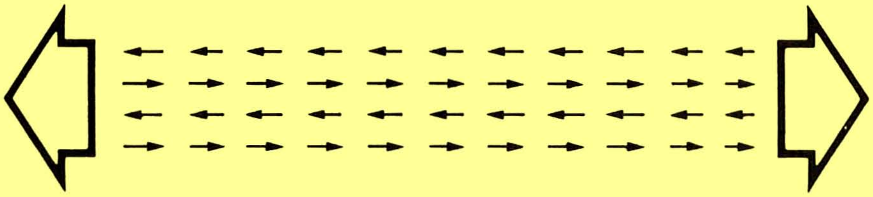
Order No. AA-D599A-TC

DDCMP



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

digital

DECNET

DIGITAL NETWORK ARCHITECTURE

Digital Data Communications
Message Protocol

DDCMP

Specification

Version 4.Ø

1-March-1978

DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS Ø1754

This material may be copied, in whole or in part, provided that the above copyright notice is included in each copy along with an acknowledgment that the copy describes the DDCMP protocol developed by Digital Equipment Corporation.

This material may be changed without notice by Digital Equipment Corporation, and Digital Equipment Corporation is not responsible for any errors which may appear herein.

Copyright (c) 1977, 1978, Digital Equipment Corporation

ABSTRACT

The Digital Data Communications Message Protocol (DDCMP) provides a data link control procedure that ensures a reliable data communication path between communication devices connected by data links. DDCMP has been designed to operate over full- and half-duplex synchronous and asynchronous channels in both point-to-point and multipoint modes. It can be used in a variety of applications such as distributed computer networks, host front-end processors, remote terminal concentrators, and remote job entry-exit systems.

This document describes the functions, characteristics, capabilities, and operation of DDCMP. It is primarily intended to assist the individual implementing DDCMP within a system. It is structured, however, to also provide general information describing the protocol for others who may need this level of information. It is not intended to instruct those unfamiliar with the principles of data communications.

Table of Contents

List of Tables

1.Ø INTRODUCTION

In the design of computer communications networks, one of the basic considerations is the physical transmission of data from one computer to another over a physical data channel. In the absence of transmission errors, this task becomes relatively simple. Once errors are introduced, however, data sequencing and synchronization problems occur between the transmitter and receiver. The solution to these problems consists of a data link control procedure or communications protocol that ensures the correct sequencing and integrity of data transmitted between computers over a data link.

Digital Equipment Corporation recognized the need for such a communications protocol to establish reliable computer-to-computer communications. A standard protocol was designed to serve the needs of interprocessor communications. That protocol has been named the DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL (DDCMP) and has been adopted as a Digital Equipment Corporation standard for intercomputer data communications.

2.0 FUNCTIONAL DESCRIPTION

The Digital Data Communications Message Protocol (DDCMP) has been designed for use over communication channels to provide data integrity, message sequencing, and management of the physical channel. The protocol defines the structure, content, and sequencing procedures for the transmission of data between computers and the techniques used for error detection and recovery. DDCMP resides at a level above the communication medium (i.e., the physical transmission of bits over the communication channel). DDCMP is concerned with the logical transmission of data grouped into physical blocks known as data messages. The primary function of the protocol is to exchange these data messages while ensuring their correct sequencing and integrity when sent over communication channels.

Computers adhering to the protocol will be able to correctly exchange data (between their respective address spaces) over a link. It is the level above the protocol that is concerned with the meaning and understanding of this data once correctly exchanged. With remote entry stations and concentrators, this includes device addressing, device control, and data formatting. With computer networks, it includes problems of network routing, process synchronization, link multiplexing, flow control, and network management.

Programs wishing to communicate using DDCMP must agree on the syntax and semantics of the data transmitted within the DDCMP envelope. DDCMP may thus be viewed as a black box creating an error-free sequential communication path over which data may be transmitted. On multipoint links, DDCMP creates multiple sequential data paths between the control station and the multiple tributaries on the link. If the physical channel connecting two computers is truly error-free, much of DDCMP is not necessary.

2.1 Relationship To DECnet

DECnet is a family of hardware and software products that create distributed networks from DIGITAL computers and their interconnecting data links. DECnet creates a general mechanism for sharing resources and providing interprogram communications within a distributed data processing environment. DECnet implementations adhere to a common network architecture that defines the structure and protocols each must use to communicate through the network. The DIGITAL Network Architecture (DNA) defines this common structure.

DNA provides a modular design for DECnet. Its functional components are defined within three distinct layers:

1. The Physical Link Control Layer (which provides the management of communications over a physical link).

2. The Network Services Layer (which routes messages between source and destination nodes and manages logical data

channels).

3. The Application Layer (which supports user services and programs and provides I/O device and file access).

The DDCMP protocol creates and supports the functions of the Physical Link Control Layer within the DECnet Architecture. This layer provides control over the physical link operation and ensures both data integrity and the sequentiality of data transmitted over a single physical link. It should be noted that DECnet is not a part of the DDCMP standard specification and that DDCMP may be used, independently, in a wide variety of systems and environments where error-free communication is desired.

2.2 Features

DDCMP includes the following features:

1. Error detection using the 16-bit, CRC-16, cyclic redundancy check error detection polynomial.

2. Error correction by retransmission.

3. Message sequencing, which permits up to 255 outstanding messages for pipelining.

4. Operation that is independent of the channel bit width (serial or parallel) and transmission characteristics (asynchronous and synchronous). DDCMP will operate with a wide variety of communication hardware (e.g., character interrupt and block transfer or DMA) and modems.

5. Common operation over full- and half-duplex, point-to-point and multipoint channels.

6. A positive startup procedure that synchronizes both ends of the link.

7. Simplicity and efficiency with only a few message formats.

8. A maintenance mode for diagnostic testing and bootstrapping functions.

9. Data transparency of any bit sequence using a length field description technique.

2.3 Operating Requirements

DDCMP was designed to serve the needs of interprocessor communications in a wide variety of applications and environments. DDCMP will provide:

1. High Performance. DDCMP will provide high data throughput on links capable of such and make optimum use of link characteristics.

2. Wide Applicability. DDCMP will ensure operation that is independent of channel type over a wide range of system configurations.

3. Use of Available Hardware. DDCMP will be able to operate with most communications equipment that is utilized with minicomputers.

2.3.1 Goals - In addition to ensuring the error-free transmission of data, DDCMP was designed to meet specific performance and compatibility requirements. These goals are:

1. Create a protocol for the transmission of data over communication links to provide the correct sequencing and integrity of the data transmitted (even when the link may distort the information transmitted).

2. Operate over a wide variety of communications devices available on micro, mini, and maxi computers in bit serial (asynchronous and synchronous) and bit-parallel modes.

3. Operate over point-to-point and multipoint circuits in both full- and half-duplex modes using a common set of messages and operating procedures.

4. Provide for the efficient transmission of binary (transparent) data.

5. Ensure both high performance and simultaneous operation over full-duplex channels where long circuit delays may be encountered.

6. Provide error recording and reporting features so that a degraded link can be detected and repaired prior to link failure.

7. Provide a positive indication (and synchronization) that the protocol module on the other end of the link has reinitialized or started.

8. Provide a basic operational mode for maintenance functions such as bootstrapping and diagnostic testing.

9. Provide a rigid enough protocol so that all implementations on the same channel type will operate together, independent of implementation techniques.

10. Create a protocol to minimize the memory requirements and execution time in the systems implementing the protocol.

11. Create a protocol that allows the physical characteristics of the channel to become transparent to the user.

2.3.2 Restrictions - Even though DDCMP is a general purpose link protocol offering high performance over a wide range of applications, there are a number of situations in which it may not be optimal. Some of these restrictions are:

1. DDCMP accepts data in blocks that are a multiple of 8-bit bytes. Within a data block, a user can interpret the data in any manner (i.e., 5-bit quantities), but the total block must be a multiple of 8 bits.

2. DDCMP may not be optimal when operating on links with long propagation delays and a high probability of error. Optimal techniques in these cases might include forward error correction and single message retransmission. When an error occurs, DDCMP must go back to the last sequential correct message, thereby losing any pipelining in effect on the link.

3. DDCMP may not be suitable in some multipoint systems having many tributaries with low utilization and fast response requirements. Optimal techniques might include contention selection and broadcast. DDCMP uses a polling selection mechanism, which in some environments results in a longer response time.

4. On multipoint links, DDCMP supports only a single control station. No messages can be exchanged directly between tributaries. Within a given system, the control station can not float among the tributaries.

5. On multipoint links there is no broadcast or multiple addressing facility.

2.4 Data Link Functions

The DDCMP protocol is an extension of the data communications link, providing a number of functions to the user of the protocol. DDCMP may be viewed as a black box creating an error-free sequential managed data link. On the transmit side, messages are given to DDCMP, which delivers them over the link and notifies the user when the delivery has successfully occurred. On the receive side, the user provides buffers that are filled with correctly received messages by DDCMP. The term "user" refers to the process or program exchanging messages with the protocol. In DECnet it might be the next higher level protocol (i.e., the Network Services Protocol). In other systems or structures it might be a service process or the end-user directly. DDCMP extends the capabilities of a data link to include the following features:

1. Creates an error-free data path. DDCMP transfers data between protocol users over a physical link, while maintaining data integrity within some very small undetected error probability. If data integrity cannot be maintained, no data will be transferred.

2. Transfers messages in proper sequence. Messages will be delivered from one user to the other in the same order as they are sent, even though DDCMP may require the use of retransmission or other error recovery techniques.

3. Manages the characteristics of the channel. If the channel requires receiver addressing and/or arbitration of transmission requests, DDCMP is responsible for that management.

4. Interface to modem control signals. DDCMP must interface with signals necessary for the operation of the physical channel, (e.g., modem control signals not handled by other components in the system). It may do this directly, leave it up to the hardware device driver, or let the user of the modem control code control these signals through the protocol interface.

5. Accesses data in blocks consisting of byte quantities. DDCMP accepts data in blocks consisting of 8-bit bytes. All 256 8-bit combinations are transmittable, and transparent to DDCMP. The protocol will allow blocks of up to 16,383 bytes to be transmitted. However, the CRC-16 error detection polynomial used is most effective with blocks up to 4093 bytes long.

6. Provides restart or initialization notification. If the other end of the link resets or initializes, DDCMP will notify the user.

7. Provides start and stop control. The user controls the protocol and can start (or reinitialize), and stop (or halt) the operation of DDCMP.

8. Provides notification of channel error. When a persistent error is detected, the user is notified of such a condition. Such errors might be (a) too high a bit error rate; (b) outages; (c) nonexistent communications; or (d) modem failure.

9. Provides a maintenance mode. DDCMP creates a data envelope with bit error-detection-only capability for use in diagnostic testing and system bootstrapping functions.

2.5 Functional Organization

From an operational viewpoint, DDCMP consists of three functional components: (1) Framing, (2) Link Management, and (3) Message Exchange. The following sections provide a generic model describing each of these components. This model is helpful in understanding DDCMP operation. It is provided as an aid in implementation design (by enabling an individual to understand the protocol, its operational intentions and motivations). It is not intended to describe specific operating details of DDCMP or subsets of DDCMP. For specific information on the actual protocol operation refer to Section 5.Ø.

2.5.1 Framing Component - Framing is the process of locating the beginning and end of a message, at the receiving end of a link. Synchronization is the process of locating some entity (e.g., a bit or byte) and then staying in step or operating at the same rate as that entity. Synchronization of data on a link must occur at the bit, byte, and message levels before framing can be accomplished. The following paragraphs describe how DDCMP provides synchronization at these levels:

1. Bit synchronization. Locating a bit on the link. This function is accomplished by the modems or interfaces on the link and is not a part of DDCMP.

2. Byte synchronization. Grouping bits into 8-bit byte quantities. Byte synchronization is the process of locating the proper 8-bit window in the bit stream and then staying in step with it for every 8-bit byte grouping. On 8-bit asynchronous links, this process is inherent in the start/stop transmission technique on the link. Byte framing is established with each byte sent. On synchronous links, byte synchronization is established by searching for a unique 8-bit sequence called the sync byte, checking for two in a

row, and then counting every 8 bits as a byte. The unique pattern is such that any skewing to the right or left will not produce a sequence match. On 8-bit or 8-bit multiple parallel links, byte synchronization is inherent in the link. For other types of links, techniques will have to be designed to locate the proper 8-bit byte window.

3. Message synchronization. Locating the first byte of a message. In DDCMP, this is done by searching for one of three special starting bytes after achieving byte synchronization. Once one is found, simple rules will locate the end of the message. The message is framed and may be processed. The starting byte also defines the format type of the message and how the remaining bytes are to be interpreted.

The byte and message synchronization techniques were chosen to allow the greatest flexibility and independence from the actual data link characteristics. By using these techniques, DDCMP can operate on serial synchronous links with typical character interrupt or block transfer devices and on serial asynchronous links using 8-bit bytes. Byte synchronization is specified in DDCMP and is specific for each data link and its characteristics. Message synchronization is the same for all link types once byte synchronization has been established.

2.5.2 Link Management Component - The Link Management Component resolves the transmission and reception on links that are connected to two or more transmitters and/or receivers in a given direction. This is true of half-duplex and multipoint channels. Link management occurs for both the transmitting and receiving functions.

On half-duplex links, one station must be receiving while the other is transmitting. The switching between transmit and receive is via a selection flag. The station that is transmitting ends transmission by setting the flag in its last message. This signals the receiver to complete reception of this message and then enter transmit mode.

For reception on multipoint links, the link appears as a party line. One station is designated the control station, the others are tributaries. All messages contain a tributary address to identify them. Messages to a tributary are received by all tributaries and ignored by all except the one with the matching address. Messages from tributaries are ignored by other tributaries and received by the control station that verifies the tributary address to be the one selected. Message traffic is only between the control station and tributaries.

For transmission on multipoint links, the control station manages the link and assigns transmission ownership or selection to tributary stations via a selection flag. Tributary stations, once selected, may transmit and will terminate transmission by sending a selection flag to the control station in the last message of a transmission sequence.

