communications controllers

reference manual

INCOTERM®
CORPORATION

COMMUNICATION CONTROLLERS

REFERENCE MANUAL

ORDER NUMBER: MS-7152.Ø

DATE: FEBRUARY, 1974

TABLE OF CONTENTS

Page

SECTION I – INTRODUCTION AND GENERAL DESCRIPTION

Introduction 1-1
General Description 1-1
Asynchronous Controller 1-1
Synchronous Controller 1-1
Party Line Controller 1-2
SPD-M Multiplexer 1-2

SECTION II – PROGRAMMING

Introduction 2-1
Controller Command Instruction Classes 2-1
I/O Class Instructions 2-1
I/O Test-Jump Class Instructions 2-2
Asynchronous Controller Command Set 2-2
Controller Operation Commands 2-2
Reset RTS' 2-2
Reset 2-4
Set Transmit Mode 2-4
Set Receive Mode 2-4
Set Line Break/Set RTS' 2-4
Mask Interrupts 2-4
Unmask Interrupts 2-4
Controller Data Transfer Commands 2-4
Write Data 2-4
Set Data Terminal Ready 2-5
Reset Data Terminal Ready 2-5
Reset RTS' 2-5
Read Data 2-5
Asynchronous Controller Test Commands 2-5
Acknowledge If Controller Present 2-5
Acknowledge If Data Set On Line 2-5
Acknowledge If No Data Set Error 2-5
Acknowledge If No Overrun 2-6
Acknowledge If No Line Break 2-6
Asynchronous Controller Command Merging 2-6
Synchronous Controller Command Set 2-7
Synchronous Controller Control Commands 2-7
General Reset 2-7
Mask Interrupts 2-8
Unmask Interrupts 2-8
Set Transmit Mode 2-8
Set Receive Mode 2-8
Synchronous Controller Data Transfer Commands 2-8
Write Data 2-8
Read Data 2-8

TABLE OF CONTENTS (Cont'd.)

Page

Synchronous Controller Test Commands 2-9
Acknowledge If Controller Present 2-9
Acknowledge If No Data Set Error 2-9
Acknowledge If No Overrun 2-9
Acknowledge If Data Set On Line 2-9
Synchronous Controller Command Merging 2-9
Party Line Controller Command Set 2-9
Multiplexer Command Set 2-12
Communications Controller/Multiplexer Command Set 2-12
Set Priority Data 2-12
Acknowledge If No EOT 2-13
Acknowledge If No Cancel 2-13
SPD-M Multiplexer Program Logic 2-14
Single Transmission Logic 2-14
Multiple Transmission Logic 2-15

LIST OF ILLUSTRATIONS

		Page
Figure 2-1	I/O Class Instruction Format	2-1
Figure 2-2	I/O Test-Jump Class Instructions	2-1
Figure 2-3	Air Lines Reservation System	2-11
Figure 2-4	Multiplexer Program Logic Single Transmission	2-15
Figure 2-5	Multiplexer Program Logic Multiple Transmission	2-16

LIST OF TABLES

		Page
Table 2-1	Asynchronous Controller Command Summary	2-3
Table 2-2	Summary of Merged CIO Commands	2-6
Table 2-3	Synchronous Controller Command Summary	2-7
Table 2-4	Party Line Controller Command Summary	2-10
Table 2-5	Asynchronous Controller/SPD-M Multiplexer Command Set	2-13
Table 2-6	Synchronous Controller/SPD-M Multiplexer Command Set	2-14

SECTION I

INTRODUCTION AND GENERAL DESCRIPTION

INTRODUCTION

This manual provides the programmer with both general and detailed information concerning the INCOTERM asynchronous and synchronous communications controllers, the party line controller, and the SPD-M multiplexer. The asynchronous and synchronous controllers interface the SPD 10/20 Terminal Processing Unit (TPU) with a modem or the SPD-M multiplexer; whereas, the party line controller provides direct local communication between clustered terminals. All the controllers are integrated solid state printed circuit modules which occupy specific internal connector slots in the SPD 10/20. The SPD-M multiplexer, however, is a separate unit which connects by cable to each associated SPD 10/20 connector slot assigned to the communication line. The SPD-M multiplexer is included in discussion of the communications controllers because its operation may be directly involved with the asynchronous and synchronous controllers in certain terminal system configurations.

GENERAL DESCRIPTION

The discussion in this paragraph provides insight into the function and operation of each communications controller and the SPD-M multiplexer. This level of information will be helpful to the programmer in understanding the implications of the instruction sets for the several communications controllers.

ASYNCHRONOUS CONTROLLER

The asynchronous communications controller interfaces the TPU with a modem, the SPD-M multiplexer or any device with a standard communications adapter. This RS-232-C compatible controller operates in a half-duplex mode performing both receive and transmit functions. Full duplex operation can be obtained by using two of these controllers, one configured for receiving data and one for transmitting data. As implied by its name, this communications controller operates in an asynchronous manner where each character is separately received or transmitted over the communications line in bit serial form. The bits comprising each character received or transmitted are shifted into or out of the controller buffer by clocks which are internal to the controller. Transfers of character data between the TPU and the controller are made in parallel mode.

Character boundaries are defined by one "Start" bit and up to three "Stop" bits which are automatically assigned to each transmitted character by the controller. The start and stop bits accompanying received serial data serve to define each character for input to the controller buffer. Start and stop bits accompanying received data are discarded by the controller as each received character is stored in the buffer.

