Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

MESSAGE FORMAT AND PROTOCOL FOR INTER-COMPUTER COMMUNICATION

by

A. K. Bhushan and R. H. Stotz

June 16, 1967
Project MAC Memorandum MAC-M-351

Need a unique identifier on every message.

Work reported herein was supported (in part) by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number NOnr-4102(01). Reproduction in whole or in part is permitted for any purpose of the United States Government.

MESSAGE FORMAT AND PROTOCOL FOR INTER-COMPUTER COMMUNICATION

A. K. Bhushan and R. H. Stotz

With the development of computer networks, the computer utility approach, and sophisticated displays maintained by satellite computers connected to multi-access time-shared computers, the need for machine-to-machine communications has become evident. Communications between computing machines require the establishment of a message format and protocol, by which we mean a uniform agreed-upon-manner of exchanging information. This paper is an attempt towards defining such standards.

A single message format for inter-computer communications may not be feasible because of the large discrepencies in the hardware and software supplied by the various manufacturers of computing machines, the wide difference in the size and capabilities of different machines, and the need to communicate at different levels. The fact that user's needs may differ must also be taken into consideration. This paper does not specify a single standard message format and protocol. What it does is to provide the necessary guide lines toward the construction of compatible message formats and outlines a general format. Some specific message formats that we propose for Project MAC and Multics standards will appear as appendices. The purpose of these standards is to provide a precedent for the manner to communicate to the system. Establishment of a standard does not preclude users within Multics with special requirements from implementing a variant method for communication.

At the simplest level of communication (e.g., to a teletype terminal), the messages are all text with an arbitrary start and end, and there is no means for error control or acknowledgment to determine if the messages were received correctly. Such messages require no formatting at all. A higher level of communication would include specifying the beginning and the end of the message, providing error checks and requiring an acknowledgment, so that if the message is in error it can be retransmitted. A still higher level would require the inclusion of auxiliary information such as routing, security, and

numbering of the message in a header that precedes the message. At a final level would be the ability to transmit multiple messages and acknowledge them individually.

The general message format described herein is compatible with the American Standard Code for Information Interchange (ASCII)[1,2,3] and follows the standard recommended by ASA, given in "Control Procedures for Data Communications - An ASA Progress Report," published in the February 1966 issue of the Communications of the ACM. The format is recommended for synchronous as well as asynchronous, and for half-duplex (simplex) as well as full-duplex transmission* at speeds typically ranging from a few hundred to hundreds of thousands of bits per second, for communication between various computers and time-sharing systems with suitable communication adapters and interfaces.

For clarity we shall first define some terms:

A transmission - a group of one or more messages which are transmitted continuously without interruption. A transmission starts with SYN characters and ends with the EOT character.

A message - a sequence of characters arranged for the purpose of conveying information from an originator to one or more destinations. It includes all text and the appropriate header. A message starts with the SOH character (STX if no header) and ends with the ETX character (EOT if no text).

A header - a sequence of characters which constitute the auxiliary information necessary to the communications of a text. Such auxiliary information may include, for example, characters representing

---

* Synchronous transmission means characters (and bits) are transmitted at a fixed rate. Asynchronous transmission means the interval of time between characters can vary arbitrarily.

Full duplex is the ability to transmit simultaneously in both directions while half duplex is the ability to transmit in both directions, but not simultaneously.

routing, priority, security, message numbering and associated separator characters. A header starts with the SOH character and ends with the STX character (EOT if no text).

A text - a block of data that is to be communicated to the user. A text starts with the STX character and ends with the ETX character (ETB if another text follows).

BASIC MESSAGE FORMAT

A typical message is illustrated in Fig. 1.

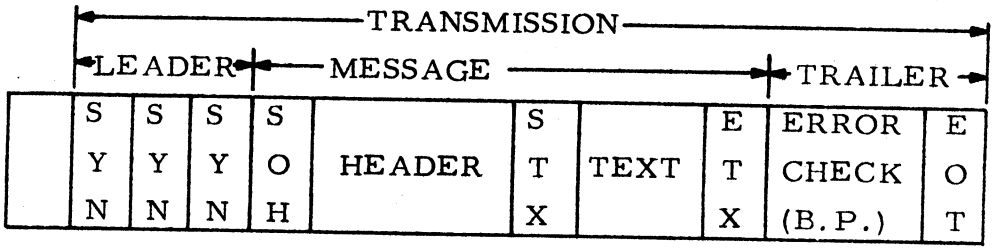


Fig. 1 Basic Message Format

Each character in the above message is a 7-bit ASCII character followed by one bit of odd parity. The least significant bit is transmitted first and the parity bit is transmitted last.[4] The Characters SYN, SOH, STX, ETX, ETB and EOT are all standard ASCII communication control characters and have the following code value and meaning:

ASCII Code Value Octal	Character	ASCII Meaning
001	SOH	Start of Heading
002	STX	Start of Text
003	ETX	End of Text
004	EOT	End of Transmission
026	SYN	Synchronous idle
027	ETB	End of Transmission Block

The leader and trailer are always automatically supplied by a communication adapter module (system-provided software and/or hardware) in the transmitting computer. Similarly, they are automatically stripped from the message on reception. The function of the leader, which consists of SYN characters, is to stabilize the line after turn-on and provide the data set with correct synchronization. The trailer contains a longitudinal error check (B.P. in Fig. 1), and signals the end of transmission with an EOT. For the error check, longitudinal block parity is recommended by ASCII. This check is easy to implement either in hardware or software. Other forms of checking, such as the cyclic redundancy check for some IBM machines, are permissible within this framework. Whatever the form of check, it includes the ETX character which ends the checked sequence but does not include the SOH or the STX character that starts the sequence. Character parity generation and checking will usually be done in the hardware, but block parity may be generated and checked either in software or hardware.

The character STX when it follows an SOH indicates the end of header information and beginning of user's text. Messages that have no header information may start with an STX as illustrated in Fig. 2. Null text messages are allowed as illustrated in Fig. 3. The character ETX indicates the end of the text and message. All messages containing text would require acknowledgments. Non-text messages begin with an SOH and end with an EOT. These messages, illustrated in Fig. 4, need not be acknowledged.

As shown in Fig. 5, some messages may be blocked for error control (or other transmission purpose) in a manner not necessarily related to the processing format. In this case each block starts with an STX and ends with an ETB. Special characters (s.c.) for error

control may follow an ETB. When used for generating error check characters, an ETB is treated like an ETX.*

A need is envisaged for sending multiple messages in a single transmission, primarily in order to avoid interrupting the receiving computer on each of many short messages which can be sent as a group. As shown in Fig. 6, this is accomplished by omitting the EOT from the trailer of all but the last message. Multiple messages are acknowledged individually, and therefore each must contain an SOH and message identification in its header.

An EOT always indicates the conclusion of a transmission, and signals the receiver that the transmitted message(s) should be acknowledged and processed. For half-duplex service the originating station then relinquishes its right to transmit and goes into receive mode. Messages are held in the transmission buffer pending receipt of an acknowledgment that it has been properly received. On receipt of a positive acknowledgment (message OK), the message is dropped from the transmit buffer. If a negative acknowledgment is received, the message is retransmitted. This process continues until successful transmission is achieved, or the message is abandoned.

On very long messages, there may be a problem with buffer sizes, either in the transmitter or receiver. By suitable protocol procedures, the transmitting computer will either control or be informed of the transmit and receive buffer sizes, and will send an EOT when the smaller of the two buffer sizes is about to be exceeded. Transmission of the message can be resumed when proper acknowledgments are received and buffer space is again available.

---

* The need for dividing the message into multiple blocks of text may be obviated, and the same error control achieved, by dividing the logical block of text into separate messages, and using the key character '&' in the header to indicate that the logical block continues into the next physical block. It would be easier to acknowledge and retransmit separate messages rather than separate blocks. A slight overhead results because separate headers are now needed, but this may not be undesirable.

