			X	
6	PWI	213	Power Indication				X
22	DLO	203	Data Line Occupied				X
13	DSC	204	Distant Station Connected				X

Table E-5 (cont). Automatic Call Unit FLAP Interface

Pin No.	Inter-change Circuit	CCITT	Description	Signal Ground/ Common Return	Digit	Control	
					To ACE	To ACE	From ACE
3	ACR	205	Abandon Call and Retry				X
5	PND	210	Present Next Digit				X
2	DPR	211	Digit Present			X	
14	NB1	206	Low Order Binary Digit		X		
15	NB2	207	Second Order Binary Digit		X		
16	NB4	208	Third Order Binary Digit		X		
17	NB8	209	High Order Binary Digit		X		
1			Shield				

[a]25-pin female D-shell.

ACUF DRIVER AND RECEIVER CHARACTERISTICS

The FLAP drivers and receivers conform to RS-232C. This will enable the FLAP to operate with Bell 801A and 801C units. It will also be possible to operate this FLAP with new RS-366A ACUs. (Interfacing of RS-423A circuits with RS-232C circuits will constrain distances to RS-232C rules.)

ACUF CABLE AND CONNECTOR CHARACTERISTICS

The ACUF will be provided with a cable having the following characteristics:

1. Length of up to 200 feet (50 feet when connecting to RS-366 ACUs)

2. Cable terminated in a 25-pin male connector (see reference 14 for details).

Wideband Data Stations 303-Type FLAP

The 303 FLAP interfaces with a 303-type data set and provides a full-duplex serial link capable of operating at data rates of 50K bps. The FLAP supports the Synchronous mode of operation for the the 303 data set. The Nonsynchronous mode of operation for the 303 data set is not supported.

FLAP REGISTERS

FLAP register bit assignments for the 303 FLAP are shown in Figure E-14.

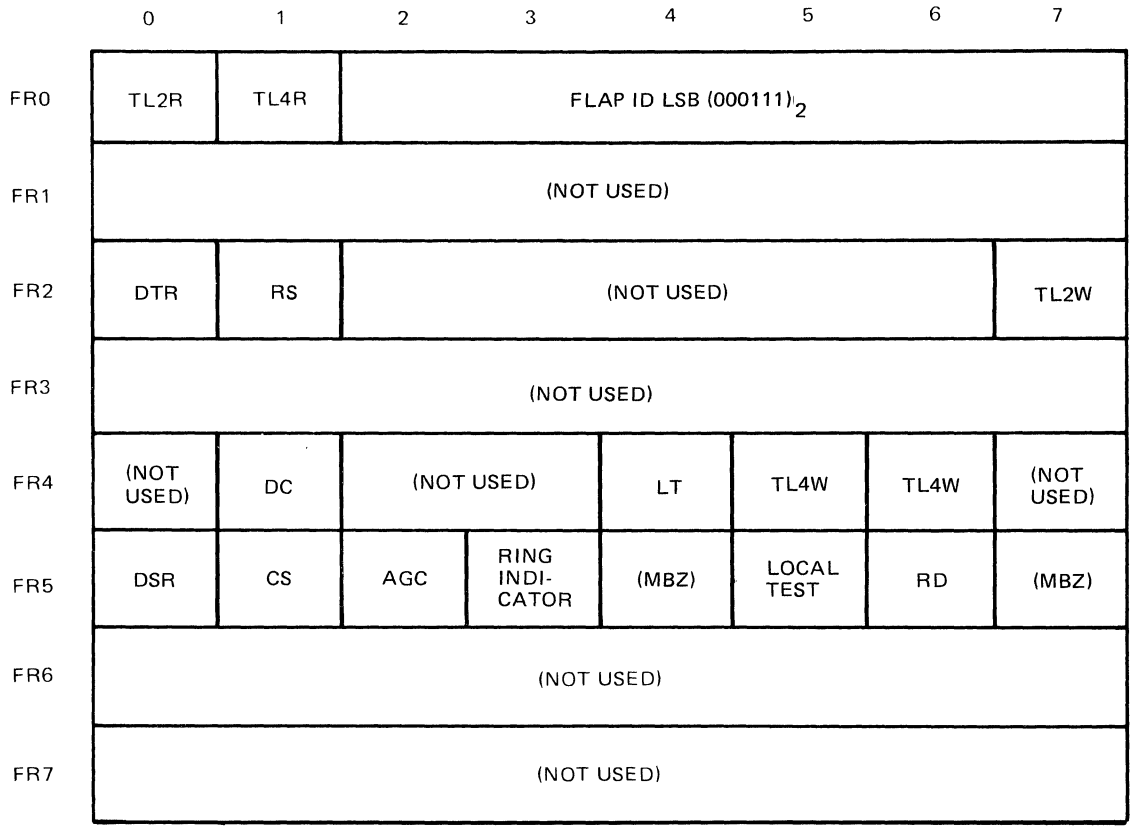


Figure E-14. 303 FLAP Register Definition (Current Mode)

FR0 - Read

Bits 0 and 1, T&D:

Used in conjunction with FR2 bit 7 and FR4 bit 6, these bits provide a loop-back capability for T&D purposes. Test loop 2 (FR2, bit 7) and test loop 4 (FR4, bit 6) are hard-wired within the FLAP to FR0, bits 0 and 1, respectively. Thus, the FLAP has a built-in T&D capability in addition to any local and remote modem loop-back.

Additionally, when FR2 bit 7 is set, the transmit data lead is connected to the received data. This connection is made from the input of the transmit driver to the output of the received data receiver, with the Transmit line being held in the marking condition.

A full loop-back capability (including all control signals) requires a wrap-around plug at the connector.

Bits 2 through 7:

Least Significant FLAP Identification Bits -$(000111)_2$. (The FLAP-extended ID will be $(07)_{16}$.)

FR2 - Write

Bits used in this register are Data Set control signals with the exception of TL2W:

Bit 0, Data Terminal Ready

Bit 1, Request to Send

Bit 7, Test Loop 2 (Write):

This bit is hard-wired to FR0 bit 0.

FR4 - Write

Bits used in this register provide a maintenance capability:

Bit 1, DC - Direct Connect

Bit 4, Local Test:

When this bit is One, the DTE can send to itself through the wideband data set.

Bit 6, TL4 - Test Loop 4 (Write):

This bit is hard-wired to FR0 bit 1.

FR5 - Read

Bit 0, Data Set Ready

Bit 1, Clear to Send

Bit 2, AGC (functionally equivalent to Data Carrier Detect)

Bit 3, Ring Indicator

Bits 4 and 7, MBZ

Bit 5, Local Test:

This bit is hard-wired from FR4 bit 4.

Bit 6, Receive Data

303 FLAP INTERFACE PIN ASSIGNMENTS

The 303 FLAP interface signals and their characteristics are shown in Table E-6.

Table E-6. 303 FLAP Interchange Circuits[a]

Pin No.	Type	Inter-change Circuit	CCITT Equiv-alent	Circuit Description	
				Name	Direction
7	Control	CS	106	Clear to Send	From DCE
5	Control	RS	105	Request to Send	To DCE
1	Data	SD	103	Send Data	To DCE
9	Control	DSR	107	Data Set Ready (Center Conductor)	From DCE
10	Control	RI	125	Ring Indicator (Center Conductor)	From DCE
19	Control	LT	---	Local Test	To DCE
21,22	Timing	SCTE	113	Serial Clock Transmit (External Sync)	To DCE
13	Timing	SCT	114	Serial Clock Transmit (Internal Sync)	From DCE
3	Data	RD	104	Receive Data	From DCE
17	Timing	SCR	115	Serial Clock Receive	From DCE
11	Control	AGC	109	Data Carrier Detect[b] (Center Conductor)	From DCE
12	Control	DTZ	108	Data Terminal Ready (Outer Conductor)	To DCE

[a]25-pin female D Shell.
[b]Functionally equivalent.

303 FLAP DRIVER AND RECEIVER ELECTRICAL CHARACTERISTICS

The Data Terminal Ready and the Ring Indicator circuits comply with EIA Standard RS-232C. All other interface circuits are to be provided on a current switching basis.

303 FLAP CABLE AND CONNECTOR CHARACTERISTICS

A cable of 50 feet maximum length must provide 10 coaxial circuits. The cable is equipped with a 12-pin Burndy MD12MXP-17TC plug and a Burndy M2H50RC-1P2 protective shield.

V.35 FLAP

The V.35 FLAP interfaces with modems which comform to V.35, providing synchronous full-duplex serial operation at data rates of 72,000 bps when operating with V.35 modems. This FLAP can also be used to interface with the Bell Digital Data System at a data rate of 56K bps.

FLAP REGISTERS

V.35 FLAP register bit assignments are shown in Figure E-15.