Optional configurations of the asynchronous controller having program implications include word lengths of five, six, seven or eight bits, data transfer rates ranging from 100 to 9600 bits per second, and even, odd or transparent parity checking. All aspects of communication line discipline are performed automatically by the controller with the only software function being the handling of read and write operations, normal communications handshaking, and interrupt processing.

SYNCHRONOUS CONTROLLER

Like the asynchronous controller, the synchronous communications controller interfaces the TPU with a modem, the SPD-M multiplexer or any device with a standard communications adapter. The synchronous controller is also RS-232-C compatible and operates in half duplex mode performing both receive and transmit functions. Full duplex synchronous communication can be achieved by using two controllers, one configured for transmitting and one for receiving.

Character data received or to be transmitted is transferred between the TPU and the synchronous controller character-by-character in parallel form. With a synchronous controller, data is received or transmitted over the communications line as a serial bit stream with no separation between characters and synchronized by a clock signal furnished by the data set. As a consequence, the operating bit rate is

determined by the data set. A serial bit stream consists of a set of characters comprising a message and is headed by two or more sync (synchronization) characters which are automatically assigned to transmitted data by the controller. Sync characters are assigned to received data by the data set. These sync characters preceding a bit stream serve to phase the bit stream with the data set clock signal. The end of a synchronous transmission or receipt is designated by a gap in the serial bit stream which forces the controller into a standby or "Marking" state. A major advantage of the synchronous communications controller over the asynchronous controller is that for a given bit rate, the number of data characters transferred during a given period is substantially higher. This increased transfer rate is a result of the data compression effect characteristic of bit stream transfers.

The synchronous communications controller is available in a wide variety of optional configurations having programming implications. Data word length can be six, seven, or eight data bits with the LSB transmitted first, or six bits only with MSB transmitted first. Parity can be selected as odd, even, or transparent, and received data polarity and the number of sync characters transmitted can be specified. The synchronous controller can be configured for no RTS delay (normal operation), for an RTS delay, or for automatic RTS reset upon detection of overrun. The controller can also be configured to operate with the SPD-M multiplexer.

As with the asynchronous controller, all aspects of communication line discipline are performed automatically by the synchronous controller. Software functions necessary to the operation of the synchronous controller include read and write processing, normal communications handshaking, and interrupt handling.

PARTY LINE CONTROLLER

The party line controller is basically an asynchronous controller whose communication line input and output circuits have been modified to support two-wire communications without the necessity for a modem. This controller provides for local communication between up to 12 SPD 10/20 display terminals at data transfer rates up to 9600 bits per second. These units can also be used within a cluster configuration to interface shared peripherals such as an auxiliary memory, a printer, or disk unit. There is a maximum cable run distance of 1,000 feet from the first terminal on the party line to the last. Data is transferred asynchronously between the TPU and the controller character-by-character in parallel form, with data being received and transmitted between party line controllers serially by character. Character boundaries are defined by one Start bit and one Stop bit which are automatically assigned to each transmitted character by the controller. Start and Stop bits accompanying received data are discarded by a receiving controller. Optional configurations of the party line controller having programming implications include word lengths of five, six, seven or eight bits, and data transfer rates selectable from 100 to 9600 bits per second. The software functions necessary to operation of the party line controller include read and write processing, normal communications handshaking, and interrupt handling.

SPD-M MULTIPLEXER

The SPD-M multiplexer is used in conjunction with asynchronous or synchronous communication controllers, operating half or full duplex, to interface eight, or optionally up to 16, TPU's with a modem. Through the use of cascading techniques, single modem configurations can incorporate more than one multiplexer, thereby permitting more than 16 TPU's to interface with a single modem.

The SPD-M operates in a single or multipoint environment with communication traffic being handled by a central processor (CPU) oriented polling system.

This multiplexer will interface to any RS-232-C compatible modem and operate at data transmission rates ranging from 110 to 4800 bits per second. Data word length for both received and transmitted data is variable from five to eight bits, and the unit is transparent to all code patterns and message envelope structures in both the receive and transmit modes. Those optional configurations of the SPD-M multiplexer having programming implication include word lengths of five, six, seven, or eight (matching word lengths of the associated communications controllers), baud rate selection and a positive response capability by each TPU to general polling messages.

SECTION II

PROGRAMMING

INTRODUCTION

This section provides the detailed technical information necessary to the development of programs which direct the communications controllers in the performance of specific data communications tasks. Discussion begins with a description of the two classes of SPD 10/20 instructions which command the controllers, and is followed by a presentation of the command set for each communication controller.

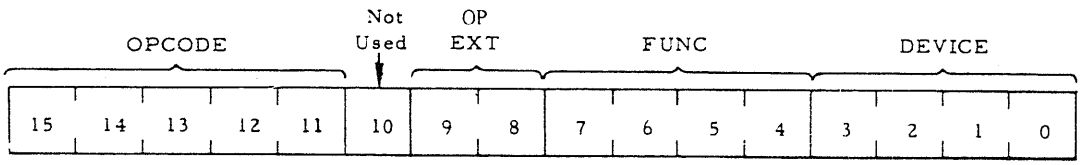
It should be noted at this point that the SPD-M multiplexer, which responds to subsets of the asynchronous and synchronous controller command sets, is discussed under a separate heading.

CONTROLLER COMMAND INSTRUCTION CLASSES

Two classes of SPD 10/20 instructions are executed by the TPU to implement controller commands. These are the I/O class and the I/O Test Jump class. Together, these two instruction classes provide the complete framework for communications controller command sets.

