January 1978

This manual provides the information required to develop and implement a software interface to the COMM IOP-DUP synchronous communications line controller.

COMM IOP-DUP
Programming Manual

Order No. AA-5670A-TC

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard. massachusetts

First Printing, January 1978

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

CONTENTS

CONTENTS (Cont.)

FIGURES

TABLES

PREFACE

MANUAL OBJECTIVES AND READER CLASS ASSUMPTIONS

The objective of this manual is to provide experienced programmers with the detailed information necessary to develop and implement a software interface to the COMM IOP-DUP synchronous communications line controller.

The level of technical detail presented in this manual assumes that the reader is proficient in the preparation of MACRO-11 assembly language programs and is versed in the use of the RSX-11M, RSX-11D or the IAS task builder to create an executable task image. In addition, the reader is assumed to be familiar with PDP-11 processor architecture and UNIBUS interfacing and to have an in-depth knowledge of PDP-11 programming techniques.

STRUCTURE OF MANUAL

This manual consists of four chapters and an appendix. Chapter 1 provides an operational overview of the COMM IOP-DUP synchronous communications line controller as well as particular details on system applications.

Chapter 2 presents the detailed information on command formats and functions and provides numerous coding examples pertinent to the operation of a COMM IOP-DUP/PDP-11 software interface.

Chapter 3 details system operation with respect to the supported line protocols, data transmission and reception, and line shutdown and reenabling.

Chapter 4 describes the procedure for loading the COMM IOP-DUP microprogram into the KMC11 microprocessor writeable control store for subsequent execution.

Appendix A describes and illustrates a suggested method for handling COMM IOP-DUP/PDP-11 interrupt dialogue.

CHAPTER 1

SYSTEM OVERVIEW

The Communications I/O Processor (COMM IOP) is a microprocessor-based, intelligent communications controller residing as a direct memory access device on the PDP-11 UNIBUS. In Digital Equipment Corporation systems, a direct memory access device is referred to as a nonprocessor request (NPR) device. COMM IOP operation is controlled by the KMC11-A microprocessor, which is equipped with a 1024-word writeable control store.

Through a series of microprograms executed from the KMC11-A writeable control store, COMM IOP can be configured to implement a family of intelligent communications line controllers. This family currently includes the synchronous communications line controller version operating with the DUP11 interface and the asynchronous communications line multiplexer version operating with the DZ11 interface. This manual presents the full range of information necessary for the preparation of user programs that most efficiently use the capabilities of the COMM IOP-DUP synchronous communications line controller.

1.1 SYSTEM CONCEPT

A COMM IOP-DUP configured as a synchronous communications line controller (Figure 1-1) consists of a single KMC11-A microprocessor that supports up to 16 DUP11 synchronous communications line interfaces. Each DUP11, in turn, controls a single communications line capable of message handling under DDCMP or under one of the bit stuffing protocols SDLC, ADCCP, HDLC, BDLC, X.25, and SNAP. In addition, each DUP11 can be programmed to operate in full- or half-duplex mode.

A COMM IOP-DUP microprogram is loaded over the UNIBUS at system startup time by a dedicated loader residing in, and executed by, the main CPU. A COMM IOP-DUP microprogram is loaded by performing a word-by-word transfer to the KMC11-A control store from a core image stored on the main CPU disk.

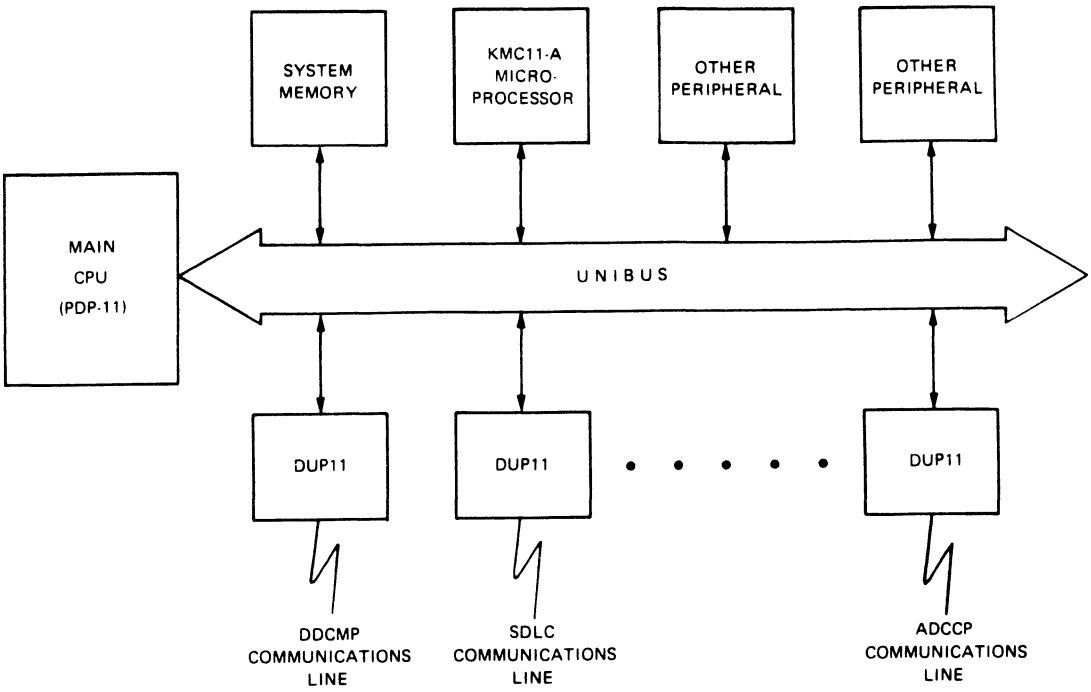

Figure 1-1 COMM IOP-DUP Synchronous Communications Line Controller Configuration

A COMM IOP-DUP loading routine that runs under RSX-11M, RSX-11D, and IAS, is described in Chapter 4 of this manual along with the necessary criteria for developing a loader to execute under a user designated operating system.

1.2 SYSTEM OPERATION

Operation of the COMM IOP-DUP microprogram is initiated and directed by a user-produced program residing in the PDP-11 (main CPU) memory space. A user program is defined as a device driver or an equivalent routine that interfaces to the COMM IOP-DUP. Communication between the user program and COMM IOP-DUP is provided by four control and status registers (CSRs), which are integral to the KMC11-A microprocessor. These four 16-bit registers are used for control input, status output, and data input and output. In general, the first two CSR words are used for control and status information and the remaining two words serve as an I/O data port.

The first two registers in this group have a fixed format and serve as the command header for the second two registers. The second two registers form a two-word data port for the exchange of unique control/status commands between COMM IOP-DUP and the user program. The contents of the data port are specified by an identification field in the command header. Other specific fields in the two-word command header control interrupt enabling, set up data transfers between the main CPU and the COMM IOP-DUP device, and identify the communications line of interest. The high byte of the first CSR is used to contain a special command issued by the user program for implementing microprocessor start, halt, and initialization. Detailed descriptions of each field in these four words are presented in Chapter 2.

A user program issues a command to COMM IOP-DUP by storing the command in the pertinent CSRs. COMM IOP-DUP then interprets the command and performs the specified actions. Similarly, COMM IOP-DUP issues a command to the user program by storing the command in the pertinent CSRs and notifying the user program that a command is available for retrieval and processing.

Message data received or transmitted by COMM IOP-DUP is written into or read from user program assigned buffers in main CPU memory. COMM IOP-DUP accesses these buffers through Non Processor Requests (NPR) to a UNIBUS address. A UNIBUS address is defined as an 18-bit address used by an NPR device to access a device on the UNIBUS or a location in main CPU memory.

1.2.1 Command Structure

The functions of the six COMM IOP-DUP control/status/data commands are described in the sections that follow.

1.2.1.1 **INITIALIZATION Command** - This command is used to clear all condition sensitive logic in the KMC11-A microprocessor and place the microprocessor in the Run state. This command must be issued by the user program once prior to starting the COMM IOP-DUP initialization procedure.

1.2.1.2 **BASE IN Command** - This command is used to initialize the DUP11 interfaces supported by the specific COMM IOP-DUP, and it is performed once, generally at startup time, for each supported DUP11. This command informs the COMM IOP-DUP of the CSR address for each DUP11 and assigns a communications line number for each interface.

1.2.1.3 **CONTROL IN Command** - This command defines the characteristics of the communications line driven by the pertinent DUP11. These characteristics include line state (enabled or disabled), protocol specification, half- or full-duplex operation, and CRC inhibit/enable. The user program must issue one CONTROL IN command for each supported DUP11.

1.2.1.4 **BUFFER ADDRESS IN Command** - The user program issues this control command to a COMM IOP-DUP to assign a new buffer descriptor list to the designated line. A buffer descriptor list is a sequential list of one or more three-word buffer descriptors in main CPU memory. Each buffer descriptor points to and describes a single user assigned buffer. The user program can assign a maximum of two receive and two transmit buffer descriptor lists to each communications line.

Each buffer descriptor contains a buffer pointer, a byte count, and a set of control fields pertinent to the DUP11. Upon completion of each transmit or receive operation, COMM IOP-DUP replaces the original byte count in the buffer descriptor with the actual number of bytes or characters transmitted or received. All buffers pointed to by a specific descriptor list are either transmit or receive buffers, as specified by the command header.

1.2.1.5 **BUFFER ADDRESS OUT Command** - COMM IOP-DUP issues this control command to the user program when the buffer assigned to a transmit or receive operation by a given buffer descriptor is terminated. Generally, a transmit or receive buffer is terminated when the buffer is full (descriptor byte count = zero). The first word in the data port for this command always contains the first 16 bits of UNIBUS address for the pertinent buffer descriptor with the two extension bits contained in the high-order byte of the second word in the port.

1.2.1.6 **CONTROL OUT Command** - COMM IOP-DUP issues this status command to the main CPU when it detects a transmission or receive error. A specific bit in the command header tells the main CPU, if relevant, whether the error occurred during data transmission or reception.

Where applicable, this command informs the user program of the UNIBUS address for the buffer descriptor, which, in turn, points to the specific buffer and the erroneous byte in the buffer. The command also includes a field containing a code that designates the nature of the detected error. If data is received on a synchronous communications line not having a buffer assigned, the CONTROL OUT buffer descriptor field will be indeterminate and the error field will contain the code for "no buffer assigned."

1.2.2 Data Transfer Operations

For the purposes of this system overview, the transmit and receive data command sequences described in this section are general and are meant to serve as background for the detailed presentations in the chapters that follow.

1.2.2.1 **Initialization Sequence** - After the COMM IOP-DUP microprogram is loaded, the first action taken by the user program is to issue an INITIALIZATION command, which performs a Master Clear on the KMC11 microprocessor and places the processor in the Run state. With this action complete, COMM IOP-DUP is ready to accept the first command from the user.

Following this initial command, the user program must issue one BASE IN command for each DUP11 supported by the COMM IOP-DUP. This command conveys the CSR address of the DUP11 associated with that line.

After all BASE IN commands are issued, the next step for the user program is to issue a CONTROL IN command for each communication line supported by a COMM IOP-DUP. This command establishes the various characteristics of the line and enables the line for subsequent transmission and reception.

1.2.2.2 **Synchronous Receive/Transmit Sequence** - Once the user program has initialized each DUP11 interface through BASE IN commands and issued a series of CONTROL IN commands to establish the characteristics of each active line, COMM IOP-DUP is ready to perform a receive or transmit data operation.

The configuration of the CONTROL IN command determines whether the line will be half- or full-duplex, and whether the data received or transmitted will be handled under DDCMP or one of the bit-stuffing

protocols. Through the CONTROL IN command, a user program can also assign a secondary station address. This feature is used in multidrop systems to designate the line address of a slave station drop. Also, a CONTROL IN command enables the receiver in the designated DUP11.

An actual reception or transmission is initiated when the user program issues a BUFFER ADDRESS IN command. COMM IOP-DUP requires a buffer descriptor list assignment through a BUFFER ADDRESS IN command in order to initiate a data transfer.

The buffer descriptor list is a sequential list of three word descriptors in main CPU memory with each descriptor pointing to and describing a single buffer. When a buffer descriptor list is assigned, it is designated for either reception or transmission. In addition to a buffer address and a byte count, each buffer descriptor contains control bits that provide for sync character transmission and for flagging the start and the end of transmit messages.