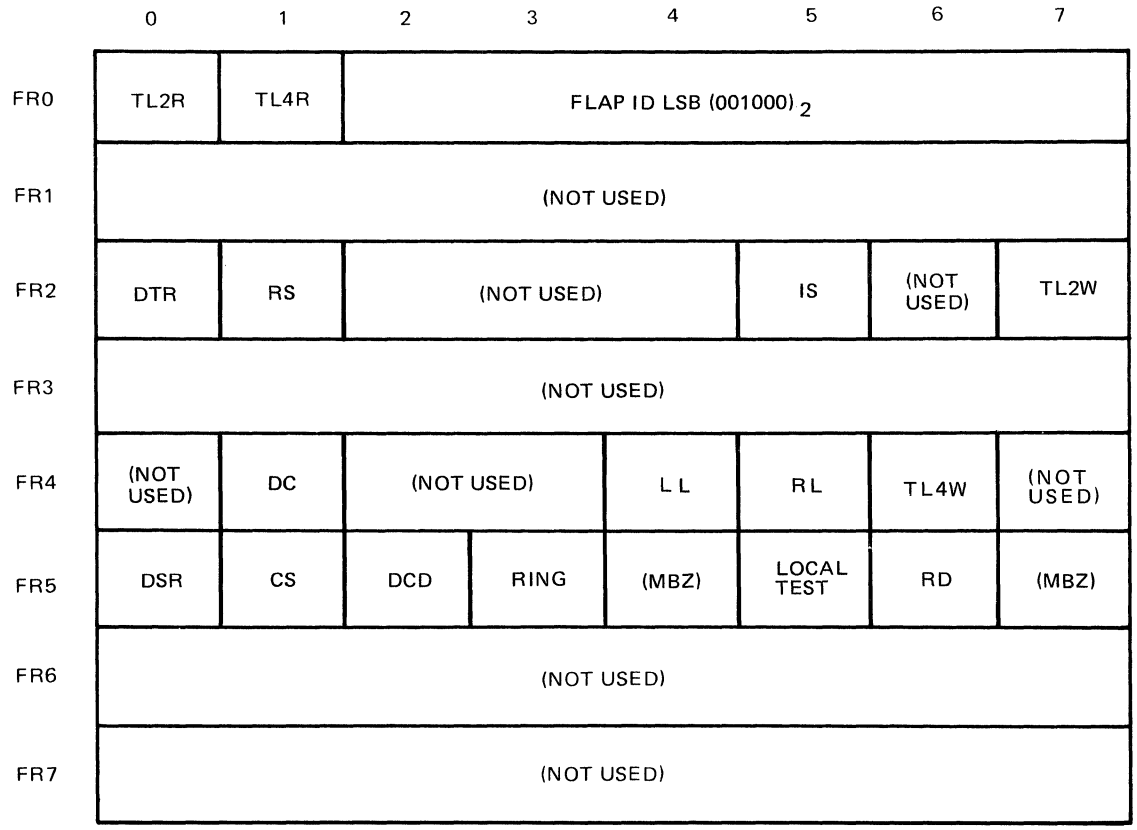


Figure E-15. V.35 FLAP Register Definition (Balanced Line)

FR0 - Read

Bits 0 and 1, T&D:

Used in conjunction with FR2 bit 7 and FR4 bit 6, these bits provide a loop-back capability for T&D purposes. Test loop 2 (FR2, bit 7) and test loop 4 (FR4, bit 6) are hard-wired within the FLAP to FR0, bits 0 and 1, respectively. Thus, the FLAP has a built-in T&D capability in addition to any local and remote modem loop-back.

Additionally, when FR2 bit 7 is set, the transmit data lead is connected to the received data. This connection is made from the input of the transmit driver to the output of the received data receiver, with the Transmit line being held in the marking condition.

A full loop-back capability (including all control signals) requires a wrap-around plug at the connector.

Bits 2 through 7:

Least Significant FLAP Identification Bits -$(001000)_2$. (The FLAP extended ID will be $(08)_{16}$.)

FR2 - Write

Bits used in this register are Data Set control signals except for TL2.

Bit 0, Data Terminal Ready

Bit 1, Request to Send

Bit 5, In Service

Bit 7, Test Loop 2W:

This bit is hard-wired to FR0 bit 0.

FR4 - Write

Bits used in this register provide a maintenance capability.

Bit 1, DC - Direct Connect

Bit 4, LL - Local Loopback

Bit 5, RL - Remote Loopback

Bit 6, TL4 - Test Loop 4W:

This bit is hard-wired to FR0 bit 1.

FR5 - Read

Bit 0, Data Set Ready

Bit 1, Clear to Send

Bit 2, Data Carrier Detect

Bit 3, Ring

Bit 6, Receive Data

Bits 4 and 7, MBZ

Bit 5, Local Test:

This bit is hard-wired from FR4 bit 4.

V.35 FLAP Interface Pin Assignments

The V.35 FLAP interface signals and their characteristics are shown in Table E-7.

Table E-7. V.35 FLAP Interchange Circuits[a]

Pin No.	Type	Inter-change Circuit	CCITT Equiv-alent	Circuit Description	
				Name	Description
1	Control	CA	105	Request to Send	To DCE
3	Control	CB	106	Clear to Send	From DCE
7	Control	CC	107	Data Set Ready	From DCE
5	Control	CF	109	Carrier Detector	From DCE
19	Control	CD	108/2	Data Terminal Ready	To DCE
21	Control	EI	125	Call Indicator	From DCE
24	Test	LL	141	Local Loop-Back	To DCE
25	Test	RL	140	Remote Loop-Back	To DCE
11	Data	BB	104	Received Data A	From DCE
12	Data	BB	104	Received Data B	From DCE
13	Timing	DD	115	Receive Clock A	From DCE
14	Timing	DD	115	Receive Clock B	From DCE
15	Timing	DB	114	Transmit Clock A	From DCE
16	Timing	DB	114	Transmit Clock B	From DCE
10	Data	BA	103	Transmitted Data A	To DCE
9	Data	BA	103	Transmitted Data B	To DCE
17	Timing	DA	113	Transmit Clock A	To DCE
18	Timing	DA	113	Transmit Clock B	To DCE
23	Control	IS		In Service	To DCE

[a]25-pin female D-shell on FLAP.

V.35 FLAP ELECTRICAL CHARACTERISTICS

The electrical characteristics of sets 103, 104, 114, and 115 conform to V.35. All others conform to V.28.

V.35 FLAP CABLE AND CONNECTOR CHARACTERISTICS

The cable from the FLAP to the DCE should be less than 100 feet in length. It is provided with a 34-pin connector in accordance with ISO-2593 at the far end of the interconnecting cable and a 25-pin connector at the FLAP end.

MIL-STD-188-114 FLAP

The MIL-STD-188 series of standards are design standards for military communications systems and include the electrical characteristics of low-level digital interfaces. Change notes dated June 1, 1976 change the electrical characteristics and data signaling sense specified in the original MIL-STD-188C and MIL-STD-188-100 standards to the characteristics specified in the new electrical interface standard MIL-STD-188-114 (see Table E-8). The change in data signaling sense makes it compatible with commercial standards; e.g., EIA RS-232C. This FLAP will utilize the signals and pinning of RS-232C, using MIL-STD-188-144 unbalanced interface circuits on all leads.

FLAP REGISTERS

FLAP register bit assignments for the MIL-STD-188-114 FLAP are as shown in Figure E-16. Definitions of these bits are the same as in the subsection entitled "Flap Registers" to the extent that the bits are provided. Note that the FLAP-extended ID will be $(OA)_{16}$.

MIL-STD-188-114 FLAP INTERFACE PIN ASSIGNMENTS

The MIL-STD-188-114 FLAP interface circuits and their pin assignments are as shown in Table E-8. Detailed functional descriptions of each interface circuit are provided in EIA standard RS-232C.

MIL-STD-188-114 FLAP DRIVER AND RECEIVER CHARACTERISTICS

All data and timing interface drivers and receivers meet the electrical requirements of MIL-STD-188-144 paragraph 5 and the Appendix, paragraph 10. Default wave shaping is based on 200 feet of cable.

Table E-8. MIL-STD-188-114 FLAP Interchange Circuits

Pin No.	Type	Inter-change Circuit	CCITT Equiv-alent	Circuit Description	
				Name	Direction
1	Common	AA	101	Shield Ground	---
2	Data	BA	103	Transmitted Data	To DCE
3	Data	BB	104	Received Data	From DCE
4	Control	CA	105	Request to Send	To DCE
5	Control	CB	106	Clear to Send	From DCE
6	Control	CC	107	Data Set Ready	From DCE
7	Common	AB	102	Signal Ground/Common Return	---
8	Control	CF	109	Received Line Signal Detector	From DCE
9					
10					
11	Control	SCA NS	120 136 --	Request to Send (Secondary) Or unused	To DCE
12	Control	SCF	122	Received Line Signal (Secondary) or Data Rate Selector (Indication)	From DCE
13					
14	Control	NS	136	New Sync or Unused	To DCE
15	Timing	DB	114	Transmitter Signal Element Timing (DCE)	From DCE
16					
17	Timing	DD	115	Receiver Signal Element Timing	From DCE
18	Control	LL	141	Local Loop-Back or Unused	To DCE
19	Control	SCA	120	Request to Send (Secondary) or Unused	
20	Control	CD	108	Data Terminal Ready	To DCE
21	Control	RL	140	Remote Loop-Back or Unused	To DCE
22	Control	CE	125	Ring Indicator	From DCE
23	Control	CH	111	Data Signal Rate Selector (DTE)	To DCE
24					
25	Control	TM	142	Test Mode, Unused, or Busy Out	From DCE To DCE