Timers are used by half-duplex or control stations to handle the case of a lost flag (i.e., the message containing the flag is in error). A timer is started when waiting for the next message. If it expires it is assumed the selection flag was received in error and the station operates as if it received a valid selection flag.

2.5.3 Message Exchange Component - The message exchange component is the part of DDCMP that creates the sequential error-free link. This component transfers the data correctly and in sequence over a link that has some probability of introducing errors. Once framing is accomplished, this component operates at the message level, exchanging data and control messages. DDCMP is a positive acknowledgment retransmission protocol. For each data message correctly received and passed to the user, a positive acknowledgment is returned on the link notifying the transmitter of the correct receipt of the data message. If incorrectly received, the data message is not passed to the user and not acknowledged. Eventually, it will be retransmitted. DDCMP uses the CRC-16 cyclic redundancy check for error detection. This section describes the component parts of the message exchange mechanism along with their design characteristics and functions.

The basic positive acknowledgment message exchange component requires the following:

1. a data message with message number n;

2. a positive acknowledgment with message number n (ACK); and

3. a timer.

It operates in the following manner:

1. The transmitter puts the next message number n in the data message, adds the CRC block check to the message, puts it in the required framing envelope, and sends it. When it has been transmitted on the link, a timer is started.

2. The receiver frames and receives the message, checks the CRC for errors, and compares the message number with the next expected. If the number is correct, the receiver returns a positive acknowledgment (ACK) with that number, passes the message to the receiving user, and increments the next expected number to n+1 (modulo 256). If the number is in error, the message is ignored.

3. The transmitter follows one of two procedures;

   a. It receives a positive acknowledgment and compares the number received with the one expected. If it agrees, the transmitter releases the message, notifies the transmitting user of successful receipt, stops the timer, and increments the next message number to n+1 (modulo 256). If the acknowledgment does not agree with the expected number it is ignored.

b. It receives nothing and the timer expires. The transmitter initiates error recovery. Various error recovery options are available. The ones used in DDCMP are presented below.

The timer in the message exchange component is keyed to the selection of a tributary (multipoint) or other station (half-duplex). That is, the controlling station must wait until a tributary or the other station is selected and transmits before it determines that a message or ACK was not properly received. This makes the timing independent of the selection of stations.

This mechanism is adequate for creating a message exchange component. The following additional messages and operational techniques of DDCMP are used to achieve higher performance (via pipelining) and faster error recovery (via error notification) but do not add to the basic integrity of the data transfer mechanisms.

1. Negative Acknowledgment (NAK). The time-out value (used to detect an error when an ACK is not returned) must be long enough to account for delays such as propagation, line turnaround, local processing of the data message, and the generation of the ACK. Time-out values might be a few seconds while the actual delay may be on the order of a few milliseconds. If the only way to determine an error is to wait for a time-out, undue waste and inefficiency are encountered. A negative acknowledgment provides a means for more immediate notification of some error conditions. If the receiver does not receive the message correctly, it sends a NAK, which triggers the retransmission long before the timer expires.

   In DDCMP, NAKs are sent in response to cyclic redundancy check errors, but not to wrong message numbers. If the receiver gets a message with the wrong number, the message is ignored and the time-out condition triggers the transmitter to retransmit. If NAKs were sent in both cases, long delays could occur under certain timing conditions.

2. Reply to Message Number (REP). When the timer expires, it is unclear whether the message was received in error or the returned ACK was in error and not received properly. (ACKs also have CRC checks on them). In this case, rather than retransmit perhaps a long message, a REP is sent with the message number of the message previously sent. If the message with that number was received correctly the response to the REP is an ACK, otherwise it is a NAK. The REP forces the transmitter and receiver to synchronize their numbering and start retransmission if required. The transmission timer

is restarted after sending a REP. If it expires again the process is repeated. After some specified number of these time-outs, the transmitter will notify the DDCMP user, who may declare the link out-of-service.

3. Pipelining. The ability to send more than one message without waiting for ACKs to each successive message is called pipelining. Within DDCMP, messages are numbered from Ø to 255. This numbering is cyclic (modulo 256) in that after message number 255 the next message number is Ø. ACKs not only confirm that the specified message number has been received correctly, but that all previous messages with numbers between the one acknowledged in the last ACK and the one acknowledged by the current ACK (modulo 256) have been received correctly. If an ACK message is in error, the information lost is automatically included in subsequent ACKs, eliminating the sending of REP messages if the ACKs are received prior to the expiration of the transmission timer. This technique is also used with the REP message, the number sent in the REP being the number of the last message transmitted.

4. Piggybacking. The purpose of an ACK is to convey the message number of the last successfully received data message. If data message traffic is going in both directions, the ACK number can be sent piggybacked on or within the frame of the message going in the other direction. This technique saves separate framing overhead for the ACK.

5. ACK implied in NAK. The number referenced in a NAK reply identifies the last successfully received message as well as noting a received error. So NAK implies that all messages prior to the one being negatively acknowledged were received correctly.

6. Initialization. The method of setting message numbers to initial values is called initialization. It is accomplished by STRT and STACK messages that reset message numbers to zero. It is used initially or after a failure to reset number values at both ends. It is designed so that one end cannot be initialized without the other.

3.0 INTERFACES

This section describes how DDCMP is viewed by a user of the protocol and how the physical interface device or driver is viewed by DDCMP. A generic description of the information that must be passed across the interfaces to the user and the device is presented as an aid for implementation design.

3.1 User Interface

The interface between DDCMP and the user consists of a number of commands to DDCMP and responses from DDCMP. In these commands and responses, the user exchanges data and control information with the protocol. The actual interface mechanism depends heavily on the features and capabilities within the operating systems running DDCMP. Mechanisms for exchanging this information might include shared tables, calls with parameter lists, I/O registers, and interrupt mechanisms.

Three kinds of information are exchanged in the command/response sequences: (1) data, (2) control information, and (3) error information. Data is the user information to be sent or received by the protocol. Its description usually consists of a starting buffer address and a length or character count, or a chain of addresses and counts. Control is information to start and stop the protocol and notify the user of protocol initialization. Error information is provided by the protocol for use in determining the physical condition of the link and when maintenance is necessary. DDCMP is totally controlled by the user of the protocol. It is only activated by a command request from the user and continues to operate even when large numbers of data errors occur on the physical link. It is started, stopped, and reinitialized only upon commands from the user. On multipoint links, independent command and response sequences are maintained between the control station and each tributary on the link. The link appears as multiple point-to-point links, one for each tributary address.

The exact interface between DDCMP and the user depends on the system implementation and will vary in the manner in which errors are handled. In some systems, the error handling code may be totally at the user level, each error being reported to the user. In other systems, the error handling code may be part of the protocol module and only persistent errors will be reported to the user. Section 6.0 describes the error information recording techniques that may be employed within DDCMP.

3.1.1 Commands To DDCMP - The basic commands to DDCMP are:

1. Initialize Link. Initialize the protocol and start the data link.

2. Stop Link. Halt the protocol. In some dial-up situations, a method may be employed to force the modem to hang-up.

3. Transmit Message. Give a message to DDCMP for transmission. As an option the user may specify that it wishes to send the message within the maintenance mode, or the protocol implementation may require a separate Maintenance Mode Initialization command prior to a transmit request.

4. Receive Message. Give an empty buffer to DDCMP for reception of the next sequential message. Alternatively, the user might supply a pool of buffers to DDCMP initially, and have the protocol select one. In this mode, there will be a command to return empty buffers to the pool so they may again be used by DDCMP.

5. Return Transmit Buffers. This optional command, which can be employed after halting DDCMP, returns outstanding transmit buffers to the user. The response to this command would include whether they were already transmitted and acknowledged, not yet acknowledged, or not yet transmitted.

6. Enter Maintenance Mode. This command is an option to first change to the maintenance mode before transmitting or receiving maintenace mode messages.

3.1.2 Responses From DDCMP - The responses from DDCMP are:

1. Initialization on Other End. The other end has restarted or initialized. This response will halt the protocol. The command to restart the protocol on this end will be an Initialize Link command.

2. Initialization Complete. Response to Initialize Link command. This response is optional. If it is omitted, the reply to a successfully received or transmitted message will serve as initialization completion notification.

3. Message Transmitted. Response to the Transmit Message command. The message was successfully received on the other end (acknowledged).

4. Message Received. The next sequential message was successfully received. Either the user buffer specified in the Receive Message command will be used, or a buffer will be taken from a pool, if such a buffering technique is employed.

Optionally, if the message was received in the maintenance mode it may be so marked, or a separate response may be first sent to the user to indicate that the other end is in maintenance mode. At that point, the protocol will halt, and the user will have to initialize the protocol into the maintenance mode before receiving maintenance mode messages.

5. Transient Error Threshold Counter Overflow. An error threshold counter has overflowed. The protocol will continue operation. It must be halted by the user if the user wishes to cease operation (refer to Section 6.0, Error Recording).

6. Persistent Error. An error has occurred from which recovery may not be possible. Some implementations of the protocol may halt operation. Some errors that are classified as persistent errors in one system, might be transient errors in another. The various types of errors are discussed in Section 6.0.

3.2 Device Driver Interface

The interface between DDCMP and the line driver includes a number of commands and responses used to transmit and receive message blocks to and from the link, respectively. The actual interface depends heavily on the mechanisms and capabilities available in the I/O structure of the system within which this interface operates. It also depends heavily on the split of protocol functions between DDCMP and the driver. The driver may be very protocol independent and rely on heavy interaction with DDCMP for message framing, CRC calculation, and the syntactic and semantic interpretation of message fields. Alternately, it may embody much of DDCMP including framing, CRC checking, link management, and link turnaround. In this mode, there would be less interaction with the semantic or message exchange portion of DDCMP. Consequently the driver would handle many of the functions related to link type and device characteristics. The choice of driver capabilities and the split of functions depends on system characteristics, device requirements, driver generality, and the interface to other protocols. The interface described here lies between these two extremes and is presented as an aid to understanding what information must pass across this interface.

Message blocks are usually passed to the driver via a buffer address and length (or a chain of addresses and lengths to allow fragmented message blocks).

3.2.1 Commands To The Driver - The driver receives the following commands:

1. Link and Modem Control. These commands activate and connect a physical link to DDCMP. They also control the modem signals necessary for proper operation. These signals may be implicit in enabling the link (i.e., turn Data Terminal Ready (DTR) on) or explicit via modem control commands to allow DDCMP to directly control the modem. Typical commands might be:

   a. Enable link. This command connects the driver to DDCMP and turns DTR on.
   b. Disable link. This command disconnects the driver from DDCMP and turns DTR off.

2. Buffer Management. Received message blocks are passed to DDCMP via buffers. The buffers may be (a) individually given to the driver via Receive commands or (b) initially allocated to the driver in a Set buffers command or the Enable link command. In this second mode there must also be a command for DDCMP to return the buffers to the driver. On disconnection (Disable link command), the buffers must be returned to DDCMP or a buffer pool.

3. Transmit a Block. This command passes a block to the driver for transmission. The request might include one of the following options: (a) proceed with a synchronization sequence; (b) end with a pad; (c) calculate CRC; or (d) shutdown the transmitter after the message. These options depend on the precise division of functions between the driver and protocol.

4. Receive a Block. This command passes buffers to the driver if individual explicit buffers are used. Otherwise, the driver might simply queue received blocks to DDCMP using buffers from a previously obtained pool (as noted in 2). This command may also request the driver to resynchronize or reframe the receiver or there may be a separate Resync Receiver command.

5. Set Parameters. If the driver design is protocol independent, this command might be included to set such parameters as synchronization sequences and the CRC polynomial.

3.2.2 Responses From The Driver - The driver issues the following responses:

1. Modem Status. The driver returns modem signals, such as Data Set Ready (DSR), if appropriate to the interface.

2. Received Block. The driver passes a received data block to DDCMP. Depending on the functional split between the driver and DDCMP, the driver may calculate CRC (either in the driver or device itself) and pass this status with the block. When DDCMP is finished with the buffer it returns it to the driver via either (a) a Receive command or (b) a Return Buffer command, depending on the buffering scheme used.

3. Transmit Complete. The driver will notify DDCMP when a previous Transmit a Block command has been completed.

4.Ø MESSAGE FORMATS

This section describes the message formats of DDCMP. Data is exchanged over DDCMP links between the data source (master) and data sink (slave) within numbered data messages. Responses and control information are returned from the slave to the master within unnumbered control messages. Stations contain both a master and slave. For the purpose of exchanging data, the station plays the role of master or slave depending on whether it is transmitting or receiving the data. It is a distinction used for easy understanding and explanation of DDCMP. In reality, data is usually exchanged in both directions. In the following explanation only a single direction is described.

Each data message carries a number assuring correct message sequencing at the slave. The numbering begins with number one after initialization via the STRT/STACK control message sequence and is incremented by one (modulo 256) for each subsequent data message. The slave always acknowledges the correct receipt of data messages by returning the message number as a response either in the response field of numbered data messages going in the reverse direction, or, in an ACK unnumbered control message. For efficiency, an acknowledgment of the data message with number n implies an acknowledgment of all data messages sent up to and including data message number n. Retransmission is used to recover from errors. The error recovery mechanism uses timeouts and NAK and REP control messages to resynchronize and cause retransmission if required. All messages also include station addresses and link control flags for use on multipoint and half-duplex channels.

4.1 Notation

The following notation is used to describe the messages:

Field (length) : coding = description of field

Field = the name of the field being described

length = the length of the field as:
(1) a number meaning the number of 8-bit bytes or
(2) a number followed by a B meaning the number of bits

coding = the representation type used:

B	= Binary
BM	= bit map (each bit has independent meaning)
C	= constant
Null	= interpretation data dependent

Fields in separate messages that have the identical name are the same field and have identical meaning.

All numeric values in this document are shown in decimal representation unless otherwise noted.

All header fields and bytes of data are transmitted low-order or least-significant bit first on the data links unless otherwise specified.