COMM IOP-DUP informs the user program of a normal data transfer completion by issuing a BUFFER ADDRESS OUT command. COMM IOP-DUP completes a normal data transfer operation for one of two reasons: the current buffer has been completed, or in the case of a receive buffer, an End-Of-Message has been detected. If a transmission or reception error is detected, COMM IOP-DUP informs the user program of the error by issuing a CONTROL OUT command containing the code designating the error condition.

1.3 SYSTEM APPLICATIONS

COMM IOP-DUP is designed to implement high-performance communication network systems for the user who does not have a sufficient number of lines to justify the additional cost of a large-scale front end. In effect, COMM IOP-DUP is a small, low cost, but extremely powerful front end and is ideal for implementing large, highly efficient message-switching systems at substantial cost savings over the more conventional approaches.

1.3.1 Protocol Support

A COMM IOP-DUP operating as a synchronous communications line controller can support multiple-buffered NPR (Non Processor Request) interfaces for up to 16 DUP11 devices. Each DUP11 handles one full- or half-duplex synchronous communications line. The maximum aggregate throughput for all communications lines in a COMM IOP-DUP system, including both input and output lines, is 19,200 characters per second. When operating half-duplex, the maximum aggregate throughput is 9600 characters per second. Data rates for 4, 8, and 16 lines operating either full- or half-duplex are as follows:

1. Data rate for 16 lines is 4800 bits per second.

2. Data rate for 8 lines is 9600 bits per second.

3. Data rate for 4 lines is 19,200 bits per second.

The speed of the fastest line in a given configuration determines the number of lines that can be supported by that configuration. This throughput rate assumes a UNIBUS band width of 500,000 Hz and relates to the speed of the KMC11-A only to the extent that the associated PDP-11 supplies sufficient transmit and receive buffers and responds

promptly to completion postings by the COMM IOP-DUP. The throughput rate for a COMM IOP-DUP has no relationship to the throughput rate for the associated PDP-11 software since this throughput rate depends on such factors as CPU model, memory type, buffering capability, and the overall efficiency of the specific software.

The COMM IOP-DUP supports DDCMP protocol or one of six bit-stuffing protocols, namely SDLC, ADCCP, HDLC, BDLC, X.25, and SNAP, and other similar protocols. In addition, the protocol assignment for a given line can be switched under user program control. In a COMM IOP-DUP-implemented synchronous communications network, user program responsibilities are minimal; they are limited mainly to command interpretation, protocol related functions such as half-duplex control, error recovery, and header control.

COMM IOP-DUP performs all modem control functions, with the exception of ring and carrier monitoring, and checks and initiates the sending of CRC characters during the respective transmit and receive data operations. Under DDCMP, COMM IOP-DUP performs the following time critical tasks, which in conventional installations tend to limit the number of DDCMP lines available for simultaneous servicing:

1. Identification of numbered versus unnumbered message headers to permit retrieval of the byte count from numbered headers for use in determining message length.

2. Automatic receiver resynchronization through analysis of the DDCMP quick sync (Q) bit.

3. Automatic receiver resynchronization upon detection of block check errors and header errors.

4. Recognition of slave station addresses in the multidrop line environment so that the main CPU is interrupted only for messages having the proper station address.

Under the bit-stuffing protocols, COMM IOP-DUP controls the flag character generation and detection, secondary station selection, frame check sequence generation and checking, and error and abort detection.

CHAPTER 2

SYSTEM PROGRAMMING

This chapter contains the detailed information necessary to support the development of main CPU user programs that most effectively employ the network communications resources provided by COMM IOP-DUP. This information includes the COMM IOP-DUP command structure, typical user program implementations, and detailed programming information on command functions and formats.

2.1 COMMAND STRUCTURE

As previously shown, COMM IOP-DUP is an NPR device residing on a PDP-11 UNIBUS. Communication between the main CPU-resident user program and COMM IOP-DUP is accomplished through a set of four 16-bit UNIBUS Control and Status Registers (CSRs). The eight bytes comprising these four registers are assigned the following addresses in the I/O page floating address space: 76xxx0, 76xxx1, 76xxx2, 76xxx3, 76xxx4, 76xxx5, 76xxx6, and 76xxx7, with the word addresses being the four even-numbered locations. All four UNIBUS CSRs are both byte and word addressable. Within the concept of floating UNIBUS addresses, the actual word and byte addresses are assigned at system configuration time. (See the KMC11 Programmers Manual, AA-5244B-TC.)

In this explanatory narrative, the eight byte addresses are designated BSEL0 through BSEL7 and the four word addresses, SEL0, SEL2, SEL4, and SEL6. The relationship of byte and word addresses for COMM IOP-DUP UNIBUS CSRs, based on these designations, are summarized in Figure 2-1. Figure 2-1 also illustrates the basic COMM IOP-DUP input/output command format for the KMC11 CSRs along with pertinent command ID codes.

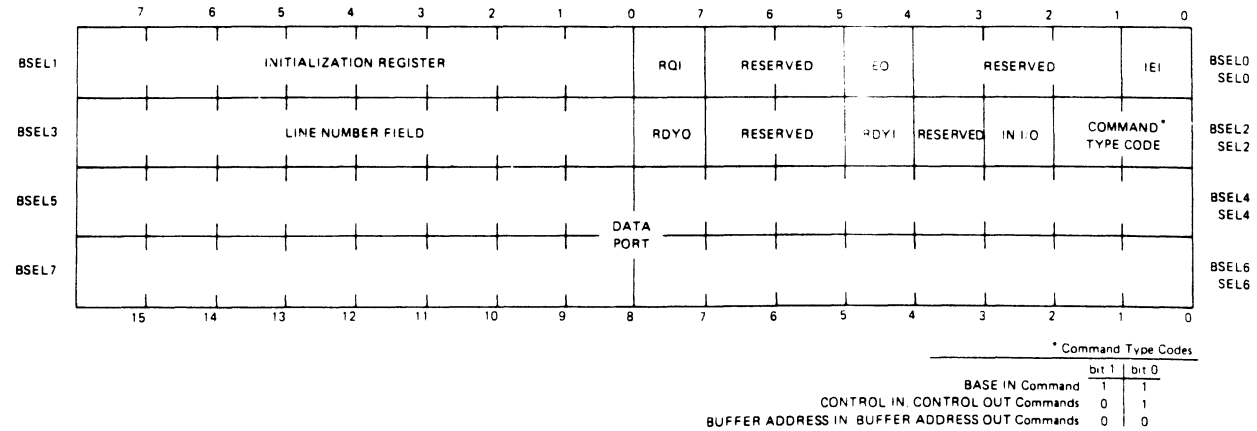

Figure 2-1 COMM IOP-DUP CSR Symbolic Addresses and Format

These address references, as designated in Figure 2-1, are the basis for CSR address identification in the following detailed descriptions of the COMM IOP-DUP commands.

Since COMM IOP-DUP is basically an input-output device, it follows that the command set for this device can be categorized as input commands and output commands. As opposed to received and transmitted data, input commands are commands issued to COMM IOP-DUP by the main CPU; output commands are those issued to the main CPU by COMM IOP-DUP. The structure and format of COMM IOP-DUP input and output commands are described in Sections 2.2 and 2.3, respectively.

2.2 INPUT COMMANDS

As previously described, COMM IOP-DUP executes four forms of input commands. These commands are listed below in the usual order of user program issuance:

1. INITIALIZATION
2. BASE IN
3. CONTROL IN
4. BUFFER ADDRESS IN

The format and field descriptions for each command are detailed in the following paragraphs. Some typical examples of PDP-11 instructions and instruction sequences are included to demonstrate the user-program command-issuing process. These examples are presented for explanation only and do not imply a single method of implementation.

2.2.1 INITIALIZATION Command

The INITIALIZATION command (Figure 2-2) is the first command issued by a user program at startup time to initialize the KMC11-A microprocessor and place the unit in the Run state.

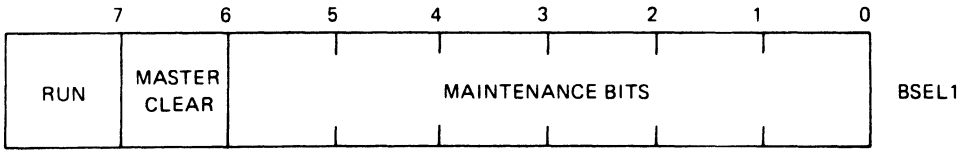

Figure 2-2 INITIALIZATION Command Format

Initializing the KMC11 microprocessor by the user program is done in two steps. First the Master Clear bit is set followed by setting of the Run bit.

After the Run bit is set, the user program must wait for 1 μs before accessing one of the command headers BSEL0 or BSEL2 since part of the COMM IOP-DUP initialization procedure involves clearing BSEL0 and BSEL2. The recommended method for setting the Master Clear and Run bits, and at the same time implementing the required delay, is to write a nonzero value into BSEL2 and wait for COMM IOP-DUP to clear the byte before proceeding. For example:

```
        MOV     #40000 , SEL0           ; Set Master Clear bit
        MOV     #377   , BSEL2          ; Write nonzero value in BSEL2
        MOV     #100000, SEL0           ; Set Run bit
A:      TSTB    BSEL2                   ; Cleared yet?
        BNE      A                      ; No
        BR       B                      ; Yes, exit to B
```

NOTE

Since the Master Clear bit is not self-clearing, a Move instead of a bit set instruction is required to clear the Master Clear bit and set the Run bit.

These actions set the Run bit placing COMM IOP-DUP in the operational state. At this point, the user program can begin setting up COMM IOP-DUP for subsequent operations. Note in Figure 2-2 that SEL0 bits 8 through 13 are designated maintenance bits. These bits are used by maintenance and diagnostic routines and are not used during normal COMM IOP-DUP operation and should never be set by the user program. (Refer to Chapter 4 of this manual and the KMC11 Programmers Manual, AA-5244B-TC.)

2.2.2 BASE IN Command

Figure 2-3 illustrates the format for the BASE IN command, which performs the initialization function for the DUP11 interfaces operating under COMM IOP-DUP. One BASE IN command must be issued for each DUP11 supported by COMM IOP-DUP.

Bit 7 of BSEL0, RQI (Request In), is set by the user program to request use of BSEL3, SEL4, and SEL6 for the transfer of data to COMM IOP-DUP. Bit 4 of BSEL2, RDYI (Ready In), is set by COMM IOP-DUP in response to the user program setting of RQI and informs the user program that data can be transferred into BSEL3, SEL4, and SEL6. After BSEL3, SEL4, and SEL6 are set up, the user program clears RQI, sets the command ID code and clears RDYI. Bit 0, IEI (Interrupt Enable Input), is set by the user program to permit COMM IOP-DUP to interrupt the main CPU when the data port (SEL4 and SEL6) is available (RDYI set). When main CPU interrupts are enabled, COMM IOP-DUP is assigned the floating vectors xx0 and xx4; xx0 is the input interrupt vector (RDYI) and xx4 is for completion interrupts (RDYO). BSEL3 comprises the line number field, which serves to identify the communications lines in a COMM IOP-DUP configuration. Bits 0 and 1 of BSEL2 (Figure 2-3) contain the 2-bit code identifying a BASE IN command, with the associated control bits for this command located in both BSEL0 and BSEL2.

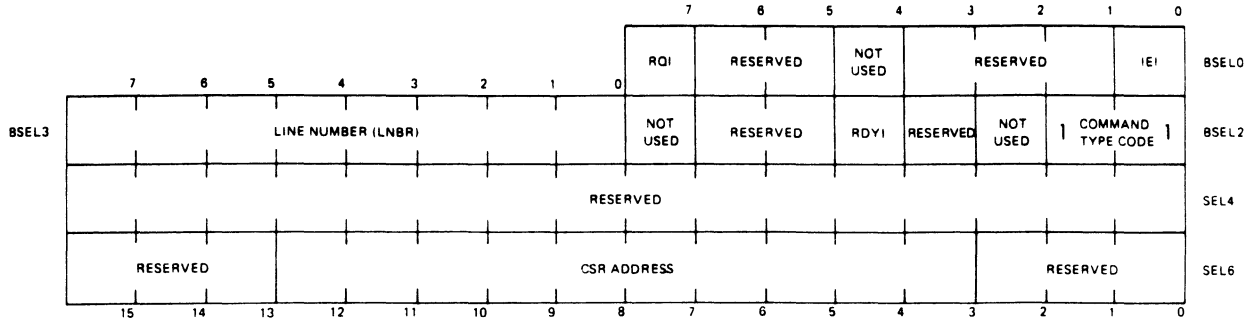

Figure 2-3 BASE IN Command Format

One BASE IN command is issued for each DUP11 supported by a COMM IOP-DUP, and the line number field in each BASE IN command issued contains an octal number in the range 0 to 17. These line numbers can be assigned to the communications lines in any order.