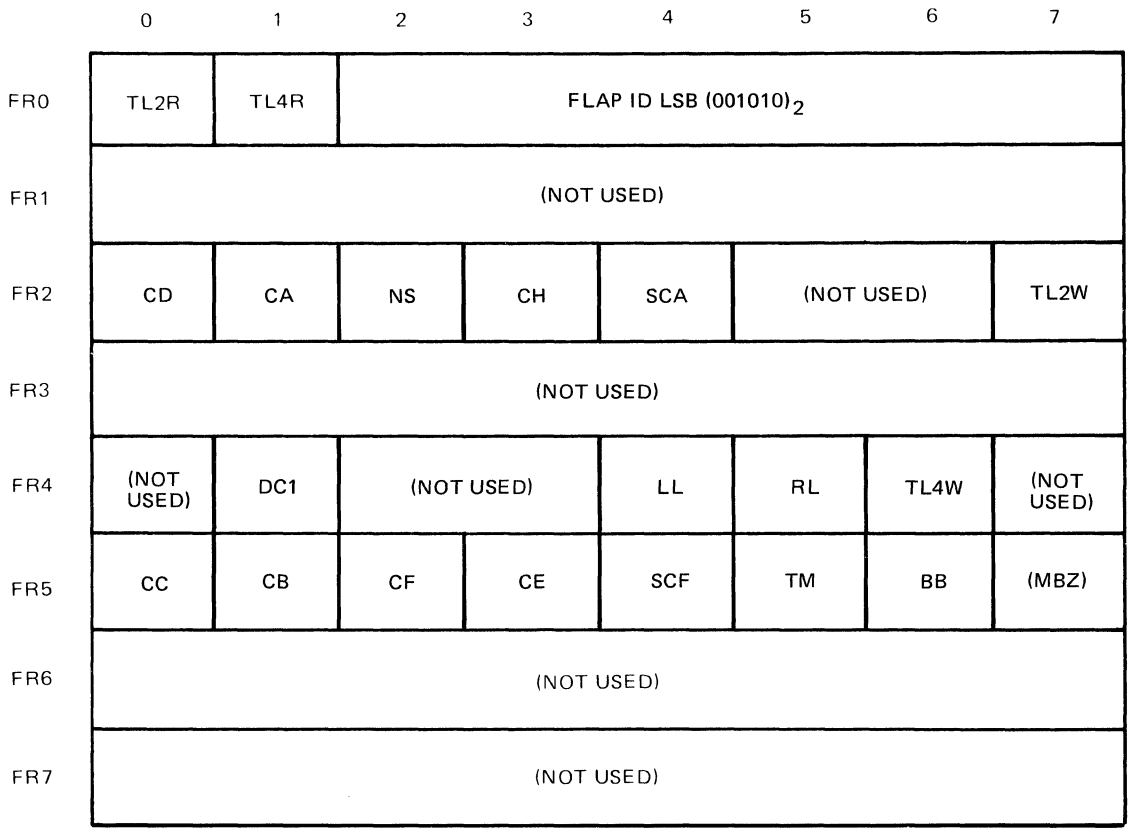


Figure E-16. MIL-STD-188-114 FLAP Register Definition

MIL-STD-188C COMPATIBILITY

This FLAP is operable with MIL-STD-188C installations made after June 1, 1976, with data sense as shown in Table E-9, footnote e. For MIL-STD-188C installations prior to June 1, 1976, the FLAP is configurable to the data sense shown in Table E-9, footnote b, and thus is operable with these installations. A DIP switch on the FLAP selects which data sense scheme is used.

Table E-9. Voltage Polarity of DTE/DCE Unbalanced Voltage Interchange Circuits

DTE/DCE Interchange Standard	Binary 1	Signal State State Mark	Control State Off	Binary 0	Signal State Space	Control State On
RS-232C	Neg	Neg	Neg	Pos	Pos	Pos
RS-423/RS-423A	Neg	Neg	Neg	Pos	Pos	Pos
MIL-188B[a]	Pos	Pos	Neg	Neg	Neg	Pos
MIL-188C[b]	Pos	Pos	Neg	Neg	Neg	Pos
MIL-188-100[c]	Pos	Pos	Neg	Neg	Neg	Pos
MIL-188-114[d]	Neg	Neg	Neg	Pos	Pos	Pos
MIL-188C[e]	Neg	Neg	Neg	Pos	Pos	Pos
MIL-188-100[f]	Neg	Neg	Neg	Pos	Pos	Pos
MIL-188-200[g]	Neg	Neg	Neg	Pos	Pos	Pos
CCITT-V.28	Neg	Neg	Neg	Pos	Pos	Pos

[a]MIL-188B as issued February 24, 1964
[b]MIL-188C as issued February 24, 1969
[c]MIL-188-100 as issued November 15, 1972
[d]MIL-188-114 as issued March 24, 1976
[e]MIL-188C per Change Notice 1 issued June 1, 1976
[f]MIL-188-100 per Change Notice 2 issued June 1, 1976
[g]MIL-188-200 as issued December 1978

FLAP BUS INTERFACE

The FLAP bus provides for simultaneous two-way, bit-serial data transfer among all the FLAPs and the controller/adapter plus an 8-bit parallel HDX transfer of register information between a FLAP and the controller/adapter. For this latter transfer, the controller/ adapter is the initiator (master). There is no communication capability among FLAPs on the bus.

The simultaneous bit-serial transfers are accomplished via the directed portion of the bus while the HDX register transfer is accomplished via the shared portion.

Appendix F
ASYNCHRONOUS DIRECT CONNECT ADAPTER

INTRODUCTION

This appendix defines the New Mutliline Controller Asynchronous Communications Line Adapter, integrated DSA-46 Direct Connect (NACLA/DC).

The adapter is packaged on a quarter-size board that attaches to the New Multiline Communications Processor (NMLCP). The NACLA/DC supports up to four communications lines operating in any mixture of full- or half-duplex communications. The protocol on any line is asynchronous, and the maximum line speed is 19.2K bps.

SOFTWARE OVERVIEW

A requirement in the development of the NMLCP is that existing Channel Control Programs (CCPs) written for use with the ACLA on the MLCP must, on an object code basis and without need for change, be usable for operating the NACLA/DC. Also new CCPs written for the NACLA/DC operation should operate without change if and when future NACLA/DC versions using different hardware are implemented.

The prime visibility of adapters on the MLCP to the CCP for control, data access, and status-sensing purposes is that of a set of Line Registers (LRs). In the NACLA/DC, this same visibility and assignment of LRs and their contents will be maintained and will continue to be maintained in future NACLA/DC versions. Firmware will map the LR content into/from the appropriate spots in the NACLA/DC.

Associated with each USART in the NACLA/DC is a set of registers for control and status indication of the USART itself. Adapters on the MLCP used USARTs that had limited capability and flexibility. All required control and status by bit positions in the LRs. Firmware automatically provides mapping of these LR bits into and from the appropriate USART registers. In order that existing CCPs may be used with the NACLA/DC, necessary access to USART registers not represented in the LRs must not require CCP instructions. This also allows future replacement of the USART component with a newer type without impacting the viability of existing CCPs. USART control and status sensing required to be done on a dynamic basis (e.g., error sensing, error reset, etc.) will be performed automatically by firmware. Thus, replacement of the USART with a newer type would impact firmware, but would not affect the CCP.

Figure F-1 shows the register sets associated with the NACLA/DC and their means of access by software.

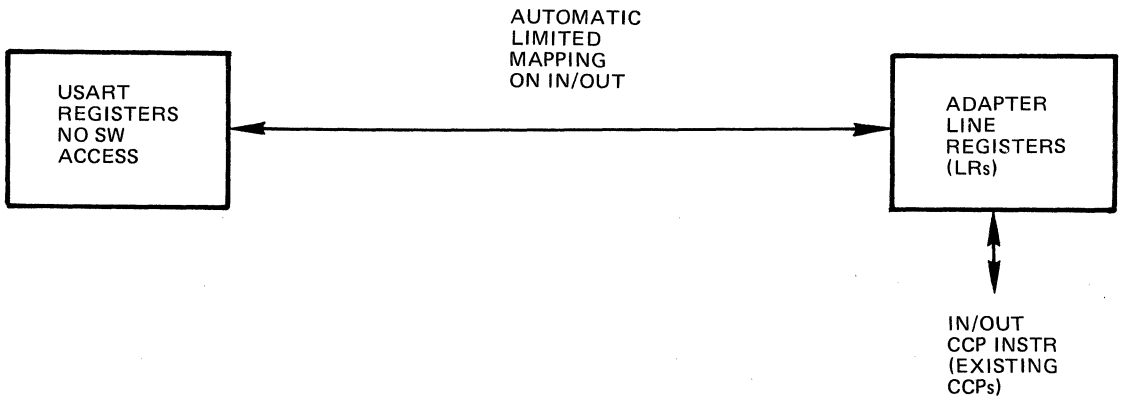


Figure F-1. NACLA/DC Register Access

To summarize, software visibility to the NACLA/DC in the case of existing CCPs is exclusively through the LRs. Firmware loads certain default values into the USART registers upon initialization. These are values needed for USART operation, that are not supplied via the LRs. Additionally, firmware maps LR bits to appropriate bit positions in the USART registers and vice versa. With the initial values plus subsequent LR-to/from-USART register mapping, existing CCPs and drivers will operate without modification.

New functionality over and above that of the MLCP is implemented by the utilization of LR bits hitherto unused. As mentioned above, new CCPs written to capitalize on this functionality will continue to be visible, and the new functionality will survive any USART changes.

DATA FORMATS

The asynchronous data format of the NACLA/DC is basically asynchronous bit serial with character synchronization using framing (start/stop) bits. The character consists of 5, 6, 7, or 8 data bits plus parity, if used, plus a start bit and one or two stop bits. The data is transmitted least significant bit first. During idle periods in transmission, a continuous stop bit (marking) is transmitted. The data format is shown in Figure F-2.

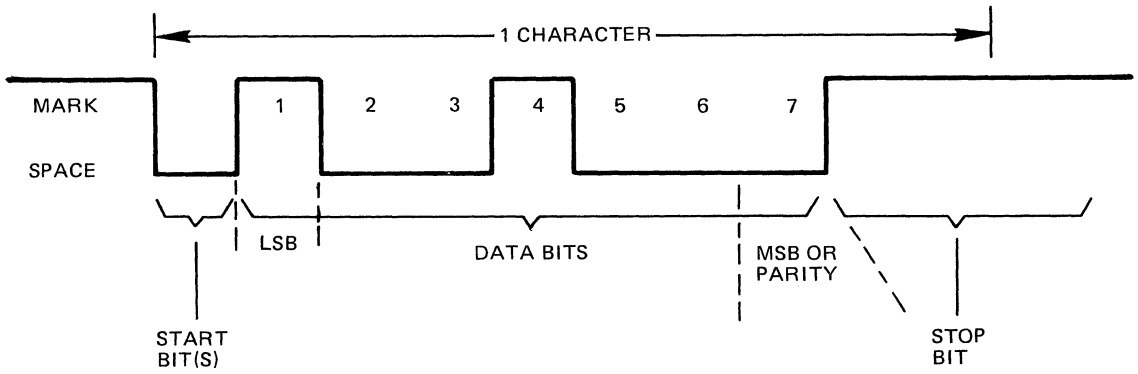


Figure F-2. Asynchronous Data Format

BASIC FUNCTIONS

A typical communications subsystem using NMLCP-family products consists of an assemblage of components connected through a set of interfaces as shown in Figure F-3. This appendix addresses one of a set of adapters that can be connected to the Processor, namely, the NACLA/DC.

