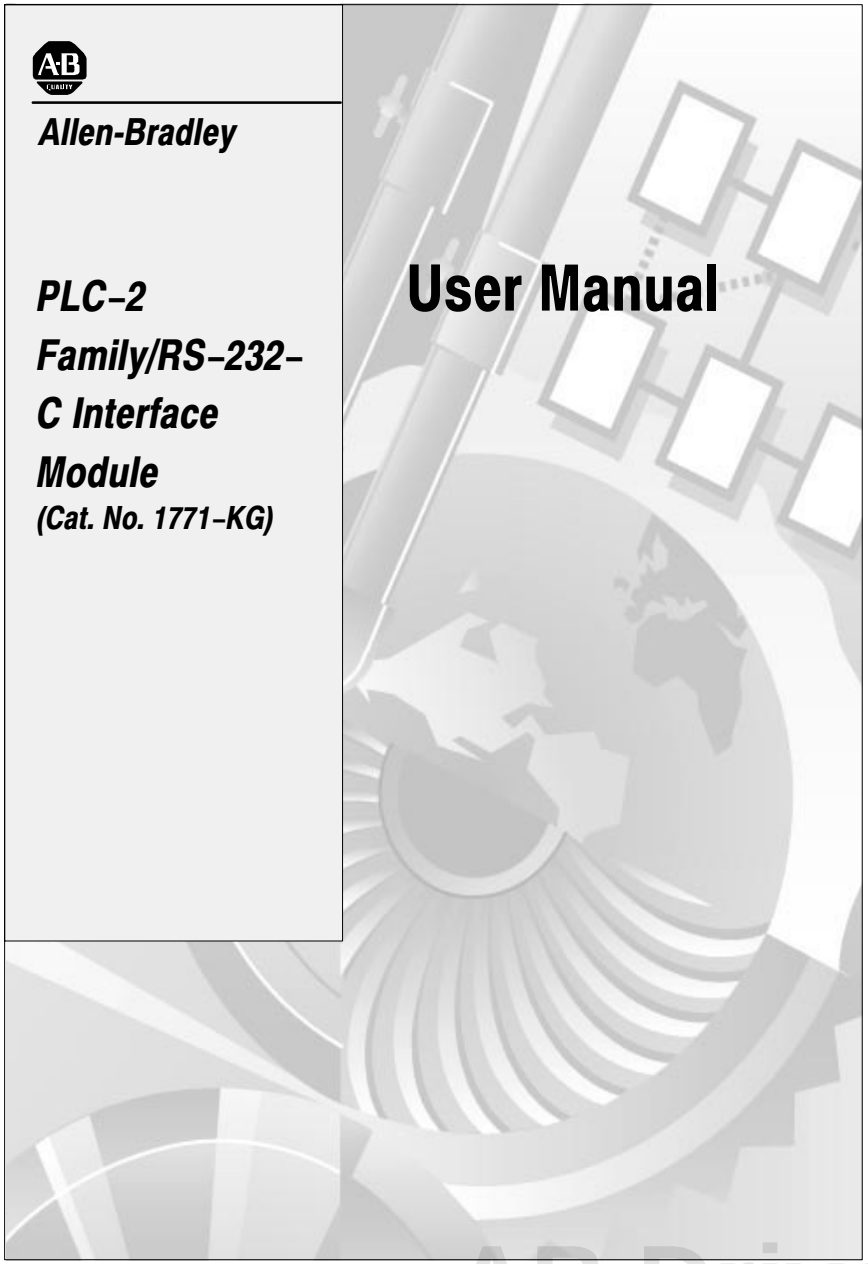


Allen-Bradley

PLC–2 Family/RS–232–C Interface Module

(Cat. No. 1771–KG)

User Manual

Table of Contents

Chapter 1

Introduction

General

The PLC-2 Family/RS-232-C Interface Module (Cat. No. 1771-KG) performs an interface function between a PLC-2 Family processor and an intelligent RS-232-C compatible device to provide:

- Direct peer-to-peer or master/slave communication with a computer or other programmable intelligent device, such as the Advisor™ Color Graphic System.
- Direct peer-to-peer communication with an Allen-Bradley communication module that provides a complementary interface function for another PC processor.
- Interface with a Communication Controller Module (Cat. No. 1771-KE, -KF) for peer-to-peer communication with other stations on an Allen-Bradley Data Highway.

Document Organization

This manual describes installation, operation, and programming necessary to use the 1771-KG module. This chapter provides a general description and lists specifications. Subsequent chapters are organized as follows:

- Chapter 2 outlines various network configurations and application considerations for them.
- Chapter 3 provides an operational overview of communication concepts. This includes message structures, error checking, data security, and Data Highway link protocol.
- Chapter 4 describes module hardware, outlines procedures for preparation, installation, and connection of the module.
- Chapter 5 describes the PC user programming of communication zone rungs needed for communication through the 1771-KG module.
- Chapter 6 describes the protocol the module uses in communication with a computer through an RS-232-C link.
- Chapter 7 outlines start-up and troubleshooting procedures.
- Appendix A lists Data Highway error codes and their meaning.
- Appendix B describes the use of diagnostic counters for analyzing communication reliability.
- Appendix C provides detailed information useful for encoding address and data fields at a computer before sending them to a PC in message packets.

Compatible Processors

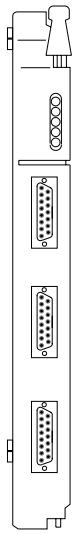
The module is compatible with the following Bulletin 1772 processors:

- Mini-PLC-2 Processor (Cat. No. 1772-LN1, -LN2, -LN3)
- Mini-PLC-2/15 Processor (Cat. No. 1772-LV)
- PLC-2/20 Processor (Cat. No. 1772-LP1)
- PLC-2/20 Processor (Cat. No. 1772-LP2)
- PLC-2/30 Processor (Cat. No. 1772-LP3)

Module Description

The 1771-KG module is shown in Figure 1.1. Install the module in any I/O module slot of a Bulletin 1771 I/O chassis located within 10 cable-feet of the PC processor.

Figure 1.1
PLC/2 Family/RS-232-C Interface Module (Cat. No. 1771-KG)



10862-I

Connections

The module receives its power through the chassis backplane. However, all communication is through the three connectors on the front of the module (Figure 1.2).

The PROCESSOR connector provides cable connection to the PC processor. At the PC processor, the cable mates with the connector provided for industrial terminal use.

The PROGRAM INTERFACE connector provides cable connection to an industrial terminal for the PC processor.

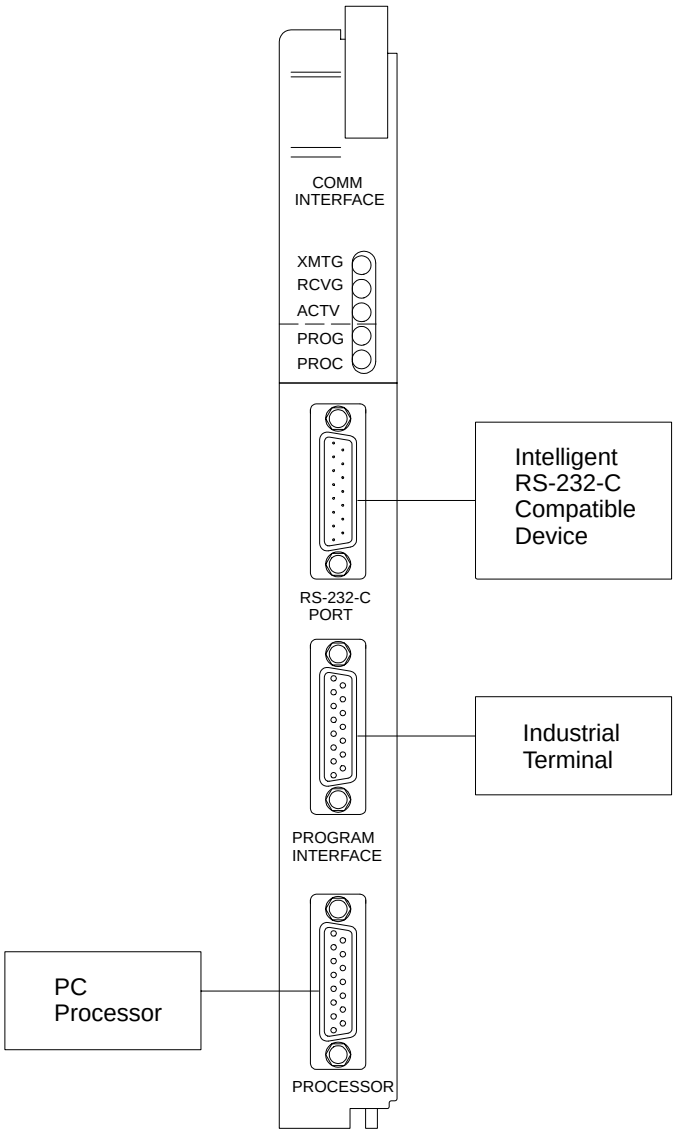
The RS-232-C PORT connector provides cable connection to an intelligent RS-232-C compatible device. To allow communication with a computer more than 50 cable-feet from the module, a MODEM link must be used. To allow communication with another Allen-Bradley communication module more than 7,000 cable-feet from the module, a MODEM link must be used.

Indicators

The module has five LED indicators:

- XMTG (Transmitting) — This green indicator is ON when the module is transmitting a message through the RS-232-C port. (Chapter 6 defines which transmissions are messages.)
- RCVG (Receiving) — This green indicator is ON when the module is receiving a message through the RS-232-C port.
- ACTV (Active) — This green indicator is ON when a cable is connected from an RS-232-C compatible device to the receiving pins of the RS-232-C PORT connector and power is on at both ends.
- PROG (Program Status) — This red indicator should go ON momentarily at power-up and when the PC processor is switched into a test or run mode. This indicator will stay ON if the module detects a format error in the communication zone of the ladder diagram program.
- PROC (Processor Status) — This red indicator will go ON when the module is not connected to the PC processor.

Figure 1.2
Communication Connections



Series B Enhancements

This manual describes both the Series A and Series B versions of the 1771-KG module. Table 1.A lists the Series B enhancements to the module. These features were not available on the Series A version of the module. Where we describe these features later in the manual, we note that they are only available on the Series B version of the module. You can see the catalog number and the series level of the module on the nameplate on the side of the module.

Table 1.A
Series B Enhancements

Feature	Benefit
Set Data Table Size Command	This command provides your computer with the capability to change the data table size during a procedure for downloading the PLC-2-family processor's memory.
Second Link Capability	This capability can provide two RS-232-C links to a PLC-2 family processor through two 1771-KG modules. It can alternately provide an RS-232-C link through a 1771-KG module and a Data Highway link through a 1772-KA2 module.
Cyclic Redundancy Check (CRC) (Switch Selectable)	The CRC provides more complete error checking than the block check character (BCC) check.
Separate Command to Enter Upload Mode	This command provides a simpler upload procedure.

Specifications

Specification	Requirements
Electrical Interface	• RS-232-C Compatible
Communication Protocol	• Full-Duplex (for Peer-to-Peer Communication)
	• Half-Duplex (for Master-Slave Communication)
Communication Rates	• 110, 300, 600, 1200, 2400, 4800, 9600, or 19.2K bits/s
Module Location	• Any I/O Module Slot of a 1771 I/O Chassis
Backplane Power Supply Load	• 1.0A
Ambient Temperature Rating	• Operational: 0º to 60ºC (32º to 140ºF)
	• Storage: -40º to 85ºC (-40º to 185ºF)
Relative Humidity Rating	• 5% to 95% (without condensation)
Keying (Top Connector)	• Between 4-6
	• Between 22-24

Chapter 2

Application Considerations

General

In this chapter we describe how the 1771-KG module relates to other Allen-Bradley modules which provide general communication functions for PC processors. We outline various networks that you might configure and application considerations for them.

Communication Modules

The following Allen-Bradley modules provide general communication functions for PC processors:

- PLC Computer Interface Module (Cat. No. 1774-CI2)
- PLC Communication Adapter Module (Cat. No. 1774-KA)
- PLC-2 Family Communication Adapter Module (Cat. No. 1771-KA, -KA2)
- PLC-3 Communication Adapter Module (Cat. No. 1775-KA)
- PLC-4 Communication Interface Module (Cat. No. 1773-KA)
- PLC-2 Family/RS-232-C Interface Module (Cat. No. 1771-KG)
- Communication Control Module (Cat. No. 1771-KC, -KD, -KE, -KF)

PC-Processor/Data Highway Interface

The following modules provide an interface function between a PC processor and a Data Highway communication link:

- PLC Communication Adapter Module (Cat. No. 1774-KA)
- PLC-2 Family Communication Adapter Module (Cat. No. 1771-KA)
- PLC-3 Communication Adapter Module (Cat. No. 1775-KA)
- PLC-4 Communication Interface Module (Cat. No. 1773-KA)

PC-Processor/RS-232-C Interface

The following modules provide an interface function between a PC processor and an RS-232-C communication link:

- PLC-3 Communication Adapter Module (Cat. No. 1775-KA)
- PLC-4 Communication Interface Module (Cat. No. 1773-KA)
- PLC-2-Family/RS-232-C Interface Module (Cat. No. 1771-KG)

RS-232-C/Data Highway Interface

The following modules provide an interface function between an RS-232-C communication link and a Data Highway communication link:

- Communication Controller Module (Cat. No. 1771-KC, -KD, -KE, -KF)

The 1771-KC and 1771-KE modules must be installed in an I/O chassis. The 1771-KD and 1771-KF modules are stand-alone modules.

The 1771-KC and 1771-KD modules provide peer-to-peer communication through an RS-232-C link. They are superseded by the 1771-KE and 1771-KF modules which provide either peer-to-peer or master/slave communication through an RS-232-C link and provide hardware handshaking.

Data Highway Network

The Data Highway is a local area network (LAN) which can allow peer-to-peer communication between up to 64 stations. A Data Highway network is illustrated in Figure 2.1.

Stations

A station is made up of a computer or PC processor and the module or modules that interface it with the Data Highway link. You must assign a unique number to each station at each communication module.

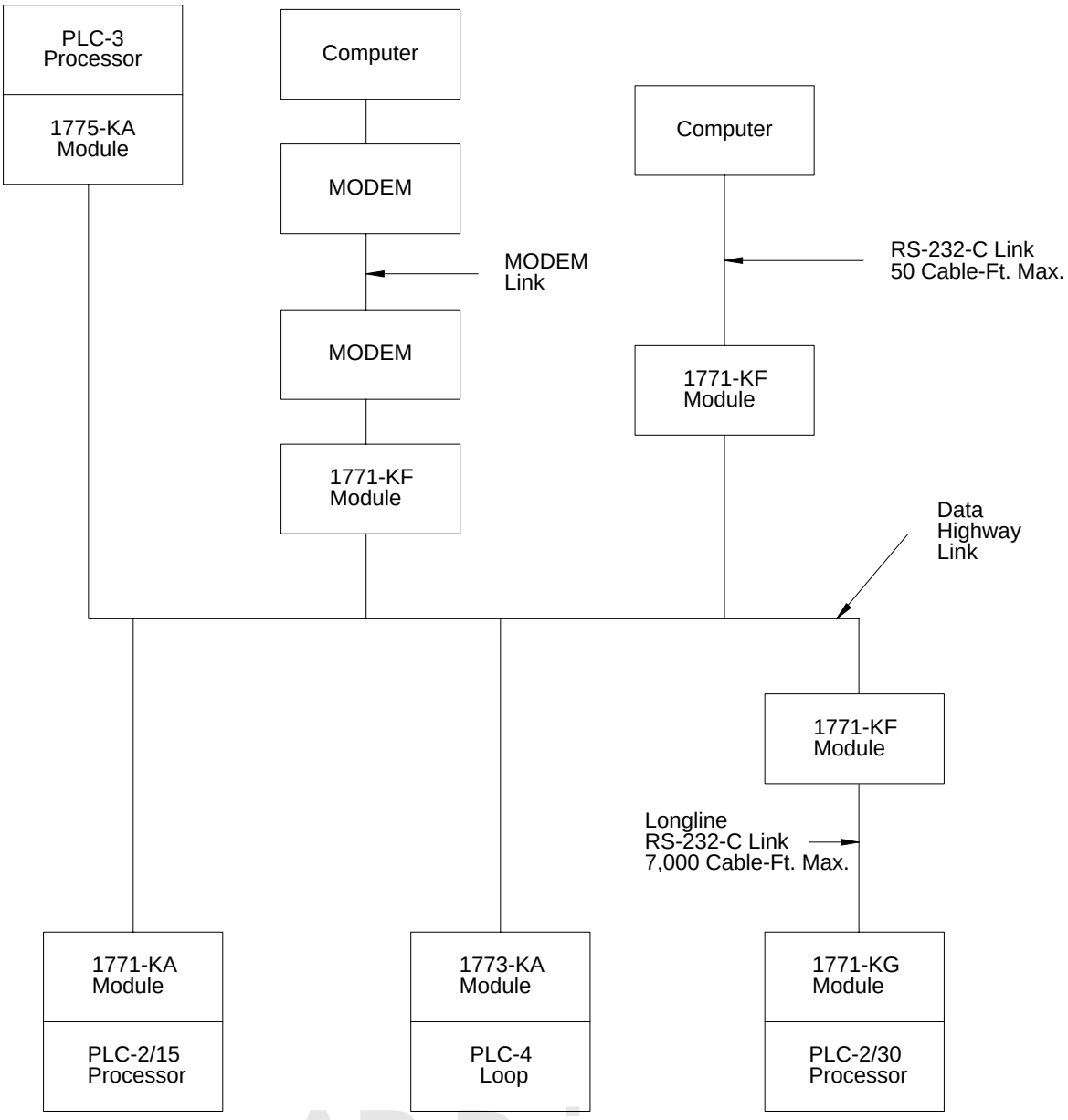
Within a station that contains a communication controller module, a link is required that is auxiliary to the Data Highway link. Three such stations are shown in Figure 2.1.

- One station is comprised of a 1771-KF module interfacing with a computer through an RS-232-C link limited to 50 cable-feet.
- Another station is comprised of a 1771-KF module interfacing with a computer through a MODEM link limited only by the nature of the MODEM link.
- The third station is comprised of a 1771-KG module interfacing a PLC-2/30 processor with a 1771-KF module through a longline RS-232-C link limited to 7,000 cable-feet. If a longer distance is required, a MODEM link could be used.

PC Programming

The ladder diagram program in a PLC-2 family processor at one station can initiate the transfer of a message to or from any other station in the network. The programming of a PLC-2 family processor for communication through a 1771-KG module is described in Chapter 5.

Figure 2.1
Data Highway Network



Data Highway Link

The Data Highway link can have a trunkline of up to 10,000 feet and droplines of up to 100 feet each. Each station is at the end of a dropline.

The peer-to-peer communication through the Data Highway link is accomplished through a modified token passing scheme called floating master. With this arrangement, each station has equal access to become the master. They bid for temporary mastership based on their need to send information.

For a more complete description of the Data Highway link refer to Chapter 3.

Computer Programming

The communication protocol on the Data Highway link is transparent to a computer in the network. However, for a computer to send or receive messages through the Data Highway network, it must be programmed to communicate with its communication controller module over their auxiliary link through the protocol described in Chapter 6.

Stand-Alone Links

A stand-alone communication link is totally separate from any Data Highway link. A stand-alone link also provides a Data Highway network. The network layer protocol is the same as for a Data Highway network which includes a Data Highway link. With a 1771-KG module, a PLC-2 family processor can communicate through a stand-alone link directly to either another PC processor or a computer. You must assign a unique number to each PC station at each communication module.

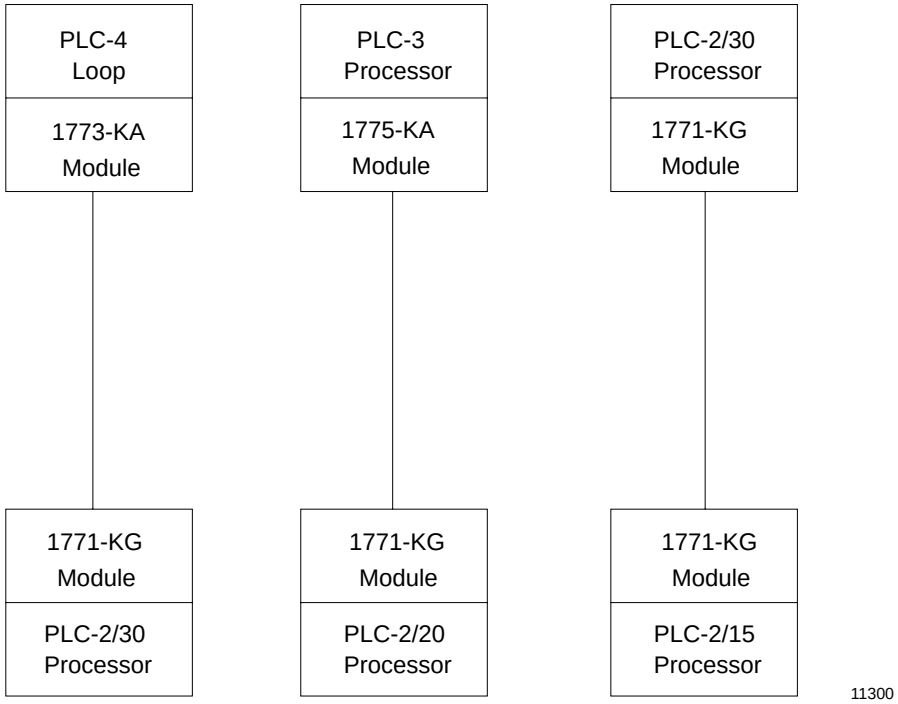
PC-to-PC

Figure 2.2 shows three possible stand-alone PC-to-PC communication links in which the 1771-KG module could be applied. Each is a point-to-point link in which two PC processors can communicate as peers.

The ladder diagram program in a PLC-2 family processor at one station can initiate the transfer of a message to or from the other PC processor. The programming of a PLC-2-family processor for communication through a 1771-KG module is described in Chapter 5.

Each of the PC-to-PC links shown in Figure 2.2 is a longline RS-232-C link limited to 7,000 cable-feet. If you require a longer distance, use a MODEM link.

Figure 2.2
Stand-Alone PC-to-PC Links



PC-to-Computer

With 1771-KG module, a PLC-2 family processor can communicate directly to a computer through either a point-to-point link or a multi-drop link.

Figure 2.3 shows a point-to-point link to a computer. This is an RS-232-C link limited to 50 cable-feet. If you require a longer distance, use a MODEM link.

Figure 2.3
Stand-Alone Point-to-Point Link to a Computer

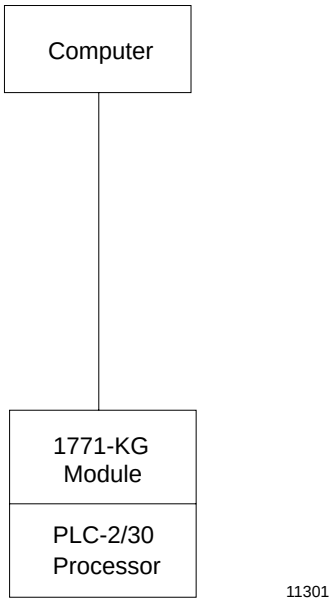
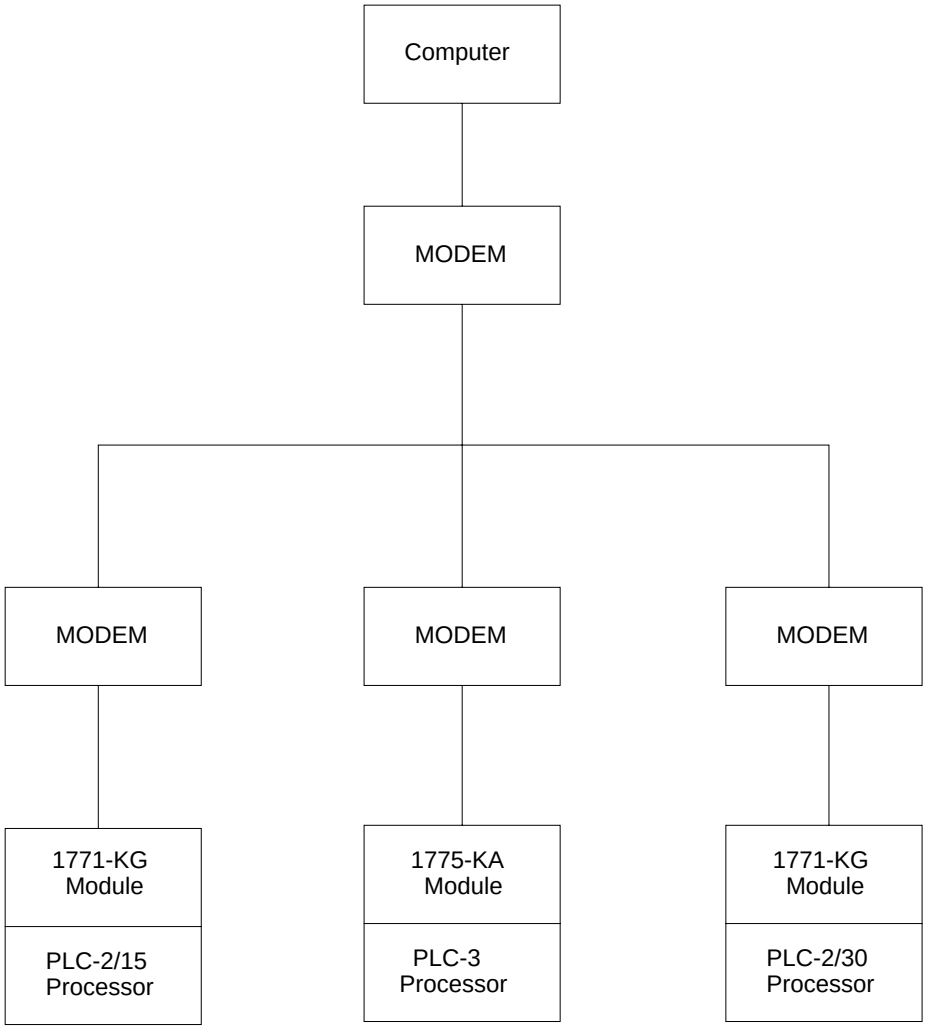


Figure 2.4 shows a multi-drop link between a computer and three PC processors. Each PC processor can communicate with the computer, but not directly with the other PC processors. This type of configuration requires a modem link.

For a point-to-point link, you can use either a peer-to-peer or master/slave communication protocol. For a multi-drop broadband MODEM link, you can use either a peer-to-peer or master/slave communication protocol. For a multi-drop baseband MODEM link, you must use a master/slave communication protocol because the link can support only one channel.

A computer can send or receive messages through a stand-alone link in the same way as through a Data Highway link network. Program the computer to follow the communication protocol described in Chapter 6.

Figure 2.4
Stand-Alone Multi-Drop Link to a Computer



Configuration Selection

Figures 2.1 through 2.4 illustrate several configurations in which a PLC-2 family processor can communicate with computers and/or other PC processors through RS-232-C ports provided by 1771-KG modules. Each is useful, depending on your application.

If you want to provide peer-to-peer communication between a PLC-2 family processor and two or more other stations, use a Data Highway network as shown in Figure 2.1. If the PLC-2-family processor is close enough to the other stations (10,000 cable-feet) you can use a 1771-KA module to interface directly with the Data Highway link. If the distance is too far, you can use a 1771-KG module to interface with a longline

RS-232-C link (or MODEM link) to a 1771-KE/-KF module which in turn interfaces with the Data Highway link.

Even with only two stations, you may want a Data Highway link. A Data Highway link would provide a 10,000 cable-foot link and the flexibility of expansion to add more stations later.

With two stations, you may want a stand-alone link. For a stand-alone link, you could use a MODEM link to provide communication between stations more than 7,000 cable-feet apart. Also, with the full-duplex (peer-to-peer) protocol and embedded responses, it could be faster than a Data Highway link because it wouldn't be burdened with polling. A Data Highway link has a communication rate of 57.6K bits/s and a peer-to-peer, half-duplex, polled protocol. An RS-232-C link has a selectable communication rate of 19.2K bits/s maximum and a selectable protocol of either master/slave, half-duplex, polled; or peer-to-peer, full-duplex, unpolled.

For a stand-alone link to a computer, you may want to use the peer-to-peer communication protocol for maximum speed. However, you may want to use the master-slave communication protocol so that the computer can select the times it will spend communication through the link.

You can select a master/slave communication protocol for any link to a computer. You can select a peer-to-peer communication protocol only for a point-to-point link or a broadband MODEM multi-drop link to a computer.

A Second Link

You can connect two Series B 1771-KG modules together to provide two RS-232-C links through which the PLC-2 family processor can communicate (Figure 2.5). You can alternately connect a Series B 1771-KG module and a Series B 1771-KA module to provide both an RS-232-C link and a Data Highway link through which the PLC-2 family processor can communicate (Figure 2.6).

The first module (connected directly to the PLC-2 family processor) has the same communication throughput as a single module. The second module (connected to the PROGRAM INTERFACE connector of the first module) has a reduced communication throughput.

This capability to provide a second communication link was not available on the superseded Series A version of the module.

Figure 2.5
A Second RS-232-C Link for PLC-2 Family Processor

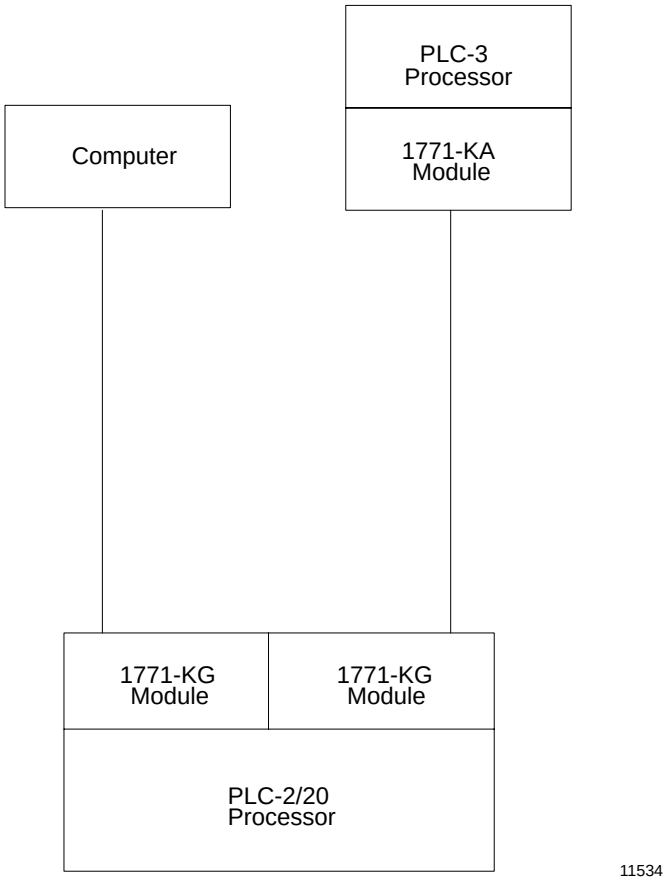
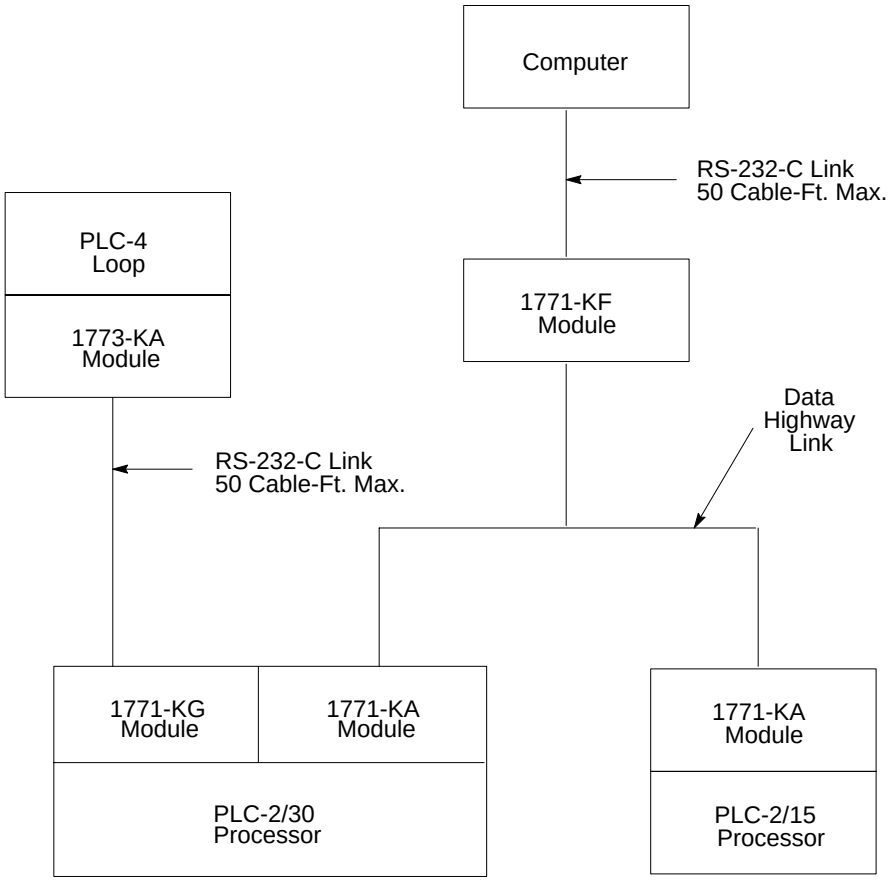


Figure 2.6
An RS-232-C Link and a Data Highway Link for a PLC-2 Family Processor



Chapter 3

Communication Concepts

General

The 1771-KG module can be applied to Data Highway networks which can include a Data Highway link or a stand-alone RS-232-C link. This chapter presents concepts of these communication applications. Message structures, error checking, and data security are discussed. Data Highway link protocol is discussed in general to help you understand the choices you must make before installing and operating your network. RS-232-C link protocol is described in detail in Chapter 6.

Communication Links

A Data Highway link has a communication rate of 57.6k bits/s. It provides peer-to-peer communication between stations through a half-duplex, polled protocol.

A 1771-KG module provides an interface to an RS-232-C link. It has a selectable communication rate of 19.2K bits/s maximum.

The RS-232-C link can be within a Data Highway network as an auxiliary link to the Data Highway link. The RS-232-C link can also be a stand-alone link. For an auxiliary link, or a stand-alone link to another Allen-Bradley communication module, it provides peer-to-peer communication through a full-duplex, unpolled protocol. For a stand-alone link to a computer, you can select it to provide either:

- Peer-to-peer communication through a full-duplex, unpolled protocol.
- Master/slave communication through a half-duplex, polled protocol.

Message Structures

All messages on a Data Highway network have the same fundamental structure, regardless of their function or destination. If you could freeze a message packet while it is in transmission, what you would see would consist of two types of bytes:

- Header Bytes
- Data Bytes

The methods by which these bytes are filled is determined by the nature of the station from which the block originates. For example, if a transaction originates from a PC station, the communication module automatically fills the header bytes with information contained in the PC program's communication zone. If the transaction originates from a

computer station, the computer programs must supply the necessary header information.

Command/Reply Cycle

Any transaction on a Data Highway network consists of two parts:

- Command
- Reply

This provides an extra layer of integrity by ensuring that a required action always returns some sort of status, whether a code or data. As a frame of reference, the command initiator is always referred to as a local station, and a reply initiator is always referred to as a remote station. Unless noted otherwise, whether in a Data Highway link or a stand-alone link, the discussion will be limited to a single local station and a single remote station.

The header distinguishes a command from a reply. Obviously, the data area of a command and its corresponding reply depends on the type of command.

Priority

Each message on a Data Highway link is classified as either:

- High Priority
- Normal Priority

The priority levels of messages determine the order in which stations are polled and allowed to transmit messages across a Data Highway link. In the polling process, stations with high priority messages are given priority over stations with normal priority messages.

You specify the priority level for each command message. The command code contains this specification. The station that receives a command message must establish the same priority level for its corresponding reply message. At a PC processor station, the communication module automatically establishes the priority level of each reply message.

NOTE: Stations with high priority messages are given priority over stations with normal priority messages throughout the command/reply message cycle. For this reason, a command should be given a high priority designation only when special handling of specific data is required. Using an excessive number of high priority commands defeats

the purpose of this feature and could delay or inhibit the transmission of normal priority messages.

Command Structures

There are four basic types of commands on a Data Highway network:

- Read
- Write
- Diagnostic
- Mode Select

Reads

There are two types of reads:

- Physical
- Unprotected

A physical read allows you to read any area of PC memory at a remote station. However, a PC processor cannot originate a physical read command. An unprotected read can access only the data table area of PC memory. Any read can request up to 122 words of contiguous data.

Writes

There are several layers of distinction to be considered regarding writes. Writes are layered both for application reasons and for security.

As an application issue, writes are divided between bit writes and block writes. Bit writes allow the local station to control bits in the data table of a remote station.

Block writes allow the local station to write up to 121 contiguous words of data into the remote station's memory, provided you abide by the restrictions described next.

As with reads, writes also are defined by the level of access to PC memory. Non-physical writes can access only the data table at a remote PC; physical writes can access all of user memory, including the program area of memory.

Non-physical writes can be further subdivided to protected and unprotected. Protected writes are accepted at a remote PC only if you have programmed memory access rungs into the remote PC's communication zone. If the protected write tries to write into an area other than that specified by the memory access rungs, the command is rejected and an error code is returned. Unprotected writes will access any area in a remote PC's data table without needing memory access rungs.

Each of these writes can be selectively blocked at the remote station by setting a switch at the PC communication module.

Diagnostics

Diagnostic commands must originate from a device other than a PC. You can use them to return status from a remote or local station or to alter some on-board parameters at a local station. Diagnostic commands are very useful during a start-up or during on-line debugging. The diagnostic status command is also useful in download procedures.

Mode Select

Mode select allows you to load a new PC program from a remote station. The operation of this command varies by PC processor type. The command can be issued only by a computer.

Summary

Messages can be divided into commands and replies.

REPLY		
COMMAND	Read	• Physical [1] • Unprotected
	Write	• Bit Write -- Protected -- Unprotected • Block Write -- Physical -- Protected [1] -- Unprotected
	Diagnostic [1]	• Counters Reset • Loop • Read • Status • Set Parameters • Set Variables • Set ENQs • Set NAKs • Set Timeout
	Mode Select [1]	• Enter Download Mode • Enter Upload Mode • Exit Download/Upload Mode
	Set Data Table Size [1]	
[1]	You cannot originate these commands from a PC application program. You can program them for transmission from a computer station.	

Error Checking and Data Security

For the most part, the interaction of communication modules is transparent to the user. This means that the application programs at the PCs and computers along the Data Highway network need not involve themselves with inter-station protocol, handshaking, or control of the Data Highway link. This is all carried out automatically by the communication modules. However, an understanding of station interaction is useful both to computer programmers and PC programmers.

Error Checking

When a command cannot be carried out due to a user programming error or a discrepancy in data handled by the module, an error code may be written into a PC data table memory word. The error code storage word is selected by the PC programmer and listed in the header rung of the communication zone of the PC program. This word stores the most recent error code written by the communication module.

Error codes can be generated at two places—the remote station and the local station interface modules. For codes that are returned from a local station module, two types of conditions can exist:

- Application programs use the wrong message format or issue illegal commands.
- The local station cannot complete a transaction due to network problems.

A remote station can return only the codes associated with an application problem at the remote station. Typically these involve either the processor being off-line (in program mode, for example) or the command trying to access memory areas blocked by either the communication module or the communication zone of the PC program.

On the network layer protocol, command message status is returned in a status byte. A value of zero in the status byte indicates successful transmission of the corresponding command. It is up to the local application program to display and act on the value of the returned byte. In PLC-2 family processors, the value of the byte is interpreted by the processor and then displayed in an error word.

Data Security

There are two checks used in message transmission:

- An 8-Bit Block Check Character (BCC)
- A 16-Bit Cyclic Redundancy Check (CRC)

Some station interface modules also let you select a parity check (even parity only).

Block Check Character (BCC)

You can use a block check to detect errors on an RS-232-C link. Any device connected to that link must be capable of generating a BCC.

The sending station adds a BCC to the end of every link packet to help detect errors of transmission within the module. The sending station generates the BCC by first summing every byte of the data field (excluding control characters), then taking the 2's complement of that sum. The result is the BCC. Notice that any final carry-out is ignored in the BCC computations.

The receiving station also sums the data field bytes, then adds that sum to the BCC to produce 00000000. Any sum other than 00000000 indicates an error has been made in the transmission and causes the receiving station to respond with DLE NAK.

Cyclic Redundancy Check (CRC)

The CRC is used to validate packets transmitted on the Data Highway link. With the Series B 1771-KG module, you can use a CRC on an RS-232-C link.

The sending station appends the CRC to the packet, based on the bit pattern of the transmitted data field. The sending station computes a CRC based on the received data and checks this against the CRC value included with the message. A discrepancy between the transmitted CRC and the CRC computed by the receiving station indicates some fault in the transmission. If the received and computed CRC values do not agree, the message is not accepted as valid.

Link Layer Protocol on Data Highway Link

The link layer protocol used between stations on the Data Highway link is a modified low-level implementation of HDLC that features bit stuffing, flag definition, and the generation of the CRC. This protocol is completely transparent to the user. However, it is discussed here in general to help you understand the choices you must make before installing and operating your network.

Floating Master

Central to the interaction of communication modules on the Data Highway link is the concept of the floating master. With this arrangement, no single station is permanent master controlling the Data Highway communication link at all times. Instead, each station bids for mastership, based on its need to send command or reply messages. This arrangement has two major features:

- Multiple Masters
- Peer-to-Peer Communication

One advantage of floating master operation is that no single station disables communication on the Data Highway link as long as other stations continue to operate. This means that even with disconnection or faulted operation of a communication module or a processor, communication between other operating stations continues. This minimizes the need for redundancy in some applications.

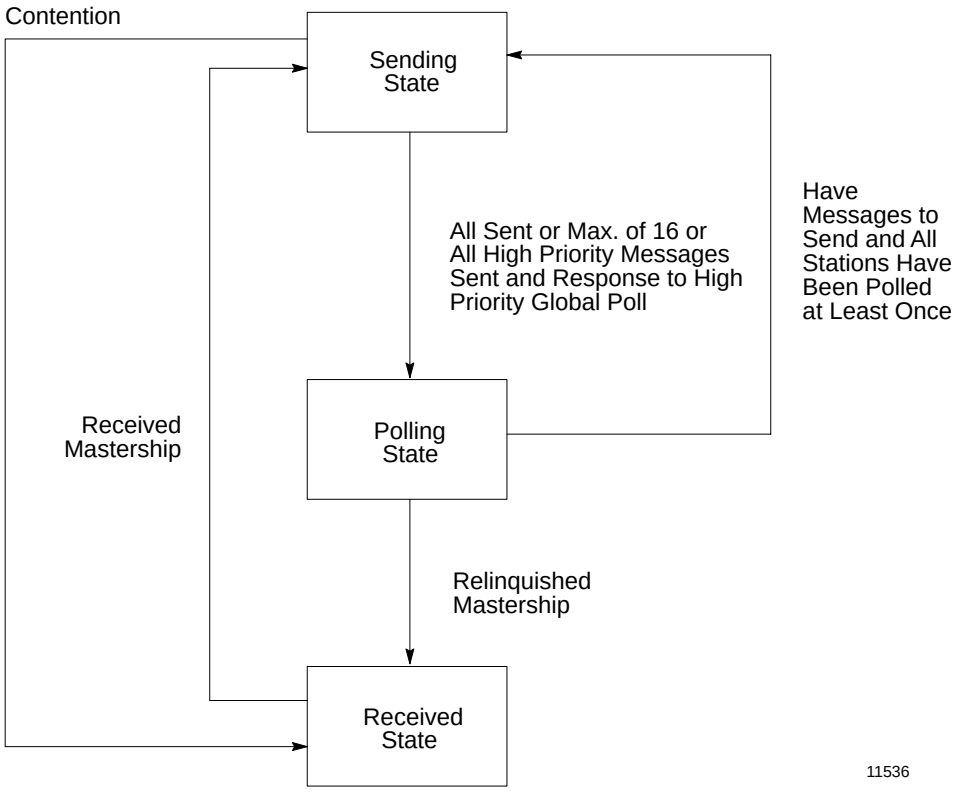
When a station gains control of the Data Highway to transmit messages, it has become a master station. All other station interface modules assume a slave mode. This enables these stations to receive and acknowledge messages sent to them. Basically, a station has three states:

- Transmitting Message Packets
- Polling to Determine which Station Gets Mastership Next
- Receiving and Responding to Message and Polling Packets

Thus, each Data Highway station can transmit and receive both message and polling packets.

Figure 3.1 shows the change of states at a station.

Figure 3.1
State Transitions at Master Station



Message Transmission

A station must possess mastership of the Data Highway before it can transmit any message or polling packets. As part of the data integrity of the highway, all commands must receive a reply before the station sending the command considers a transaction complete. Since the highway link treats both commands and replies as packets, it takes at least one change of mastership to complete a single transaction.

You must format any command in the application program of the local, or transmitting, station. For a PC, the format is part of the PC user program (Chapter 5). For a computer, you format the message as part of the computer program (Chapter 6).

A station generates a reply message in response to a command message it receives. The reply message indicates that a station received the command message and that the communication module has completed the sequence of events required of it for command execution. For commands that read data, the reply message contains the data specified by the

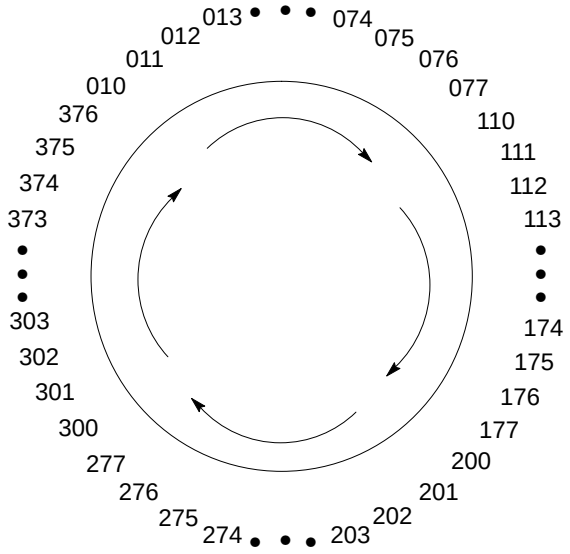
command. For commands that write data, the reply message indicates that the write operation has been completed at the receiving station.

When the replying station is a programmable controller, the reply message is an automatic function of the communication module operation and is transparent to the user program. If the replying station is a computer, you must program the computer to formulate the response/reply message.

Polling

To transfer mastership between stations, an orderly polling scheme is used. Normally, mastership passes from one station to another in a round-robin fashion. During installation, you assign each station a unique number from 010_8 to 77_8 or 110_8 to 376_8. Each master passes mastership to the station with the next higher address that is requesting mastership. All polling arithmetic is performed modulo-256 relative to the address of the current master. That is, mastership is offered to the station with the next higher of the 256 possible addresses; when the highest address is reached, it starts over at the lowest address. For example, when the master is 200, Station 220 will receive mastership before Station 30, since 220 is less than 30 relative to 200. Figure 3.2 illustrates this polling scheme.

Figure 3.2
Polling Scheme



11537

The polling algorithm recognizes two levels of message priority and gives preference to stations with high priority messages when selecting the new master. This bypasses the normal round-robin pattern and causes mastership to pass to the station with a high priority message and the next higher number than the previous master. Stations with high priority messages pass mastership either to another station with high priority messages or to the station with normal priority messages that has the next higher number than the most recent master with normal priority messages.

Polling Selection

Two group selection parameters are used in polling. The first group selection parameter is priority. There are two kinds of polling packets:

- High Priority Polls — Do not contain a current master number and can only be responded to by stations that have a high priority message to send.
- Normal Priority Polls — Contain the number of the current master and can be responded to by any station that has a normal or high priority message.

The second group selection parameter is station number. A group of stations is specified by giving the number of the polling station and the size of the group. A station is in the polled group if that station's number minus the polling station's number (modulo-256) is less than the group size.

A slave station will respond to a poll if it meets all of the following conditions:

- Slave station's priority is greater than or equal to the poll priority.
- Slave station is in the polled group.
- Slave station has a message to send.

Polling Sequences

A polling sequence is a series of polls. The range of stations specified in the poll command is usually altered as successive polls narrow in on the station to be selected. The following types of polling sequences are used:

- Global High Priority Poll
- Descending Binary Search (High or Normal Priority)
- Ascending Binary Search (Normal Priority Only)
- Continuous Global Poll (Normal Priority Only)
- Single-Station Poll (High or Normal Priority)

High priority polling sequences consist of high priority polls. Normal priority sequences contain only normal priority polls.

Before starting a non-global normal priority sequence, the master station uses a single global high priority poll to determine whether any station has a high priority message to transmit. The global poll encompasses all stations on the link.

A descending binary search narrows in on a station any time a response to a group poll is detected. At each step the group selected by the previous poll is divided into two halves, and the first half is polled. If there is a response, that half is selected for the the next poll; otherwise, the half that wasn't polled is selected. This continues until the last group is divided into two groups of one each, and either:

- The first station responds, and mastership is transferred.
- The second station responds, and mastership is transferred.
- Neither station responds, and the master starts continuous polling.

After a station enters the polling state but before it starts a global poll, it uses an ascending binary search to check the stations having addresses immediately higher than itself. The station with the address of CM+1 (current master plus 1) is polled first, then CM+1 through CM+3, CM+1 through CM+7, and so on, in group sizes of 1, 3, 7, 15, and 31. If the first poll gets a response, mastership transfers immediately. If a subsequent poll receives a response, a descending binary search starts. If there is no response to the fifth ascending poll, continuous global polling starts.

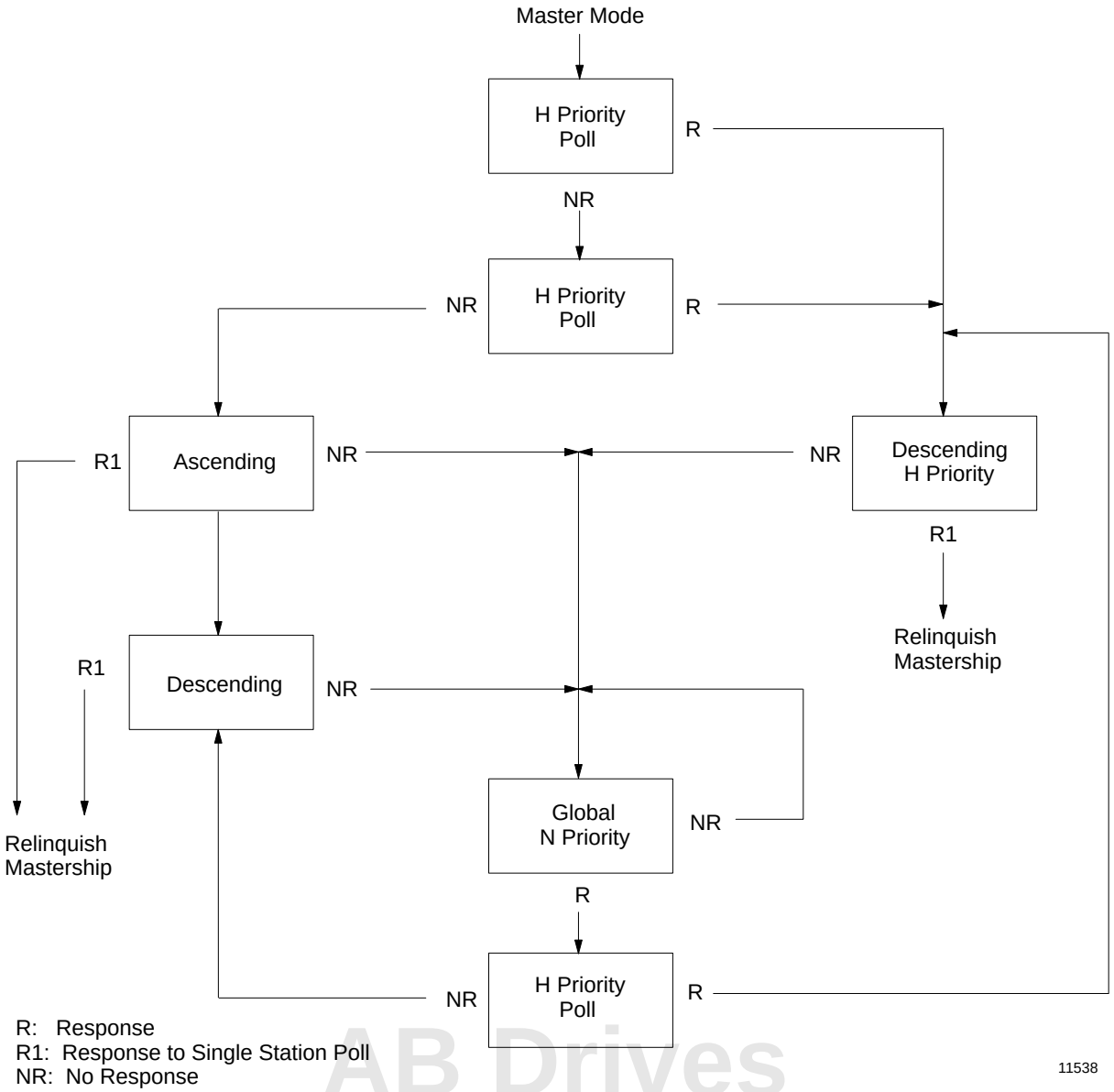
The current master enters a state of continuous global polling after any other polling sequence has failed to select a station. The master continuously sends normal polls to all stations until it receives a response. It then gives one global priority poll to determine whether a high priority or normal priority descending binary search should be done, then starts the appropriate descending search.

A single-station poll is a special case that is encountered at the end of a descending search or the beginning of an ascending search. Any time a single station is polled and it responds, the old master relinquishes mastership to the polled station. Figure 3.3 shows this polling process.

Link Disconnect

Floating master operation continues normally as long as all stations share mastership of the Data Highway link. However, if any one station retained continuous control of the communication link due to a fault condition, floating master operation would not be possible and Data Highway communication would be disabled. As a guard against this type of situation, each station interface module has automatic link disconnect circuitry.

Figure 3.3
Polling Process



Chapter 4

Installation

General

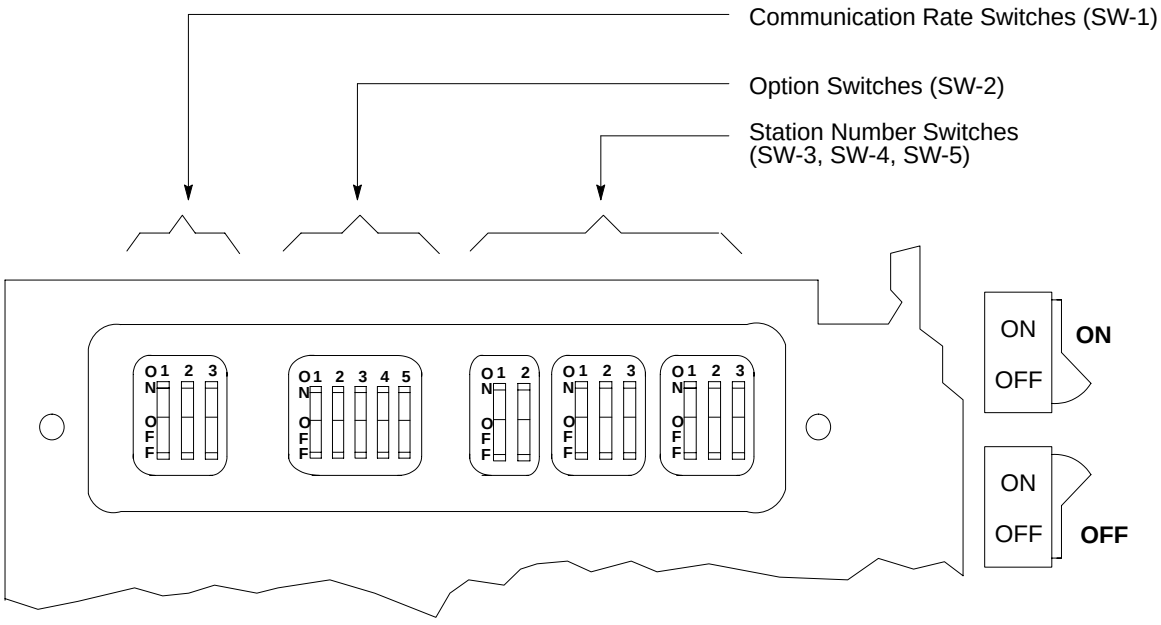
After studying Chapter 2 (Application Considerations) and Chapter 3 (Communication Concepts) you can then make decisions about what kind of communication network you want and how you will configure your 1771-KG module. After making those decisions, you can install the module. Module installation includes:

- Making Module Operation Selections through Switch Settings
- Keying the Backplane Connector to Match the Module
- Connecting Cables to the Module

Switch Settings

Switches are located behind a cover plate at the top of the module (Figure 4.1).

Figure 4.1
Switch Assembly Locations



The switches are in five switch assemblies. Switches are shown and described in this publication as being ON or OFF. Printed on the actual switch assemblies are the words ON and OFF or the word OPEN. Open corresponds to OFF.

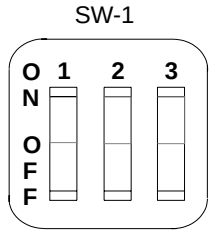
These switch assemblies are identified by the nomenclature SW-1 through SW-5. This designation is printed on the printed circuit board just above each switch assembly.

Use a blunt-pointed instrument, such as a ball-point pen, to set these switches. **Never** use a pencil; lead could jam the switch.

Communication Rate

Select the communication rate of the RS-232-C port through Switch Assembly SW-1. Set the switches as indicated in Figure 4.2. For a longline RS-232-C communication link, limit the communication rate according to the line length as indicated in Table 4.A.