4.2 Data Messages

Numbered data messages carry user data over DDCMP links. The format of a numbered message is:

```
+---+-----+-----+----+---+----+------+----+------+
!SOH!COUNT!FLAGS!RESP!NUM!ADDR!BLKCK1!DATA!BLKCK2!
+---+-----+-----+----+---+----+------+----+------+
```

where:

SOH(1) : C = the numbered data message identifier. It has a value of 129 (octal - 201).

COUNT(14B) : B = the byte count field. It specifies the number of 8-bit bytes in the DATA field. The value zero is not allowed.

FLAGS(2B) : BM = the link flags. They are used to control link ownership and message synchronization. These flags are:

bit 0 = quick sync flag (QSYNC flag), used to notify the receiver that the next message will not abut this message and resynchronization should follow this message. The quick sync flag reduces the length of sync sequences on synchronous links.

bit 1 = select flag (SELECT flag), used to control transmission ownership on multipoint and half-duplex links. Reverses link direction on half-duplex links. Invites a tributary to send and signals end of tributary selection on multipoint links.

NOTE

COUNT and FLAGS form a 2-byte quantity. The first byte contains the 8 low-order bits of the COUNT. The second byte contains the 6 high-order bits of the COUNT, the SELECT flag the highest order or most significant bit of the byte, and the QSYNC flag the next bit in the byte.

```
                      -----------------
                      ! S ! Q ! COUNT !
                      -----------------
 high order bit                          low order bit
 transmitted last                        transmitted first
```

RESP(1) : B = the response number. It is used to acknowledge correctly received messages (the piggybacked ACK). It is the number of the last consecutive correctly received message received from the addressed station by the station transmitting this message. It implies that all unacknowledged messages between the one acknowledged in the last RESP field received and the one acknowledged by this RESP field (modulo 256), have been received correctly.

NUM(1) : B = the transmit number. It is used to denote the number of this data message.

ADDR(1) : B = the station address field. It is used to designate the address of tributary stations on multipoint links. Stations on point-to-point links use the address value 1.

BLKCK1(2) : B = the block check on the numbered message header. It is computed on SOH through ADDR using the CRC-16 polynomial $(X^{16}+X^{15}+X^{2}+1)$. BLKCK1 is initialized to zero prior to computation and transmitted X^{15} bit first. On reception the inclusion of BLKCK1 in the computation will result in a zero remainder or CRC if no errors exist. See Appendix C for a description of CRC computation.

DATA (COUNT) = the numbered message data field. This field is totally transparent to the protocol and has no restrictions on bit patterns, groupings, or interpretations. The only requirement is that it contain the number of 8-bit bytes specified in the COUNT field.

BLKCK2(2) : B = the block check on the data field. It is computed on the DATA field only using the polynomial and technique described above for BLKCK1.

4.3 Control Messages

Unnumbered control messages carry channel control information, transmission status, and initialization notification between the protocol modules themselves. The individual fields are specific for each type of control message. Control messages have the following general form:

```
+---+----+-------+-----+----+----+----+------+
!ENQ!TYPE!SUBTYPE!FLAGS!RCVR!SNDR!ADDR!BLKCK3!
+---+----+-------+-----+----+----+----+------+
```

where:

ENQ(1) : C = the unnumbered control message identifier. It has a value of 5 (octal - ØØ5).

TYPE(1) : B = the control message type. This value denotes each control message.

SUBTYPE(6B) : B = the subtype or type modifier field. It provides additional information for some message types. Its use is specific for each message type.

FLAGS(2B) : BM = the link flags. They are the same as described for numbered data messages (See Section 4.2).

RCVR(1) : B = the control message receiver field. It is used to pass information from the data message receiver or slave station to the data message sender or master station. Its use is specific for each control message type.

SNDR(1) : B = the control message sender field. It is used to pass information from the data message sender or master to the data message receiver or slave. Its use is specific for each control message type.

ADDR(1) : B = the station address field. It is the same as described for numbered data messages (See Section 4.2).

BLKCK3(2) : B = the block check on the control message. BLKCK3 is computed on fields ENQ through ADDR using the polynomial and technique described for numbered data message BLKCK1 (See Section 4.2).

NOTE

The common fields in data and control messages are in the same position relative to the beginning of the message. The two types line up as follows:

```
+---+-------------+-----+-----+----+----+------+----+------+
!SOH!  C O U N T  !FLAGS!RESP ! NUM!ADDR!BLKCK1!DATA!BLKCK2!
+---+-----+-------+-----+-----+----+----+------+----+------+
!ENQ!TYPE !SUBTYPE!FLAGS!RCVR !SNDR!ADDR!BLKCK3!
+---+-----+-------+-----+-----+----+----+------+
```

4.3.1 Acknowledge Message (ACK) - The ACK message is used to acknowledge the correct receipt of numbered data messages. It conveys the same information as the RESP field in numbered messages and is used when acknowledgments are required, and when no numbered messages are to be sent in the reverse direction. The form of the ACK message is:

```
+---+-------+------+-----+----+----+----+------+
!ENQ!ACKTYPE!ACKSUB!FLAGS!RESP!FILL!ADDR!BLKCK3!
+---+-------+------+-----+----+----+----+------+
```

where:

ENQ(1) : C = the control message identifier.

ACKTYPE(1) : C = the ACK message type with a value of 1.

ACKSUB(6B) : C = the ACK subtype with a value of Ø.

FLAGS(2B) : BM = the link flags.

RESP(1) : B = the response number used to acknowledge correctly received messages. It is the same as described for numbered data messages (See Section 4.2).

FILL(1) : C = a fill byte with value Ø.

ADDR(1) : B = the station address field.

BLKCK3(2) : B = the control message block check.

4.3.2 Negative Acknowledge Message (NAK) - The NAK message is used to pass error information from the slave (or data receiver) to the master (or data sender). The error reason is included in the subtype field. The NAK message also includes the same information as the ACK message, thus serving two functions: acknowledging previously received messages and notifying the master of some error condition. The form of the NAK message is:

```
+---+-------+------+-----+----+----+----+------+
!ENQ!NAKTYPE!REASON!FLAGS!RESP!FILL!ADDR!BLKCK3!
+---+-------+------+-----+----+----+----+------+
```

where:

ENQ(1) : C = the control message identifier.

NAKTYPE(1) : C = the NAK message type with a value of 2.

REASON(6B) : B = the NAK error reason. Identifies the source and reason for the NAK.

1. Error usually due to transmission medium:

   Value and Reason

   1 = header block check error (data message BLKCK1 or control message BLKCK3).
   2 = data field block check error (data message BLKCK2).
   3 = REP response.

2. Error usually due to computer/interface:

   Value and Reason

   8 = buffer temporarily unavailable.
   9 = receive overrun.
   16 = message too long.
   17 = message header format error.

FLAGS(2B) : BM = the link flags.

RESP(1) : B = the response number used to acknowledge correctly received messages. When used in a NAK message usually implies some error in a message with number RESP+1 (modulo 256) or beyond.

FILL(1) : C = a fill byte with a value of Ø.

ADDR(1) : B = the station address field.

BLKCK3(2) : B = the control message block check.

4.3.3 Reply To Message Number (REP) - The REP message is used to request received message status from the slave or data receiver. It is usually sent when the master has transmitted data messages and has not received a reply within a timeout period. The response to a REP is either an ACK or NAK depending on whether the slave has or has not received all messages previously sent by the master. The form of the REP message is:

```
+---+-------+------+-----+----+---+----+------+
!ENQ!REPTYPE!REPSUB!FLAGS!FILL!NUM!ADDR!BLKCK3!
+---+-------+------+-----+----+---+----+------+
```

where:

ENQ(1) : C = the control message identifier.

REPTYPE(1) : C = the REP message type with a value of 3.

REPSUB(6B) : C = the REP subtype with a value of Ø.

FLAGS(2B) : BM = the link flags.

FILL(1) : C = a fill byte with a value of Ø.

NUM(1) : B = the number of the last sequential numbered data message (not including retransmissions) sent by the master. This is compared against the number of the last sequential message received by the slave and results in either an ACK being returned if they agree or a NAK if they do not. The NAK will contain the number of the last sequential message that was received.

ADDR(1) : B = the station address field.

BLKCK3(2) : B = the control message block check.

4.3.4 Start Message (STRT) - The STRT message is used to establish initial contact and sychronization on a DDCMP link. It is used only on link startup or reinitialization. It operates with the start acknowledge message STACK described below. The start sequence resets message numbering at the transmitter and addressed receiver. The form of the STRT message is:

```
+---+--------+-------+-----+----+----+----+------+
!ENQ!STRTTYPE!STRTSUB!FLAGS!FILL!FILL!ADDR!BLKCK3!
+---+--------+-------+-----+----+----+----+------+
```

where:

ENQ(1) : C = the control message identifier.

STRTTYPE(1) : C = the STRT message type with a value of 6.

STRTSUB(6B) : C = the STRT subtype with a value of Ø.

FLAGS(2B) : C = the link flags. For STRT, both flags are ones (flag value of 3).

FILL(1) : C = a fill byte with a value of Ø.

FILL(1) : C = a fill byte with a value of Ø.

ADDR(1) : B = the station address field.

BLKCK3(2) : B = the control message block check.

4.3.5 Start Acknowledge Message (STACK) - The STACK message is returned in response to a STRT when the station has completed initialization and reset message numbering. The form of the STACK message is:

```
+---+--------+-------+-----+----+----+----+------+
!ENQ!STCKTYPE!STCKSUB!FLAGS!FILL!FILL!ADDR!BLKCK3!
+---+--------+-------+-----+----+----+----+------+
```

where:

ENQ(1) : C = the control message identifier.

STCKTYPE(1) : C = the STACK message type with a value of 7.

STCKSUB(6B) : C = the STACK subtype with a value of Ø.

FLAGS(2B) : C = the link flags. For STACK, both flags are ones (flag value of 3).

FILL(1) : C = a fill byte with a value of Ø.

FILL(1) : C = a fill byte with a value of Ø.

ADDR(1) : B = the station address field.

BLKCK3(2) : B = the control message block check.

4.4 Maintenance Messages

The DDCMP protocol operates in two basic modes: (1) on-line or the normal running mode and (2) off-line or the maintenance mode. The previous messages and operation describe the on-line mode. The off-line or maintenance mode may be used for basic diagnostic testing and simple operating procedures such as bootstrapping, down-line loading, or dumping. It provides a basic envelope compatible with DDCMP framing, link management, and the CRC check for bit errors, but does not include any error recovery, retransmission time-outs, or sequence checks. All these functions, if necessary, are handled by the user of this mode within the data field. The maintenance message is similar in format to the data message. The format of the maintenance message is:

```
+---+-----+-----+----+----+----+------+----+------+
!DLE!COUNT!FLAGS!FILL!FILL!ADDR!BLKCK1!DATA!BLKCK2!
+---+-----+-----+----+----+----+------+----+------+
```

where:

DLE(1) : C = the maintenance message identifier, has the value 144 (220-octal).

COUNT(14B) : B = the byte count field, specifies the number of 8-bit bytes in the DATA field. The value zero is not allowed.

FLAGS(2B) : C = the link flags. Both flags are ones for maintenance messages (flag value of 3).

FILL(1) : C = a fill byte with a value of 0.

FILL(1) : C = a fill byte with a value of 0.

ADDR(1) : B = the station address field.

BLKCK1(2) : B = the header block check on fields DLE through ADDR. Same as described for data messages (See Section 4.2).

DATA (COUNT) = the data field. It consists of COUNT 8-bit bytes.

BLKCK2(2) : B = the block check on the DATA field only. Same as described for numbered data messages (See Section 4.2).

5.0 OPERATION

The DDCMP functions may be grouped into three areas: Framing, Link Management, and Message Exchange. These functional components are:

Framing
The process of locating the beginning and end of a message. It may involve bit, byte, and message synchronization. Once framing is accomplished the protocol operates at the logical message level, both sending and receiving message blocks.

Link Management
The process of controlling the transmission and reception on links connected to two or more transmitters and/or receivers on a common signal channel. This is true of half-duplex and multipoint links. There must be an orderly mechanism for the proper receiver to identify its data and for only one transmitter on a common signal channel to be active at a given time.

Message Exchange
The process of transferring user data over the link sequentially and without bit errors. DDCMP is a positive acknowledgment retransmission protocol, returning an indication to the transmitter for each message that has been successfully received.

5.1 Framing

The basic concepts of framing were presented in Section 2.5. This section discusses the specific details of framing for each link type on which DDCMP operates. Framing occurs at the bit, byte, and message levels.

Bit framing is handled by the modems and interfaces, and is not a part of this standard.

5.1.1 Byte Framing - This process entails framing on the proper 8-bit byte sequence so that bits may be grouped into meaningful 8-bit bytes. Byte framing differs for each type of link employed. The byte framing procedures for asynchronous, synchronous, and parallel links are provided below.

1. Asynchronous Links. DDCMP operates on serial asynchronous links using 8-bit bytes. Byte framing is inherent in the asynchronous operation. An asynchronous link is a communications path that has no fixed time relationship between bytes. When bytes are to be sent, the link is activated. When there is no byte flow the link remains in a steady state. This steady state is called the mark state, 1 state, stop state, or Z condition and by convention is the

binary 1 condition. For sending a byte (or 8 bits) over the line, the framing technique used is based on a start and a stop bit placed at each end of the byte.

The presence of the start bit is recognized by the receiving computer as a change from the 1 to Ø state. This change (a) starts the receiver sampling the line at preset timing intervals and (b) tells the receiver that the next 8 bits will be the data byte followed by a ninth bit, the stop bit. An important consideration is that there is no framing until the start bit is received for each byte.

A potential problem on asynchronous links is that framing may be lost during the transmission of multiple abutting bytes (i.e., where the next start bit immediately follows the preceding stop bit). If the receiver shifts in bit timing, it may time its search for a start bit at the exact interval when a data bit with the same value as the start bit is received. The receiver would then think that the next eight bits were data, look for the stop bit, and wait for the next start bit to reframe. The error will be caught by the block check for the current message but may lead to misframing and missing the next message if uncorrected.