The primary NACLA/DC functional components are illustrated in Figure F-4.

The NACLA/DC supports four lines, full or half duplex, at rates up to 19.2K bps. Each line can be separately configured as to speed, data format, character size, and error control.

ACLA Functionality

The NACLA/DC provides the total functionality of the ACLA. Visibility to software (via Line Registers) is identical to the ACLA, as applicable.

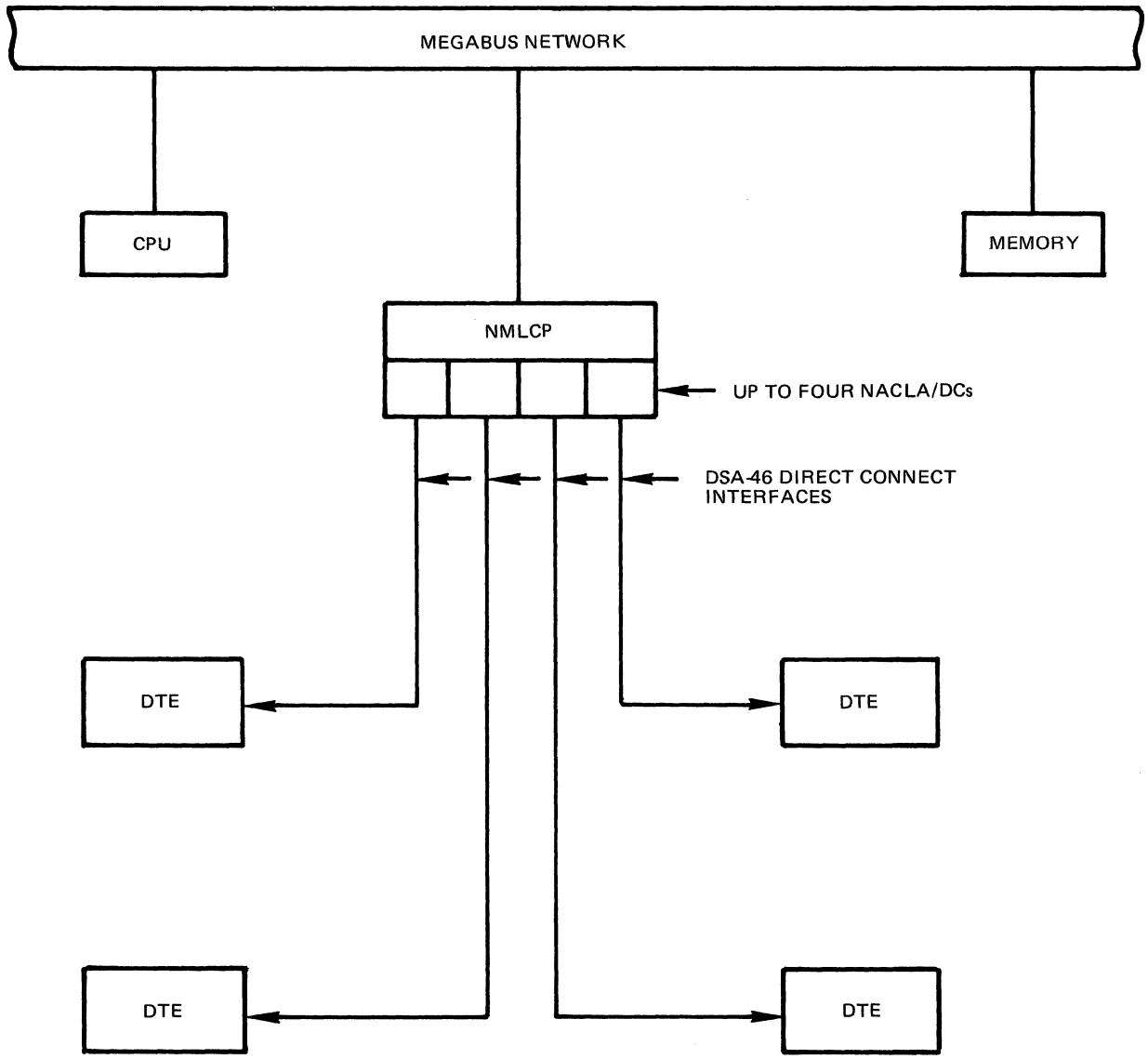


Figure F-3. NMLCP Communications Subsystem

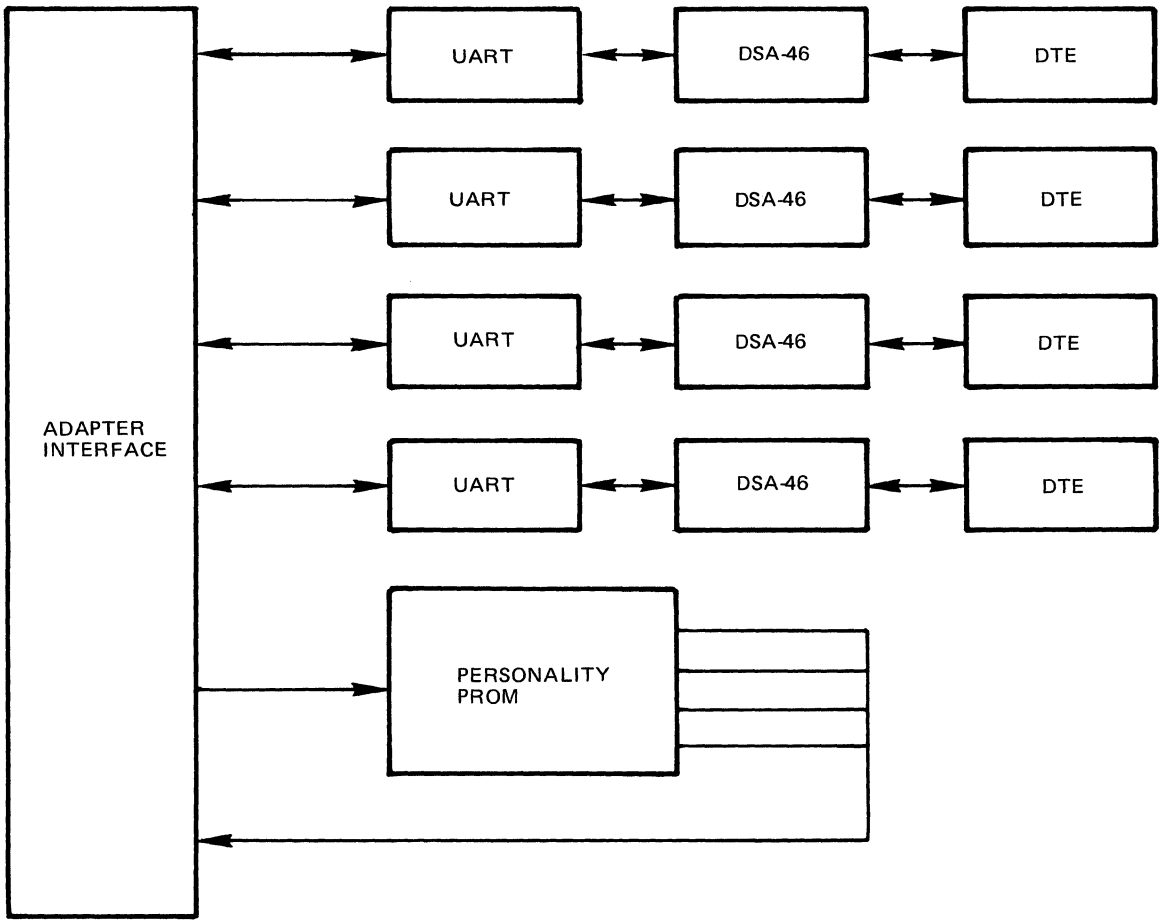


Figure F-4. NACLA/DC Block Diagram

New Functionality

The NACLA/DC provides functionality over and above that required to emulate the ACLA. This new functionality consists of:

1. Parity Generation/Detection
2. Support for Enhanced Break Detection
3. Eight bits plus parity support.

This use of the new functionality is optional; it can be incorporated with new CCPs. In no way does it impair the operation of existing CCPs written to be run on the MLCP/ACLA.

The NACLA/DC functionality, including new functionality as well as the ACLA subset, is described in the following subsections.

LINE REGISTERS

Software visibility to the NACLA/DC is via a set of Line Registers (LRs).

The NACLA/DC and Processor exchange program-visible data, configuration, and control information and status by means of these LRs.

Register information is passed between the Processor and NACLA/DC by the Processor specifying to the NACLA/DC the address and the direction of the transfer. The transfer is 8 bits in parallel.

In this regard, the following subsections detail the bit assignments of the LRs; reference is made to various CCP instructions (IN, OUT, SEND, RECEIVE, etc.). These instructions are decoded and converted by the Processor into commands for the NACLA/DC on the NMLCP/NACLA/DC interface, to which the NACLA/DC responds by either sending an LR or presenting the contents of an LR to the Processor, as appropriate.

It should be noted that a register identifed as READ is written or vice versa. If a SEND/RECEIVE instruction is misused, or if the Transmit channel accesses a Receive Channel register or vice versa, the results are unpredictable.

Line Register Bit Assignments (ACLA Mode)

The LR bit assignments are shown in Figure F-5. These assignments are identical to those used in the ACLA.

LINE REGISTER 0 (LR0)

LR0 contains the two high-order hex digits of the NACLA/DC extended ID number which is $(3A00)_{16}$.