Figure 4.2
Communication Rate Switch Settings



Communication Rate in Bits/s			
110	OFF	OFF	OFF
300	ON	OFF	OFF
600	OFF	ON	OFF
1,200	ON	ON	OFF
2,400	OFF	OFF	ON
4,800	ON	OFF	ON
9,600	OFF	ON	ON
19,200	ON	ON	ON

11303

Table 4.A
Communication Rate List for Longline RS-232-C Link

Cable Length	Communication Rate Limit
Up to 2,000 feet	19,200 bits/s
2,000 to 4,000 feet	9,600 bits/s
4,000 to 6,000 feet	4,800 bits/s
6,000 to 7,000 feet	2,400 bits/s

Series A Module Options

On the Series A 1771-KG module, each switch of Switch Assembly SW-2 provides an option selection for the module (Figure 4.3.A).

Parity

The parity switch selections are:

- ON — The module transmits and accepts characters made up of a start bit, eight data bits, an even parity bit, and a stop bit.
- OFF — The module transmits and accepts characters made up of a start bit, eight data bits, no parity bit, and a stop bit.

Embedded Response

The embedded-response switch selections are:

- ON — If the module receives a message transmission which ends while the module is transmitting a message, the module will interrupt its message transmission long enough to transmit its acknowledgment of the received message.
- OFF — If the module receives a message transmission which ends while the module is transmitting a message, the module will complete its message transmission before transmitting its acknowledgment of the received message.

Enabling embedded responses allows the most efficient use of the communication link. However, it applies only if you set the protocol-type switch to ON for full-duplex.

Accept Unprotected Writes

The accept-unprotected-writes switch selections are:

- ON — The module accepts unprotected write commands.
- OFF — The module rejects unprotected write commands.

Accept Physical Writes

The accept-physical-writes switch selections are:

- ON — The module accepts physical write commands.
- OFF — The module rejects physical write commands.

With a programmed EPROM installed in a Mini-PLC-2/15 processor, the module rejects any write command addressing words above 177_8 regardless of this switch setting. Also, with the memory write protect jumper removed from a PLC-2/30 processor, the module rejects any write command addressing words above 377_8 regardless of this switch setting.

Protocol

The protocol switch selections are:

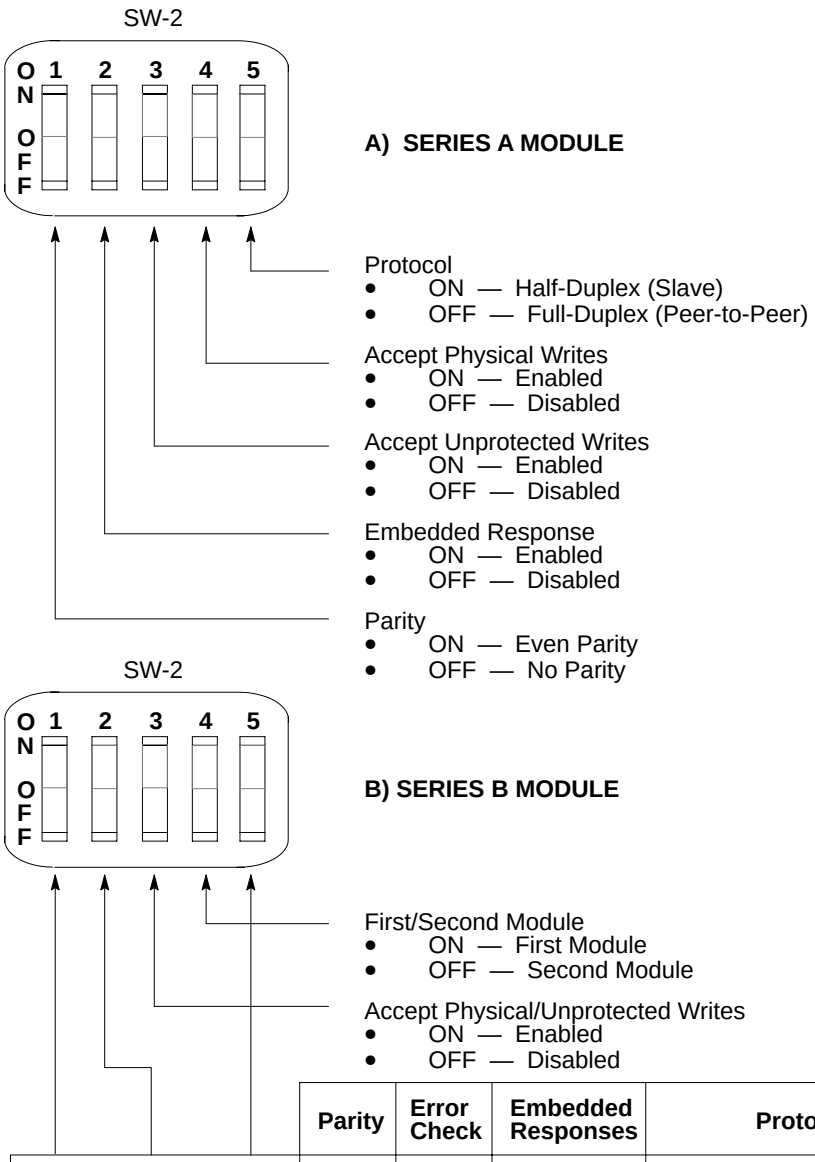
- ON — The module communicates as a slave in a master/slave relationship through a half-duplex, polled protocol.
- OFF — The module communicates in a peer-to-peer relationship through a full-duplex unpolled protocol.

If you use the module in an RS-232-C link to another Allen-Bradley communication module, set the switch to OFF for the full-duplex unpolled protocol.

Series B Module Options

On the Series B 1771-KG module, Switches 3 and 4 of switch assembly SW-2 each provide an option selection; the other three switches provide eight combinations of selections for four other options (Figure 4.3.B).

Figure 4.3
Option Switch Settings



			Parity	Error Check	Embedded Responses	Protocol
OFF	OFF	OFF	None	BCC	Disabled	Full-Duplex (Peer-to-Peer)
ON	OFF	OFF	Even			
OFF	ON	OFF	None	BCC	Enabled	
ON	ON	OFF	Even			
OFF	OFF	ON	None	BCC	Disabled	Half-Duplex (Slave)
ON	OFF	ON	Even			
OFF	ON	ON	None	CRC	Enabled	Full-Duplex (Peer-to-Peer)
ON	ON	ON	None	CRC	Disabled	Half-Duplex (Slave)

11539

First/Second Module

The first/second module switch selections are:

- ON — First module. Use this selection if you connect the module directly to the PC processor.
- OFF — Second module. Use this selection if you connect the module to the PROGRAM INTERFACE connector on another Series B 1771-KG module or a 1771-KA2 module (see Chapter 4, “To Second 1771-KG Module” and “To 1771-KA2 Module”).

Accept Physical/Unprotected Writes

The accept-physical/unprotected-writes switch selections are:

- ON — The module accepts physical write commands and unprotected write commands.
- OFF — The module rejects physical write commands and unprotected write commands.

With a programmed EPROM installed in a Mini-PLC-2/15 processor, the module rejects any write command addressing words above 177_8 regardless of this selection. Also, with the memory write protect jumper removed from a PLC-2/30 processor, the module rejects any write command addressing words above 377_8 regardless of this selection.

Parity

The parity selections are:

- Even Parity — The module transmits and accepts characters made up of a start bit, eight data bits, an even parity bit, and a stop bit.

  Note that you cannot select even parity if you select CRC for the error check selection.

- None — The module transmits and accepts characters made up of a start bit, eight data bits, no parity bit, and a stop bit.

Error Check

The error check selections are:

- BCC — The module transmits and accepts packets which end with a BCC byte for error checking.
- CRC — The module transmits and accepts packets which end with a two-byte CRC field for error checking.

When deciding whether to use BCC or CRC, consider the following:

- CRC provides more complete error checking.
- BCC is easier to implement in the computer driver and provides adequate error checking for most applications.
- For communicating with another Series B 1771-KG module, you must make the same selection at both modules.

Embedded Response

The embedded-response selections are:

- Enabled — If the module receives a message transmission which ends while the module is transmitting a message, the module will interrupt its message transmission long enough to transmit its acknowledgment of the received message.
- Disabled — If the module receives a message transmission which ends while the module is transmitting a message, the module will complete its message transmission before transmitting its acknowledgment of the received message.

Enabling embedded responses allows the most efficient use of the communication line. However, it applies only if you select the full-duplex protocol.

Station Number

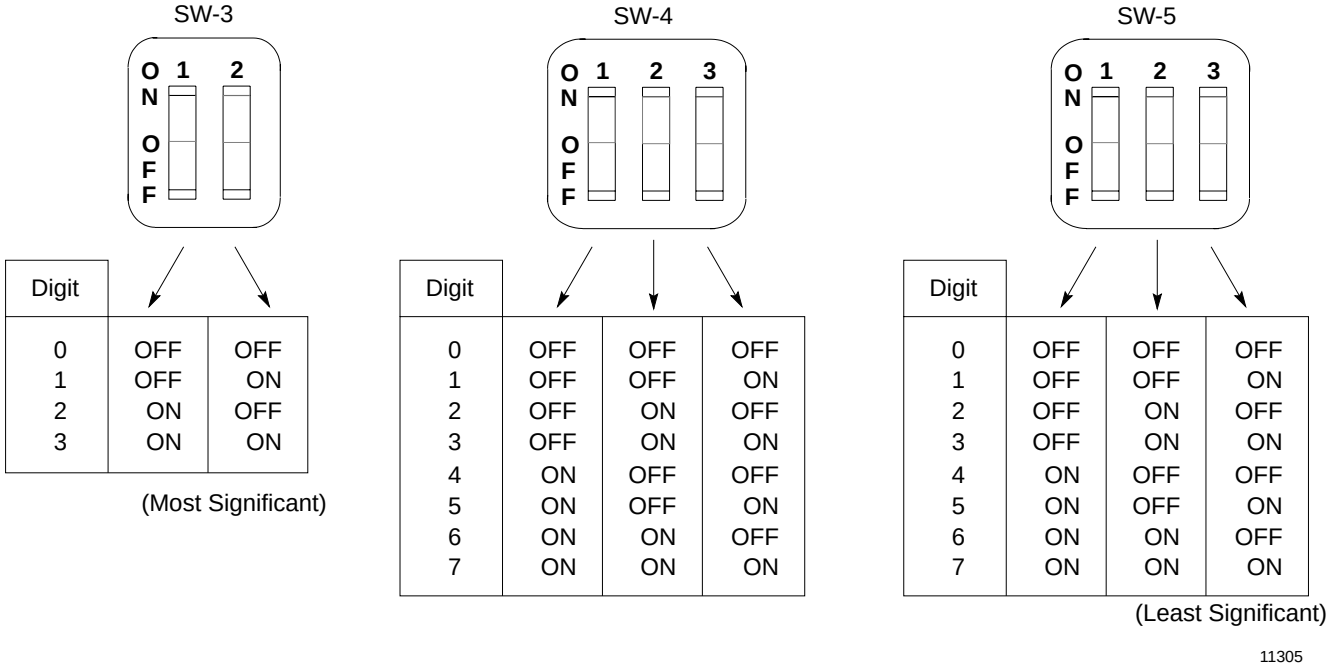
Select the station number through Switch Assemblies SW-3, SW-4, and SW-5. Set the switches as indicated in Figure 4.4.

Select a number from 10 to 77_8 or 110 to 376_8. To minimize Data Highway link polling time, select station numbers that are close together.

If the 1771-KG module is linked to a communication controller module, they are each part of the same Data Highway station and you must therefore assign each the same station number.

If you use the 1771-KG module in a stand-alone link, assign it a different station number from the other stations in the link.

Figure 4.4
Station Number Switch Settings



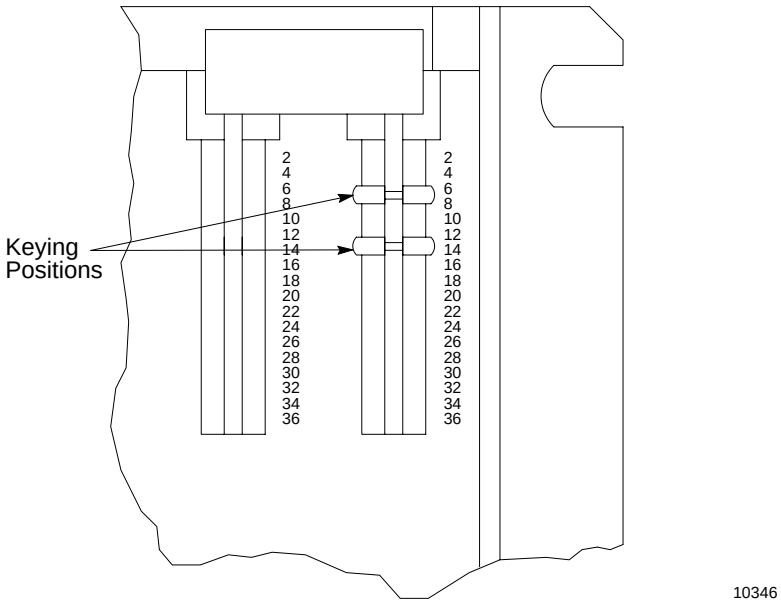
Keying

A package of plastic keys (Cat. No. 1777-RK) is provided as standard with each I/O chassis. When properly installed, these keys can guard against the seating of all but a selected type of module in a particular I/O chassis module slot. Keys also help align the module with the backplane connector. Misalignment could result in short circuits.

Each module is slotted in two places at the rear edge. Position the keys on the chassis backplane connector to correspond to these slots to allow the seating of the module.

Use long-nosed pliers to insert key into the upper backplane connectors. Position the keys between the numbers at the right of the connectors. Refer to Figure 4.5 for the keying positions.

Figure 4.5
Keying Diagram



You can reposition keys if it becomes necessary to use a different type of module.

Module Insertion

Remove power from the I/O chassis before inserting or removing a module. Open the module locking latch on the I/O chassis and insert the module into the slot keyed for it. Firmly press to seat the module into its backplane connector. Secure the module in place with the module locking latch.

CAUTION: Do not force a module into a backplane connector; if you cannot seat a module with firm pressure, check the alignment and keying. Forcing a module can damage the backplane connector or the module.

Cabling

After you have placed the 1771-KG module into a 1771 I/O chassis slot within 10 cable-feet from the PC processor, you can begin connecting the cables necessary for your application. Cables that you might connect to the 1771-KG module are listed in Table 4.B.

Table 4.B
Cables

Cat. No.	Length	Connects
1770-CG	16.5 feet (5 m)	RS-232-C PORT connector to a computer.
1770-CP	16.5 feet (5 m)	RS-232-C PORT connector to a MODEM.
1770-CN	1.5 feet (0.46 m)	PROCESSOR connector to a Mini-PLC-2 or -2/15 processor.
		PROGRAM INTERFACE connector to a 1770-RG module.
1771-CO	3.5 feet (l m)	PROCESSOR connector to a Mini-PLC-2 or -2/15 processor.
		PROGRAM INTERFACE connector to a 1770-RG module.
1771-CR	10 feet (3.1 m)	PROCESSOR connector to a PLC-2/20 or -2/30 processor.
		PROGRAM INTERFACE connector to a 1770-RG module.
1772-TC	10 feet (3.1 m)	PROGRAM INTERFACE connector to a programming terminal.

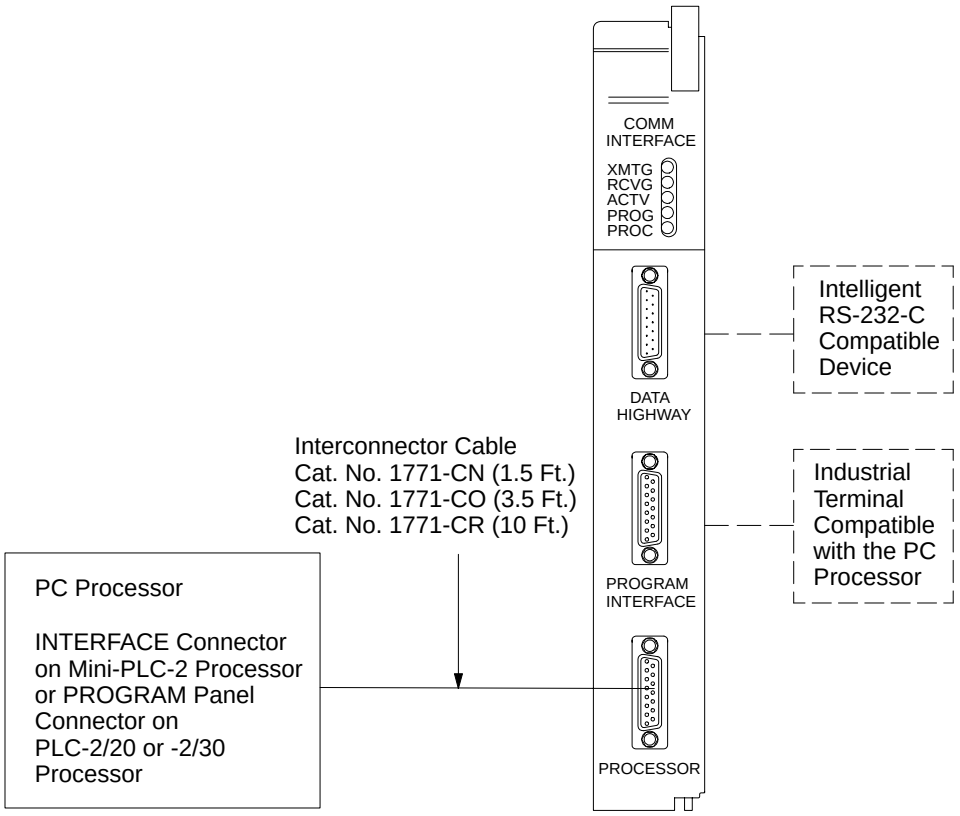
To PC Processor

You must connect a cable between the PROCESSOR connector on the 1771-KG module and the PC processor (Figure 4.6).

Use a 1771-CN or 1771-CO cable for connection to a Mini-PLC-2 or Mini-PLC-2/15 processor. Use a 1771-CR cable for connection to a PLC-2/20 or PLC-2/30 processor.

Connect to the INTERFACE connector on a Mini-PLC-2 or Mini-PLC-2/15 processor. Connect to the PROGRAM PANEL connector on a PLC-2/20 or PLC-2/30 processor.

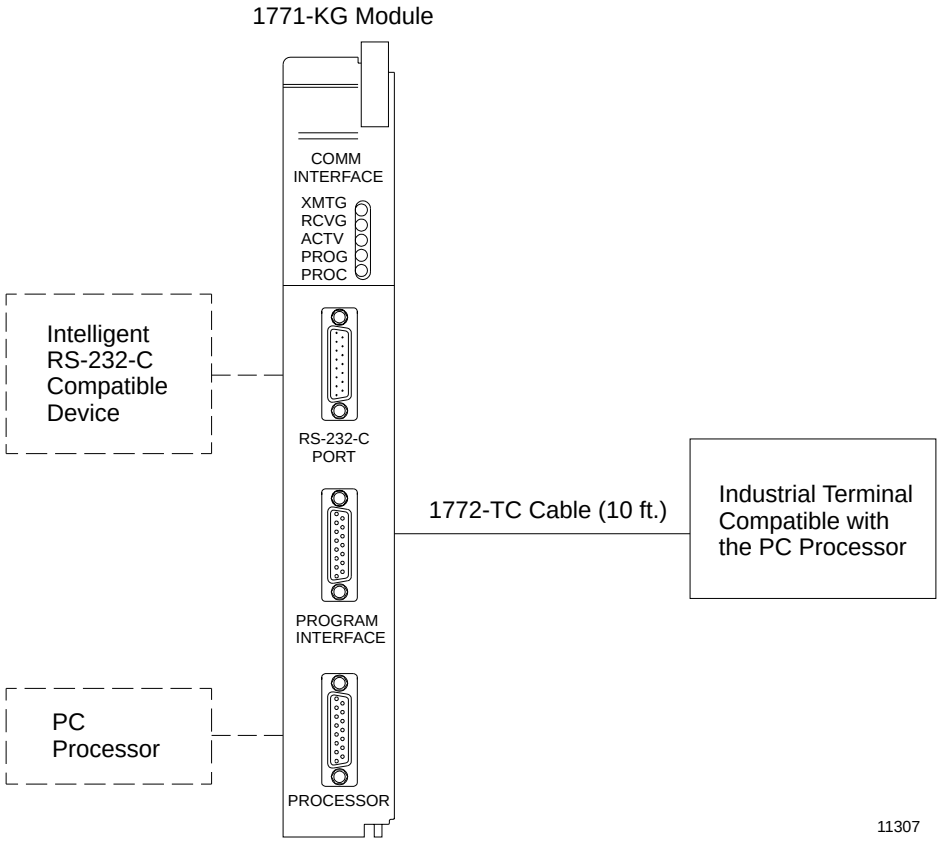
Figure 4.6
Connection to the PC Processor



To Programming Terminal

To use an industrial terminal with the PC processor, connect a 1772-TC cable between the PROGRAM INTERFACE connector on the 1771-KG module and the connector on the industrial terminal (Figure 4.7).
Note: With this connection you cannot use tape cassette functions through the industrial terminal. If you use two modules with the same processor (Figure 4.9) you must use a 1770-T3 industrial terminal with Revision F or later firmware.

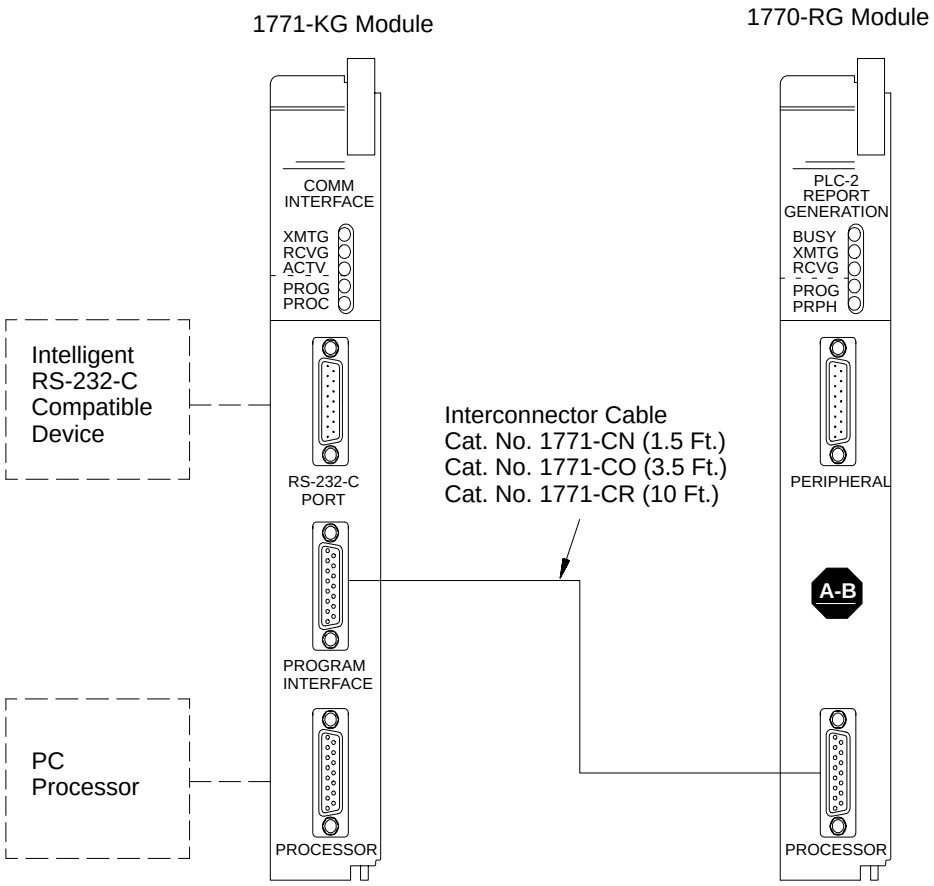
Figure 4.7
Connection to a Programming Terminal



To 1770-RG Module

To use a PLC-2 Family Report Generation Module (Cat. No. 1770-RG) with the PC processor, connect a 1771-CN, -CO, or -CR cable between the PROGRAM INTERFACE connector on the 1771-KG module and the PROCESSOR connector on the 1770-RG module (Figure 4.8).

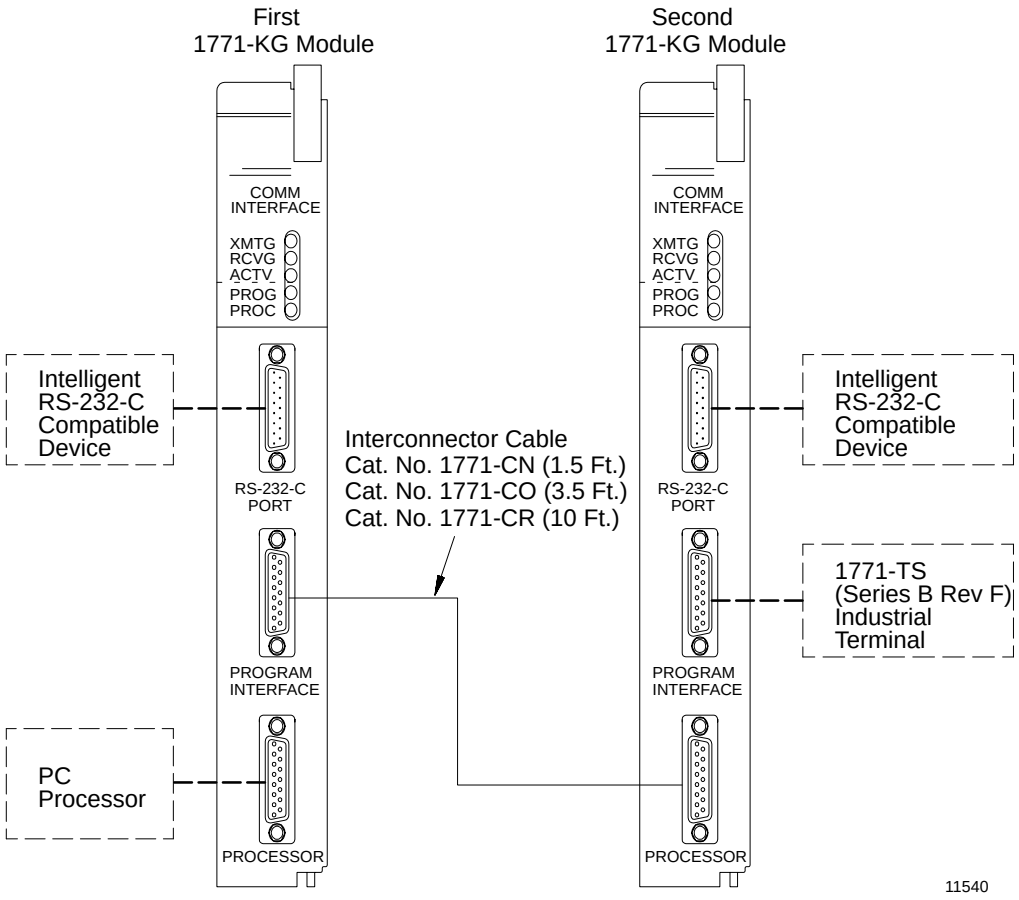
Figure 4.8
Connection to a 1770-RG Module



To Second 1771-KG Module

To use two Series B 1771-KG modules (for a second RS-232-C link) connect a 1771-CN, -CO, or -CR cable between the PROGRAM INTERFACE connector on the first module and the PROCESSOR connector on the second module (Figure 4.9).

Figure 4.9
Connection to a Second 1771-KG Module



To 1771-KA2 Module

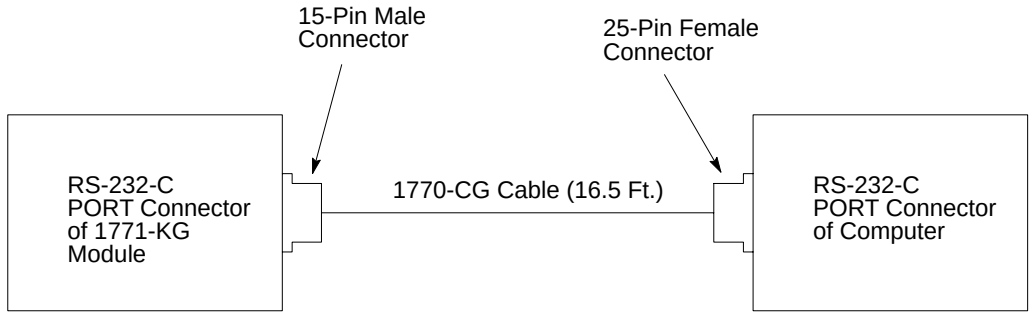
Another possibility of connecting two modules (for a second link) is to connect a 1771-KG module (for an RS-232-C link) and a 1771-KA2 module (for a Data Highway link). You can connect the 1771-KA2 module as either the first or second module. Connect a 1771-CN, -CO, or -CR cable between the PROGRAM INTERFACE connector on the first module and the PROCESSOR connector on the second module.

To Computer

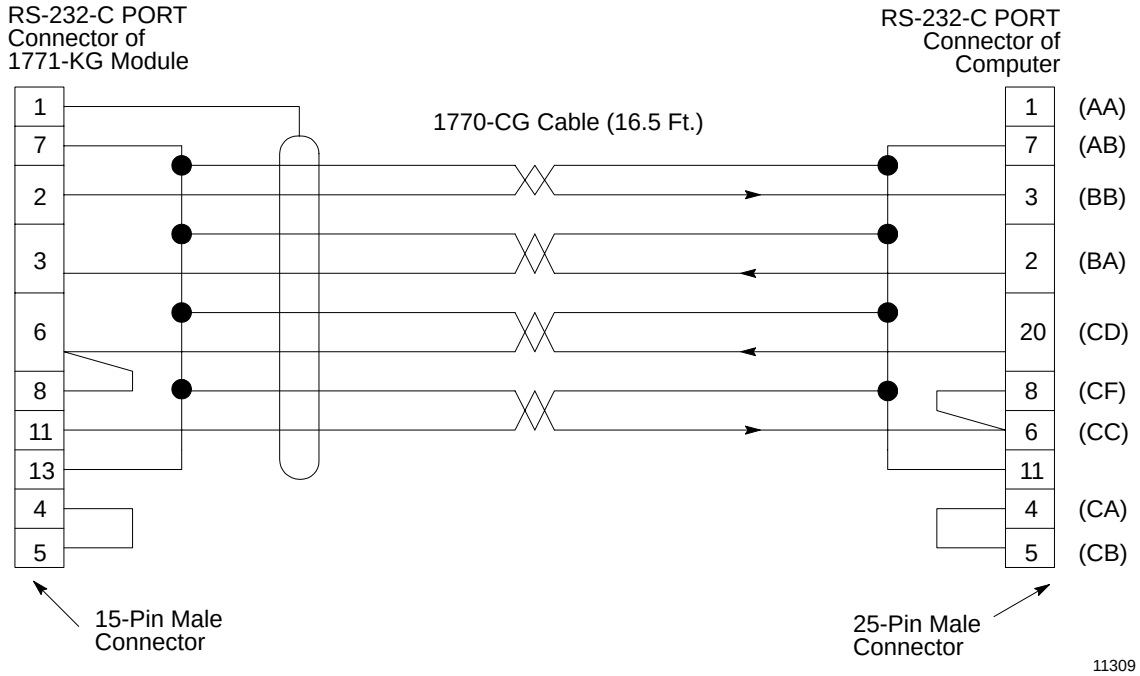
To provide an RS-232-C communication link with a computer, connect a 1770-CG cable between the RS-232-C PORT connector on the 1771-KG module and the connector for an RS-232-C compatible port of the computer (Figure 4.10).

Figure 4.10
Connection to a Computer

A) Connection Diagram



B) Wiring Diagram



The cable length is 16.5 feet (5 m). If you need a longer cable or a male/female adapter cable, refer to the wiring diagram (Figure 4.10.B) to assemble your own cable. Connect the shield at one end only. Do not exceed the RS-232-C limit of 50 cable-feet.

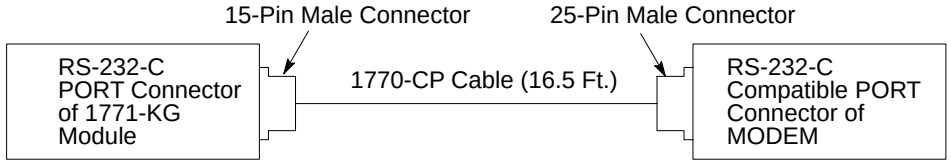
This connection configuration provides the data terminal ready signal to allow each end to detect the loss of the other end's ability to communicate.

To Modem

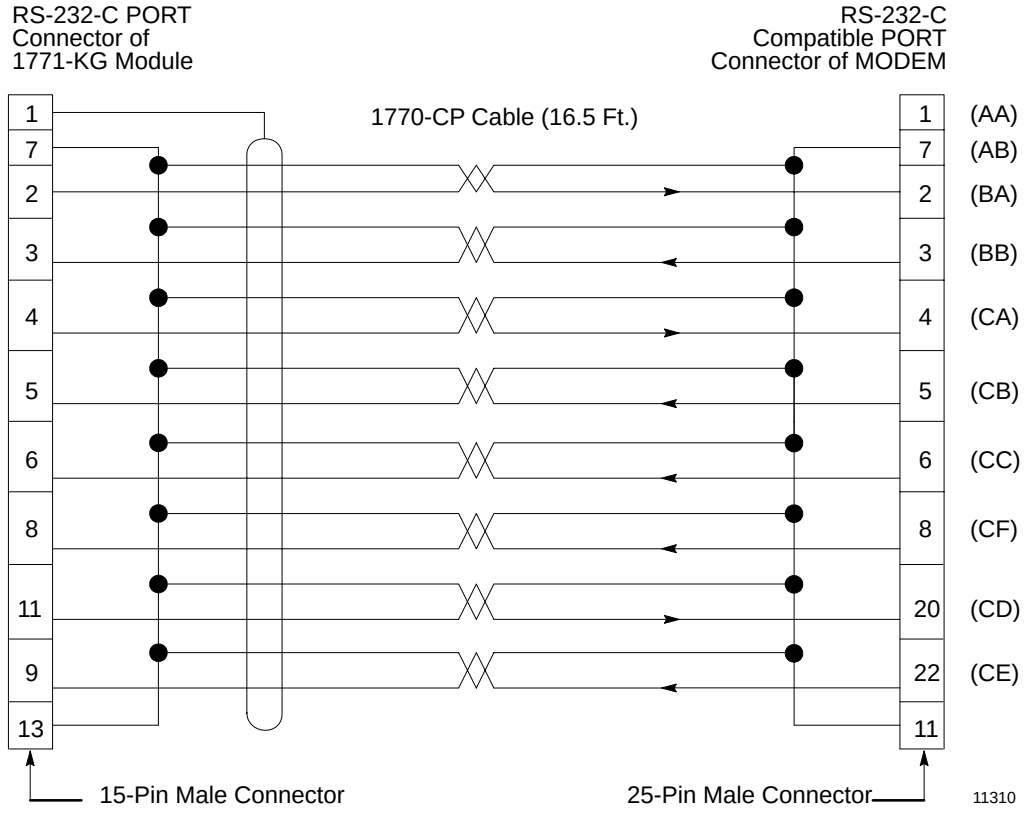
To provide an RS-232-C communication link with a MODEM, connect a 1770-CP cable between the RS-232-C PORT connector on the 1771-KG module and the connector for the RS-232-C compatible port of the MODEM (Figure 4.11).

Figure 4.11
Connection to a MODEM

A) Connection Diagram



B) Wiring Diagram



The cable length is 16.5 feet (5 m). If you need a longer cable or a male/female adapter cable, refer to the wiring diagram (Figure 4.11.B) to assemble your own cable. Connect the shield at one end only. Do not exceed the RS-232-C limit of 50 cable-feet. Most MODEMS hold the handshake lines in the proper states. If not, you can jumper them.

You can connect the module to standard American dial-up MODEMS and some European MODEMS. Other European standards specify that the DTR signal will cause the MODEM to answer the phone whether it is ringing or not, causing the phone to always be "busy." Since the 1771-KG asserts a data terminal ready (DTR) signal while waiting for a call, the module cannot be used with such MODEMS.

The types of dial-up network MODEMS that you can use are classified into the following types:

- Manual — These are typically acoustically coupled MODEMS. The connection is established by human operators at both ends, who then insert the handsets into acoustic couplers to connect the link.
- DTE-Controlled Answer — These unattended MODEMS are directly connected to the phone lines. The module as the data terminal equipment controls the MODEM via the DTR, data set ready (DSR), and data carrier detect (DCD) signals. The module uses timeouts and tests to properly operate these types of MODEMS.
- Auto-Answer — These MODEMS have self-contained timeouts and tests and can connect and disconnect the phone line automatically.

The module has no means to control an auto-dial MODEM, although it is possible for you to use it in conjunction with a separate auto-dialer.

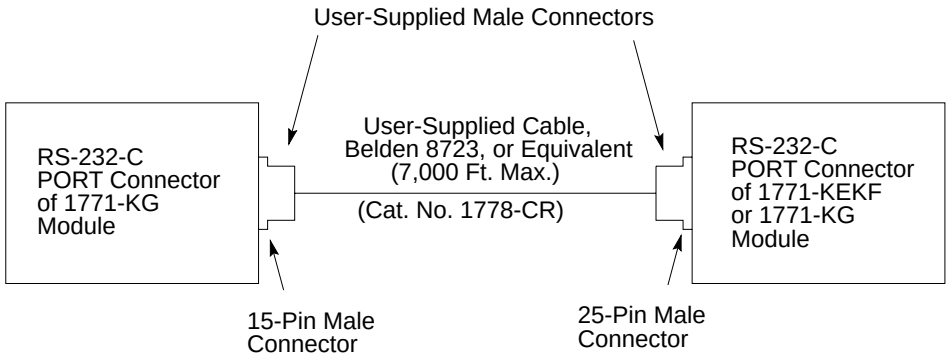
To Communication Module

To provide a longline RS-232-C communication link with a 1771-KE/-KF module or another 1771-KG module, refer to Figure 4.12. To provide a longline RS-232-C communication link with a 1773-KA or 1775-KA module, refer to Figure 4.13.

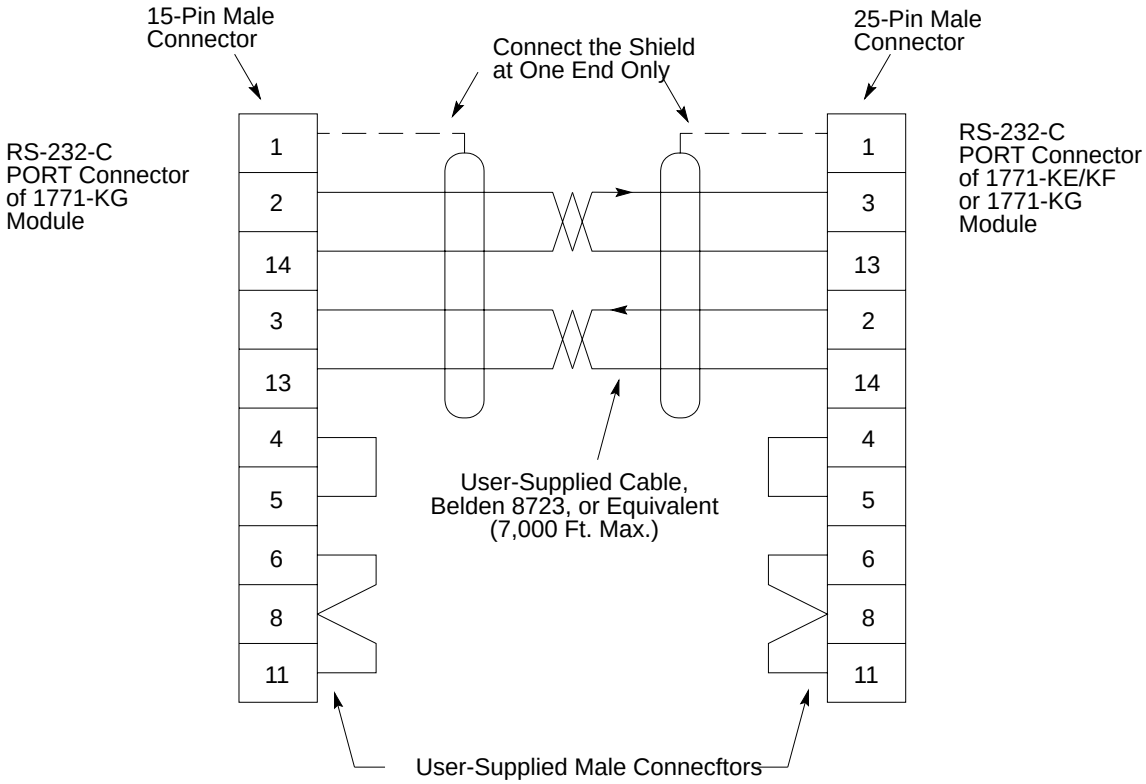
Assemble your own cable as shown in the wiring diagram. Use Belden 8723 or equivalent cable which you can purchase from Allen-Bradley using Cat. No. 1778-CR. Connect the shield at one end only. Use a 15-pin male connector on the 1771-KG module end. Use a 15-pin or 25-pin male connector on the other end as required. You can purchase a 25-pin male connector kit (Cat. No. 1770-XXP) from Allen-Bradley.

The cable length is limited to 7,000 feet. If you select a communication rate greater than 2,400 bits/s, the cable length is further limited (Table 4.A).

Figure 4.12
Connection to 1771-KE/-KF or Another 1771-KG Module



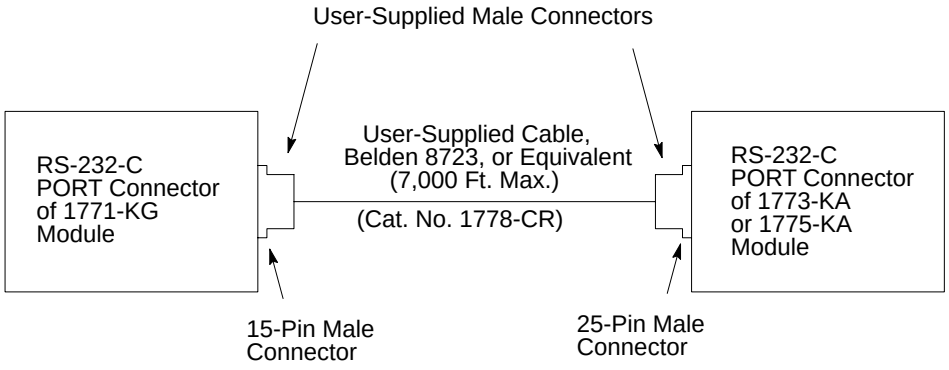
A) Connection Diagram



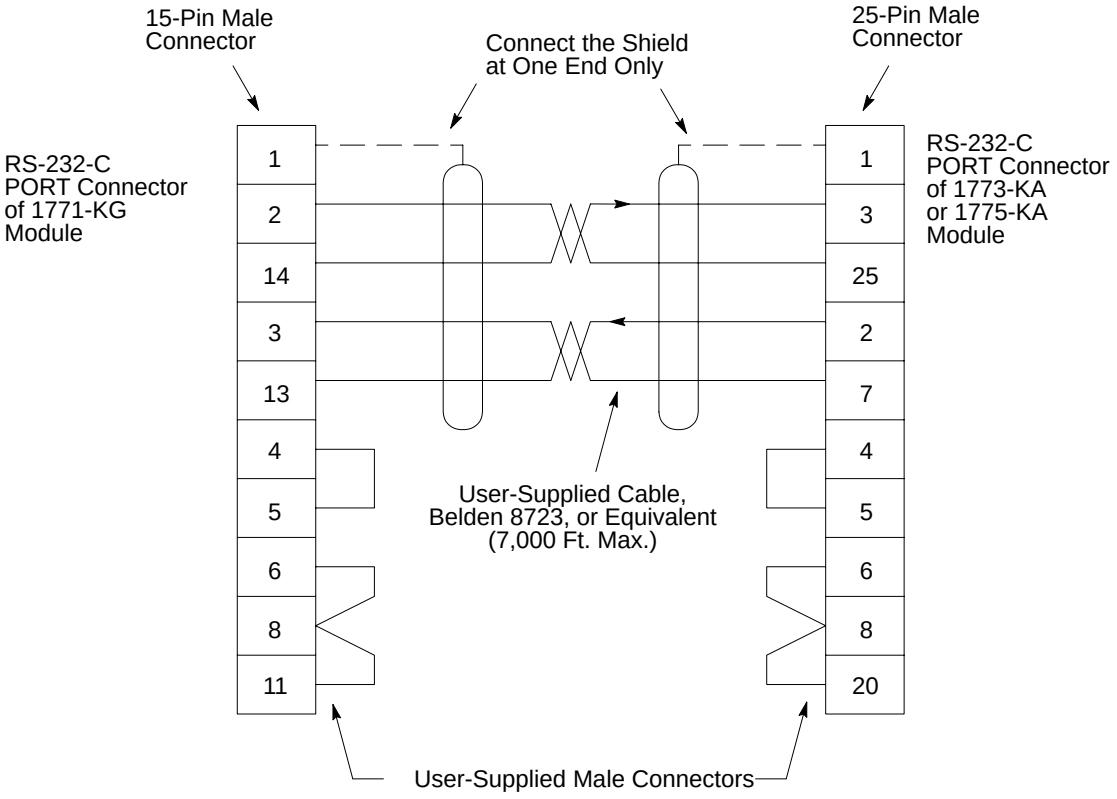
B) Wiring Diagram

11311

Figure 4.13
Connection to 1773-KA or 1775-KG Module



A) Connection Diagram



B) Wiring Diagram

11312

RS-232-C Port

The RS-232-C PORT connector on the module does not conform to RS-232-C mechanical specifications because it is only a 15-pin connector. However, it does connect to circuits which conform to RS-232-C electrical specifications. Additionally, two circuits are provided for isolated longline transmission which do not conform to RS-232-C specifications. All connections are listed in Table 4.C.

Table 4.C
RS-232-C Port Connections

Signal	Abbreviation	Pin	Input/Output
Chassis/Shield Drain		1	
Transmitted Data	TXD	2	Output
Received Data	RXD	3	Input
Request to Send	RTS	4	Output
Clear to Send	CTS	5	Input
Data Set Ready	DSR	6	Input
Signal Ground	GND	7	
Data Carrier Detect	DCD	8	Input
Unused		9	
Unused		10	
Data Terminal Ready	DTR	11	Output
Unused		12	
Received Data Return	RXDRET	13	Input
Transmitted Data Return	TXDRET	14	Output
Unused		15	

The definitions of the signals into and out of this port are:

- TXD carries serialized data. It is output from the module.
- RXD is serialized data input to the module. RXD and RXDRET are isolated from the rest of the circuitry on the module.
- RTS is a request from the module to the MODEM to prepare to transmit. It typically turns the data carrier ON. In full-duplex mode, RTS is always asserted. In half-duplex mode it is turned ON when the module has permission to transmit, otherwise it is OFF.

- CTS is a signal from the MODEM to the module that the carrier is stable and the MODEM is ready to transmit. The module will not transmit until CTS is ON. If CTS is turned OFF during transmission, the module will stop transmitting until CTS is restored.
- DTR is a signal from the module to the MODEM to connect to the phone line (i.e., "pick up the phone"). The module will assert DTR except during the phone hang-up sequence. MODEMS built to American standards will not respond to DTR until the phone rings. Some European MODEMS will always connect the phone line, whether the phone is ringing or not. The module will not work with these types of European MODEMS.
- DSR is a signal from the MODEM to the module that the phone is off-hook. (It is the MODEM'S answer to DTR.) The module will not transmit or receive unless DSR is ON. If the MODEM does not properly control DSR, or if no MODEM is used, DSR must be jumpered to high signal at the connector. (You can jumper it to DTR.)
- DCD is a signal from the MODEM to the module that the carrier from another MODEM is being sensed on the phone line. It will not be asserted unless the phone is off-hook. Data will not be received at the module unless DCD is ON. In full-duplex mode the module will not transmit unless DCD is true. If the MODEM does not properly control DCD, or if a MODEM is not being used, you must jumper DCD to DTR at the module.
- TXDRET is the return signal for TXD. It is connected to module logic ground through a resistor. It does not conform to RS-232-C specifications.
- RXDRET is the return signal for RXD. It is connected to the isolated receiver and is isolated from all other circuitry on the module. It does not conform to RS-232-C specifications.

Answering

The module continually asserts DTR when it is waiting for a call. Under this condition, the MODEM will answer a call and assert DSR as soon as it detects ringing. The module does not monitor the RING indicator signal. Once it detects DSR, the module starts a timer (approximately 10 seconds) and waits for the DCD signal. When it detects DCD, communication can start.

If it does not detect DCD within the timeout, the module turns DTR OFF. This causes the MODEM to hang up and break the connection. When the hang-up is complete the MODEM turns OFF DSR. This causes the module to reassert the DTR line and wait for another call. This feature protects access to the phone if someone calling a wrong number reaches this station.

Once DCD is detected the module continues to monitor the DCD line. If DCD goes OFF, the module restarts the timeout. If the MODEM does not restore DCD within the timeout, the module initiates the hang-up sequence. This feature allows the remote station to redial in the event the connection is lost by the phone network.

Note that this handshaking is necessary to guarantee access to the phone line. If this handshaking protocol is defeated by improper selection of MODEM options, or jumpers at the connectors, the MODEM may answer a call; but if the connection is lost, the MODEM will not hang up. It will be impossible for the remote station to re-establish the connection because it will get a busy signal.

Chapter 5

PC Programming

General

A Data Highway network allows access to the memory of each remote station. This means that data table information at any operating PC can be transmitted to another PC or to a computer and used by that other station to control its own part of an operation. In addition, it allows computer-originated commands to access and upload or download user programs at a PC station (physical reads and writes).

However, if the local station PC processor doesn't have a communication zone in its ladder diagram program, the 1771-KG module will not initiate communication with any remote station. This chapter provides you with information you need for programming a PC processor (local station) for communication through a 1771-KG module to a remote station.

In this chapter, the generation of commands is discussed as if they are transmitted to a PLC-2 family processor at a remote station. If another type of PC processor is at the remote station, refer to the manual for the communication module of that processor for information on how to address that processor's data table. For a computer (and some processors) at a remote station to accept commands from this local station, you must provide a file at that remote station to simulate a PLC-2 family data table.

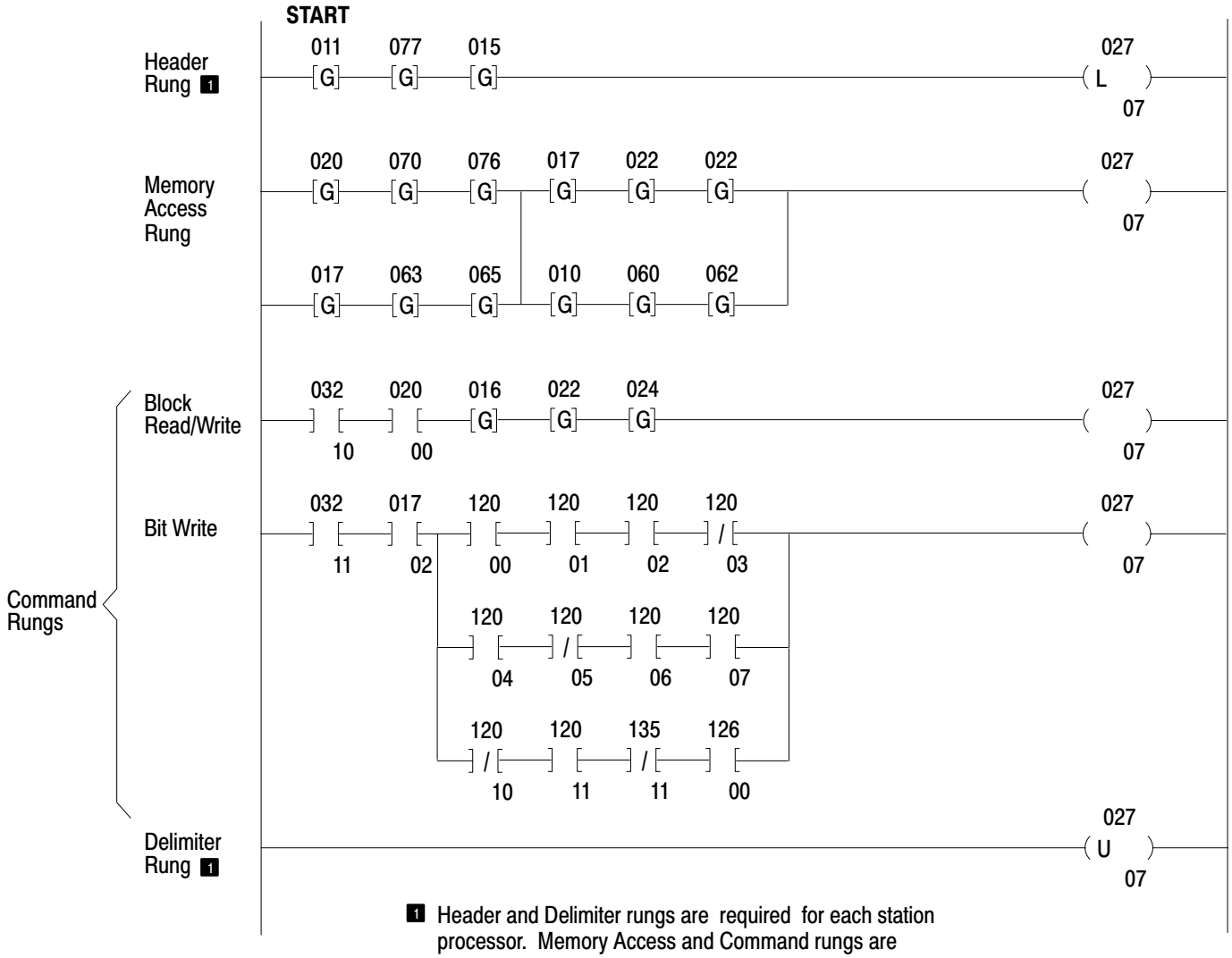
Communication Zone

Before any communication can occur between a remote station and your local station PC (through the 1771-KG module) you must provide a communication zone in your ladder diagram program. (Figure 5.1)

The overall format for the communication zone rungs of the program is shown in Figure 5.1. This figure shows each type of rung which can be entered in this zone.

Actual communication zone rungs for any station processor may vary. The length of this zone is a function of the number of remote stations with which the local station processor communicates and the number of types of commands to be sent to these remote stations.

Figure 5.1
Communication Zone Format



Note that the order of these communication zone rungs is as follows:

1. Header Rung
2. Memory Access Rung(s) (as Needed)
3. Command Rung(s) (as Needed)
4. Delimiter Rung

The communication zone rungs must always appear in this order.

Note that the figures in this section show the addresses above most GET instructions but not the three-digit data value displayed below the GET symbol. This convention is used for clarity, since, for the most part, only the GET address is significant when entering your program. In entering communication zone rungs, no data need be programmed into the GET instructions.

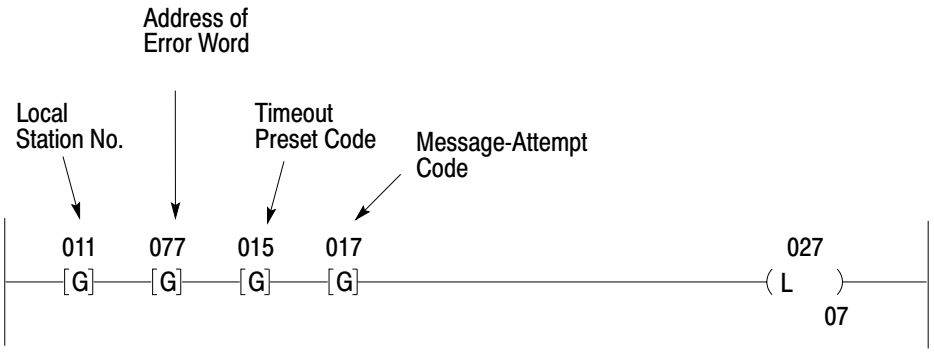
Header/Delimiter Rungs

A minimum communication zone consists of a header rung and a delimiter rung. With these two rungs, the local station can accept and reply to commands (other than protected write) from remote stations. Without these two rungs, the 1771-KG module turns on its PROG indicator to tell you that there is a problem with the program.

The delimiter rung defines the end of the communication zone. This rung must be exactly as shown in Figure 5.1. The unlatch 02707 instruction is interpreted as a special code. Word 027 is reserved for special functions such as this and cannot be used for other data table use.

The header rung defines the start of the communication zone. It must consist of three GET instructions and a latch 02707 instruction (Figure 5.2).

Figure 5.2
Header Rung



11542

The three GET instructions in the header rung specify the following:

- Local Station Number
- Address of the Error Word
- Timeout Preset Code

The local station number must match the three-digit number you selected with the station number switches on the 1771-KG module. This number must be an octal number from 010_8 to 077_8 or from 110_8 to 376_8.

Select any available word in the data table as the error word. The 1771-KG module will write error codes into the word you select (see "Error Word").

The timeout preset code gives a programmed timeout interval for command completion. Based on the three-digit value entered in the address field of this GET instruction, 1771-KG module monitors command execution for all commands sent from the local station. In the examples of this publication, we use 015 as the timeout preset code. This value, which designates a five-second timeout preset is suitable for most applications. The significance of this preset code, its computation, and timeout considerations are described later in "Timeout Preset Value."