The solution to synchronization on asynchronous links specifies that the transmitter must either send an all ones byte DEL (value 255. or 377 octal) or idle the link for 1Ø bit times when resynchronization is required. This technique will guarantee proper byte framing for the next byte at the receiver. The receiver does not look for this DEL, it simply causes proper framing on the next byte. If the DEL is received, it is ignored when resyncing. On asynchronous links bytes always abut due to their independence from each other in time. So resynchronization is required only for error recovery and is not required preceding messages where the link has just been idle for some time.

2. Synchronous Links. DDCMP operates on serial synchronous links using 8-bit bytes. Bit timing is provided by the modem or superimposed on the data signal. On synchronous links, byte framing is established by searching for a unique 8-bit pattern or sequence in the bit stream. Once this is found every 8 bits forms the next byte. This pattern is called the SYN byte (value 15Ø. or 226 octal). The receiver must locate and lock on to a sequence of two consecutive SYN bytes to achieve byte synchronization. The transmitter must send four or more SYN bytes to allow for the loss from missynchronization and hardware interface constraints. Additional SYN bytes are passed over on receive while searching for the first non-SYN byte. This sequence of 4 or more SYN bytes is the synchronous synchronization sequence.

   Since timing between bytes is determinate, messages must either abut (i.e., the first byte of the next message immediately follows the last byte of the current message with no intervening time) or byte framing is assumed lost following the end of the current message. The next message must reestablish or resynchronize byte framing at the receiver. This resynchronization requires the next message to be preceded by a synchronization sequence. On some communication interfaces (usually block-oriented or DMA type) received data may be buffered in the device and by the time the software driver determines that the next message does not abut, the device may have buffered many bytes further ahead on the link. Therefore, on synchronous links, a long synchronization sequence is required when messages do not abut to account for the potential buffering in the interface. This value is the number passed over or buffered by the device plus 4 more for resynchronization. Current devices and programming techniques have set the long sync sequence to 8 or more SYN bytes. This allows for 4 bytes of buffering in the device followed by the 4-byte SYN sequence. A feature has been included in DDCMP that can be used to reduce the length of this sequence and improve efficiency of the protocol. This is implemented via the QSYNC or quick sync flag present in all messages.

If set in a message, the QSYNC flag notifies the receiver that the next message will not abut and the synchronization sequence preceding the next message may be the short sequence. When the transmitter knows the next message will not abut the current message, it may set the QSYNC flag. The receiver seeing this may set resynchronization on the device immediately following the current message without looking ahead into the next message (and possibly requiring a long synchronization sequence due to device buffering). This allows the receiver to sync on a short sequence. A long sequence may always be used; the additional sync bytes are simply ignored.

3. Parallel Links. If the transmission rate over the link is a multiple of 8 bits, then byte framing is inherent on the link. If the transmission rate is a multiple of some other number of bits, then other means must be sought to achieve byte framing. Such a way is not currently specified by this standard.

5.1.2 Message Framing - Message framing is achieved by searching for one of the three starting message bytes SOH, ENQ, or DLE. One of these bytes must appear immediately after the byte framing sequence or immediately after the previous message's last byte (if abutting). If these bytes do not appear at the receiver in the proper location, then the next message will not abut and byte framing is assumed lost. Once one of these starting bytes is found, the end of the message is determined by a single set of rules for all link types:

1. If the starting byte is SOH or DLE then: the next 5 bytes will complete the message header, followed by 2 bytes of header block check (CRC), followed by COUNT bytes of data (where COUNT is the 14-bit field following SOH or DLE), followed by 2 bytes of data block check.

2. If the starting byte is ENQ then: the next 5 bytes will complete the message, followed by 2 bytes of header block check.

The data field is totally transparent. No pattern searching is done once the starting byte is found. If the header CRC is in error the framing stops and resynchronization will occur (see section 5.3, Message Exchange). If the station is a multipoint tributary and the ADDR in the header does not match the station address the tributary still tracks the message by framing it (see section 5.2, Link Management).

In some cases, where errors caused by the loss of synchronization on synchronous links have resulted in one or more bits being either removed or added, the performance of the CRC block check is not as good as in cases of simple bit changes. To increase the error detection capability in these cases, the following additions should be made to the techniques described above (synchronous only):

Transmission - Whenever idling the link, ensure that the transmission ends with 8 or more 1 bits (DEL bytes). This is a restriction on the optional leader preceding a sync sequence separating messages, and on the trailer, before shutdown, where the modem may require 2 or more DEL bytes.

Reception - (Optional to achieve better detection capability). After receiving the end of a data message with a valid CRC, do not process until one more byte is received. Discard the message (treat as a data CRC error) if this byte is not either: DEL, SYN, SOH, ENQ, or DLE.

5.1.3 Synchronization - Synchronization is the process of establishing both byte and message framing.

5.1.3.1 Receiver Synchronization - Synchronization takes place at the receiver under the following conditions:

1. Initially on receiver start-up, or for half-duplex and multipoint operation on link turnaround or selection (see section 5.2, Link Management).

2. If messages do not abut (i.e., the next abutting byte is not SOH, ENQ or DLE).

3. If the QSYNC flag is set in the current message, resynchronize at the end of the message.

4. If a block check (CRC) error or other error occurs that might have caused synchronization to be lost (e.g., receiver overrun).

The receiver should track the link as much as possible and only resynchronize when synchronization may have been lost. This will increase the efficiency in receiving abutting messages and reduce the chance of synchronizing on a false message inside a data message (the aliasing problem).

5.1.3.2 Transmitter Synchronization - The transmitter will send a synchronization sequence prior to transmitting messages under the following conditions:

1. Initially on transmitter start-up or following link turnaround (half-duplex and multipoint).

2. If a multipoint control station when changing the ADDR in messages to different tributaries.

3. If messages do not abut (synchronous mode only - they always abut in the asynchronous mode).

4. If QSYNC is set in the current message, precede the next message with the sync sequence.

5. In the next message sent after receiving a NAK.

6. Preceding all control (ENQ) messages except ACK. A synchronization sequence will also precede an ACK if one of the other conditions is satisfied.

7. Preceding all maintenance mode (MAINT) messages.

8. Preceding all messages with the SELECT flag on (see section 5.2, Link Management).

9. Preceding the next message if a hardware/driver error (e.g., overrun) occurs while transmitting the current message.

The transmitter should send a synchronization sequence when it believes synchronization may have been lost at the receiving end. Preceding every message with a a synchronization sequence is legal in the protocol but will reduce the potential overall efficiency.

5.1.3.3 Synchronization Rules - Table 5-1 summarizes the DDCMP Synchronization rules.

Table 5-1. Summary of Synchronization Rules

Type of Framing	Mode - Synchronization Rules		
1. Byte Framing			
Asynchronous:	Receiver	-	inherent in transmission mode (none for DDCMP)
	Transmitter	-	Send an all ones byte or idle the link for a 10 bit time interval. Not required on initial message or link turnarounds.
Synchronous:	Receiver	-	Search for 2 adjacent SYN bytes. Strip any more abutting.
	Transmitter	-	Send short sequence if: 1. Initial message (after startup) or link turnaround. 2. QSYNC set in previous message. Send long sequence if: 1. QSYNC not set in previous message (not initial message).
2. Message Framing			

	Receiver	-	Search for SOH, ENQ, DLE immediately following byte framing.
			After SOH or DLE:
			Header = 5 bytes
			CRC = 2 bytes
			Data = COUNT bytes
			CRC = 2 bytes
			After ENQ:
			Header = 5 bytes
			CRC = 2 bytes

Transmitter - Send message immediately after byte framing sychronization sequence.

Notes:

1. The recommended short synchronization sequence is 4 or more SYN bytes.

2. The recommended long synchronization sequence is 8 or more SYN bytes.

3. In response to a NAK, to assure that a receiver is in the reframing mode and has not already framed on an erroneous message, the transmitter may optionally increase the synchronization sequence to:

   a. 8 or more all ones bytes (DEL bytes - 255.) for the asynchronous mode.

   b. 10 or more SYN bytes (150.) for the synchronous mode.

4. A simple implementation may precede all messages with a sync sequence. It may also always send a long synchronization sequence, at some cost in time and efficiency, since extra SYN bytes are ignored.

5. If modems and/or interfaces require PAD sequences to clear them and to assure transmission of the last message byte prior to transmitter shutdown they should use all ones bytes (DEL bytes - 255.) for synchronous and asynchronous modes.

5.2 Link Management

Link management is the process of controlling the transmission and reception of data on links where there may be two or more transmitters and/or receivers actively connected to the same signal channels. This will be true of half-duplex, point-to-point links, as well as full- and half-duplex multipoint links. On half-duplex links, only one transmitter may be active at a time; on full-duplex links, only one transmitter may be active in each direction on the link at a time.

A station on such a link may transmit when it has been selected or granted ownership of the link. This ownership is passed by use of the SELECT flag existing in all messages. A SELECT flag set in a received message allows the addressed station to transmit after completing reception of the message. The SELECT flag also means that the transmitter will cease transmitting after the message is sent, if this is necessary for the proper operation of the link type. The process of receiving a valid SELECT flag and transmitting is called selection. The interval between receiving the SELECT flag, transmitting, and terminating selection by sending a SELECT flag is called the selection interval. A SELECT flag may be sent and received in any DDCMP message. If there is no message to send and a SELECT flag needs to be transmitted, the ACK message is used for sending the flag. A selection timer is used to detect a lost SELECT flag. It is started when a station is selected and reset when valid messages are being received. If it expires, no message was received for a timer interval and it is assumed the messages with the SELECT flag set were either transmitted or received in error.

The length of a selection interval depends on the response and throughput considerations of a particular implementation or system. When a station is selected, it should limit the length of the selection interval to some maximum period, either: a) by using a timer to time the length of the interval, b) by using a counter to count the number of bytes transmitted, or c) by limiting the number of data messages transmitted during the selection interval. If no explicit limit is implemented for half duplex point-to-point and multipoint stations, transmissions will eventually cease due to uacknowledged data messages exhausting buffer capacity or sequence numbers. If no explicit limit is implemented for full-duplex multipoint stations, the selection interval might never terminate, since messages may be acknowleged as other messages are being transmitted. Therefore full-duplex multipoint stations must have an explicit limit to their selection interval.

When there are two or more receivers on a link (multipoint) the ADDR field is used to address the tributary stations. A multipoint configuration consists of a single control station and a number of tributary stations. In a point-to-point configuration both stations are considered control stations. Address "1" is used in the ADDR field on point-to-point links. Addresses 1-255 are used to address tributaries on multipoint links and appear in the ADDR field of all messages both to and from tributaries. Address "Ø" is reserved. A tributary will only accept messages that contain its address (after CRC checks) and will ignore all others. Only SELECT flags set (i.e. on) in messages with the matching tributary address are accepted as selecting the addressed station. The SELECT flag is always set in STRT, STACK, and MAINTENANCE messages for all link types. The start-up procedure and maintenance mode operate in the half-duplex, single message at-a-time mode for consistency on all link types.

5.2.1 Link Management On Full-Duplex Point-To-Point Links - There is effectively no link management on these links. Both stations are control stations and always selected. The address value 1 is used in the ADDR field of all messages. The checking of this address is optional on reception. For all messages other than STRT, STACK, and MAINT, in which the SELECT flag is set, the SELECT flag is optional in the full-duplex point-to-point mode and is essentially ignored. That is, it may be optionally set on transmission but is not checked on reception.

5.2.2 Link Management On Half-Duplex Point-To-Point Links - As in the full-duplex mode, both stations are control stations and use the ADDR value 1. Checking of this on reception is optional. Stations transfer control back and forth by use of the SELECT flag. A station setting the SELECT flag in a message invites the other station to transmit and will shut down its transmitter at the end of the current message after sending a number of DEL pad bytes appropriate to the modem and circuit. A station receiving a valid message with the SELECT flag set will transmit after completion of reception of the current message. The station ends its selection interval with the SELECT flag set in its last message transferring control back to the other station.

When a station sends a SELECT it starts a selection timer. The selection timer is stopped and restarted when a valid data message, maintenance message, or control message is received; or when resynchronization occurs at the receiver. See the message exchange section for the description of a valid message. The purpose of the selection timer is to detect a lost SELECT flag either to or from the other station. It also serves to detect a loss of data signal partway through a message that may not be detected by other components of the system (e.g. loss of signal on an asynchronous link after receiving part of the data field of a message), preventing deadlocking of the protocol due to a link failure. If no message is received when the timer expires, the station assumes the message sent with the SELECT flag was received in error or the message sent by the other station which contained the return SELECT was in error. The station assumes ownership of the link and transmits as if it had received a valid SELECT return. When a station is selected and transmitting, its selection timer is not running. The timer value should be different at both stations to avoid a deadlock race condition. The value of the timer must include such factors as maximum message length, sync sequence lengths, link speed, link turnaround time, and processing delays. The selection timer detects the loss of the SELECT flag by timing the interval it takes to receive the longest message from the other station. It is reset after every message is received. Typical values might be on the order of a few seconds. To avoid excessive overhead, (when there is no data traffic) of constantly turning the channel around and selecting the other station, a station can delay replying to a selection for a short period of time (typical value would be 10-20% of select timer value). The decision to do this and the value of the delay is based on such factors as: response time requirements, message queuing delays, turnaround time, and the duration of the selection timer.

5.2.3 Link Management On Full-Duplex Multipoint Links - DDCMP supports configurations with a single control station and up to 255 tributaries. Messages are only exchanged between the control station and the tributaries. There is no direct tributary to tributary traffic. The tributaries use address values 1-255. These addresses are used in the ADDR field in messages sent both to and from tributaries.

Transmission from the control station to any tributary can be simultaneous with transmission from a selected tributary to the control station. The control station must maintain separate error counters, starting sequence and state transition logic, and data message number sequences for each active tributary. The messages transmitted to a given tributary are independent of the messages sent to any other tributary. A tributary is only allowed to send after it receives a message addressed to it with the SELECT flag on from the control station. The control station is allowed to send data messages (with SELECT off) to either the tributary it has just selected or to any other tributary. The control station is allowed to transmit continuously. A message and associated SELECT flag sent to a tributary are valid only if the data message header CRC, maintenance message header CRC, or control message CRC checks and the ADDR field matches that of the tributary.