LINE REGISTER 1 (LR1)

The NACLA/DC and the Processor exchange communications line data in parallel data elements on a channel basis through LR1. The NACLA/DC's USARTs convert parallel data to bit serial data for transmission and convert bit serial data to parallel data on receive. Each channel contains a serial-input or serial-output buffer.

A separate LR1 is provided for each channel. The receive channel, via RECV or IN1 instructions, may obtain data characters from this channel. The CCP servicing the receive channel should not use SEND or OUT1 instructions.

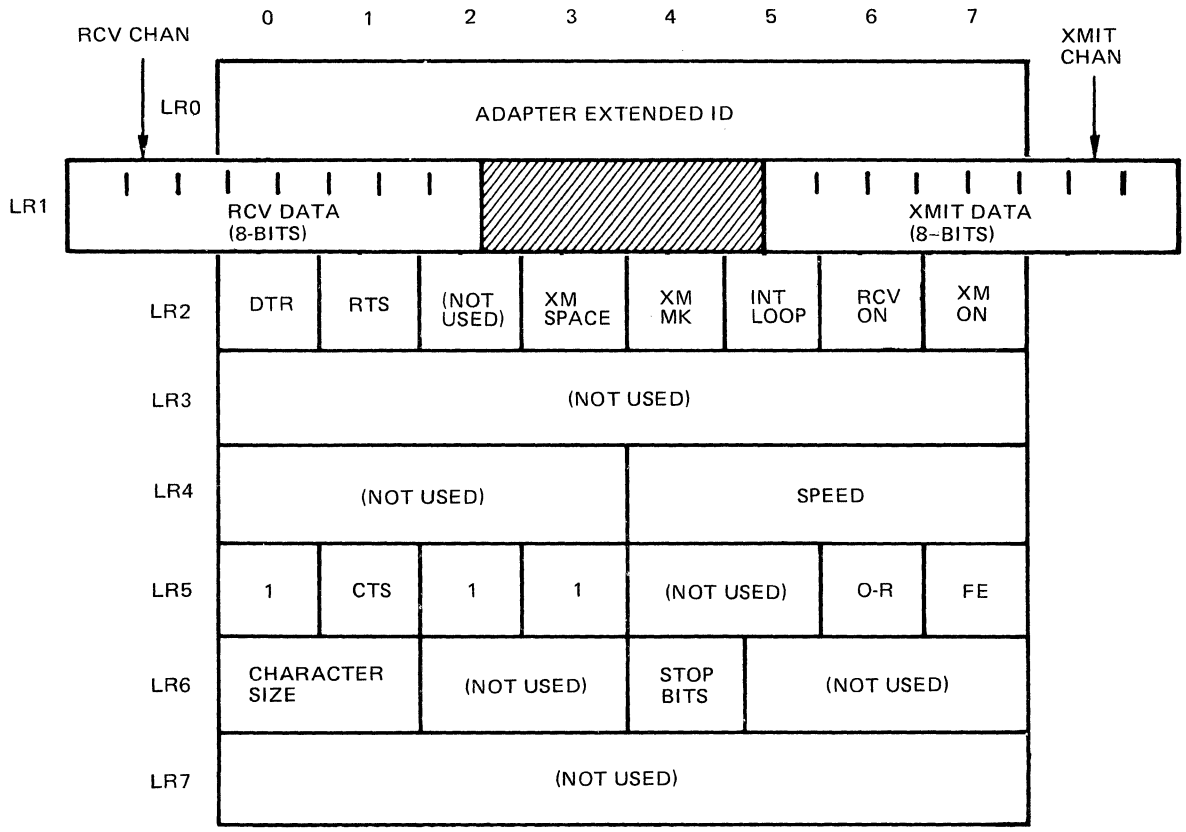


Figure F-5. NACLA/DC Line Registers - Compatible Functionality Only

The transmit channel delivers data characters to the register via SEND or OUT1 instructions. The CCP servicing the transmit channel should not use RECV or IN1 instructions. Bit 7 (least significant bit) is the first bit transmitted or received. If the character size, including parity, is less than 8 bits, the most significant bits will be Zeros.

LINE REGISTER 2 (LR2)

LR2 is used for control purposes by both channels and can be written. An OUT2 instruction delivers bits 0 through 7 to LR2.

Bit names are as follows:

```
Bit 0, DTR      - Data Terminal Ready
Bit 1, RTS      - Request To Send
Bit 2,          - Not Used
Bit 3, XM SPACE - Transmit Space (Break)
Bit 4, XM MARK  - Transmit Mark (Mark Line)
Bit 5, INT LOOP - Internal Loop (Test)
Bit 6, RCV ON   - Receive ON
Bit 7, XM ON    - Transmit ON
```

Bits 0 and 1 are DTE-to-DTE control signals retained for software compatibility with the ACLA.

Other bit meanings are:

Bit 3 - Transmit Space:

0 - Transmit Data supplied by CCP
1 - Hold Data Output in SPACE (logical 0) condition

Bit 4 - Transmit Mark:

0 - Transmit Data supplied by CCP
1 - Hold Data Output in MARK (logical 1) condition

NOTE

Bits 3 and 4 are mutually exclusive and therefore both must not be set to One.

Bit 5 - Internal Loop (Test): Each line in the NACLA/DC has a software-controlled Test mode that internally loops back data from the transmit channel to the receive channel.

Setting the INT LOOP bit on the transmit or receive channel invokes the Test mode. During Test mode, the data rate is determined by the Processor.

Bit 6 - Receive ON:

0 - Set the Receiver OFF for this line
1 - Set the Receiver ON for this line

Bit 7 - Transmit ON:

0 - Set the Transmitter OFF for this line
1 - Set the Transmitter ON for this line

LINE REGISTER 3 (LR3) - Not Used

LINE REGISTER 4 (LR4)

LR4 contains a code in bits 4 through 7 that defines the line speed to be used. The codes and corresponding speeds are:

Bits 4-7	Rate (bps)
0000	50
0001	75
0010	110
0011	134.5
0100	150
0101	200
0110	300
0111	600
1000	1050
1001	1200
1010	1800
1011	2000
1100	2400
1101	4800
1110	9600
1111	19200

LINE REGISTER 5 (LR5)

LR5 is used for status purposes by both channels.

Whenever LR5 is read via the IN5 instruction, the NACLA/DC delivers the entire contents of LR5 to the Processor.

Bit names are as follows:

Bit 0 - Binary 1
Bit 1, CTS - Clear To Send (This reflects the state of RTS in LR2.)
Bit 2 - Binary 1
Bit 3 - Binary 1
Bit 4 - Not Used
Bit 5 - Not Used
Bit 6, O-R - Overrun (Receive)
Bit 7, FE - Framing Error (Receive).

Bits 0 through 3 are DTE-to-DTE Control Signals. Other bit meanings are:

Bit 6 - Overrun:

0 - No Receive overrun has occurred.
1 - A Receive overrun has occurred. The NACLA/DC was not answered fast enough by the receive CCP, and one or more data characters were overwritten (and thus lost) in the receive channel's LR1.

Bit 7 - Framing Error:

0 - No Framing error has occurred.
1 - A Framing error has occurred. The NACLA/DC has detected a missing stop bit.

LINE REGISTER 6 (LR6)

LR6 can be written or read, and it provides configuration data to the NACLA/DC. Bit meanings are as follows:

Bits 0 and 1, Character Size:

00 - 5 bits
01 - 6 bits
10 - 7 bits
11 - 8 bits.

Bit 4 - Stop bits:

0 - 1 stop bit per 5-, 6-, 7-, or 8-bit data character
1 - 1.5 stop bits per 5-bit data character
1 - 2 stop bits per 6-, 7-, or 8-bit data character.

LINE REGISTER 7 (LR7) - Not Used

Line Register Bit Assignments (Non-ACLA Mode)

Non-ACLA Mode functionality is provided by the NMLCP beyond that of the MLCP. The register bits involved are illustrated in Figure F-6. In this figure, bits not shown are as in Figure F-5. The adapter control and adapter status registers are accessed via the IAS and OAC instructions respectively.

1. Line Register 5, bits 2 and 3, when written via the OUT5 instruction:

   Bit 3 (PE) - Parity Enable
   Bit 2 (PT) - Parity Type.

If bit 3 is set, a parity bit is added to the transmitted character, and the receiver performs a parity check on incoming 8-bit data:

Bit 2 = 0 selects odd parity
Bit 2 = 1 selects even parity.

Bit 2 is ignored unless bit 3 = 1.

Eight bits plus parity can therefore be supported in the adapter by software setting LR6 bits 0 and 1 to 11 (8-bit) and setting LR5 bits 2 and 3 as appropriate. For other character sizes, parity must be supported in the NMLCP analogous to the MLCP.

2. Line Register 5, bit 5 (Parity Error), when read via the IN5 instruction:

   Bit 5 = 0, no error detected
   Bit 5 = 1, parity error detected (8 bits plus the parity bit was enabled by the PE bit).