Memory Access Rungs

Memory access rungs define data table words which can be accessed by the following commands from a remote station:

- Protected Block Write
- Protected Bit Write

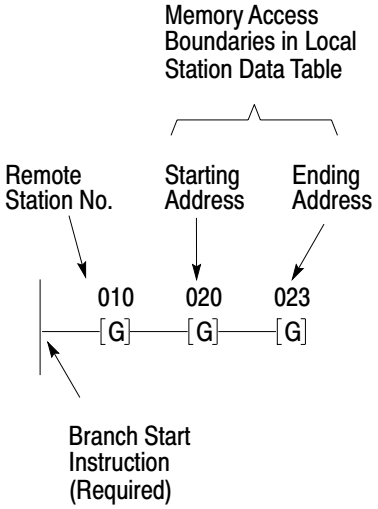
Protected commands, received from a remote station, may control only those memory areas in the local station processor that are listed in memory access rungs. (Without memory access rungs, the local station will not accept protected commands from a remote station.)

A memory access rung is composed of one or more memory access branches, as shown in Figure 5.3.A. In this format, a branch start precedes a group of three GET instructions. The first get instruction address is the station number of a remote station. The next two GET addresses define the word boundaries of the accessible data table area in the local station processor. The specified remote station may control any bit or word within these boundaries through protected commands.

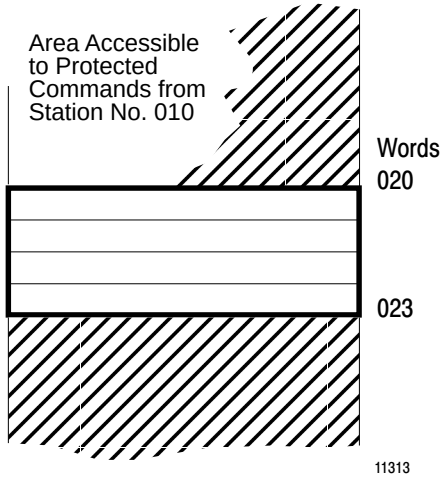
Figure 5.3.B shows the memory area that is now accessible to protected commands from Remote Station 010, due to the memory access branch of Figure 5.3.A.

Figure 5.3
Memory Access Example

(A) Memory Access Branch



(B) Local Station Processor Data Table



Multiple memory access branches can be listed in a single memory access rung. Note that each group of three GET instructions must be preceded by a branch start instruction. (This is true in all cases, even where only one memory access branch is defined.) Branch end instructions must be used to fit the memory access rung into the ladder diagram-display format.

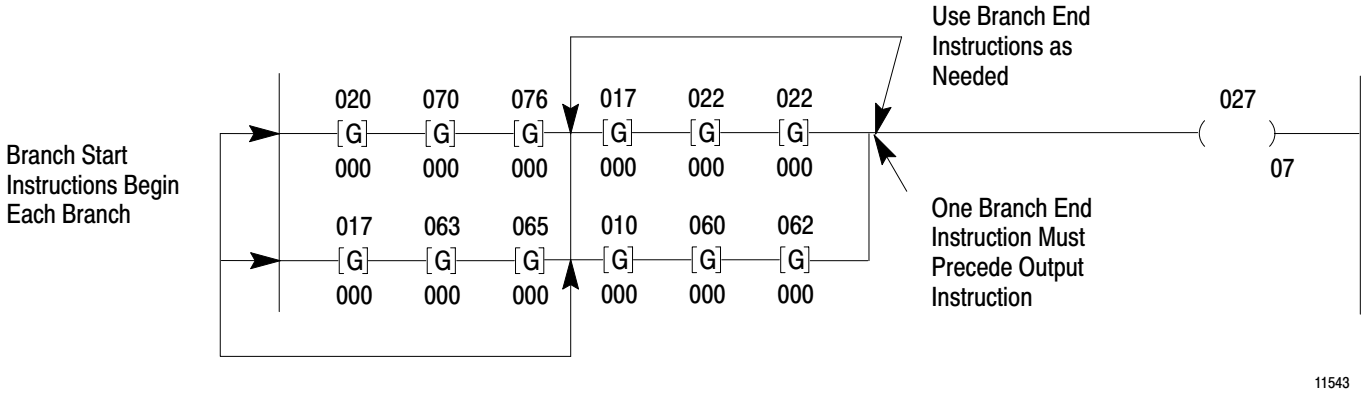
You should end each memory access rung with an energize 02707 instruction.

Figure 5.4 shows memory access rung with multiple branches. This rung lists the remote station which may control specific data table words using protected commands, as follows:

- Station 020 can control Words 070_8-076_8.
- Station 017 can control Words 063_8-065_8 and Word 022_8.
- Station 010 can control Words 060_8-062_8.

As shown in Figure 5.4, a single remote station may be identified in more than one memory access rung branch.

Figure 5.4
Typical Memory Access Rung



For practical reasons, do not exceed the display area of the programming terminal when entering these rungs. More than one memory access rung can be programmed if needed. Note, however, that should you require multiple memory access rungs, you must enter them in succession in the communication zone, immediately following the header rung and before any command rung.

Command Rungs

The command rungs direct the 1771-KG module to send command messages to a remote station. Each command rung lists the type of command and the memory areas affected and allows command execution to be initiated through the user program.

There are two basic command rung formats which differ only in terms of the unit of memory which they control. The basic command rung formats are:

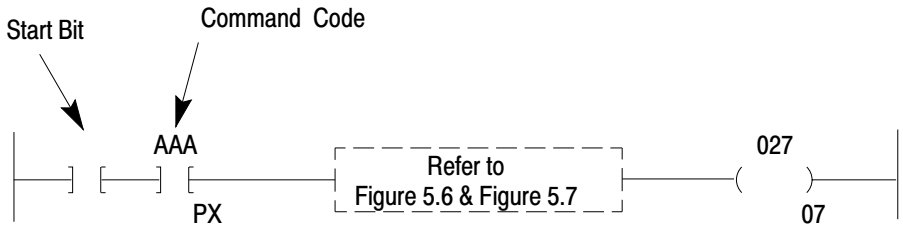
- Block Command Format
- Bit Command Format

The block command format is for commands which transfer one or more data table words between stations. These are unprotected block write, unprotected block read, and protected block write commands.

The bit command format is for commands which control one or more data table bits at another station. These are unprotected bit write and protected bit write commands.

In both formats the command rung begins in a similar manner (Figure 5.5). The first rung element is an EXAMINE ON instruction, addressing the start bit. The second rung element, termed the command code, tells the remote station number, type, and priority of the command. For most commands, use normal priority.

Figure 5.5
Command Rung Format



Legend

AAA	Remote Station Number
P	Priority Message
	1 = High
	0 = Normal
X	Command Type
	0 = Protected Block Write
	1 = Unprotected Block Read
	2 = Protected Bit Write
	3 = Unprotected Block Write
	4 = Unprotected Bit Write

11544

After the command code, the command rung then lists the memory areas affected by the command. The format of this area varies, based upon the type of memory area controlled by the command.

The communication zone may have up to 255 command rungs. A unique command rung is required for each command.

You should end each command rung with an energize 02707 instruction.

Examine Start Bit

Assign a unique start bit to each command rung. The start bits are examined by the 1771-KG module. When a start bit is ON, the module carries out the programmed command.

Control the start bit through your ladder diagram program. Turn it on when you want to send the command message to the remote station. Programming methods for start bit control are given later in "Controlling the Start Bit."

Command Code

Use the second element in a command rung for the command code. Through the command code you must identify the following:

- Number of the Remote Station to Which the Command Is to Be Sent
- Priority Status of the Command (High or Normal)
- Type of Command

The command code uses the address of an EXAMINE ON instruction but does not examine or control any bit in the data table of the local station processor (Figure 5.5).

Block Command Format

When you enter a command code specifying a protected block write, unprotected block read, or unprotected block write command, use the format shown in Figure 5.6.

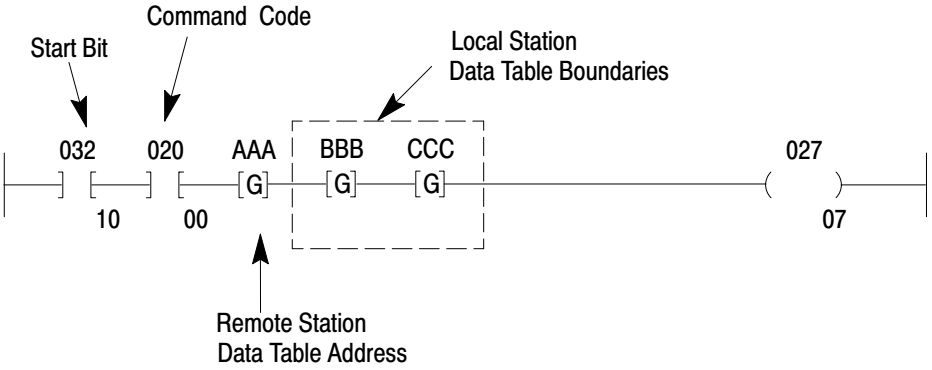
This format uses three GET statements. Use the address of the first GET statement to specify a remote station data table word. The specified command operation begins at this address. Use the second and third GET statement addresses to define the start and end of a block of data table words in the local station processor. This block contains words to be transferred in the write or read operation.

In the block write operation, data words are written to a remote station from the local station data table. The first GET element in the command rung specifies a beginning address at the remote station. Data is to be written into this word, and succeeding words, from the local station data table. The second and third GET elements in this type of rung list the starting and ending boundary for the word or words to be sent from the local station data table.

In the block read operation, data words are read from a remote station into the local station data table. The first GET element in the block read command rung lists the beginning address from which to read data. Remote data table words are read in succession, beginning with this address. This block of words is sent into the area of the local data table bounded by the second and third GET elements in the rung.

Only one set of GET instructions, as shown in Figure 5.6, is allowed in a single command rung.

Figure 5.6
Block Command Format



Legend

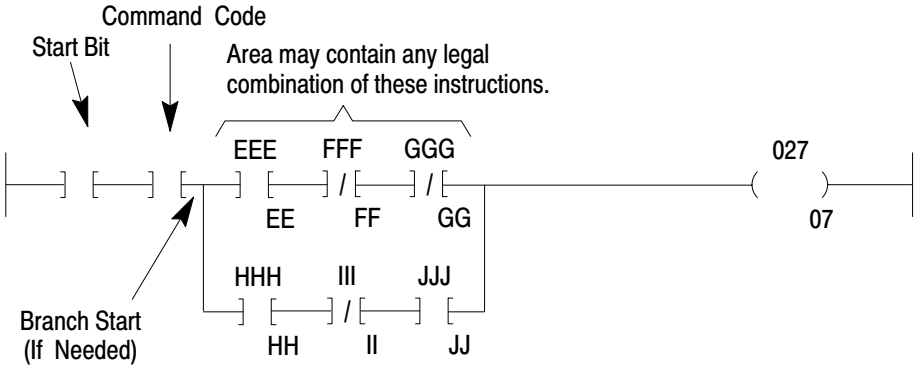
AAA	Beginning Word Address at REMOTE Station for WRITE/READ Operation
BBB	Beginning Word Address at LOCAL Station for WRITE/READ Operation
CCC	Ending Word Address at LOCAL Station for WRITE/READ Operation

11545

Bit Command Format

When you enter a command code specifying a protected bit write or unprotected bit write command, use the format shown in Figure 5.7.

Figure 5.7
Bit Command Format



Legend	EEEEE
-] [-	Turn ON the Addressed Bit at the Remote Station.
-] / [-	Turn OFF the Addressed Bit at the Remote Station.

11546

This format uses examine elements which address bits in the remote station. These elements control remote station data table bits as follows:

- -] [-EXAMINE ON — This rung element instructs the remote station to turn the address bit ON.
- -]/[-EXAMINE OFF — This rung element instructs the remote station to turn the addressed bit OFF.

These elements are programmed immediately following the command code. Any combination of these elements may make up this type of command rung. Where necessary, branch start instructions and a branch end instruction may be used to fit these elements into the display area of the programming terminal. For practical reasons, do not exceed this display area. Multiple rungs of this type can be used as necessary.

Status Words

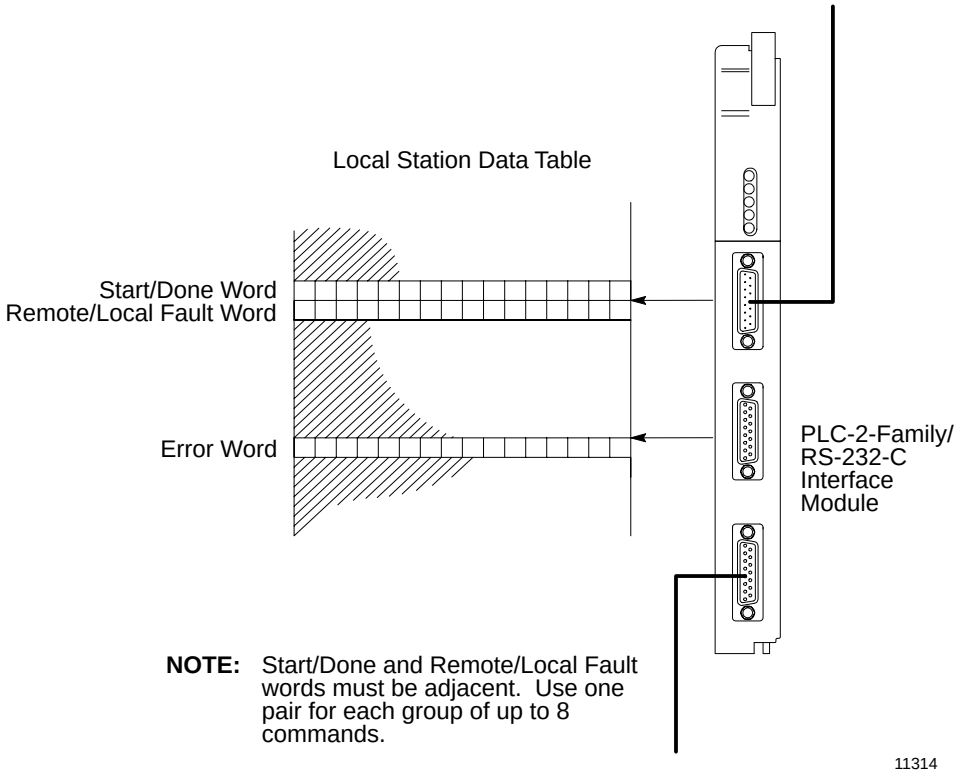
The module controls certain data table words specified by the programmer in the local station processor. These data table words indicate the status of command execution and provide various types of diagnostic information for start-up and troubleshooting. These words are:

- One or More Pairs of Adjacent Words for Start/Done and Remote/Local Fault Storage
- An Error Word

Refer to Figure 5.8. You specify the locations of these status words when entering the communication zone rungs. A pair of start/done and remote/local fault words is defined by the selection of the start bit in a command rung. You specify the error code storage word in the header rung.

Any accessible data table words may be used as status words in the station processor.

Figure 5.8
User-Selected Status Words



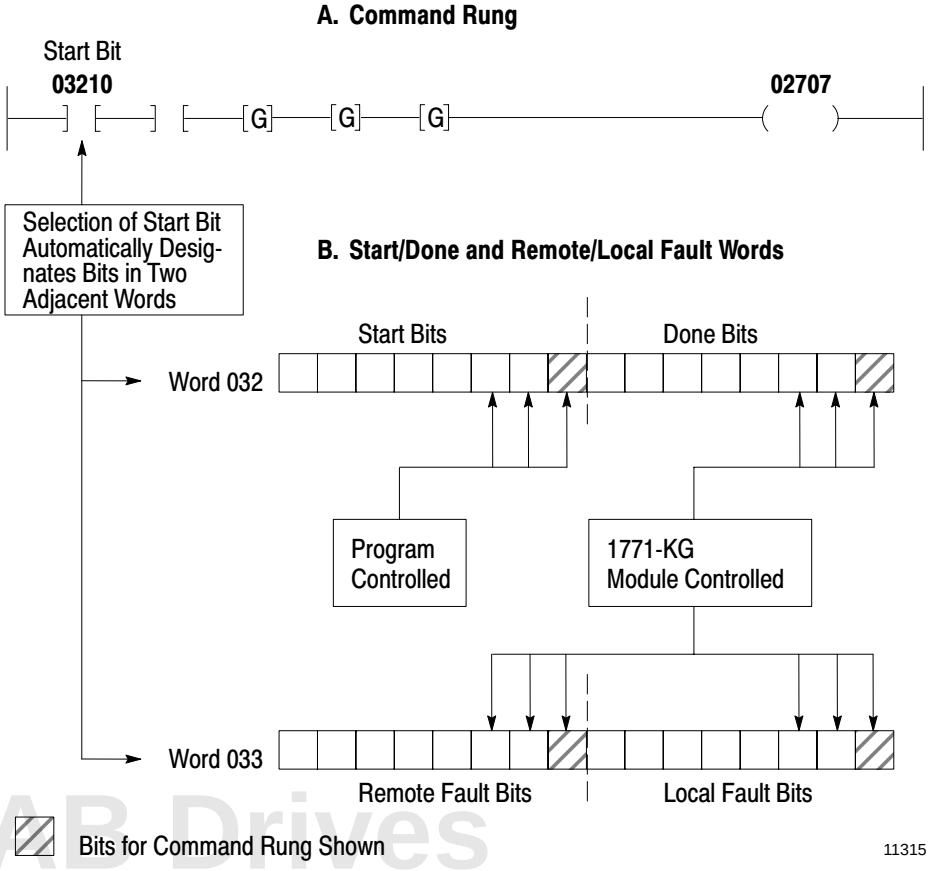
Start/Done Word

The start/done word is the first of a pair of adjacent status words for module use. This word stores a start bit and corresponding done bit for each of up to 8 commands. (NOTE: Should more than 8 command rungs be programmed at a station, additional pairs of start/done and remote/local fault words must be used.)

Select the start bit for any command in the upper byte (Bits 10-17) of the start/done word. The done bit for this command is then the corresponding bit in the lower byte (Bits 00-07) of the same word.

In Figure 5.9.A, the sample command rung examines Bit 03210 as its start bit. As Figure 5.9.B shows, the corresponding done bit is Bit 03200. Note that the address of start/done bits for each command differs only in the next to last digit; the start bit always has a one (1) in the next to last digit, the done bit always has a zero (0).

Figure 5.9
Adjacent Status Words



As its name implies, the start bit initiates command execution. This bit, controlled by the user program, is set ON to initiate the sending of a command. The 1771-KG module monitors the status of start bits and executes the corresponding command when its start bit is set ON.

As its name implies, the done bit, indicates command completion. This bit, controlled by the module, is set ON when the execution of a command is completed successfully.

Each command rung must examine a unique start bit. Thus, a single pair of start/done and remote/local fault words has enough bits for up to eight command rungs. Should more command rungs be programmed, select additional word pairs as necessary.

To optimize memory use and minimize the time required by the 1771-KG module to scan start bits, use all eight start bits in one word pair before using another start/done word. For the same reason, sequentially group command rungs which use the same start/done word. You cannot use Word 077 or the last data table word for start/done words.

Remote/Local Fault Word

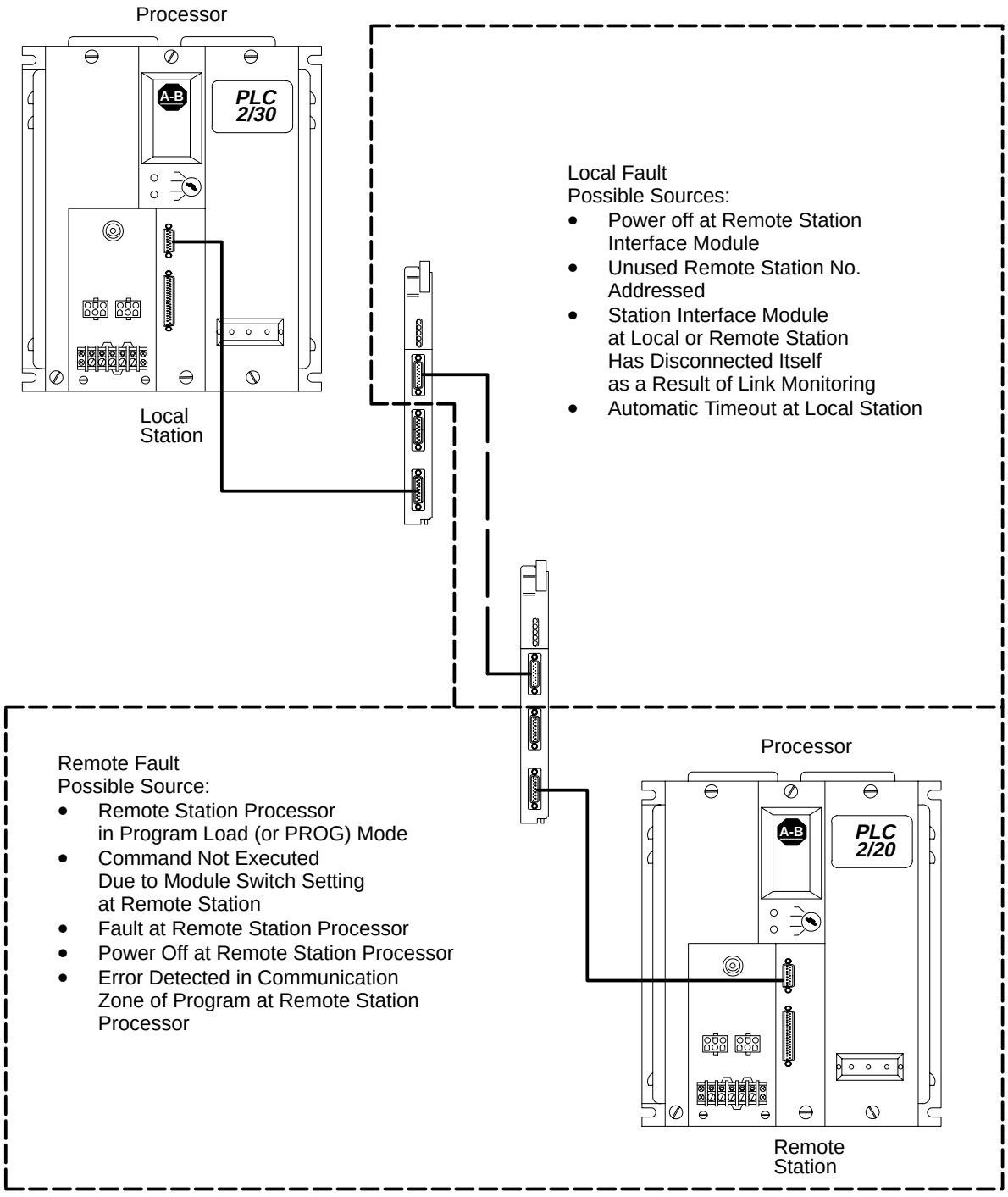
As Figure 5.9 shows, that selection of a start bit in the program not only causes the 1771-KG module to control a corresponding done bit to be controlled in that same word, but also causes the 1771-KG module to control remote and local fault bits in the next data table word. A remote fault bit and local fault bit are controlled for each command. For a command, the position of each of these fault bits within their respective bytes corresponds directly to the position of start and done bits for that command.

Remote and local fault bits are controlled by the module. The module sets a fault bit ON when a command cannot be executed due to a hardware-related fault either between stations or between the remote station module and its station processor. Figure 5.10 summarizes the significance of these bits.

Note that a remote fault bit indicates that the remote module received a message, but could not communicate with its station processor to execute that command.

A local fault bit indicates that the local station has not received a response from the remote station to acknowledge that it received a valid command.

Figure 5.10
Remote/Local Fault Bit Significance



11316

Immediately before it sets a remote or local fault bit ON, the communication adapter module enters a four-digit error code into the error code storage word. The error code storage word is described in the next section.

By monitoring fault bits in the program, operator personnel can be alerted to hardware conditions which prevent normal transmission and execution of commands. (Programming techniques for monitoring fault bits are described in "Command Initiation and Monitoring.")

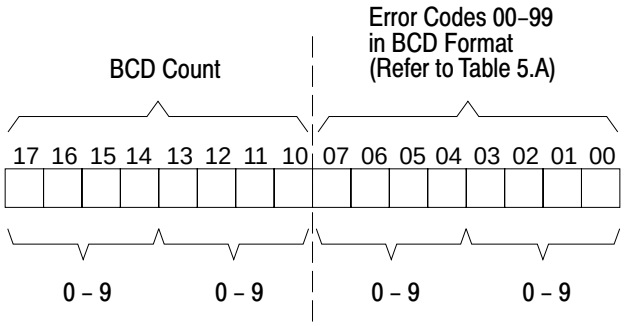
Error Word

When the 1771-KG module cannot carry out a command due to a programming error or a discrepancy in data, it writes an error code into the error word you select on the header rung of the communication zone. This word stores the most recent error code written by the module.

Figure 5.11 shows the structure of the error word. The lower byte of this word (Bits 00-07) stores any error code entered by the module. In this byte the error code is represented as a two-digit hexadecimal number from 00-99. Appendix A lists and describes these error codes.

Figure 5.11
ERROR CODE Word Format

Two–Digit Value:
- Reference Number (for Codes 01–29)
- Counter (for Codes 30–99)



Error codes can be grouped as follows:

- Codes 01-29 indicate a processor problem at power-up or that the 1771-KG module has detected some programming error in the communication zone of program. The program status indicator (PROG) may be ON if one of these codes is displayed.
- Codes 30-99 generally indicate that the 1771-KG module has detected some programming or hardware related fault during attempts at communication between stations. These codes are intended to serve as diagnostic indicators after the initial power-up checks of program have been completed.

The 1771-KG module writes a code in the 80-99 group whenever a remote or local fault bit is set ON.

The upper byte of the error code storage word (Bits 10-17) stores a two-digit BCD value. This value gives supplemental error or fault information, depending on the type of error code displayed. This value may have one of two meanings:

- Reference Number
- Counter

For Error Codes 01-29, the upper byte stores a two-digit reference number. This number points to the location of a programming error within the communication zone of the program. This error may be an incorrect instruction or an improper address entered within a rung of the communication zone. In this numbering scheme, the header rung is designated as “00.” Subsequent communication zone rungs are numbered sequentially.

Note that Error Codes 01-29 are intended as aids in start-up debugging of the communication zone. Thus, the reference number stored in this word can be a valuable tool for debugging purposes.

For Error Codes 30-99, the upper byte stores a two-digit counter. This counter shows the number of Error Codes 30-99 written into the storage word by the 1771-KG module. The counter increments each time the module enters a different error code.

Because Codes 01-29 are intended for start-up situations and Codes 30-99 are intended for situations subsequent to power-up, there is no conflict in controlling this upper byte of the error code storage word.

Error codes are generally used in the application program only for display. They have special value in station start-up, when program errors are detected in the communication zone. By viewing the header rung of the communication zone, you can examine a displayed error code and the least significant digit of the counter in this word. (The header rung is described earlier in "Header/Delimiter Rungs.")

In some instances, however, it may be preferable to display the two-digit error code using a 7-segment display controlled from output modules of the controller. This 7-segment display, mounted at an operator's station, can be a useful troubleshooting aid for quickly locating fault conditions.

NOTE: Error Codes 8A and 8B may not display correctly on some 7-segment displays.

Control of PC Programs

In addition to data table access, a computer on the Data Highway can access the user program area of memory of any station PC on the Data Highway. This enables computer control of PC operation, allowing storage of PC programs and downloading of these programs from the computer.

The following commands enable computer control of the PC program:

- Physical Read
- Physical Write

The prefix "physical" distinguishes these commands from the data transfer commands available to both the PC and the computer. Physical commands can only be issued by a computer.

Using physical read commands, the computer can store the ladder diagram program, including the data table, from any station PC. The computer can then use physical write commands to download a PC program, including the data table.

There are many advantages to computer capability for storing and downloading of programs for station PCs. The computer can store multiple programs for a PC, allowing quick change of a PC program without the need to manually load a new program. Programs can also be stored for backup, to prevent costly downtime if PC memory is inadvertently lost.

Command Initiation and Monitoring

This section describes the support programming for commands at each station processor. This programming uses the start/done and remote/local fault bits to initiate and monitor command execution. Using these recommended techniques, you coordinate communication zone programming with your application program.

Start/Done/Fault Bit Timing

The start bit which initiates a command is program-controlled; the done bit, which indicates command completion, is 1771-KG module-controlled. The timing relationship of start and done bits is important when you initiate and terminate commands. The following sections describe this relationship in normal operation and describe the automatic responses which result from faulted operation.

NOTE: The 1771-KG module's scan and its control of the fault and done bits is asynchronous to the PC processor's scan and its control of the start bits.

Normal Operation

Command execution begins when the user program turns a start bit ON, normally with a latch instruction. The module detects the ON state of this bit and then begins the operations necessary to format and transmit a command message.

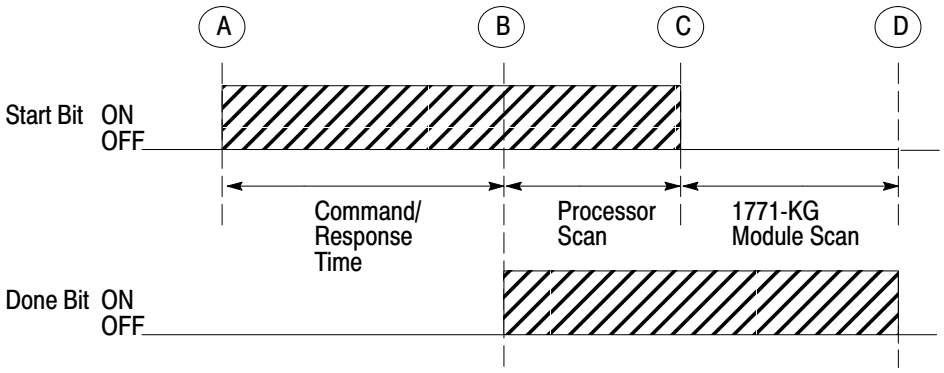
When the remote station module receives the command message, it acknowledges it. Then, while normal operation continues, the remote station module reacts to the command and prepares a reply message. (A reply message is sent for each type of command.) The remote station module responds to a poll for mastership, then transmits its reply message to the local (calling) station.

When the local station module receives the reply, it sets the done bit ON at the local station processor. The done bit, in turn, is examined in the user program to turn OFF the start bit.

After the start bit has been turned OFF, the module resets the done bit.

Timing of START and DONE bits for a command is shown in Figure 5.12. The significance of START/DONE bit status is summarized in Table 5.A.

Figure 5.12
START/DONE Bit Timing — Normal Operation



LEGEND:

(A) Start Bit turned ON by the program.

(B) Done Bit set ON by the 1771-KG module to indicate that a command has been completed.

(C) Start Bit turned OFF by the program.

(D) Done Bit set OFF by the 1771-KG module after it senses that the Start Bit has been set OFF.

11318

Table 5.A
Start/Done Bit Status

Status Start Bit	Done Bit	Significance
0	0	Idle
1	0	Command initiated or in progress.
1	1	Command/reply operation complete.
0	1	The processor, based on the program, acknowledges completion of the command/reply operation (transient condition, since the 1771-KG module turns the done bit OFF in its next scan).

Faulted Operation

Certain fault conditions can prevent normal reception and execution of commands by the remote station. To indicate the source of such fault conditions, the module controls remote and local fault bits at the local station processor.

In general, the local fault bit indicates that the intended receiving station is unable to receive (and acknowledge) a command message addressed to it. The remote fault bit, on the other hand, indicates that the intended remote station communication module has received the command message, but is unable to execute the command at the remote station processor. (For a summary of the distinction between these fault types, refer to Figure 5.10.)

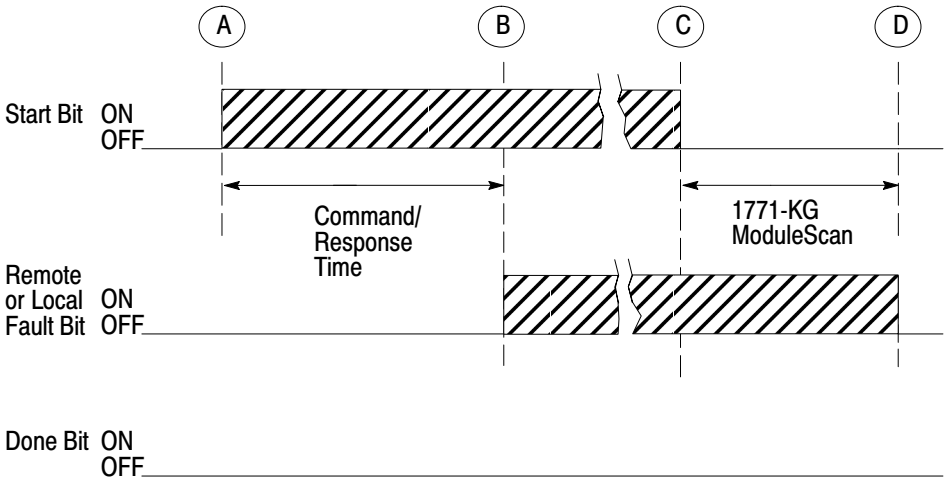
The timing relationship of these fault bits in command execution is shown in Figure 5.13. In this example, the command initiated at the setting of the start bit cannot be executed due to some fault condition. The module, detecting this fault condition, sets either a local or remote fault bit.

Recall that the start bit is program-controlled. The remote/local fault bits, meanwhile, are module-controlled. You must keep this relationship in mind when planning START bit control and FAULT bit monitoring in the ladder-diagram program.

Note from Figure 5.13 that the fault bit, once ON, remains ON until your program turns OFF the start bit. Only after it has detected that the program-controlled start bit is OFF does the module then turn the fault bit OFF. Note also that the done bit is not set ON in the event of a fault condition.

NOTE: In the special case where the start bit is turned OFF by the program before the module sets a done or fault bit, attempts to send that command are terminated. The local fault bit is pulsed ON for approximately 100 ms in this instance. This type of situation may occur, for example, if some event is programmed to unlatch the start bit before command completion.

Figure 5.13
START/FAULT Bit Timing - Faulted Operation



LEGEND:

(A) Start Bit turned ON by the program.

(B) Remote Fault or Local Fault Bit set ON by the 1771-KG module to indicate that a Fault Condition has been detected.

(C) Start Bit turned OFF by the program.

(D) Fault Bit turned OFF by the 1771-KG module after it senses that the Start Bit has been set OFF.

11319

Controlling the Start Bit

Your program controls the start bit; setting it ON to initiate command execution; turning it OFF after command completion or after a fault is detected. In several examples that follow, start bits are turned ON with latch instructions and OFF with unlatch instructions. This is done for ease of explanation but is not necessarily the best technique for your application.

To turn ON the start bit, the program examines application conditions. These may include input/output device data values, or other information from the controlled process. For the most part, an application condition used to initiate a command is one of these general types:

- ON or OFF Status of a Bit
- Transition of a Bit
- Timed Condition

To turn OFF the start bit, the program examines the response from the 1771-KG module. This response may be one of the following:

- Done Bit
- Remote Fault Bit
- Local Fault Bit

The next three sections describe commonly used forms of start bit control. Each section describes a different method for initiating command execution. All examples show how the program uses both done and fault bits to turn OFF the start bit.

An important assumption underlies the examples outlined in the next three sections and shown in Figures 5.14 through 5.16. The assumption is that the program should automatically retry transmission of a command in the event of a fault. Only the done bit terminates attempts at command execution. A local or remote fault bit response causes the program to re-initiate command execution. (This type of programmed retry is not to be confused with the retry procedure of the module itself. Transparent to your programming, the module automatically attempts three retries of a message before it sets a fault bit.)

Programmed retry has distinct advantages. As the module continuously tries to send the command message, any data content of the message (for a write command) is continuously updated. As soon as the fault condition is corrected, the message is sent, with the latest data. This eliminates the need for a manual reset of the start bit when a fault condition is corrected. Once the command is completed, the program automatically unlatches the start bit.

On/Off Input Status

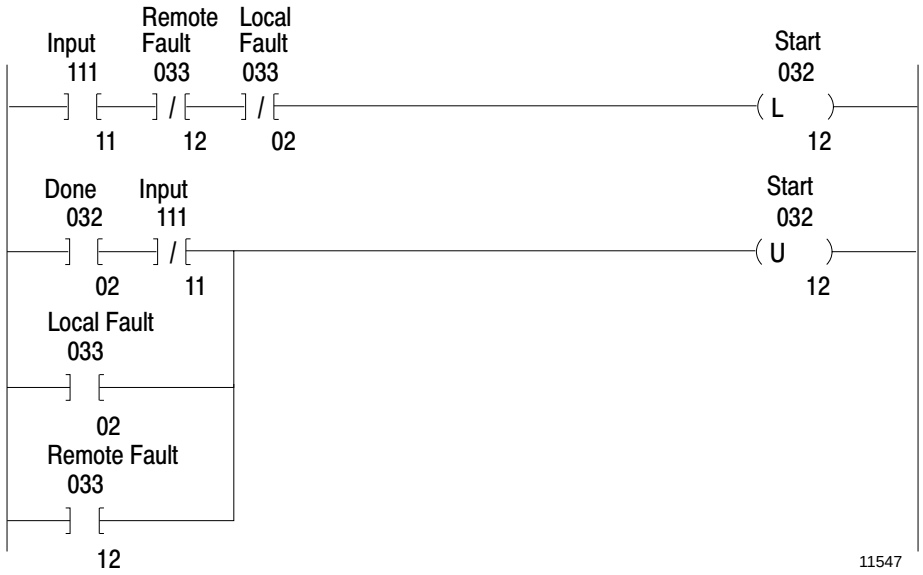
You can use the ON or OFF status of an input device to latch the start bit. Figure 5.14 shows example rungs for this type of command initiation.

In this example, the first rung is programmed to latch the start bit when the input, Bit 11111, is ON, provided that both fault bits are OFF. The second rung is programmed to unlatch the start bit based on the response of the module.

In normal operation, the start bit is latched by the input 11111. The fault bits initially are OFF. This causes the module to format and send the command message. Then, when the command is completed, the module turns ON the done bit. Note that input bit, 11111, must also be OFF to

unlatch the start bit in this example. With this arrangement, the command message is sent only once; input 11111 must be turned OFF, then ON again, to execute this command a second time. In normal operation, the start bit, after successful command completion, remains ON until input 11111 goes OFF. Recall from Figure 5.12 that the module holds the done bit ON until after the start bit is turned OFF.

Figure 5.14
Status-Initiated Command



Should a fault condition prevent normal execution, these rungs provide a programmed retry of the command as long as Bit 11111 is ON. A remote or local fault bit resets the start bit in the second rung. In the first rung, the start bit is latched again after the module resets the fault bit. As Figure 5.13 shows, the module resets a fault bit only after the start bit has been turned OFF.

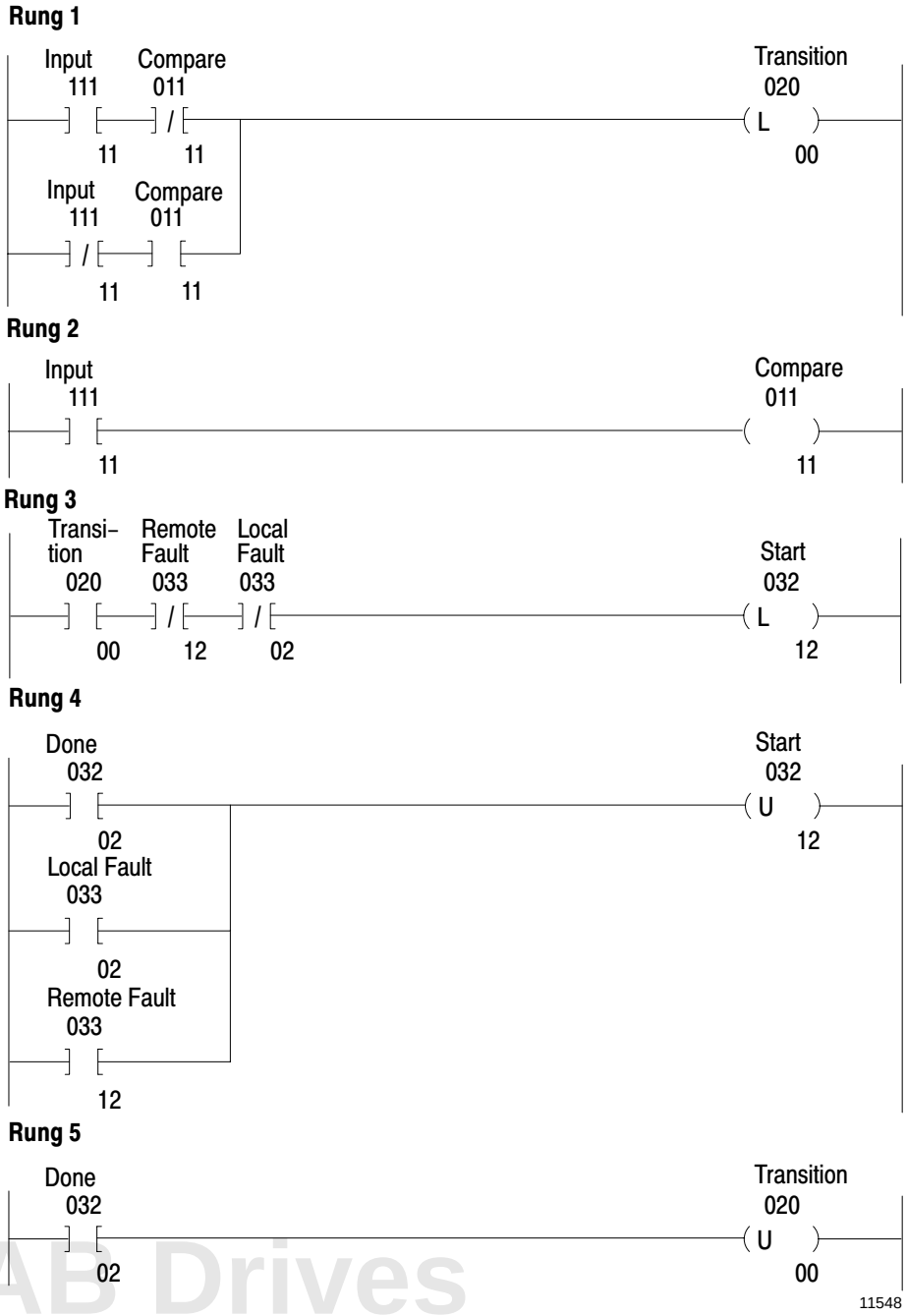
Even though the fault bits are continually reset with this method, their usefulness must not be overlooked. "Remote/Local Fault Bit Monitoring," which follows, outlines a useful method to monitor fault bits and control an output indicator based on fault bit status.

In some applications, it may be useful to send a command continuously between stations. With the example of Figure 5.14, you can accomplish this by eliminating the EXAMINE OFF instruction for Input 11111 in the second rung. This would cause the command to be sent continuously as long as Input 11111 remains ON.

Transition

You can use the transition of an input device from ON to OFF and from OFF to ON to latch the start bit. This allows you to send a command each time a condition changes state. Figure 5.15 shows example rungs for this type of command initiation.

Figure 5.15
Transition-Initiated Command



In this example, a storage bit, called the transition bit, is manipulated to control the sending of the command. This bit is latched whenever a transition of Input 11111 is detected, unlatched only when the done bit is set ON. A compare bit, 01111 is this example, is used to manipulate the transition bit. In Rung 2, the compare bit is controlled to match the ON/OFF status of the input. Because the input and the compare bit are programmed to have matching states, both ON or both OFF, the conditions of Rung 1 can be true only when the input has just changed from ON to OFF or from OFF to ON. Thus, Rung 1 conditions set up a one-shot, true only long enough to latch the transition bit. Note that these rung conditions are false as soon as the processor scans Rung 2. The order of these rungs is important for this reason.

With the transition bit latched, the start bit, in turn, is latched in Rung 3. This initiates the command. In normal operation, the done bit unlatches the start bit in Rung 4 and then unlatches the transition bit in Rung 5. In faulted operation, however, Rungs 3 and 4 repeatedly retry the command in much the same manner as in the example of Figure 5.14.

Timed

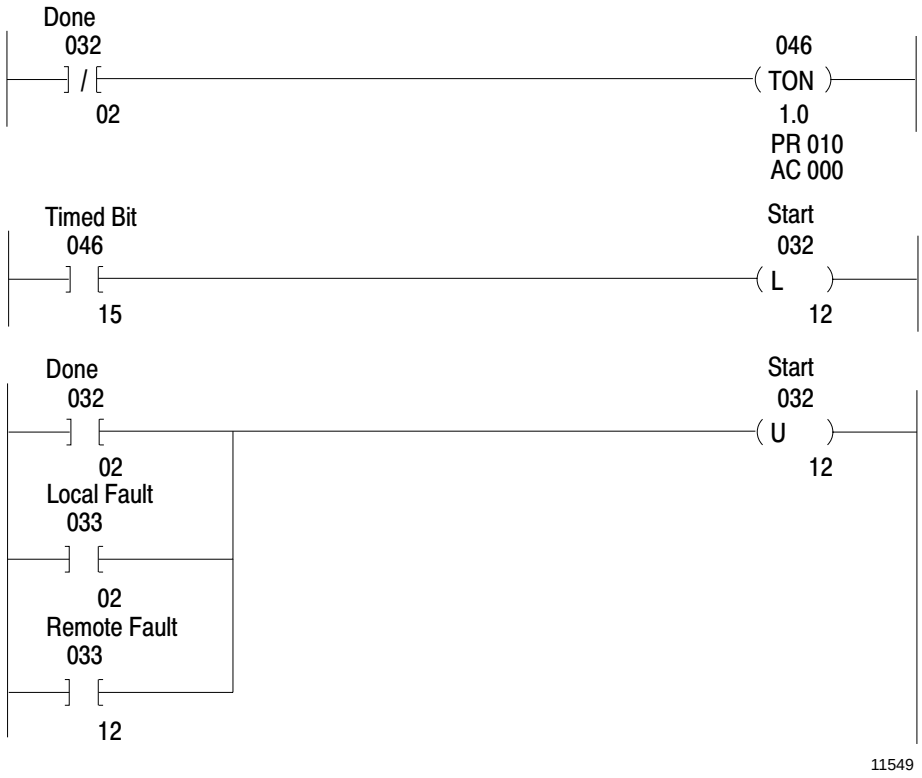
You can latch the start bit periodically to send a command at a user-determined interval. Figure 5.16 shows example rungs for this type of command initiation.

In this example, Timed Bit 04615 is used to initiate the command at every preset interval, 10 seconds. This bit is examined to latch the start bit. The done, local, fault, and remote fault bits are examined in parallel branches to unlatch the start bit.

In normal operation, after the command is executed, the done bit is set ON by the module. This causes the program to unlatch the start bit. The timer then begins timing again once the done bit is OFF. (As Figure 5.12 shows, the done bit is reset only after the start bit is reset.)

Note that this programming causes continuous retry of a command in the event of faulted operation.

Figure 5.16
Timer-Initiated Command

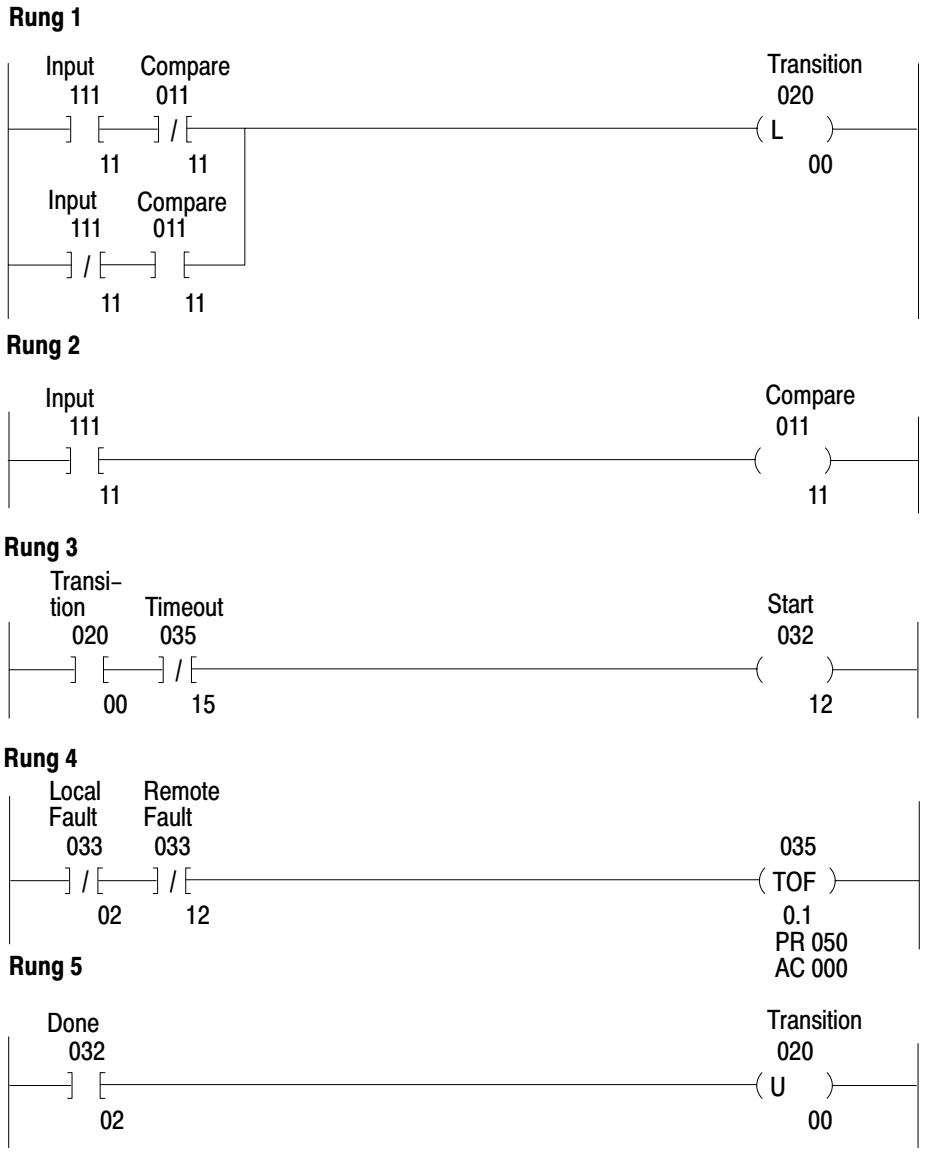


Retry Delay

In some cases, automatic retry as implemented in Figure 5.14, Figure 5.15, and Figure 5.16 can load down the network unnecessarily. Message transfer attempts that result in local faults can take many times as much time as the normal message transfer time. This could adversely affect the performance of other stations.

You can reduce the impact of automatic retries by providing a time delay before each retry as shown in Figure 5.17. In this example, a 5s time delay was added to the example in Figure 5.15. When the local or remote fault bit goes ON, the timer starts. Only after the 5s time delay is the start bit turned ON again to initiate an automatic retry of the command.

Figure 5.17
Retry Delay



11550

Remote/Local Fault Bit Monitoring

When it cannot execute a command, the module sets a remote or local fault bit ON. These bits, in the data table of the local station processor, are located in the word immediately following the start/done word. They indicate not only that a command was not executed but also point to the general type of fault condition which prevented command completion.

The user program must monitor the remote and local fault bits for each command. The recommendations of this section describe two methods for monitoring fault bits and using these bits to signal a fault condition.

Diagnostic Fault Rungs

The purpose of remote/local fault bit monitoring is to control one or more output indicators to signal a fault condition. Fault indicators controlled for this purpose may be as simple as an annunciator or as complex as a printer or CRT terminal used to display a fault message. By controlling the fault indicator device, the user program can alert personnel to the nature and location of a fault condition.

In order to monitor the remote and local fault bits, the programmer must understand their timing relationship to the corresponding start bit. Figure 5.13 summarizes this relationship.

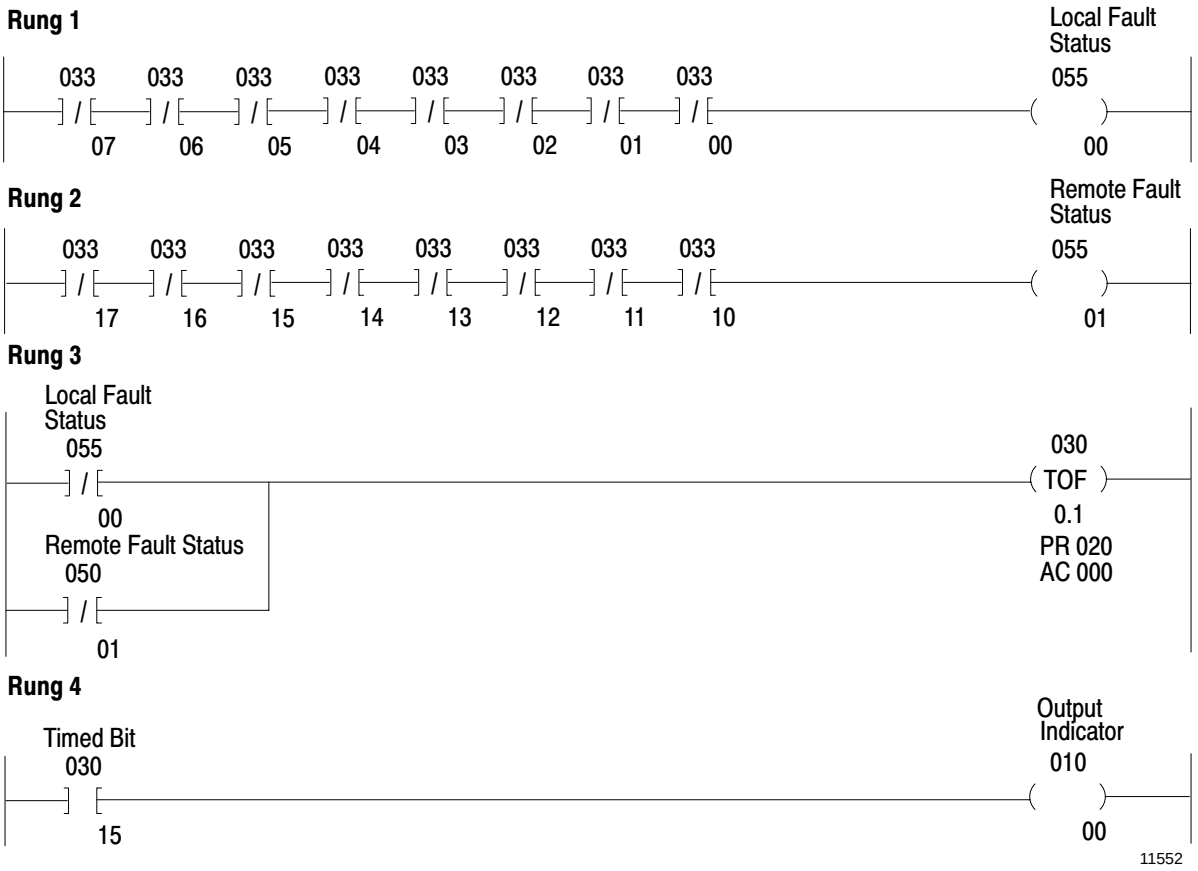
"Controlling the Start Bit" showed how the fault bits can be programmed to unlatch the start bit in a fault situation and thus provide automatic retries through the program. When used in this manner, however, a fault bit will be rapidly cycled ON and OFF if a fault is detected. Because the fault bit can be constantly changing state at a rapid rate, the program must use some method of detecting this transient state of any fault bit and controlling the output device based on this state.

Figure 5.18 shows a simple method for the control of a fault indicator. Here, either the remote or local fault bit can turn ON the output indicator. The indicator remains ON until the done bit is energized. This then unlatches the output indicator in the second rung. This example allows for the transience of the fault bits, since the first rung need only be true once for the output indicator to be latched.

The method of Figure 5.18 can be extended to monitor multiple commands from a station, controlling multiple output indicators as necessary. However, where more than one command is sent from a station, the use of multiple output indicators may not be practical. In this instance, a single output indicator can be used to signal all remote or local fault conditions for commands from that station. Figure 5.19 shows a method for fault indicator control assuming multiple commands.

Figure 5.18
FAULT Bit Diagnostic Rungs (Single Command Example)

Figure 5.19
FAULT Bit Diagnostic Rungs (Multiple Commands Example)



This example shows the fault bit monitoring for eight commands. The eight local fault bits are monitored in Rung 1. As long as all eight bits are OFF, Status Bit 05500 remains ON. However, should any local fault bit be ON, Status Bit 05500 is de-energized. In Rung 2, the eight remote fault bits are monitored in the same manner, to control Status Bit 05501.

The status bits controlled by Rungs 1 and 2 are, in turn, used to control an off-delay timer in Rung 3. The off-delay timer begins to time when either of the status bits goes from OFF to ON, that is when rung conditions go from true to false. Bit 03015, the timed bit of the timer, controls the output indicator. As soon as the conditions of the timer in Rung 3 are true, this bit is set ON, causing the indicator to be energized. Once ON, this bit remains ON as long as the timer is timing; that is, for at least as long as the preset interval. In the example of Figure 5.19, this preset is 2 seconds. It is recommended this value be set at no less than 0.5 seconds.

The off-delay timer is useful in this application because it is continually reset when its rung conditions go true. This means that the timed bit, Bit 03015, remains ON for as long as any fault bit is changing state during programmed retries. This keeps the output indicator ON until after the done bit indicates command completion.

NOTE: Using the rungs of Figure 5.19, the indicator goes ON automatically at power-up, or whenever the mode select switch on the processor is changed from PROGRAM LOAD to any other position. However, the indicator only remains on initially for the preset interval. After this time, the indicator is valid for fault conditions.

Note that Rungs 1 and 2 examine all eight fault bits of each type. Should fewer than eight command rungs be programmed at a station processor, fewer bits need be examined. Then, should command rungs be added subsequently, the appropriate bits could be addressed in Rungs 1 and 2. Conversely, if more than eight command rungs were programmed at a station, additional rungs would be needed to examine both remote and local fault bits for the additional commands. Status bits controlled by these additional rungs could then be examined in branches of Rung 3, parallel to those shown.