Bits 3 through 12 of SEL6 contain the corresponding bits of the CSR address for each DUP11 being initialized. Since bits 0, 1, and 2 of the 18-bit CSR address are always zeros and bits 13, 14, 15, 16, and 17 are always ones, only bits 3 through 12 are necessary to specify that address. This value is stored in bit positions 3 through 12 of SEL6 for the pertinent BASE IN command. For example, if a given DUP11 CSR address is octal 760110, SEL6 of the BASE IN for that device would contain the octal value 110. Note that bits 0 through 2 and bits 13 through 15 of SEL6 should always be zero.

2.2.2.1 **Issuing a BASE IN Command** - All input commands are issued by a user program in two successive steps. In general, the first step involves a request for permission to issue an input command and a response by COMM IOP-DUP that it is ready to accept the command. Although the programming sequences for the first step are described for the BASE IN command, they also apply to the CONTROL IN and BUFFER ADDRESS IN commands.

The sequence for the second step involves completing the command by loading BSEL3, SEL4, and SEL6 with the data appropriate to each command. This sequence is therefore different for each command.

The initial task to be performed by the user program is to set the RQI bit, and if necessary the IEI bit. With the RQI bit set, the user program must then wait for COMM IOP-DUP to set RDYI. Generally this procedure involves the following steps:

1. The user program sets RQI and then waits for COMM IOP-DUP to set RDYI. This wait can be implemented through a delay loop or the user program can wait for an interrupt if interrupts are enabled.

2. When RDYI is set, the user program clears RQI if a single command is involved; if multiple commands are being issued, RQI can be left set and the next step performed. In this case RQI would be cleared just prior to completing the next step for the last command issued.

3. The user program sets up BSEL3, SEL4, and SEL6.

4. Set up the command type code in BSEL2 (BSEL2 bits 0 and 1 = 1) and clear RDYI to inform COMM IOP-DUP that the data port can be read.

NOTE

The command type code may be set at the same time RDYI is cleared by using a MOVB instruction.

Upon completion of processing for a given command, COMM IOP-DUP clears all bits in BSEL2.

NOTE

Since COMM IOP-DUP does not clear BSEL3, SEL4, or SEL6, the user must ensure that these registers are cleared by executing appropriate clear instructions prior to issuing a command or by issuing the command with MOV or MOVB instructions.

How this procedure is actually programmed depends on whether the state of the IEI bit is set to enable interrupts. The latency between the user program setting the RQI bit and COMM IOP-DUP responding by setting RDYI can range from a minimum of 3 μs to a maximum of 250 μs providing an output completion is not pending. When using the IEI bit, the user has three alternatives:

1. Set the IEI bit to enable interrupts. As a consequence, COMM IOP-DUP interrupts the main CPU when it sets the RDYI bit. The PDP-11 instruction implementing this alternative can have the form

```
BISB    #201,BSEL0      ;set RQI and IEI
```

When interrupted the user program can proceed directly to load BSEL3, SEL4, and SEL6 with the appropriate data and to set the BASE IN ID code in BSEL2 bits 0 and 1.

NOTE

COMM IOP-DUP will not set RDYI if RDYO is set.

2. Leave the IEI bit cleared and check the state of the RDYI bit by setting a timer and performing a test or performing a continuous test loop. The form of the bit test sequence based on a timer is as follows:

```
C:      TSTB    BSEL2           ;RDYO set?
        BMI     G               ;If RDYO set, exit to process
                                ;completion.  This assumes
                                ;that IEO is not set otherwise
                                ;the setting of RDYO would
                                ;generate an interrupt
                                ;(Section 2.3.1).
```

```
        BITB    #20,BSEL2
        BNE     A               ;RDYI set, exit to load
                                ;command routine
        BR      B               ;RDYI not set, reset timer and
                                ;resume prior task at B.  When
                                ;timer goes off reenter at C.
```

If a bit test loop is required, the sequence form is:

```
E:      TSTB    BSEL2           ;RDYO set?
        BMI     F               ;If RDYO set, exit to process
                                ;completion.
                                ;This assumes that IEO is
                                ;cleared otherwise the setting
                                ;of RDYO would generate an
                                ;interrupt (Section 2.3.1).
        BITB    #20,BSEL2       ;Test RDYI
        BEQ     E               ;RDYI not set, branch to E and
                                ;test again.
        BR      D               ;RDYI set, exit to complete
                                ;command processing
```

3. Using this alternative, the user program clears IEI (if set by the prior command), sets RQI, and then performs the housekeeping associated with issuing the current command. With the housekeeping done, the user program checks RDYI. If RDYI is set, the user program completes the command issuing process. If not set, the user program sets IEI and resumes a prior task while waiting for an interrupt on RDYI set.

   The advantage of this alternative is that interrupt overhead is substantially reduced since COMM IOP-DUP usually sets RDYI within a few microseconds after the user program sets RQI. The form of the instruction sequence for this approach is as follows:

```
        BICB    #1,BSEL0        ;Clear IEI
        BISB    #200,BSEL0      ;Set RQI
                  .
                  .
              Do housekeeping
        BITB    #20,BSEL2       ;Test RDYI
        BNE     H               ;Exit to complete command
                                ;processing
        BISB    #1,BSEL0        ;Set IEI
```

NOTE

If RDYI is already set at the point that the user program sets IEI, an interrupt will be generated. In addition, with IEI set, only one interrupt is generated for each setting of RDYI. However, if the user program clears IEI, an interrupt may still be generated if COMM IOP-DUP sets RDYI within 3 µs after it clears IEI. This situation can be avoided by always clearing IEI while RQI and RDYI are in the cleared state.

A suggested sequence for processing input interrupts is described in Appendix A.

2.2.2.2 **Completing a BASE IN Command** - The instruction sequence to initialize the first DUP11 is:

```
        BICB    #200,BSEL0      ;Clear RQI
        CLR     R0              ;
        MOVB    R0,BSEL3        ;Set line number to line zero
        MOV     CSR(R0),SEL6    ;Set CSR for line zero
        MOVB    #3,BSEL2        ;Set BASE IN command ID and clear
                                ;RDYI
```

For the second, the sequence is as follows:

```
        BIC     #200,BSEL0      ;Clear RQI
        ADD     #2,R0           ;Perform a word increment of
                                ;line number
        MOVB    R0,BSEL3        ;Set line number
        ASRB    BSEL3           ;convert word index into line number
                                ;by a right shift
        MOV     CSR(R0),SEL6    ;Set CSR for line
        MOVB    #3,BSEL2        ;Set BASE IN command ID and clear
                                ;RDYI
             .
             .
             .
```

and so on until all supported DUP11 units are initialized.

NOTE

In the coding example above, CSR is a table containing the DUP11 CSR addresses in the required format.

2.2.3 CONTROL IN Command

This command establishes the characteristics of each communications line supported by a COMM IOP-DUP. Figure 2-4 shows the format for a DUP11 CONTROL IN command. One CONTROL IN command must be issued for each DUP11 in a given configuration.

2.2.3.1 **CONTROL IN Command Format** - With the exception of the command identification code (bit 0 = 1; bit 1 = 0), the format for the DUP11 CONTROL IN command header bytes (BSEL0 and BSEL2) is the same as for the BASE IN command (Figure 2-3). For a detailed description of the functions performed by the BSEL0 and BSEL2 RQI, IEI, and RDYI bits, refer to Section 2.2.2.

For each CONTROL IN command issued, the line number field (BSEL3) contains the line number assigned to the communications line of interest.

As shown in Figure 2-4, BSEL5 along with bits 2, 3, and 6 of BSEL7 are reserved for future expansion of COMM IOP-DUP capability and should always be zeros. Figure 2-4 also shows that a CONTROL IN command is formed by BSEL0, BSEL2, BSEL3, BSEL4, BSEL6, and BSEL7. In the description of fields contained in BSEL6 and BSEL7, clarity dictates that BSEL7 be described first starting at the high-order bit position.

The Digital Data Communications Message Protocol (DDCMP) bit (Bit 7 of BSEL7) informs COMM IOP-DUP which message protocol will be employed in the reception and transmission of messages over the pertinent communications line. In the one state, this bit designates that the pertinent communications line is to operate under DDCMP. In the zero state, this bit specifies that a bit-stuffing protocol such as SDLC, ADCCP, HDLC, BDLC, X.25, or SNAP will be used on the pertinent line.

Half- or full-duplex operation of the pertinent communications line is controlled by bit 5 of BSEL7. This bit is set when the pertinent communications line is a half-duplex circuit. In this mode, data reception is inhibited whenever data is being transmitted over the line. Conversely, this bit is cleared when the pertinent communications line is a full-duplex circuit.

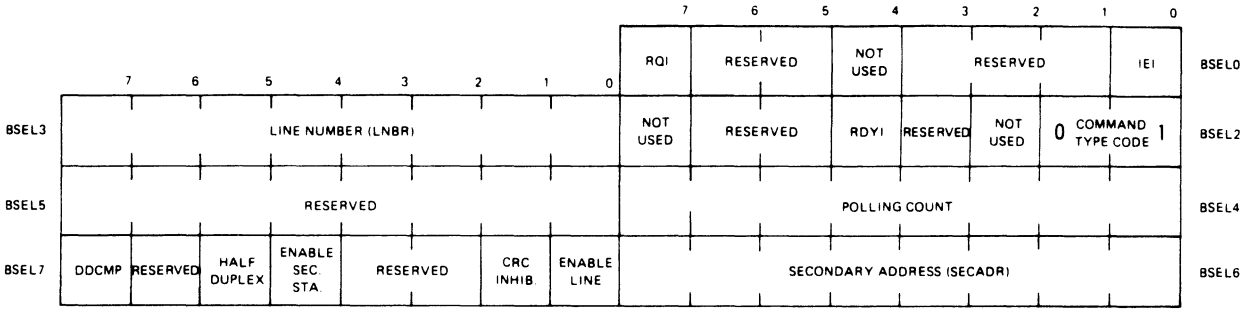

Figure 2-4 CONTROL IN Command Format

If a secondary station address field is stored in BSEL6, bit 4 of BSEL7 must be set. This bit serves to flag COMM IOP-DUP to process BSEL6. If this bit is cleared, BSEL6 is ignored.

Under bit stuffing protocols, when the CRC inhibit bit (bit 1 of BSEL7) is set, CRC calculations for the message data being transmitted or received are not performed by COMM IOP-DUP. Conversely, when this bit is cleared, CRC calculations are performed. The CRC inhibit bit must be cleared to zero under DDCMP operations.

Each communications line controlled by a DUP11 and directed by COMM IOP-DUP can be selectively enabled or disabled at startup time or at some subsequent time through the CONTROL IN command enable line bit (bit 0 of BSEL7). When set, this bit enables the line by forcing the setting of the DTR bit (Data Terminal Ready) at the associated modem and enabling the DUP11 receiver. When this bit is cleared, DTR is cleared and the DUP11 receiver is disabled.

The secondary address field (BSEL6) permits the user program to designate the pertinent DUP11 as a slave station on a multidrop line. In the multidrop environment, the COMM IOP-DUP automatically compares the secondary address field assigned to the pertinent line with the station address field in the protocol header of each message received. COMM IOP-DUP accepts only messages having the secondary address assigned to the pertinent communications line.

BSEL4 contains a count that designates the time interval at which COMM IOP-DUP will check the associated DUP11 for a transmit or receive done. The COMM IOP-DUP polling interval is specified in 50 µs units so that a count of zero in BSEL4 would result in a 50 µs polling interval, a count of one produces a 100 µs polling interval, a count of 2 a 150 µs polling interval and so on.

The required polling interval for each supported line is determined by the speed of the pertinent line, the number of active lines supported by a COMM IOP-DUP and the band width of the UNIBUS. In general, the higher the line data rate, the more lines active; and the lower the UNIBUS band width, the faster the polling rate (lower polling count) should be. For example, for a COMM IOP-DUP supporting eight active full-duplex lines, each with a line data rate of 9600 bps, the polling count would be set to one (100 µs). However, with only one active line having the same data rate, a polling count of three (200 µs) could be used. The general relationship between line speed and polling count is shown in Table 2-1.