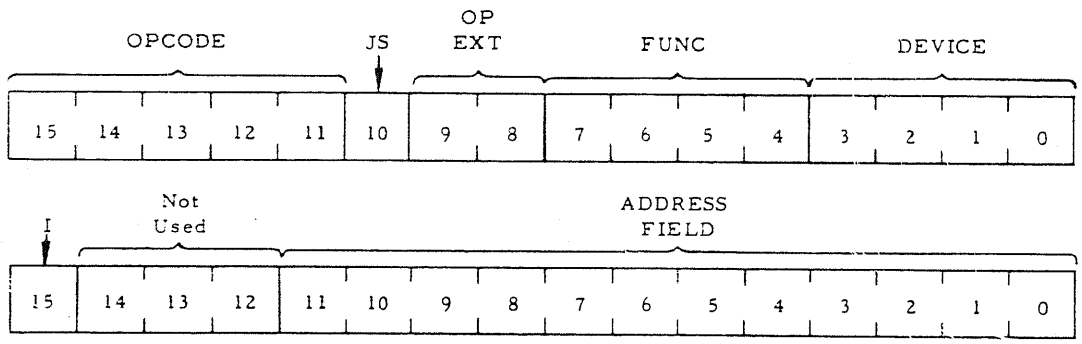
I/O Class Instructions

I/O class instructions are used to implement the controller CIO (control), WIO (write data), and RIO (read data) commands. As shown in Figure 2-1, the format for this single word instruction class contains four operational fields. The Controller Address field contains the four bit device address of the specific communications controller being referenced by the instruction. Communication controller bus address assignments are determined by the specific system configuration. Therefore, all references to the content of this field will be represented by the letter n.



OPCODE = Operation Code
OP EXT = Extension to the Opcode
FUNC = Function

Figure 2-1. I/O Class Instruction Format



OPCODE = Operation Code
JS = Jump Sense Flag
OP EXT = Extension to the Opcode
FUNC = Function
I = Indirect Address Flag

Figure 2-2. I/O Test-Jump Class Instructions

The function field contains a four bit binary code which defines the specific command to be performed by the addressed controller as a consequence of instruction execution by the TPU. The two-bit OPEXT field serves to logically differentiate between CIO, RIO and WIO instructions. For CIO commands, the OPEXT field contains the binary code 01; for a WIO command, the OPEXT field contains the binary code 11; and for a RIO command, the OPEXT field contains the binary code 10.

Bit 10 is unused, and the Opcode field, bits 11 through 15, define the instruction class. With reference to Figure 2-1, note that the content of bit positions 8 through 15 serves to define the first two hexadecimal numerals of each I/O class instruction.

I/O Test-Jump Class Instructions

This two-word instruction class serves to implement the controller test (TIO) commands through either a JFACK or JTACK instruction. As shown in Figure 2-2, the format for this instruction class has seven fields, five in the first word and two in the second.

In the first word, the Controller Address field contains the four bit device address of the specific communications controller to be tested by the instruction. The function field contains a four bit binary code which identifies the test to be performed as a consequence of instruction execution by the TPU. The OPEXT field, bits 8 and 9, contain the binary code 00 to identify the instruction as a TIO command. The Jump Sense (JS) field, bit 10, designates whether a jump is to be made on the tested condition being true or false. Bits 11 through 15, which comprise the Opcode field, serve to identify the instruction as a test-jump class instruction.

In the second word, the Jump Address field, bits 0 through 11, contain the 12-bit address to which the Program Counter is set when the sense of the condition tested matches the sense specified by the Jump Sense field. As a consequence, the program jumps to the memory location specified by this address field when the test is successful. Bits 12, 13 and 14 are unused. The I field, bit 15, denotes whether the address contained in the Jump Address field is a direct or indirect address. I = 0 designates a direct address and I = 1 designates an indirect address.

ASYNCHRONOUS CONTROLLER COMMAND SET

This discussion concerns the 16 commands to which the asynchronous communications controller responds when operating in non-multiplexer configurations. INCOTERM furnishes five versions of the asynchronous communications controller; these are the Models 702, 703, 705, 706 and 707. The applicability of each command in the asynchronous controller command set to these five versions is summarized in Table 2-1. Inspection of this table will show that, with the exception of four commands, this command set can be used with all versions of the controller. These four exceptional commands, noted in Table 2-2, result from the nature of the Model 703, 706 and 707 controllers which are designed for use in communicating with non-multiplexer, IBM 2260 based systems. These controllers can also be used in other conventional communication environments as implied by the applicability of the remaining command set to the Model 703, 706 and 707 controllers. The 707 controller may also be used for auto answer/auto dial.

The asynchronous controller command set is formed by three command categories; these are controller operation (CIO) commands, data transfer and control (WIO, RIO) commands, and test-condition (TIO) commands.

Controller Operation Commands

Controller operation commands provide the programmer with the mechanisms for initializing the controller, establishing I/O modes, setting line breaks, and control of interrupt capability. The instruction format for CIO commands is shown in Figure 2-1. Each of these commands is discussed and defined in detail below.

CIO 0 Reset RTS' (C90n)

This command resets Request-to-Send (RTS') flag only in the Model 703, 706 and 707 controllers which are designed for use in 2260 – based terminal systems. This command is invalid for the remaining asynchronous communications controller versions.