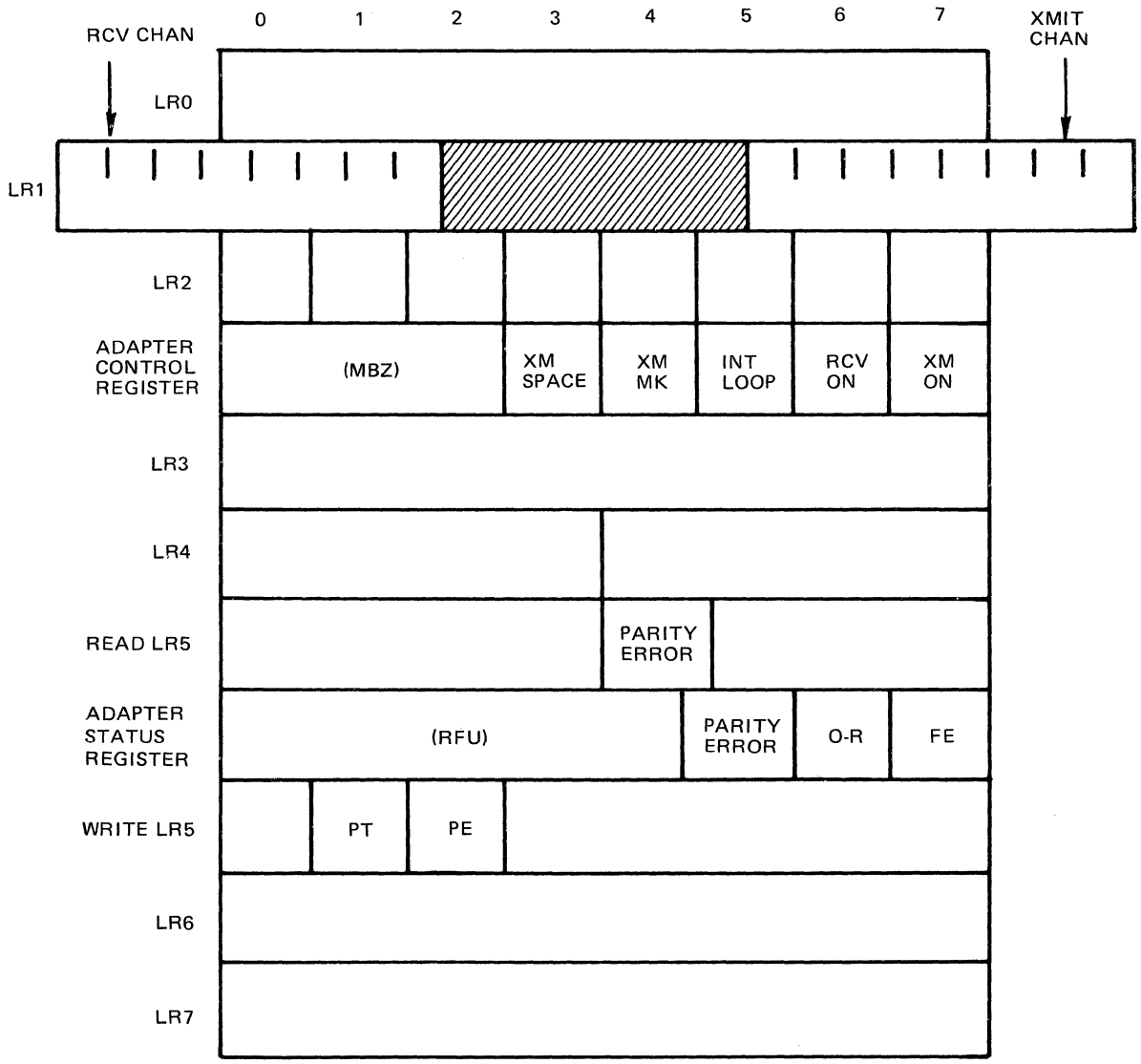


Figure F-6. NACLA/DC Line Registers - New Functionality Only

Composite Line Register Bit Assignments

The complete Line Register bit assignments are shown in Figure F-7.

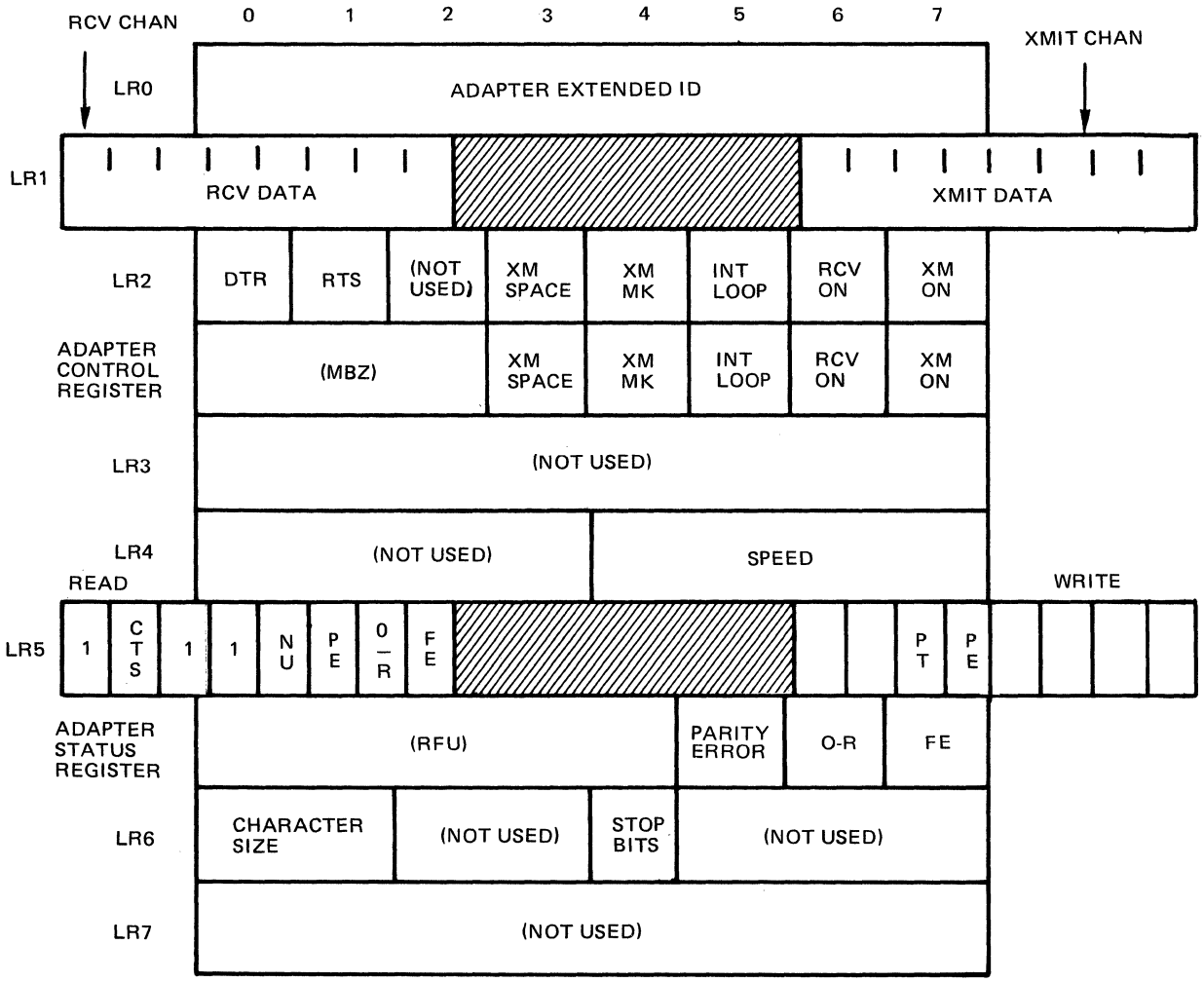
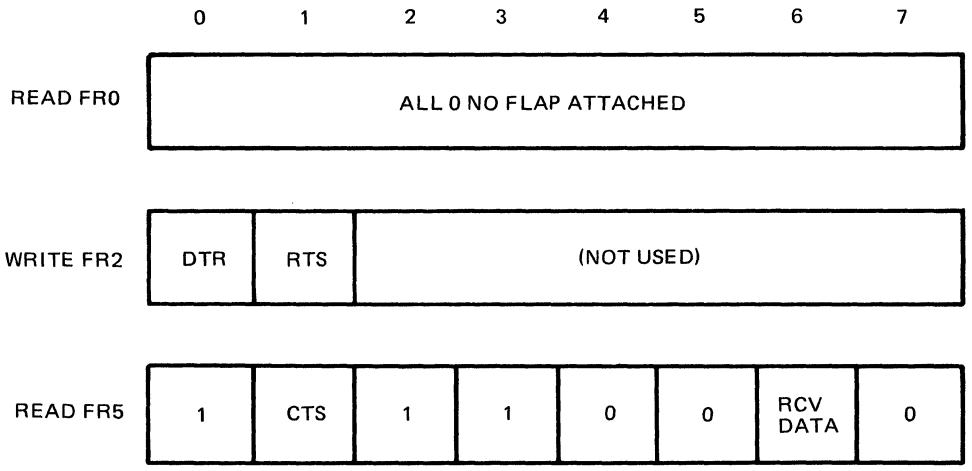


Figure F-7. NACLA/DC Line Registers - Composite Bit Assignments

FLAP Register Bit Assignments

Although there is no FLAP hardware attachable to the NACLA/DC, a subset of the FLAP registers is provided on the adapter itself. The FLAP register definitions are as follows:



The DTR, RTS, and CTS bits are the same bits accessed by OUT2 and IN5 instructions.

MISCELLANEOUS ADDITIONAL FUNCTIONALITY

Other new functionality is available with the NACLA/DC. This functionality includes:

Enhanced Break Detection:

Framing errors, as reported by LR5, bit 7, are retained from previous designs. This status indication can be used by CCPs and can be used to indicate the start of a BREAK. FR5, bit 6, tracks receive data, and it can therefore be monitored to indicate the cessation of a BREAK.

CHANNEL REQUEST INTERRUPT (CRI)

Each channel is capable of generating its own CRI to signal a need for data service.

On receive, when a previously empty receive-holding register is loaded with a data character by the USART, a CRI is generated by that channel.

If the RCV ON bit is turned off, channel CRIs are disabled and any characters in the USART are discarded.

On transmit, a CRI is generated by the channel when the last character loaded has been serialized. If the XM ON bit is turned off, all characters in the USART are transmitted completely and channel CRIs are disabled.

The CRI generation conditions mentioned above are those in which the channel would have previously entered a WAIT condition.

If the RCV ON bit or the XM ON bit is on, the USART channel is enabled, and the CRI function is enabled. Turning these bits on will eventually result in a CRI, since the channel would previously have been in a WAIT condition.