The control station uses a selection timer to recover from messages where the SELECT flag has been transmitted or received in error. It operates the same as in the half-duplex, point-to-point case. The tributaries have no timer and wait to be selected again after ending a selection interval.

Tributaries only accept messages with their own addresses. A tributary is selected by receiving a message with its address with the SELECT flag on. It may then transmit and it ends the selection interval by setting the SELECT flag in its last message. While transmitting it may be simultaneously receiving messages addressed to it.

The order in which the control station selects tributaries is not defined in this protocol. The control station might select each station in turn in round-robin fashion, or it might have one or more lists that determine the order and frequency of selection. The lists might be supplied from a higher level process or be developed by selecting all the possible stations to determine which ones are on-line.

A single physical station is allowed to respond to several different addresses on a multipoint channel only if it acts as if it were multiple separate tributaries. That is, the control station does not know that several addresses are located at a single physical station.

Once the control station has selected a tributary, it must wait for either (1) the addressed tributary returning a message with the SELECT flag on, or (2) the selection timer expiring, before selecting another tributary. On data and maintenance messages received by the tributary, the SELECT flag on is valid if the header CRC and ADDR check even if the data CRC is in error. The tributary must wait, however, until the entire message is received before starting transmission.

Tributary stations track all messages on the link by framing them and accepting only those with matching addresses. They resynchronize only on non-abutting messages or messages with CRC errors.

5.2.4 Link Management On Half-Duplex Multipoint Links - These links operate in the same way as the full-duplex multipoint links except that the control station must cease transmitting when it selects a tributary. It regains control only when (1) it receives a SELECT flag from the addressed tributary or (2) the selection timer expires. The control station will operate in the same manner as the half-duplex point-to-point control station does, except that the ADDR field addresses one specific tributary rather than the only other station on the link. The tributary stations will operate as if they were full-duplex tributaries, except, they do not receive while transmitting.

5.2.5 Link Management Summary Rules - Table 5-2 summarizes the major modes of operation.

Table 5-2. Link Management Summary

MODE	SUMMARY
Full-Duplex Point-to-Point Control Station	- Always selected - Uses ADDR = 1 - Checks ADDR on receive (optional) - SELECT flag set (on) in STRT, STACK, and MAINTENANCE messages - SELECT flag ignored in other messages (may be set but not checked on receive) - No selection timer
Half-Duplex Point-to-Point Control Station	- Selected alternately - Uses ADDR = 1 - Checks ADDR on receive (optional) - SELECT flag turns link around and selects other station - SELECT flag set (on) in STRT, STACK and MAINTENANCE messages - Selection timer used by both stations. Selection timer rules: 1. Start when sending SELECT to select other station. 2. Stop and restart when a valid data message, maintenance message, or control message is received.

Table 5-2. Link Management Summary (cont'd)

MODE	SUMMARY
	3. Stop and restart when resynchronization occurs at the receiver. 4. Stop timer when receiving a SELECT (being reselected). 5. If the timer expires, treat as if a valid SELECT had been received.
Full-Duplex Multipoint Control Station	- Always selected - Uses tributary address in ADDR field (1-255) - Checks for proper tributary ADDR on receive - SELECT flag set (on) in STRT, STACK, and MAINTENANCE messages - Start selection timer when selecting a tributary - Timer operates as indicated for Half-Duplex Point-to-Point mode - Select another tributary when SELECT is received or selection timer expires
Full-Duplex Multipoint Tributary Station	- Selected on receipt of SELECT where CRC and ADDR checks - Ends selection with SELECT set in last message - Uses its own address in ADDR field - Accepts received messages with matching ADDR. - SELECT flag set in STRT, STACK, and MAINTENANCE messages. - No selection timer for tributaries
Half-Duplex Multipoint Control Station	- Selected alternately with tributaries - Use tributary address in ADDR field - Checks for proper tributary ADDR on receive - SELECT flag set in STRT, STACK, and MAINTENANCE messages. - Selection timer used when selecting a tributary (same as for full-duplex multipoint control stations)

Table 5-2. Link Management Summary (cont'd)

MODE	SUMMARY
Half-Duplex Multipoint Tributary Station	- Same as Full-Duplex Multipoint Tributary station except that it does not receive while transmitting

5.3 Message Exchange

Once framing and link management have been handled, messages are exchanged by the DDCMP processes to create a protocol that ensures the sequentiality and integrity of data sent via DDCMP. The basic concepts of this exchange were presented in section 2.5 Functional Organization. This section presents the details of that message exchange mechanism.

Before discussing the actual message exchanges, three concepts must first be presented: (1) Initialization, (2) Reply Time-outs, and (3) Valid messages.

5.3.1 Initialization - DDCMP can be operated in two modes: (1) the on-line or running mode and (2) the off-line or maintenance mode. The on-line mode creates the sequential error-free link. The off-line or maintenance mode provides a block check on the data, DDCMP framing, and link management procedures. The on-line mode is entered via a DDCMP start sequence. This startup or initialization resets the message numbering, clears out any outstanding messages and prepares for the start of data message transfers. The start sequence is designed so that both stations must enter the start sequence before either station can complete the sequence and the message exchange can commence. Startup or restart is initiated by the user of DDCMP when it first starts up, on a fatal error, usually on a persistent error, and wherever else restarting is appropriate (See Section 6.Ø, Error Recording). DDCMP does not initiate startup itself.

5.3.2 Reply Time-outs - A necessary component of DDCMP is the reply time-out. The replies to some transmitted messages are timed and if there is no response within a time-out interval the expiration of the timer triggers some action. The time-out is necessary to recover from outages and messages distorted by the link such that even framing may be lost. Time-outs prevent the protocol from being deadlocked.

The timer is keyed to the selection of the other station. That is, the other station is given a chance to respond during a selection interval and if no proper response is received before the end of the selection interval then error recovery is initiated. In full-duplex point-to-point links, where the other station is always selected, a clock is used as the timer. On the other link configurations, the time-out interval is set to be one selection interval and expires at the end of the next selection of the other station. A station is given one interval to reply. A more detailed description of the selection process is given in section 5.2, Link Management.

Reply timer interval:

The reply timer controls the maximum period a station will wait between sending a message and receiving a proper response before taking error recovery action. The timer interval for each station type is as follows:

a. Point-to-Point

Full-duplex: uses a real clock. The clock period is the same as that of the selection timer used in other modes. It must include link delays, turnaround, processing and message transmission times. A typical value is 3 seconds.

Half-duplex: next selection interval. The other station is given a selection interval in which to respond. The other station is selected, it transmits (if it received the selection correctly), and selection is ended (by the other station sending a select or by the selection timer expiring). If the proper reply is not returned in that interval the reply timer is assumed to have expired.

b. Multipoint

Full-duplex-Control Station: Next complete selection interval. A tributary is given at least one complete selection interval in which to reply. For messages transmitted to a tributary which is not selected, operation is as described for half-duplex point-to-point (see above). For messages transmitted to a tributary which is selected, the tributary is expected to reply prior to the ending of the next selection interval rather than prior to the ending of the current interval. That is, a reply timer started during a selection interval does not expire until the end of the next selection interval, not the current interval.

Full-duplex-Tributary station: before the next selection. The tributary station expects the control station to reply in the interval starting with the tributary sending the message requiring the reply, completing the current selection, and ending with the tributary being selected again by the control station.

The message that selects the tributary again is included in that interval and may contain the valid reply. If there is no proper response within that interval between selections the tributary assumes the timer has expired.

Half-duplex-Control station: the next selection interval. Same as described above for half-duplex point-to-point stations.

Half-duplex-Tributary station: before the next selection. Same as described above for full duplex multipoint tributary stations with the addition that the reply cannot be received before ending the current selection due to the half-duplex mode.

Only point-to-point full-duplex control stations use a real clock; all others key reply time-out to a selection interval as indicated. Half-duplex and multipoint control stations use a real clock to time the selection interval. Tributary stations have no clock and rely on the control station for timing intervals (selection).

5.3.3 Valid Messages - Only valid messages are processed by the protocol. A message that has been properly framed is checked for:

a. Good block checks (header and data).

b. Valid message TYPE and SUBTYPE - control messages.

c. Matching ADDR for multipoint stations.

Optionally, a station may check for:

a. FILL fields for being zero.

b. ADDR=1 for point-to-point stations.

Multipoint tributary stations must ignore messages with header block check errors, due to the uncertainty of the ADDR field, and rely on time-outs for recovery. Control stations (point-to-point and multipoint) may process messages with header block check errors (e.g. reply with a NAK), as they would process messages with data block check errors.

Additional checks are specific to each message and are described in the state tables where appropriate.

5.3.4 Message Exchange Variables And States - The following states and variables are used in the message exchange state tables and descriptions. On multipoint links, there is a set of states and variables for each control station - tributary station pair. A multipoint link appears as multiple point-to-point links operating on a single physical channel. Arithmetic and comparisons between these variables are always modulo 256.

5.3.4.1 Data Variables -

R - the number of the highest sequential data message received at this station. Sent in the RESP field of data messages, ACK messages, and NAK messages as acknowledgment to the other station.

N - the number of the highest sequential data message transmitted by this station. Sent in the NUM field of REP messages. N is the number assigned to the last user transmit request which has been transmitted (sent in the NUM field of that data message).

A - the number of the highest sequential data message that has been acknowledged to this station. Received in the RESP field of data messages, ACK messages, and NAK messages.

T - the number of the next data message to be transmitted. When sending new data messages T will have the value N+1. When retransmitting T will be set back to A+1 and will advance to N+1.

X - the number of the last data message that has completed transmission. When a new data message has been completely transmitted X will have the value N. When retransmitting and receiving acknowledgments asynchronously with respect to transmission, X will have some value less than or equal to N.

Other variables are as defined in the message format section and refer to fields in specific messages (e.g., ACK(RESP=Ø) refers to the RESP field of an ACK message with value = Ø).

Relationship of data variables(modulo 256. arithmetic):

A <= N The data message numbers from A+1 to N are the current unacknowledged data messages. ACKs within this range are valid and accepted. ACKs outside this range are ignored.

A < T <= N+1 If new data messages are being sent T = N+1. If retransmissions are being sent T will lie between A and N+1. If an ACK is received that increments A, T is set to A+1 to start retransmission, possibly skipping some data messages in a retransmission sequence already in progress.

X <= N The last data message transmitted is always less than or equal to the highest sequential data message transmitted (N) and may be less than the highest one acknowledged (A) due to retransmissions and the NAKing of control message CRC errors.

5.3.4.2 Control Flag Variables -

These flags control the sending of DDCMP control messages. They are indicators specifying which control messages to send when the transmitter is idle.

SACK - send ACK. This flag is set when either R is incremented, meaning a new sequential data message has been received which requires an ACK reply, or a REP message is received which requires an ACK reply. The SACK flag is cleared when sending either a DATA message with the latest RESP field information, an ACK with the latest RESP field information, or when the SNAK flag is set.

SNAK - send NAK. This flag is set when a receive error occurs that requires a NAK reply. It is cleared when a NAK message is sent with the latest RESP information, or when the SACK flag is set.

The SACK and SNAK flags are mutually exclusive. At most one will be set at a given time. The events that set the SACK flag, R is incremented or a REP is received requiring an ACK, also clear the SNAK flag. Similarly, the events that set the SNAK flag, reasons for sending a NAK (see 5.3.7), also clear the SACK flag. Setting or clearing a flag that is already set or cleared respectively has no effect. For the SNAK flag, a reason variable (or field) is also maintained which is overwritten with the latest NAK error reason. Whenever the SNAK flag is set the NAK reason variable is set to the reason for the NAK.

SREP - send REP. This flag is set when a reply timer expires in the running state and a REP should be sent. It is independent of the SACK and SNAK flags.

It is desirable for DDCMP implementations to implement the sending of control messages via the use of these control flags. Events that require control messages to be sent will overwrite previous events, for which the control messages have not yet been sent, and only the necessary control messages will be sent. In general, this will increase the performance of DDCMP by eliminating the sending of unnecessary control messages which add to protocol overhead and may result in unnecessary retransmissions. It is only necessary to have a single ACK or NAK, and/or a single REP marked for transmission with the latest information. A station when selected to transmit must send all pending control messages (i.e., clear all pending control flags) before deselecting itself.

It is also possible to implement the sending of control messages via a queuing mechanism, which actually builds and adds control messages to the transmit queue. In this case all events that would cause a control message to be sent must generate a message for the queue.

5.3.4.3 DDCMP States -

HALTED - protocol stopped and not running. The user can halt the protocol anytime by giving it a stop command (see user interface description). Commands to start cause a transition through halted first.

STARTING - an attempt is being made to initialize the protocol via an exchange of STRT and STACK messages. This state has two internal states.

1. ISTRT (Initiate Start) sends the STRT message.

2. ASTRT (Acknowledging Start) sends the STACK message.

RUNNING - signifies DDCMP is in the on-line mode exchanging data messages.

MAINTENANCE - signifies DDCMP is in the off-line mode sending and receiving maintenance messages.

5.3.5 Message Queuing, Reply Timers, And Buffering -

Messages are passed from DDCMP to a device driver for actual transmission on the link. They may be passed either one message at a time, waiting for a transmit complete before passing the next one to the driver; or more than one message may be passed to the driver, letting the driver maintain a transmit queue. One message at a time is the most efficient method from the message exchange viewpoint, since it defers decisions on which message to transmit next to the latest possible instant, making use of the latest available acknowledgment information. This technique, however, may not be optimal for implementations requiring high performance via queuing and pipelining techniques. The state tables for DDCMP message exchange operation assume single message at a time operation. If queuing techniques are used in an implementation, the operation described in the state table as "send" will mean "add to transmit queue" or "pass to device driver".