TABLE 2-1

ASYNCHRONOUS CONTROLLER COMMAND SUMMARY

Command Mnemonic	Command Function	Function Code Hex	Function Code Binary	Hexadecimal Model	Controller Versions Used On
CIO	Reset RTS'	0	0000	C90n	703, 706 and 707
CIO	General Reset	1	0001	C91n	All Versions
CIO	Set Transmit Mode	2	0010	C92n	All Versions
CIO	Set Receive Mode	3	0011	C93n	All Versions
CIO	Set Line Break	4	0100	C94n	701, 702, 705, and 707
CIO	Set RTS'	4	0100	C94n	703, 706
CIO	Mask Interrupts	C	1100	C9Cn	All Versions
CIO	Unmask Interrupts	8	1000	C98n	All Versions
WIO	Write Data	1	0001	CB1n	All Versions
WIO	Set Data Terminal Ready	2	0010	CB2n	707 only
WIO	Reset Data Terminal Ready	4	0100	CB4n	707 only
WIO	Set RTS'	8	1000	CB8n	707 only
RIO	Read Data	1	0001	CA1n	All Versions
TIO	ACK If Controller Present	0	0000	C80n or CC0n	All Versions
TIO	ACK If Data Set On-Line	1	0001	C81n or CC1n	All Versions
TIO	ACK If No Data Set Error	2	0010	C82n or CC2n	All Versions
TIO	ACK If No Overrun	4	0100	C84n or CC4n	All Versions
TIO	ACK If No Line Break	8	1000	C88n or CC8n	All Versions

CIO 1 Reset (C91n)

This command clears all controller error conditions and places the unit in the read mode. This command also resets line break, and the Request-to-Send (RTS') flag, and clears any interrupt pending from the controller. However, the controller interrupt mask flag remains unchanged. This command is used at start-up time and to re-initialize after an error condition and must be issued immediately after a controller error condition (Data Set Error, Overrun, or Line Break) is detected through execution of a test command. If a Reset command is issued within 2 character times after a WIO 1 command, data can be sent incorrectly.

This command also resets Data Terminal Ready (DTR) in those Model 707 controllers configured for the optional DTR control.

CIO 2 Set Transmit Mode (C92n)

This command is used to place the controller in the Transmit mode. When executed, this command causes the controller to assert the signal RTS for external use. Receive data is inhibited and, if the controller is unmasked and the TPU is enabled, an interrupt will occur when the external Clear-to-Send (CTS) signal is received by the controller.

CIO 3 Set Receive Mode (C93n)

This command places the controller in the Receive mode and should be issued by the TPU within one character time after execution of the last Write Data command. As a consequence of turn-around from the Transmit to the Receive mode, a delay is initiated to permit all characters in the current send message to be transmitted. Therefore, once the Receive mode is entered, the Transmit mode must not be entered for three character times thereafter in order that the previous Transmit mode can properly terminate.

CIO 4 Set Line Break or Set RTS' (C94n)

For all asynchronous controllers except the Models 703 and 706, this command causes the controller to establish a line break (spacing) condition on the communication line. A line break condition can be tuned by issuing a CIO Line Break command followed by Write Data (WIO) commands writing hex FF characters where each character written equals one character time of line break.

When used with the Models 703 and 706 controllers, which are designed especially for communication with IBM 2260 based system, this command also performs the Set RTS' function so as to maintain symmetry with the CIO Reset RTS'.

CIO C Mask Interrupts (C96n)

This command, when issued, prevents the controller from interrupting the TPU. If the controller becomes ready to interrupt while masked, the interrupt is saved until an Unmask Interrupts command is issued. The masked condition remains in force until an Unmask Interrupts command is executed.

CIO 8 Unmask Interrupts (C98n)

This command, when issued, permits the controller to interrupt the TPU only when the TPU is enabled for interrupt processing. The unmasked condition remains in force until a Mask Interrupts command is issued, or a system reset (IOR instruction, see Appendix A) is executed.

Asynchronous Controller Data Transfer and Control Commands

The primary purpose of this command category is to implement I/O operations. However, some controller operation functions concerning 2260 operations on the part of the Model 707 controller are also implemented. This instruction format for these commands is shown in Figure 2-1. The commands in this category are summarized in Table 2-1 and discussed and defined in detail below.

WIO 1 Write Data (CB1n)

This command, when issued, first asks the controller if it is ready to accept a character for transmission. If the controller is ready, it acknowledges this condition with an ACK which in turn causes the TPU to transfer a character to the controller buffer for transmission. Receipt of a character from the TPU forces the controller into the not-ready condition until the character has been transmitted and the controller buffer becomes clear.

If the controller is not ready when the WIO Write Data command is executed, it does not respond with an ACK and the AUTO-EXEC transfers control to

background. When the controller buffer does become clear, and the controller is unmasked and the TPU is enabled, the controller will interrupt the TPU to initiate the issuing of another WIO Write Data command.

WIO 2 Set Data Terminal Ready (CB2n)

This command, when issued, causes a Model 707 asynchronous controller congifured for Data Terminal Ready (DTR) control to assert the DTR signal line for use by external equipment such as auto-answer and auto-call equipment. The controller always acknowledges this command with an ACK.

WIO 4 Reset Data Terminal Ready (CB4n)

This command, when issued, causes a Model 707 asynchronous controller configured for Data Terminal Ready (DTR) control to reset the DTR signal line for use by external equipment such as auto-answer and auto-call equipment. The controller always acknowledges this command with an ACK.

WIO 8 Reset RTS' (CB8n)

This command resets the Request-to-Send (RTS') flag on a Model 707 asynchronous communications controller which is designed especially for use in 2260 based terminal systems. This command is invalid on the remaining controller versions.

Note: the models 702, 703, 705, and 706 asynchronous controllers are not designed to handle any WIO commands other than WIO 1. Therefore the WIO 2, WIO 4, and WIO 8 commands should not be issued to these controllers because they will not be ACKed.