Table 2-1
Relationship between Line Data Rate and Polling Count

Line Data Rate (Full-Duplex)	Polling Count
19,200 bps	0
9600 bps	1 to 3
4800 bps	2 to 6
2400 bps	4 to 12

When frequent overruns or underruns are encountered on a given line, the polling count for that line should be decreased. In general, a polling interval that is shorter than required has no detrimental effect on COMM IOP-DUP operation. However, it will result in superfluous UNIBUS cycles that can degrade system throughput.

2.2.3.2 **Issuing a CONTROL IN Command** - The first steps to be taken by a user program when issuing a CONTROL IN command are to set the RQI bit in BSEL0 and to test the RDYI bit for COMM IOP-DUP response. The procedure followed is exactly the same as that used for issuing a BASE IN command (Section 2.2.2.1). A CONTROL IN command must not be issued to a DUP11 that has not been initialized by a prior BASE IN command.

2.2.3.3 **Completing a CONTROL IN Command** - Upon detecting RDYI set, the user program can complete the issuing of a CONTROL IN command. As previously stated, each DUP11 supports one synchronous communications line and one CONTROL IN command must be issued at startup time for each DUP11 supported by a COMM IOP-DUP. The following example describes the form of the user program-executed PDP-11 instructions required to transfer the data comprising a CONTROL IN command to the COMM IOP-DUP CSR registers BSEL3, BSEL4, BSEL6, and BSEL7:

1. The line number field (LNBR) is set in BSEL3;
2. bit 7 of BSEL7 is set to one for DDCMP operation;
3. bit 5 of BSEL7 is cleared to zero to designate a full-duplex line;
4. bit 4 of BSEL7 is set to one to indicate a secondary address assignment;

5. bit 1 of BSEL7 is cleared to zero to enable CRC calculation;

6. set secondary address in BSEL6 (consider the secondary address to be octal 10);

7. bit 0 of BSEL7 is set to one to enable the pertinent line; and

8. the polling count (BSEL4) is set to 3.

The instructions implementing this example can take the following form:

```
MOVB    #LNBR1,BSEL3    ;Set line number
MOV     #110410,SEL6    ;Set BSEL6 for secondary address
                        ;and BSEL7 for line characteristics
MOVB    #3,BSEL4        ;Set polling count to 3
MOVB    #1,BSEL2        ;Set CONTROL IN code and clear
                        ;RDYI
```

In response to a given CONTROL IN command with the Enable Line bit set to one, COMM IOP-DUP will assert DTR and enable the receiver. If the Enable Line bit in that command is cleared to zero, DTR will be cleared and the receiver disabled. In addition, the DUP11 designated by the line number field (BSEL3) will be assigned the line characteristics specified by BSEL7 (Figure 2-4).

2.2.4 BUFFER ADDRESS IN Command

This command provides the user program with the mechanism for assigning, deassigning, and reassigning transmit and receive buffers. The format for this command is presented in Figure 2-5. Also, Figure 2-5 shows that, in addition to the command identity bits in bits 0 and 1 of BSEL2 (code=0,0), BSEL2 also contains the control bit IN I/O in bit position two. The function of this bit is to designate the assigned buffers as either transmit buffers or receive buffers. When set, the buffers for the designated line are assigned as receive buffers, and when cleared the pertinent buffers are assigned as transmit buffers. The user program must set the appropriate state of the IN I/O bit at the same time the command type code is set (Section 2.2.2.1). Aside from the value of the identity bits and the presence of the IN I/O bit, the format for BSEL0 and BSEL2 is the same as that for BASE IN and CONTROL IN commands. Consequently, the user program/COMM IOP-DUP processing sequence involving the RQI, IEI, and RDYI bits is exactly the same for the BUFFER ADDRESS IN command as for the BASE IN and CONTROL IN commands (Sections 2.2.2 and 2.2.3).

As in the CONTROL IN command, the line number field (BSEL3) specifies the DUP11 communications line to which a BUFFER ADDRESS IN command is to apply.

In COMM IOP-DUP, buffers are assigned to a communications line through a buffer descriptor list. Note in Figure 2-5 that SEL4 and bits 6 and 7 of BSEL7 contain an 18-bit UNIBUS address that is the starting address of the buffer descriptor list assigned to the pertinent communications line.

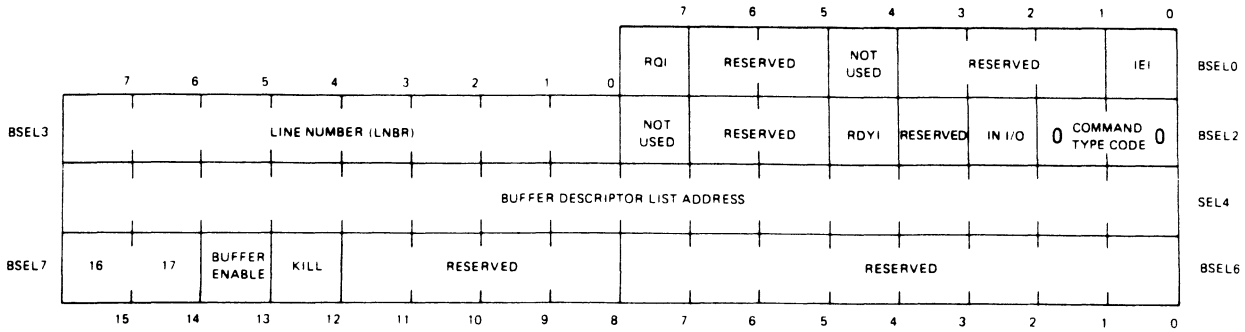

Figure 2-5 BUFFER ADDRESS IN Command Format

A buffer descriptor list is a sequential list of three word blocks in main CPU memory space. Each 3-word block points to and describes the boundaries of a single buffer also in main CPU memory space. The length of a buffer descriptor list is user-defined and a maximum of two transmit and two receive lists can be assigned to each COMM IOP-DUP supported communications line. Finally, by definition, the starting address of a buffer descriptor list must be word aligned, that is on an even address boundary. The format and function of the buffer descriptor list are described in Section 3.1.

Buffer descriptor lists are deassigned through use of the Kill bit (bit 4 of BSEL7) and also reassigned when this bit is used in conjunction with the Buffer Enable bit (bit 5 of BSEL7). For the relevant data transfer (receive or transmit as determined by the state of the IN I/O bit) issuing a BUFFER ADDRESS IN command with the Kill bit set and the Buffer Enable bit cleared deassigns all buffer descriptor lists currently assigned to the communications line designated by the line number field.

When the user program issues a BUFFER ADDRESS IN command with both the Kill and Buffer Enable bits set, the current buffer descriptor receive or transmit list or lists are deassigned, and a new buffer descriptor list address specified by SEL4 and bits 6 and 7 of BSEL7 is assigned to the pertinent line. When the Kill bit is cleared, COMM IOP-DUP ignores the Buffer Enable bit and treats the command as a normal buffer descriptor list address assignment. Note that setting the Kill bit during a receive operation also causes the pertinent DUP11 to be placed in the search sync mode, if operating under DDCMP, or in the flag search mode if operating under one of the bit stuffing protocols. Under DDCMP, when an active transmission is stopped, the pertinent line is brought back to a mark hold condition. Under a bit stuffing protocol, when an active transmission is stopped, an abort character is transmitted and the pertinent line is brought back to a mark-hold condition. COMM IOP-DUP signals the completion of a particular kill operation by issuing a CONTROL OUT command (Section 2.3.3).

2.2.4.1 **Issuing a BUFFER ADDRESS IN Command** - The first steps to be taken by a user program when issuing a BUFFER ADDRESS IN command, are to set the RQI bit in BSEL0 and to test the RDYI bit for COMM IOP-DUP response. The procedure followed is the same as that used for issuing a BASE IN command (Section 2.2.2.1) except that the state of the IN

I/O bit, in BSEL2 must be established when the command type code is set. This action defines the buffers described by the assigned buffer descriptor list as either receive or transmit buffers.

The various methods of using the IEI bit and the RQI-RDYI request/response cycle described in Section 2.2.2.1 apply directly to the processes of issuing a BUFFER ADDRESS IN command.

2.2.4.2 **Completing a BUFFER ADDRESS IN Command** - A user program completes this command upon detecting RDYI set. Various examples of implementing PDP-11 instruction sequences are described below to demonstrate the variations of this command.

An example of an instruction sequence to assign a buffer descriptor list address to a DUP11 takes the following form:

```
        MOVB    #LNBR00,BSEL3   ;Set line number
        MOV     #100004,SEL4    ;Set 16 bits of UNIBUS address
        MOVB    #200,BSEL7      ;Set the state of the two most
                                ;significant bits of UNIBUS
                                ;address (18-bit
                                ;address = 500004)
        MOVB    #4,BSEL2        ;Assign a receive buffer and
                                ;clear RDYI
or
        CLRB    BSEL2           ;Assign a transmit buffer and
                                ;clear RDYI
```

An example of an instruction sequence to kill all current buffers assigned to a given line and assign a new buffer descriptor list to that line, takes the following form:

```
        MOVB    #LNBRnn,BSEL3   ;Set line number
        MOV     #105000,SEL4    ;Set least significant 16 bits of
                                ;UNIBUS address
        MOV     #160,BSEL7      ;Set the state of the two most
                                ;significant UNIBUS address
                                ;bits plus Kill and
                                ;buffer enable bits (18-bit
                                ;address = 305000)
        MOVB    #4,BSEL2        ;Kill and assign a receive buffer and
                                ;clear RDYI
or
        CLRB    BSEL2           ;Kill and assign a transmit buffer and
                                ;clear RDYI
```

An example of an instruction sequence to kill all buffers assigned to a given communications line without reassigning a new buffer descriptor list takes the following form:

```
        MOVB    #LNBRnn,BSEL3   ;Set line number
        MOVB    #20,BSEL7       ;Set Kill bit
        MOVB    #4,BSEL2        ;Kill all receive buffers and
                                ;clear RDYI
or
        CLRB    BSEL2           ;Kill all transmit buffers and
                                ;clear RDYI
```

2.3 OUTPUT COMMANDS

Output commands provide the vehicle whereby COMM IOP-DUP communicates with the main CPU. COMM IOP-DUP uses the output commands to convey two categories of information:

1. Information pertinent to the normal completion of data transfers;

2. information concerning the forced completion of data transfers due to detection of an error condition.

The BUFFER ADDRESS OUT command is used to post normal completions to the user program and the CONTROL OUT command for completions posted due to the detection of an error condition.

Note that the IEO (Interrupt Enable Out) bit, when set, will cause COMM IOP-DUP to interrupt the main CPU each time an output command is ready for retrieval by the user program. When cleared, COMM IOP-DUP does not interrupt, making the user program responsible for recognizing that an output command is ready for retrieval.

2.3.1 Output Command Structures

COMM IOP-DUP issues output commands in two steps. First, the data pertinent to the command being issued is stored in BSEL3, SEL4, and SEL6 (Figure 2-1). Once this data storage is complete, COMM IOP-DUP sets the RDYO and identity bits in BSEL2 and generates an interrupt through vector xx4 if the IEO bit is set. If the command issued is a BUFFER ADDRESS OUT, the IN I/O bit is set to one to indicate that the completion posted involves a receive data operation or cleared to zero to designate a completion posting for a transmit data operation. Generally, processing an output command involves the following steps:

1. The user program checks for RDYO set. This can be done through periodic checking or by waiting for an interrupt, assuming that interrupts are enabled.

2. When RDYO is detected as set, the user program would check BSEL2, bits 0, 1, and 2 to determine type of completion (receive or transmit, normal or error), then read BSEL3, SEL4, and SEL6, and process as necessary.

3. Upon reading the data port (SEL4 and SEL6), the user program clears RDYO to inform COMM IOP-DUP of port availability. This can be done with the PDP-11 instruction:

```
CLRB    BSEL2
```

NOTE

If RDYO is already set at the time that the user program sets IEO, an interrupt will be generated. In addition, with IEO set, only one interrupt is generated for each setting of RDYO. However, when the user program clears IEO, an interrupt can still be generated if COMM IOP-DUP sets RDYO within 3 µs of clearing IEO.