The reply timer on full duplex point-to-point links uses a real clock. The description of the message exchange operation assumes the timer is started after the message has been completely transmitted. In this case, the timer value must include link delays and processing time, but is independent of the length of the transmitted message. If the timer is started, however, when the message is passed to the driver it must also include the time to actually transmit the message. If many messages are being queued to the driver, then the timer must also include the time to transmit all the messages ahead on the queue. These increases in the timer value make it less precise and, therefore, it does not perform as well as a reply timer that is started after transmission. The state table describes the reply timer starting at two possible times: one when messages are actually transmitted and the other when they are passed to the driver. The operation affects the setting of the variable X and the starting of the reply timer. The choice of technique in a particular implementation depends on the split of functions between the driver and DDCMP, the sharing of the protocol data base, and the capabilities and environment presented by the operating system. All other factors being equal, the best technique uses queuing for high performance and starts the timer after actual message transmission. On other than full-duplex point-to-point links the timer is keyed to station selection. The driver transmit queue is always emptied before ending selection, and, thus, has no effect on the reply timing.

A message must not be marked complete and returned to the user until the message is acknowledged. If the user interface does not separate the completion or acknowledgment of data messages from the returning of buffers to the user (e.g. transmit buffering is supplied by the user), then the message must not be returned to the user while it is still residing on the transmit queue. This is required to prevent buffers from being released prematurely and given back to the user while they are still being used for transmission. If the user interface separates the completion of data messages from the returning of buffers (e.g. transmit buffering is supplied by DDCMP, the message is copied to a local DDCMP buffer), then the message may be marked complete to the user as soon as it is acknowledged, even though the actual transmit buffer (local to DDCMP) may still be on the transmit queue (i.e., being retransmitted).

5.3.6 Reply Timer Operation Rules -

1. Start timer:

   When timer is a clock - set clock to the interval value, start it running. When interval expires clock will issue a signal and stop.

   When timer is another event (e.g. ending selection interval) - set flag marking timer as running. When event occurs and flag is set, timer will issue a signal and turn flag off.

   If timer is running when start command is issued, it is first stopped and then started.

2. Stop timer:

   When timer is a clock - clock is halted, no signal is issued. The timer can be set running via the start timer command.

   When timer is another event - turn flag off.

5.3.7 NAK Reasons -

NAKS are returned in response to message, device, and buffering errors in the running state.

Specifically the NAK reasons are (Note: the number is the NAK error reason code):

1. Header block check error - a data message header CRC or a control message CRC is in error.

2. Data field block check error - a data message data field CRC is in error.

3. REP response - the NAK is sent in response to a received REP message where the NUM field in the REP did not equal R (the last message received).

8. Buffer temporarily unavailable - a buffer was not available to store the incoming data message. Either the user did not allocate one in time, or the buffer pool was empty.

9. Receive overrun - The receiving hardware and/or software was not able to respond fast enough to incoming bytes and an overrun occurred.

16. Message too long - a received data message (COUNT) was too long for the allocated buffer.

17. Message header format error - the header CRC checked but one or more header fields was invalid. Possible errors are:

    a. Invalid SELECT flag,

    b. Invalid ADDR value (optional check for point-to-point),

    c. FILL fields not zero (optional check),

    d. Invalid control TYPE or SUBTYPE value.

5.3.8 Startup -

Startup is the process of initializing the protocol states and variables, and synchronizing both stations on a link. Table 5-3 is the startup state table.

Startup Notes:

1. Implementations may ignore messages in error (invalid messages) or messages other than that expected and wait for the reply timer to expire during the start sequence to initiate recovery action.

2. The SELECT flag is always set in STRT and STACK messages, the starting sequence being an alternate exchange, one message at a time. For all stations, except multipoint control stations, the receipt of a STRT or STACK will trigger an immediate response due to selection by the start sequence messages. Multipoint control stations need not respond immediately to the starting tributaries, but may select and send messages to other tributaries before returning and completing startup with tributaries waiting for start sequence replies.

3. After startup is complete the data variables have the following values: R=Ø, N=Ø, A=Ø, T=1, X=Ø. The control flag variables are all clear.

Table 5-3. Startup State Table

STATE	EVENT	NEW STATE	ACTION
any state	User requests halt	HALTED	None-stop timer if running
HALTED	User requests startup	ISTRT	Send STRT- reset variables start timer
	User requests Maintenance Mode	MAINTENANCE	See section 5.4 Maintenance Mode
ISTRT	Receive STACK	RUNNING	Send ACK (RESP=0) stop timer if running
	Receive STRT	ASTRT	Send STACK- start timer
	Timer expires	ISTRT	Send STRT- start timer
	Receive MAINT message	MAINTENANCE	See section 5.4 Maintenance Mode
	Message in error or other message received	ISTRT	Either: Send STRT, start timer; or ignore (do nothing)
ASTRT	Receive ACK (RESP=0) or Receive Data msg (RESP=0)	RUNNING	See Data Transfer stop timer
	Receive STACK	RUNNING	Send ACK(RESP=0) stop timer
	Receive STRT	ASTRT	Send STACK- start timer
	Timer expires	ASTRT	Send STACK- start timer
	Receive MAINT message	MAINTENANCE	See section 5.4 Maintenance Mode
	Message in error or other message received	ASTRT	Either: Send STACK, start timer; or ignore (do nothing)

Table 5-3. Startup State Table (cont'd)

STATE	EVENT	NEW STATE	ACTION
RUNNING	Receive STRT	HALTED	Notify user of STRT received
	Receive MAINT	MAINTENANCE	See section 5.4 Maintenance Mode
	Receive STACK	RUNNING	Send ACK (RESP=R)

5.3.9 Data Transfer -

This is the process of sending data messages, waiting for positive (ACK) or negative (NAK) acknowledgment, retransmitting on NAK, and sending REP on reply timeout to cause an acknowledgment to be returned. Table 5-4 is the state table for the RUNNING State. These events always leave the stations in the RUNNING state. The entries that change from RUNNING to the other states are listed in the Startup State Table above.

Running State Notes:

1. All message numbering is modulo 256 (i.e., after message number 255 is message Ø, then 1 and so on). For example 3>25Ø is correct where 3 follows messages: 255,Ø,1,2.

2. An ACK always sends RESP=R. A NAK always sends RESP=R and an appropriate NAK reason. A REP always sends NUM=N. A data message always sends RESP=R, NUM=assigned sequential number. A retransmitted message always uses the same NUM value and DATA field as the original message, but sends the latest receive value in the RESP field.

3. The maximum number of outstanding unacknowledged messages is 255. No more can be sent until some are acknowledged. It is always required that (A+255) >= N (modulo 256).

4. Positive acknowledgments can be sent in the RESP field of data messages transmitted in the reverse direction, in ACK messages, or in NAK messages. They are all equivalent on receive to providing a positive acknowledgment for outstanding messages. In the state table, ACK refers to any of the above forms of acknowledgment.

5. The order for transmitting messages in the running state is: NAK, REP, DATA messages, ACK.

6. A data message contains four independent pieces of information:

a. Data consisting of the NUM, COUNT, and DATA fields.

b. An ACK response consisting of the RESP field.

c. Link management information consisting of an ADDR and SELECT flag.

d. Framing information consisting of the QSYNC flag.

These are independent of each other and treated separately. Thus, a received data message is treated as both a received data message and a received ACK. For example, even if the DATA CRC is in error the RESP field in a good header is still accepted and processed.

7. When a message is received which starts or ends a selection interval (SELECT flag set), the message exchange fields (RESP, NUM, COUNT, DATA) are processed according to the message exchange rules (Table 5-4) prior to the starting or ending of the selection interval. For example, a RESP field which acknowleges one or more outstanding messages will complete those messages and stop the reply timer prior to processing the SELECT flag. IF the SELECT flag were processed first, the reply timer might erroneously expire.

8. If there is no message to send in the RUNNING state, and a SELECT flag wants to be sent, use the ACK message for this purpose. ACKs are legal at any time and may be used for time fill and to select and/or terminate selection.

9. Messages should never be truncated on transmission. A station should always finish transmitting the current message before starting a new one, including retransmissions.

10. The SELECT flag can be set in any message. All outstanding control messages must be sent before a station ends a selection interval by sending a SELECT flag. A selection interval is ended when the message with the SELECT flag set is added to the transmit queue (given to the driver for transmission), even though the message has not yet been transmitted on the physical link.

11. The transmitter is idle when it is permissible to pass a message to the driver. It is busy when this is not permissible. For non-queued transmission, passing a message to the driver is permissble only when nothing is being transmitted and the station is selected. For queued transmission, passing a message to the driver is permissible only when the driver will accept a message (queuing space available) and the station is selected.

The message with the SELECT flag set is the last message passed to the driver in a selection interval. The transmitter will remain busy after passing this message to the driver until another selection interval is started by receiving a message with the SELECT flag set. This technique is necessary to maintain the proper relationship between the reply timer and station selection.

12. The notation "A<-B" is the notation for the assignment statement which sets the value of the variable A to the value of the variable B.

Table 5-4. Running State Table

EVENT	ACTION
Receive Data Msg (NUM = R+1) (Also see Receive ACK)	If buffer available, R <- R+1, give msg to user, set SACK flag; otherwise set SNAK flag, (reason 8. or 16.)
Receive Data Msg	Ignore
Receive message in error	Set SNAK flag, see NAK reasons (Section 5.3.7)
Receive REP (NUM = R)	Set SACK flag
Receive REP (NUM not= R)	Set SNAK flag, (reason 3)
Receive ACK, or Data Msg (A < RESP <= N)	For all messages (A < NUM <= RESP), complete msg to user, A <- RESP If T <= A, then T <- A+1 If A < X, start timer If A >= X, stop timer
Receive ACK or Data Msg (RESP <= A or RESP > N)	Ignore

Receive NAK
(A <= RESP <= N)
	For all messages (A < NUM <= RESP), complete msg to user,
	A <- RESP
	T <- A+1, stop timer

Receive NAK (RESP < A or RESP > N)	Ignore
Reply timer expires	Set SREP flag

Table 5-4. Running State Table (cont'd)

EVENT	ACTION
Transmitter is idle and SNAK flag is set	Send NAK with reason value, clear SNAK flag
Transmitter is idle, SNAK flag is clear, SREP flag is set	Send REP, clear SREP flag, * start timer
Transmitter is idle, SNAK and SREP flags are clear, T < N+1	MSG(T) is retransmitted, * X <- T, * if timer not running start timer T <- T+1, clear SACK flag
User requests message to be sent: T < N+1, transmitter is busy, SNAK flag is set, or SREP flag is set	User waits until: T = N+1, transmitter is idle, SNAK flag is clear, and SREP flag is clear

User requests message to be sent, T = N+1, transmitter is idle, SNAK and SREP flags are clear	NUM <- N+1
	Send MSG(NUM)
	N <- NUM
	T <- N+1, clear SACK flag
	* X <- N,
	* if timer not running start timer

Transmitter is idle, SNAK and SREP flags are clear, T = N+1, no user requests waiting, SACK flag is set	Send ACK, clear SACK flag

* If messages are queued for transmission and the timer is started and stopped after actual transmissions of messages are completed rather then when messages are queued, then ignore the starred (*) actions listed above for the events: Transmitter is idle (SREP set), Transmitter is idle (T < N+1), and User requests message to be sent (T = N+1) and add the following events and actions:

Data message (NUM) transmission completed on link	X <- NUM If A < X and timer not running, then start timer If A >= X, then stop timer
REP message transmission completed on link	start timer

5.4 Maintenance Mode

Maintenance mode operation is used where the code requirements necessitate minimal coding and compatibility with the DDCMP message structure. This is true of bootstrap, dumping, and testing facilities where the code might reside in read-only memory (ROM). Compatibility is necessary for running on multipoint channels where one station may be in the maintenance mode and the other stations in the on-line or running mode. By using the same structure both functions can occur concurrently on the link using the same control station protocol.

The maintenance mode utilizes the framing and link management portions of DDCMP without having the complexity or the functionality of the message exchange facility. That is, it creates a frame or envelope for transmitting and receiving message blocks and providing a CRC-16 error detection check. In this mode, there is no message sequencing or acknowledgment. Sequencing and acknowledgment must be handled by the user of the maintenance mode if necessary.

Maintenance mode messages always have the SELECT flag set, so they operate in the alternate half-duplex mode on both point-to-point and multipoint channels. The link management function operates in the same manner as described for on-line mode except that only a single message is transferred before ending selection. Transmit complete is returned to the user immediately after the message is transmitted, since there is no ACK returned to guarantee successful delivery over the link.

The maintenance message header is similar in format to the data message header except the RESP and NUM fields are zero FILLs (since maintenance messages are neither numbered nor acknowledged). The maintenance mode is entered via either a command from the user or a received maintenance message. To return to the on-line (or running mode) the protocol must first go through the halted and starting states. Both operating modes cannot be mixed within a single station-to-station pair. The exact details of how maintenance mode is entered is implementation specific and may vary in different systems. If a maintenance message is received while not in maintenance mode, some implementations may enter maintenance mode and present the message to the user. Other implementations may discard the message, enter the HALTED state, and provide an indication that a maintenance message was received. In the maintenance mode, the protocol adds the header and block check to transmitted messages and starts a select timer if necessary for proper operation of the channel (e.g., multipoint). On received messages, the protocol removes the header and checks the block check. If in error it may either discard the message or notify the user that a message was received with a block check error. The latter action simply improves responsiveness rather than waiting for the timer to expire.

The maintenance mode is used for functions such as bootstrapping, dumping, and link testing. If sequencing or acknowledgment are necessary it must be done within the data field of the maintenance messages. In DECnet, the maintenance message protocol is the

Maintenance Operation Protocol (MOP) and is not part of this standard. Maintenance mode messages are not guaranteed to be delivered or have the assurance that they will arrive in the proper sequence. If they are delivered, however, they will have been block checked within the error limits of the CRC-16 check. DDCMP ignores all non-maintenance mode messages received while it is in the maintenance state.