RIO 1 Read Data (CA1n)

This command, when issued, asks the controller if it is ready to transfer a received data character to the TPU. If the controller is ready, it acknowledges this condition with an ACK, then transfers the received data character from the controller buffer to the TPU. Transfer of a character to the TPU forces the controller into the not-ready condition until the next received data character has been assembled from the communications line and stored in the controller buffer.

If the controller is not ready when the RIO command is executed, it does not respond with an ACK and the Auto-EXEC transfers control to the background. When a received data character has been assembled in the controller buffer, and the controller is unmasked and the TPU is enabled, the controller will interrupt the TPU to initiate the issuing of another RIO command.

Asynchronous Controller Test Commands

This command category permits the programmer to test for the presence of an asynchronous controller and for various operational error conditions. Each of these TIO commands can be implemented by either a JTACK or a JFACK instruction (see Figure 2-2). If the JTACK form is executed, a jump is made to the location specified by the Jump Address field when the controller responds with an ACK. If the JFACK form is executed, a jump is made to the location specified by the Jump Address field when the controller does not respond with an ACK. The commands in this category are summarized in Table 2-1 and discussed and defined in detail below.

TIO 0 Acknowledge If Controller Present (C80n or CC0n)

This test command is used to determine if the asynchronous controller is present. If the controller is present, it will acknowledge execution of this command with an ACK.

TIO 1 Acknowledge If Data Set On Line (C81n or CC1n)

This test command is used to determine if the Data Set Ready (DSR) signal from the modem is present. If the controller detects the presence of DSR, it responds to this command with an ACK.

TIO 2 Acknowledge If No Data Set Error (C82n or CC2n)

This test command is used to detect a lost or intermittent carrier signal while in the Receive mode, or a loss of Clear-to-Send (CTS) in the Transmit mode. If one of these conditions is detected, the controller will not respond with an ACK. A CIO Reset command (C91n) must be issued if it is necessary to re-issue this command.

TIO 4 Acknowledge If No Overrun (C84n or CC4n)

This test command is used to detect an overrun condition which indicates the loss of one or more received data characters. An overrun condition occurs when the controller did not complete transfer of the current received data character to the TPU before the next received data character was ready for transfer. When this command is executed, the controller responds with an ACK if an overrun has not occurred. Either a CIO Reset (C91n) or a CIO Set Transmit Mode (C92n) command must be issued if it is necessary to re-issue this command.

TIO 8 Acknowledge If No Line Break (C88n or CC8n)

This command is used to detect a line break condition during operation in the Transmit mode. A line break occurs when the controller detects a start bit on the received data line from the data set when the controller is in the process of transmitting stop bits. A line break, when detected, can indicate that the equipment on the opposite end of the communication line is trying to interrupt transmission from this TPU, or, in current line communication, that a network failure has occurred. When this command is executed, the controller responds with an ACK if a line break is not detected. This test continues to be valid for one character time after an overrun has been detected. Also, this test can be made in both foreground and background.

Asynchronous Controller Command Merging

Inspection of the binary function codes for CIO and TIO commands (see Table 2-1) will show that command functions are defined exclusively by either the two LSB's or the two MSB's of the function code field. Given this structured allocation of function code bits, it is possible to merge two different commands so that two different functions can be simultaneously performed.

Table 2-2 summarizes the extent to which the various CIO commands can be merged to programming advantage. When merged, CIO commands are executed, the function codes are ORed together to implement the two functions simultaneously.

TABLE 2-2

SUMMARY OF MERGED CIO COMMANDS

Merged CIO Commands	Hexadecimal Model
General Reset plus Mask Interrupts	C9Dn
General Reset plus Unmask Interrupts	C99n
Set Transmit Mode plus Mask Interrupts	C9En
Set Transmit Mode plus Unmask Interrupts	C9An
Set Transmit Mode plus Set Line Break	C96n
Set Receive Mode plus Mask Interrupts	C9Fn
Set Receive Mode plus Unmask Interrupts	C9Bn

Since the function code for TIO commands are formed by a one-out-of-four bit code, and each command performs a separate and exclusive function, these commands can be merged in any combination desired. In this case, the function bits are logically ANDed to implement the pertinent functions. For example, the merged command C8Fn (JTACK) or CCFn (JFACK) can be executed to simultaneously test for all asynchronous controller error conditions.

TABLE 2-3

SYNCHRONOUS CONTROLLER COMMAND SUMMARY

Command Mnemonic	Command Function	Function Code Hex	Function Code Binary	Hexadecimal Model
CIO	General Reset	2	0010	C92n
CIO	Mask Interrupts	4	0100	C94n
CIO	Unmask Interrupts	8	1000	C98n
CIO	Set Transmit Mode	A	1010	C9An
CIO	Set Receive Mode	6	0110	C96n
WIO	Write Data	0	0000	CB0n
RIO	Read Data	0	0000	CA0n
TIO	Acknowledge If Controller Present	0	0000	C80n or CC0n
TIO	Acknowledge If No Data Set Error	2	0010	C82n or CC2n
TIO	Acknowledge If No Overrun	4	0100	C84n or CC4n
TIO	Acknowledge If Data Set On Line	8	1000	C88n or CC8n

SYNCHRONOUS CONTROLLER COMMAND SET

This discussion concerns the command set to which the synchronous controller responds in non-multiplexer configurations. These commands are summarized in Table 2-3. Although the command set for the synchronous communication controller is similar in many respects to the asynchronous controller command set, omissions, differences in function code assignments, and in some cases the results of execution, requires a separate discussion. All non-multiplexer commands for the asynchronous controller are equally applicable to all synchronous controller versions.