Each NACLA/DC attached to the Processor via the adapter interface has an Interrupt High Priority and an Interrupt Low Priority line on the bus. Receive channels are high priority, and transmit channels are low priority; and, within each set, the lower the channel number, the higher the priority. The NMLCP responds to the interupt by requesting from the adapter the channel requiring service. The NACLA/DC then provides the channel number on the bus. Thus, the Processor decides which adapter is to be serviced, and the NACLA/DC determines which channel.

CONFIGURATION

Upon initialization, firmware will load the USART registers with those default values that are not provided by current CCPs via the LRs. Firmware then sets up the following USART default value:

Parity Generation/Detection disabled.

This initial value, plus those subsequently mapped from the LRs, enable existing software designed for the MLCP and its synchronous adapters to operate on the NMLCP/NACLA/DC without change.

New CCPs can override these initial values by writes to LR4, LR5, and LR6, and in so doing capitalize on the new functionality discussed in the subsection entitled "Line Register Bit Assignments (Non-ACLA Mode)." The CCPs should ensure that the Transmitter and Receiver are OFF by a write to LR2 with bits 6 and 7 equal to Zero. Line Registers 4, 5, and 6 should thus be configured, and finally the Transmitter and Receiver should be enabled.

CHANNEL NUMBER ASSIGHNMENT

The channel numbering for a fully configured NMLCP with four NACLA/DCs is given in Table F-1. Channels 0 through 7 apply to one NACLA/DC, channels 8 through 15 apply to the second NACLA/DC, etc.

Table F-1. Channel Numbering

Channel	Line	Direction
0	0	R
1	0	T
2	1	R
3	1	T
4	2	R
5	2	T
6	3	R
7	3	T
.	.	.
.	.	.
.	.	.
30	15	R
31	15	T

DEVICE INDENTIFICATION NUMBER

The basic identification number is furnished by the Processor. The extended device identification provided by the NACLA/DC is $(3A00)_{16}$.

DATA CLOCK

A baud rate generator in the NACLA/DC frequency divides a clock source provided by the Processor to produce a line-speed bit-rate clock at the rate specified by the SPEED field in LR4 (see subsection entitled "Line Register 4 (LR4)"). The USART uses this line-speed clock to time data serialization.

RECEIVE SYNCHRONIZATION

Synchronization is established for each character received by the presence of framing (start and stop) bits associated with each character. On a Break, as evidenced by the NACLA/DC detecting a message-stop bit, the framing-error bit is set and remains set until reset by software, which can be accomplished by switching the Receiver off and then on.

RECEIVE OVERRUN

If a receive channel overflows because the Processor did not service the channel's CRI soon enough, receive data is lost and the receive-overrun bit (LR5, bit 6) is set. This bit will be sensed by the next RECV instruction. Clearing the error and resync of the receiver is the responsibility of the CCP, and it can be performed by switching the receiver off and then on.

TRANSMIT UNDERRUN/TRANSMIT FILL

If a transmit-channel character is lost because the Processor did not service the channel's CRI soon enough, a transmit underrun condition exists.

The transmitter will simply leave the line in a continuous marking (stop bit) condition when there are no characters to send and no underrun indication is reported.

INITIALIZATION

The Processor Hard Initialize clears all USARTs and all line registers. Each line must be configured before it can be operated.

Channel Initialize clears interrupts and errors and switches off the receiver and transmitter.

SPECIAL FUNCTIONALITY FOR THE SCF

The adapter provides some special functionality to support connection of the System Control Facility (SCF) and use of existing console-driver software. The SCF will connect to the adapter via a special cable that is limited to a 10-foot maximum length. The SCF and the cable provide an unbalanced input signal (Data Flow Control) on pin 15 of the connector.

The Data Flow Control (DFC) signal is ANDed with Request to Send (LR2, bit 1), and the result is set into Clear to Send (LR5, bit 1) on Line 0 only.

The adapter includes internal pull-up circuitry so that for DTEs other than the SCF that do not implement this signal, Clear to Send will operate normally; that is, it is set when Request to Send is set. Note that because of the distance limitation imposed, when the DFC signal is implemented, the receiving circuit in the adapter need not necessarily meet normal specifictions. Flow control is provided on one of the four lines only; that is, Line 0.

Appendix G
GLOSSARY

ACLA

Asynchronous Communications Line Adapter

ACUF

Automatic Calling Unit FLAP

ASCII

American (National) Standard Code for International Interchange

BSC

Binary Synchronous Communication

CA

Communications Adapter

CCB

Communication Control Block

CCITT

The International Telegraph and Telephone Consultative Committee

CCOB

Communication Control Order Block

CCP

Channel Control Program

CDB

Communications Data Block

CLA

Communications Line Adapter

CLM

Configuration Load Manager

CPU

Central Processing Unit

CRC

Cyclic Redundancy Check

CRI

Channel Request Interrupt

Channel

A simplex communication path with associated control. Two channels are required for each HDX or FDX line.

Command

An I/O operation generated by the DPS 6/Level 6 CPU used to control the NMLCP.

DCE

Data Communications Equipment

DMA

Direct Memory Access

DSA

Distributed Systems Architecture

DTE

Data Terminal Equipment

EA

Effective Address

FC

Function Code

FDX

Full Duplex

FIFO

First-In/First-Out

FLAP

Flexible Line Adapter Package

FR

FLAP Register

HDLC

High-Level Data Link Control

HDX

Half Duplex

I/O

Input/Output

ID

Identification (Device)

IMA

Immediate Memory Address

IMO

Immediate Memory Operand

Isochronous

Asynchronous Operation with an external clock supplied by the data set.

LCT

Line Control Table

LIFO

Last-In/First-Out

LR

Line Register

MBZ

Must Be Zero

MLCP

Multiline Communications Processor

MSB

Most Significant Bit

MTBF

Mean Time Between Failures

MTTR

Mean Time To Repair

NACLA/DC

New Asynchronous Communications Line Adapter, Integrated DSA-46 Direct Connect

NAK

Negative Acknowledgement

NMLCA

New Multiline Controller Adapter

NMLCP

New Multiline Communications Processor

NSAA

New Multiline Controller Synchronous/Asynchronous Adapter

ORU

Optimum Replaceable Unit

PARS

Programmed Airline Reservation System

QLT

Quality Logic Test

RAM

Random Access Memory

RFU

Reserved for Future Use

RHU

Reserved for Hardware Use

ROM

Read Only Memory

RSU

Reserved for Software Use

SNA

Systems Network Architecture

TBD

To Be Defined

T&V

Test and Verification

USART

Universal Synchronous/Asynchronous Receiver/Transmitter (LSI Communication Chips)

USRT

Universal Synchronous Receiver/Transmitter

WCLA

Wideband Communications Line Adapter

Instruction	Functional Description	Time	Page
ADD R,lct	(R) <— (R) + (LCT)	0.75	3-20
ADD B,lct	(B) <— (B) + (LCT)	0.75	3-21
ADD R,B	(R) <— (R) + ((B))	0.75	3-22
ADD R,=imo	(R) <— (R) + IMO	0.75	3-23
ADD B,=imo	(B) <— (B) + IMO	0.75	3-24
AND R,lct	(R) <— (R) ^ (LCT)	0.75	3-25
AND B,lct	(B) <— (B) ^ (LCT)	0.75	3-26
AND R,B	(R) <— (R) ^ ((B))	0.75	3-27
AND R,=imo	(R) <— (R) ^ IMO	0.75	3-28
AND B,=imo	(B) <— (B) ^ IMO	0.75	3-29
B label	(P) <— (P) + disp	1.5	3-30
BS label	(LCT 18) <— (P) + 1 (P) <— (P) + disp	3.75	3-31
BARF label	If AR=0; then (P) <— (P) + disp	3.0	
	else (P) <— (P) + 1	4.5	3-32
BART label	If AR=1; then (P) <— (P) + disp	3.0	
	else (P) <— (P) + 1	4.5	3-33
BCF label	If C=0; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-34
BCT label	If C=1; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-35
BEF label	If E=0; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-36
BET label	If E=1; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-37
BLBF label	If LB=0; then (P) <— (P) + disp	1.5	
	else (P) <— (P) + 1	2.8	3-38
BLBT label	If LB=1; then (P) <— (P) + disp	1.5	
	else (P) <— (P) + 1	2.8	3-39
BLCF label	If LC=0; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-40
BLCT label	If LC=1; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-41
BSBF label	If SB=0; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-42
BSBT label	If SB=1; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-43
BSF label	If Z=0; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-44
BST label	If Z=1; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-45
BVBF label	If VB=0; then (P) <— (P) + disp	1.8	
	else (P) <— (P) + 1	2.8	3-46
BVBT label	If VB=1; then (P) <— (P) + disp	1.8	
	else (P) <— (P) + 1	2.8	3-47
BZF label	If Z=0; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-48
BZT label	If Z=1; then (P) <— (P) + disp	0.75	
	else (P) <— (P) + 1	2.0	3-49
C R,lct	(W) <— (R) - (LCT)	0.75	3-50
C B,lct	(W) <— (B) - (LCT)	0.75	3-51
C R,B	(W) <— (R) - ((B))	0.75	3-52
C R,=imo	(W) <— (R) - IMO	0.75	3-53
C B,=imo	(W) <— (B) - IMO	0.75	3-54
CADD R,lct	(R) <— (R) + C + (LCT)	3.25	3-55
CADD B,lct	(B) <— (B) + C + (LCT)	3.25	3-56
CADD R,B	(R) <— (R) + C + ((B))	3.4	3-57
CADD R,=imo	(R) <— (R) + C + IMO	3.2	3-58
CADD B,=imo	(B) <— (B) + C + IMO	3.2	3-59