Table 5-5. Maintenance State Table

STATE	EVENT	NEW STATE	ACTION
MAINTENANCE	Receive MAINT message	MAINTENANCE	If buffer available give to user, otherwise ignore
	Message in error or other message received	MAINTENANCE	None
	User requests MAINT message to be sent	MAINTENANCE	If transmitter idle: send MAINT message, If transmitter busy: wait until idle

Note: Transmitter busy and idle conditions have the same meaning as described in the RUNNING State notes (section 5.3.9).

6.0 ERROR RECORDING

Link and protocol error recording in DDCMP is implementation specific. This section presents recommendations on how error recording should be performed within DDCMP implementations. The actual techniques used and statistics maintained depend on the system environment and requirements of the application. These recommendations should be used as a guideline in developing those techniques. It is recommended that error recording be employed to compile statistics on the condition of the link, determine overall error rates and detect a malfunctioning link. The protocol has been designed so that even if one of the stations on the link cannot record errors, the other station can be used to record errors for all communications sent in both directions on the link. This is accomplished by use of the NAK reason field. For most errors, the error is recorded locally and a NAK is returned with the error reason. The other station, upon receiving the NAK, can record the error for the remote station. On occasion, a NAK will be lost, but this should not noticeably affect the record being kept of a given link.

DDCMP Error Counters can be divided into three groups: Threshold Counters, Cumulative Counters, and Background Counters. Threshold counters are used to detect persistent error conditions; cumulative counters record overall error statistics; and background counters provide a base on which to evaluate the cumulative counters. It is suggested that the user of DDCMP record these values into a permanent log, thereby establishing a long-term record on the data link and communication equipment.

A station that has no means for reading out its counters, or whose controlling user cannot maintain such records, would not maintain them (e.g. a transaction terminal). It is strongly recommended, however, that at least one station on a link maintain error statistics. On multipoint links the control station will maintain a separate set of counters for each control-tributary pair. When the number of counters represents an excessive burden on an implementation, it may combine the counters into groups in order to reduce the memory and processing requirements. In the maintenance mode, error counters and records need not be maintained.

6.1 Threshold Counters

The threshold counters are used to notify the protocol user of a persistent error condition. Threshold counters operate by counting consecutive errors of a given class and will trigger a notification to the user when the threshold value has been reached. A threshold value of seven events is recommended. However, a value that is more appropriate for the implementation environment may be utilized.

Each counter is cleared when the threshold is reached or when the operation it is monitoring is performed correctly. Threshold counters may also be cleared by a start command from the user. Error threshold counters can be used to monitor the following operations:

1. Messages sent to remote stations - counts NAKs received and response timer expirations; cleared on ACKs (explicit or implicit) received.

2. Messages received from remote station - counts errors that cause NAKs to be sent; cleared on data message received.

3. Selection errors for multipoint and half-duplex control stations - counts no response or wrong response to select; cleared on a proper select received.

In the RUNNING state, counter 1 is incremented when a NAK is received (which causes one or more outstanding messages to be retransmitted) or when a REP is sent. The counter is cleared when an unacknowledged message is acknowledged (whether or not there are still other unacknowledged messages). In the Initiate Start state (ISTRT), counter 1 is incremented when a STRT is sent and cleared when a STRT or a STACK is received. In the Acknowledging Start state (ASTRT), counter 1 is incremented when a STACK is sent and cleared when an ACK or a STACK is received.

Counter 2 is incremented for all DDCMP NAK reasons (see operation section 5.3.7). Tributary stations do not count header CRC errors as they cannot determine if the address was their own. Counter 2 is cleared when data messages are successfully received.

Counter 3 is only required in a multipoint control station or a half duplex point-to-point station. This counter is incremented when one of the following errors is detected.

1. Message Received from Wrong Tributary (does not apply to point-to-point operation.)

2. No Select Received and select timer expires.

3. Tributary or half-duplex station exceeds an implementation-dependant maximum total transmission interval without deselecting itself.

Counter 3 is cleared when a message with the select bit set is received from the selected tributary or from the other station if a half-duplex point-to-point link.

It is recommended that, whenever a threshold counter reaches its threshold value, causing DDCMP to notify the user, the user read out the cumulative counters. This will establish a base in determining which specific error has caused the next threshold counter overflow.

DDCMP continues to operate across threshold counter overflows. It clears the threshold counters and counts again, notifying the user on subsequent overflows. Protocol and link shutdown is left up to the user.

6.2 Cumulative Counters

The cumulative counters record and total all occurrences of an error or receipt of a NAK. The contents of these counters should be accessible to the user (to determine specific threshold counter overflow and statistics on network operation). It should be the responsibility of the user to read and clear these counters and to accumulate the counts.

In DECnet, a higher level network maintenance process accumulates the total of each of these counts and provides them for network management.

The cumulative error counters should be able to record an adequate number of occurances of an error to be useful in the maintenance of the link. The cumulative error counters should count to their maximum value and hold at that value until the higher level process clears them.

Cumulative error counters may be arranged in a number of different configurations (e.g., specific individual counters or general group counters) depending upon the specific implementation requirements. The following list summarizes the types of counters that could be maintained:

1. Counters for received NAK reasons (either individually by reason or in groups),

2. Counters for received errors (i.e., errors that cause a NAK to be sent),

3. Timer Counters resulting in a REP message being sent or received,

4. Hardware error indicator counters (e.g., hardware framing errors, transmitter overrun),

5. Selection Timer Counters (e.g., wrong tributary responding, no ending select received.

6.3 Background Counters

Background counters are used to provide a statistical base for the cumulative error counters. Like the cumulative error counters, their usage depends on implementation requirements. It is recommended that two background counters be maintained: one to record the number of data messages sent (count all attempts), a second to record the number of data messages received (exclude duplicates and out of sequence messages). An implementation may use additional counters for debugging purposes or for additional statistics or control.

APPENDIX A

Glossary

Asynchronous Serial Data Transmission: A technique in which the information required to determine when each byte (bit grouping) begins is sent along with the symbol (the "start" and "stop" bits). The time interval between successive bytes is unspecified.

Channel: The data path joining two or more stations, including the communications control capability of the associated stations.

Control Station: A station that has overall responsibility for orderly operation of the channel.

Duplex Channel: A channel over which simultaneous (in time) communications in both directions (to and from the station) is possible. Also called a full-duplex channel.

Half-Duplex Channel (alternate Simplex): A channel that permits two-way communications, but in only one direction at any instant.

Master Station: A station that has control of a channel at a given instant, for the purpose of sending data messages to a slave station (whether or not it actually does). Also referred to as a "transmitting station".

Multipoint Channel: A channel connecting more than two stations.

Parallel Data Transmission: A data communication technique in which more than one code element (e.g. bit) of each byte is sent or received simultaneously.

Point-to-Point Channel: A channel connecting only two stations.

Serial Data Transmission: A data communication technique in which the code element (bits) of each encoded symbol of the data are sent and received one after the other (serially), rather than simultaneously.

Slave Station: A station to which a master station intends to or does send a data message. Also referred to as a "receiving station".

Station: An independently controllable configuration of logical elements (e.g., a computer) to or from which messages are transmitted on a channel.

Synchronous Serial Data Transmission: A data communication technique in which the information required to determine when each byte begins is sent at the beginning of a group of bytes (the "sync bytes"). The time interval between successive bytes in the group is zero. The time interval between successive groups of bytes is unspecified.

Tributary Station: A station on a channel that is not a control station.

APPENDIX B

Formal Syntax Definition

B1.0 Symbology

In the following formal definition of the protocol grammar, the symbols have the following meanings:

1. < > denotes a metalinguistic variable.

2. := means "is produced by" or "has the value of".

3. <a><b> means "a followed by b" or "b concatenated with a".

4. <a>!<b> means "a OR b" (exclusive OR).

5. quantities not surrounded by brackets < > are constants or literal values.

6. (<a>*b) means "a repeated b times" (<a><a>---<a>).

7. <a>** means "a occurs from zero to infinity times in succession".

8. null means "the empty set".

B2.0 Message Syntax

The following definition of the protocol does not include the specific sync sequences and rules when they are used on each link type. Refer to the operation section for these specifics.

<protocol>:=<transmission><trailer>!<transmission><protocol>

<transmission>:=<msg>!<syncseq><msg>

> Note that the form <syncseq><msg> is required in numerous cases. See operation section 5.1.3.2.

<syncseq>:=<leader><sync>*m

> Where m is a parameter determined by hardware and interchange considerations. (m >= 1 for asynchronous circuits, m >= 4 for synchronous circuits.)
>
> Note that <syncseq> is used for inter-message padding as well as for synchronizing.

```
<sync>:=<syn>                      for synchronous circuits
      :=10-bit idle line ! <del>   for asynchronous circuits
      :=null                       for parallel circuits

               Where "10-bit idle line" means that the channel is held
               continuously in the state of the stop element (i.e.
               Mark, condition Z, 1) for a period not less than 10 bit
               times.

<leader>:=(<one>*j)(<sync>*k) for synchronous circuits

               Where j+8k >=  0  if  qsync  is  set  in  the  previous
               message.  This is a short sync sequence.

               Where j+8k >= 32 if qsync is not set  in  the  previous
               message.  This is a long sync sequence.

               Either j=0 or j>=8

        := idle line !  <del>**   for asynchronous circuits

               Where "idle  line"  means  that  the  channel  is  held
               continuously  in  the  state  of the stop element (i.e.
               Mark, condition Z, 1)

        :=null   for parallel circuits

<trailer>:=<one>*j             for synchronous circuits, where j>=8
         :=<leader>            for asynchronous circuits
         :=null                for parallel circuits

<one>:=1

<msg>:=<nummsg>!<unnummsg>!<maintmsg>

<nummsg>:=<soh><header><bcc><data><bcc>

<unnummsg>:=<enq><body><bcc>

<maintmsg>:=<dle><mhdr><bcc><data><bcc>

<header>:=<count><qsync><select><resp><num><addr>

<data>:=(<byte> * value of <count>)

<bcc>:=<byte><byte>

<body>:=<ackm>!<nakm>!<repm>!<strtm>!<stackm>

<mhdr>:=<count><qsync1><select1><fill><fill><addr>

<count>:=(<bit>*14)

<select>:=<bit>
```

```
<selectl>:=l

<qsync>:=<bit>

<qsyncl>:=l

<resp>:=<byte>

<num>:=<byte>

<addr>:=<byte>

<ackm>:=<ack><fill6><qsync><select><resp><fill><addr>

<nakm>:=<nak><rnak><qsync><select><resp><fill><addr>

<repm>:=><rep><fill6><qsync><select><fill><num><addr>

<strtm>:=<strt><fill6><qsyncl><selectl><fill><fill><addr>

<stackm>:=<stack><fill6><qsyncl><selectl><fill><fill><addr>

<rnak>:=000001!000010!000011!001000!001001!010000!010001

<byte>:=00000000!00000001!...!11111111

<bit>:=0!1

<syn>:=10010110

<soh>:=10000001

<enq>:=00000101

<del>:=11111111

<fill>:=00000000

<fill6>:=000000

<dle>:=10010000

<ack>:=00000001

<nak>:=00000010

<rep>:=00000011

<strt>:=00000110

<stack>:=00000111
```

APPENDIX C

DDCMP Block Check Computation

C1.0 DDCMP Error Detection

Error detection is provided in this protocol by means of block check bytes after each of the message headers and message blocks. This block check consists of computing a 16-bit cyclic redundancy check using a polynomial known as the CRC-16 polynomial and appending the check bits computed to each block. This polynomial and scheme have been widely used in BiSync and other protocols.

C2.0 The CRC-16 Polynomial

The CRC-16 polynomial [X^16 + X^15 + X^2 + 1 = (X + 1) * (X^15 + X + 1)] [see section C3.0 step 3 below] has the following error detection properties:

1. It will detect all errors that change an odd number of bits (i.e., 1, 3, 5, ... bit errors).

2. It will detect all errors that change two bits provided that the block length is less than 32767 bits including the CRC bits. Thus the maximum <count> (length of data field) should be 4093.

3. It will detect all errors that consist of a single burst error of 16 or fewer bits. A burst error is a group of bits in which the first bit and the last bit are in error and the intervening bits may or may not be in error. Thus a 16-bit burst error might have as many as 16 bits in error. The partitioning of bits in error into burst errors is not unique.

4. It will detect all errors that consist of two occurences of two adjacent bits in error provided that the block length is less than 32767 bits including the CRC bits.

5. It will detect all except the fraction 1/2^15 of errors that consist of a single burst error of 17 bits.

6. It will detect all except the fraction 1/2^16 of errors that consist of a single burst error of 18 or more bits.

7. It will not detect some errors that change four bits. For example, it will not detect the error pattern that is identical to the CRC polynomial. Thus the minimum Hamming distance between two valid messages (including the CRC bits) is four bits.

C3.0 CRC Computation

The algorithm for computing the cyclic redundancy check is as follows:

1. Consider the header or data portion of the message as it appears on a serial line (LSB of the first byte first, MSB of the final byte last) and append 16 zeros after the header or data.

2. Take the string of bits constructed in 1, and treat each bit as the coefficient of a term of a polynomial with the LSB of the first byte being the coefficient of the highest order polynomial term. The highest order term is A * X^63 for a header block and A * X^(8 * <count> + 15) for a data block where A is the least significant bit of the first byte of the header or data. The lowest order term is 0 * X^0 for both cases.

3. Divide the polynomial constructed in 2, by the CRC-16 polynomial X^16 + X^15 + X^2 + 1 using synthetic division and using modulo 2 arithmetic on the coefficients (i.e., addition = subtraction = XOR. All carries and borrows are ignored) obtaining a quotient that is discarded and a 16-bit remainder.

   Transmit the coefficients of the remainder as the block check bytes following the original message bits, transmitting the coefficients of the highest order term (X^15) first. Thus, the first byte represents coefficients of the X^8 through X^15 terms of the remainder (from left to right) and the second byte represents coefficients of the X^0 through X^7 terms of the remainder (also from left to right).

4. On reception, perform the same algorithm and compare the received block check bytes with the computed block check bytes. If the bytes are not identical, an error has occurred.