Like the asynchronous controller, the command set for the synchronous controller is formed by three command categories: these are control (CIO) commands, data transfer (WIO, RIO) commands and test for condition (TIO) commands.

Synchronous Controller Control Commands

This command category allows the programmer to reset the controller and set it up for I/O operations and interrupt processing. The instruction format for these CIO commands is shown in Figure 2-1. The commands in this category are summarized in Table 2-3 and discussed and defined in detail below.

CIO 2 General Reset (C92n)

This command clears all synchronous controller error conditions and resets the Request-to-Send flag. Any pending interrupt from the controller is cleared, however, the interrupt mask flag remains unchanged. This command is used at start-up time and to re-initialize after detection of a controller error condition through execution of a test condition command. When used to re-initialize after error

detection in the Receive mode, a CIO Reset command should be issued within one character time after error detection. In the Transmit mode, however, a two-character time delay should be initiated after transmission of the last character before issuing a CIO Reset command in order to prevent loss of transmitted data.

CIO 4 Mask Interrupts (C94n)

This command, when issued, prevents the controller from interrupting the TPU. If the controller becomes ready to interrupt while masked, this condition is saved until an Unmask Interrupts command is issued. The masked condition remains in effect until an Unmask Interrupts command is executed.

CIO 8 Unmask Interrupts (C98n)

This command, when issued, permits the controller to interrupt the TPU, only when the TPU is enabled for interrupt processing. The unmasked condition remains in effect until a Mask Interrupts command is issued, or a system reset (IOR instruction, see Appendix A) is executed.

CIO 9 Set Transmit Mode (C96n)

This command is used to place the controller in the Transmit mode. The issuing of this command also clears all controller error conditions. In this mode the controller continuously asserts the signal RTS for external use. When the external signal CTS is received, the controller will interrupt the TPU if the controller is unmasked and the TPU enabled. In the Transmit mode, received data from the TPU is inhibited.

CIO 6 Set Receive Mode (C96n)

This command is used to place the controller in the Receive mode. The issuing of this command also clears all controller error conditions. As a consequence of turn-around from the Transmit to the Receive mode, a delay is initiated to permit all characters in the current send message to be transmitted. Therefore, once the Receive mode is entered, a return to the Transmit mode must be delayed at least three character times to permit the previous Transmit mode to properly terminate.

When in the Receive mode, the controller is always in one of two conditions; namely "Sync Search" or "Sync Found". In the Sync Search condition, the controller monitors the communication line for the presence of the carrier signal and for a synchronization pattern which is one or more hardware defined sync characters. When a sync character or characters followed by a non-sync or data character is detected, the controller reverts to the Sync Found condition. At this point, if the controller is unmasked and the TPU is enabled, the controller will interrupt the TPU.

Synchronous Controller Data Transfer Commands

The synchronous communications controller responds to two data transfer commands: RIO Read Data, and WIO Write Data. The instruction format for these commands is shown in Figure 2-1. Both of these I/O commands are summarized in Table 2-3 and discussed and described in detail below.

WIO 0 Write Data (CB0n)

This command, when issued, asks the controller if it is ready to accept a character for transmission. If the controller is ready, it acknowledges a ready condition with an ACK. This acknowledgment then causes the TPU to transfer a character to the controller buffer for transmission over the communications line. The receipt of this character by the controller forces it into the not-ready condition until the buffer is cleared and the character transmitted.

If the controller is not ready when the WIO is executed, it does not respond with an ACK and Auto-EXEC transfers program control to the background. When the controller does become ready and it is unmasked and the TPU is enabled, the controller will interrupt the TPU to initiate the issuing of another WIO command.

RIO 0 Read Data (CA0n)

This command, when issued, asks the controller if it is ready to accept a received data character to the TPU. If ready, the controller acknowledges this condition with an ACK. It then transfers the received data character from the controller buffer to the TPU. This transfer forces the controller into the not-ready condition until the next received data character has been stored in the controller buffer.

If the controller is not ready when an RIO command is executed, it does not respond with an ACK causing

Auto-EXEC to transfer program control to the background.

When a received data character is stored in the controller buffer, and the controller is unmasked and the TPU is enabled, the controller will interrupt the TPU to initiate the issuing of another RIO command.

Synchronous Controller Test Commands

This command category permits testing for the presence of a synchronous controller and for various operational and error conditions. As with the TIO commands for the asynchronous controller, each of the synchronous controller TIO commands can be implemented by either a JTACK or a JFACK instruction (see Figure 2-2). If the JTACK form is executed, a jump is made to the location specified by the Jump Address field when the controller responds with an ACK. If the JFACK form is executed, a jump is made to the location specified by the Jump Address field when the controller does not respond with an ACK. Synchronous controller TIO commands are summarized in Table 2-3 and discussed and defined in detail below.

TIO 0 Acknowledge If Controller Present (C80n or CC0n)

This test command is used to determine if the synchronous controller is present. If the controller is present, it will acknowledge the issuing of this command with an ACK.

TIO 2 Acknowledge If No Data Set Error (C82n or CC2n)

This test command is used to detect the absence of a carrier signal in the Receive mode during Sync Search, and a lost or intermittent carrier signal in the Sync Found condition. In the Transmit mode, this command tests for the loss of CTS. If none of these conditions is detected, the controller will respond with an ACK. If it is necessary to re-issue this command, the re-issued command should be preceded by either a CIO Set Receive mode or a CIO Set Transmit mode command, whichever is appropriate.