A user program, designed to operate in a noninterrupt mode, must be set up to periodically test the state of the RDYO bit. A PDP-11 instruction sequence to periodically test the RDYO bit, and when set check the ID bits, can take the following form:

```
A:      TSTB    BSEL2           ;Test RDYO bit.
        BPL     B               ;Cleared, exit to perform
                                ;user task and reenter
                                ;at A during next period.
        BITB    #1,BSEL2        ;Test ID bits.
        BEQ     C               ;Exit to retrieve
                                ;BUFFER ADDRESS OUT command
                                ;and process.
        BR      D               ;Exit to retrieve CONTROL
                                ;OUT command and process.
```

With interrupts enabled (IEO set) the first user program action upon receiving the interrupt is to test the ID bits to determine the command type to be processed. If the command is ascertained to be a BUFFER ADDRESS OUT command, the user program can determine whether the completion being posted involves a transmit or receive data operation by checking the IN I/O bit. For example,

```
C:      BITB    #4,BSEL2        ;Check IN I/O bit
        BEQ     F               ;Exit to perform transmit
                                ;operation
        BR      G               ;Exit to perform receive
                                ;operation
```

Upon completing the retrieval of the pertinent command, the user program must clear RDYO to inform COMM IOP-DUP that the retrieval is complete. For example:

```
        CLRB    BSEL2           ;Clear RDYO
```

Upon completion of an output command, COMM IOP-DUP will check for an input command (RQI = 1) and service that command before issuing the next output command, if any are pending. On this basis, the user program can set RQI before clearing RDYO to guarantee that the next command serviced will be an input command. (A suggested sequence for processing interrupts is described in Appendix A.)

NOTE

COMM IOP-DUP will not respond to a user program request to input a command (RQI set to 1 in BSEL0) if an output completion is pending (RDYO = 1). In addition, the states of RDYO and RDYI are mutually exclusive. Therefore, these bits are never set simultaneously.

2.3.2 BUFFER ADDRESS OUT Command

This command is used to post the normal completion of data-transfer operations to the user program. Normal completions are posted when a message is completed or when the current buffer is filled. All other completions are posted by a CONTROL OUT command as error completions.

2.3.2.1 **BUFFER ADDRESS OUT Format** - Figure 2-6 illustrates the format for this command. Since the control bits in BSEL0 and BSEL2 have been described in detail previously, the narrative that follows describes the fields in BSEL3, SEL4, and SEL6. It should be reiterated that bit 2 of BSEL2 (Figure 2-6), the IN I/O bit, serves to specify to the user program whether the pertinent BUFFER ADDRESS OUT command applies to the completion of a receive data operation (IN I/O = 1) or a transmit data operation (IN I/O = 0).

The line number field in BSEL3 identifies for the user program the communications line to which the information in the remaining command fields apply. Bit 7 of the line number field (BSEL3, Figure 2-6), when set to one indicates that COMM IOP-DUP has overrun the internal completion stack. At this point one or more completions have been lost. This situation results from the user program not retrieving completions at a fast enough rate.

SEL4 and bits 6 and 7 of BSEL7 form the 18-bit UNIBUS address for the buffer descriptor describing the completed buffer. The format and function of the buffer descriptor is described in Section 3.1.

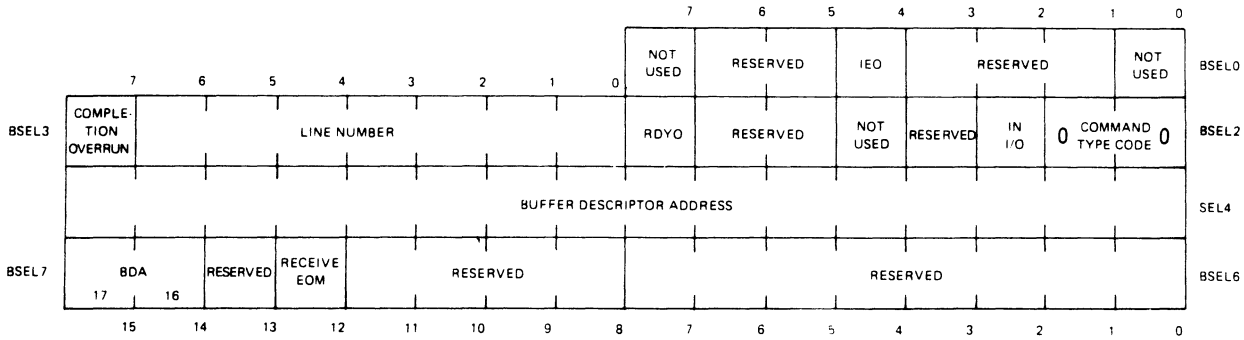

Figure 2-6 BUFFER ADDRESS OUT Command Format

Bit 4 of BSEL7 specifies, for receive data operations, whether the completion resulted from an end-of-message (RECEIVE EOM) signal being detected or as a consequence of buffer completion. Bit 4 = 1 designates that a RECEIVE EOM signal has been detected and bit 4 = 0 specifies that the current buffer is filled without detection of a RECEIVE EOM signal (message not complete). If a message exactly fills a buffer, the RECEIVE EOM bit will be set to one.

2.3.3 CONTROL OUT Command

This command is used by COMM IOP-DUP to inform the user program of the nature of an error condition detected during the current receive or transmit data operation. In addition to identifying the line on which the error occurred, this command provides the address of the buffer descriptor pointing to the buffer that was active when the error occurred. The byte count field in the descriptor reflects the number of characters processed in the pertinent buffer prior to occurrence of the error.

2.3.3.1 **CONTROL OUT Command Format** - Figure 2-7 illustrates the format for this command. Since the control bits in BSEL0 and BSEL2 have been described in detail in Section 2.3.1, the narrative that follows concerns the fields in BSEL3, SEL4, BSEL6, and BSEL7. Note that bit 2 of BSEL2 (compare Figures 2-6 and 2-7) is used in this command to distinguish between transmit and receive errors.

The line number field in BSEL3 identifies for the user program the line number to which information in the remaining command fields apply. Bit 7 of the line number field (BSEL3, see Figure 2-7), when set to one, indicates that COMM IOP-DUP has overrun the internal completion stack. At this point one or more completions have been lost. This situation results from the user program not retrieving completions at a fast enough rate.

SEL4 and bits 6 and 7 of BSEL7 form the buffer descriptor address, which in turn describes the buffer in which the error condition occurred. The format and function of the buffer descriptor is described in Section 3.1.

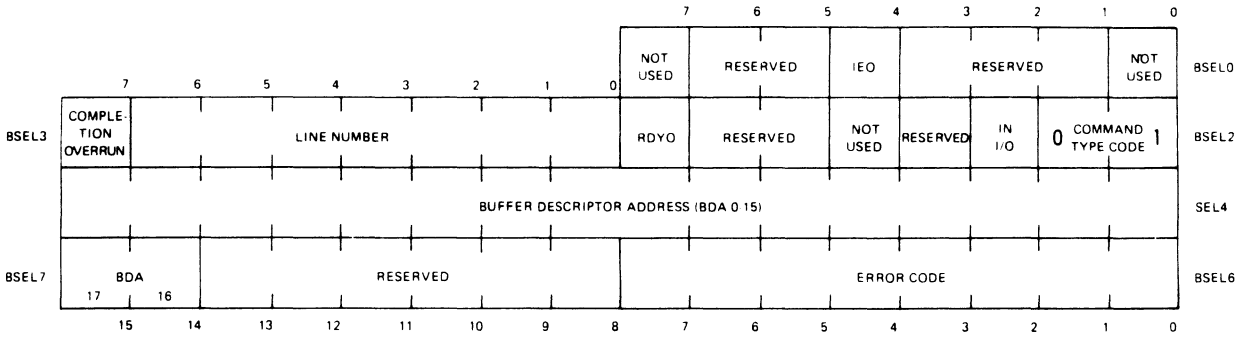

Figure 2-7 CONTROL OUT Command Format

BSEL6 contains a code defining the specific error which initiated the issuing of the associated CONTROL OUT command. Table 2-2 lists the error codes for the COMM IOP-DUP. When the Kill Complete condition is detected, all transmit or receive buffers currently assigned to the communications line designated by the line number field in the command are flushed. To resume operation on that line, the user must assign new buffer descriptor lists to that communications line (Section 2.2.4). The DSR (Data Set Ready) transition condition does not affect buffer assignments. All other error conditions cause the pertinent reception or transmission operation and the currently associated buffer to be terminated. All other assigned buffers are unaffected and remain assigned. In addition, all receive errors cause the pertinent DUP11 to be placed in the search sync mode for DDCMP operations, or the flag search mode for bit stuffing operations. When a transmit error is detected, the pertinent DUP11 conditions the associated line to mark hold.

NOTE

The address in SEL4 designates the buffer descriptor that was active when the error occurred. The byte count in the pertinent buffer descriptor is changed to reflect the number of characters actually transmitted or received.

Table 2-2
Error Codes for the COMM IOP-DUP Synchronous
Communications Controller Configuration

Code (Octal)	Error Type	Reason for Error
6	Abort	A receive data operation conducted under one of the bit stuffing protocols was terminated by the sending station, and seven consecutive one's were received.
10	Receive DDCMP Header CRC Error	A DDCMP header with an invalid CRC was received.
12	Receive Data CRC Error	In DDCMP mode a numbered message with an invalid CRC was received. In bit stuffing mode, a frame with an invalid CRC was received.
14	No Buffer Assigned	Received data was received without a buffer being assigned by the user program (Section 2.2.2). The pertinent DUP11 will resynchronize the communications line in anticipation of a buffer assignment. In this case, the content of SEL4 in the pertinent CONTROL OUT command is undefined.
16	Data Set Ready (DSR) Transition	Each transition of DSR on each enabled line is posted to the user program through this code. For each line, the first posting designates a DSR transition from off-to-on, the second from on-to-off, the third from off-to-on, and so on. A DSR transition does not affect any currently assigned buffers.
20	Nonexistent Memory	COMM IOP-DUP attempted to access a nonexistent main CPU memory location. This error condition applies to any user program assigned memory location, including a DUP11 CSR address. This error condition only indicates that a nonexistent memory error condition was detected while servicing the communications line designated by BSEL3. To isolate the actual invalid address, the user must determine the addresses of the receive and transmit buffers active on that line.

(continued on next page)

Table 2-2 (Cont.)
Error Codes for the COMM IOP-DUP Synchronous
Communications Controller Configuration

Code (Octal)	Error Type	Reason for Error
22	Transmit Underrun	COMM IOP-DUP is not processing transmitted characters fast enough. This error condition usually results when the user program does not assign subsequent buffers for a message in time, thereby causing COMM IOP-DUP to wait for a buffer assignment. Note that the user program should assign all buffers for a given message as a single list. UNIBUS latency or a slow polling rate (Section 2.2.3.2) can also cause this error.
24	Receiver Overrun	COMM IOP-DUP is not processing received data fast enough. Typical causes of this type of error are UNIBUS latency and a slow polling rate (Section 2.2.3.2).
26	Kill Complete	This error return informs the user program that the kill request has been completed and also serves to help the user program resynchronize with the COMM IOP-DUP completion silo. Note that all transmit or receive buffers assigned to the pertinent line are de-assigned. The IN I/O bit in BSEL2 of the pertinent CONTROL OUT command informs the user program whether the killed buffers were transmit buffers (IN I/O = 0) or receive buffers (IN I/O = 1). In addition, the buffer descriptor address field of the pertinent CONTROL OUT command contains the address of the buffer descriptor in use when the kill was executed. The byte count field of the pertinent buffer descriptor (Figure 3-1) contains a count of the number of bytes transferred prior to execution of the kill.

CHAPTER 3

SYSTEM OPERATIONS

A COMM IOP-DUP configured as a synchronous communications line controller has two basic modes of protocol operation:

1. Digital Data Communications Message Protocol (DDCMP) V4.0
2. The bit stuffing protocols:
    a. SDLC - IBM
    b. ADCCP - ANSI
    c. HDLC - ISO
    d. BDLC -Burroughs
    e. X.25 - ISO
    f. SNAP - Data Pac

As described in Section 2.2.3.1 under CONTROL IN Command Format, the protocol mode for each COMM IOP-DUP controlled communications line is assigned at initialization time by the pertinent CONTROL IN command.

Transmit and receive operations under both protocol modes require the assignment of buffer descriptor lists to the pertinent communications line. These lists contain buffer descriptors which in turn point to buffers in the main CPU physical address space. COMM IOP-DUP accesses these assigned buffers on an NPR basis to store received data or to retrieve data to be transmitted.

Under both DDCMP and the bit stuffing protocols, a single message can encompass a number of buffers. In addition, a message always starts at a new buffer. The narrative that follows starts with a detailing of the structure and format of the component buffer descriptors in COMM IOP-DUP buffer descriptor lists. This description is used as the basis for a detailed delineation of DDCMP and bit stuffing protocol operations.