Of course, other methods can be used to monitor remote and local fault bits. Such factors as availability of output terminals, memory space, and type of application dictate the specifics of fault bit monitoring and program response.

The use of fault bits in start-up and troubleshooting procedures is described in Chapter 7.

Synchronizing Fault Bits

As noted earlier, the 1771-KG module's scan and its control of the fault and done bits is asynchronous to the PC processor's scan and its control of the start bits.

In the examples discussed so far, as in most applications, this lack of synchronization does not present any problems. However, you may want to use the PC program to count the number of times a fault bit turns ON and/or to store the error code each time a fault bit turns ON. Because of the asynchronous relationship, you may not be able to do these types of functions accurately by examining fault bits directly.

Instead of examining the fault bits directly, use a GET/PUT rung at the beginning of the program to copy the fault word into another word. You can then examine the corresponding bits of the other word any time during the program scan to determine the status of the fault bits as copied at the beginning of the program scan. Even though the 1771-KG module may change the state of fault bits in the middle of a program scan, the status of the bits in the copy word will not change until the start of the next program scan.

Timeout Preset Value

In addition to its remote/local fault bit control, the 1771-KG module also provides an automatic timer for monitoring command completion. While it functions automatically during module operation, this timer uses a preset value entered in the user program. This feature enables the module to monitor command execution time without using timer (TON) instructions in the user program.

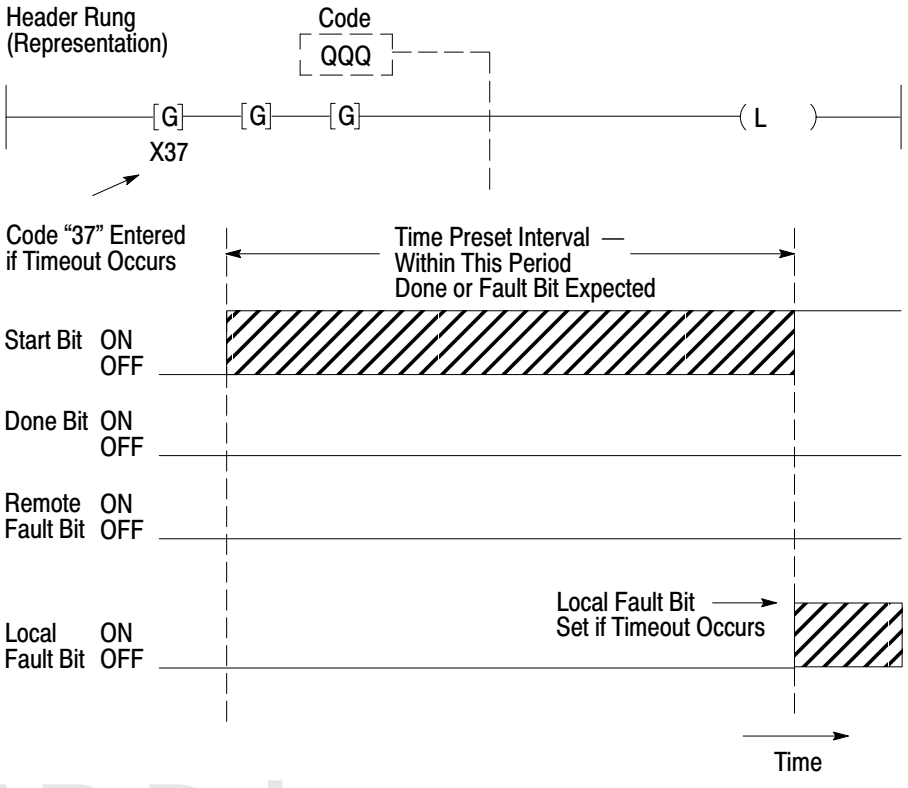
Figure 5.20 shows the significance of the timeout preset interval. From the time the start bit is set ON, the 1771-KG module must set either a done bit or a remote or local fault bit within the timeout preset interval. Should the module detect no done or fault bit response within the preset interval, a fault is assumed. As a response to this type of timeout situation, the module turns ON the local fault bit and enters the value 37 in the lower byte of the error word defined by the header rung.

Timeout preset monitoring is intended as a backup for the other communication monitoring functions of the 1771-KG module. It is designed to signal any condition where the module has not completed its communication with another station or detected some fault condition within a short time. Because this timer is primarily intended as a backup for some local fault type of situation, its preset value is not critical. In the examples of this publication, a nominal value of 5 seconds, (coded 015) is programmed as the timeout preset value. This value is appropriate for most applications.

Programming the Preset Code

You enter the timeout preset code in the Header rung of the communication zone of program. The address field of the third GET instruction in this rung is used for the timeout preset code. Figure 5.20 shows the position of this rung element.

Figure 5.20
Timeout Preset Significance



The address of a GET instruction is an octal number. Because only octal values can be entered in this address field the timeout preset value is a code, computed as outlined in this section.

The timeout preset is not a critical value. For most applications, a five-second preset is acceptable. The code for the timeout preset is 015. This code is used in all header rungs shown in this publication. However, there may be instances where another timeout preset interval is desired. Table 5.B lists the three-digit codes for intervals from 2 to 10 seconds. The tolerance of this timer is +0 and -1 second.

Table 5.B
Timeout Preset Codes

Timeout Interval (Sec.)	Code
Disabled	010
2	012
3	013
4	014
5	015
6	016
7	017
8	020
9	021
10	022

If it is necessary to use some value other than those provided in Table 5.B, compute the three-digit timeout preset code as follows:

1. Select a timeout preset interval. This interval must be larger than 1 second. For the purpose of computing the code, label this number S.

EXAMPLE: Desired Interval = 7 = S

2. Compute a decimal number using the desired interval of Step 1 in the following formula:

S + 8

EXAMPLE: 7 + 8 = 15

3. Convert this value to an octal value.

EXAMPLE: $15_{10} = 17_8$

For a brief description of decimal-to-octal conversion, refer to the Programming and Operation Manual of the controller being used.

User-Programmed Timeout (Optional)

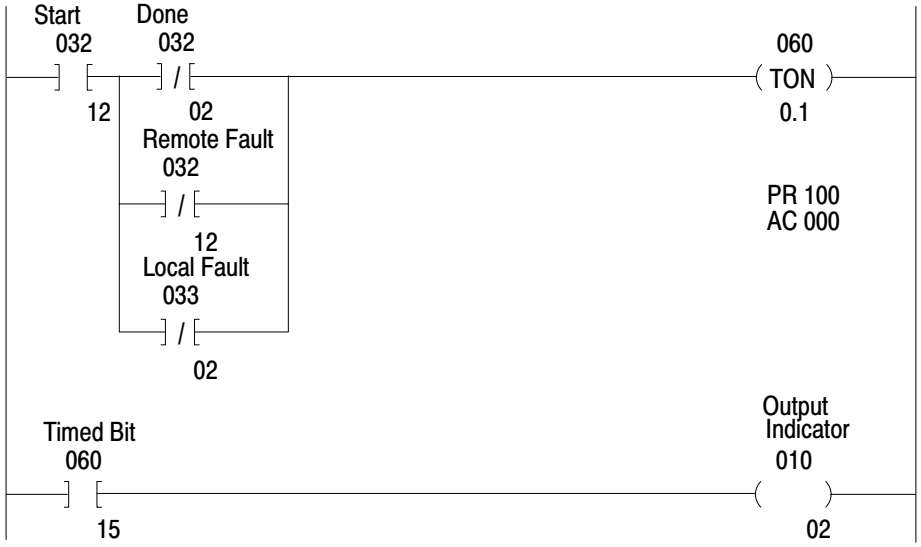
The automatic timeout of the module is a backup function. This automatic monitoring routine continuously checks module interaction with other stations, timing the execution of commands. This automatic timeout does not check module communication with its own station processor. For a backup check on module/processor communication, you might use an optional programmed on-delay timer instruction (TON).

With proper module processor communication, the module sets either a done bit or a remote or local fault bit as a response to a command. However, in the event of faulted module/processor communication or faulted module operation, a response bit might not be set. Instead, the start bit would remain ON in this instance until the fault situation was corrected. You can use several programming methods to detect such a condition; the simplest of these methods uses an on-delay timer. Figure 5.21 shows typical rungs that you can program for this purpose.

In the first rung of this figure, Timer 060 times the interval between the setting of the start bit for a command and the done, local fault, or remote fault response of the module. If no response is received within the preset interval of this timer, here 10 seconds, a fault may be indicated and Bit 06015 set ON. The second rung examines this bit to turn ON an annunciator. Depending on the individual application, you could also use this bit to enable or disable various parts of the program.

The preset value of this programmed TON instruction is not critical. For this type of backup monitoring, the programmed preset must exceed the timeout preset interval entered as a code in the header rung. (Remember that the automatic timeout of the module gives a local fault response to a command which would indicate normal module/processor communication but faulted communication with some other station.)

Figure 5.21
Typical User-Programmed Timeout



As with automatic timeout preset monitoring, a user-programmed timeout is useful as a backup to the other monitoring functions of the 1771-KG module. Remote and local fault bits at other stations indicate the same types of faults that can be detected using a user-programmed timeout. A programmed timeout would not be necessary for each command from a station. Instead, you can monitor a single command at each station in this manner. Select a command that is sent regularly for this type of monitoring.

There may be other instances where program monitoring of commands is useful. In some cases, you might program a timeout to monitor the execution time of critical commands. An application may require that a critical message, such as a priority command, be sent within a certain limited amount of time. You can program a TON instruction for this purpose; here, however, its preset interval will generally be shorter than the interval entered as the timeout preset for the module.

Memory Access Limitations

Before transmitting write commands to a remote station, you should become aware of what areas of that station's data table is accessible to you. This information is available either in the programming manual for that processor or the manual for its communication module. Further general memory access limitations are discussed in the following sections.

Protected/Unprotected

The memory access of protected commands is controlled by the receiving station. These commands may access only data table areas specified in the program at the receiving station processor. A memory access rung, programmed at the receiving station processor, defines which memory areas are accessible to a protected write command. Should a protected command address a memory area not defined by a memory access rung at the receiving station, that command is rejected.

The memory access of both unprotected read and write commands is not restricted by the program at the receiving station. These commands can access any addressable data table word without the need for a memory access rung at the receiving station. Unprotected commands may be disabled by option switches at either remote or local station for most communication modules.

The primary distinction between protected and unprotected commands is program restriction of memory access.

NOTE: For most write operations between station processors, you should use protected commands. Because memory access must be allowed by the program at the receiving station processor, protected commands allow programmed write protection, which gives the programmer an added degree of control over command execution. Commands of the unprotected type provide the same functions in transferring data but without this write protection at the receiving station.

Bit Write Access

You can use the bit write command to control any accessible data table bit. However, this command must not be used to control the following:

- Any bit whose status is controlled by a programmed output instruction.
- In any PLC-2 Family Processor: Any bit in a byte which also contains program-controlled bits.

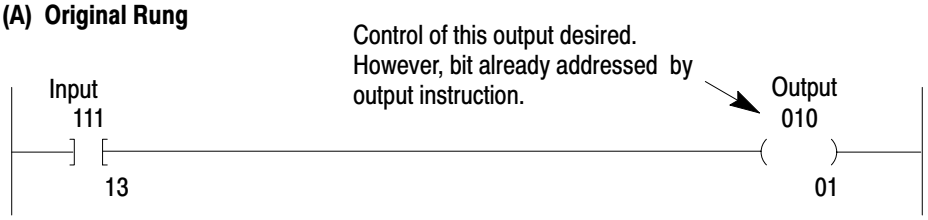
The **first restriction** simply states that you should not attempt to directly control a bit by both an output instruction at its local station processor and a bit write command from some remote station processor. This rule applies to all processors.

Bit write commands are generally used to set storage bits in a remote station processor data table. These storage bits may then be examined in the user program as conditions to energize an output bit. This indirect programming technique allows control using bit write commands but helps to prevent the confusion that can result if you attempt to control a bit directly from both an output energize instruction and a bit write command.

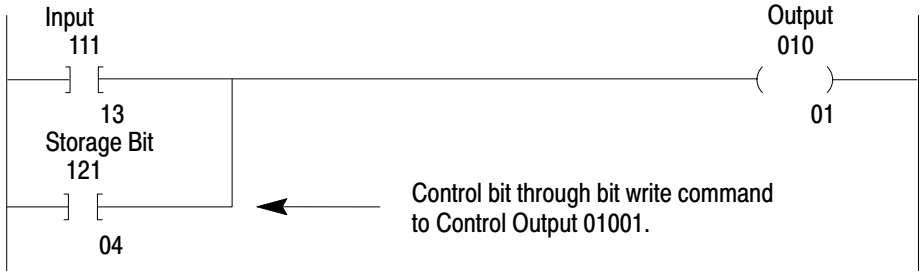
Figure 5.22 gives an example of an indirect programming technique used to control Bit 01001. Here Storage Bit 12104 is controlled by the bit write command. This bit is then examined by the program to control the status of Bit 01001. Note that Output Bit 01001 cannot be directly addressed by the bit write command. However, by controlling the storage bit and examining that storage bit in the program, the desired effect is achieved.

Figure 5.22
Bit Control Use (Example)

(A) Original Rung



(B) Recommended Technique



11554

The **second restriction** listed here applies to all PLC-2 Family processors. For these processors, when the station communication module receives a bit write command, it manipulates the eight-bit byte of the 16-bit memory word in which the addressed bit is located. This may be the low byte, containing Bits 00-07, or the high byte containing
Bits 10-17. Should program instructions control other bits within this same byte, there is a slight possibility that the module may write over programmed status for these program-controlled bits. This would occur only if the program caused a bit to be altered while the module was executing a received bit write command.

Therefore, when using the bit write command, address only bits within a byte which is set aside exclusively for control by these commands.

Note that this byte restriction for bit write commands does **not** apply to PLC processors.

Command Priority

Occasionally, it is necessary to carry out critical commands ahead of the normal sequence. You can select high priority commands to be handled more quickly along the Data Highway link.

Each message handled by the 1771-KG module has one of these priority levels:

- High Priority
- Normal Priority

Priority levels determine the order in which messages are transmitted through a Data Highway link. High priority messages are transmitted and executed before normal priority messages.

The programmer designates a priority level for each command message. The command code, an element in each command rung, specifies the priority level of the command message. The station which receives a command message automatically establishes the same priority level for its corresponding reply message. (The command code is described this chapter under “Command Code.”)

NOTE: High priority commands are executed ahead of normal commands throughout the command/reply message cycle. For this reason, give a command a high priority designation only when special handling of specific data is required. Using an excessive number of high priority commands defeats the purpose of this feature and could delay or inhibit the transmission of normal priority messages.

Chapter 6

Communication Protocol

General

This chapter describes the communication protocol used on an RS-232-C link to the 1771-KG module. It explains the programming you need to do at both the data link level and the network management level to enable communication over the RS-232-C physical link. Specifically, this chapter outlines the logic for input/output drivers (transmitters and receivers) used on the RS-232-C link. Also described are the formats of the different types of messages that can be generated at the application level for transmission over the link.

If you are connecting the 1771-KG module to another Allen-Bradley communication interface module (such as a 1771-KG, 1775-KA, 1773-KA, or 1771-KE/-KF module), then you need not be concerned with the protocol described here because the modules automatically take care of it. However, if you are connecting the 1771-KG module to a computer, then you must program the computer to understand and to issue the proper protocol as described in this chapter.

Definition of Link and Protocol

A physical link consists of a cable and associated hardware, such as transmitter and receiver circuits. Protocol is the set of programming rules for interpreting the signals transmitted over the physical link by the hardware devices.

You can connect the 1771-KG module to either of two types of physical links:

- Point-to-Point Link
- Multi-Drop Link

You can select the 1771-KG module to provide either:

- Peer-to-Peer Communication through a Full-Duplex, Unpolled Protocol
- Master/Slave Communication through a Half-Duplex, Polled Protocol

The type of communication protocol you can use depends on the type of physical link you have:

- For a point-to-point link, you can use either a peer-to-peer or master/slave communication protocol.
- For a multi-drop broadband MODEM link, you can use either a peer-to-peer or master/slave communication protocol.
- For a multi-drop baseband MODEM link, you must use a master/slave communication protocol because the link can support only one channel.

In general, full-duplex protocol gives higher data throughput, but it can handle communication between only two peer stations. Half-duplex protocol provides master-slave polling capability and can handle communication with as many as 255 slave stations, but it gives lower data throughput.

Half-duplex protocol has a data link layer to establish a communication channel between the master and one of the slave stations, and it has a network management layer to control the flow of communication across the established channel.

Whether you use full-duplex or half-duplex protocol, you should use a layered approach to developing communication software for your computer. You don't have to design your communication software in this layered fashion, but your software must perform all the functions described for the layers in this manual. In most cases, it will be easier for you to implement and debug the communication software if you do follow this layered approach. The three layers we refer to are:

- Data Link Layer
- Network Layer
- Application Layer

"Full-Duplex Protocol" describes the data link layer for full-duplex protocol. "Half-Duplex Protocol" describes the data link layer for half-duplex protocol. "Network Layer" describes the network layer. "Application Layer" describes the application layer. "Upload/Download Procedures (Series B Modules)" describes upload and download procedures.

Full-Duplex Protocol

The full-duplex protocol resembles ANSI X3.28-1976 specification, combining features of Subcategories D1 (data transparency) and F1 (two-way simultaneous transmission with embedded responses).

You can use full-duplex protocol for a point-to-point link or a multi-drop broadband MODEM link that allows two-way simultaneous transmission. It is more difficult to implement than half-duplex because it requires you to use interrupts and multi-tasking programming techniques. It is intended for high-performance applications where you need to get the highest possible throughput from the available communication medium.

Transmission Codes

Full-duplex protocol is a character-oriented protocol that uses the following ASCII control characters extended to eight bits by adding a zero for Bit 7. See ANSI X3.4, CCITT V.3, or ISO 646 for the standard definition of these characters.

Control Character	Hexadecimal Code
STX (Start of Text)	02
ETX (End of Text)	03
ENQ (Enquiry)	05
ACK (Acknowledge)	06
DLE (Data Link Escape)	10
NAK (Negative Acknowledge)	15

Additionally, a block check character (BCC) or two-byte cyclic redundancy check (CRC) field is used at the end of each packet for error checking. These bytes can be any value from 00 to FF hex.

As used in the following paragraphs, a code is an indivisible sequence of one or more bytes having a specific meaning to the protocol. Indivisible means that the component bytes of a code must be sent one after another with no other bytes between them. It does not refer to the timing of the bytes.

Full-duplex protocol uses these codes:

- Control Codes:
  - DLE STX
  - DLE ETX BCC/CRC
  - DLE ACK
  - DLE NAK
  - DLE ENQ

- Link-Layer Data Codes:
  - Data (single bytes having values 00-0F and 11-FF hex)
  - DLE DLE (to represent the value 10 hex)

We can also group codes into two classes according to their use: codes issued from a station transmitting a message, and response codes issued from a station receiving a message. By this classification, the full-duplex codes are:

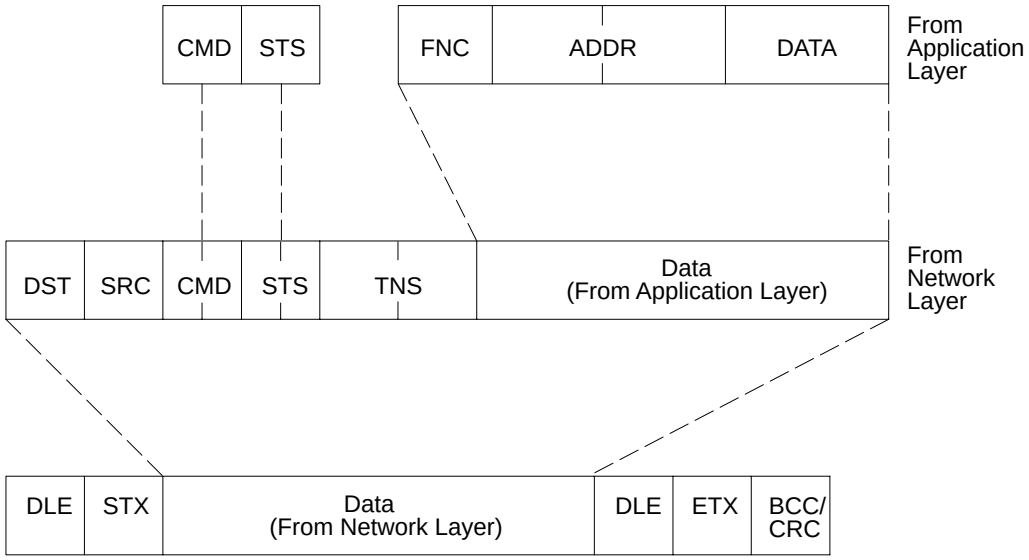
- Codes from Station Transmitting a Message:
  - DLE STX — Indicates the start of a message packet.
  - Link-Layer Data (00-0F and 11-FF hex) — Encodes the bytes of the network packet.
  - DLE DLE — Encodes the value 10 hex in the network packet. This is necessary to distinguish a text code of 10 hex from a DLE control code of 10 hex.
  - DLE ETX BCC/CRC — Terminates a message packet.
  - DLE ENQ — Requests the retransmission of the last received transmission.
- Response Code from Station Receiving a Message:
  - DLE ACK — Signals that the receiver has successfully received the last message sent.
  - DLE NAK — Signals that the receiver did not successfully receive the last message sent.

Link-Layer Message Packets

A link-layer message packet starts with a DLE STX, ends with a DLE ETX BCC/CRC, and includes all link-layer data codes in between. Data codes can occur only inside a message packet. Response codes can also occur between a DLE STX and a DLE ETX BCC/CRC, but these response codes are not part of the message packet; they are referred to as embedded responses.

Figure 6.1 shows the format of a link-layer message packet for full-duplex protocol, and the layer at which each portion should be implemented. At the end of each message packet is either the one-byte BCC or two-byte CRC field. With the Series A module, you must use BCC. With the Series B module, you can select either BCC or CRC through switch settings (Chapter 4).

Figure 6.1
Link Packet Format for Full-Duplex Protocol



Block Check

The block check character (BCC) is a means of checking the accuracy of each message packet transmission. It is the 2's complement of the eight-bit sum (modulo-256 arithmetic sum) of all data bytes between the DLE STX and the DLE ETX BCC. It does not include any other message packet codes or response codes.

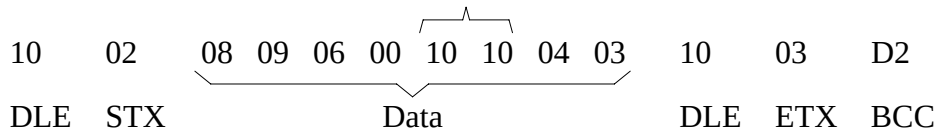
For example, if a message packet contained the data codes 8, 9, 6, 0, 2, 4 and 3, the message packet codes would be (in hex):

10	02	08	09	06	00	02	04	03	10	03	E0
DLE	STX	Data							DLE	ETX	BCC

The sum of the data bytes in this message packet is 20 hex. The BCC is the 2's complement of this sum, or E0 hex. This is shown in the following binary calculation:

0010 0000	20 hex
1101 1111	1's complement
+1	
1110 0000	2's complement (E0 hex)

To transmit the data value 10 hex, you must use the data code DLE DLE. However, only one of these DLE data bytes is included in the BCC sum. For example, to transmit the values 8, 9, 6, 0, 10, 4, and 3 hex, you would use the following message codes:



In this case, the sum of the data bytes is 2E hex because only one DLE text code is included in the BCC. So the BCC is D2 hex.

The BCC algorithm provides a medium level of data security. It cannot detect transposition of bytes during transmission of a packet. It also cannot detect the insertion or deletion of data values of zero within a packet.

Cyclic Redundancy Check

The cyclic redundancy check (CRC) algorithm provides a high level of data security. It can detect:

- All Single-Bit and Double-Bit Errors
- All Errors of Odd Numbers of Bits
- All Burst Errors of 16 Bits or Less
- 99.997% of 17-Bit Error Bursts
- 99.998% of 18-Bit and Larger Error Bursts

The CRC value is calculated based on the value of the data bytes and the ETX byte (using the polynomial $x^{16} + x^{15} + x^2 + x^0$). To transmit the data value of 10 hex, you must use the data code DLE DLE. However, only one of these DLE data bytes is included in the CRC value. Embedded responses are not included in the CRC value.

At the start of a message packet, the transmitter clears a 16-bit register for the CRC value. As a byte is transmitted, it is exclusive-ored (with Bit 0 to the right) to the right eight bits of the register. The register is then shifted right eight times with zeros (0s) inserted on the left. Each time a one (1) is shifted out on the right the following binary number is exclusive-ored with the 16-bit register value:

1010 0000 0000 0001

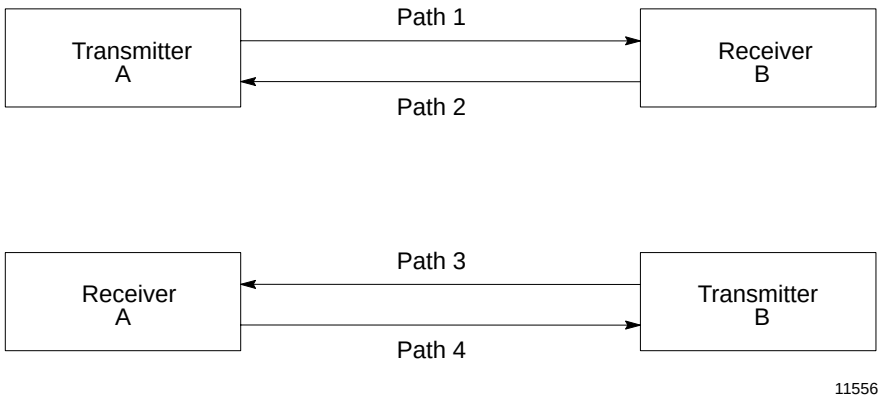
As each additional byte is transmitted, it is included into the value in the register in the same way. After the ETX value is included into the value in the register and is transmitted, the value in the register is transmitted (right bit first) as the CRC field.

The receiver also calculates the CRC value and compares it to the received CRC value to verify the accuracy of the data received.

Two-Way Simultaneous Operation

On a two-way simultaneous link, two physical circuits connect four distinct and independent software routines. Figure 6.2 shows these software routines as Transmitters (XMTR) A and B and Receivers (RCVR) A and B.

Figure 6.2
Data Paths for Two-Way Simultaneous Operation



Transmitter A and Receiver B manage the transfer of messages from Station A to Station B by sending message packets from A to B and returning response codes from B to A. At the same time, Transmitter B and Receiver A carry out the transfer of messages from Station B to

Station A by sending message packets from B to A and returning response codes from A to B.

There are also four independent data paths involved. Path 1 carries message codes from A to B; Path 2 carries response codes from B to A; Path 3 carries message codes from B to A; and Path 4 carries response codes from A to B.

To implement all these data paths with only two physical circuits, you need a software multiplexer that combines the message codes with the response codes going in the same direction. At the other end of the link, you need a software separator that separates the message codes from the response codes again. Internal software should direct the message codes to the receiver and the response codes to the transmitter. On each physical circuit, you can intermingle response codes from a receiver to a transmitter with message codes sent from a transmitter to a receiver (unless the embedded response switch is OFF). Figure 6.3 shows this implementation.

Figure 6.3
Software Implementation of Data Paths

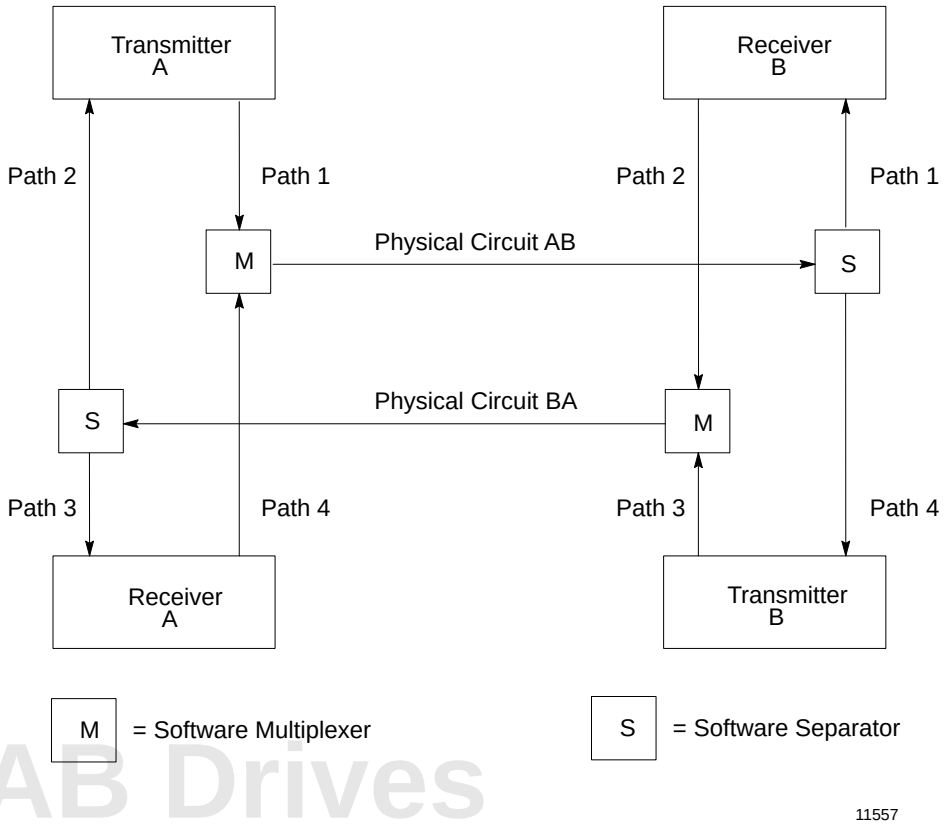
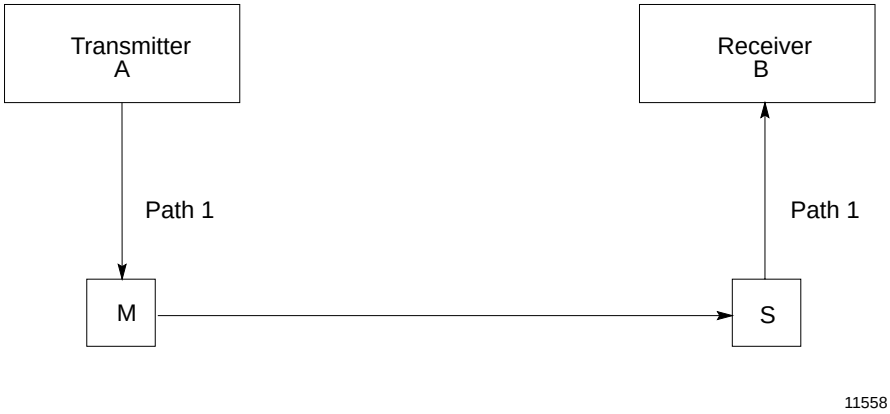


Figure 6.4 shows Path 1 with unrelated parts of Figure 6.3 removed.

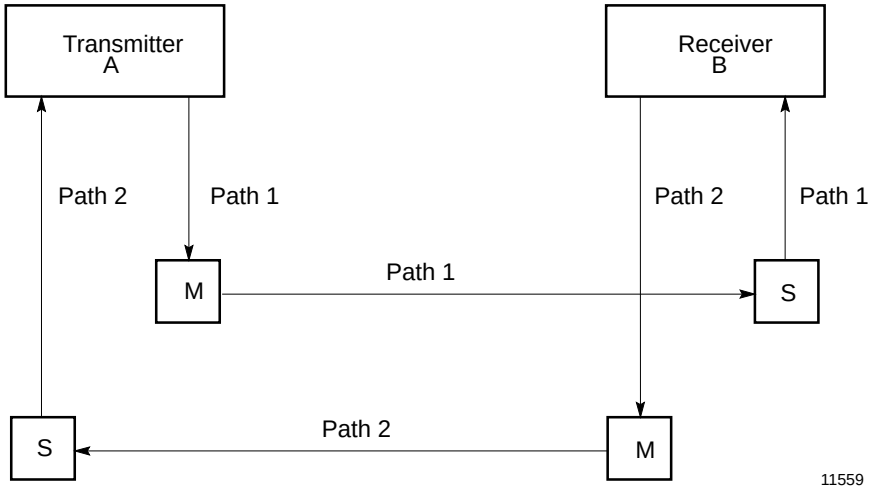
We could show Paths 2, 3, and 4 in a similar way.

Figure 6.4
Data Path 1



The full-duplex protocol is symmetrical; that is, anything that we can say about Transmitter A, Receiver B, and Paths 1 and 2 applies equally to Transmitter B, Receiver A, and Paths 3 and 4. There are actually two independent instances of the protocol operating simultaneously. For simplicity, we define the link protocol on the subsystem that carries messages from A to B, with reference to Figure 6.5.

Figure 6.5
Message Transmission from A to B



Although the protocols on each subsystem operate independently, there is a slight delay when you transmit a response code in the middle of a stream of message codes. Also, any non-transient hardware problem that affects message codes traveling over a hardware circuit affects response codes on the same circuit.

Protocol Environment Definition

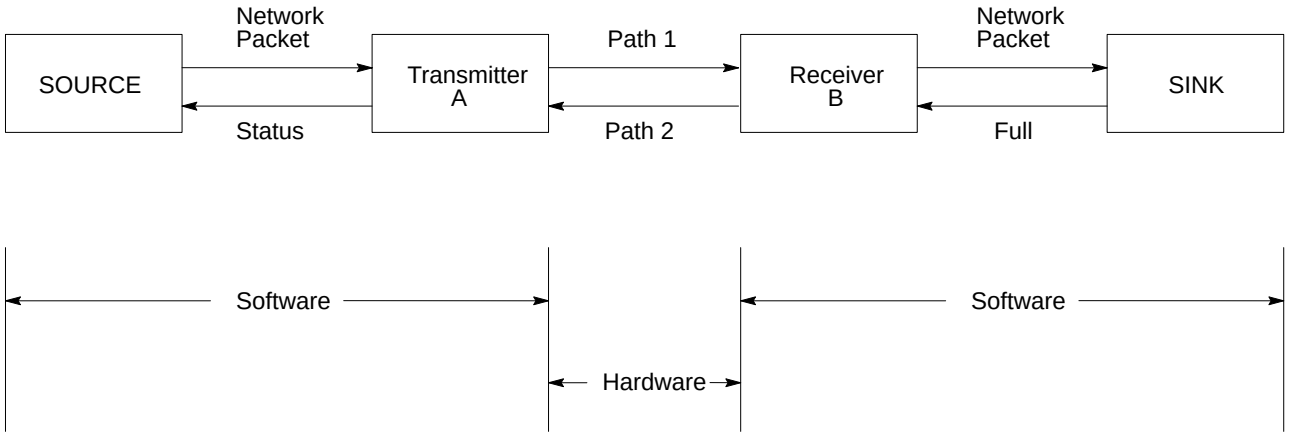
To fully define the protocol environment, you must tell the transmitter where to get the messages it sends and tell the receiver what to do with the messages it receives. You can do this in your network layer (see "Network Layer"). For purposes of discussion in this section, these functions are called the message source and the message sink respectively.

We assume that the message source supplies one network packet at a time upon request from the transmitter and that it requires notification of the success or failure of the transfer to Station B before supplying the next. When the message source is empty, the transmitter waits in an inactive state until a message is available.

Whenever the receiver has received a link packet successfully, it attempts to give the network packet portion to the message sink. The message sink may be full. The message sink must notify the receiver when it is full.

Figure 6.6 represents the protocol environment.

Figure 6.6
Protocol Environment



Message Characteristics

Full-duplex protocol places the following restrictions on the network packet that is submitted to the link layer for transfer:

- The size of a valid network packet is 6 bytes minimum and 250 bytes maximum.
- The first byte of a network packet must be the station number of the receiver station (see DST in “Network Layer”). The receiver ignores messages that do not contain the correct station number.
- As part of the duplicate message detection algorithm, the receiver checks the second, third, fifth, and sixth bytes of each network packet. At least one of these bytes of the current network packet must differ from the corresponding byte of the previous network packet in order for the receiver to accept the current network packet. Otherwise, the receiver assumes that the current packet is a retransmission of the previous packet, so it discards the current packet.

Transmitter Actions

Whenever the message source can supply a packet and the transmitter is not busy, Transmitter A sends a link packet on Path 1 (Figure 6.6). It then starts a timeout and waits for a response on Path 2. You can use the diagnostic set timeout command (see “Set Timeout”) to set this timeout period for the 1771-KG module. The default setting is 3 seconds.

When Transmitter A gets a DLE ACK, the message transfer is complete. After signaling the message source that the message has been sent successfully, Transmitter A proceeds with the next message.

If Transmitter A gets a DLE NAK, it retransmits the same message. The transmitter restarts the timeout and waits again for a response. By using the diagnostic set NAKs command (see “Set NAKs”), you can specify how many times the 1771-KG module will attempt to retransmit a given message. The default setting is three. Once the number of retransmissions exceeds this limit, the transmitter should notify the message source that the transmission has failed. The transmitter can then proceed with the next message.

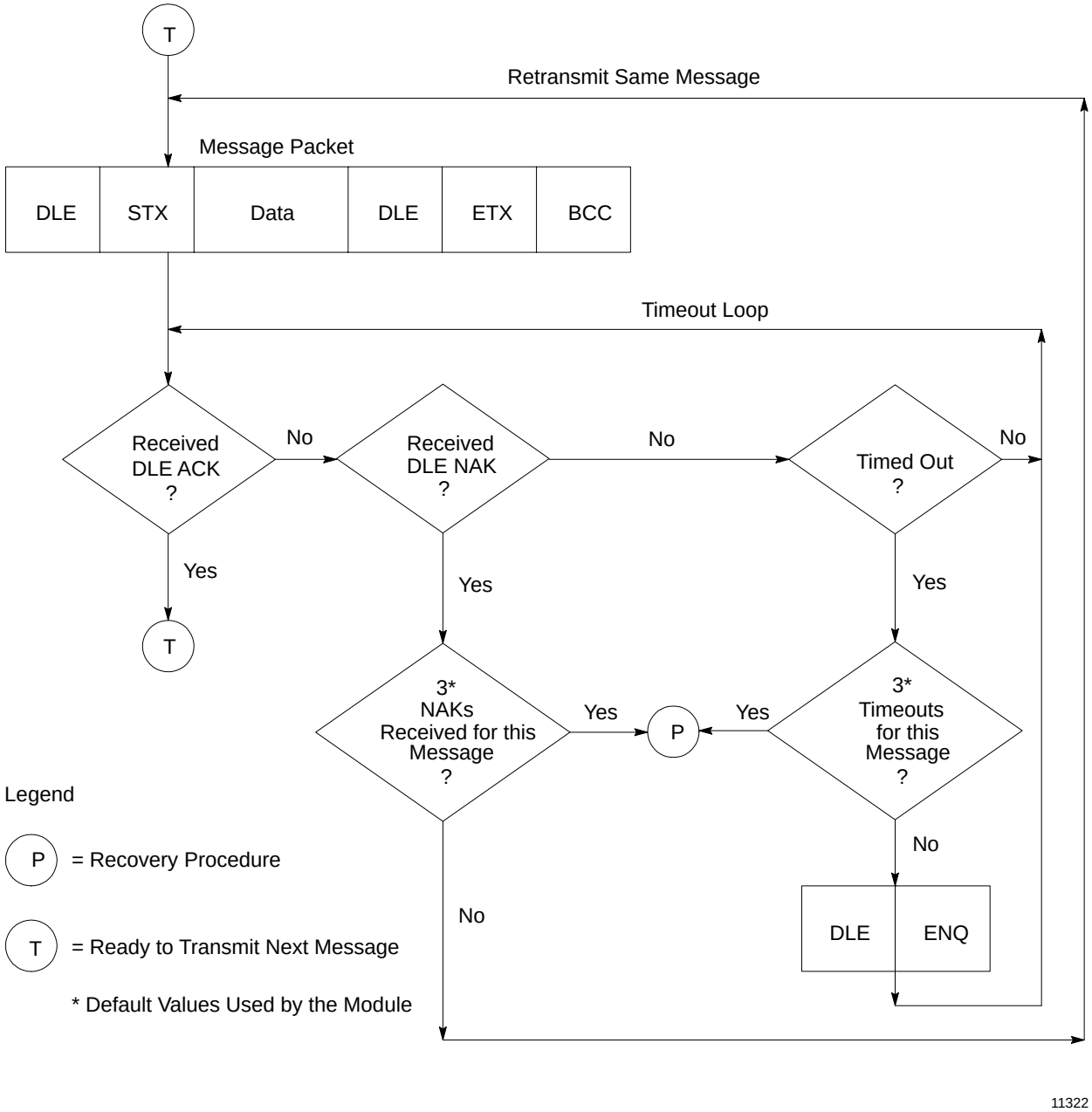
If the timeout expires before Transmitter A gets a response, it sends a DLE ENQ on Path 1 to request a retransmission of the last response sent on Path 2. Transmitter A restarts the timeout and waits for a response. By using the diagnostic set ENQs command (see "Set ENQs"), you can specify how many timeout periods the 1771-KG module will allow per message it transmits. The default setting is ten. If this ENQ limit is exceeded, the transmitter should notify the message source that the transmission has failed. The transmitter can then proceed with the next message.

DLE ACK and DLE NAK are the only response codes defined. If the receiver gets an invalid response code, it should ignore it.

Note that the transmitter must encode a text value of 10 hex as two consecutive (indivisible) bytes, each of value 10 hex. This is necessary to distinguish the text value of 10 hex from the DLE control code of 10 hex. This technique is known as DLE stuffing. The receiver must be able to reverse this process and extract the original text value of 10 hex.

Figure 6.7 is a flowchart which gives a simplified view of an example of software logic for implementing the transmitter. Table 6.A gives a detailed description of an example of software logic for implementing the transmitter in structured English procedures. In Appendix D are flowcharts which give a detailed view of an example of software logic for implementing the transmitter.

Figure 6.7
Transmitter for Full-Duplex Protocol



11322

Table 6.A
Transmitter for Full-Duplex Protocol

```
TRANSMITTER is defined as
    loop
    Message = GET-MESSAGE-TO-SEND
    Status = TRANSFER (Message)
    SIGNAL-RESULTS (Status)
    end

TRANSFER (Message) is defined as
    initialize NAK-limit and ENQ-limit
    SEND (Message)
    start timeout
    loop
    WAIT for response on Path 2 or timeout.

    if received DLE ACK then return SUCCESS
    else if received DLE NAK then
        begin
            if NAK-limit is exceeded then return FAILURE
                else
                    begin
                    count NAK retries;
                    SEND-MESSAGE (Message);
                    start timeout
                    end
        end
    else if timeout
        begin
            if ENQ-limit is exceeded then return FAILURE
            else
                begin
                count ENQ retries;
                send DLE ENQ on Path 1;
                start timeout
                end
        end
    end loop

SEND (Message) is defined as
    begin
    BCC = 0
    send DLE STX on Path 1
    for every byte in the message do
        begin
        add the byte to the BCC;
        send the corresponding data code on Path 1
        end
    send DLE ETX BCC on Path 1
    end

GET-MESSAGE-TO-SEND
    This is an operating-system-dependent  interface routine that waits and allows the rest
    of  the system  to run until  the message source  has supplied a message to be sent.

SIGNAL-RESULTS
    This is an  implementation-dependent  routine  that  tells the message source of the results
    of the attempted message transfer.

WAIT
    This is an operating-system-dependent  routine  that waits for any of several events to
    occur while allowing other parts of the system to run.
```

Receiver Actions

Since the receiver gets dirty input from the physical world, it is more complex and must be capable of responding to many adverse situations. Some of the things that can conceivably happen are listed here:

- The message sink can be full, leaving the receiver with nowhere to put a message.
- A message can contain a parity error.
- The BCC can be invalid.
- The DLE STX or DLE ETX BCC may be missing.
- The message can be too long or too short.
- A spurious control or text code can occur outside a message.
- A spurious control code can occur inside a message.
- Any combination of the above can occur.
- The DLE ACK response can be lost, causing the transmitter to send a duplicate copy of a message that has already passed to the message sink.

Receiver B must keep a record of the last response code (DLE ACK or DLE NAK) sent on Path 2 (Figure 6.5). If it receives a DLE ENQ, the receiver sends this recorded response code again.

The receiver also keeps a record of the first six link-level data bytes of the last message received. If the SRC, CMD, and both TSN bytes of a new message are identical to the corresponding bytes of this record, the receiver responds with a DLE ACK but ignores the new message. This process is known as duplicate message detection and is part of the link-level data security. It guards against re-execution of a message that has already been received successfully but for which the response code (DLE ACK) has been lost.

Until it receives a DLE STX or a DLE ENQ, the receiver ignores all input from Path 1 except to set the last response variable to NAK. With the last response variable set to NAK, the receiver responds with DLE NAK to a DLE ENQ input. Otherwise, the receiver responds to a DLE ENQ input by sending its last response on Path 2 and continues waiting for input. If the receiver gets a DLE STX, it resets its BCC accumulator and data buffer to zero and starts storing the link-level data in the data buffer so that it can later pass the link-level data to the network layer.

While the receiver stores all link-level data codes in the data buffer, it adds the link-level data code values to the BCC. If the data buffer overflows, the receiver continues summing the BCC, but it discards the data.

The receiver also sets an error flag to indicate the occurrence of a parity, buffer overrun, message framing, or modem handshaking error. If the receiver receives any control code other than DLE ETX during this time, it aborts the message and sends a DLE NAK on Path 2. When the receiver gets a DLE ETX BCC, it checks the error flag, the BCC, the message size and the destination station number. If any of the tests fail, the receiver sends a DLE NAK on Path 2.

If the current message packet passes the above tests, the receiver next begins the duplicate message detection process. In this process, the receiver compares the SRC, CMD, and both TNS bytes of the current message with the corresponding bytes of the previous message received. If these bytes are the same, the receiver discards the current message and sends a DLE ACK.

If the current message differs from the previous one, the receiver next tests the state of the message sink. If the message sink is full, the receiver sends a DLE NAK. Otherwise, the receiver:

- Forwards the current link-level data to the message sink.
- Keeps a copy of the first six bytes of the current link-level data for purposes of duplicate messages detection.
- Sends a DLE ACK.

Figure 6.8 is a flowchart which gives a simplified view of an example of software logic for implementing the receiver. Table 6.B gives a detailed description of an example of software logic for implementing the receiver in structured English procedures.

In Appendix D are flowcharts which give a detailed view of an example of software logic for implementing the transmitter.

Figure 6.8
Receiver for Full-Duplex Protocol

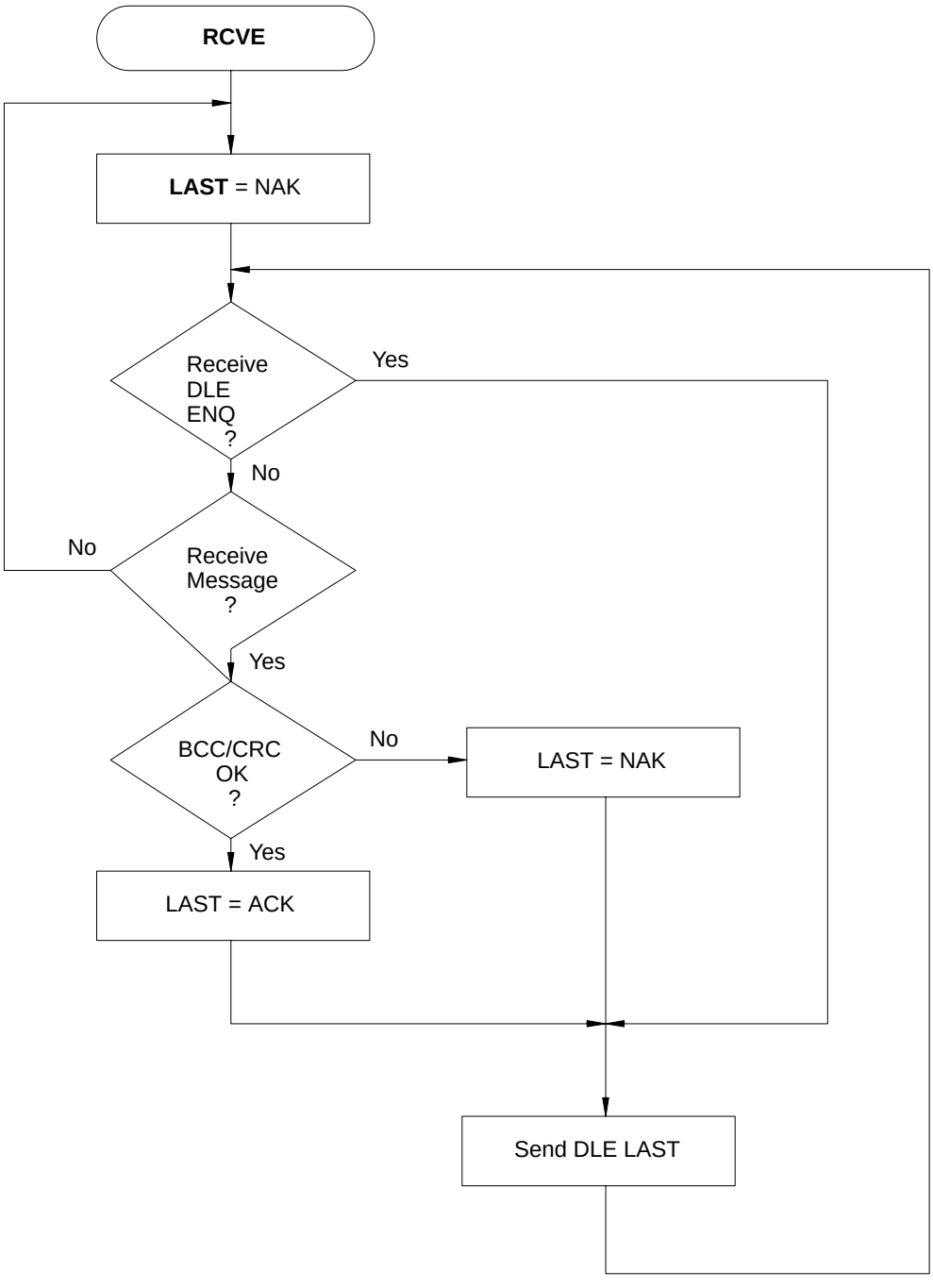


Table 6.B
Receiver for Full-Duplex Protocol

```
RECEIVER is defined as
    variables
        LAST-HEADER is four bytes copied out of the last good message
        RESPONSE is the value of the last ACK or NAK sent
        BCC is an eight-bit block check accumulator

    LAST-HEADER = Invalid
    LAST RESPONSE = NAK

    loop
        reset parity error flag
        GET-CODE
        if DLE STX then
            begin
                BCC =  0
                GET-CODE
                while it is a data code
                    begin
                    if buffer is not overflowed put data in buffer
                    GET-CODE
                    end
                if the control code is not a DLE ETX then send DLE NAK
                else if error flag is set then send DLE NAK
                else if BCC is not zero then send DLE NAK
                else if message is too small then send DLE NAK
                else if message is too large then send DLE NAK
                else if header is same as last message send a DLE ACK
                else if message sink is full send DLE NAK
                else
                    begin
                        send message to message sink
                        send a DLE ACK
                        save last header
                    end
            end
        else if DLE ENQ then send LAST-RESPONSE
        else LAST-RESPONSE = NAK
    end
```

```
GET-CODE is defined as
    loop
        variable
        GET-CHAR
        if char is not a DLE
            begin
                add char to BCC
                return the char and data flag
            end
        else
            begin
                GET-CHAR
                if char is a DLE
                    begin
                        add char to BCC
                        return a DLE and a data flag
                    end
                else if char is an ACK or NAK send it to the transmitter
                else if char is an ETX
                    begin
                        GET-CHAR
                        add char to BCC
                        return ETX with a control flag
                    end
                else return character with a control flag
            end
        end
    end
GET-CHAR is defined as
    an implementation-dependent function that returns one byte of data from the link interface hardware.
```

Full-Duplex Protocol Diagrams

The following figures show some events that can occur on the various interfaces. Control characters are shown in bold type. Link-level data is represented by xxxx. Line noise is represented by ???. BCC is shown at the end of each message packet. However, you could alternately use CRC. Time is represented as increasing from the top of the figure to the bottom. Figure 6.9 shows normal message transfer.

Figure 6.9
Normal Message Transfer

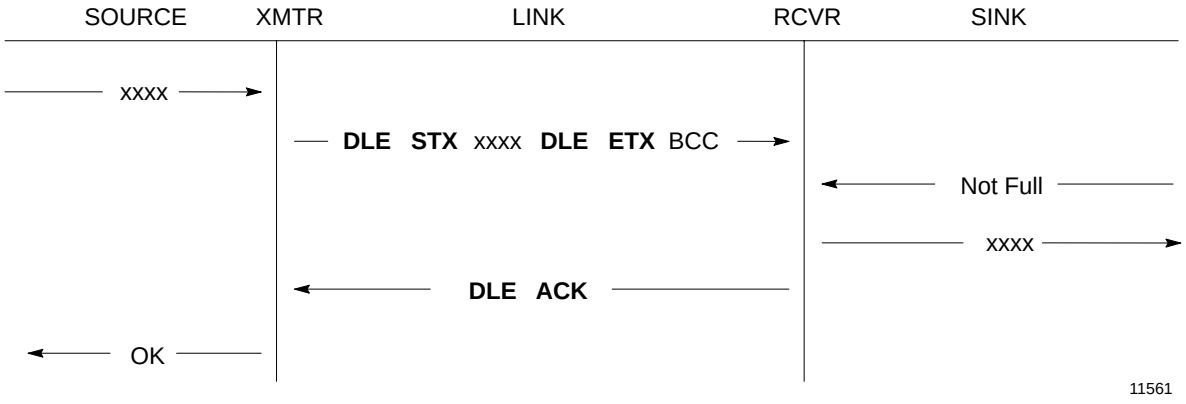


Figure 6.10 shows a DLE NAK response to the initial message transmission. After the message is retransmitted, a DLE ACK response is given.

Figure 6.10
Message Transfer with NAK

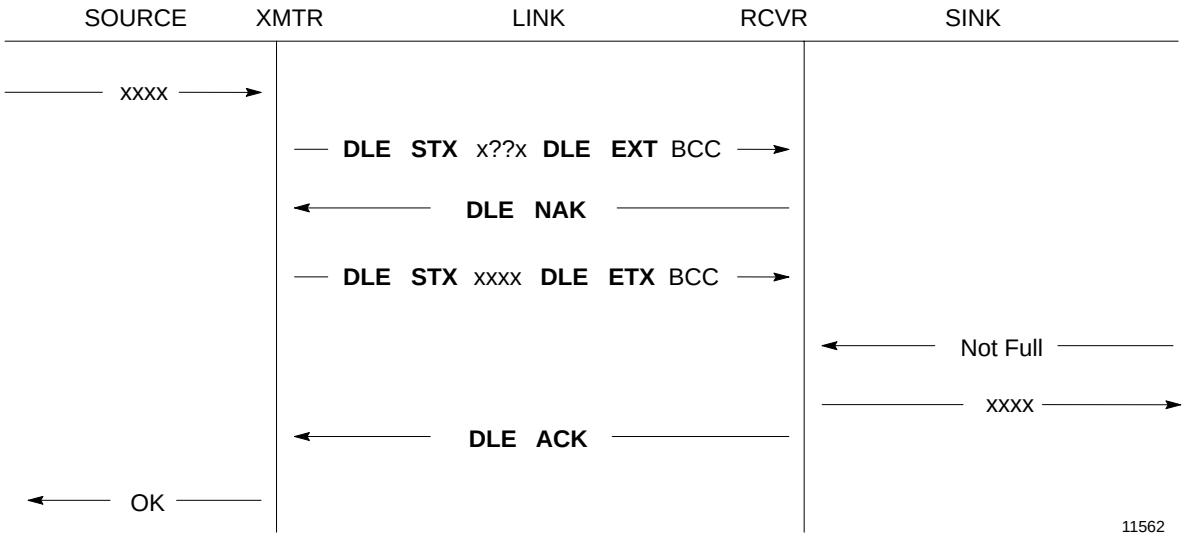
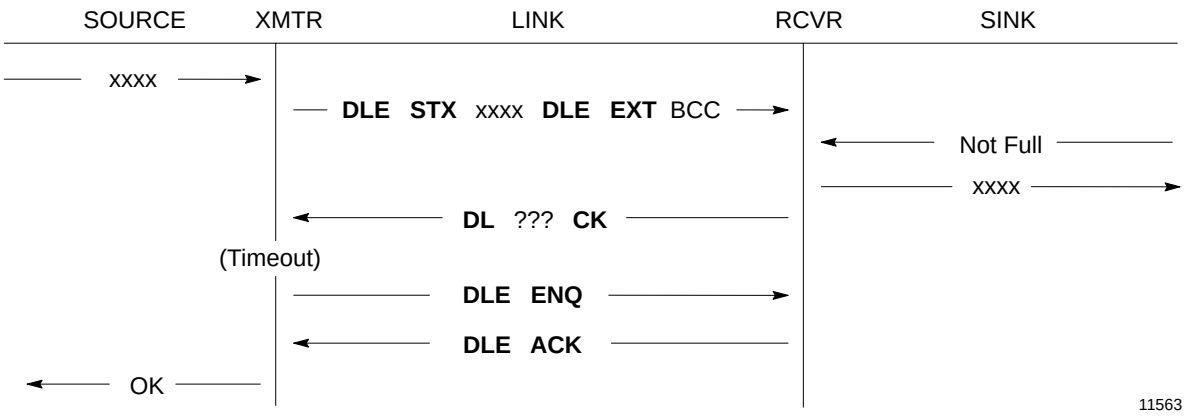


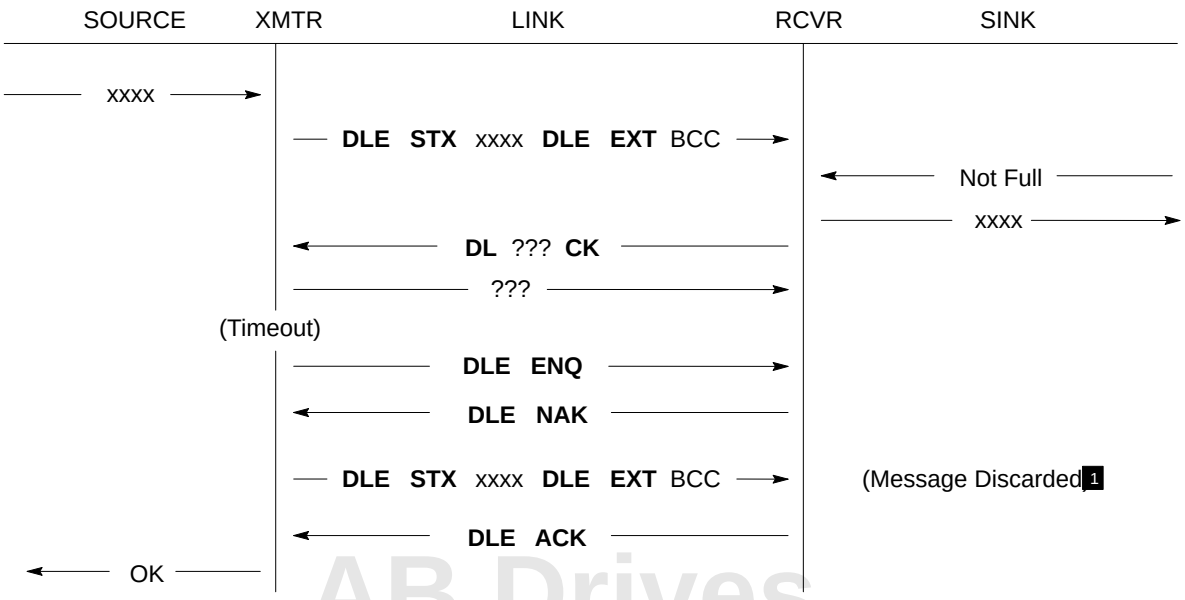
Figure 6.11 shows the transmitting station sending a DLE ENQ sequence after a timeout because it did not receive the initial DLE ACK response.