TIO 4 Acknowledge If No Overrun (C84n or CC4n)

This command is used to detect an overrun condition which occurs when the transfer of a send or received character between the TPU and the controller was not completed before the next character was transferred. If an overrun occurs in the Receive mode, a character is lost, however, character synchronization with the data set is maintained. When an overrun occurs in the Transmit mode, the controller transmits all ONEs over the communication line until a valid character is ready for transmission or the controller is placed in the Receive mode. During an overrun in the Transmit mode, character synchronization with the data set is maintained. In both modes, the controller responds to execution of this command with an ACK only if an overrun did not occur. If it is necessary to re-issue this command, the re-issued command should be preceded by either a CIO Set Receive Mode or a CIO Set Transmit Mode, whichever is appropriate.

TIO 8 Acknowledge If Data Set On-Line (C88n or CC8n)

This command is used to determine if the DSR signal from the modem is present. If the controller detects the presence of this signal, it responds to this command with an ACK.

Synchronous Controller Command Merging

Unlike the asynchronous controller, the function code bit patterns for synchronous controller CIO commands preciudes command merging. However, TIO commands for the synchronous controller can be merged since the TIO function codes are formed by a one-out-of-four bit code. TIO commands can be merged in any combination desired. The function bits are logically ANDed to implement the pertinent functions. For example, the merged command C8En (JTACK) or CCEn (JFACK) can be executed to simultaneously test for all synchronous controller error conditions.

PARTY LINE CONTROLLER COMMAND SET

The command set for the party line controller is a sub-set of the asynchronous controller command set. Those asynchronous controller commands used by the party line controller perform exactly the same functions on both controllers. The asynchronous controller commands applicable to use by the party line controller are summarized in Table 2-4. Note that CIO and TIO commands can be merged in the same manner as with the asynchronous controller (see Table 2-2).

TABLE 2-4

PARTY LINE CONTROLLER COMMAND SUMMARY

Command Mnemonic	Command Function	Function Code Hex	Binary	Hexadecimal Model
CIO	General Reset	1	0001	C91n
CIO	Set Transmit Mode	2	0010	C92n
CIO	Set Receive Mode	3	0011	C93n
CIO	Set Line Break	4	0100	C94n
CIO	Mask Interrupts	C	1100	C9Cn
CIO	Unmask Interrupts	8	1000	C98n
WIO	Write Data	1	0001	CB1n
RIO	Read Data	1	0001	CA1n
TIO	ACK If Controller Present	0	0000	C80n or CC0n
TIO	ACK If No Overrun	4	0100	C84n or CC4n
TIO	ACK If No Line Break	8	1000	C88n or CC8n

Unlike the asynchronous and synchronous controllers, which have general application across the range of telecommunications systems, the party line controller represents a special case. This controller is designed specifically for use with terminals operating without modems in local multi-drop system configurations.

In order to clearly define the role of the party line controller in such system applications, a representative terminal system based on the use of this controller is discussed herein. This representative system is designed for use in an airlines reservation environment. As shown in Figure 2-3 such a system generally consists of a supervisor's set and up to twelve agents' sets. At the supervisor's set a supervisor can, through keyboard input, address any one of the agents' sets in order to monitor input using a specific addressing convention. Then, by pressing a special key at the supervisor's keyboard, the current display at the addressed agent's set, including the state of the keyboard indicator, can be continuously displayed on the CRT at the supervisor's set. This data transfer is performed by separate I/O programs, at the supervisor's set and at the addressed agent's set, using the party line controllers at each terminal.

In this type of system, the supervisor can terminate one agent's display and address another agent's set at will. In addition, the supervisor has the option to operate his set as an agent's set.

Each agent's set operates in a monitoring mode whereby it is always open to being addressed by the supervisor's set for monitoring. In this mode, the party line controller at each agent's set is continuously looking for its address on the line and, upon detecting its address, will interrupt the background program at the addressed terminal. As a consequence, Auto-EXEC at this terminal transfers program control to the foreground to initiate transfer