Notes:

1. On a parallel circuit, the same algorithm is used and the same block check bytes are computed although the bytes are sent in parallel instead of serially. Notice that for the purposes of the block check byte computation, the LSB of the first byte is always treated as the highest order term (i.e., the term with the largest exponent) in the message polynomial.

2. On reception, the message may be handled with the block check bytes included (two bytes longer) and the algorithm computed based on this longer message. If the remainder is not zero, an error has occurred.

APPENDIX D

MESSAGE FORMAT SUMMARY

D1.Ø General Message Formats

Numbered Messages	Unnumbered Messages	Maintenance Messages
<soh>	<enq>	<dle>
<count>	<type>	<count>
	<subtype>	
<qsync>	<qsync>	<qsync1>
<select>	<select>	<select1>
<resp>	<rcvr>	<fill>
<num>	<sndr>	<fill>
<addr>	<addr>	<addr>
<bcc1>	<bcc1>	<bcc1>
<bcc2>	<bcc2>	<bcc2>
<data>		<data>
<bcc3>		<bcc3>
<bcc4>		<bcc4>

D2.Ø Unnumbered Message Summary

message	ACK	NAK	REP	STRT	STACK
enq	enq	enq	enq	enq	enq
type	ack	nak	rep	strt	stack
subtype	fill6	rnak	fill6	fill6	fill6
qsync	qsync	qsync	qsync	qsync1	qsync1
select	select	select	select	select1	select1
rcvr	resp	resp	fill	fill	fill
sndr	fill	fill	num	fill	fill
addr	addr	addr	addr	addr	addr
bcc1	bcc1	bcc1	bcc1	bcc1	bcc1
bcc2	bcc2	bcc2	bcc2	bcc2	bcc2

APPENDIX E

EXAMPLES

Message exchange examples - These examples are presented here to show the operation of the message exchange component of DDCMP in specific cases of states and occuring events. They do not show actual link timings, as these vary with link characteristics and station selection. The variables used are those presented in section 5.3 and in the message formats in section 4.0. The variables of importance are listed just above the message being sent or just below the message received. Not all variables, user requests, and timers are shown in the examples. They are only shown when needed to illustrate a specific operational detail of DDCMP. The diagram symbols have the following meaning:

symbol	meaning
------>	Send message, received with no errors
---/-->	Send message, received with errors
--//-->	Send message, not received
!	Reply timer running

Example 1 - Startup sequence with no errors.

```
User requests startup
R=0,N=0,A=0,T=1,X=0

                           ------>
                           STRT           Notify user of startup at
                                          other station

                                          User requests startup
                                          R=0,N=0,A=0,T=1,X=0

                           <------
                           STRT

                           ------>
                           STACK          Enter running state

                           <------
Enter running state        ACK(RESP=0)
```

Example 2 - Startup sequence with errors.

```
User requests startup
                           --//-->
                           STRT
                           !
                           !
Timeout                    !
                           !
                           !
                           ------>
                           STRT

                                          Notify user, user requests
                                          startup

                           <------
                           STRT
                                 !
                                 !
                           --//-->
                           STACK !
                                 !
                                 !        Timeout
                                 !
                           <------
                           STRT

                           ------>
                           STACK          Enter running mode

                           <--//--
                           ACK(RESP=Ø)
                           !
                           !
                           !
Timeout                    !
                           !
                           !
                           ------>
                           STACK

                           <------
                           ACK(RESP=Ø)

Enter running state
```

Example 3 - Data transfer with no errors.

```
User requests transmit
R=Ø,N=1,A=Ø,T=2,X=1
                           ------>
                           DATA(NUM=1,RESP=Ø)

                                         Message received and
                                         given to user
                                         R=1,N=Ø,A=Ø,T=1,X=Ø

                           <------
                           ACK(RESP=1)

User requests transmit
R=Ø,N=2,A=1,T=3,X=2
                           ------>
                           DATA(NUM=2,RESP=Ø)

                                         Message received and user
                                         requests transmit
                                         R=2,N=1,A=Ø,T=2,X=1

                           <------
                           DATA(NUM=1,RESP=2)

Message received and user
requests transmit
R=1,N=3,A=2,T=4,X=3
                           ------>
                           DATA(NUM=3,RESP=1)

                                         Message received
                                         R=3,N=1,A=1,T=2,X=1

User requests transmit
R=1,N=4,A=2,T=5,X=4
                           ------>
                           DATA(NUM=4,RESP=1)

                                         Message received
                                         R=4,N=1,A=1,T=2,X=1

                                         User requests transmit
                                         R=4,N=2,A=1,T=3,X=2

                           <------
                           DATA(NUM=2,RESP=4)

Message received
R=2,N=4,A=4,T=5,X=4
                           ------>
                           ACK(RESP=2)
                                         ACK received
                                         R=4,N=2,A=2,T=3,X=2
```

Example 4 - Data transfer with CRC errors and NAKing.

```
User requests transmit
R=0,N=1,A=0,T=2,X=1
                           ----/-->
                           DATA(NUM=1,RESP=0)

                                          Received in error
                                          R=0,N=0,A=0,T=1,X=0

                           <------
                           NAK(RESP=0)

Nak received
R=0,N=1,A=0,T=1,X=1

Retransmit
R=0,N=1,A=0,T=2,X=1
                           ------>
                           DATA(NUM=1,RESP=0)

                                          Message received and
                                          user requests transmit
                                          R=1,N=1,A=0,T=2,X=1

                           <------
                           DATA(NUM=1,RESP=1)

                           ---/-->
                           ACK(RESP=1)
                                          Queue NAK for R=1

User requests 3 transmits
R=1,N=2,A=1,T=3,X=2
                           ------>
                           DATA(NUM=2,RESP=1)

                                          Message received
                                          R=2,N=1,A=1,T=2,X=1

R=1,N=3,A=1,T=4,X=3
                           ------>
                           DATA(NUM=3,RESP=1)

                                          Message received
                                          R=3,N=1,A=1,T=2,X=1

R=1,N=4,A=1,T=5,X=4
                           ------>
                           DATA(NUM=4,RESP=1)

                                          Message received
                                          R=4,N=1,A=1,T=2,X=1
                                          Queue ACK for R=4
                           cont'd
                           <------
                           NAK(RESP=1)    Queued NAK returned

NAK received
R=1,N=4,A=1,T=2,X=4

Retransmit
R=1,N=4,A=1,T=3,X=2
                           ------>
                           DATA(NUM=2,RESP=1)

                                          Message ignored
                           <------
                           ACK(RESP=4)    Queued ACK returned

All messages complete
R=1,N=4,A=4,T=5,X=2
```

Example 5 - Data transfer with errors causing reply timeouts

```
User requests transmit
R=0,N=1,A=0,T=2,X=1
                           ------>
                           DATA(NUM=1,RESP=0)
                           !
                           !              Message received
                           !              R=1,N=0,A=0,T=1,X=0
                           !
                           <--//--
                           !  ACK(RESP=1)
Timeout                    !
                           !------>
                           REP(NUM=1)

                           <------
                           ACK(RESP=1)    ACK response to REP
                                          R=1,N=0,A=0,T=1,X=0

ACK received
R=0,N=1,A=1,T=2,X=1

User requests transmit
R=0,N=2,A=1,T=3,X=2
                           !--//-->
                           ! DATA(NUM=2,RESP=0)
                           !
                           !
                           !
Timeout                    !
                           !------>
                           REP(NUM=2)
                           <------
                           NAK(RESP=1)    NAK response to REP
                                          R=1,N=0,A=0,T=1,X=0

NAK received
R=0,N=2,A=1,T=2,X=2

Retransmit
R=0,N=2,A=1,T=3,X=2
                           ------>
                           DATA(NUM=2,RESP=0)

                                          Message received
                                          R=2,N=0,A=0,T=1,X=0
                           <------
                           ACK(RESP=2)

ACK received
R=0,N=2,A=2,T=3,X=2
```

cont'd

```
User requests transmit
R=0,N=3,A=2,T=4,X=3

                           !------>
                           ! DATA(NUM=3,RESP=0)
                           !
                           !              Message received
                           !              R=3,N=0,A=0,T=1,X=0
User requests transmit     !
R=0,N=4,A=2,T=5,X=4        !
                           !--//-->
                           ! DATA(NUM=4,RESP=0)
User requests transmit     !
R=0,N=5,A=2,T=6,X=5        !
                           !------>
                           ! DATA(NUM=5,RESP=0)
                           !
                           !              Message ignored
                           !              R=3,N=0,A=0,T=1,X=0
                           !
                           <--//--        ACK to received message
                           ! ACK(RESP=3)
Timeout                    !
                           !------>
                             REP(NUM=5)

                           <------
                             NAK(RESP=3)  NAK response to REP
                                          R=3,N=0,A=0,T=1,X=0

NAK received
R=0,N=5,A=3,T=4,X=5

Retransmit
R=0,N=5,A=3,T=5,X=4
                           ------>
                           DATA(NUM=4,RESP=0)

                                          Message received
                                          R=4,N=0,A=0,T=1,X=0

Retransmit
R=0,N=5,A=3,T=6,X=5
                           ------>
                           DATA(NUM=5,RESP=0)

                                          Message received
                                          R=5,N=0,A=0,T=1,X=0

                           <------
                           ACK(RESP=5)    ACK to received messages
All messages complete
R=0,N=5,A=5,T=6,X=5
```

APPENDIX F

REVISION HISTORY

This list of DDCMP changes from version 3.Ø to the current version 4.Ø is provided as a guide for those who have implemented and/or are familiar with version 3.Ø and are interested in the changes from that version to the present 4.Ø level. Revision levels between 3.Ø and 4.Ø are shown for those who have such documentation. The composite list of changes are all the technical changes from version 3.Ø to version 4.Ø. Only technical changes and major clarifications are listed. Changes in wording or documentation level are not included in this revision history.

Version 3.Ø

There were 3 printed version of this level spec, May 1974, August 1974, and December 1974. The December 1974 document was the most readable, correct, and widely circulated of the three. This document is taken as the base 3.Ø level and all changes are from this copy of version 3.Ø.

Changes 3.Ø to 3.Ø2:

1. Changed FINAL flag bit to QSYNC flag bit. The SELECT flag bit is now used for both selecting and ending selection. The QSYNC flag signals non-abutting messages allowing short sync sequences between messages on synchronous links.

2. Removed the RESET and RESAK messages. The link is reset in both directions at the same time by use of the STRT/STACK startup sequence.

3. Clarified the synchronous and asynchronous synchronization and pad sequences. The sequences were made dependant on the link transmission characteristics.

4. Removed RESP and NUM fields from the STRT and STACK messages, replacing then with FILLs. After a startup sequence message numbering always starts with message number 1.

5. Changed ADDR field to always be tributary address in messages both to and from tributaries. This change added a positive identification to the messages being returned from a tributary to the control station.

6. Framing requires first byte to be SOH, ENQ, or DLE. If not one of these then no message is framed and, thus, no NAKs are generated as in the previous version.

7. Startup sequence revised to send an acknowledge to a received STACK. The new sequence adds an ACK response to a STACK, creating a positive three-way handshake.

8. Multipoint operation and sync sequences clarified. Many of the operational details for multipoint were added to the specification.

Version 3.Ø2

This version was the result of the above listed changes made by the DDCMP committee. This and intermediate review edition 3.Ø1 were dated July 1975 - August 1975. The document was also partially rewritten to add additional clarification to the specification.

Changes 3.Ø2 to 3.Ø3:

1. Require full duplex tributary to wait until the entire message with the SELECT flag bit on is received before transmitting. Previously it had to wait only for the header. This change provides for common operation among the different modes and link types.

2. Require that STRT, STACK, and MAINT messages have the SELECT and QSYNC flag bits always set. These messages always operate one message at a time alternately. This change makes their use common in all modes and link types.

3. Multipoint control stations must precede a message with a sync sequence if addressed to a tributary different from the one addressed in the previous message. This change improves receiver synchronization at the tributaries.

4. DATA and MAINT messages with zero length data fields are not allowed.

5. The BOOT message is changed in name to the MAINTENANCE message.

Version 3.Ø3

This version resulted from the changes listed above by the DDCMP committee. Version 3.Ø3 dated December 1975 was never totally reviewed by the DDCMP committee and contains a number of errors.

Changes 3.Ø3 to 4.Ø:

1. Divided protocol functions into 3 semi-independent components: framing, link management, and message exchange. This is not a technical change but helps clarify and more precisely specify the protocol.

2. Clarified the user and device interfaces to the protocol. This is not a technical change but assists in the understanding of the protocol.

3. Allowed the SELECT flag bit to be set in full duplex point-to-point mode with no checking by the receiver. This change makes for more compatible full and half duplex implementations.

4. Modified the link idle signal and optional end-of-message checking to increase CRC-16 error detection capability in cases of synchronous clock loss.

5. Allowed an optional increase in the sync sequence length to recover from framing errors when responding to a NAK. Especially useful on asynchronous links.

6. Made checking of ADDR field optional in point-to-point mode and checking of FILL fields optional in all modes.

7. Described the operation of the reply timer in cases of retransmission. This is not a technical change but a precise description of the timer operation during retransmission.

8. Changed error recording counters to be a recommendation not a requirement. Error recording is now a recommended optional part of a DDCMP implementation.

9. Clarified messages to be retransmitted when NAKs and ACKs are received asynchronous to transmission. This is not a technical change but a more precise definition of the operation in these cases.

10. Removed the option of multipoint tributaries operating in sheltered mode. The only valid mode is circumspect mode, where the tributaries always track messages on the link. This mode is now simply called multipoint tributary operation.

11. Changed the operation of the select timer to be stopped and restarted after every good received message or when resynchronizing. This changes the timer to one that times a lost select flag (message with flag in error), rather than one that times the duration of a total selection interval.

Version 4.0

This version is a total rewrite of the DDCMP protocol specification.

Three versions of 4.Ø were used for review in limited circulation, dated June 1977, October 1977, and December 1977. It is an attempt to more clearly and precisely define the DDCMP protocol. It incorporates the changes listed above from the previous version.

[END OF DDCMP SPECIFICATION]

digital

digital equipment corporation

Printed in U.S.A.