3.1 BUFFER DESCRIPTOR FORMAT

A buffer descriptor list is a contiguous set of word aligned locations in main CPU memory assigned to a specific communications line by the user program through the issue of a BUFFER ADDRESS IN command. Structurally, a buffer descriptor list consists of a sequential series of 3-word blocks with each block forming a single buffer descriptor. The only limits to the length of a buffer descriptor list are the practicalities of memory allocation and the size of main CPU memory.

A user program can assign up to two receive and two transmit buffer descriptor lists to each COMM IOP-DUP communications line.

The format for a COMM IOP-DUP communications line buffer descriptor is shown in Figure 3-1. The first word of each descriptor contains the 16 low-order bits of the UNIBUS address for the associated buffer, starting with the two high-order bits contained in bit positions 10 and 11 of the third word.

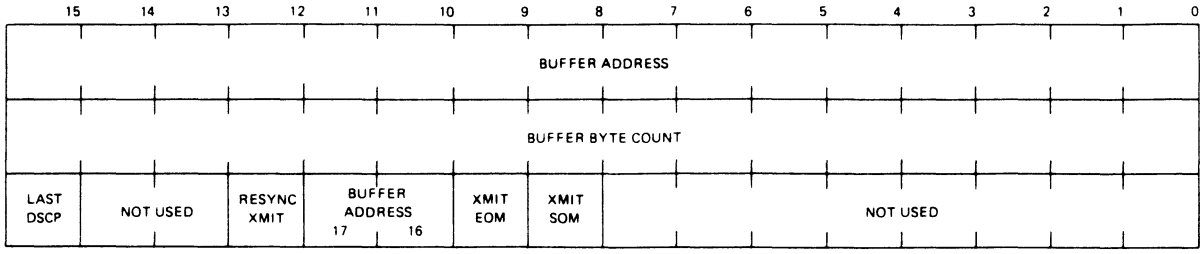

Figure 3-1 COMM IOP-DUP Synchronous Communications Controller Buffer Descriptor Format

A 16-bit byte count field defining the length of the associated buffer is contained in the second word of each buffer descriptor. When used in conjunction with the UNIBUS address of the buffer starting location, this field provides COMM IOP-DUP with the necessary information to access the user assigned buffer in a true DMA fashion through NPR executions. This capability relieves the main CPU of a significant portion of the overhead normally associated with communications line activities.

Bit positions 8 and 9 in the third word of each buffer descriptor contains the XMIT SOM (Start Of Message) and XMIT EOM (End Of Message) bits, respectively. When set to one, these bits define the boundaries of a transmitted message. Note that in buffer descriptors assigned for receive data, the XMIT SOM and XMIT EOM bits must be zero.

For example, if a given transmit buffer descriptor has the XMIT SOM bit set, COMM IOP-DUP knows that the first byte in the buffer pointed to by the buffer descriptor is the first byte of the message to be transmitted. In addition, COMM IOP-DUP knows that the message continues in the buffer pointed to by the next assigned buffer descriptor. Similarly, in a given transmit buffer descriptor if the XMIT EOM bit is set, COMM IOP-DUP knows that the last byte in the buffer pointed to by that buffer descriptor is the last byte of the message. Therefore, in a given buffer descriptor, if both the XMIT SOM and XMIT EOM bits are set, the message to be transmitted is completely contained in that buffer.

With reference to Figure 3-1, bit 12 of the third buffer descriptor word, the RESYNC XMIT bit, provides the user program with the mechanism for separating messages and informing COMM IOP-DUP when the pertinent line requires resynchronization prior to transmitting data

from the assigned buffer. The need for line resynchronization typically results from line turn around on half-duplex lines or a transmit error such as detection of an erroneous CRC character (Block Check Characters, BCC). Since this bit is pertinent to only transmit buffers, it must be cleared to zero in buffer descriptors assigned for receive data.

For lines operating in DDCMP mode, COMM IOP-DUP will transmit eight sync characters (ASCII 226) when bit 12 is set prior to accessing the assigned buffer. On lines operating under one of the bit-stuffing protocols, COMM IOP-DUP will first initiate transmission of a stream of 16 zero bits when bit 12 is set and then send the starting flag followed by the assigned transmit buffer. In the bit-stuffing mode, bit 12 can only be set after the pertinent line has been idled; for example, after a turn-around on a half-duplex line.

As previously denoted in this section, a buffer descriptor list is an ordered sequential list of 3-word buffer descriptors in main CPU memory space and accessed by COMM IOP-DUP to determine the location and size of each related buffer. The end of a buffer descriptor list is designated by bit 15 of the third descriptor word, specifically the LAST DSCP bit. This bit is always set to one in the last buffer descriptor in a list. Upon reaching the end of a buffer descriptor list, COMM IOP-DUP will continue processing with the next assigned list. For a receive data operation, if a second buffer descriptor list is not assigned and another character is received COMM IOP-DUP will post a No Receive Buffer Assigned Error Return (Table 2-2).

For a transmit operation, if the last buffer descriptor in the assigned list does not have the XMIT EOM bit set, COMM IOP-DUP will wait until another buffer is assigned before continuing the transmit operation. However, if a new buffer is not assigned within one character time after the last character in the last assigned buffer was transmitted, the DUP11 will underrun and as a result COMM IOP-DUP will post an error return (CONTROL OUT) with the code for a Transmit Underrun Error (Table 2-2).

3.2 DDCMP OPERATIONS

DIGITAL's standard message protocol, Digital Data Communications Message Protocol (DDCMP), is a character-oriented protocol designed for error-free transmission and reception over both half- and full-duplex synchronous communications lines. DDCMP is characterized by extremely high efficiency, low overhead, and operation over point-to-point and multidrop lines within the same message format.

Under DDCMP, the message handling function performed by COMM IOP-DUP includes header analysis, message assembly and disassembly, CRC calculation and checking, I/O buffer management, and error checking.

However, COMM IOP-DUP does not perform any higher level protocol functions. This activity must be performed under control of the main CPU operating system. In addition, the higher level must provide a valid DDCMP header with each transmitted message, and error recovery must be handled by the user program for both transmit and receive operations. Finally, line control on half-duplex lines is the responsibility of the user program, using the facilities provided by COMM IOP-DUP. Based on the functional specification for DDCMP Version 4.0, COMM IOP-DUP is primarily responsible for the framing function, and the user program is primarily responsible for the link management and message exchange functions.

Under DDCMP, CRC calculation for transmit and receive messages must be enabled for the pertinent line by the user program. In DDCMP mode, the DUP11 Synchronous Line Units associated with a COMM IOP-DUP use the CRC algorithm CRC-16. At the beginning of a transmit or receive message, CRC-16 is automatically initialized to zero.

For half-duplex operation under DDCMP, transmission will always take precedence over reception. As a consequence, assignment of a transmit buffer by the user program will initiate a transmit operation on the pertinent line irrespective of a receive operation being conducted. On this basis, it is the responsibility of the user program to control buffer assignments so that a transmit buffer is not assigned to a given line when a receive operation is pending or under way.

3.2.1 DDCMP Transmission

A DDCMP message by the main CPU physical link control layer consists of either a message consisting of a 6-byte header or a 6-byte header followed by a variable-length message. A header alone is referred to as an "unnumbered" message, whereas a header followed by a message is called a "numbered" message or a "maintenance message".

With reference to Sections 2.2.2 and 2.2.3, the user program initializes, enables, and establishes the characteristics of each COMM IOP-DUP supported communications line through the BASE IN and CONTROL IN commands. With a given line enabled for DDCMP operation, the user program starts a transmit operation by issuing BUFFER ADDRESS IN commands to assign the appropriate buffer descriptor lists. For example, in the buffer descriptor list assigned to a given line for transmission in DDCMP mode, the first descriptor for that message will point to a 6-byte buffer containing the header. For maximum efficiency, the header should be contained in a single buffer descriptor. The descriptor for this buffer must have both the XMIT SOM and XMIT EOM bits set. After transmitting the header, COMM IOP-DUP will transmit the CRC value calculated for the six DDCMP header bytes contained in the associated buffer. This CRC value is transmitted immediately following the last header byte transmitted.

For the numbered message, the next descriptor in the list points to a buffer having the XMIT SOM bit set (Figure 3-1) and containing all or part of the message to be transmitted. COMM IOP-DUP again initiates CRC calculations starting with the first character in that buffer and continues calculation until an XMIT EOM bit is detected in that, or a subsequent buffer descriptor. The resulting two block check characters are appended to the message and transmitted.

COMM IOP-DUP also performs all necessary modem control, setting Request-to-Send (RTS), waiting for Clear-to-Send (CTS) and transmitting the message. When a transmission is complete, COMM IOP-DUP delays clearing RTS until the last character has been clocked onto the communications line. If the user program decides to resynchronize the communication prior to transmitting a message, the RESYNC XMIT bit (bit 12 of word three) in the respective buffer descriptor should be set to one.

If an error occurs during transmission, COMM IOP-DUP will immediately halt transmission, store the count of characters transmitted in the pertinent descriptor, and post an error through a CONTROL OUT command. The user program can halt transmission by issuing a BUFFER ADDRESS IN command with the Kill bit set and the Buffer Enable bit cleared (Section 2.2.4). In this case, COMM IOP-DUP directs the pertinent DUP11 to transmit two characters made up of all ones, drop RTS, and

issues an error completion specifying Kill Complete (Table 2-2). To restart transmission after an error completion the user program must assign new buffer descriptor lists to the pertinent communications line, either in conjunction with the kill operation or at some later time.

3.2.2 DDCMP Reception

Line initialization, specification of line characteristics, and buffer descriptor list assignments are performed by the user program in the same manner as in DDCMP transmission operations. In addition, the same list assignment limitation also applies; namely, up to two buffer descriptor lists can be asssigned to each line for receive operations. The major processing distinction between DDCMP transmission and reception is that COMM IOP-DUP is required to do considerable header analysis as part of receive data processing; and, by definition, the header is received first followed by the message body.

As previously indicated the buffer descriptor XMIT SOM and XMIT EOM bits define the boundaries of a transmitted message only and these bits must be zeros in buffer descriptors assigned for receive data operations. For numbered received messages and maintenance messages, COMM IOP-DUP determines the length of a message by extracting the byte count from the header and using that count to differentiate between the last data character and the two succeeding block check characters. CRC verification must be enabled for received messages under DDCMP.

NOTE

CRC characters are not stored but are discarded by COMM IOP-DUP after these characters are checked.

At the beginning of a receive data operation, COMM IOP-DUP checks the first byte in the DDCMP message header. If that byte is not an SOH, ENQ, or DLE character, the pertinent DUP11 is resynchronized. In this case a CONTROL OUT command is not issued.

During a received data operation, COMM IOP-DUP tests the two block check characters following the 6-byte header and the two following the last message data character. When an erroneous CRC value is detected, COMM IOP-DUP halts reception, posts an error completion through the CONTROL OUT command (Section 2.3.3), and begins a search for sync characters.

COMM IOP-DUP considers a DDCMP message to consist of a header (unnumbered message) or a header followed by message data (numbered or maintenance message). As a consequence, COMM IOP-DUP will only post a completion after receiving the entire DDCMP message unless the receive buffer assigned is smaller than the size of the DDCMP message, COMM IOP-DUP will post a BUFFER ADDRESS OUT, with the RECEIVE EOM flag in BSEL7 cleared, and use the next assigned buffer, if any. COMM IOP-DUP starts storing each DDCMP message in a new buffer, thus no two DDCMP messages are ever stored in the same buffer.

COMM IOP-DUP also checks the DDCMP header QSYNC bit, and when set, it places the DUP11 supporting the pertinent communications line in the sync search mode at the completion of the current message.

If the QSYNC bit is not set, COMM IOP-DUP expects the next message to be abutted to the current message or to be preceded by a sync sequence that is at least 8 bytes long.

When a line is designated a slave station line in a multidrop environment (bit 4 of BSEL 7 was set with a secondary address in BSEL 6 of the initializing CONTROL IN command - see Figure 2-4), COMM IOP-DUP checks the header secondary address field for each message received on that line. If the two addresses do not compare, COMM IOP-DUP will flush the message.

In slave station operation, COMM IOP-DUP validates the CRC before checking the secondary station address. If the CRC is invalid, it posts a Receive CRC error return, irrespective of secondary address validity. COMM IOP-DUP ignores a message received at a slave station having a valid CRC but a different secondary station address by receiving each character and then discarding it rather than resynchronizing the associated DUP11. In this circumstance the actual message is received, but it is not stored in the message buffer.