Figure 6.11
Message Transfer with Timeout and ENQ



In Figure 6.12, retransmission occurs when noise hits both sides of the line. This type of noise destroys the DLE ACK while also producing invalid characters at the receiver. The result is that the receiver changes its last response to NAK and the transmitter retransmits the original message packet.

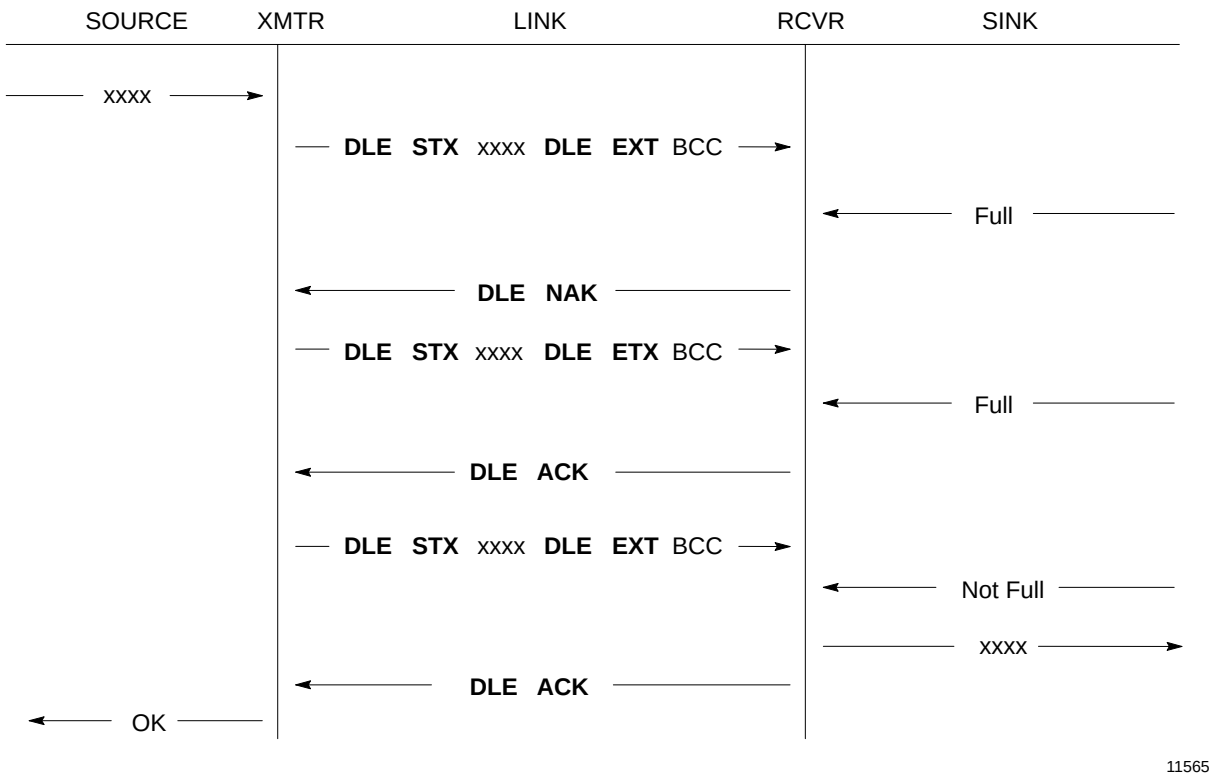
Figure 6.12
Message Transfer with Retransmission



1 Note that this is detected as a duplicate message.

Figure 6.13 shows a DLE NAK response to the initial message transmission because the message sink is full. After the message sink is no longer full, a retransmission of the message causes a DLE ACK response.

Figure 6.13
Message Transfer with Message Sink Full



Examples

If you were to connect a line monitor to the wires between Station A and B, and only the A to B subsystem were active, you could observe the following:

Normal Message

```
Path 1:  DLE STX xxx DLE ETX BCC          DLE STX xxxx DLE ETX BCC
Path 2:                          DLE ACK                          DLE ACK
```

Message with Parity or BCC Error and Recovery

```
Path 1:  DLE STX xx???xx DLE ETX BCC          DLE STX xxxx DLE ETX BCC
Path 2:                          DLE NAK                          DLE ACK
```

Message with ETX Destroyed

```
Path 1:  DLE STX xxxxx???? [timeout] DLE ENQ          DLE STX xxxx DLE ETX BCC
Path 2:                                   DLE NAK                          DLE ACK
```

Good Message but ACK Destroyed

```
Path 1:  DLE STX xxx DLE ETX BCC          [timeout] DLE ENQ          DLE STX xxx etc.
Path 2:                          DL ??? CK                DLE ACK
```

Messages Being Sent in Both Directions

```
Path 1:  DLE STX xxx DLE ETX BCC          DLE STX xxxx DLE ETX BCC          DLE STX
Path 2:                          DLE ACK                          DLE ACK
Path 3:            DLE STX xxx          xxxx DLE ETX BCC                    DLE STX
Path 4:                                                          DLE ACK
```

Combined

```
Circuit AB:  DLE STX xxx DLE ETX BCC          DLE STX xxxx DLE ETX BCC  DLE ACK DLE STX
Circuit BA:            DLE STX xxx  DLE ACK xxxx DLE ETX BCC            DLE ACK DLE STX
```

Embedded Response

ACK on AB delayed slightly because ETX BCC are indivisible.

Embedded Response Option

To simplify the design of the receiver in some cases, you can disable transmission of embedded responses by turning OFF the embedded response switch. If you turn this switch OFF, the 1771-KG module's multiplexer cannot embed response codes while sending a message. Instead, it delays sending response codes until after it sends the next DLE ETX BCC sequence.

Half-Duplex Protocol

Half-duplex protocol serves as an alternate to full-duplex protocol. You can select half-duplex protocol by turning ON the protocol switch. Half-duplex protocol is similar to full-duplex in most respects. Two major differences are:

- Half-duplex protocol provides for polling of slave stations.
- Half-duplex protocol does not allow embedded responses.

Half-duplex protocol is for one master and one or more slaves. You must use MODEMS for this type of link (unless there is only one slave). The 1771-KG module has slave mode capability only, you must provide the master function through a computer.

Half-duplex protocol provides a lower effective utilization of resources than full-duplex, but it is easier to implement. You should use half-duplex protocol if:

- You are using multi-drop baseband MODEMS to connect multiple slave stations to a single master computer.
- You are using MODEMS that have only half-duplex capability.
- You are willing to sacrifice data throughput in exchange for ease of implementation.

Multi-Drop Link

One environment for half-duplex protocol is a multi-drop link with all stations interfaced through half-duplex MODEMS. The actual nature of the link does not matter much, as long as the MODEMS support request-to-send, clear-to-send, and data-carrier-detect signals. If you use dial-up MODEMS, they must also support data-set-ready and data-terminal-ready; otherwise, you should jumper data-set-ready to data-terminal-ready at the 1771-KG module.

You may have from two to 256 stations simultaneously connected to a single multi-drop link. Each station must have a receiver connected to the circuit and transmitter that can be enabled or disabled by request-to-send.

You must program a computer to serve as a master that controls which station has access to the link. All other stations are slaves and must wait for permission from the master before transmitting. Each slave station has a unique station number from ten to 77 and 110 to 376 octal. The number 377 is a broadcast address. When the master sends a message addressed to 377, all slaves receive it.

The master can send and receive messages to and from each station on the multi-drop link. If the master is programmed to relay messages, then

slave stations on the multi-drop link can engage in peer-to-peer communication.

Your multi-drop link may be either a two-circuit system (master sends and slaves receive on one circuit, slaves send and master receives on the other) or a one-circuit system (master and slaves send and receive on the same circuit).

You may use a half-duplex, dial-up modem to connect the 1771-KG module to the multi-drop link. The modem must signal data-carrier-detect at least once every 8 seconds. If it does not, the module will hang up.

You cannot use multiple masters unless one master is limited to acting as a backup to the other and does not communicate until the primary is shut down.

Transmission Codes

Half-duplex protocol is a character-oriented protocol that uses the following ASCII control characters extended to eight bits by adding a zero for Bit 7. See ANSI X3.4, CCITT V.3, or ISO 646 for the standard definition of these characters.

Control Character	Hexadecimal Code
SOH (Start of Header)	01
STX (Start of Text)	02
ETX (End of Text)	03
EOT (End of Transmission)	04
ENQ (Enquiry)	05
ACK (Acknowledge)	06
DLE (Data Link Escape)	10
NAK (Negative Acknowledge)	15

Additionally, a block check character (BCC) or two-byte cyclic redundancy check (CRC) field is used at the end of each transmission packet for error checking. These bytes can be any value from 00 to FF hex.

The term code means (in the following paragraphs) an indivisible sequence of one or more bytes having a specific meaning to the protocol. Indivisible means that the component bytes of a code must be sent one after another with no other bytes inserted between them. It does not refer

to the timing of the bytes. (This definition has less significance than for full-duplex protocol since there is no multiplexing of transmission codes in half-duplex protocol).

Half-duplex protocol uses the following codes:

- Control Codes:
  - DLE SOH
  - DLE STX
  - DLE ETX BCC/CRC
  - DLE ACK
  - DLE NAK
  - DLE ENQ
  - DLE EOT
- Link-Layer Data Codes:
  - Data (single bytes having values 00-0F and 11-FF hex)
  - DLE DLE (to represent the value 10 hex)
- Link-Layer Address Code:
  - STN (slave station number)

We can group these codes into two classes according to their use: codes issued from a station transmitting a message (or poll) and response codes issued from a station receiving a message (or poll):

- Codes from Station Transmitting a Message (or Poll):
  - DLE SOH — Indicates the start of a message packet.
  - STN — Station number of a slave station.
  - DLE STX — Separates the data link protocol information from the network packet.
  - Link-Layer Data (00-0F and 11-FF hex) — Encodes the bytes of the network packet.
  - DLE DLE — Encodes the value 10 hex in the network packet. This is necessary to distinguish a text code of 10 hex from a DLE control code of 10 hex.
  - DLE ETX BCC/CRC — Terminates a message or polling packet.
  - DLE ENQ — Indicates the start of a polling packet.

- Response Codes from Station Receiving a Message (or Poll):
  - DLE ACK — Signals that the receiver has successfully received the last message sent.
  - DLE NAK — Serves as a global link reset command. It causes all slaves to cancel all messages they have ready to transmit to the master. The 1771-KG module will respond to this by writing Error Code 84 into its error word in the PC data table.
  - DLE EOT — Is the response that a slave sends to a poll from the master when the slave has no messages to send.

Link-Layer Packets

Half-duplex protocol uses three types of transmissions:

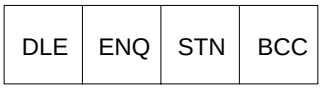
- Polling Packet
- Master Message Packet
- Slave Message Packet

The master station transmits both polling packets and master message packets, while slave stations transmit slave message packets.

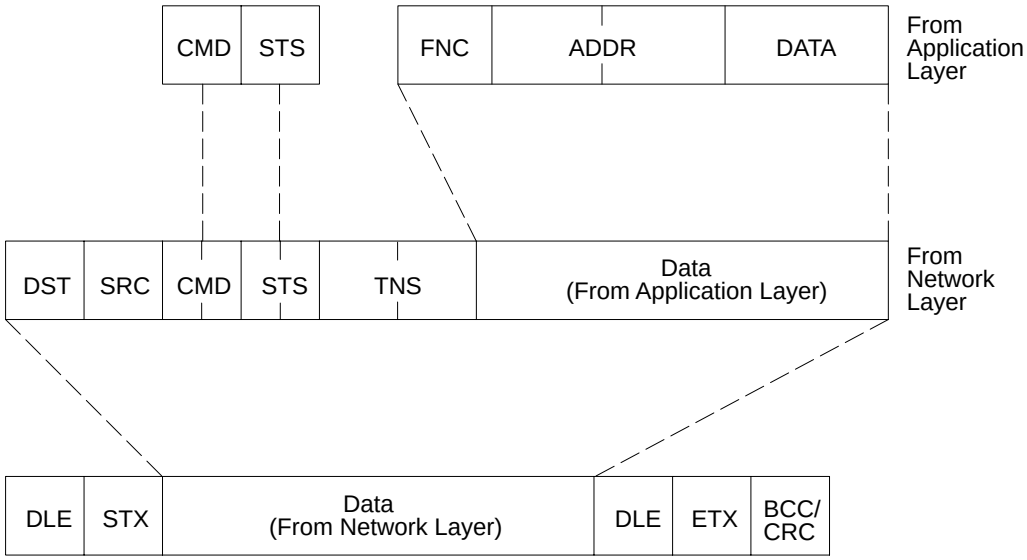
Figure 6.14 illustrates the formats of these packets. Note that the slave message packet has the same format as the full-duplex message packet (see “Link-Layer Message Packets”). The master message packet is the same as the slave message packet except that it is prefixed with DLE SOH and an address code to specify a slave station number.

At the end of each polling packet is a BCC byte. At the end of each message packet is either a one-byte BCC or two-byte CRC field. With a Series A module, you must use BCC. With a Series B module, you can select BCC or CRC through switch settings (Chapter 4).

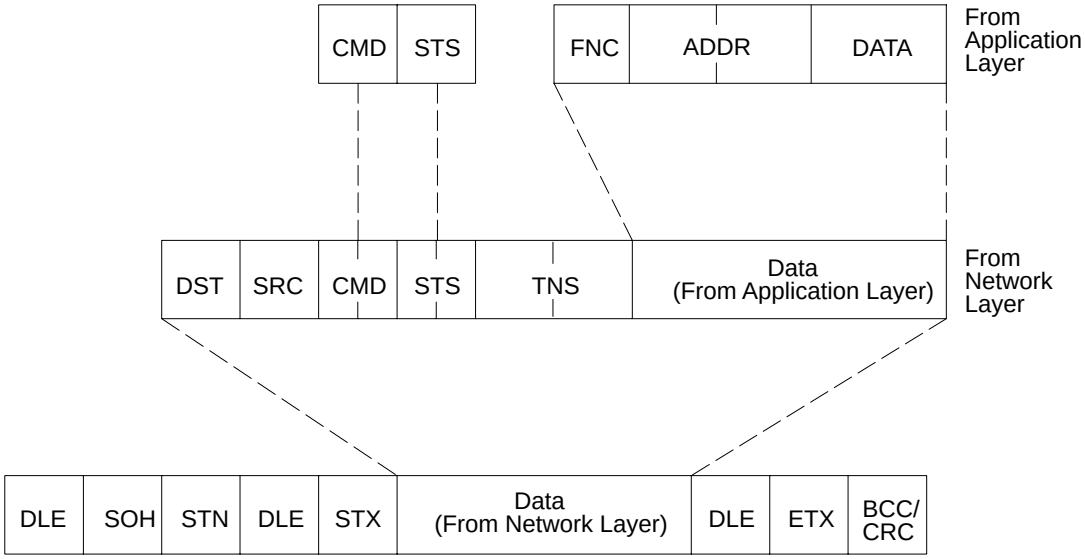
Figure 6.14
Formats for Half-Duplex Protocol



a) Polling Packet



b) Slave Message Link Packet



c) Master Message Link Packet

11327

Block Check

The block check character (BCC) is a means of checking the accuracy of each packet transmission. It is the 2's complement of the eight-bit sum (modulo-256 arithmetic sum) of the slave station number (STN) and all the data bytes in the packet. For polling packets, the BCC is simply the 2's complement of STN. The BCC does not include any other message packet codes or response codes.

For example, if the master station wanted to send the data codes 8, 9, 6, 0, 2, 4, and 3 to Slave Station 20 hex (40 octal), the master message codes would be (in hex):

10	01	20	10	02	08	09	06	00	02	04	03	10	03	C0
DLE	SOH	STN	DLE	STX	Data							DLE	ETX	BCC

The sum of the STN and data bytes in this message packet is 40 hex. The BCC is the 2's complement of this sum, or C0 hex. This is shown in the following binary calculation:

0100 0000	40 hex
1011 1111	1's complement
+1	
1100 0000	2's complement (C0 hex)

To transmit the STN or data value 10 hex, you must use the data code DLE DLE. However, only one of these DLE text characters is included in the BCC sum. For example, to transmit the values 8, 9, 6, 0, 10, 4, and 3 hex, a slave station would use the following message codes:

Represents Single Text Value of 10

10	02	08	09	06	00	10	10	04	03	10	03	D2
DLE	STX	Data								DLE	ETX	BCC

In this case, the sum of the data bytes is 2E hex because only one DLE text code is included in the BCC. So the BCC is D2 hex.

The BCC algorithm provides a medium level of data security. It cannot detect transposition of bytes during transmission of a packet. It also cannot detect the insertion or deletion of data values of zero within a packet.

Cyclic Redundancy Check

The cyclic redundancy check (CRC) algorithm provides a high level of data security. It can detect:

- All Single-Bit and Double-Bit Errors
- All Errors of Odd Numbers of Bits
- All Burst Errors of 16 Bits or Less
- 99.997% of 17-Bit Error Bursts
- 99.998% of 18-Bit and Large Error Bursts

The CRC value of a slave message packet is calculated based on the value of the data bytes and the ETX byte (using the polynomial $x^{16} + x^{15} + x^2 + x^0$). The CRC value of a master message packet is calculated based on the value of the STN byte, the STX byte, the data bytes, and the ETX byte. To transmit the data value of 10 hex, you must use the data code DLE DLE. However, only one of these DLE data bytes is included in the CRC value.

At the start of a message packet, the transmitter clears a 16-bit register for the CRC value. As a byte is transmitted, it is exclusive-ored (with Bit 0 to the right) to the right eight bits of the register. The register is then shifted right eight times with zeros (0s) inserted on the left. Each time a one (1) is shifted out on the right, the following binary number is exclusive-ored with the 16-bit register value:

1010 0000 0000 0001

As each additional byte is transmitted, it is included into the value in the register in the same way. After the ETX value is included into the value in the register and is transmitted, the value in the register is transmitted (right bit first) as the CRC field.

The receiver also calculates the CRC value and compares it to the received CRC value to verify the accuracy of the data received.

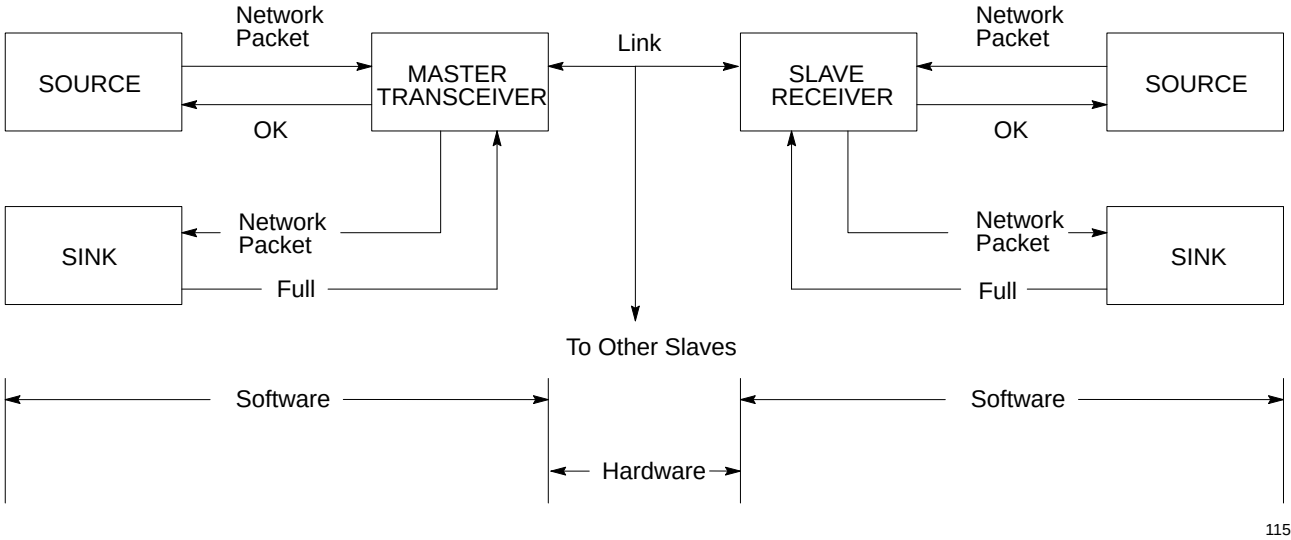
Protocol Environment Definition

Each station on the multi-drop link must contain a software routine, known as a transceiver, that can both transmit and receive message packets. The 1771-KG module already contains a slave transceiver routine, so it will function as a slave station if you turn ON the protocol switch. To establish a master station, you have to program a transceiver routine at a computer. In addition to transmitting and receiving message packets, the master transceiver must also be able to transmit polling packets.

Note that you can program separate transmitter and receiver routines instead of a single transceiver. For purposes of the discussion here, however, we assume that the transceiver is a single software routine.

Figure 6.15 illustrates the operation of master and slave transceivers. To fully define the protocol environment, you must tell the master transceiver where to get the messages it sends and how to dispose of messages it receives. These are implementation-dependent functions that we call the message source and the message sink respectively.

Figure 6.15
Slave Transceiver



We assume that the message source supplies one network packet at a time upon request from the transceiver and that the source requires notification of the success or failure of transfer before supplying the next. Whenever the transceiver has received a link packet successfully, it attempts to give the network packet portion to the message sink. The message sink may be full. The message sink must notify the transceiver when it is full.

Message Characteristics

Half-duplex protocol places the following restrictions on the network packet that is submitted to the link layer for transfer:

- The size of a valid network packet is six bytes minimum and 250 bytes maximum.
- The first byte of a network packet must be the station number of the receiver station (see DST in “Network Layer”). The receiver ignores messages that do not contain the correct station number.
- As part of the duplicate message detection algorithm, the transceiver checks the second, third, fifth, and sixth bytes of each network packet. At least one of these bytes of the current network packet must differ from the corresponding byte of the previous network packet in order for the transceiver to act upon the current network packet. Otherwise, the transceiver assumes that the current network packet is a retransmission of the previous network packet, so it discards the current network packet.

Master Polling Responsibilities

You may vary the master polling algorithm, depending on the how much activity you expect on your network.

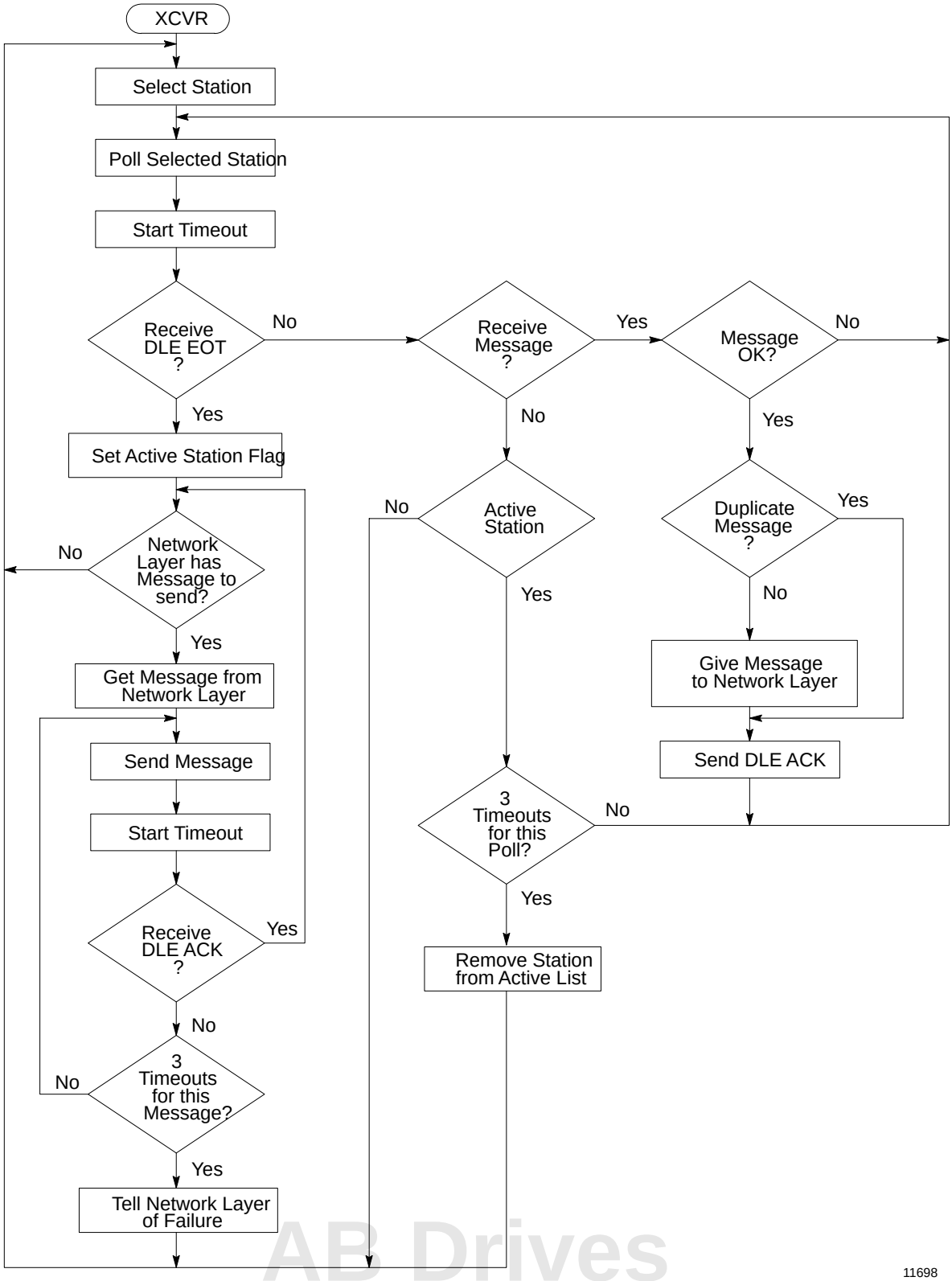
The master should poll each slave repeatedly until the slave has transmitted all of its messages. The master should then send any messages it has for that slave. Then the master can poll the next slave in the same way.

If a slave station fails to respond to a poll, the master should remove that slave from the list of active slaves. To save time, the master should poll only the active slaves on a regular basis. The master should poll the inactive slaves occasionally to see whether they will respond.

It is best not to allow the master station’s transceiver to relay messages directly from one slave station to another. Instead, the transceiver should funnel all received messages to the message sink (network layer). The network layer can then analyze the messages and retransmit any that are addressed to a slave station.

Figure 6.16 is a flowchart which gives a simplified view of an example of software logic for implementing half-duplex protocol from the master station’s point of view.

Figure 6.16
Implementation of Half-Duplex Protocol



Transceiver Actions

Since the transceiver receives dirty input from the physical world, it must be capable of responding to many adverse situations. Some of the things that can conceivably happen are listed here:

- The message sink can be full, leaving the transceiver with nowhere to put a message.
- A message can contain a parity error.
- The BCC can be invalid.
- The DLE SOH, DLE STX, or DLE ETX BCC may be missing.
- The message can be too long or too short.
- A spurious control or data code can occur outside a message.
- A spurious control code can occur inside a message.
- Any combination of the above can occur.
- The DLE ACK response can be lost, causing the transceiver to send a duplicate copy of a message that has already been passed to the message sink.

Each slave station is in a passive mode until it receives a DLE SOH or DLE SOH code. While in a passive mode a slave ignores any transmission code that is not DLE ENQ or DLE SOH.

When a slave receives a DLE SOH, it resets its BCC accumulator and message receiving buffer. The next code it receives must be its specific station number or the global Station Number 377 (octal). If the packet does not contain the appropriate station number, the slave ignores it and waits for the start of a new transmission.

If a slave receives a message packet with the appropriate station number, it adds the value of that station number to its accumulated BCC. If the next characters after the station number are DLE STX, then the slave transceiver starts storing the incoming link-layer data in a buffer. The transceiver stores all data codes in the buffer and adds these code values to the accumulated BCC. Even if the storage buffer overflows, the transceiver continues summing the BCC, while discarding the data.

The slave also sets an error flag to indicate the occurrence of a parity, buffer overrun, message framing, or MODEM handshaking error. When the slave gets a DLE ETX BCC, it checks this error flag, the BCC, and the message size. If any of these tests fail, the slave ignores the message.

If the current message packet passes the above tests, the slave next begins the duplicate message detection process. In this process, the slave compares the SRC, CMD, and both TNS bytes with the corresponding bytes of the previous message received. If these bytes are the same, the slave discards the current message and sends a DLE ACK.

If the current message differs from the previous one, the slave next tests the state of the message sink. If the message sink is full, the transceiver discards the current message and does not respond. Otherwise, the transceiver:

- Forwards the current link-level data to the message sink.
- Keeps a copy of the first six bytes of the current link-level data for purposes of duplicate message detection.
- Sends a DLE ACK.

While waiting to receive a message, a slave station could receive a polling packet that begins with a DLE ENQ sequence. The slave will ignore the poll if the polling packet does not contain the slave's station number or if the BCC in the polling block is incorrect. If the poll is valid, then one of three conditions can exist:

- The slave is still holding a message that it had transmitted previously but that had not been acknowledged by the master station. There is a limit on the number of times the slave will attempt to transmit a message. If this limit has been exceeded, the slave responds to this by writing an error code into its error word in the PC data table and then tries to transmit the next message from the message source. If the NAK limit is not exceeded, the slave tries to retransmit the current message.
- If the slave does not currently have a message to send, it tries to get one from the message source. If a message is available, the transceiver initializes its retry counter and transmits the message in response to the poll.
- If no message is available, the transceiver responds to a poll by transmitting a DLE EOT.

To transmit a message, the slave transceiver uses the same message block format as the full-duplex format (see "Link-Layer Message Packets"). After sending a message, the transceiver keeps a copy of that message until it receives a DLE ACK from the master station or until its retry limit is exceeded.

When the slave transceiver receives a DLE ACK, it discards the current message. The next time the slave is polled, it will send the next message available from the message source. If no message is available in the message source, the slave responds to a poll with DLE EOT.

When the slave transceiver receives a DLE NAK, it takes messages from the source until the source is empty. It discards each message while sending an error code back to the source. The master can use this to clear the message source buffer of each slave after the master has been down.

Half-Duplex Protocol Diagrams

The following figures show the events that occur on various interfaces. Control characters are shown in bold type. Link-level data is represented by xxxx. Line noise is represented by ???. Each message packet is shown ending in BCC, although you can alternately use CRC. Time is represented as increasing from the top of the figure to the bottom. Figure 6.17 shows normal message transfer from the master to a slave.

Figure 6.17
Normal Message Transfer

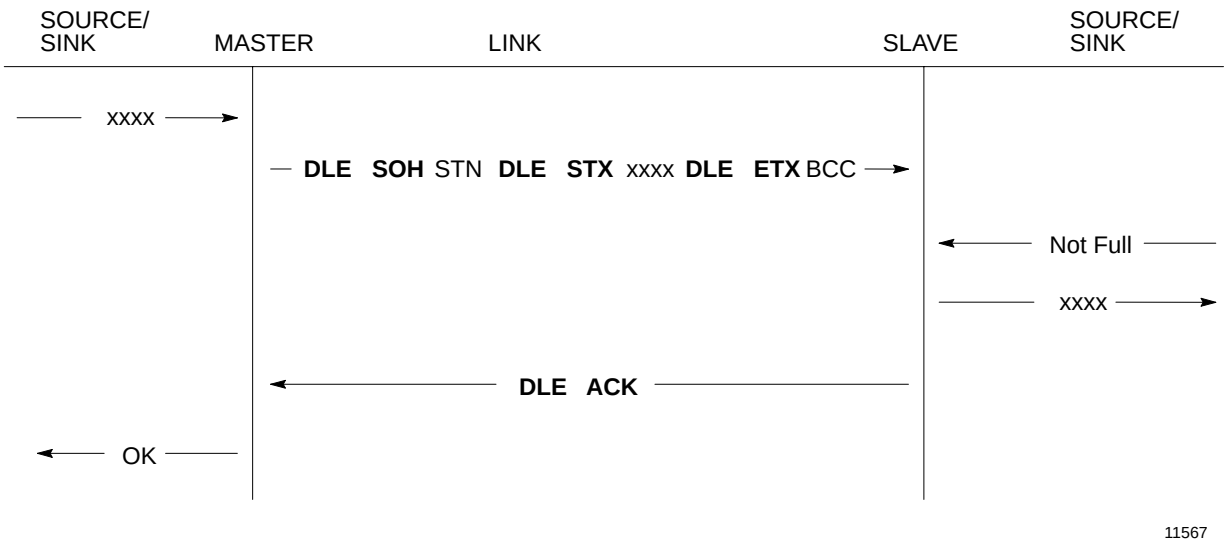


Figure 6.18 shows a message transfer in which the BCC was invalid. After a timeout, the message is retransmitted. After the retransmission the response is DLE ACK.

Figure 6.18
Message Transfer with Invalid BCC

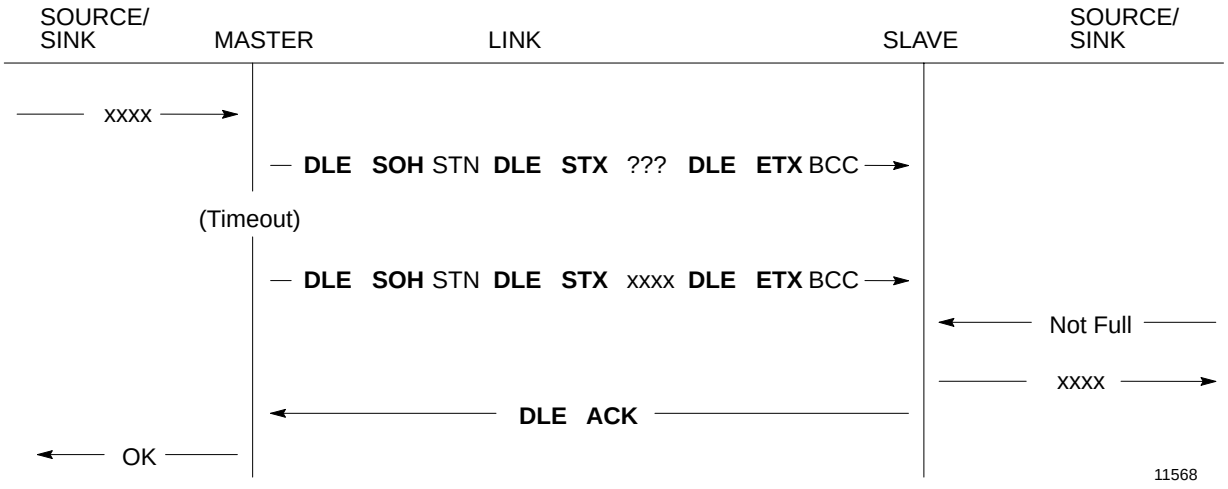


Figure 6.19 shows a message transfer in which the acknowledgment was destroyed by noise. After a timeout, the message is retransmitted and the DLE ACK response is detected.

Figure 6.19
Message Transfer with ACK Destroyed

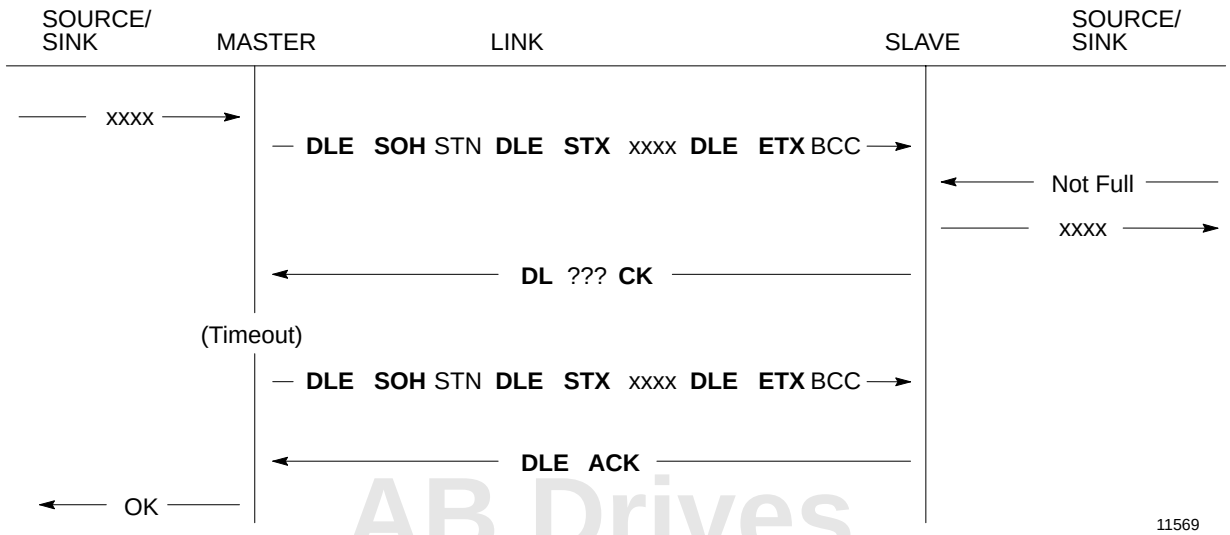


Figure 6.20 shows a slave being polled, and responding with DLE EOT because it has no messages to transfer.

Figure 6.20
Poll with No Message Available

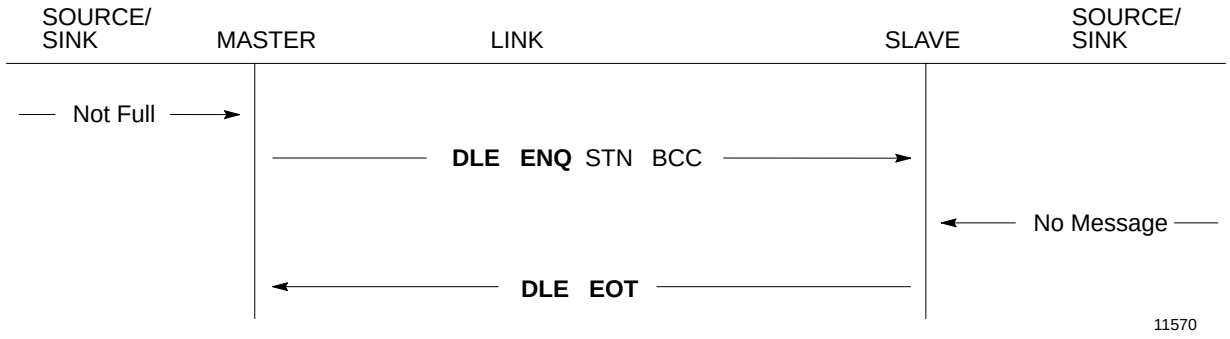


Figure 6.21 shows a slave being polled, and answering with a message. Because a block check error is found, the master does not acknowledge; instead, it sends the poll to the slave again. Since the slave did not receive an acknowledgment to its first message transmission, it retransmits the same message in answer to the second poll. The master receives the second transmission of the message with no error and responds with DLE ACK.

Figure 6.21
Poll with Message Returned

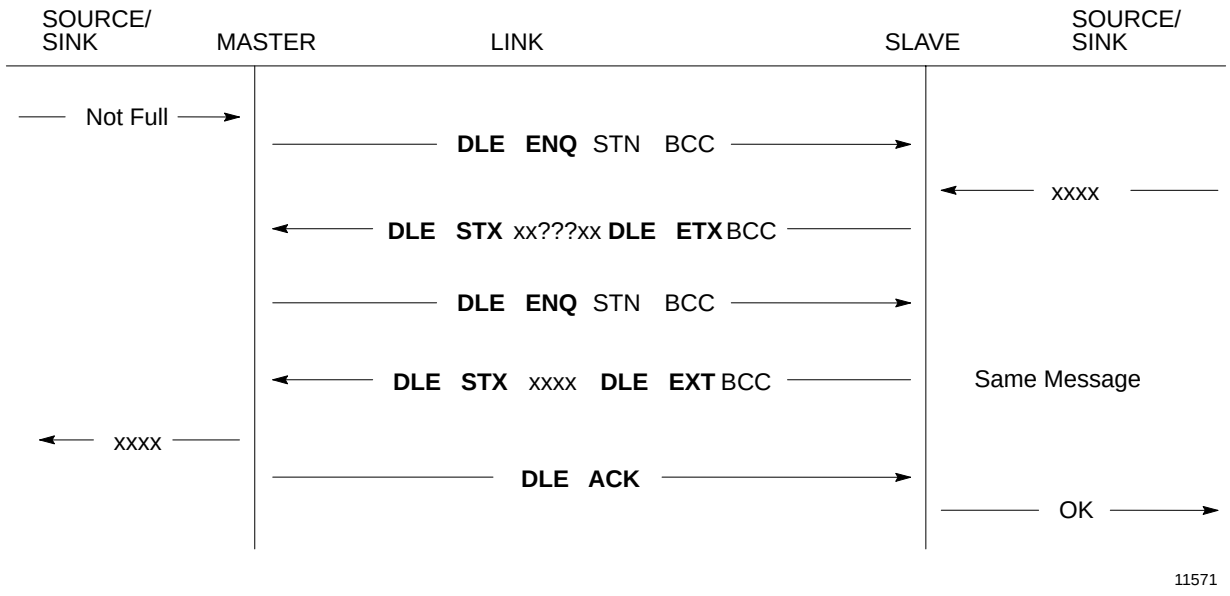
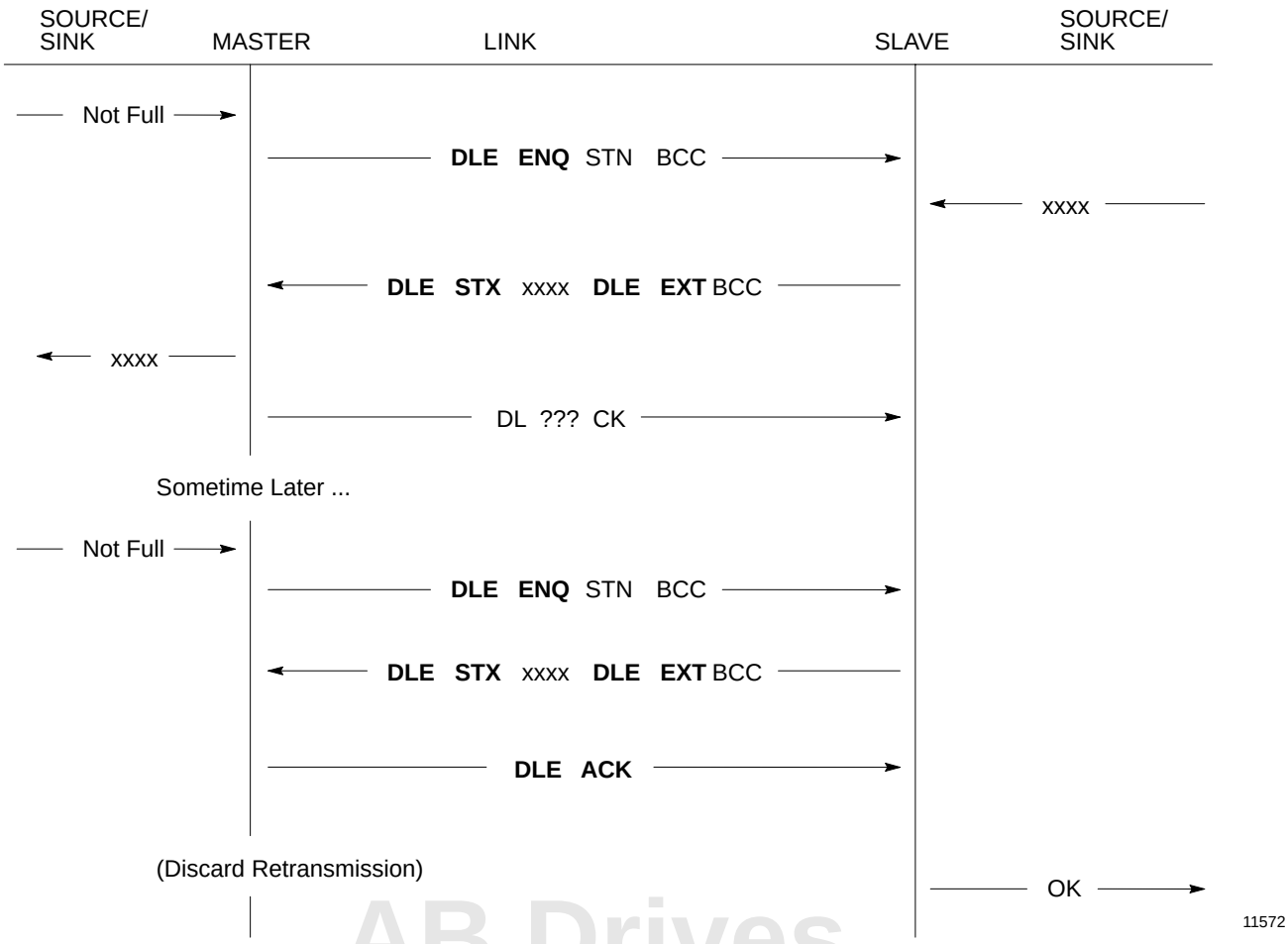


Figure 6.22 shows slave unable to receive the acknowledgment from the master after the master successfully received the message from the slave. Sometime later when the master polls that same slave again, the slave sends the same message again. The master responds with DLE ACK, but discards the received transmission block because it detects it to be a duplicate message received from the slave. With multiple slaves, to implement this duplicate message detection, the master must do either of the following:

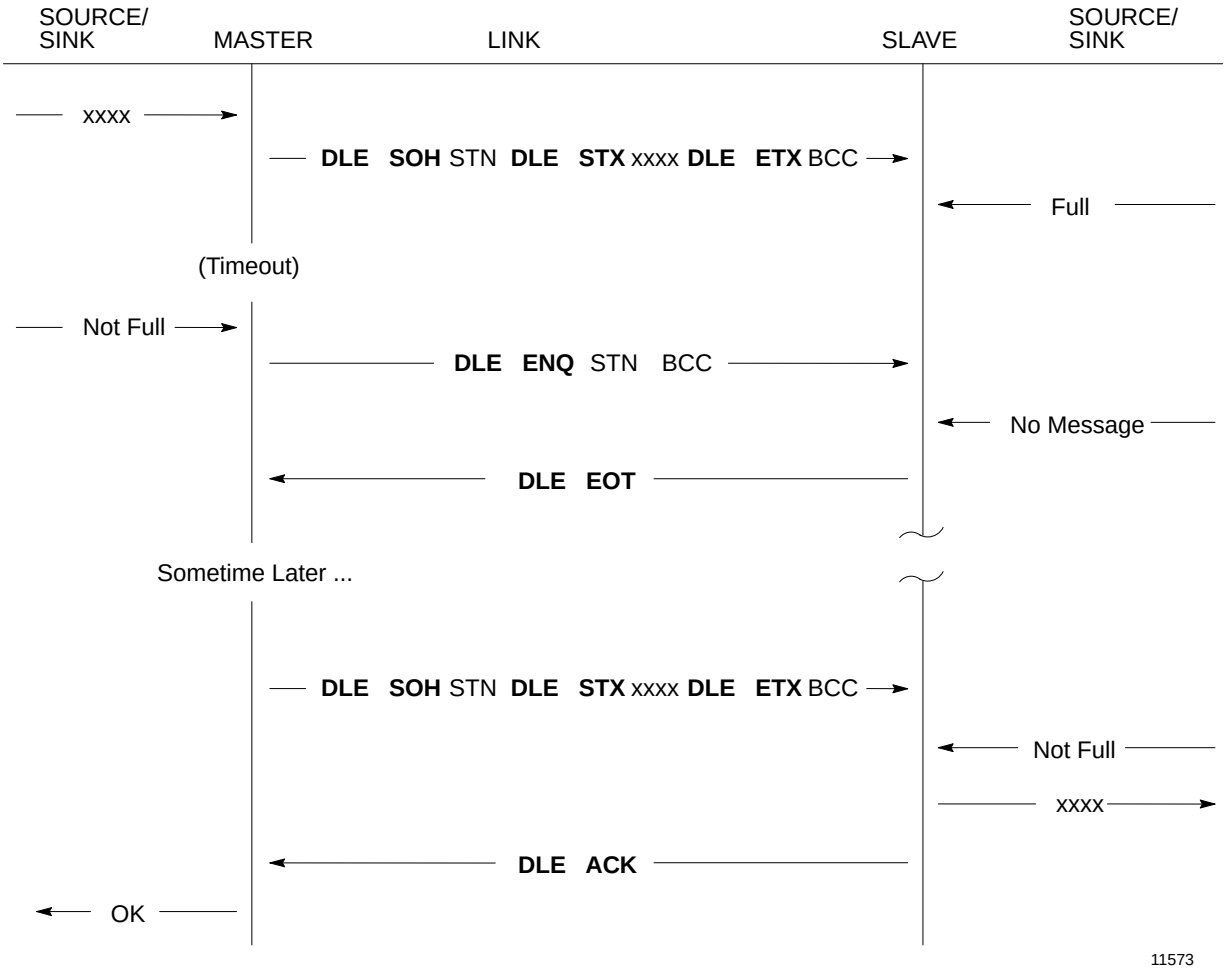
- Poll a station repeatedly (without polling any other station) until it receives a DLE EOT to be sure it has detected any retransmissions.
- If each station is polled only once per cycle, the master must keep a record of the first six link-level data bytes of the last transmission from each station, since other stations may transfer messages between retransmissions from a given station.

Figure 6.22
Duplicate Message Transmission



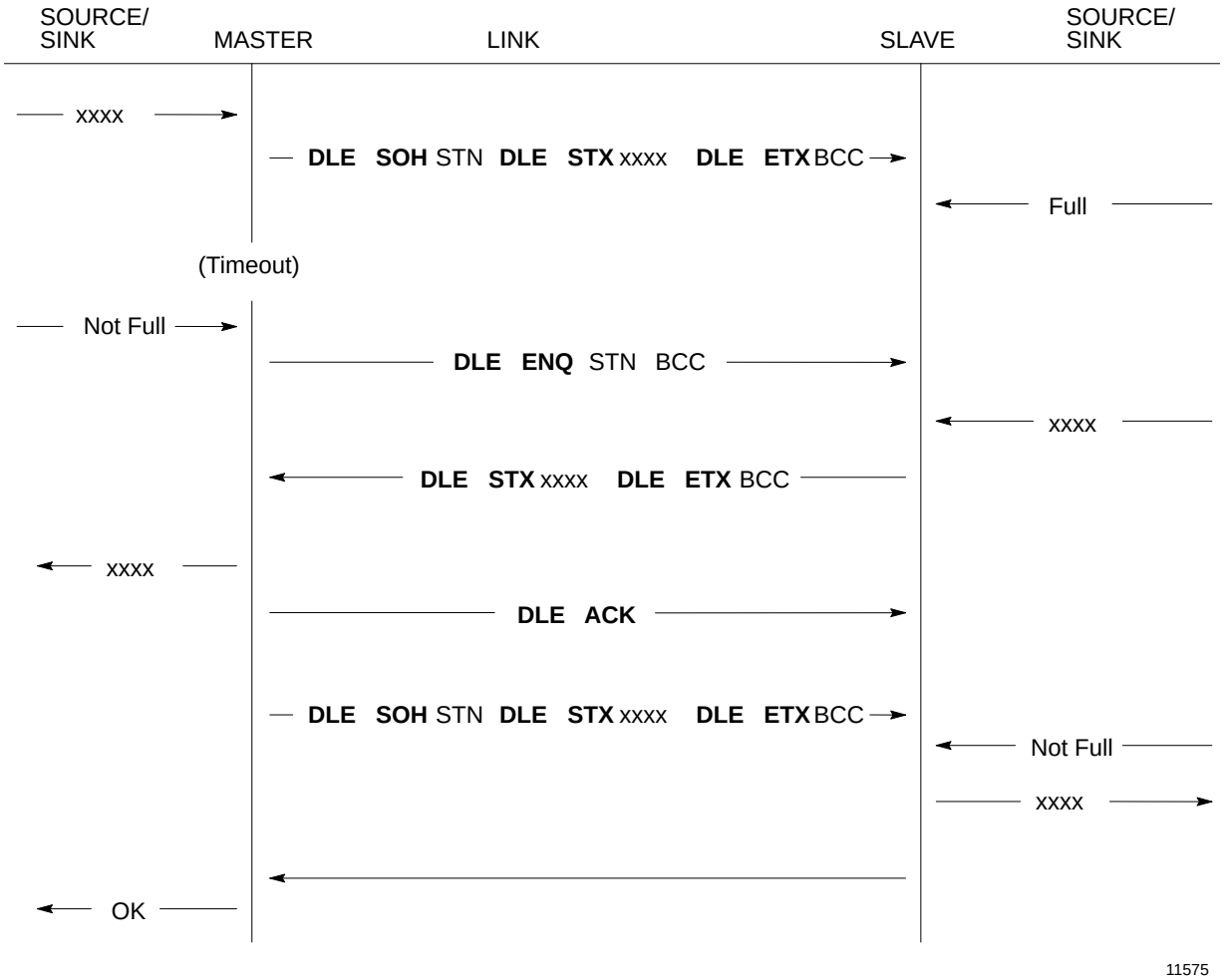
When a slave station fails to respond to a message from the master, you should poll the slave to see if it is there. If it answers the poll with a DLE EOT but consistently fails to ACK the master's message, the slave's message sink is probably full. If the slave answers with DLE EOT to a poll, you should wait for the slave's receive buffers to clear. This situation is to clear. This situation is illustrated in Figure 6.23.

Figure 6.23
Message Sink Full, Case 1



When a slave station's message source and sink share a common memory pool (as in the 1771-KG module) it may be that the message sink full indication results from an abundance of messages in the message source, which use up all free pool memory. In this case, the memory can be freed up by receiving messages from that slave station. Waiting for the memory to clear by the action of the slave station alone may not work since it could be that the only way to free up space is for the slave to send a message to the master. This situation is illustrated in Figure 6.24.

Figure 6.24
Message Sink Full, Case 2



Line Monitoring

When monitoring half-duplex protocol on a two-wire link, you need to monitor only one line. The example below shows a message sent by the master and a reply sent by the slave in answer to a poll. Slave responses are in bold.

Message from Master to Slave:

DLE SOH STN DLE STX xxxx DLE ETX BCC **DLE ACK**

Message Sent from Slave to Master in Answer to Poll:

DLE ENQ STN BCC **DLE STX xxxx DLE ETX BCC** DLE ACK

Poll with a DLE EOT Answer:

DLE ENQ STN BCC **DLE EOT**

Network Layer

The network protocol defines a network packet format for interaction between application programs. Regardless of which link protocol is used, the link protocol merely serves to carry data blocks between two applications. The application programs may be located at opposite ends of a point-to-point full-duplex link or at different points on a multi-drop half-duplex link. The network protocol can even handle the transfer of messages between application programs resident in the same device.

The network layer ignores the internal functioning of link protocols. It requires that the link level driver accepts a message for delivery, tries to send it, and indicates whether it was delivered.

Program and Message Types

Design of the network protocol is based on the assumption that application programs are of two types—command initiators and command executors. Corresponding to this division there are two message types:

- Command Messages — Initiated by command initiators and carried over the network to a command executor.
- Reply Messages — The replies that command executors send to command initiators.