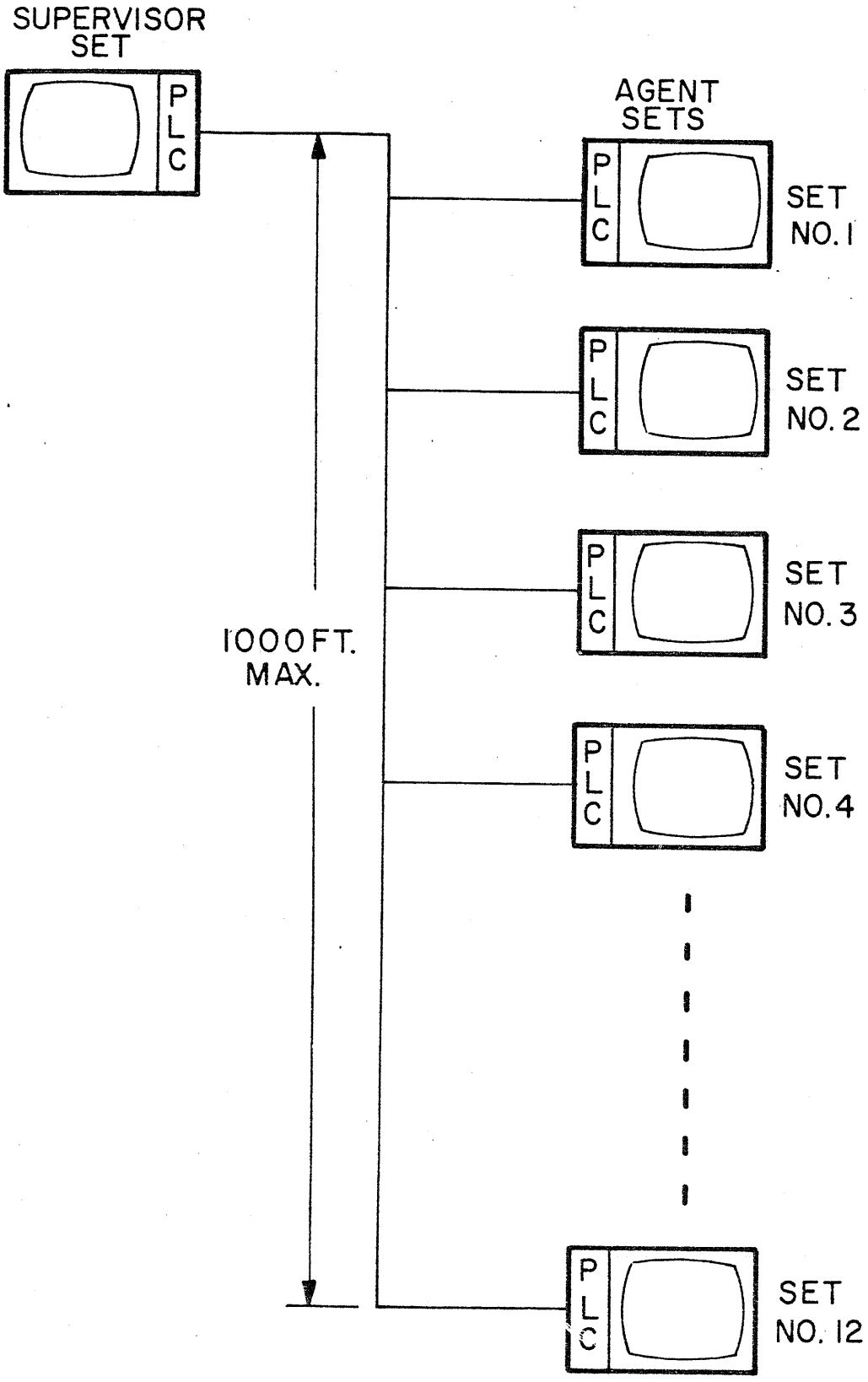


Figure 2-3. Sample Reservation System

of the display and indicators on a continuing basis to the supervisor's set.

Note that all terminals in such a system can also be equipped with either asynchronous or synchronous controllers for communication with a remote data base. The terminals in such a system can connect to individual modems or be multiplexed by an SPD-M multiplexer into a single modem.

MULTIPLEXER COMMAND SET

The SPD-M multiplexer is designed to be used with both the asynchronous and synchronous controllers and operate in conjunction with its associated terminals in a software polling environment. The terminal selection process performed by the multiplexer can be cyclical or be determined by a hardware implemented priority scheme.

SPD-M multiplexers can be configured for two modes of operation; namely single transmission operation and multiple transmission operation. The major operational difference between these two configurations involves the disposition of polling message replies from associated terminals. Single transmission configurations can handle a reply from only one terminal per poll, whereas with multiple transmission configurations, the number of replies per poll is limited only by the number of multiplexed terminals.

Terminals associated with the SPD-M multiplexer are selectively addressed by software generated message envelopes containing characters which define the addressing hierarchy. This open ended structure permits the implementation of virtually any general or selective terminal addressing scheme.

The SPD-M multiplexer is completely transparent to all data and control codes. Received data is presented to all associated terminals by the multiplexer with the address message contained in the poll designating the receiving terminal.

In the Transmit mode, each terminal can be selected by the multiplexer either cyclically or according to a hardware priority scheme. When selected, the terminal has the option of responding to the poll with data or with a message acknowledging the poll and indicating that the terminal has no data to send.

Note that the multiplexer will deselect, or cancel a terminal after five seconds of data transmission time. If such a time-out occurs, the terminal cannot be selected again until a general reset is manually executed at the multiplexer. This time-out function can be optionally eliminated. The timer is used to prevent one terminal from tying up the line in the event of a failure.

The command sets for both the asynchronous and synchronous controllers when used with the SPD-M multiplexer each have several commands redefined in order to properly operate with the multiplexer. The discussions that follow describes these redefined commands for both communications controllers in terms of multiplexer operation. Program logic pertinent to the use of the SPD-M multiplexer in conjunction with either communications controller in a software polling environment, is presented following the discussions of controller/multiplexer command sets.

COMMUNICATIONS CONTROLLER/MULTIPLEXER COMMAND SET

Table 2-5 summarizes the complete command set recognized by an asynchronous controller when part of a terminal system which time shares the communications line through an SPD-M multiplexer. Table 2-6 summarizes the complete command set recognized by a synchronous controller operating in the same system circumstances.

Set Priority Data

This command is executed in response to a poll when the terminal has actual data to transmit; in contrast to a reply to the poll indicating no data to transmit. When executed, this command sets the Priority Data flag at the multiplexer which permits the multiplexer to distinguish between terminals having no data to send, but volunteering poll responses and a terminal having an actual data message to transmit.

In multiple transmission operation, the multiplexer will transmit data in response to a poll from the first terminal it encounters with its Priority Data flag set. The multiplexer notifies the terminal to begin transmission by asserting CTS at that terminal. All other terminals raising their priority data flags in response to the same poll will be taken in order of encounter. Single transmission multiplexers will allow data transmission in response to a given poll from the