When the CRC is valid and a secondary station address comparison does occur, COMM IOP-DUP verifies the header and then posts a normal completion to inform the user program that the pertinent line is being addressed by the primary station. When assigning buffers to a line enabled for secondary station reception, the user program must assign buffers that are a minimum of 6 bytes in length. This 6-byte buffer length is imposed by the requirement that the complete header be received and stored before verifying the secondary address. To flush a message with a wrong secondary station address, the value 6 is subtracted from the internal buffer pointer to reset the pointer to the starting address of the buffer.

As each receive buffer is completed, COMM IOP-DUP posts a completion through a BUFFER ADDRESS OUT command and stores the accumulated byte count in the associated buffer descriptor. If a received message is still incomplete after the last buffer in an assigned list is completed and a second list has not been assigned, COMM IOP-DUP will post a No Buffer Assigned error completion through a CONTROL OUT command (Table 2-2).

When the user program initiates a kill receive operation, COMM IOP-DUP will terminate the current active receive buffer and will designate the number of characters received prior to the kill by placing that count in the Byte Count field of the associated buffer descriptor. COMM IOP-DUP will also place the associated DUP11 in the Search Sync mode.

3.3 BIT STUFFING PROTOCOL OPERATIONS

COMM IOP-DUP is capable of transmitting and receiving messages under virtually any currently used synchronous bit stuffing protocol including SDLC, ADCCP, HDLC, BDLC, X.25, and SNAP. This bit stuffing protocol transparency is based on three requirements:

1. In receive messages the flag bytes must have a value of 176 (octal).

2. Where applicable, a secondary address must be the first 8-bit byte in the message.

3. All frames must be a multiple of 8 bits.

In processing bit stuffing protocols, COMM IOP-DUP performs message transmission and reception; flag character, abort character, and CRC character generation and detection; and modem control. However the actual processing of message data such as message formatting, routine acknowledgment, and error recovery are the responsibility of the user program and the higher level involved.

The information required at initialization time for a bit stuffing line is the same as for a DDCMP line. For each communications line operating as a bit stuffing line, the user program must assign a line number and a CSR address. In addition, line characteristics such as half/full-duplex operation, CRC enable/inhibit, and operation as a secondary station must also be established (Figure 2-4).

Under the bit stuffing protocols, CRC calculation for transmit and receive messages can be enabled for the pertinent line by the user program. For operation in a bit stuffing mode, the DUP11 Synchronous Line Units associated with a COMM IOP-DUP use the CRC algorithm CRC-CCITT. At the beginning of a receive or transmit operation, CRC-CCITT is initialized to all ones.

For half-duplex operation under a bit stuffing protocol, transmission will always take precedence over reception. As a consequence, assignment of a transmit buffer by the user programs will initiate a transmit operation on the pertinent line irrespective of a receive operation being conducted. On this basis, it is the responsibility of the user program to control buffer assignments so that a transmit buffer is not assigned when a receive operation is pending or underway.

3.3.1 Bit Stuffing Protocol Transmission

A user program starts a transmission of a message under the bit stuffing protocols in the same manner as with DDCMP, namely by assigning up to two buffer descriptor lists describing transmit buffers to the pertinent communications line. The format for the buffer descriptors within the list is the same for bit stuffing protocols as for DDCMP (Figure 3-1). The boundaries of a transmitted message under the bit stuffing protocols are established by the XMIT SOM and XMIT EOM bits in the manner as under DDCMP. In addition, bit stuffing protocols do not place any restrictions on buffer size other than the practical restrictions of memory size and program requirements.

Prior to a transmission, COMM IOP-DUP checks the state of the RESYNC XMIT bit in the assigned buffer descriptor (Figure 3-1). If this bit is set, indicating that the receiving device requires line resynchronization, COMM IOP-DUP will transmit a string of 16 zero-bits to resynchronize the pertinent line.

NOTE

The buffer descriptor bits XMIT SOM and XMIT EOM must be set in the appropriate descriptors to designate the starting and ending buffers comprising a transmitted message.

At the start of a transmitted message, and prior to retrieving the first byte in the message, COMM IOP-DUP sets RTS, waits for CTS to be set by the modem, then transmits a single flag character. This action defines the beginning of the message for the receiver.

Immediately following the flag character, COMM IOP-DUP transmits all designated buffers until a buffer descriptor is detected with the XMIT EOM bit or the LAST DSCP bit set (Figure 3-1). COMM IOP-DUP will post a completion for each completed buffer, through a BUFFER ADDRESS OUT command. COMM IOP-DUP uses the byte count field in each descriptor to determine the number of bytes to be transmitted and therefore when to post the completion of a given buffer.

If a message is incomplete at the last buffer descriptor in an assigned list, COMM IOP-DUP will resume transmission at the beginning of the second assigned list until a buffer descriptor having the XMIT EOM bit set is detected. At that point, a final completion for that message will be posted. If a second buffer descriptor list has not been assigned, COMM IOP-DUP will wait for the user program to assign that list. A slow response by the user program in this circumstance could cause the posting of a Transmit Underrun error (Table 2-2).

Upon transmitting the last data byte of a message, the pertinent DUP11 sends calculated CRC characters for the message body, if CRC calculations are not inhibited, and transmits a flag character to conclude the message. The DUP11 supporting a given line automatically idle marks between messages. The resulting message gap is a flag character followed by all ones and terminated by the flag character starting the next message.

Under the bit stuffing protocols, the modem control requirements for half-duplex line turn-around are handled by COMM IOP-DUP. However, the protocol aspects of half-duplex line turn-around are the responsibility of the user program and the higher level involved.

Once a transmit operation is started, it continues until completed or an error is detected. On half-duplex lines the user program must be careful not to assign a transmit buffer descriptor list to a line on which receive data is expected.

If an error occurs during transmission, COMM IOP-DUP will immediately halt transmission, store the count of charcters transmitted in the pertinent descriptor, and post an error through a CONTROL OUT command. Also, COMM IOP-DUP causes the associated DUP11 to enter the idle mark mode. The user program can halt transmission by issuing a BUFFER ADDRESS IN command with the Kill bit set and the Buffer Enable bit cleared (Section 2.2.4). In this case COMM IOP-DUP directs the pertinent DUP11 to transmit a bit stuffing protocol abort character (177 octal) informing the receiver of line shutdown and issues an error completion specifying Kill Complete (Table 2-2). To restart transmission after an error completion, the user program must assign new buffer descriptor lists to the pertinent communications line either in conjunction with the kill operation or at some later time.

3.3.2 Bit Stuffing Protocol Reception

Bit stuffing protocol reception can begin on a line any time after the line is initialized, its characteristics established, and buffer descriptor lists are assigned. COMM IOP-DUP identifies the beginning and end of a received message by detection of the pertinent flag character. A single receive message can encompass multiple buffer descriptors. However, each new received message will be started at a new buffer descriptor.

NOTE

CRC characters are not stored but are discarded by COMM IOP-DUP after these characters are checked. If CRC checking is not enabled, COMM IOP-DUP stores all characters between the starting and ending flags of a message.

As each receive buffer is completed, COMM IOP-DUP posts a completion through a BUFFER ADDRESS OUT command and stores the accumulated byte count in the associated buffer descriptor. If a received message is still incomplete after the last buffer in an assigned list is completed and a second list has not been assigned, COMM IOP-DUP will post a No Buffer Assigned error completion through a CONTROL OUT command (Table 2-2). Also, if an error, such as an invalid CRC character is encountered, or the detection of an abort character occurs, COMM IOP-DUP will post the appropriate error completion and terminate the current buffer. When a given communications line is designated as a multidrop line, with this line being designated as a slave at initialization time, COMM IOP-DUP will initiate secondary address checking. As previously indicated, COMM IOP-DUP, in performing secondary address checking on received messages, expects the secondary address to be the byte directly following the flag byte. If the secondary address in a received message does not match the secondary address assigned to that line, the message will be ignored and a search initiated for the next flag. CRC validation is not performed in this circumstance.

NOTE

In order for a DUP11 to detect an abort character, the abort character must be preceded by a character that is neither a flag character nor an abort character.

3.4 SHUTTING DOWN AND REENABLING A LINE

During system operation, it is sometimes necessary to shut down a communications line. The procedures to shut down a line and to reestablish that line should be included in the user program.

In the first step of the shutdown procedure, the user program disables the pertinent line by issuing a CONTROL IN command with the Enable Line bit (bit 0, BSEL7) set to zero. This command is then followed by a two BUFFER ADDRESS IN commands with the Kill bit (bit 2, BSEL7) set to one to deassign all transmit and receive buffers currently assigned to the pertinent line. When all buffers are deassigned, the pertinent line is shut down.

At this point, the line can be reenabled. To reenable a shutdown line, the user program issues a CONTROL IN command with the Enable Line bit set to one and with the specific fields set to establish the required line characteristics.

This command is followed by a BUFFER ADDRESS IN command with the Kill bit set zero for each buffer descriptor list to be assigned to the pertinent line. If an existing line is to be reassigned to a new DUP11, the user program must first issue the appropriate BASE IN command, which is then followed by the CONTROL IN and BUFFER ADDRESS IN commands.

Formats for the CONTROL IN and BUFFER ADDRESS IN commands are shown in Figures 2-4 and 2-5. In addition, the specific command fields are described in detail in Sections 2.2.3 and 2.2.4, respectively.

CHAPTER 4

COMM IOP-DUP-KMC11 MICROPROGRAM LOADER

Before a COMM IOP-DUP microprogram can be initialized, it must be loaded into the KMC11 Control RAM (CRAM). Similarly, following a power failure, the COMM IOP-DUP microcode must be reloaded and again initialized. This is necessary because the states of the KMC11 internal registers and memories are lost as a consequence of a power failure.

The COMM IOP-DUP microcode is supplied as a microprogram image file on a variety of media. Two steps are required to load the COMM IOP-DUP microcode into the KMC11 CRAM:

1. Enter the COMM IOP-DUP microcode into a preassigned space in main CPU memory.

2. Using the loader described in this chapter, load the COMM IOP-DUP microcode contained in the main CPU memory space into the KMC11 CRAM.

The second step can be implemented in one of two ways. The utility KMCLDR can be run as a task by the pertinent operating system or the required portions of KMCLDR code can be incorporated in the user program.

The COMM IOP-DUP/KMC11 microprogram loader is a utility that runs as a privileged task under RSX-11M, RSX-11D, or IAS. If the user wants to develop his own loader, the basic loader subroutine in Section 4.2 is an example that can be used as a basis for development of a user specific loader. An example is provided by the KMC11 Loader (KMCLDR), which operates on RSX-11M (Section 4.3).

In this chapter, the KMC11 is used as the reference point for all transfers of information between the PDP-11 processor program and the microprocessor. An OUT-transfer transfers information from the KMC11 to the PDP-11 program; an IN-transfer transfers information from the PDP-11 program to the KMC11.

As indicated in Chapter 2, the KMC11 CSRs are used for the exchange of control and status information between the PDP-11 program and the COMM IOP-DUP microprogram. The only CSR having a fixed or hardware-defined format is the maintenance register (Figure 4-1). The maintenance register (BSEL1) is used primarily for initializing and servicing the KMC11; two BSEL1 bits (RAM 0 and RAM I), however, are used when loading the COMM IOP-DUP microprogram into KMC11 CRAM.

RAM 0, when set, modifies the KMC11 data paths for SEL4 to be the CRAM maintenance address register, enabling CRAM read and/or write through SEL6 to the location addressed by that register. A write is accomplished by loading the new CRAM data into SEL6 and asserting CRAM WRITE. CRAM read is accomplished by reading SEL6.

CRAM WRITE, when set, allows the contents of SEL6 to be loaded into the CRAM at the address specified by SEL4. Note that RAM I must also be set to accomplish the loading procedure.