For each command message there is normally one and only one reply. The only exception is when a link delivers a message but an acknowledgment is not received to verify delivery. In this case the network layer of the command initiator station generates a reply message and sends it to the command initiator in the application layer. At the same time, the command packet will be delivered and executed, and the

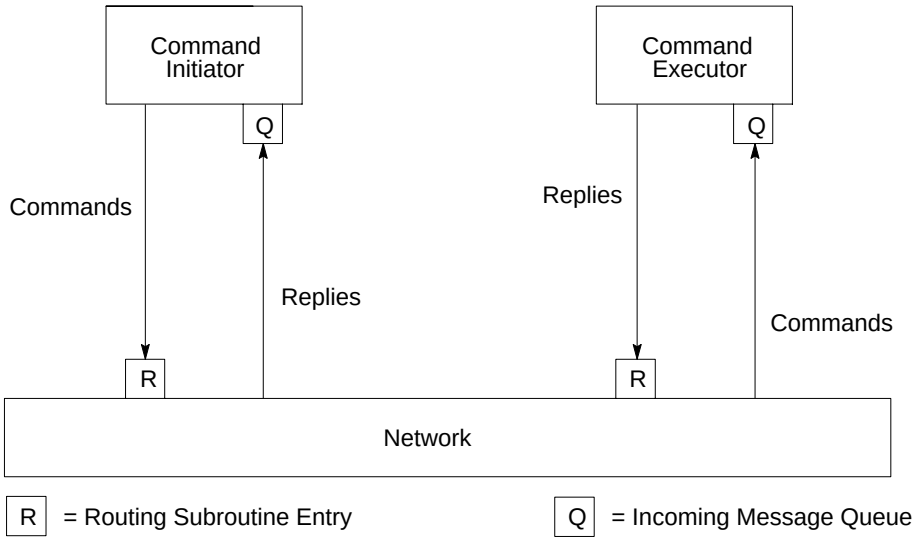
executor returns a reply message. This exception is rare. In any case, the command executor must generate only one reply message for each command it receives.

If the network layer of the command initiator station cannot deliver a command to another station, it generates a reply message with an error code for the command initiator in its own application layer. If a reply cannot be delivered, the network layer destroys it.

Network Model

To implement your Data Highway network layer software, use a routing subroutine and a queue. Messages that have been created by the application are sent to the router for transmission over the network. Messages that are delivered by the network are placed on an incoming message queue that is unique for each application. Figure 6.25 illustrates this model.

Figure 6.25
Application Model



Reply messages are not necessarily sent in the same order that their corresponding command messages were received. It is impossible for the network to guarantee delivery, and in some cases it may not be possible to provide notification of non-delivery. Therefore, it is advisable that the command initiator maintain a timer for each outstanding command message. Non-deliverable reply messages are not returned to the command executor.

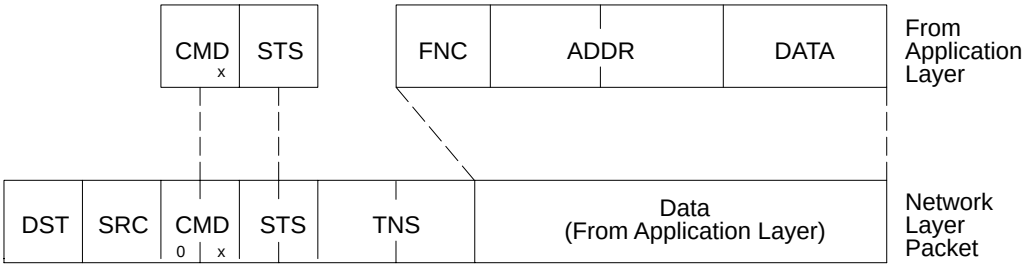
The application task is notified via the operating system when a message arrives on the queue. Messages do not necessarily have to be removed from the queue in order of arrival.

Network Packet Fields

As we discussed the communication protocol used on the link level, we described control characters framing the network packet. Here at the network level, you must generate the network packet. In this protocol the network packet characters are generated directly from binary coded bytes of data. This provides faster throughput on the link than if this data were coded into ASCII characters.

Figure 6.26 shows the general format of the network packet for a command message. Figure 6.27 shows the general format of the message text for a reply message. Note that bytes are shown from left to right in the order in which they are transmitted on the link.

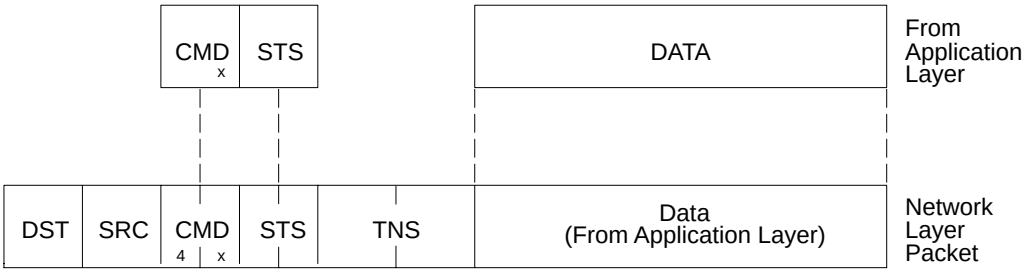
Figure 6.26
Command Message Packet Format



Legend: x = low hex digit of CMD byte supplied by application layer

11329

Figure 6.27
Reply Message Packet Format



Legend: x = low hex digit of CMD byte supplied by application layer

11330

Note that the only difference between the network packet for a command message and the network packet for a reply message is in the high nibble of the CMD byte.

DST and SRC

The DST (destination) byte is the number of the station to which this packet is delivered. The SRC byte is the number of the station that sent the packet. There are 239 possible station numbers from zero to 63 and 72 to 254 decimal. For multi-drop links, a DST value of 255 indicates a broadcast command message. The DST and SRC of a reply message are formed by reversing the DST and SRC of the corresponding command message.

CMD (High Nibble)

The high nibble of the CMD (command) byte is supplied by the network layer. Bit 6 (2^6 value) of the CMD byte is the command/reply indicator. It is zero for command messages and one for reply messages. Therefore the high hex digit of the command byte is zero for command messages and four for reply messages.

STS (Low Nibble)

The low nibble of the STS (status) byte is supplied by the network layer. In a command message, this field is set to zero. In a reply message reporting no error or a remote error, this field is also set to zero.

If the network layer of your computer cannot deliver a command to another station, it writes a local error code into this field to generate a reply message which it returns to the command initiator in your application layer. All error codes are listed in Table A.A.

TNS

The two TNS (transaction) bytes contain a unique 16-bit transaction identifier field. A complete transaction consists of a command message and its corresponding reply message. The TNS value in the reply must be the same as the TNS value in its associated command. This enables the command initiator to associate an incoming reply message with one of the command messages it transmitted previously.

For command messages transmitted by a PC station, the 1771-KG module assigns the TNS values. For each command message transmitted by your computer station, your application programs must assign a unique 16-bit transaction number. A simple way to generate the transaction number is to maintain a 16-bit counter in your application program. Increment the counter every time your command initiator (application program) creates a new message, and store the counter value in the two TNS bytes of the new message.

When your computer program receives a reply to one of its command messages, it can use the TNS value to tie the reply message to its corresponding command. If the TNS value of a reply message matches the TNS value of a command message, then that reply is the appropriate one for that command.

Whenever your computer network layer receives a command from another station, it should copy the TNS bytes of the command message into the same bytes of the corresponding reply message. Do not change the TNS value in a reply message. If you do, the command initiator will not be able to match its command to the corresponding reply message.

Note that the low byte (least significant bits) of your TNS value will be transmitted across the link before the high byte (most significant bits).

At any instant, the combination of SRC, CMD, and TNS values are sufficient to uniquely identify every message packet in transit. At least one of these fields in the current message must be different than the corresponding field in the last message received by a command executor. If none of these fields is different, the command executor ignores the current received message. This is the process of duplicate message detection.

During an upload or download, the TNS value is the only way to distinguish between the physical read or write reply messages.

Application Layer

Any station may have one or more command initiators. It should have at least one command executor to handle diagnostics, and may have more.

The command initiators are responsible for:

- Creating a message packet, submitting that packet to the network layer.
- Maintaining the sequence number and the timeout.
- Accepting the reply.
- Canceling the timeout and sequence number.
- Destroying the reply message packet when it is no longer needed.

Command executors must:

- Create the reply message packet.
- Copy over certain information from the command.
- Fill in any reply information.
- Submit the packet to the network.
- Destroy the command packet.

Application programs communicate by sending information back and forth in the command, status, and data fields of network packets. Application protocols may vary depending on the types of application programs that are communicating.

Application Message Fields

Figure 6.26 shows the general format of the application fields for a command message. Not all command messages have FNC, ADDR, or DATA bytes.

Figure 6.27 shows the general format of the application fields for a reply message. Not all reply messages have DATA bytes.

CMD and FNC

The low nibble of the CMD (command) byte and the FNC (function) byte define the type of action to be performed by the command executor at the destination station for these message formats which include an FNC byte. The CMD byte alone defines the type of action to be performed by the command executor at the destination station for those message formats which do not include an FNC byte. For each of the types of messages that can be transmitted across this link, the hexadecimal values of the CMD and FNC bytes are listed in .

Table 6.C
Hexadecimal Values of CMD and FNC Bytes

Command	Command Message		CMD
	CMD	FNC	
Diagnostic Counter Reset	06	07	46
Diagnostic Loop	06	00	46
Diagnostic Read	06	01	46
Diagnostic Status	06	03	46
Enter Download Mode	07	04	47
Enter Upload Mode(1)	07	06	47
Exit Download/Upload Mode	07	05	47
Physical Read	04		44
Physical Write	03		43
Protected Bit Write	02		42
Protected Block Write	00		40
Set Data Table Size(1)	06	08	46
Set ENQs	06	06	46
Set NAKs	06	05	46
Set Timeout	06	04	46
Set Variables	06	02	46
Unprotected Bit Write	05		45
Unprotected Block Read	01		41
Unprotected Block Write	08		48

(1) Available on Series B module only.

Bits 0 through 3 of the CMD byte must be the same in the reply message as they are in the corresponding command message. In current implementations this is either a command code or a command executor selector. The application program must always copy this field from the command to the reply message.

STS

The high nibble of the STS (status) byte is supplied by the application layer. In command messages the STS byte is set to zero.

In reply messages the STS is used for reporting either application or network error codes. A value of zero should be interpreted as no error (that is, the message was delivered and executed successfully). Non-zero status can be divided into two categories—remote errors and local errors.

Remote errors mean that a command was successfully delivered by the network, but the remote station was unable to execute the command. The remote station then formatted a reply with the high nibble of the STS byte containing some error code.

Local errors mean that your network layer was unable to deliver the message to the remote station and receive an acknowledgment. Your network layer then turns the command around, stuffs the low nibble of the STS byte with the appropriate error code, and returns it to your application level.

All error codes are listed in Appendix A. Only the last 11 apply to PC-to-computer communication.

When you receive a reply message from a PC station, check the STS byte at the application layer. If the STS byte is non-zero, refer to Appendix A for the type or error that has occurred.

When your application layer detects an error while attempting to execute a received command message, it should format a reply message with a remote error code in the high nibble of the STS byte for the remote station to interpret.

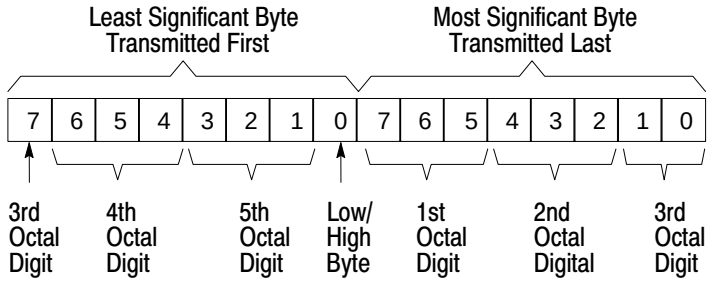
ADDR

The ADDR field is a two-byte field sent low byte first. In PC programming, logical octal byte addressing is used. A protected/unprotected (logical) read/write command operates on the ADDR field based on the same octal addressing format as shown in Figure 6.28.A. A physical read/write command operates on the ADDR field based on the physical addressing format shown in Figure 6.28.B. If you access a word through a logical read/write command with a logical address and then attempt to access the same word through a physical read/write command with the same logical address, you would actually address a different word because a physical address is assumed. If you always read and write into the data table with commands that use logical addressing you will not have to be concerned with the difference.

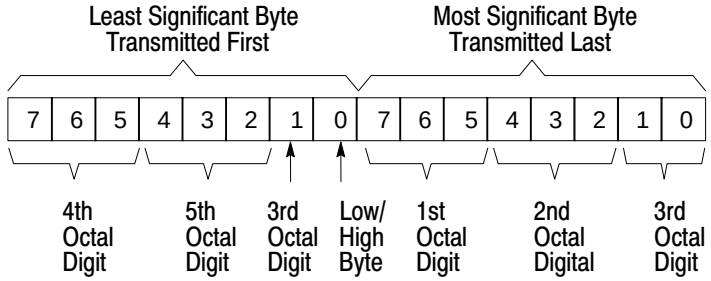
For a block read/write command, always set the least significant bit of the ADDR field to zero to select the low byte of the word. To allow PC read/write commands to the computer, set up a file in the computer to be addressed in the same way as if it were a PC data table. Refer to Appendix C for a detailed description of PC addressing.

Figure 6.28
PC Data Table Byte Addressing

A. Protected/Unprotected Read/Write ADDR Field



B. Physical Read/Write ADDR Field



11331

DATA

The DATA field must always contain an even number of bytes because the PC data table and program instructions are made up of 16-bit words. Transmit the low byte first.

Application Message Formats

PLC-2-family processors use the message formats described in this section. You must follow these message formats when generating a command or reply message from your computer application program.

The format for each command and its corresponding reply is listed below in the following order:

- Diagnostic Counters Reset
- Diagnostic Loop
- Diagnostic Read
- Diagnostic Status
- Enter Download Mode
- Enter Upload Mode
- Exit Download/Upload Mode
- Physical Read
- Physical Write
- Protected Bit Write
- Protected Block Write
- Set Date Table Size
- Set ENQs
- Set NAKs
- Set Timeout
- Set Variables
- Unprotected Bit Write
- Unprotected Block Read
- Unprotected Block Write

In each of these formats, the hexadecimal value is given for the CMD and FNC bytes. In each of these formats, the fields provided by the network layer are also shown to indicate where the application layer fields fit in the network packet. However, the fields provided by the network layer are shown shaded to indicate that they are not provided by the application layer.

Diagnostic Counters Reset

This command reset to zero all the diagnostic timers and counters used by the 1771-KG module. The diagnostic status command gives the starting address for this block of counters and timers.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 07

Reply Format:

DST	SRC	CMD 46	STS	TNS

Diagnostic Loop

This command checks the integrity of transmissions over the communication link. It transmits up to 243 bytes of data to a 1771-KG module. The receiving module should reply to this command by transmitting the same data back to the originating station.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 00	DATA – Max. of 243 Bytes

Reply Format:

DST	SRC	CMD 46	STS	TNS	DATA – Max. of 243 Bytes

Diagnostic Read

This command reads up to 244 bytes of data from the internal ROM and RAM areas of the 1771-KG module. It can be used to read the module's diagnostic timers and counters (refer to Appendix B).

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 01	ADDR	SIZE

Reply Format:

DST	SRC	CMD 46	STS	TNS	DATA – Max. of 244 Bytes

Diagnostic Status

This command reads a block of status information from the 1771-KG module.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 03

Reply Format:

DST	SRC	CMD 46	STS	TNS	DATA – 10 Bytes

The information is as follows:

Byte	Meaning
1	Operating status of PC processor, as shown in Table 6.D.
2	Type of station interface module, as shown in Table 6.E.
3, 4	Octal word address of the start of PC program.
5, 6	Always zero.
7, 8	Starting address of diagnostic counters and timers.
9	Revision level of the module, as shown in Table 6.F.
10	Settings of communication rate and option switches on the module, as shown in Table 6.G.

Table 6.D
Status Byte 1 Bit Values

Bits	Value	Meaning
0 to 2	0	Program Load
	1	Test
	2	Run
	3	(Reserved for Future Use)
	4	Remote Program Load
	5	Remote Test
	6	Run/Program
	7	(Reserved for Future Use)
3	0	Normal
	1	No Communication with PC Processor
4	0	Normal
	1	Download Mode
5	0	Normal
	1	Module Shutdown Due to Errors

Table 6.E
Status Byte 2 Bit Values

Bits	Value	Meaning	
0 to 3	4	1771-KG	**Module Type**
4 to 7	2	PLC-2/20 (LP1)	
	3	Mini PLC-2	
	5	PLC-2/20 (LP2)	**PC Processor Type**
	6	PLC-2/15	
	7	PLC-2/30	

Table 6.F
Status Byte 9 Bit Values

Bit	Value	Meaning
0 to 4	0	Revision A
	1	Revision B
	.	.
	.	.
	.	.
5 to 7	0	Series A
	1	Series B
	2	Series C
	.	.
	.	.
	.	.

Table 6.G
Status Byte 10 Bit Values

Bits	Value	Meaning			
1	1 0	First Module Second Module			
2	1 0	Accept Physical/Unprotected Reads/Writes Enabled Accept Physical/unprotected Reads/Writes Disabled			
7, 6, 5	000 100 010 110 001 101 011 111	110 bits/s 300 bits/s 600 bits/s 1,200 bits/s 2,400 bits/s 4,800 bits/s 9,600 bits/s 19,200 bits/s	**Communication Rate**		
4, 3, 0		**Parity**	**Error Check**	**Embedded Responses**	**Protocol**
	000	None	BCC	Disabled	Full-Duplex
	100	Even			
	010	None	BCC	Enabled	
	110	Even			
	001	None	BCC	Disabled	Half-Duplex
	101	Even			
	011	None	CRC	Enabled	Full-Duplex
	111	None	CRC	Disabled	Half-Duplex

Enter Download Mode

This command puts the PC processor into the download mode. You must only use it together with other commands in a download procedure [see "Upload/Download Procedures (Series B Modules)"].

Command Format:

DST	SRC	CMD 07	STS 00	TNS	FNC 04

Reply Format:

DST	SRC	CMD 47	STS	TNS

Enter Upload Mode

This command puts the PC processor into the upload mode. You must only use it together with other commands in an upload procedure [see "Upload/Download Procedures (Series B Modules)"]. This command is available on the Series B module only.

Command Format:

DST	SRC	CMD 07	STS 00	TNS	FNC 06

Reply Format:

DST	SRC	CMD 47	STS	TNS

Exit Download/Upload Mode

This command exits the PC processor from either the upload mode or the download mode. You must only use it together with other commands in an upload or download procedure [see "Upload/Download Procedures (Series B Modules)"].

Command Format:

DST	SRC	CMD 07	STS 00	TNS	FNC 05

Reply Format:

DST	SRC	CMD 47	STS	TNS

Physical Read

This command reads a block of data, specified by a physical address, from the PC data table or program area of memory. You must only use it together with other commands in an upload procedure [see "Upload/Download Procedures (Series B Modules)"]. With a Series A module, you must first put the PC processor into the program load mode before using the physical write command. It requires a physical address.

Command Format:

DST	SRC	CMD 04	STS 00	TNS	ADDR	DATA

In the ADDR field, specify an even address. In the DATA byte, specify an even number of bytes to read.

Reply Format:

DST	SRC	CMD 44	STS	TNS	DATA – Max. of 244 Bytes

Physical Write

This command writes a block of data specified by a physical address into the PC data table or program area of memory. You must only use it together with other commands in a download procedure [see "Upload/Download Procedures (Series B Modules)"]. With a Series A module, first put the PC processor into the program load mode before using the physical write command. It requires a physical address.

Command Format:

DST	SRC	CMD 03	STS 00	TNS	ADDR	DATA – Max. of 242 Bytes

In the ADDR field, specify an even address. In the data field, enter an even number of bytes to be written.

Reply Format:

DST	SRC	CMD 43	STS	TNS

Protected Bit Write

This command sets or reset individual bits within limited areas of the PC data table memory. Its access is limited by memory access rungs in the communication zone of the PC's ladder diagram program.

Command Format:

DST	SRC	CMD 02	STS 00	TNS	ADDR	DATA	ADDR	DATA
					Up to 61 ADDR and DATA Fields			

Reply Format:

DST	SRC	CMD 42	STS	TNS

Following each ADDR field you must include a DATA field. Each DATA field must consist of a set mask byte followed by a reset mask byte. In the set mask, set a one for each bit you want to set. In the reset mask, set a one for each bit you want to reset. The bits left at zero in both masks are left unchanged by this command. If you set a bit to one in both masks, the bit becomes reset.

Protected Block Write

This command writes words of data into limited areas of the PC data table memory. Its access is limited by memory access rungs in the communication zone of the PC's ladder diagram program.

Command Format:

DST	SRC	CMD 00	STS 00	TNS	ADDR	DATA – Max. of 242 Bytes

Reply Format:

DST	SRC	CMD 40	STS	TNS

Set Data Table Size

This command sets the data table size in bytes. The data table size is also the physical address of the start program. Select a proper data table size as specified in the programming manual for the PLC-2-family processor. You must only use this command together with other commands in download procedure [see "Upload/Download Procedures (Series B Modules)"]. This command is available on the Series B module only.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 08	DATA

Reply Format:

DST	SRC	CMD 46	STS	TNS

SET ENQS

This command sets the maximum number of ENQs that the module issues per message transmission. The default setting is three.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 06	DATA 1 Byte

Reply Format:

DST	SRC	CMD 46	STS	TNS

SET NAKS

This command sets the maximum number of NAKs that the module accepts per message transmission. The default setting is three.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 05	DATA 1 Byte

Reply Format:

DST	SRC	CMD 46	STS	TNS

Set Timeout

This command sets the maximum amount of time that the module waits for an acknowledgment to its message transmission. The setting is expressed as the number of cycles of an internal clock, where 38 cycles equals 1 second. The default setting is 38 cycles, or 1 second.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 04	DATA 1 Byte

Reply Format:

DST	SRC	CMD 46	STS	TNS

Set Variables

This command is a combination of the set timeouts, set NAKs and set ENQs commands. It sets the maximum ENQs, NAKs, and timeouts all at once. You must specify all three.

Command Format:

DST	SRC	CMD 06	STS 00	TNS	FNC 02	DATA – 3 Bytes Timeouts NAKs ENQs

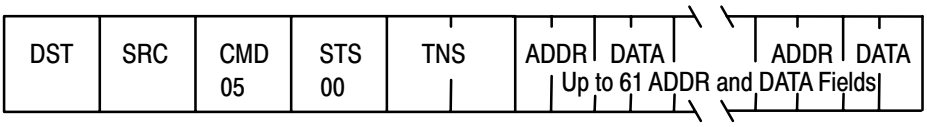
Reply Format:

DST	SRC	CMD 46	STS	TNS

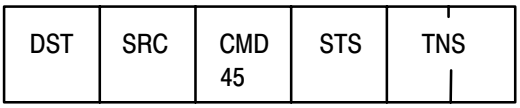
Unprotected Bit Write

This command sets or resets individual bits in any area of PC data table memory.

Command Format:



Reply Format:

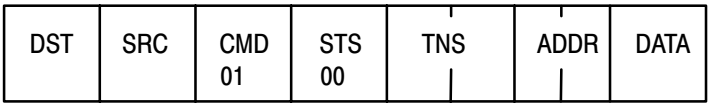


Following each ADDR field you must include a DATA field. Each DATA field must consist of a set mask byte followed by a reset mask byte. In the set mask, set a one for each bit you want to set. In the reset mask, set a one for each mask you want to reset. The bits left at zero in both masks are left unchanged by this command. If you set a bit to one in both masks, the bit becomes reset.

Unprotected Block Read

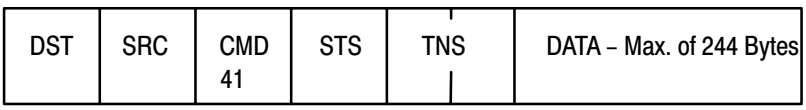
This command reads a block of data from any area of PC data table memory.

Command Format:



In the ADDR field, specify the starting address. In the DATA byte, specify the number of bytes to read.

Reply Format:



Unprotected Block Write

This command writes a block of data into any area of PC data table memory.

Command Format:

DST	SRC	CMD 08	STS 00	TNS	ADDR	DATA – Max. of 242 Bytes

Reply Format:

DST	SRC	CMD 48	STS	TNS

Several PLC-2 program scans may go by while the entire block is written into the data table. Therefore, it is generally good practice to buffer the data in one area and to allow the program to move it to the data table area where it is to be used only after the operation is complete.

Upload/Download Procedures (Series B Modules)

A common application for a 1771-KG module is to conduct data between programmable controllers and computers. Their massive storage capacities make computers ideal places to deposit and store entire master programs. Master programs are copied from controllers to computers (upload) and from computers to controllers (download). You can use those masters to verify programs as the PCs run them.

In an upload, the user accessible memory from the PC can be copied to the bulk storage device in a computer. In a download, the user accessible PC memory can be copied from the bulk storage device, to the PC memory.

The upload and download procedures which we describe here use commands which are available on the Series B module only.

Download

Since the program is replaced in a download operation, and this replacement may take several seconds, the PC must be stopped during the download so that it does not try to execute a program that is changing.

The data table is usually changed during a download, so that 1771-KG module must prevent other PC's from accessing what may be a partial data table. The module must also prevent an industrial terminal from accessing the program during a download. The module will block other access to the PC memory during download if the general procedure for downloading follows this order:

Enter Download Mode
Set Data Table Size (if required)
Physical Write
.
.
.
Physical Write
Physical Read (verification)
Exit Download/Upload Mode

If you download a program that requires a change in the data table size, send a set data table size command before sending physical write command containing the new program. The set data table size command is only available on the Series B 1771-KG module. If you have a Series A 1771-KG module, the only way you can change the data table size is through an industrial terminal.

If you send a set data table size command that increases the size of the data table, send a physical write command to clear out that part of the data table that had previously been in the program area.

When downloading a program, always write zeros into the unused portion of the memory following the program.

After you write the new program into the PC memory, verify the new program through physical read commands before exiting the download mode. However, you will be unable to verify the first word in the program until after exiting the download mode. The 1771-KG module does not write that word into memory until exiting the download mode. While in the download mode, the module writes an end (of program) instruction in the first word of the program. This end instruction keeps the PC processor from executing the program until the download mode is exited. If the 1771-KG module cannot complete a download, the end instruction is left as the first word of the program.

For the enter download mode command to be accepted by the 1771-KG module connected to a PLC-2/15 or PLC-2/30 processor, the mode select switch must be in the RUN/PROG or PROG position. The mode select switch of a PLC-2/20 or Mini-PLC-2 processor can be in any position;

however, we recommend that you select the PROG position. Results of the enter download mode command are:

- The PC program scan will be inhibited (Series B module only).
- The industrial terminal will be disabled.
- A PLC-2/15 or PLC-2/30 processor will be put into the program load mode (Series B only); with the last state switch at each I/O chassis set to OFF, the outputs will be disabled (Series B module only); the outputs of a PLC-2/20, or Mini-PLC-2 processor (or PLC-2/15 or PLC-2/30 processor with a Series A 1771-KG module) will remain enabled if the mode select switch is in the RUN or RUN/PROG position.

WARNING: Leaving outputs on during a download could cause unexpected machine motion. If you send an enter download mode command to a 1771-KG module connected to a PLC-2/20 or Mini-PLC-2 processor with the mode select switch in the RUN or RUN/PROG position, the only way to turn OFF all outputs is to immediately send a physical write command to set all output image bits to zero.

- The station will not send any commands.
- The station will not accept any commands except set data table size, mode setting, physical read, physical write, or diagnostic commands.

Upload

Physical reads during an upload operation have no effect on the PC memory intact.

However, to insure a stable program image, the 1771-KG module must provide a means of prohibiting program changes during an upload. The module will block other access to the PC memory during upload if the general procedure for uploading from a PC follows this order:

Enter Upload Mode
Physical Read
.
.
.
Physical Read
Exit Download/Upload Mode

For the enter upload mode command to be accepted by a 1771-KG module connected to a PLC-2/15 or PLC-2/30 processor, the mode select switch must be in the RUN/PROG or PROG position. The mode select switch of a PLC-2/20 or Mini-PLC-2 processor can be in any position. Results of the enter upload command are:

- The industrial terminal will be disabled.
- The station will not accept any commands except mode setting or physical read commands.

If you have a Series A 1771-KG module, the upload procedure includes sending an enter download mode command instead of an enter upload mode command.

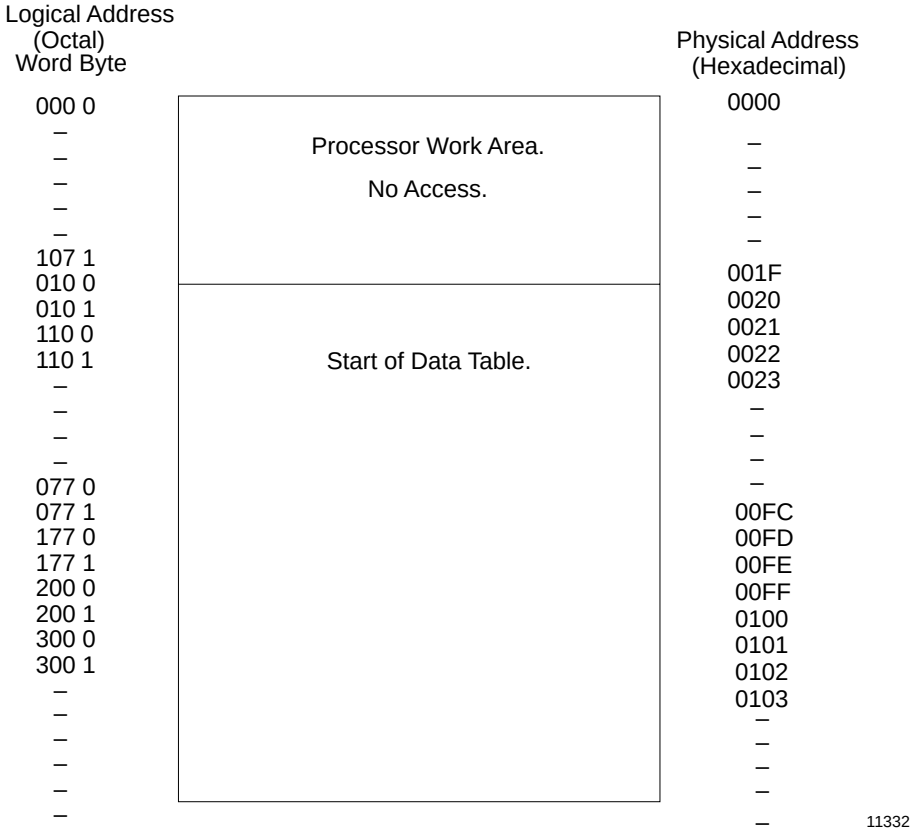
Timeout

A two-minute timer will be started when the 1771-KG module receives an enter upload mode command or an enter download mode command. The timer will be restarted when any command is received. If the timer times out, the mode is exited as if an exit download/upload command were received.

Addressing

For physical read and write commands, you must use physical addressing. Figure 6.28 shows PC data table addressing. Figure 6.29 shows PLC-2 family memory organization. For further information on addressing, refer to Appendix C.

Figure 6.29
PLC-2 Family Memory Organization



Chapter 7

Start-Up and Troubleshooting

General

This chapter outlines an approach to start-up and troubleshooting procedures.

Necessarily, exact procedures that would be followed vary from one application to the next. However, these guidelines are useful when starting up an installation or when trying to locate specific fault conditions.

You can use the methods described in this chapter to test any station which contains a PLC-2 family programmable controller and a 1771-KG module.

When you have a computer communicating to the module, additional procedures are available for start-up and troubleshooting. Among these added capabilities, a set of diagnostic commands is available. Even if a Data Highway network includes a computer, follow the procedures of this chapter for initial testing of the station. Once you have tested each station containing a programmable controller, you can then check operation of the computer interfacing.

Troubleshooting Aids

This section describes the tools which are available for start-up and troubleshooting personnel. These aids are provided by:

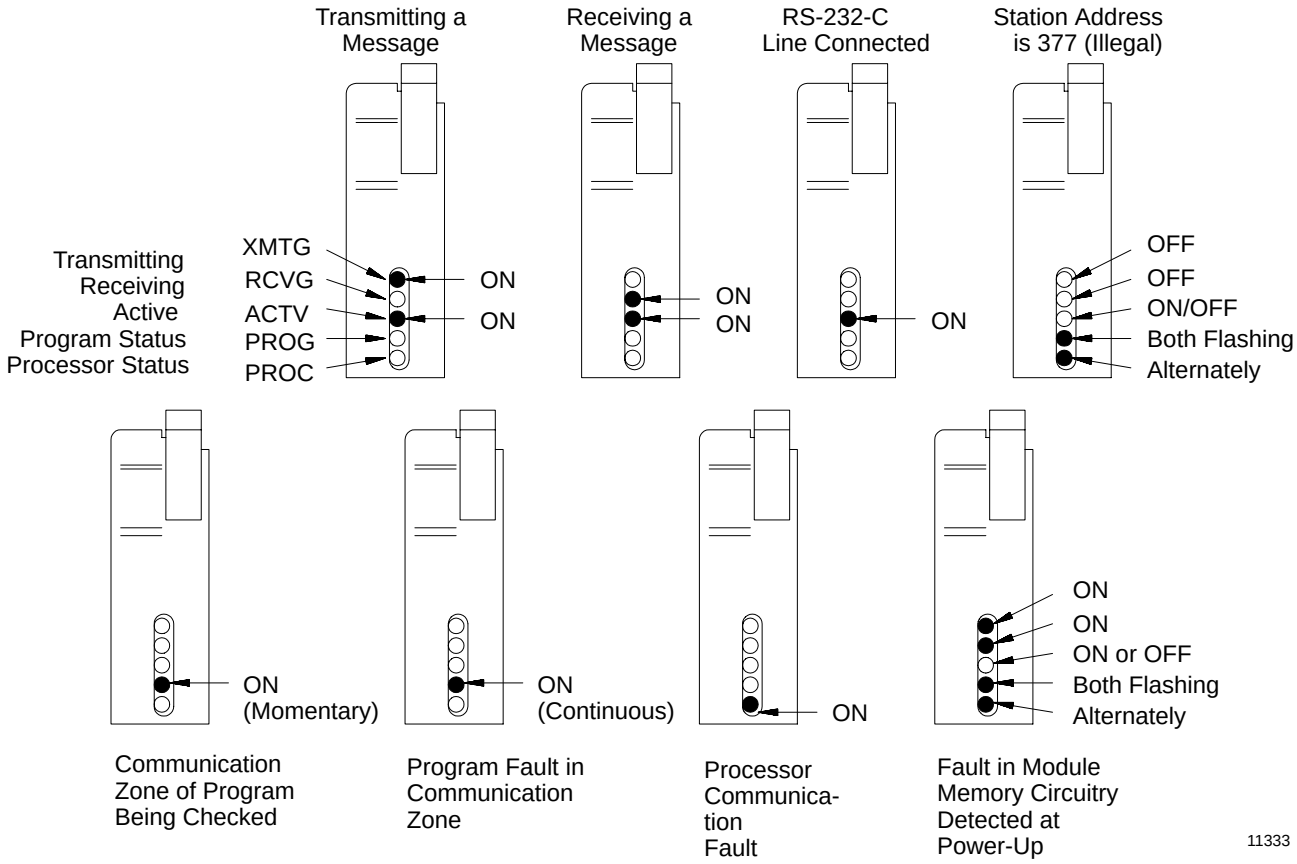
- 1771-KG Module
- Industrial Terminal
- Documentation Provided by the Programmer of the Station Processor

Module Indicators

Module diagnostic indicators show the status of module operation with its station processor. This paragraph describes these indicators and their significance in start-up and troubleshooting.

Figure 7.1 shows the significance of various combinations of energized indicators. Note that the three green indicators show module status in normal message transfer. The two red indicators show the status of module tests of the program and 1771-KG module communication with its station processor.

Figure 7.1
Module Indicator Combinations



The following paragraphs describe each indicator and its significance to the troubleshooter.

- **XMTG (Transmitting)** — This green indicator is ON when the module is transmitting command or reply messages. However, it is not ON when response codes are transmitted.
- **RCVG (Receiving)** — This green indicator is ON when the module is receiving command or reply messages.
- **ACTV (Active)** — This green indicator is ON when a cable is connected from an RS-232-C compatible device to the receiving pins of the RS-232-C PORT connector and power is ON at both ends.
- **PROG (Program Status)** — This red indicator goes on during a series of communication zone rung checks in the user's program. If the module detects an error it will remain ON until the PC processor enters program mode or is downloaded. The program status indicator is normally OFF.

The module checks the communication zone rungs of a program at power-up in run or test and when the processor is switched from the program mode to the run or test mode. It also checks these rungs after executing an exit download/upload mode command or when the cable to the processor is reconnected. If the module finds a valid header rung, during this initializing procedure, the PROG indicator turns ON. After the module has checked these communication zone rungs; and if it has found no errors in programming format for these rungs, the module turns the PROG indicator OFF. However, should any programming error be detected in the communication zone of program, this indicator remains ON. In this event, you can check the error code storage word for an indication of the problem. (Refer to "Start-Up Procedures" which appears later in this chapter.)

Should the PROG indicator fail to turn ON momentarily at power-up or when the mode select switch is turned from program load mode, the switch-selected station number may not match the station number in the header rung of the communication zone of the program or there may be some other error in the header rung.

- **PROC (Processor Status)** — This red indicator will go ON when a processor is not connected to the module. The processor status indicator is normally OFF. It flashes once for every retry due to a communication error between the 1771-KG module and the processor.

Should the PROC indicator go ON, one of the following problems may have occurred:

- Disconnection of the cable which connects the 1771-KG module and the processor.
- Power OFF at the processor.
- Fault in processor operation.

Processor troubleshooting is described in the appropriate manual for each controller.

After the processor fault is corrected, the 1771-KG module automatically checks the communication zone rungs. Should it detect proper programming, module-to-processor interaction is resumed and the PROC indicator turns OFF.

Industrial Terminal

The industrial terminal is an invaluable aid for start-up and troubleshooting. It makes available such aids as status indicators, value display, bit ON/OFF status control, fault bit monitoring, and error code display.

During initial start-up procedures for multiple PC networks, it is best to have at least two industrial terminals available. This allows both a local and a remote station to be monitored at the same time.

This section briefly reviews the following industrial terminal functions which have a special importance in troubleshooting testing:

- Search Functions
- Status Indication
- Force Functions

Each of these tools is used in the sections that follow on “Start-Up Procedures” and “Troubleshooting.”

Search Functions

The search functions enable you to quickly locate and display the various parts of the program. Table 7.A lists the search functions of the industrial terminal.

Status Indication

The industrial terminal provides a status indication for most program instructions. An instruction symbol is intensified when the instruction is logically true. For an input instruction, this means that the condition specified by the instruction has been met. For an output instruction, this means that the instruction is being carried out.

The status indication is provided when the processor is in a run or test mode.

The status indication is useful in monitoring the ON/OFF status of start/done and remote/local fault bits.

Table 7.A
Industrial Terminal Search Functions

Key Sequence	Response
→	Positions cursor on the following program instruction.
←	Positions cursor on preceding program instruction.
↑	Displays previous rung.
↓	Displays following rung.
Search ↑	Displays first rung of program.
Search ↓	Displays END statement of program.
Search Display	Single rung display. (Press same key sequence to restore multiple rung display.)
Search A xxxxxx	The specified instruction is displayed as it appears in one location in the program. Press [SEARCH] to display other locations where this instruction is used.
Search B xxx	
Search -[B]- xxxx	Displays output instructions which control the Word xxx or any of its 16 bits. After initially pressing this key sequence, simply press [SEARCH] to display each other program location in which Word xxx or its individual bits are addressed. (EXAMINE instructions not displayed by this SEARCH function.)
Search 8 xxx	

Legend:

A	A relay-type instruction which addresses a single bit. This includes EXAMINE ON, EXAMINE OFF, OUTPUT ENERGIZE, LATCH, and UNLATCH instructions.
B	An instruction which addresses a three-digit word. This includes all Timer/Counter, GET/PUT, LES, EQU, IMMEDIATE I/O, and Arithmetic instructions.
x	Numeric key.

Force-On Function

The force-on function can be a useful troubleshooting tool. When used in conjunction with the optional test rungs of "Diagnostic Counters," this function controls the initiation of each command programmed at a station. This function is in effect only as long as the programming terminal is connected to the PROGRAM INTERFACE socket of the 1771-KG module. When the industrial terminal is disconnected, the force function is removed.

Remote/Local Fault Bits

The remote and local fault bits provide the chief indicator of a hardware-related fault which prevents normal communication. As recommended in Chapter 5, these bits may be programmed to control some output device to alert plant personnel of a fault condition. When this output device signals a fault condition, you can then use an industrial terminal to display the ON/OFF status of any remote or local fault bit at the station processor.

Figure 7.2 summarizes the significance of both remote and local fault bits.

Error Word

The error word is especially useful for station start-up. The codes displayed in the lower two digits of the error word may indicate an error in programming, switch-setting, or certain other conditions that prevent normal communication.

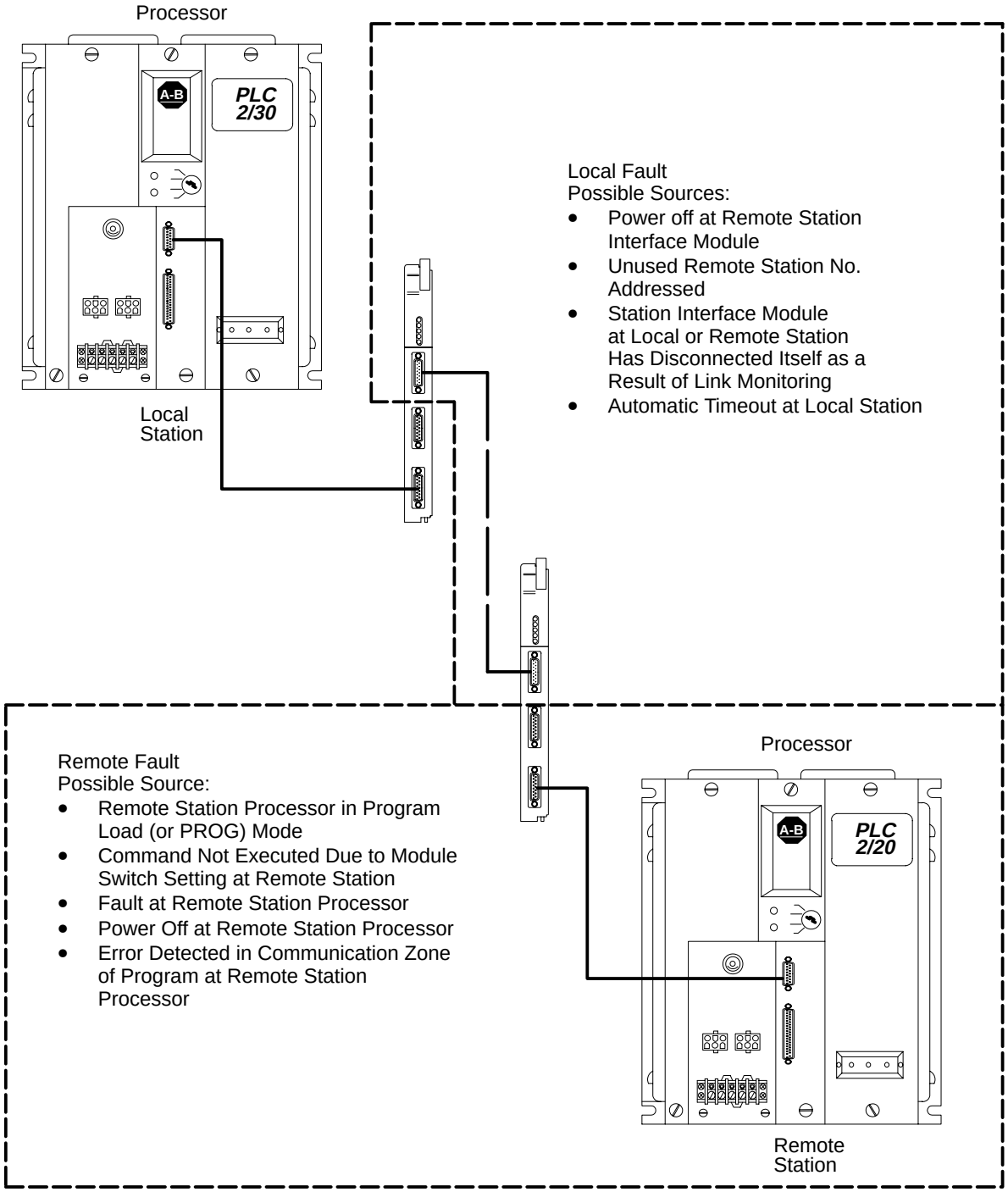
The error word is addressed by the second GET instruction in the header rung (Figure 7.3). This is the first rung of the communication zone.

Appendix A details the meaning of error codes in the status word.

Diagnostic Counters

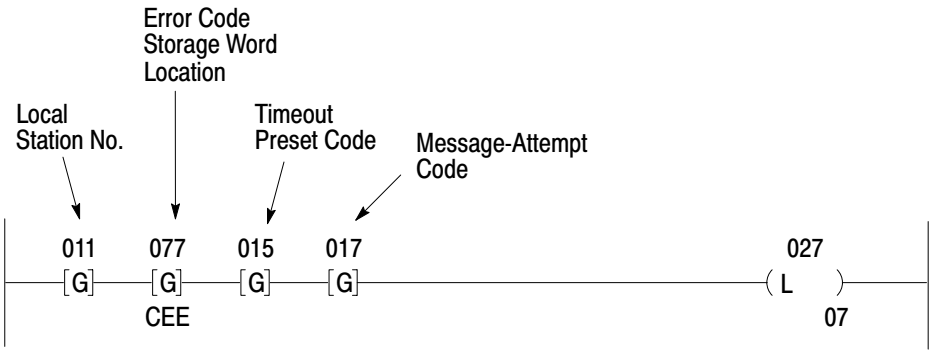
The 1771-KG module has several diagnostic counters. They keep track of communication link statistics, such as the number of messages and responses transmitted and received. You can access these counters through diagnostic commands from a computer at a remote station. Appendix B lists these diagnostic counters.

Figure 7.2
REMOTE/LOCAL FAULT Bit Significance



11316

Figure 7.3
Header Rung



Legend

C	Least Significant Digit of Rung Number for Programming Error, or Least Significant Digit of Error Code Counter
EE	Error Code

11574

Test Rungs

For start-up and troubleshooting testing, you should have some means to control each start bit. To execute a command during testing, you can then turn ON the start bit for each command using these means.

The start bit is controlled by a rung of your application program. The most direct way to control the start bit in troubleshooting, therefore, would be to manipulate the conditions of this rung to turn ON the start bit. If you can do this easily for each start bit at a station processor, the use of any special procedure for command initiation is unnecessary.

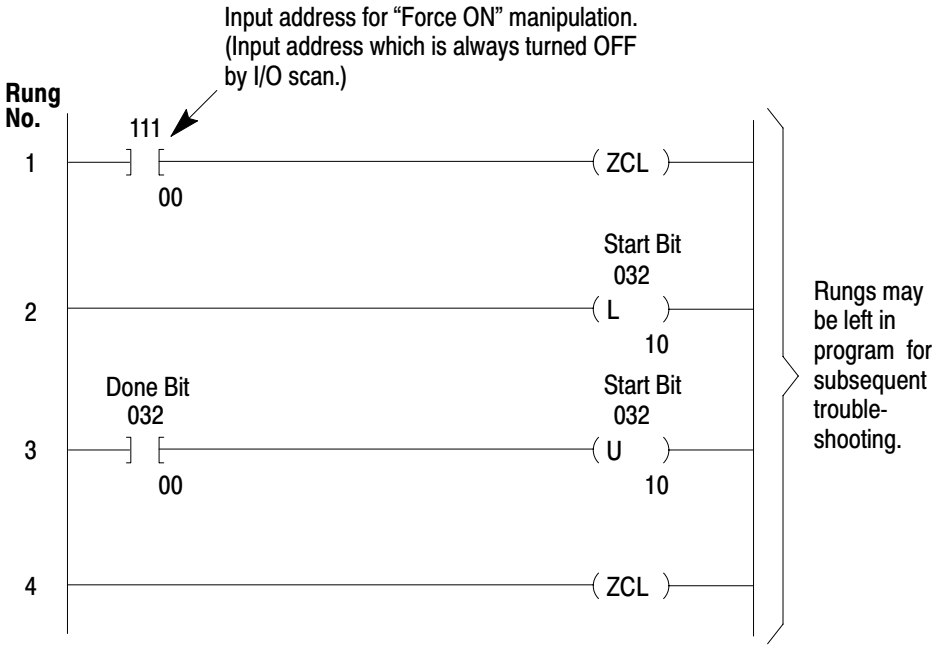
However, it may not always be practical to simulate application conditions to test a command. In this case, you could add a special set of test rungs at the end of the user program. Using these rungs, you can control command initiation directly from the programming terminal.

CAUTION: Do not alter the application program for troubleshooting purposes. This can cause undesired machine operation, since the program may no longer operate as had been intended.

Optional test rungs described here are recommended with this caution in mind. The specific format of these rungs allows control of commands without altering the main body of the application program. Because these tests rungs are within a ZCL zone, the output instructions of these rungs are executed only under strict programmer-controlled conditions and only when the programming terminal is connected. Except where intentionally activated by the proper key sequence, these rungs are ignored by program logic in normal operation.

The optional test rungs are shown in Figure 7.4. Within this ZCL zone, the start bit is unconditionally latched on (Rung 2) and unlatched when the done bit is on (Rung 3). In Rung 1, a single input image table bit is the condition for this ZCL zone.

Figure 7.4
Optional Test Rungs



The input image table bit examined in Rung 1 must always be turned OFF by the processor I/O scan. This bit can be any unused input image table bit. For this purpose, choose a bit for which the corresponding I/O chassis slot contains an output module.

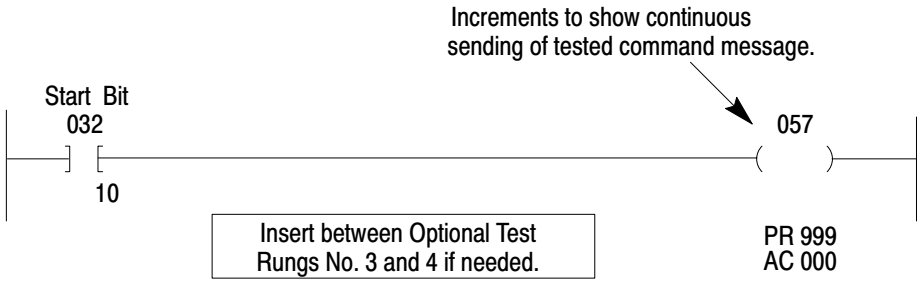
Because the processor turns this bit OFF on each I/O scan, the ZCL zone is disabled unless both of these conditions are met:

- The industrial terminal is connected to the module PROGRAM INTERFACE socket.
- The addressed input image table bit is forced ON.

When both conditions are met, you can initiate and monitor the commands programmed at a station processor. To test individual commands, enter the addresses of the corresponding start and done bits into Rungs 2 and 3. In this manner, you can test each command individually.

These rungs send the command continuously, as long as the ZCL zone is enabled. As a quick check of this continuous command execution and completion, you can add another rung to the test rungs within the ZCL zone. This rung examines the start bit as the input condition to a counter, as shown in Figure 7.5. Using this optional counter, you verify that the command is being executed continuously. To use this optional counter, insert this additional rung within the ZCL zone, between Rungs 3 and 4.

Figure 7.5
Optional Test Counter



Note that the counter value shown on the programming terminal may not display the actual number of times a command has been sent, due to CRT delay time. However, the purpose of the counter is to provide an indication to the troubleshooter that the command is being continuously executed, rather than to give an actual count of the number of times it is executed.

Again, the use of these test rungs is optional, subject to the discretion of the programmer. An advantage of these rungs is that they may be kept at

the end of the user program after start-up is completed. This enables the use of these rungs in troubleshooting or later testing, as when a command rung is subsequently added to the communication zone of program. Of course, these rungs can be removed after start-up is completed, at the programmer's option.

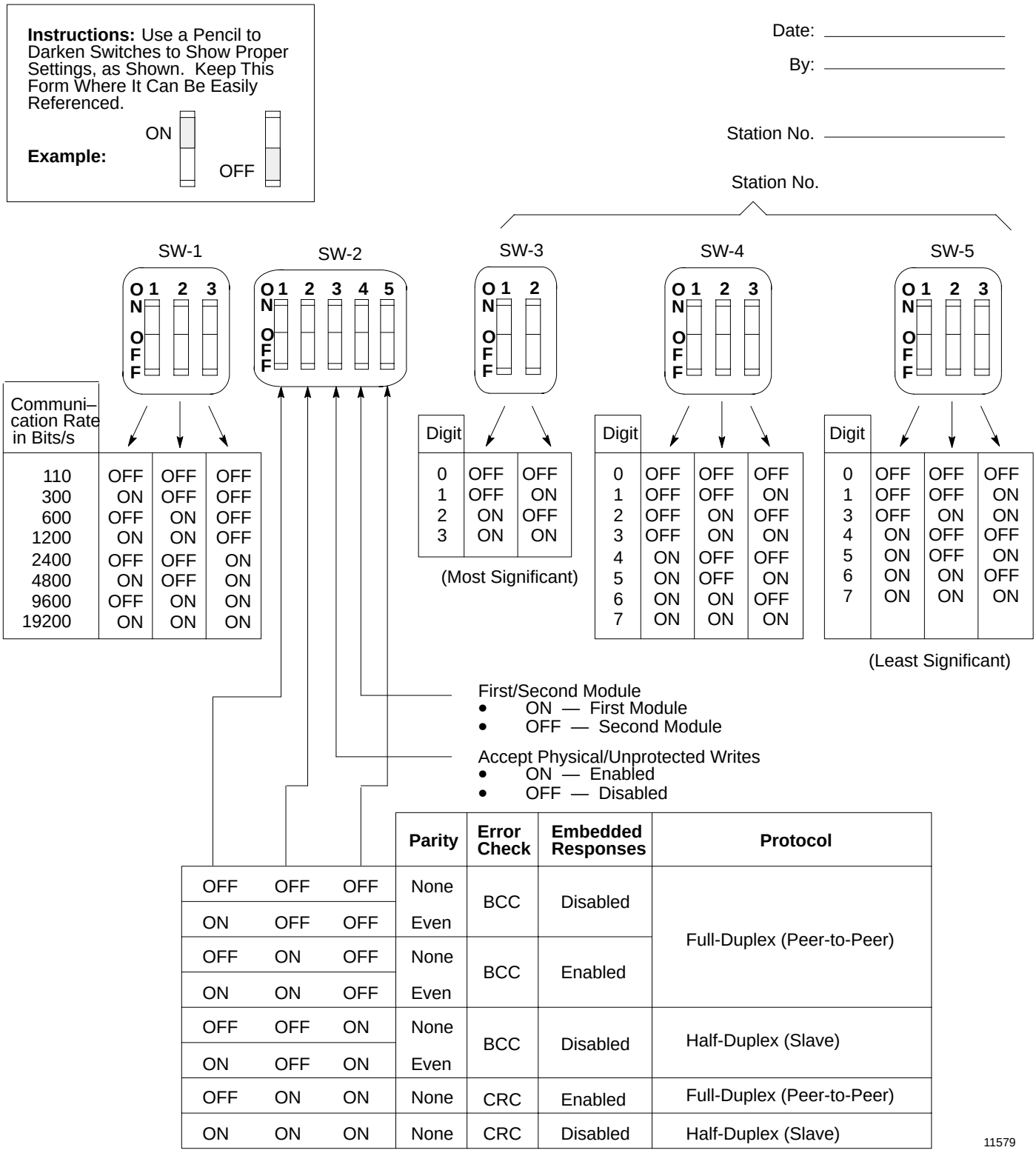
Necessary Documentation

For testing and troubleshooting command execution, the following documentation should be available at each station:

- Copy of the ladder-diagram program in the station processor.
- Completed forms giving the following information:
  - Communication module switch settings (Figure 7.6).
  - Listing of commands sent by the station.
  - Listing of commands received by the station.

A ladder-diagram printout can be generated using an RS-232-C compatible printer connected through an industrial terminal.

Figure 7.6
Switch Setting Record for 1771-KG Module



			Parity	Error Check	Embedded Responses	Protocol
OFF	OFF	OFF	None	BCC	Disabled	Full-Duplex (Peer-to-Peer)
ON	OFF	OFF	Even			
OFF	ON	OFF	None	BCC	Enabled	
ON	ON	OFF	Even			
OFF	OFF	ON	None	BCC	Disabled	Half-Duplex (Slave)
ON	OFF	ON	Even			
OFF	ON	ON	None	CRC	Enabled	Full-Duplex (Peer-to-Peer)
ON	ON	ON	None	CRC	Disabled	Half-Duplex (Slave)

11579

Start-Up Procedures

A careful start-up procedure is essential to proper network operation. With a well planned start-up procedure, you can test cabling connections, module set-up, and support programming for module communication at each station.

The start-up of a newly-installed network requires combined efforts of maintenance personnel, the programmable controller programmer, and in many cases, the computer programmer. Because applications vary widely, our recommendations in this section are general. Specific start-up procedures depend on the exact nature of the individual application.

For the procedures of this section, we assume that you have made the following preparations:

- The station is assembled using compatible components, as described in Chapter 4.
- Your application programming at the station processor includes the communication zone and the necessary support programming for initiating and monitoring user commands, as described in Chapter 5.
- Your application program is fully tested.

This last point is very important. The testing of your application program at each station is essential for proper operation. This means that you must test the application program before the network is tested. You must test this program which controls output devices. Include testing of those parts of the program which use data transferred from other stations.

In a new installation, you must complete all start-up procedures of the controller before attempting to test operation of the 1771-KG module. These procedures include complete testing of I/O devices, I/O wiring, and program sequencing. Procedures for start-up of a new controller are covered in the installation and operation manual for the controller.