Instruction	Functional Description	Time	Page
CANB	Curr Address <— Init Address Curr Range <— Init Range	6.75	3-60
CANC	(W) <— Curr Range - Init Range If Z = 0; then Curr Address <— Curr Address - 1 Curr Range <— Curr Range + 1		3-61
CCH	(CRC) <— (CRC) + (R)	3.9-9.5	3-62
CL lct	(LCT) <— 0	0.75	3-63
CL B	((B)) <— 0	0.75	3-64
CL =R	(R) <— 0	0.38	3-65
CL =B	(B) <— 0	0.38	3-66
DADD R,lct	(R) <— (R) + (LCT)	5.88	3-67
DADD R,=imo	(R) <— (R) + IMO	5.75	3-68
DEC =R	(R) <— (R) - 1	0.38	3-69
DEC lct	(LCT) <— (LCT) - 1	0.38	3-70
DEC B	((B)) <— ((B)) - 1	0.38	3-71
DEC =B	(B) <— (B) - 1	1.0	3-72
DEC *lct[a]	((LCT):(B)) <— ((LCT):(B)) - 1	7.0	3-73
DEQ[b]	(R) <— ((SOQ)) (SOQ) <— (SOQ) + 1	1.2	3-74
ENQ[b]	((EOQ)) <— (R) (EOQ) <— (EOQ) + 1	1.2	3-75
FIN fr	(R) <— (FR)	2.6	3-76
FOUT fr	(FR) <— (R)	4.4	3-77
GIVE	Post Current CCB - Extended		3-78
GNB	Post Current CCB - Compatible	10.0	3-79
IAS	(R) <— CA Status	3.4	3-80
ILP lct,label	(LCT) <— (P) + Disp	1.2	3-81
IN lr	(R) <— (LR)		3-82
INC lct	(LCT) <— (LCT) + 1	0.38	3-83
INC B	((B)) <— ((B)) + 1	0.38	3-84
INC =R	(R) <— (R) + 1	0.38	3-85
INC =B	(B) <— (B) + 1	1.0	3-86
INC *lct[a]	((LCT):(B)) <— ((LCT):(B)) + 1	7.0	3-87
INTR	Interrupt CPU		3-88
INZ	Stop I/O		3-89
JS label[b]	(SP) <— (SP) - 1 ((SP)) <— (P) + 1 (SP) <— (SP) - 1 (P) <— (P) + Disp	6.2	3-90
JS *lct[b]	(SP) <— (SP) - 1 ((SP)) <— (P) + 1 (SP) <— (SP) - 1 (P) <— (LCT)		3-91
JUMP label	(P) <— (P) + Disp	3.0	3-92
JUMP *lct[a]	(P) <— (LCT)		3-93

CCP Programming Instruction Card (Side 1)

Instruction	Functional Description	Time	Page
LB lct,=imo	(W) <— (LCT) ^ IMO	1.25	3-94
LB B,=imo	(W) <— ((B)) ^ IMO	0.25	3-95
LB =R,=imo	(W) <— (R) ^ IMO	0.75	3-96
LB =B,=imo	(W) <— (B) ^ IMO	0.75	3-97
LB *lct,=imo[a]	(W) <— ((LCT):(B)) ^ $\overline{IMO}$	6.1	3-98
LBF lct,=imo	(W) <— (LCT) ^ IMO (LCT) <— (LCT) ^ IMO	2.3	3-99
LBF B,=imo	(W) <— ((B)) ^ IMO ((B)) <— ((B)) ^ IMO	2.3	3-100
LBF *lct,=imo[a]	(W) <— ((LCT):(B)) ^ IMO ((LCT):(B)) <— ((LCT):(B)) ^ IMO	7.9	3-101
LBT lct,=imo	(W) <— (LCT) ^ IMO (LCT) <— (LCT) v IMO	2.3	3-102
LBT B,=imo	(W) <— ((B)) ^ IMO ((B)) <— ((B)) v IMO	2.3	3-103
LBT *lct,=imo[a]	(W) <— ((LCT):(B)) ^ IMO ((LCT):(B)) <— ((LCT):(B)) v IMO	2.3	3-104
LD,	(R) <— (CDB)	0.75	3-105
LD R,lct	(R) <— (LCT)	0.75	3-106
LD B,lct	(B) <— (LCT)	0.75	3-107
LD R,B	(R) <— ((B))	0.75	3-108
LD R,=imo	(R) <— IMO	0.75	3-109
LD B,=R	(B) <— (R)	0.75	3-110
LD B,=imo	(B) <— IMO	0.75	3-111
LD R,=B	(R) <— (B)	0.75	3-112
LD R,*lct[a]	(R) <— ((LCT):(B))	6.1	3-113
MC	Master Clear		3-114
NEG[c]	(R) <— $\overline{(R)}$ + 1		3-115
NOP	(P) <— (P) + 1	0.38	3-116
OAC[d]	CA Control <— (R)		3-117
OR R,lct	(R) <— (R) v (LCT)	0.75	3-118
OR B,lct	(B) <— (B) v (LCT)	0.75	3-119
OR R,B	(R) <— (R) v ((B))	0.75	3-120
OR R,=imo	(R) <— (R) v IMO	0.75	3-121
OR B,=imo	(B) <— (B) v IMO	0.75	3-122
OUT lr	(LR) <— (R)		3-123
PULL B[b]	(SP) <— (SP) + 1 (B) <— ((SP))	1.1	3-124
PULL R[b]	(SP) <— (SP) + 1 (R) <— ((SP))	1.1	3-125
PUSH B[b]	((SP)) <— (B) (SP) <— (SP) - 1	1.1	3-126
PUSH R[b]	((SP)) <— (R) (SP) <— (SP) - 1	1.1	3-127
RECV 0	(R) <— (LRl) $\overline{par;crc}$	5.4	3-128
RECV 1	(R) <— (LRl) par;$\overline{crc}$	11.1	3-128
RECV 2	(R) <— (LRl) $\overline{par}$;crc	8.5	3-128
RECV 3	(R) <— (LRl) par;crc	13.8	3-128
RETB	(P) <— (LCT 18)	1.5	3-129
RETJ[b]	(SP) <— (SP) + 1 (P) <— ((SP)) (SP) <— (SP) + 1	2.5	3-130
RHB	Return Held Block		3-131
SCF	C <— 0	1.2	3-132
SCL R	(R) <— (R) . 2 + C	1.3	3-133
SCL B	(B) <— (B) . 2 + C	1.3	3-134
SCR R	(R) <— (R) / 2 + C . X'80'	1.3	3-135
SCR B	(B) <— (B) / 2 + C . X'80'	1.3	3-136
SCT	C <— 1	1.2	3-137
SEND 0	(LRl) <— (R) $\overline{par;crc}$	5.8	3-138
SEND 1	(LRl) <— (R) par;$\overline{crc}$	10.9	3-138
SEND 2	(LRl) <— (R) $\overline{par}$;crc	7.9	3-138
SEND 3	(LRl) <— (R) par;crc	12.9	3-138
SFS	Search For Synchronization	23.7	3-139
SOL R	(R) <— (R) . 2	0.38	3-140
SOL B	(B) <— (B) . 2	0.38	3-141
SOR R	(R) <— (R) / 2	1.5	3-142
SOR B	(B) <— (B) / 2	1.3	3-143
ST,	(CDB) <— (R)	7.3	3-144
ST R,lct	(LCT) <— (R)	0.75	3-145
ST B,lct	(LCT) <— (B)	0.75	3-146
ST R,B	((B)) <— (R)		3-147
ST R,*lct	((LCT):(B)) <— (R)	6.8	3-148
SUB R,lct	(R) <— (R) - (LCT)	0.75	3-149
SUB B,lct	(B) <— (B) - (LCT)	0.75	3-150
SUB R,B	(R) <— (R) - ((B))		3-151
SUB R,=imo	(R) <— (R) - IMO	0.75	3-152
SUB B,=imo	(B) <— (B) - IMO	0.75	3-153
TLU *lct	(W) <— ((LCT) + (R)) If SB=0, then (R) <— (W) (P) <— (P) + 1	6.0	
	else (W) <— (W) ^ X'7F' (W) <— (W) . 2 (P) <— (P) + 3 + (W)	7.5	3-154
TQE[b]	(W) <— (SOQ) - (EOQ)	0.75	3-155
TQF[b]	(W) <— (SOQ) - 15 - (EOQ)	1.1	3-156
WAIT	Suspend CCB execution	13.25	3-157
XOR R,lct	(R) <— (R) ⊕ (LCT)	0.75	3-158
XOR B,lct	(B) <— (B) ⊕ (LCT)	0.75	3-159
XOR R,B	(R) <— (R) ⊕ ((B))	0.75	3-160
XOR R,=imo	(R) <— (R) ⊕ IMO	0.75	3-161
XOR B,=imo	(B) <— (B) ⊕ IMO	0.75	3-162

[a] Extended LCTs are not channel related and must be handled on a controller basis.
[b] Stack operation pointers must be set prior to issuing these instructions.
[c] Exclusive OR.
[d] Plus Personality PROM, if applicable.

CCP Programming Instruction Card (Side 2)

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

CUT ALONG LINE

TITLE DPS 6 & LEVEL 6
NMLCP
COMMUNICATIONS HANDBOOK

ORDER NO. GA02-00

PART NO. 71017570-00

DATED MAY 1983

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

CUT ALONG LINE

FOLD ALONG LINE

FOLD ALONG LINE

Together, we can find the answers.

Honeywell

Honeywell Information Systems
U.S.A.: 200 Smith St., MS 486, Waltham, MA 02154
Canada: 155 Gordon Baker Rd., Willowdale, ON M2H 3N7
U.K.: Great West Rd., Brentford, Middlesex TW8 9DH **Italy**: 32 Via Pirelli, 20124 Milano
Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F. **Japan**: 2-2 Jinbou-Cho Kanda, Chiyoda-Ku Tokyo
Australia: 124 Walker St., North Sydney, N.S.W. 2060 **S.E. Asia**: Mandarin Plaza, Tsimshatsui East, H.K.

37120, 2C583, Printed in U.S.A.

GA02-00