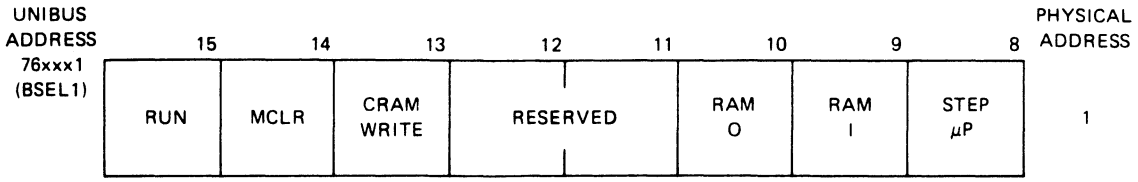

Figure 4-1 Control and Status Register CSR1 Bit Map

4.1 KMC11 BASIC LOADER SUBROUTINE

The KMC11 loader subroutine is an example of the utility running on a user-developed driver. (See Figure 4-2.)

```
; WRITE THE RAM


; INPUTS:
;       R0 = NUMBER OF WORDS TO WRITE
;       R3 = CSR ADDRESS OF KMC11
;       R5 = CRAM ADDRESS AT WHICH TO START LOADING
;       BUFF = BUFFER CONTAINING MICRO-INSTRUCTIONS
;       S.LOAD (STATUS) = FLAG TO INDICATE A LOAD (1) OR COMPARE (0)
;                IS TO BE PERFORMED

WTRAM:
        MOV     #BUFF,R4        ;GET BUFFER ADDRESS
10$:    BIT     #S.LOAD,STATUS  ;LOAD KMC?
        BEQ     15$             ;NO, JUST COMPARE
        MOV     #2000,(R3)      ;SELECT CRAM
        MOV     R5,4(R3)        ;LOAD ADDRESS
        MOV     (R4),6(R3)      ;PUT THE DATA IN THE REGISTER
        BIS     #20000,(R3)     ;CLOCK IT IN
15$:    CLR     (R3)            ;CLEAR CSR 0
        CLR     4(R3)           ;CLEAR CSR 4
        CLR     6(R3)           ;CLEAR DATA PORT
        MOV     #2000,(R3)      ;READ CURRENT CRAM LOCATION
        MOV     R5,4(R3)        ;LOAD ADDRESS AGAIN
        CMP     (R4),6(R3)      ;EQUAL TO WHAT JUST WRITTEN THERE?
        BNE     25$             ;NO,RAM WRITE ERROR
20$:    ADD     #2,R4           ;ADDRESS NEXT WORD IN INPUT FILE
        INC     R5              ;INCREMENT THE RAM ADDRESS
        DEC     R0              ;ONE LESS WORD
        BNE     10$             ;KEEP GOING
        CCC                     ;CLEAR CONDITION CODE
        RETURN
25$:    ERROR ROUTINE
          .
          .
          .
        BR      20$             ;CONTINUE
```

Figure 4-2 KMC11 Loader Subroutines

The following procedure for loading the KMC11 CRAM fully uses the KMC11 hardware and provides for future compatibility:

1. Set BSEL1 bit 2 (RAM OUTPUT).
2. Load the right-justified PC into SEL4.
3. Load CRAM data into SEL6.
4. Set BSEL1 bit 5 (CRAM WRITE).
5. Clear SEL0.
6. Repeat Steps 2 through 5 as necessary to load the required instructions.

The following procedure is used to verify the CRAM:

1. Set BSEL1 bit 2.
2. Load the right-justified PC into SEL4.
3. Read CRAM data from SEL6.
4. Clear SEL0.
5. Repeat Steps 2 through 4 as necessary to verify the instructions.

4.2 KMC11 LOADER RUNNING ON RSX-11M

Figure 4-3 is printout of KMCLDR running on RSX-11M, and Figure 4-4 is an error printout example. In both examples, the underscored text is system-generated and the remaining text is user-generated.

```
>RUN KMCLDR
KMC LOADER
CSR? 170
FILE NAME? COMIOPDUP
LOAD OR COMPARE? L
KMC LOAD COMPLETE
>
```

Figure 4-3 KMC11 Loader Printout Example

```
>RUN KMCLDR
KMC LOADER
CSR? 170
FILE NAME? COMIOPDUP.TSK
LOAD OR COMPARE? C
***KMC COMPARE ERROR AT 000050 SOURCE=101020     KMC RAM=101024 ***
KMC COMPARE COMPLETE
>
```

Figure 4-4 KMC11 Loader Error Printout Example

In Figures 4-3 and 4-4, the user-generated answer (170) to the system-generated question "CSR?" is ORed by the system hardware with 760000 to obtain the address of CSR0; i.e., 760170. The default for the device is SY (system device) and the default for user code is the current user identification code.

In Figures 4-3 and 4-4, the file name is COMIOPDUP; there are no defaults for file name. In Figure 4-4 the file type is .TSK; the default for file type is .TSK. For the file version, the default for an input file is the highest-numbered existing version.

During a load operation, a compare is automatically performed and a message is typed for all errors. If the error printout of a load operation indicates a faulty CRAM location, DIGITAL Field Service should be called to correct the situation.

A compare is also useful during debugging to obtain a listing of all modified locations when CRAM locations have been changed. The error printout example in Figure 4-4 indicates this use of the loader. Alternatively, an error could indicate a faulty CRAM location if the user has not modified the CRAM since loading.

4.2.1 Loader Assembly

To assemble the loader, the user should type the following statement after the prompt, which is underlined for clarity:

```
> MAC KMCLDR=[1,1]EXEMC/ML,[user UIC],KMCLDR
```

NOTE

The KMC11 loader must be assembled and the microcode must be built on the same version of RSX-11M.

4.2.2 Loader and COMM IOP-DUP Microcode Task Building

To task build the loader, the user should type the following statement after the underlined prompt:

```
> TKB KMCLDR/PR=KMCLDR
```

To task build the microcode, the user should type the following statements after the underlined prompts:

```
TKB>file name/-HD/-MM=file name.OBJ
TKB>/
ENTER OPTIONS:
TKB>STACK=0
TKB>PAR=:0:1000
TKB>//
```

NOTE

File name .OBJ is the output of the assembler. (See Chapter 4 of the KMC11 Programmer's Manual, AA-5244B-TC.)

The output of the task builder results in a file with at least two label blocks of 512 bytes each, followed by the microcode instructions. These label blocks are stripped (ignored or skipped) by the KMCLDR and should also be skipped if a user-designed utility is used to read this file.

Detailed task building instructions are contained in the RSX-11M Task Builder Reference Manual, DEC-11-OMTBA.

APPENDIX A

COMM IOP-DUP INTERRUPT HANDLING

Figure A-1 is a flow chart of a suggested user program routine for the handling of interrupt dialogue between COMM IOP-DUP and the user program. Note that steps A0 and B0 disable KMC11 interrupts. In addition, steps A1 and A2 or B1 through B3 can be performed by the user program at a priority level lower than that normally assigned to the KMC11 so that devices at a higher priority level will not be inhibited from interrupting. This method minimizes interrupt lockout time for other devices on the UNIBUS at the same time that the user program is receiving and processing COMM IOP-DUP completions.

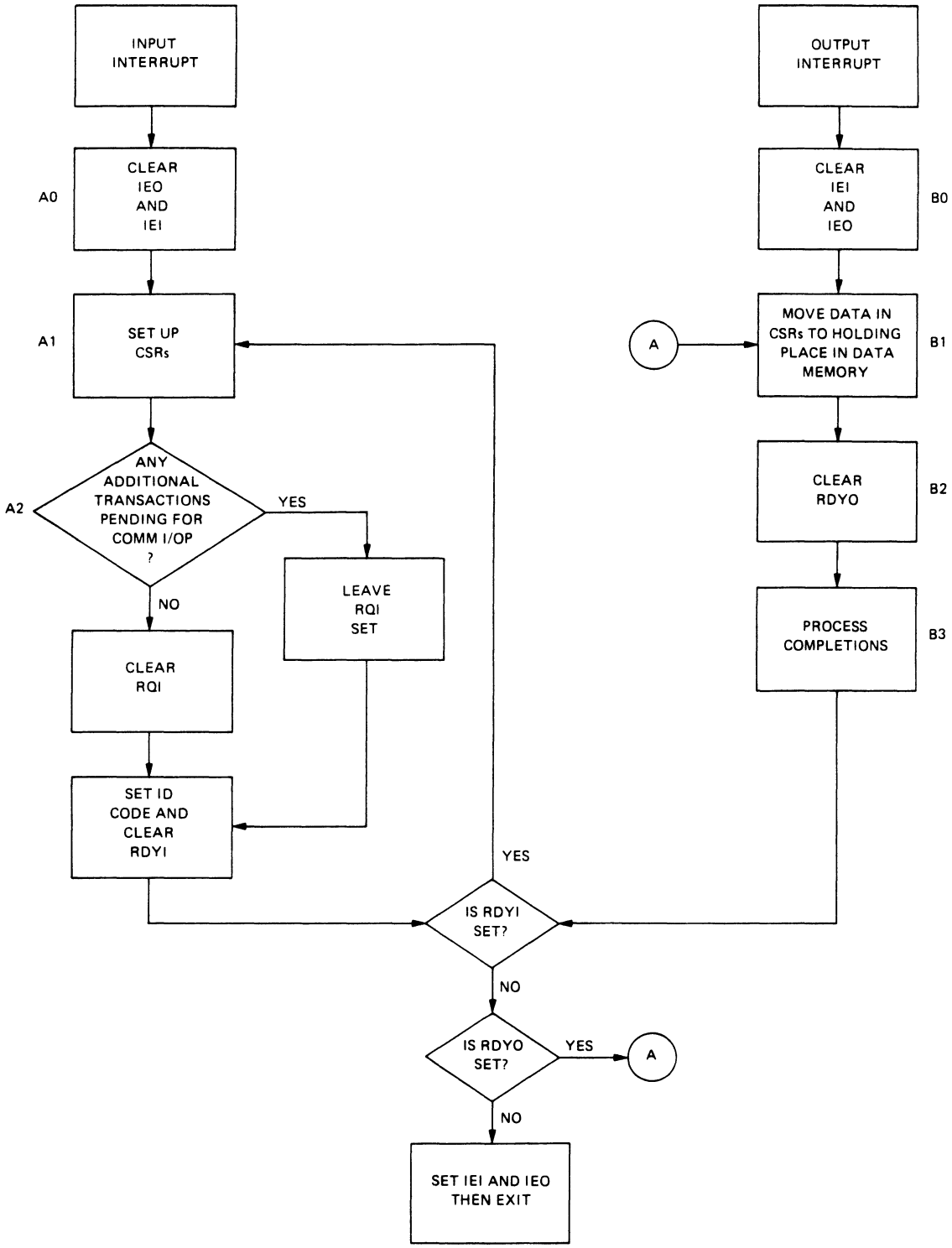

Figure A-1 Flow Chart of a User Program Routine to Handle COMM IOP-DUP Interrupt Processing

INDEX

COMM IOP-DUP
Programming Manual
AA-5670A-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ **Assembly language programmer**
- ☐ **Higher-level language programmer**
- ☐ **Occasional programmer (experienced)**
- ☐ **User with little programming experience**
- ☐ **Student programmer**
- ☐ **Non-programmer interested in computer concepts and capabilities**

Name _____ **Date** _____

Organization _____

Street _____

City _____ **State** _____ **Zip Code or Country** _____

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Documentation
146 Main Street ML5-5/E39
Maynard, Massachusetts 01754

COMM IOP-DUP V1.0

Problem Statement:

Messages received in DDCMP mode by the Comm IOP-DUP are randomly ignored if they are abutted to the preceeding message. A message is defined to be abutted if the first character of the DDCMP header immediately follows the last CRC character of the preceding message without any intervening sync characters. The DDCMP protocol will be able to recover using the REP message after the acknowledgment timeout but throughput will be impacted significantly.

Technical Analysis:

The microcode will randomly decide to search for sync after the end of a DDCMP message instead of using the QSYNC flag. No problem exists if it does not decide to search for sync and the next message is not abutted, since it will initiate a search for sync when it finds the next character is an invalid DDCMP start of header character. However, if multiple messages are abutted and the microcode initiates a search for sync after one of them, then the following abutted messages will be ignored until a message preceded by sync characters is received.

Since the lost messages will not be acknowledged by the PDP-11 DDCMP module, the sending station will timeout and cause these messages to be retransmitted.

Fix:

If the user has an RSX-11M/D or IAS system, the ZAP utility can be used to patch the microcode. The procedure is as follows:

```
>ZAP
ZAP>COMIOPDUP.TSK/AB
_3:2000/
003:002000/ 117567
_60531
_<CR>
003:002002/ 010572
_114762
_3:3744/
003:003744/ 000000
_117564
_<CR>
003:003746/ 000000
_100572
_X
```

For other users, you must patch the microcode file yourself. The file consists of six(6), 256 word blocks. The specific words to be changed are as follows:

Block	Offset in the block	Current	New
5	0	117567	060531
5	1	010572	114762
6	242.	000000	117564
6	243.	000000	100572

The Current and New values are given in octal, all the rest are decimal. The Offset is the offset from the start of the block in words.