In many cases, you are adding capability to an existing controller installation. Here, the application program may be only slightly modified to use data from other stations. However, you must test any change to the program at the controller so that you can correct errors before the program is put into full operation. Again, this testing is your responsibility.

Overall Approach

In start-up testing, limit the number of things happening at one time. By carefully limiting the scope of start-up testing to a small number of variables, you can more readily detect the source of a problem.

Paired Testing

Do start-up testing with two stations communicating. After comprehensive testing of both stations, place the station processors in the run mode for normal operation.

Select one of the two stations as a starting point. For the purpose of this description, this station is termed Station A. The station to which Station A sends a command is Station B. First test message transfer between Stations A and B. Test and monitor each command from A to B to verify proper operation. Then check Station B for any commands which it received from Station A and for any commands which it sends to Station A.

By following this procedure, test each command for proper execution by the communication modules at each station. This testing also checks the support programming done at each station to initiate and monitor commands, including remote/local fault monitoring.

Use the procedures in “Testing the Local Station” and “Testing the Remote Station” for paired testing of stations. For a station sending a command, carry out the procedures of “Testing the Local Station.” At the receiving station, carry out the procedures of “Testing the Remote Station.”

Operation

Once you have checked the interaction of all station communication modules, you can put the station processors into operation, one at a time. You must determine which controller to put in the run mode initially and in what order you should add other stations in the run mode. Monitoring of station interaction can continue during these procedures.

By adding stations one at a time, you can exercise maximum control over the application and monitor controller behavior carefully.

Power-Up

Only the two stations being examined in the first phase of start-up should be ON. Follow these steps in powering up each of these stations:

1. Put the processor into the program load mode.

2. Turn power ON at the processor and communication module. Observe power supply and processor indicators for proper status.

3. While observing the indicators of the module, put the processor into the test mode.

Within a few seconds, the PROG indicator on the 1771-KG module should turn ON briefly, then OFF. This indicates that the module has checked the communication zone rungs of the program. If this indicator remains ON, an error may have been detected in these rungs. Should this be the case, check the error code storage word, as described in Step 4.

If this indicator does not turn ON, the header rung may be incorrect or missing. Check this rung if necessary.

The indicator labeled PROC should be OFF. If this indicator is ON, check for a processor fault indication or poor cable connection.

4. If the PROG indicator does not turn OFF after a few seconds, check the error code storage word.

The error code storage word is listed in the header rung of the communication zone of the program.

The significance of the error code storage word is described in “Error Word.” Correct the communication zone rungs as indicated by the error code. Then repeat Step 3, checking the status of the PROG indicator.

When the indicators on the station communication module and processor show normal operation for both stations being tested, perform the procedures in “Testing the Local Station” and “Testing the Remote Station.”

Testing the Local Station

Test the commands from each station using these procedures. After completing these steps for a command, verify data transfer at the receiving station, as outlined in "Testing the Remote Location."

You can use the procedures outlined here for any phase of start-up and troubleshooting testing and at any time you add a command rung at a station processor. During initial testing, only two stations have power ON and are connected for these procedures. In addition, both station processors must be in test mode for initial start-up testing. During later phases of start-up testing, more than two stations may have power ON for these procedures.

There are two steps for testing of each command from a sending station:

1. Set the start bit.

2. Monitor the remote and local fault bits.

Each of these steps is described in a subsequent paragraph.

Setting the Start Bit

The most direct way to control the start bit for test purposes is to duplicate the input conditions of the user program rung which latches this bit ON. If it is possible to do this easily during testing, duplicate these conditions and proceed to Step 2. However, because this may not always be practical, you can use the set of optional test rungs, as described in "Test Rungs." You can use these rungs, entered at the end of the application program, to test each command individually.

Sample test rungs are shown in Figure 7.4. The following steps outline the procedures for programming these rungs for testing. With the industrial terminal connected, follow these procedures:

1. Display the end (of program) statement. The key sequence which displays this part of the program is as follows:

[SEARCH] [↓]

2. Turn the processor mode select switch to the PROGRAM or PROG position.

3. Enter the test rungs in the format of Figure 7.4, using the following addresses:

- Rung 1 — Enter the address of an unused input image table bit for the EXAMINE ON instruction. ("Test Rung" describes the reason for selection of this bit.)
- Rung 2 — In the LATCH instruction, enter the start bit address for the command being monitored.
- Rung 3 — Enter the address of the done bit for the EXAMINE ON instruction. Enter the address of the start bit for the LATCH instruction.

Check that the format of these test rungs resembles that shown in Figure 7.4.

With these rungs entered, you can now initiate a command. To do this, use the FORCE ON function of the industrial terminal. Perform the following steps:

4. Turn the processor mode select switch to the TEST position.

5. Position the cursor on the EXAMINE ON instruction of Rung 1.

6. On the programming terminal, press the key sequence for the FORCE ON function:

[FORCE ON] [INSERT]

The start bit is now forced ON. Under normal operation, this bit is being set and reset as the processor executes its scan, and the command is sent continuously.

With the start bit turned ON in this fashion, proceed to Step 2 of the next section. Once these checks have been completed, you can enter the next start/done bit addresses in the test rung for testing of the next command.

You can remove the FORCE ON function using the industrial terminal. On the industrial terminal, position the cursor on the forced instruction and press the following keys:

[FORCE ON] [REMOVE]

The FORCE ON function is also removed when you disconnect the industrial terminal from the 1771-KG module, or the 1771-KG module from the processor, or when the module receives an enter upload or enter download command.

Monitoring Remote/Local Fault Bits

This second step in command checking requires that you monitor the fault bits. Use the following steps to observe the instructions which examine fault bits:

1. Turn the processor mode select switch to TEST position.

2. Use the search functions of the industrial terminal to locate instructions which examine fault bits. (Search functions are described in "Industrial Terminal.")

3. Observe the ON/OFF status for each fault bit corresponding to the command.

Should the remote or local fault bit be ON for the command sent, some hardware-related fault or programming error can be suspected. Check the connections and equipment indicated in Figure 7.2.

After checking a command in this manner, check the receiving station, as described in "Testing the Remote Station." Then, check any other commands as necessary, using the applicable procedures in "Power-Up," "Testing the Local Station," and "Testing the Remote Station."

Testing the Remote Station

Check the receiving station with the sending station for one purpose—verification of data transfer. Although this procedure may be time-consuming, it is essential in initial start-up testing and for testing whenever you add a command at a station.

Upon receiving a command message, the function of the station communication module is to execute the command at its station processor and to format and transmit a reply message to the sending station. Proper execution of this procedure is indicated by the start/done and remote/local fault bits.

The checks performed in testing the sending station, therefore, can indicate and help isolate the source of a problem which prevents the command from being executed. The checks at the receiving station help to verify that program addressing and station number, switch selection is correct and that data is being sent to the intended station.

When a command has been initiated and tested at its sending station, perform the following procedures at the receiving station:

1. On the programming terminal at that receiving station, display the instructions for the application program which are affected by data transferred from the sending station.

To identify areas of the program which are affected by transferred data, refer to Form 5033, which is recommended programmer documentation for each station.

2. Check that the data at these locations matches the data in the proper locations at the sending station processor. For write or read commands, observe instructions which address transferred words in the program at the receiving station processor. (These instructions include GET, PUT, TON, TOF, RTO, CTU, and CTD.)

For bit write commands, check the ON/OFF status at instructions which examine those bits in the program.

Troubleshooting

For troubleshooting, you can use the same tools and procedures you used in start-up of a station containing a 1771-KG module. This section outlines procedures you can use in addition to the start-up procedures in "Start-Up Procedures."

Remote/Local Fault Indicator ON

As recommended in Chapter 5, you should control some indicators by the status of the remote and local fault bits at each station processor. Should this indicator go ON, connect and initialize a programming terminal and follow these procedures to isolate the source of the fault condition: (The steps of this procedure are outlined in Figure 7.7.)

1. Using the search functions, display the rung of program which controls the external fault indicator device. Determine whether a status bit indicates that either a remote or local fault bit is set ON.

2. Using the search functions, display the rung of program which examines either the remote or the local fault bits and controls a status bit or bits based on fault bit states.

3. Examine each element of the rung which may have been set ON, causing the fault indicator to be ON.

When you use the rungs recommended in Figure 5.18, EXAMINE OFF instructions address each fault bit. When these instructions are displayed on the program panel or industrial terminal, check the status indicators of the terminal carefully. Due to CRT delay time, the intensity of an EXAMINE OFF instruction may not change as rapidly as the actual ON/OFF status of the fault bit changes. Recall that programmed retries continuously turn the fault bit ON and OFF. Thus, it may take a few seconds for the programming terminal to show a change in fault bit status.

Should you have difficulty detecting the changing ON/OFF bit status for an individual fault bit, use the contact histogram feature to display changes in status.

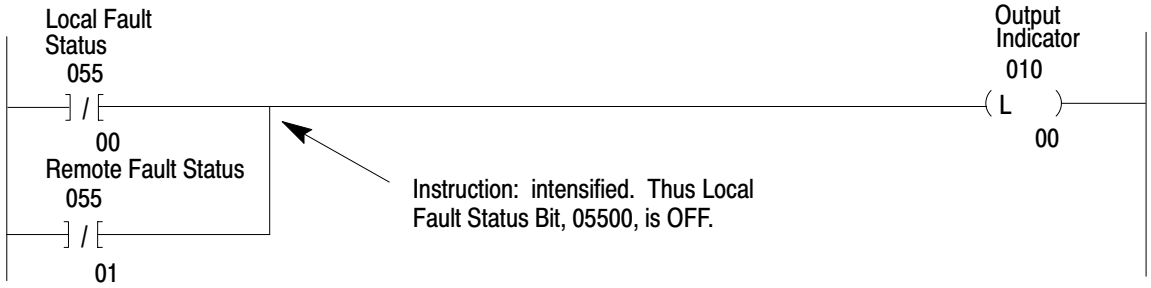
4. From the results of Step 3, determine the start bit address for the corresponding command.

The start bit has a strictly defined correspondence to a remote or local fault bit (Figure 5.9).

Figure 7.7
REMOTE/LOCAL FAULT Troubleshooting Example

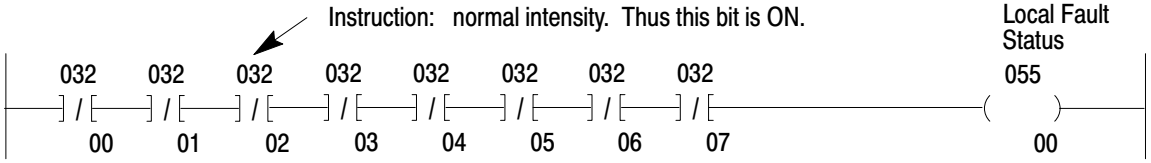
Step 1: Search Output Address

Key Sequence:
[SEARCH] [-(L)-] [0] [1] [0] [0] [0]
Display:



Step 2: Search Local Fault Bit Monitoring Rung

Key Sequence:
[SEARCH] [-( )-] [0] [5] [5] [0] [0]
Display:



If bit monitoring is available, display Remote/Local Fault Status Word to examine Fault Bit status.

Step 3: Locate any Fault On in this rung. Here Bit 03202 is ON.

Step 4: Identify Start Bit for this Fault Bit. For Local Fault Bit 03202, the Start Bit is Bit 03112.

Step 5: Search the Start Bit Examine instruction.

Key Sequence:
[SEARCH] [-][-] [0] [3] [1] [1] [2]
Display:

Step 6: Determine Area of Hardware-Related Problem

Here the Command Code identifies Station No. 023.
Conclusion: Local Fault between Local Station and Remote Station with Station No. 023.

11580

5. Use the search functions to locate the command rung which begins with the start bit, as determined in the previous step.

6. Examine the command code of this rung to determine the remote station to which the command was sent.

7. Determine the nature of the faulted condition and correct this condition.

   Use Figure 7.2 as a guide to tracing possible sources of a faulted condition.

8. Verify that you corrected the fault condition. Observe the fault indicator at the sending station.

Module Replacement

After you have made other troubleshooting checks, you may have to replace the module.

Removing the Module

1. Remove I/O chassis power and processor power.

2. Disconnect all cables connected to the module sockets.

3. A plastic latch on the top of the chassis holds the module in place. Pivot this latch upwards, out of the way of the module.

4. Lift up the plastic lever on the module to break its backplane connection.

5. Firmly grasp the sides of the module and pull it from the I/O chassis slot.

Installing the Replacement Module

1. Set module switches to match the switch settings of the module you are replacing. Replace the switch cover.

2. Insert the replacement module in to the I/O chassis. Snap down the latch on the top of the chassis and reconnect the cables to module sockets.

Module power-up is described in “Power-Up.”

Appendix A

Error Reporting

General

This appendix describes error reporting for communication with the module. You can get error information from three sources:

- The Error Word in the PC Ladder Diagram Program
- The STS Byte in a Reply Message
- The Diagnostic Counters in the 1771-KG Module

Table A.A lists the possible errors reported by the 1771-KG module. We can group these errors into five classes:

- Syntax Errors (1 thru 29) — Relate to errors in programming communication zone rungs. The module checks for these errors when you power it up and when you turn the processor mode select switch from PROGRAM to RUN. If it detects a syntax error, the module turns on its PROG indicator and shuts down. Note that the module cannot detect errors in the header rung of the communication zone; if the header rung is incorrect, the module assumes that no communication zone exists.
- Message Formatting Errors (30 thru 36) — Relate to errors that the module detects when it attempts to format a command message but before it actually tries to transmit the message. For some of these errors, the module turns on its PROG indicator and shuts down.
- Local Errors (90 thru 99) — Relate to errors that the local station module detects as it tries to transmit a command message. When one of these errors occurs, the module sets a local fault bit in PC memory.
- Remote Errors (80 thru 89) — Relate to errors that a remote station might report in its reply to a command message from the local station. The remote station puts the error code in the STS byte of its reply message, and the local station module copies this code into the error code storage word at the local processor. The local module also sets a remote fault bit in the local processor's memory.
- Reply Errors (50 thru 59) — Relate to errors that the local module detects in reply messages it receives from other stations. These include the timeout error (Number 37), which occurs when the local station receives no reply message at all for one of its command messages.

Error Word

The error word contains the last error code reported by the module whenever the local station transmits a command message to another station. The first column in Table A.A lists the possible error codes stored in this word. Note that a code value of zero indicates no error has occurred. Also note that some of the errors cause the module to set a local or a remote fault bit in the PC data table, as indicated in the third and fourth columns of the table.

The error word contains a four-digit BCD (binary coded decimal) number. The lower byte of this word (last two digits) contains the last error code reported by the 1771-KG module. The upper byte (first two digits) has one of two uses, depending on the error code:

- For Errors 01-26 — It represents the rung number where the error occurred within the communication zone of the ladder diagram program. The first rung in the communication zone is Rung 00, and subsequent rungs are numbered sequentially.
- For Errors 30 to 99 — It represents a count of the different errors reported by the module. This counter is incremented each time a different error occurs, not each time the same error occurs.

STS Byte

Whenever a remote station transmits a command message to the 1771-KG module, the module reports the status of that command in the STS byte of its corresponding reply message. If the command station is another PC, the status information is reported in the error word, as explained in "Error Word." If the command station is a computer, it is up to you to write computer application programs to monitor the STS byte in whatever way you see fit.

The second column of Table A.A lists the possible STS codes in hexadecimal notation. Note that an STS value of zero indicates no error has occurred.

Table A.A
Error Codes

Low Byte of Error Word (in Hex)	STS Byte of Reply Message (in Hex)	Fault Bits Local Remote	Meaning of Error Code
00	00		No error.
02			Processor fault.
03			Module ran out of internal memory space. This error can occur if the communication zone is too large. Try using fewer communication zone rungs.
04			First GET instruction incorrectly entered in memory access rung.
05			Invalid station number used in a memory access rung.
06			Second GET instruction incorrectly entered in memory access rung.
07			Third GET instruction incorrectly entered in memory access rung.
08			Starting address for memory access boundary is greater than ending address.
09			Branch-end missing in memory access rung.
10			Output instruction missing at end of memory access rung.
11			Invalid first element in a communication zone rung.
12			Invalid address for start bit. Start bits must be in the upper byte (Bits 10- 17) of a word.
13			Invalid second element in a command rung.
14			Invalid command code.
15			Invalid station number in command rung.
16			Invalid element in bit write command rung.
17			First GET instruction incorrectly entered in word read or write command rung.
18			Second GET instruction incorrectly entered in word read or write command rung.
19			Third GET instruction incorrectly entered in word read or write command rung.
20			Output instruction missing at end of command rung.
21			Invalid beginning of command rung, or memory access rung follows command rung.
22			Local/remote fault word is outside the data table (start/done word incorrectly chosen).
23			Transmission of unprotected commands is disabled by switch settings.

Low Byte of Error Word (in Hex)	STS Byte of Reply Message (in Hex)	Fault Bits Local Remote	Meaning of Error Code
26			Too many command rungs (255 is maximum).
27			Timeout setting too large. Valid timeout settings are 011 to 407. A setting of 010 disables the timeout.
28			Bad address in command rung.
29			Bad size in read/write command.
31			Communication problem on link between 1771-KG module and processor.
36			Processor turned OFF the start bit before the module set the done bit.
37			Transmission timed out because the local station did not receive a valid reply to its command. Frequent occurrence of this error might indicate that the timeout period is too short. A minimum timeout of 2 seconds is recommended.
38			Local processor entered the program mode.
50			Processor fault occurred at local station while module was processing a reply message.
51			Reply message contains invalid value in low byte of TNS field.
52			No communication zone in PC program.
53			Module received a reply message for which the start bit is OFF. This error occurs if the processor turns off the start bit or if the timeout period is too short. This error often occurs after Error 35 or 37.
55			Reply message received when module PROG indicator was ON or when processor was in program mode.
56			Reply message contains an invalid value in the high byte of the TNS field. This error occurs if the start bit is turned off and back on before the done bit is turned on.
57			Reply message is of incorrect size.
81	10	ON	Invalid command due to error in CMD byte, FNC byte, SIZE byte, or overall size of command message.
83	30	ON	Processor fault or disconnection at remote station.
84	40	ON	Remote station module could not communicate with its associated processor.
85	50	ON	Access denied at remote station. This error occurs if the local station tries to access a word outside the remote station's data table, or if the local station sends a protected command bit for which the remote station's memory access rungs do not give the local station any access.
86	60	ON	Execution of the command message is disabled by switch settings at the remote station.

Low Byte of Error Word (in Hex)	STS Byte of Reply Message (in Hex)	Fault Bits Local Remote	Meaning of Error Code
87	70	ON	The remote station processor is in program mode or is in the process of being downloaded from another station.
88	80	ON	Remote station is in error shutdown mode. This error can occur if the remote module's PROG indicator is ON.
89	90	ON	Remote station interface module is out of buffer space and cannot store the received command message.
8B	B0	ON	Remote station is in download mode; or error in download command; or operation not allowed in upload or download mode; or operation not allowed when not in download mode.
91	01	ON	There is a bad connection to module's RS-232-C port, or data-set-ready signal was not detected by that port.
92	02	ON	Remote station fails to respond. This error occurs when the module has exceeded its maximum number of times to retry transmitting a command message. This error does not necessarily mean that the remote station did not receive the command message. If the remote station did receive the command, the local module may still set a done bit when it receives the proper reply.

Appendix B

Diagnostic Counters

General

The module contains a set of internal counters that keep track of communication statistics—such as the number of messages sent and received, number of NAKs sent and received, number of ACKs sent and received, and timeout limits. You can use this information to monitor the general efficiency of communication with the module's RS-232-C port.

Table B.A lists the diagnostic counters for the 1771-KG module. To monitor these counters from your computer station:

- Send a diagnostic status command to the module to determine the starting address of the counter block. The seventh and eighth data bytes of the module's reply to the diagnostic status command will contain this starting address.
- Use a diagnostic read command to read the counter values, which start at the address determined in Step 1 above and extend over 52 consecutive bytes of module memory. For each counter, Table B.A lists the byte offset from the starting address of the block.

Table B.A
Diagnostic Counters

Byte Offset	Counters
00	16-bit count of the number of times the station attempted to send a message.
02	16-bit count of the number of messages that were successfully transmitted and ACKed.
04	16-bit count of the number of ACKs that were received.
06	Number of ACKs successfully passed from the receiver's separator to the transmitter.
07	Number of NAKs received.
08	Number of NAKs passed from the separator to the transmitter.
09	Number of timeouts while waiting for a response.
10	Number of ENQs sent.
11	Number of messages that could not be successfully sent.
12	Number of reply messages that could not be forwarded and were destroyed.
13	16-bit count of messages received.
15	16-bit count of ACKs sent.

Byte Offset	Counters
17	Number of NAKs sent.
18	Number of ENQs received.
19	Number of transmissions received and ACKed. A retransmission is a message which has a transaction word, command, and source that match the previous message.
20	Number of STX (full-duplex mode) or SOH (half-duplex mode) received. This is in effect a count of the number of messages that were started.
21	Number of messages, characters, or message fragments that were ignored.
22	Number of messages that were aborted by receipt of a DLE ENQ.
23	Number of messages that were aborted by the receipt of an unexpected control code other than DLE ENQ.
24	Number of times there was no buffer ready for the next message when a previous message was ACKed.
25	Number of times there was no buffer for a received message (the message was NAKed).
26	Number of broadcast messages received.
27	Number of broadcast messages that were successfully received.
28	Number of messages seen that were not for this station.
30	Number of DLE EOTs sent.
31	Number of calls received.
32	Number of times that phone was hung up by the 1771-KG module.
33	Number of times that DCD was lost.
34	Number of times that the phone was hung up because of a DCD timeout.
35	Number of messages routed to RS-232 port.
36	Number of commands routed to command executor.
37	Number of replies routed to reply processor.
38	Number of messages sent to self.
39	Number of routing errors on inbound messages.
40	Number of routing errors on outbound messages.
41	Number of messages with incorrect network address.
42	16-bit count of messages sent by command initiator.
44	16-bit count of commands received by command executor.
46	16-bit count of replies sent by command executor.
48	16-bit count of replies received by command initiator.
50	Number of breaks sent to industrial terminal.
51	Number of resyncs sent to PC.

Appendix C

Data Manipulation

General

This appendix explains two areas of special concern when you are transmitting messages between computers and PCs:

- Data Encoding
- Addressing Formats

The information contained in this appendix gives some application details that relate to the data and address fields of the message formats.

Data Encoding

In general, PCs store binary data (1s and 0s) in 16-bit groups called words. If you are looking at this data from a computer, however, you may interpret it in a number of different ways, depending on your application needs.

Number Systems

You may use any one of the following number systems to represent data in your computer application programs:

- Binary
- Binary Coded Decimal
- Decimal
- Hexadecimal
- Octal

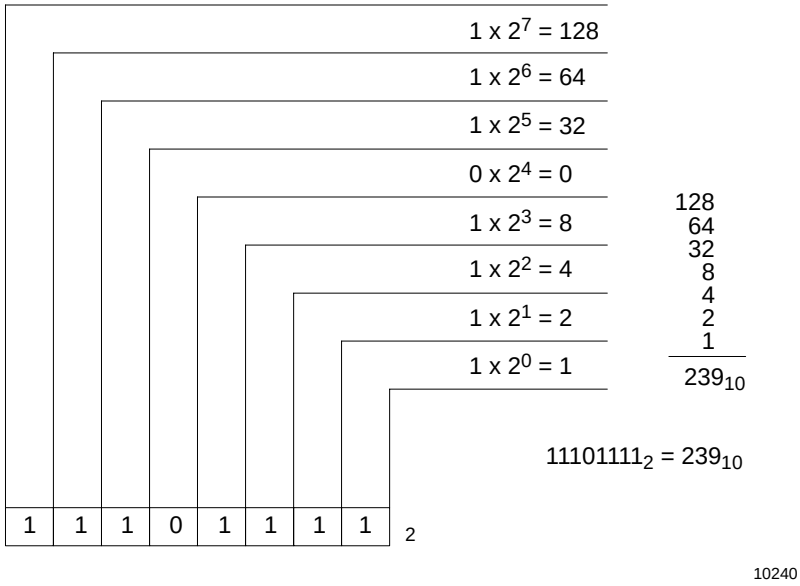
It is up to you to design your computer application programs to make any necessary conversions from one number system to another. Once you have selected the number system that is best for your applications, you should try to use only that one system and convert all data values to that base to avoid confusion.

Binary

The binary number system is probably the simplest to use for computer and PC applications because it is the most natural way to represent data bits. However, since the binary system uses only the digits 0 and 1, it is cumbersome to show values in binary format.

Each digit in a binary number has a certain place value expressed as a power of 2. You can calculate the decimal equivalent of a binary number by multiplying each binary digit by its corresponding place value and then adding the results of the multiplications. Figure C.1 shows the binary representation of the decimal number 239.

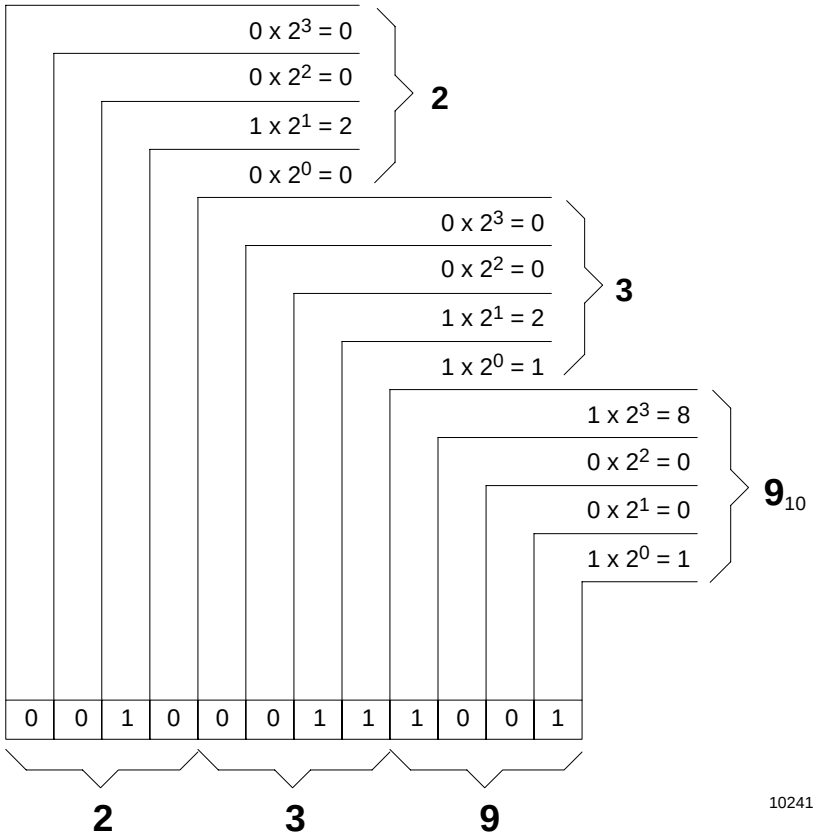
Figure C.1
Binary Numbers



Binary Coded Decimal

Quite often, PC data is represented in binary coded decimal (BCD) form. In this system, each group of four bits in a PC word represents one decimal number between 0 and 9. In this way, each 16-bit word can represent a BCD value between 0 and 9,999. Figure C.2 shows the BCD representation of the decimal number 239.

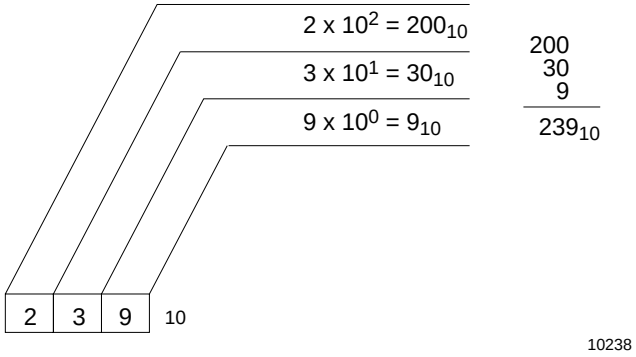
Figure C.2
Binary Coded Decimal Numbers



Decimal

The decimal number system is probably the easiest for us to use because it is the most familiar to us. It uses the common digits 0 through 9, and each digit has a place value that is a power of 10 (Figure C.3). However, despite the convenience of decimal numbers, it is often easier to convert binary data to a number system other than decimal.

Figure C.3
Decimal Numbers



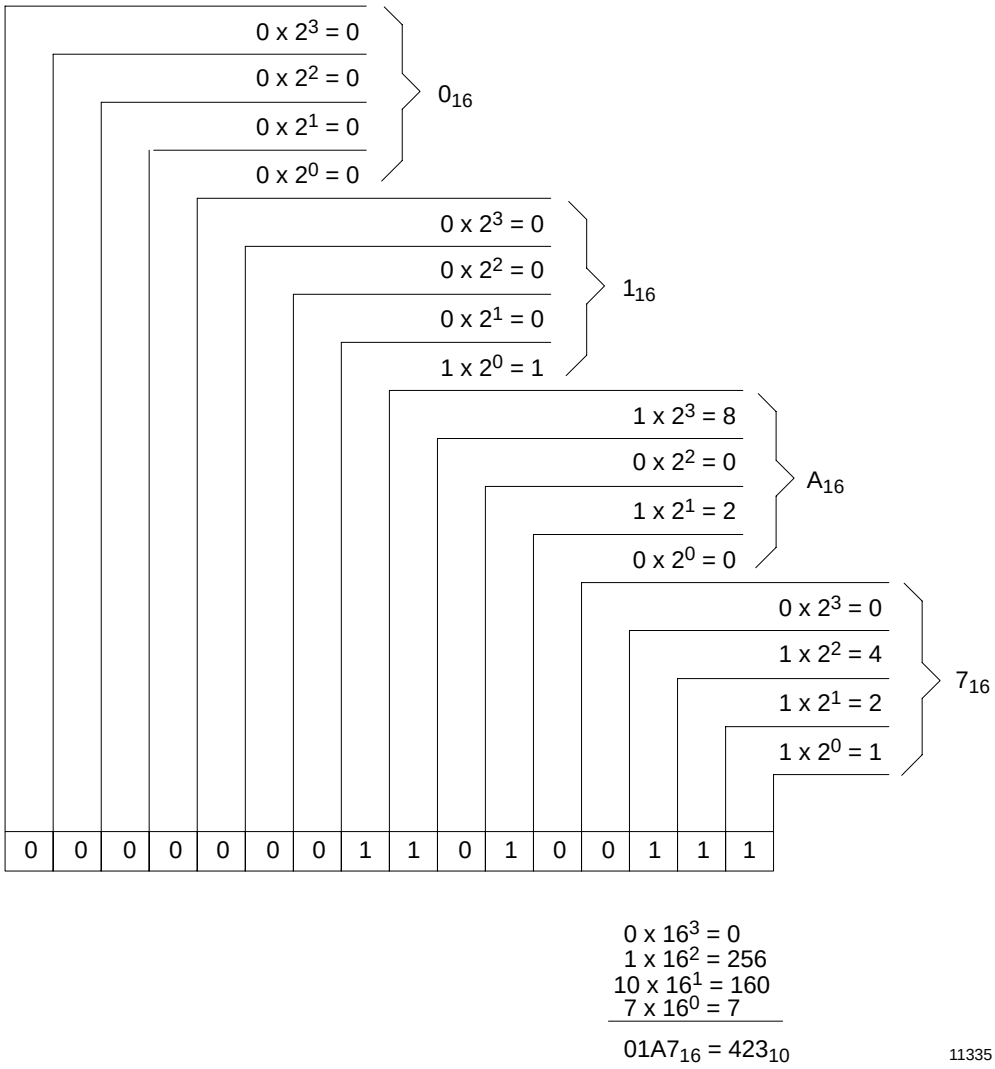
Hexadecimal

The hexadecimal number system is the most compact way to represent binary data, and it allows for the easiest conversion to and from binary. This system uses a number set of 16 digits: the numbers 0 through 9 and the letters A through F (where the letters A through F are equivalent to the decimal numbers 10 through 15, respectively).

Each group of four data bits represents one hexadecimal digit between 0 and F. In this way, each 16-bit data word can have a hexadecimal value between 0 and FFFF.

Each digit of a hexadecimal number has a place value that is a multiple of 16. To convert a hexadecimal number to its decimal equivalent, multiply each hexadecimal digit by its corresponding place value and add the results of the multiplications. Figure C.4 shows the hexadecimal representation of the decimal number 423.

Figure C.4
Hexadecimal Numbers



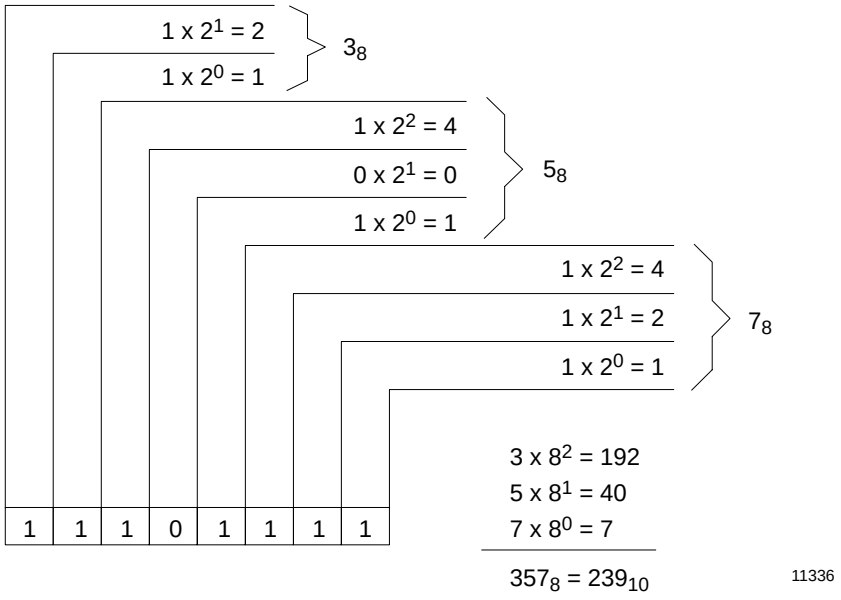
Octal

The octal number system is also a relatively easy way to represent binary data. This system uses the eight digits 0 through 7.

Each group of three data bits represents one octal digit between 0 and 7. This presents a slight conversion problem because bytes and words usually contain an even number of bits. Thus, an eight-bit byte can have an octal value between 0 and 377, while a 16-bit word can have an octal value between 0 and 177777.

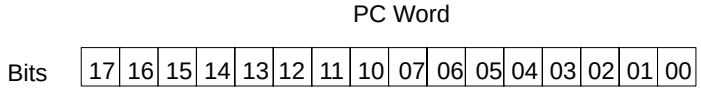
Each digit of an octal number has a place value that is a multiple of eight. To convert from octal to decimal, multiply each octal digit by its corresponding place value and add the results of the multiplications. Figure C.5 shows the octal representation of the decimal number 239.

Figure C.5
Octal Numbers

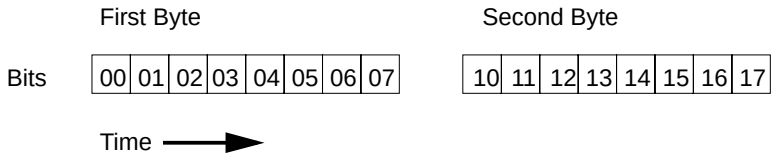


Order of Transmission

PCs store data in 16-bit (two-byte) words. The bits in these words are numbered (addressed) 0 through 17 octal, going from right to left within a word, as follows:



In PC memory, the words are arranged as shown above. However, when the 1771-KG module transmits data over its RS-232-C link, it transmits one byte at a time. The 1771-KG module transmits the low byte (Bits 00 through 07) of a word before the high byte (Bits 10 through 17) of the same word. Also the UART transmits the low bit first within a byte. Thus, when a PC word is traveling over the RS-232-C link, it will look like this:



This does not present a problem at PC stations on the link because PCs store and retrieve their data in the same order of low byte first. But you must consider this order when you write a computer application program.

Three factors that can influence the ability of your computer to handle PC data are:

- The size of words in your computer's memory.
- The left-to-right or right-to-left ordering of bits within a word in your computer's memory.
- Whether the computer considers the low order byte of a word to have an even or an odd address.

If your computer uses something other than two-byte, 16-bit words, you should design your application programs to make the proper conversions from PC word addresses to computer word addresses.

Figure C.6.A shows a 16-bit word in PC memory.

Figure C.6.B shows a 16-bit computer word with right-to-left byte and bit order (as in DEC PDP-11/34 or VAX 11/780). It also represents a 16-bit word in an eight-bit processor (such as Zilog Z-80 or Intel 8086 micro-processor). If your computer has this type of word order, the conversion is straightforward.

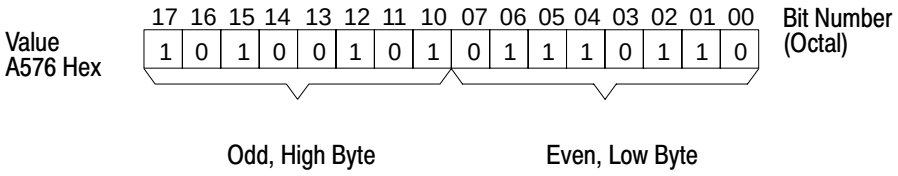
Figure C.6.C shows a 16-bit computer word with left-to-right byte and bit order (as in IBM Series 1). If your computer has this type of word order, the conversion is more complex. You will have to swap bytes onto and out of buffers.

Figure C.6.D shows a 16-bit computer word with left-to-right byte order and right-to-left bit order (as in Zilog Z 8000 or Motorola 68000 micro-processors). If your computer has this type of word order, your communication driver must handle the task of byte-swapping as it loads data into a buffer. Successive bytes received from the PC must be stored in the following order:

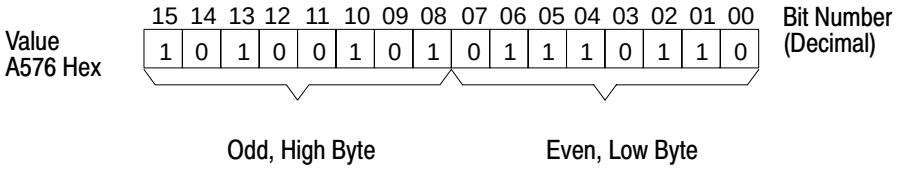
1,0,3,2,5,4,7,6,9,8,...

Figure C.6
Result of Transmitting Low Byte First

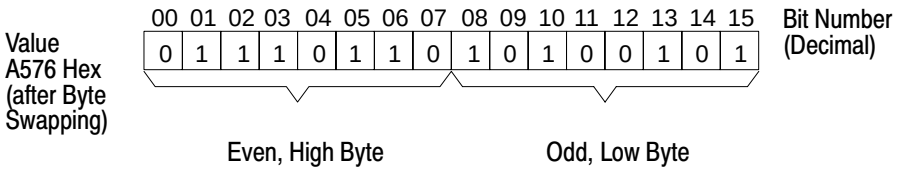
A. PC Word



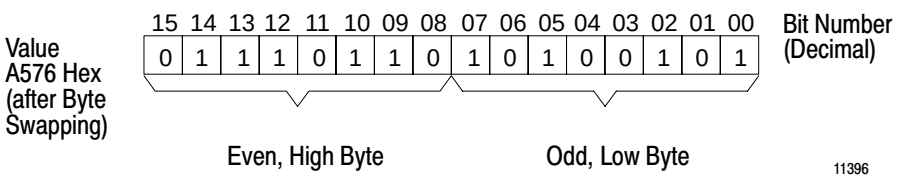
B. 16-Bit Computer Word with Right-to-Left Byte and Bit Order



C. 16-Bit Computer Word with Left-to-Right Byte and Bit Order



D. 16-Bit Computer Word with Left-to-Right Byte Order and Right-to-Left Bit Order



Addressing

There are two types of addressing that a computer can use in command messages it transmits to PLC-2 family stations:

- Logical
- Physical

Logical Addressing

Logical addressing refers to the type of addressing that a PC uses in its ladder diagram program to access its own data table memory. This is the same type of addressing you would use in non-physical commands (that is, in commands that access only PC data table memory).

PLC-2 family controllers access their data tables by using an octal word address. In PLC-2 command messages, this type of logical word address must be represented as an equivalent byte address. This byte address appears in the two-byte field labeled ADDR in the message block formats.

To encode a logical PLC-2 address, first convert the octal word address to whatever number system you are using in your computer application programs. Next, double this converted word address to get the corresponding byte address. Place the result in the ADDR field, low byte first.

For example, to address Word 020, you would first convert the octal value 20 to the desired base. In this example, let's use hexadecimal values. Octal 20 is 10 hex. Doubling this value gives 20 hex for the byte address. You would then code the value 0020 hex in the ADDR field of the message, low byte first. In binary format, ADDR would look like:

NOTE: PLC-2-family controllers use this same logical addressing format when they transmit command messages to another station. If you plan to transmit a command message to your computer from one of these PCs, you should set up a buffer space in your computer to simulate PC memory. You would then have to write computer application programs to accept and execute commands from the PC stations and to translate the ADDR value into the corresponding address in the simulated PC memory.

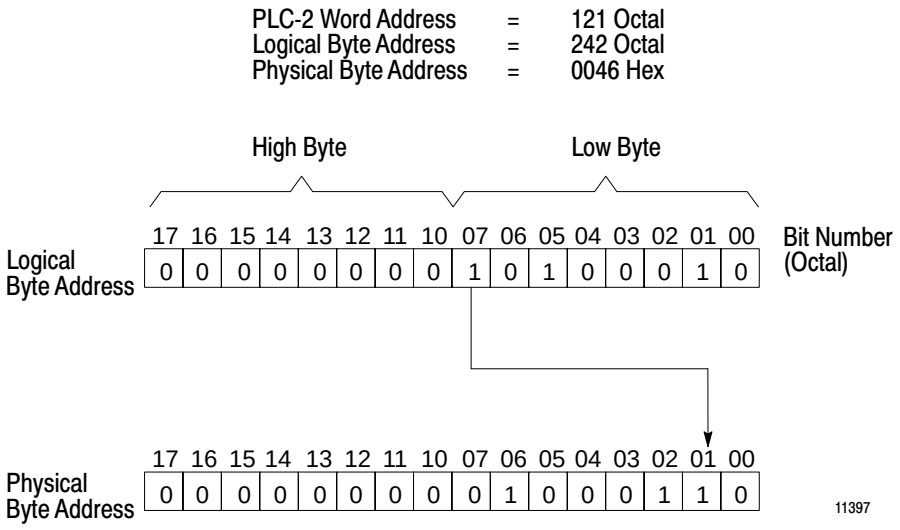
Physical Addressing

Physical addressing is the type of addressing a computer would use to send a physical read/write command to a PC station. In particular, you would use physical addressing to upload or download PC memory. The recommended procedure for doing this is to use a series of physical read or write commands that begin at Physical Address 0000 and proceed sequentially to the end of PC memory.

PLC-2-family controllers use physical addresses that are directly related to logical addresses. To convert a given logical address to its corresponding physical address, move Bit 7 of the logical address to Bit Position 1 and shift Bits 1 through 6 to the left one position. Figure C.7 shows the conversion process for Logical Word Address 121. Remember that the logical PLC-2 address is a byte address, so the physical address will also be a byte address.

To send a physical read or write command to a PLC-2 station, put the PLC-2 physical address in the ADDR field of the command message format. Be sure to encode the low byte of the physical address as the first byte in the ADDR field.

Figure C.7
Converting PLC-2 Logical to Physical Address



Appendix D

Detailed Flowcharts

General

In Chapter 6 are flowcharts which give a simplified view of an example of software logic for implementing full-duplex protocol. In this appendix, we present flowcharts which give a detailed view of an example of software logic for implementing full-duplex protocol.

The flowcharts in this appendix are based on the use of CRC rather than BCC for error checking.

Figure D.1 shows a data flow diagram. Figure D.2 shows a logic flowchart of a transmitter routine. Figure D.3 shows a logic flowchart of a receiver routine. Each subroutine appears in a subsequent figure.

Figure D.1
Data Flow Diagram for Full-Duplex Protocol

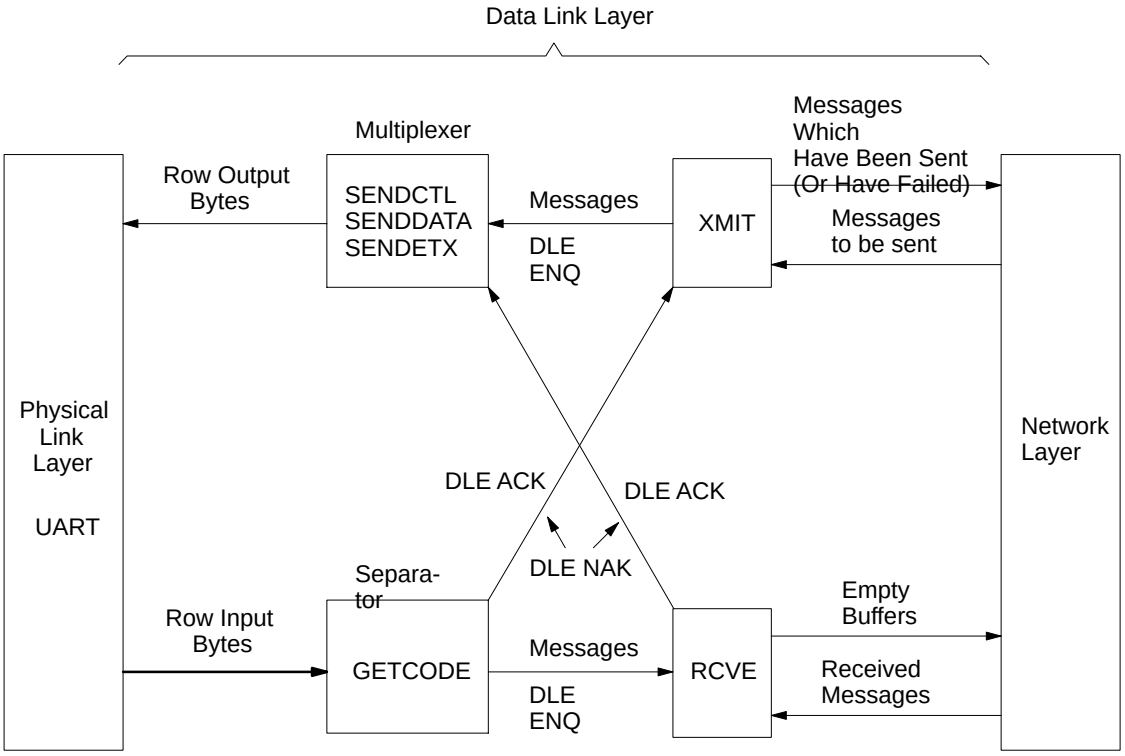


Figure D.2
Transmitter Routine for Full-Duplex Protocol

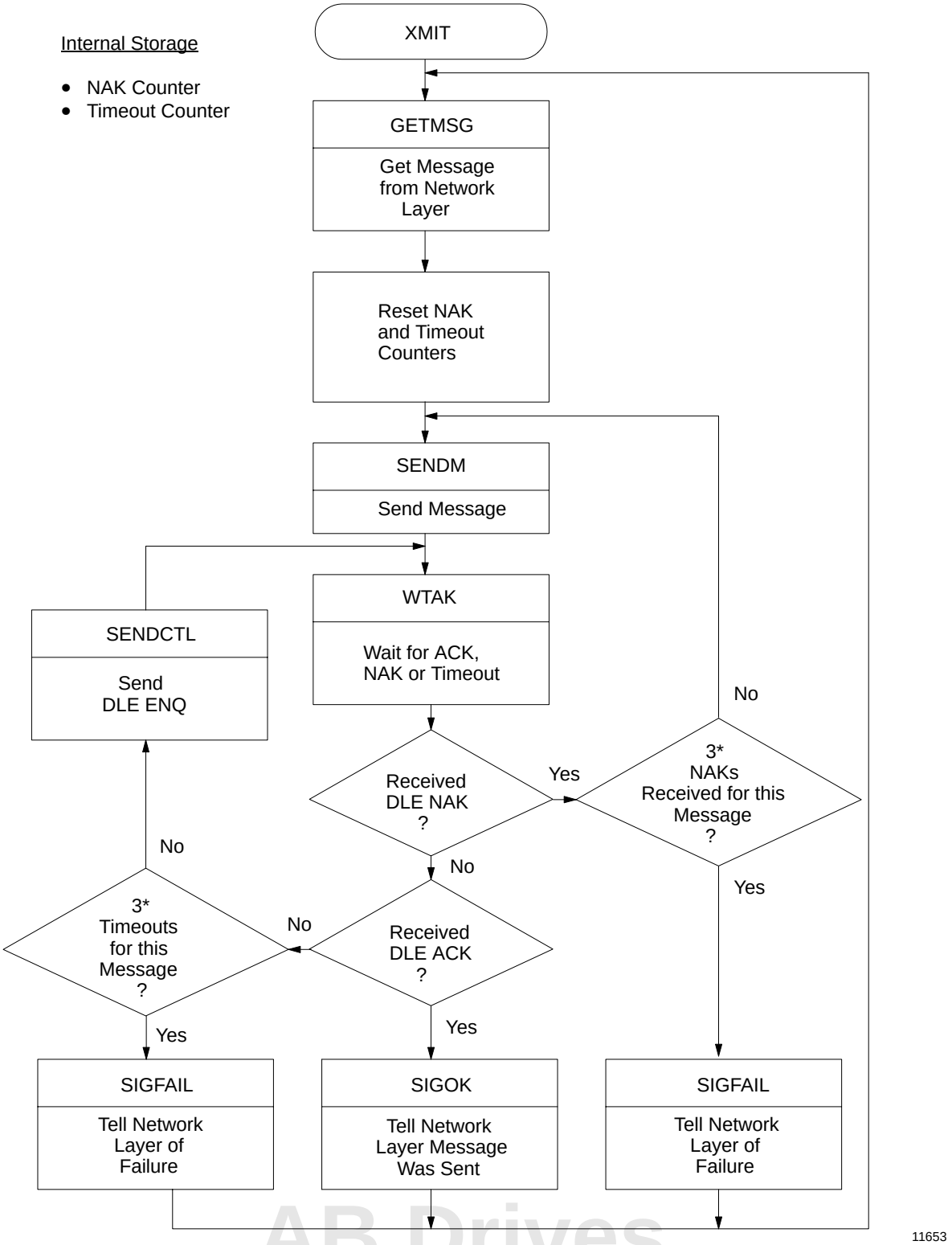


Figure D.3
Receiver Routine for Full-Duplex Protocol

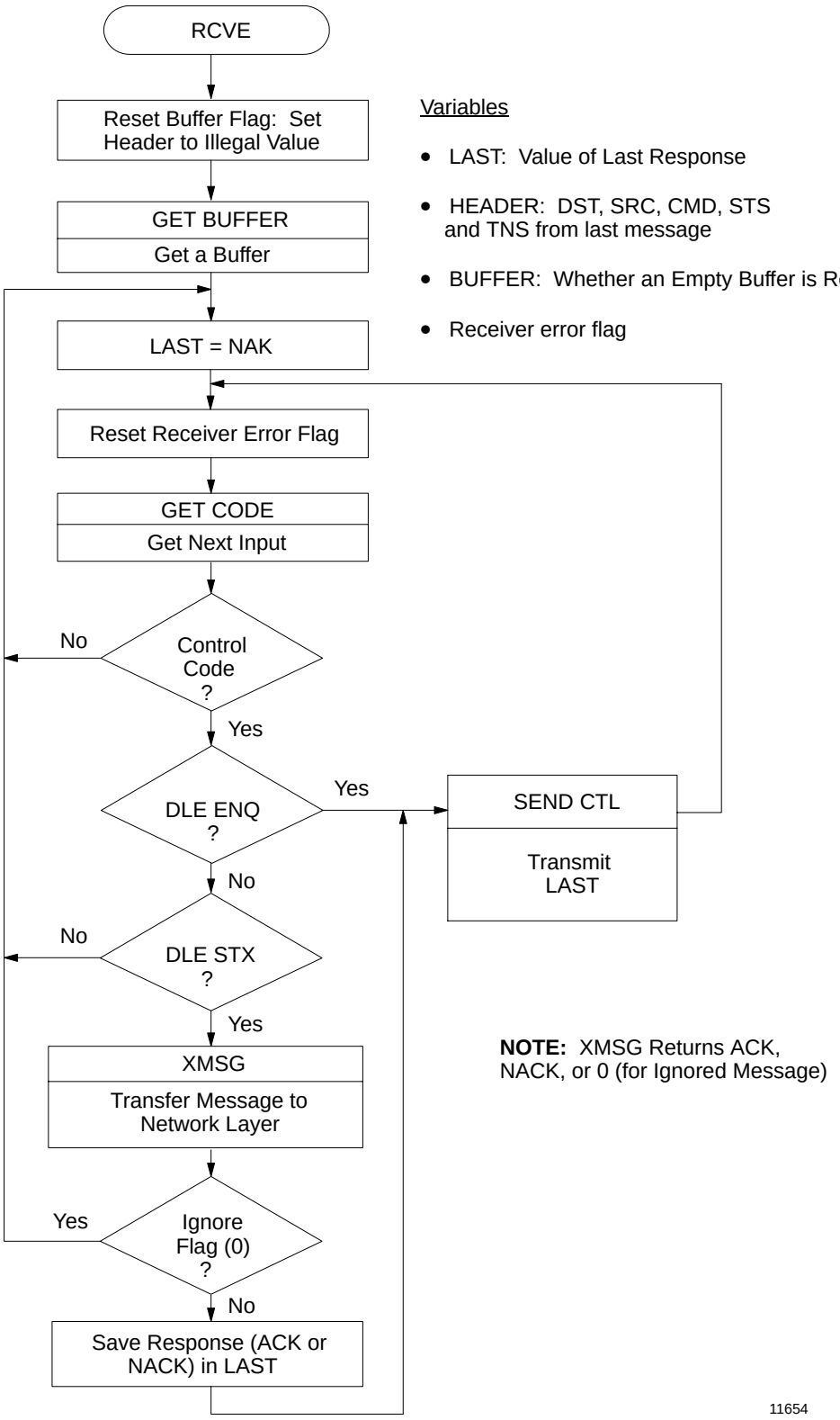


Figure D.4
SENDM Subroutine

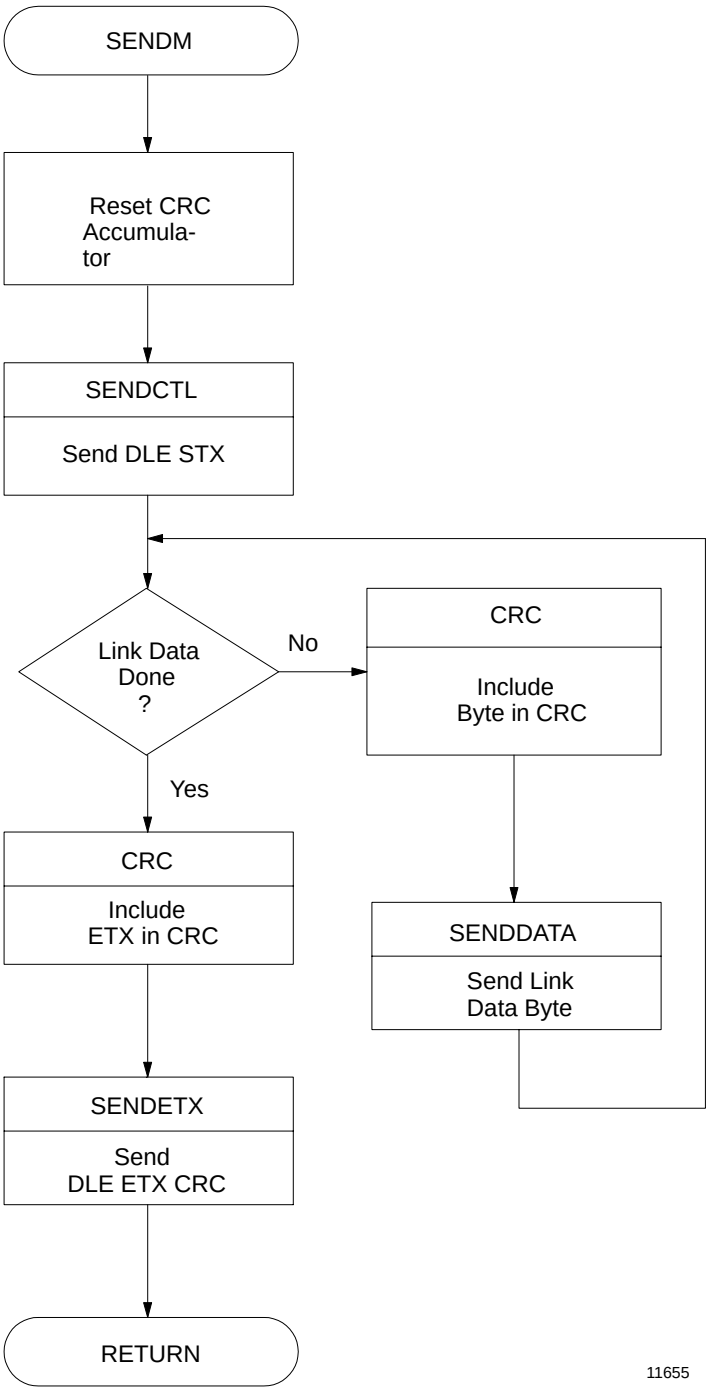


Figure D.5
WTAK Subroutine

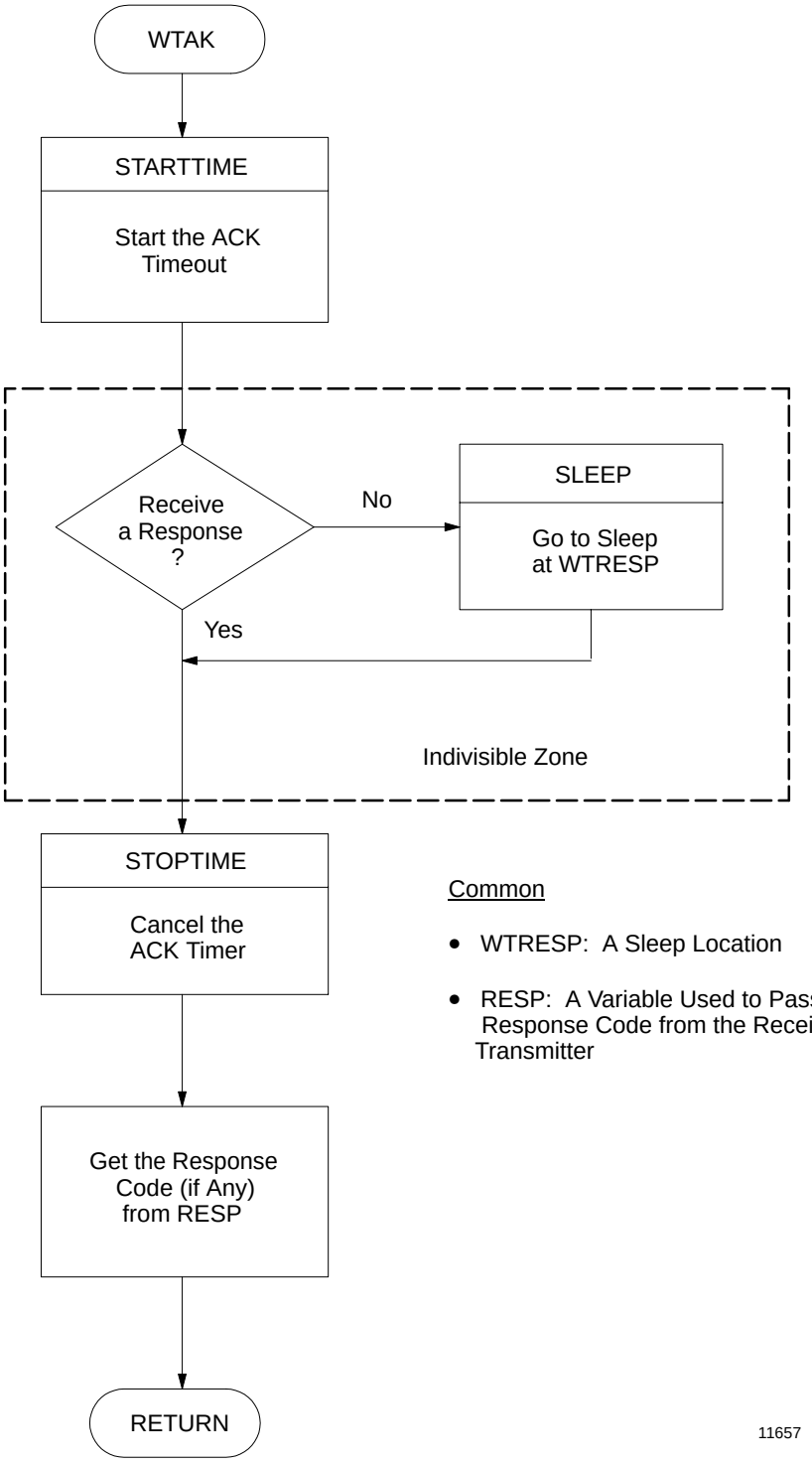


Figure D.6
STARTTIME Subroutine

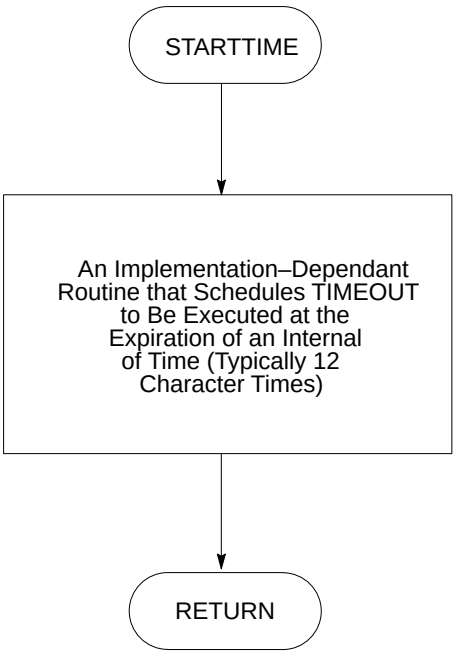


Figure D.7
STOPTIME Subroutine

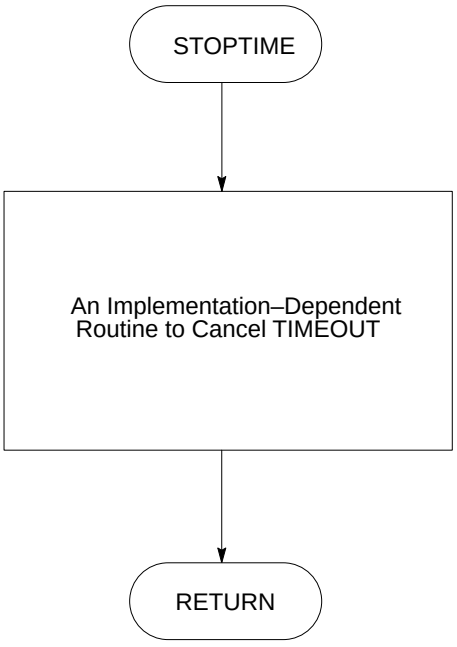


Figure D.8
TIMEOUT Subroutine

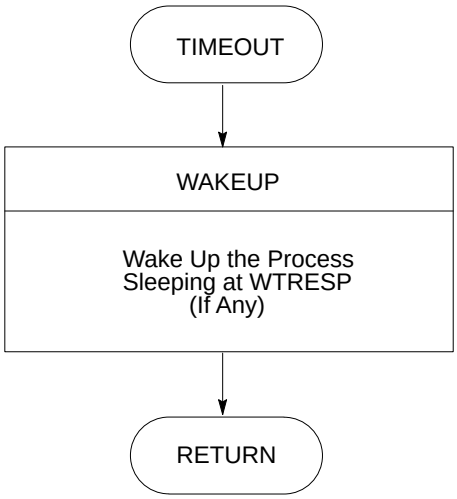


Figure D.9
GETMSG Subroutine

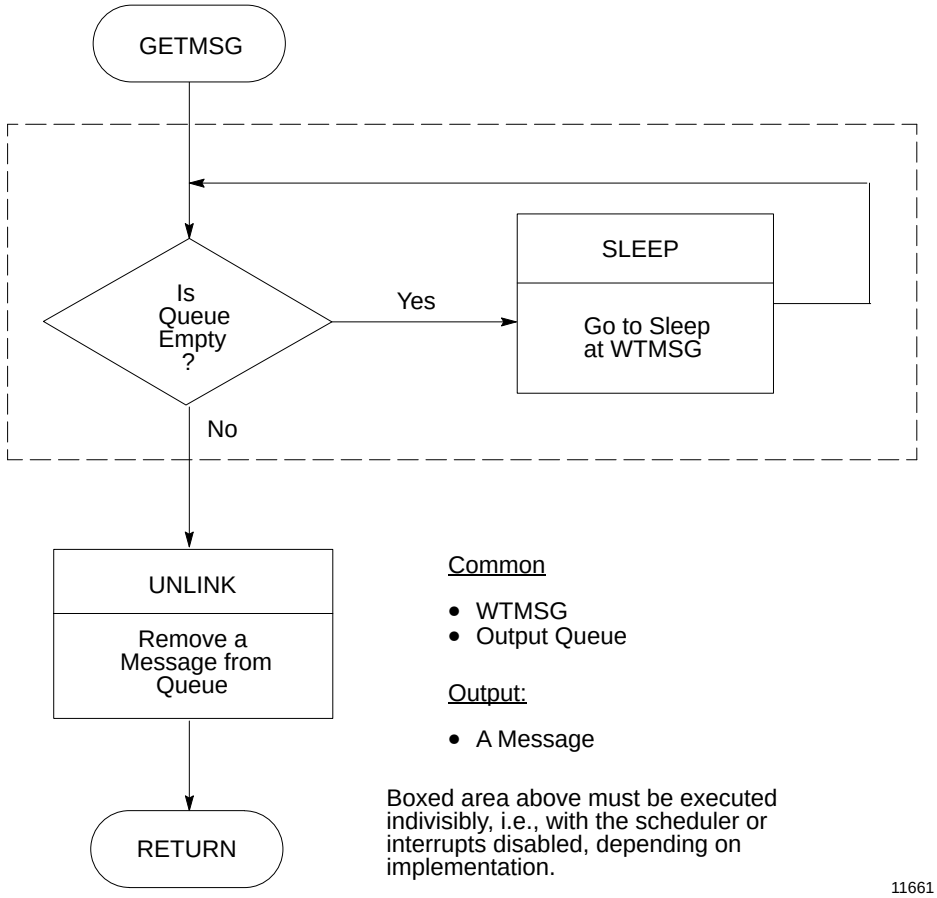
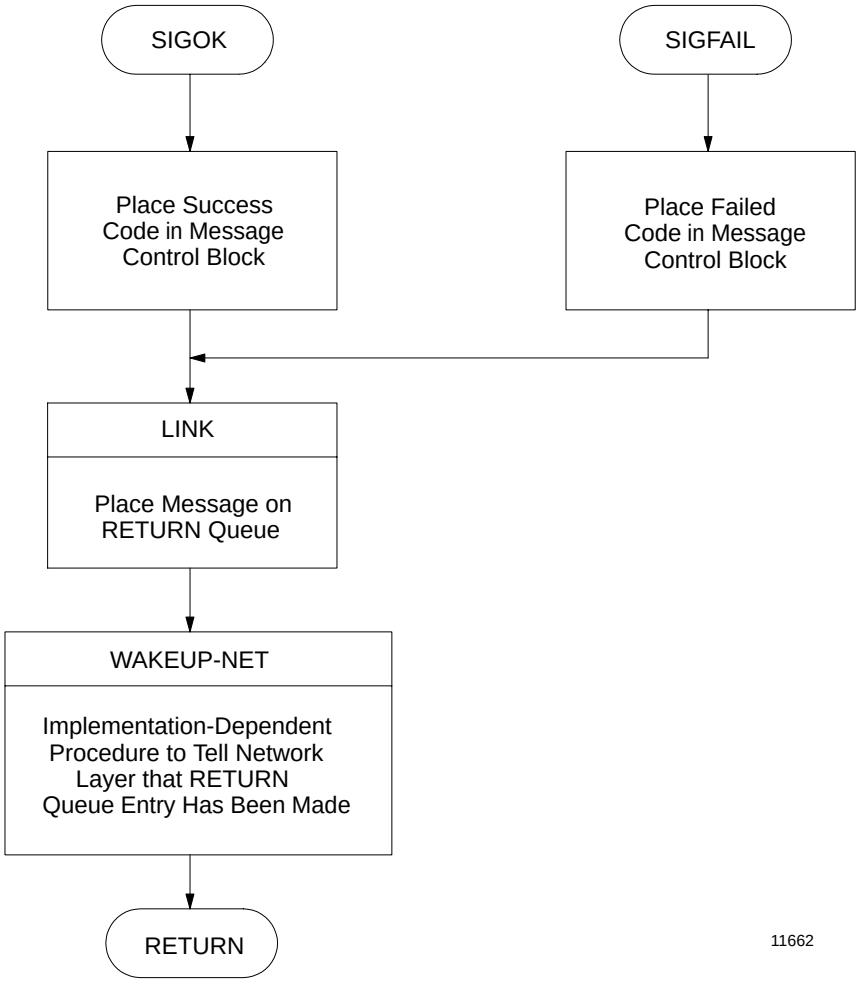


Figure D.10
SIGOK/SIGFAIL Subroutine



UART Sharing

Figure D.11 shows the transmitter (XMIT) and receiver (RCVE) routines sharing the transmit side of the UART. XMIT transmits messages and inquiries. RCVE transmits responses. Therefore, each must wait until the other is through before it can transmit through the UART. The TXALLOC and TXFREE subroutines work together to ensure that XMIT and RCVE do not try to use the UART at the same time. TXALLOC and TXFREE are called in the SENDCTRL, SENDETX, and SENDDATA subroutines (Figure D.12, Figure D.13, and Figure D.14.)

Figure D.11
Sharing the Transmit Side of the UART

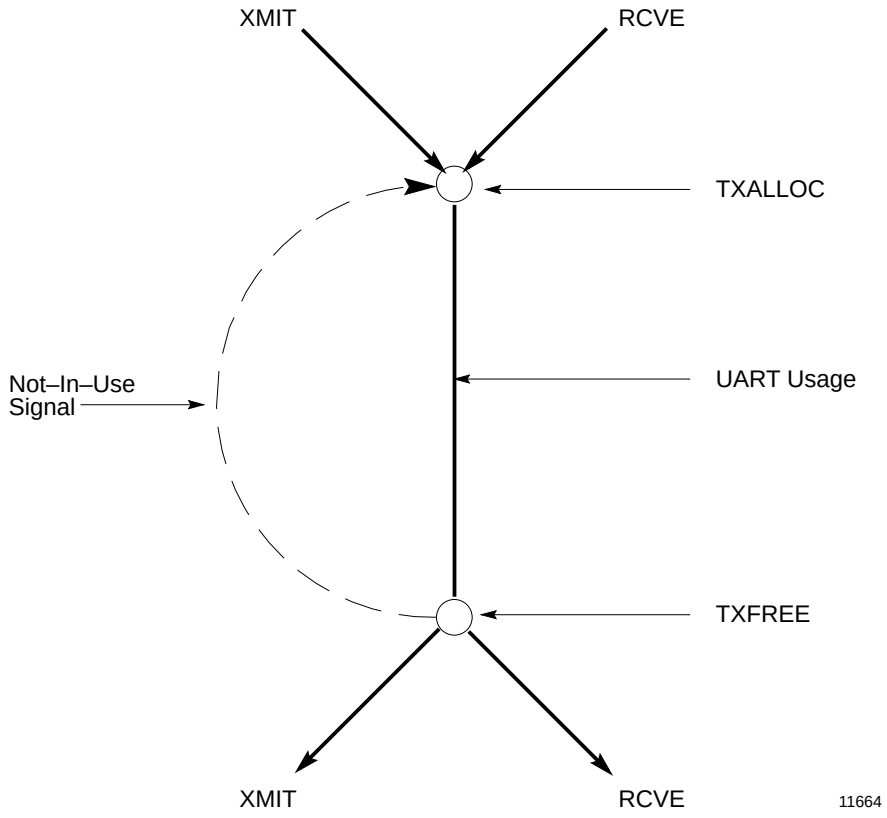
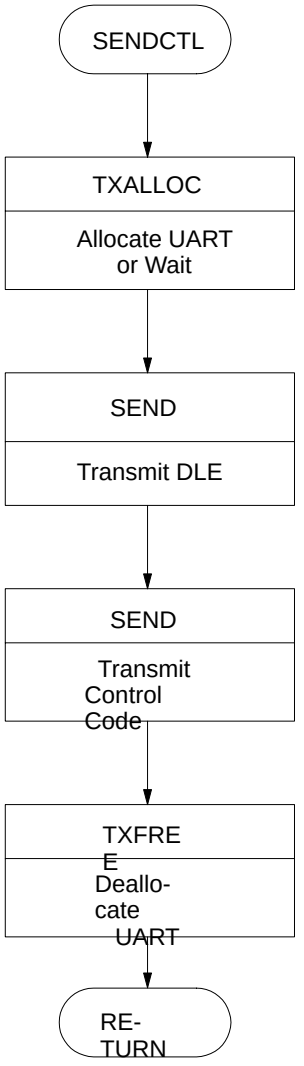


Figure D.12
SENDCTL Subroutine

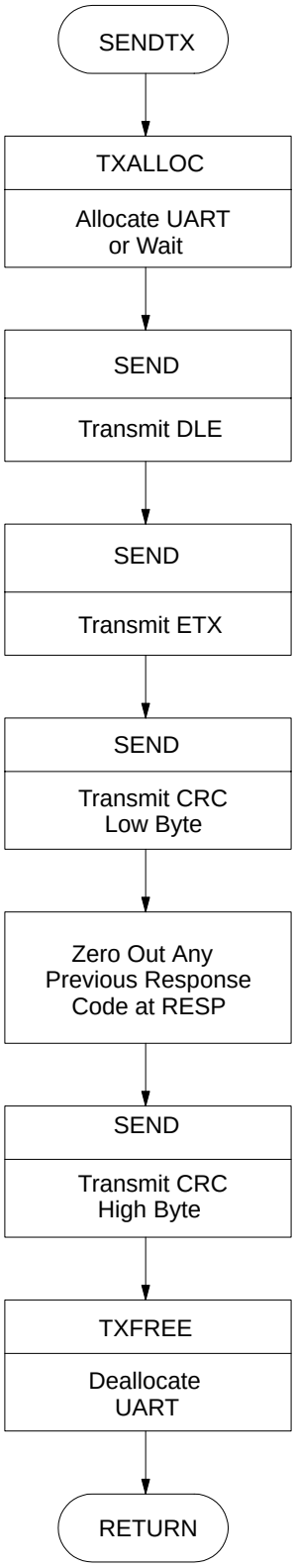


Input:

- Control Code

11663

Figure D.13
SENDETX Subroutine



Input:

- CRC

Common

- RESP: The Response Code Variable

11665

Figure D.14
SENDDATA Subroutine

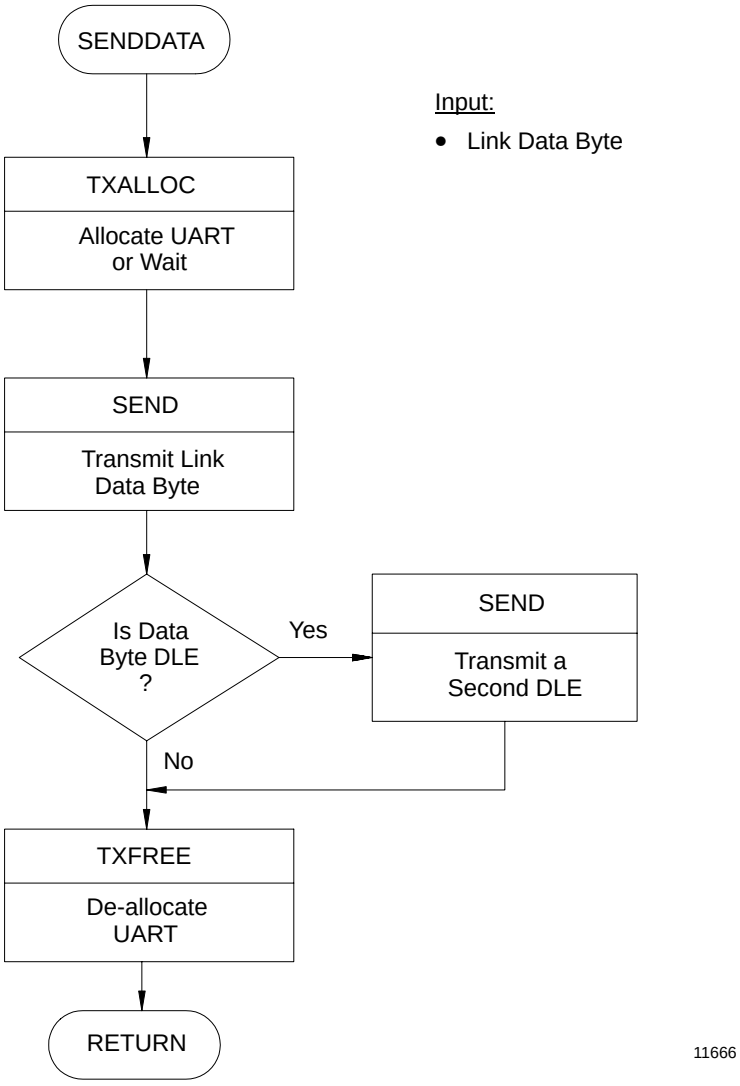


Figure D.15
TXALLOC Subroutine

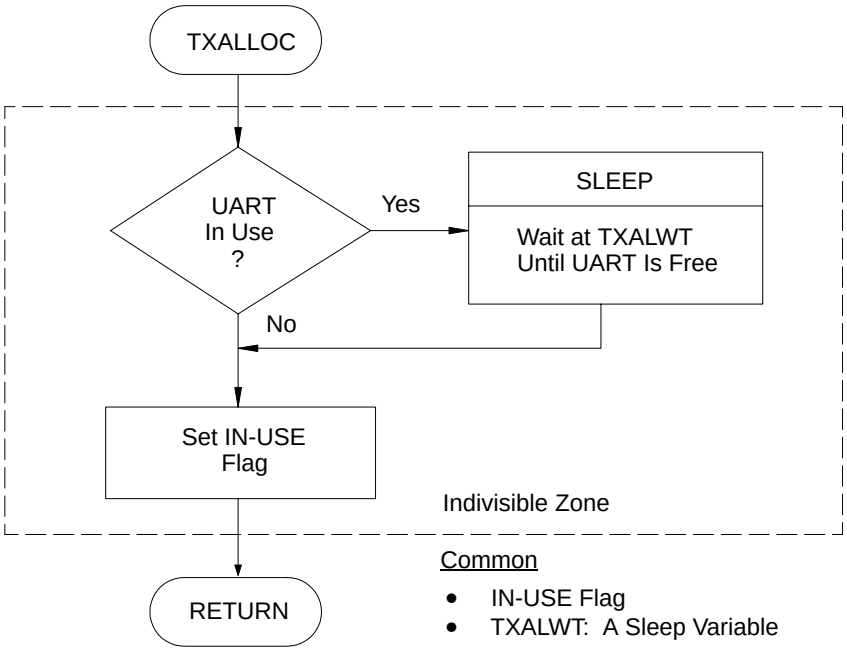


Figure D.16
TXFREE Subroutine

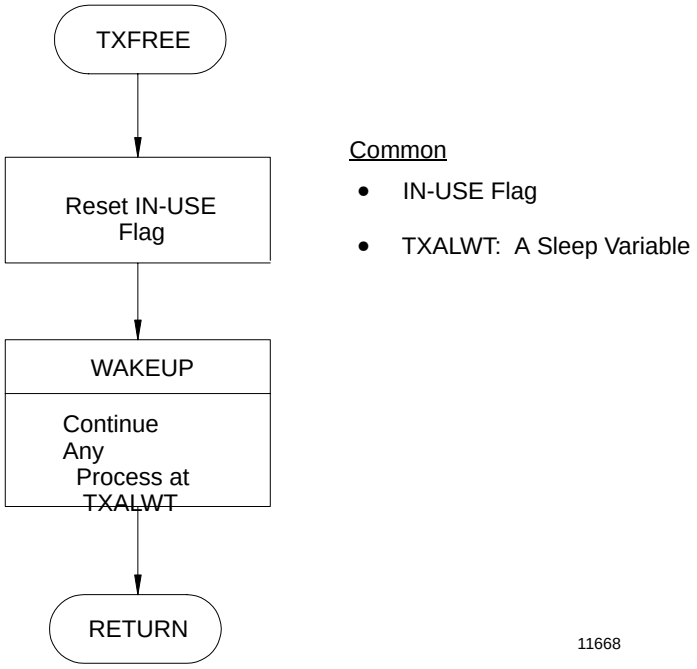
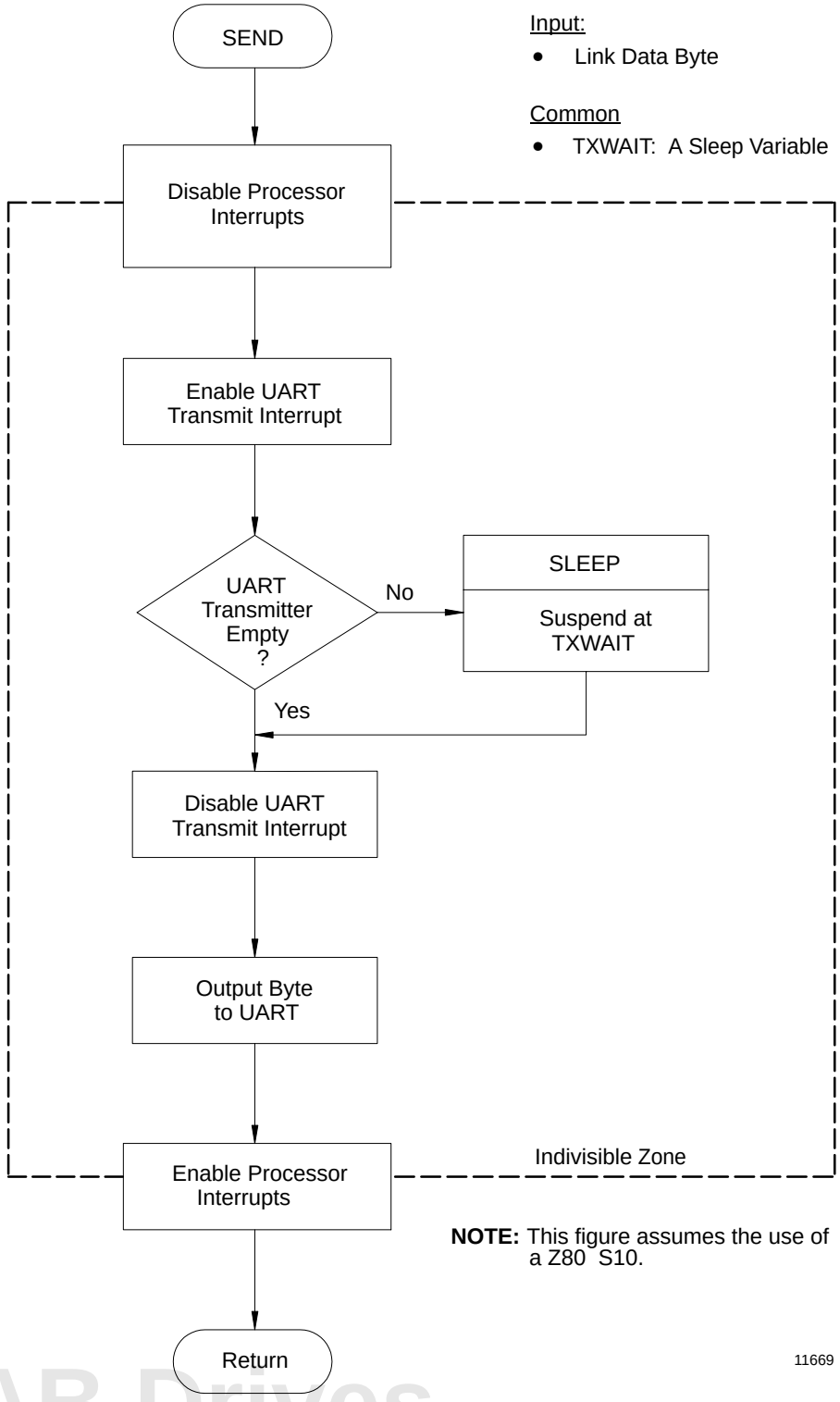
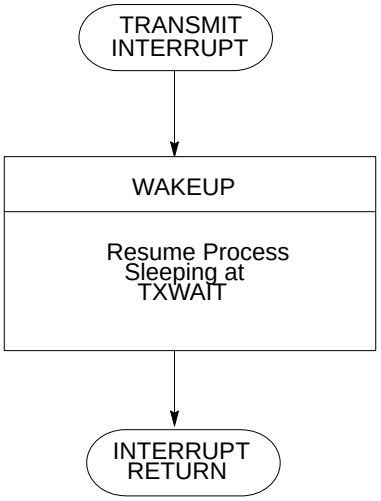


Figure D.17
SEND Subroutine



NOTE: This figure assumes the use of a Z80 S10.

Figure D.18
TRANSMIT INTERRUPT Subroutine

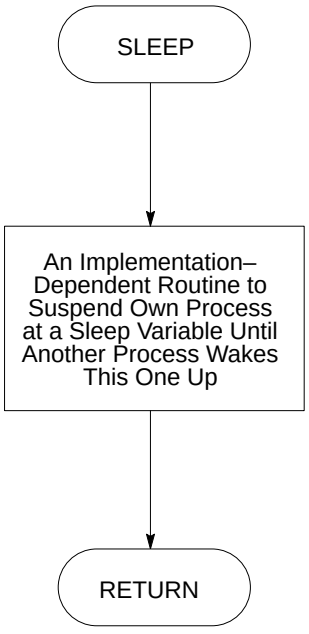


NOTE: This figure assumes the use of a Z80 S10.

NOTE: UART transmit interrupt must be enabled and disabled without affecting the current state of the **receive** and **status interrupt** enable flags.

11670

Figure D.19
SLEEP and WAKEUP Subroutines

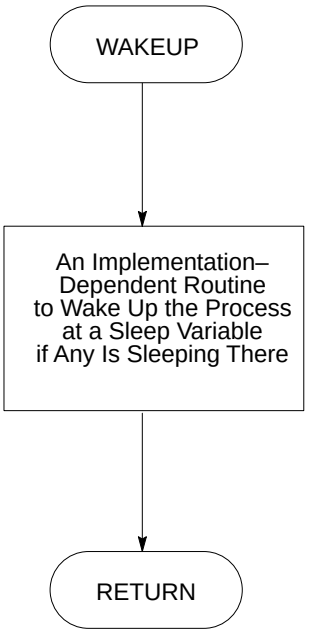


Input:

- The Address of a Sleep Variable.

Sleep Variables

- Typically an address of a stack or a process or context save area.
- A process can suspend itself and place its address in a sleep variable.
- Subsequently another process can wake up the sleeping process by referring to sleep variable. When no process is sleeping at a sleep variable, a WAKEUP has no effect.



Input:

- The Address of a Sleep Variable.

11671

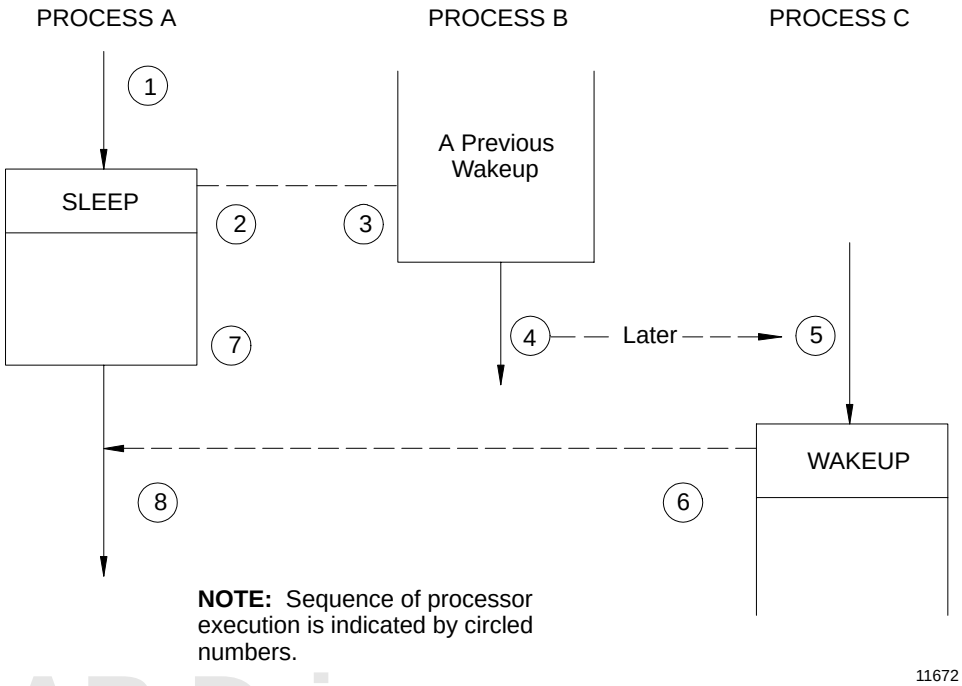
SLEEP and WAKEUP

The SLEEP and WAKEUP subroutines are always used in connection with some type of indivisible process interlock. Indivisibility is achieved on many processors (as on the Z80) by disabling processor interrupts. For this reason, SLEEP and WAKEUP assume that interrupts are OFF when they are called. They will always return with interrupts OFF.

When one process calls SLEEP, the result is a return from a call to WAKEUP by another process. When a process calls WAKEUP, the result is a return from a call to SLEEP by another process. An interrupt subroutine that calls WAKEUP is viewed as a subroutine of the interrupted process.

Figure D.20 shows an example of interaction between SLEEP and WAKEUP. In this example, Process B woke up Process A some time ago. Now at 1, when A goes to sleep, actual execution resumes after the wakeup call in B at 3 and 4. Some time later, Process C (at an interrupt, for example) calls WAKEUP at 5. Execution flow proceeds to the instructions at 8 following the call to SLEEP in Process A. The next time A calls SLEEP, the WAKEUP call in C will terminate.

Figure D.20
SLEEP and WAKEUP Interaction



This is not the only possible implementation of SLEEP and WAKEUP. A second alternative implementation would allow a process to call WAKEUP without losing immediate control of the processor. If B wakes up A, context switching would be deferred until B itself has executed a SLEEP.

A third alternative would cause a context switch if a process performed a WAKEUP on a higher priority process. If a WAKEUP had been performed on a lower priority process, the context switch would be deferred until the first process has gone to SLEEP.

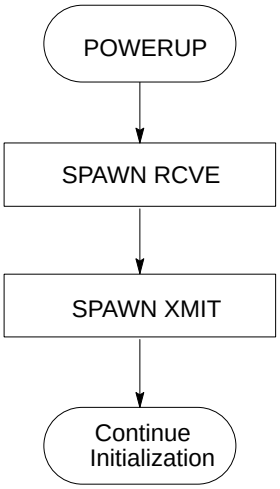
The first alternative is the result of implementing the driver totally at the interrupt level where scheduling is dictated by the interrupt daisy chain hardware.

The third alternative would be used if the driver were implemented as tasks under a multi-tasking operating system. Such an implementation might be easier, but would probably be limited to lower communication rates.

POWER-UP

At power-up, the Z80 processor starts executing code at Location 0. The POWER-UP subroutine starts the XMIT and RCVE processes by executing a SPAWN. A SPAWN is very similar to a WAKEUP except that the corresponding SLEEP is imaginary and is located prior to the first instruction of the spawned process.

Figure D.21
POWER-UP Subroutine



11673

Figure D.22
Message Queue

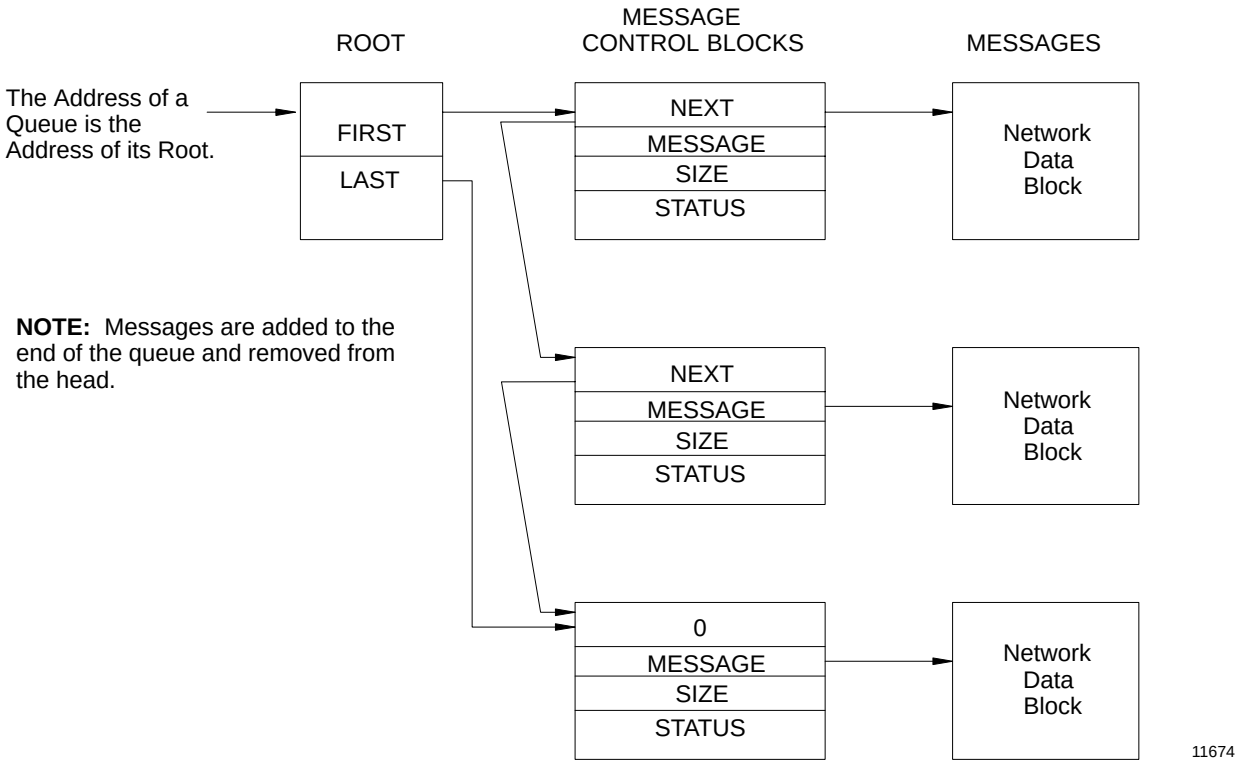
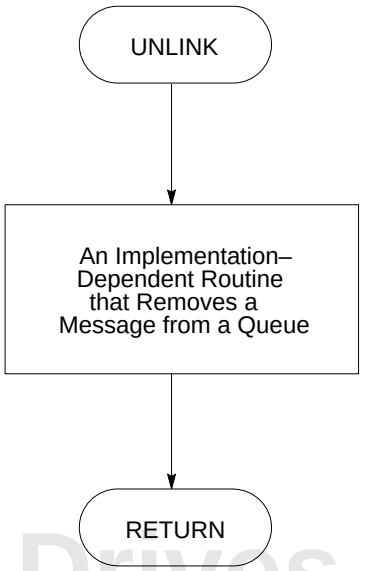


Figure D.23
UNLINK Subroutine



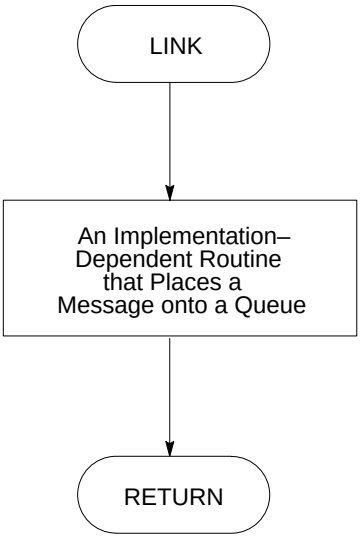
Input:

- Address of Queue

Output:

- Message Control Block

Figure D.24
LINK Subroutine



Input:

- Address of Queue Message Block

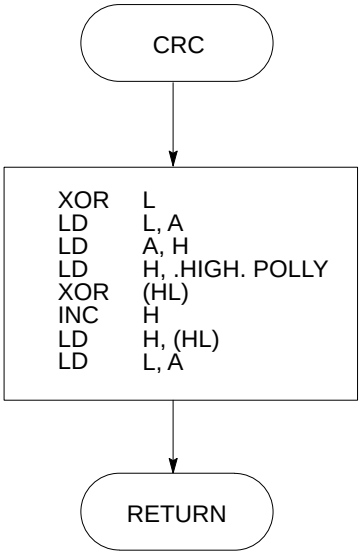
11676

Figure D.25
CRC Subroutine

NOTES: CRC can be implemented using a 256–word table.
CRC = (CRC >>8) XOR (Table [DATA XOR (CRC AND 255)])

>> = Shift Right [ ] Implies Indexed Lookup

The flowchart assumes the use of a Z80 S10.



Input:

- A = Data Byte

Common

- HL = CRC Accumulator

Clobbers

- A

Note: The table is organized as two 256–byte tables aligned on 256–byte boundaries. The first table contains the high byte and is labeled POLY.

11677

Figure D.26
XMSG Subroutine

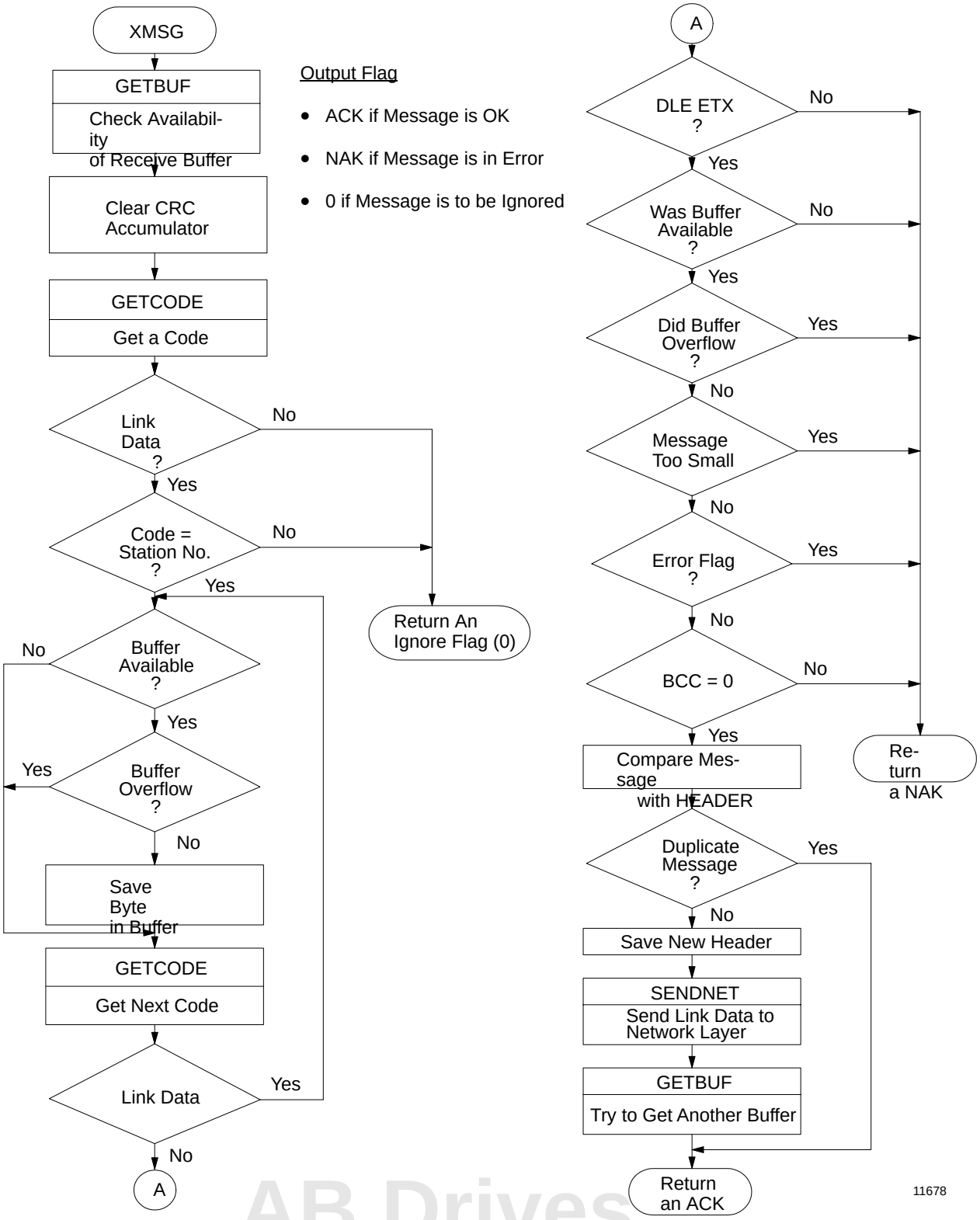


Figure D.27
GETCODE Subroutine

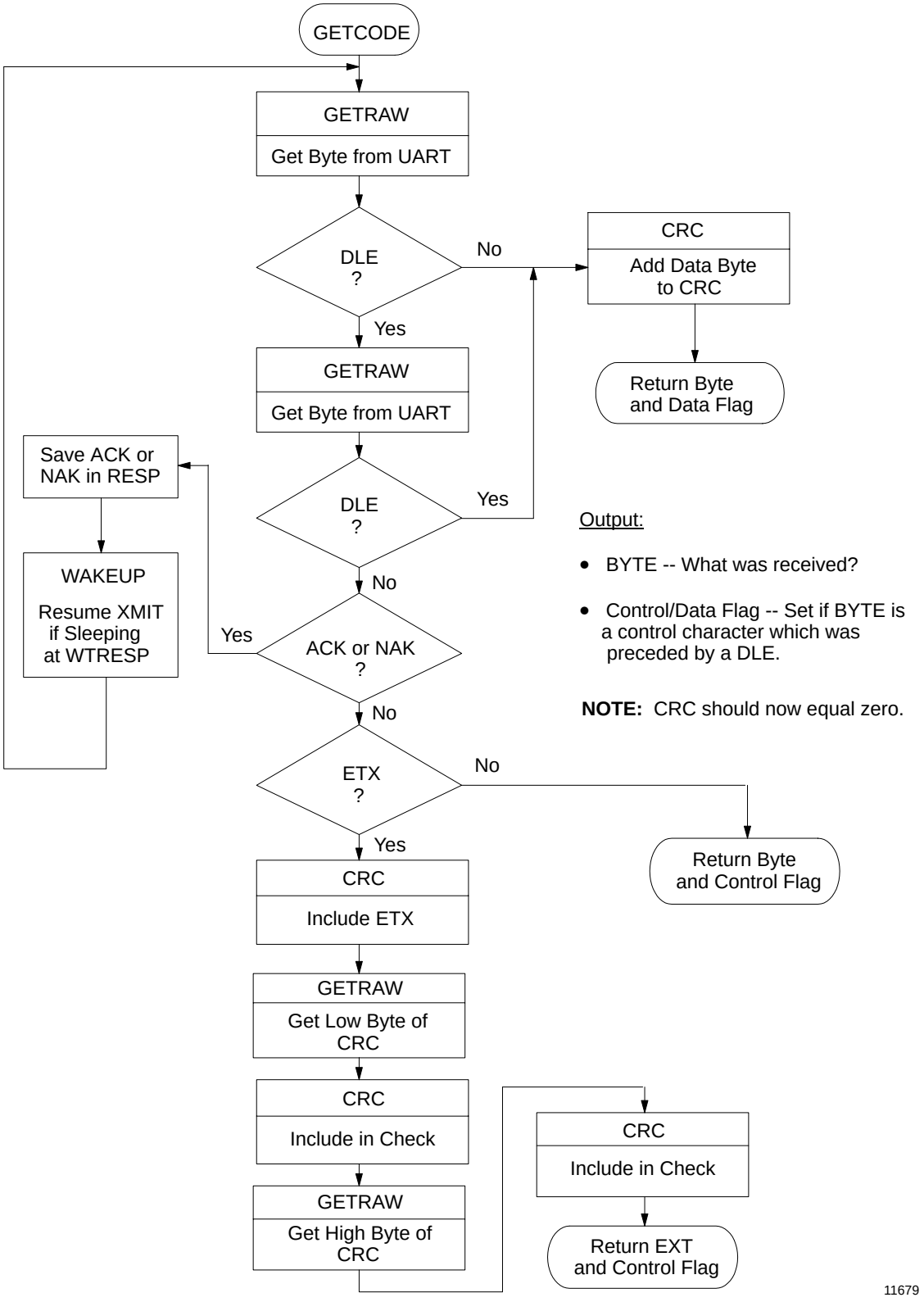
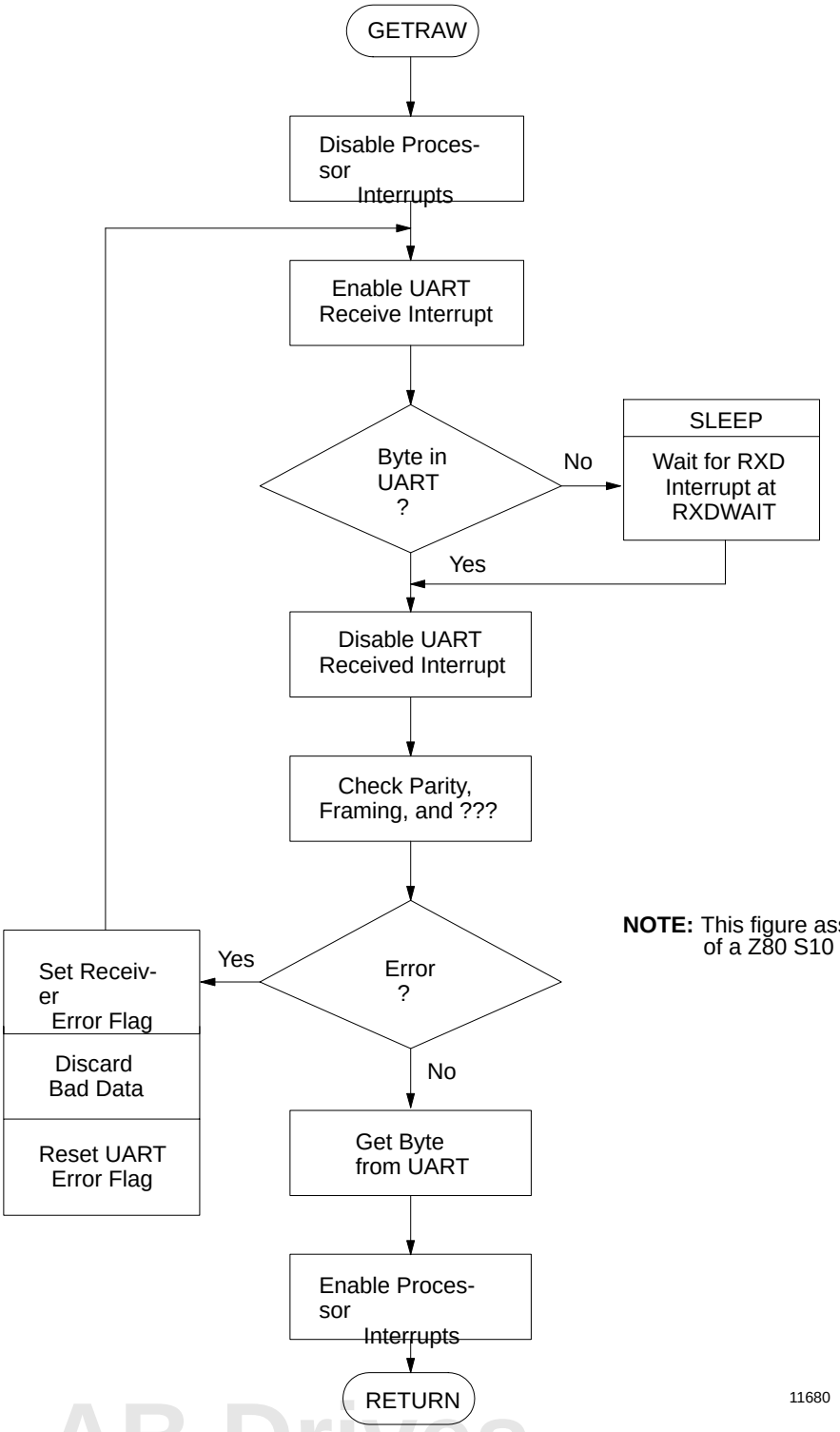


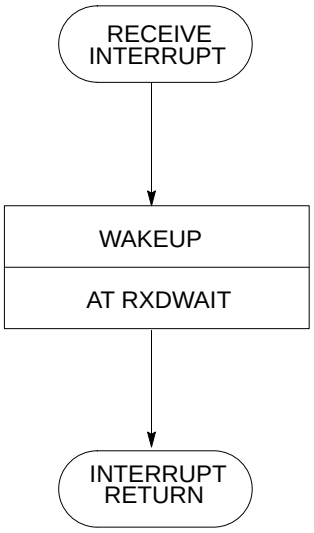
Figure D.28
GETRAW Subroutine



NOTE: This figure assumes the use of a Z80 S10

11680

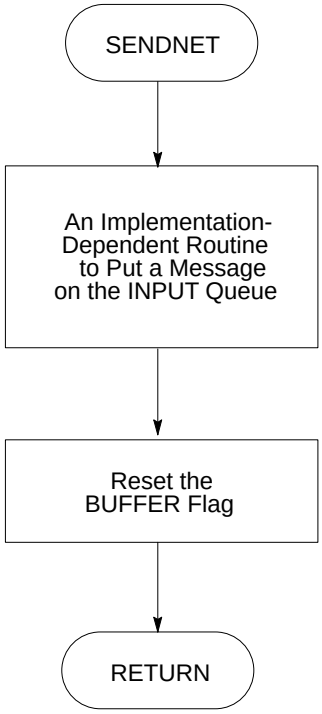
Figure D.29
RECEIVE INTERRUPT Subroutine



NOTE: This figure assumes the use of a Z80 S10

11681

Figure D.30
SENDNET Subroutine



Input

- Message Buffer

11682

Figure D.31
GETBUF Subroutine

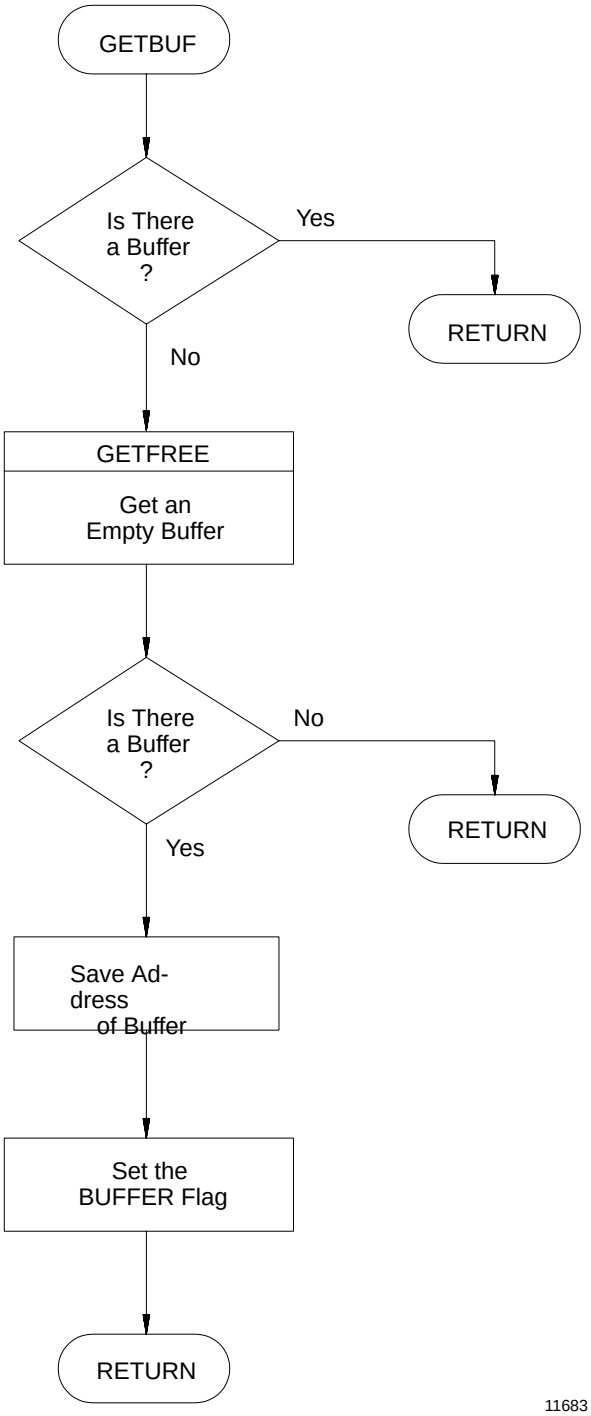
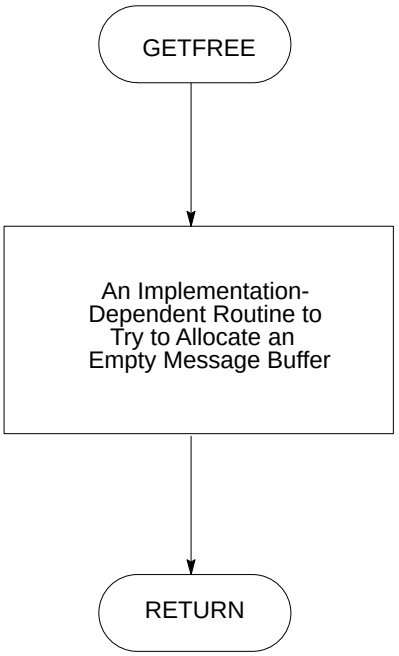


Figure D.32
GETFREE Subroutine



Rockwell Automation

Allen-Bradley

Allen-Bradley, a Rockwell Automation Business, has been helping its customers improve productivity and quality for more than 90 years. We design, manufacture and support a broad range of automation products worldwide. They include logic processors, power and motion control devices, operator interfaces, sensors and a variety of software. Rockwell is one of the worlds leading technology companies.

Worldwide representation.

Argentina • Australia • Austria • Bahrain • Belgium • Brazil • Bulgaria • Canada • Chile • China, PRC • Colombia • Costa Rica • Croatia • Cyprus • Czech Republic • Denmark • Ecuador • Egypt • El Salvador • Finland • France • Germany • Greece • Guatemala • Honduras • Hong Kong • Hungary • Iceland • India • Indonesia • Ireland • Israel • Italy • Jamaica • Japan • Jordan • Korea • Kuwait • Lebanon • Malaysia • Mexico • Netherlands • New Zealand • Norway • Pakistan • Peru • Philippines • Poland • Portugal • Puerto Rico • Qatar • Romania • Russia–CIS • Saudi Arabia • Singapore • Slovakia • Slovenia • South Africa, Republic • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • United Arab Emirates • United Kingdom • United States • Uruguay • Venezuela • Yugoslavia

Allen-Bradley Headquarters, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414 382-2000 Fax: (1) 414 382-4444

Publication 1771–6.5.8 – November, 1985
Supersedes Publication 1771–6.5.8– March, 1984

PN 955098–90