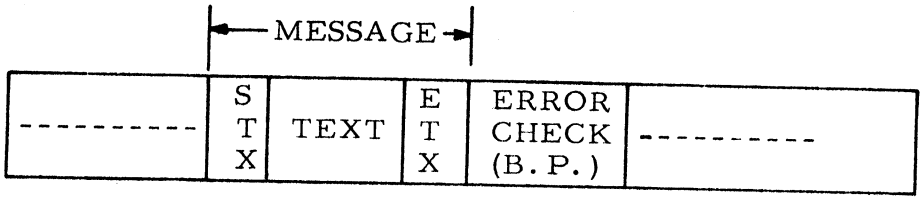


Fig. 2 Message with No Header

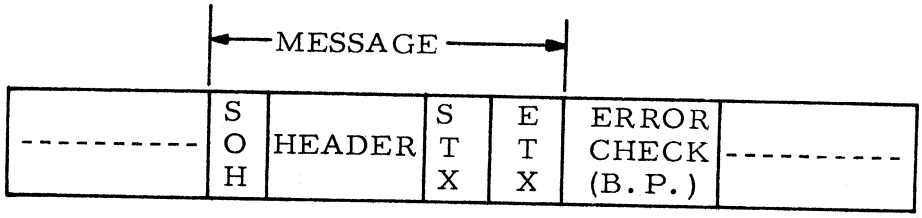


Fig. 3 Null Text Message

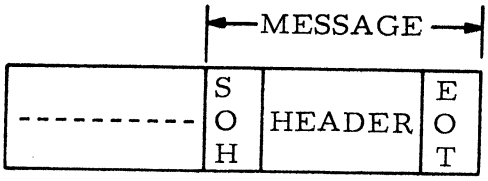


Fig. 4 Non-text Message -- Transmission Ends

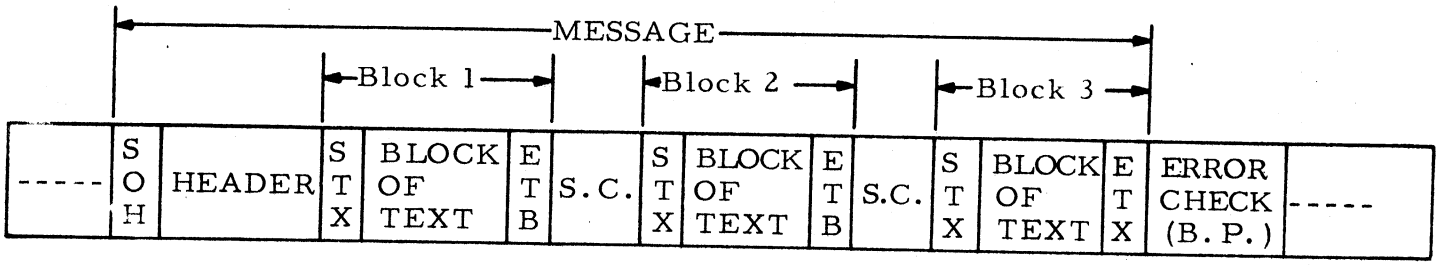


Fig. 5 Message with Multiple Blocks of Text

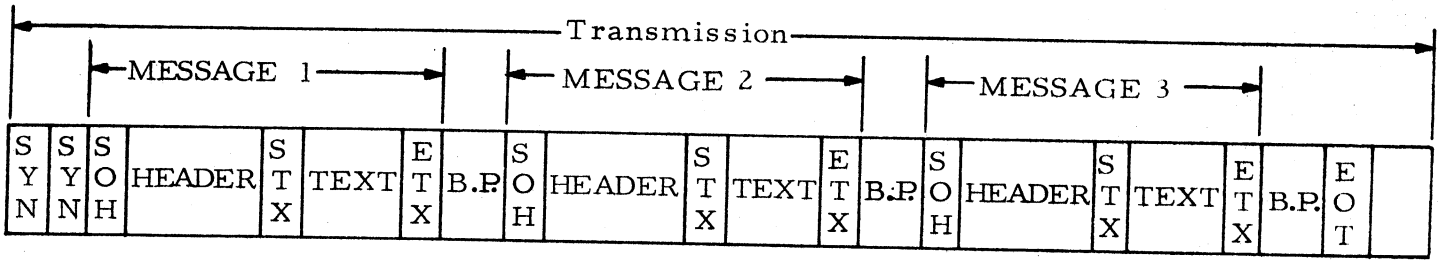


Fig. 6 Multiple Messages in a Single Transmission

HEADER INFORMATION

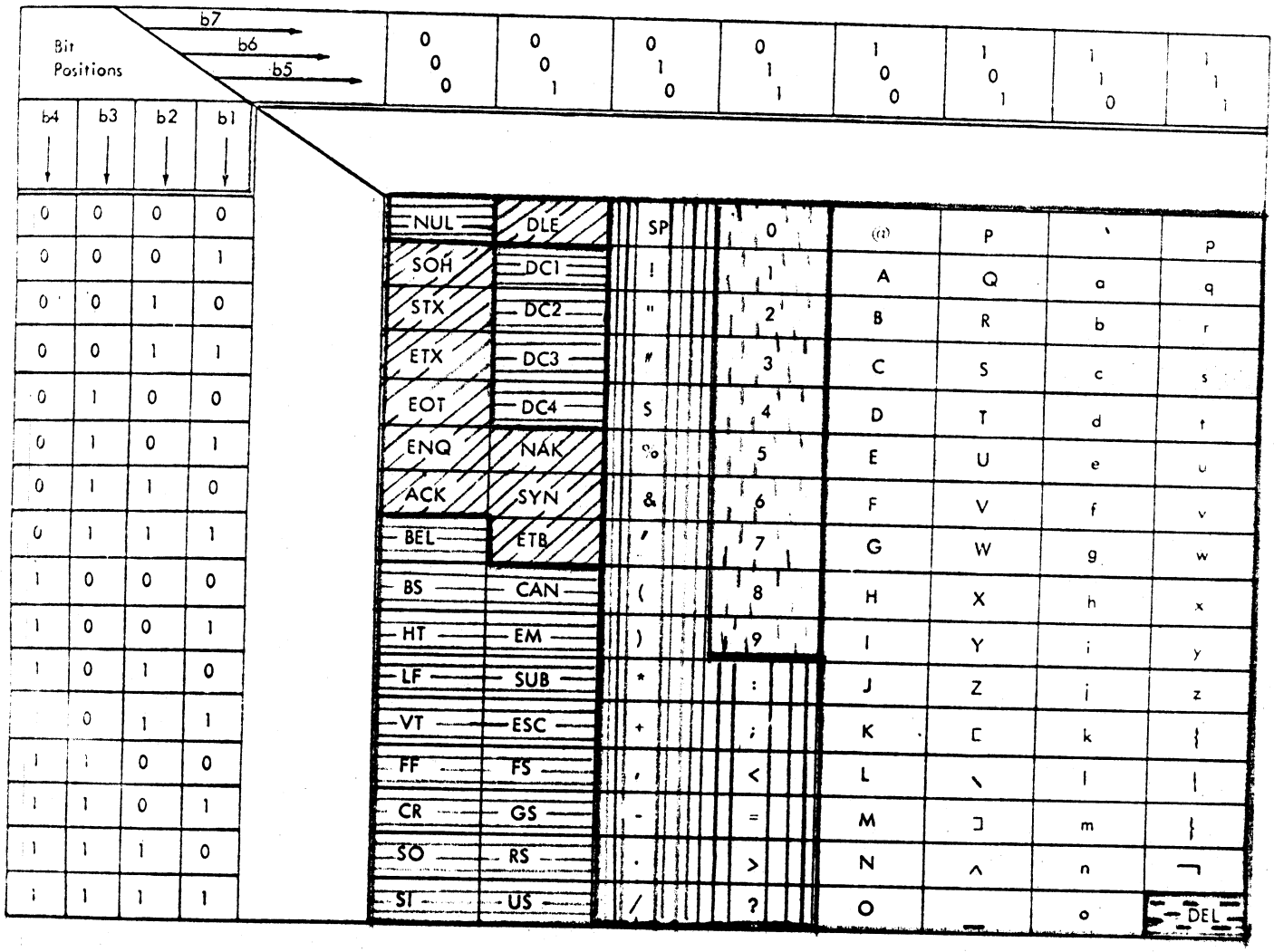
Information in the header of the message is machine-sensible address or routing information used only to aid the communications program to efficiently transmit, receive, sequence and route the messages. The header information is never seen by the user, and all information regarding the text portion of the message (e.g., should the ASCII characters in the text be interpreted as ASCII or binary) is contained in the text portion of the message.

The ASCII characters in the header are separated into two categories for convenience in transmission of information, and also to achieve a measure of compatibility between different users. The first category of characters is those that have a 'zero' as their most significant bit. These characters (octal 000 to 077) are the ASCII control characters, punctuation characters and some graphic symbols. The non-control ASCII characters in this category shall be predefined and the predefined characters will be hereafter referred to as Key characters. The second category of characters is the remaining ASCII characters which have a 'one' as their most significant bit (octal 100 to 177), and may be interpreted as ASCII or binary (the lower order six bits representing the binary information). The ASCII character set and its division is shown in Fig. 7.

The key characters have a predefined meaning and may be followed by a string of characters from the second category called the argument of the key. The argument of a particular key is interpreted by the computer in accordance with the predefined meaning assigned to the particular key. The key completely defines the interpretation of the information contained in the argument. Additional separator characters are not needed, as key characters themselves act as separators.

The key characters together with some control characters serve as header control characters. In general it will be wise to preserve the meaning of the ASCII control characters when used for the purpose of header control characters. It is advisable not to use communication

United States of America Standard
Code for Information Interchange
(USASCII)



Control characters

Communication control characters

Key characters

Numerals interpreted as special arguments

Second category representing argument (includes the control character DEL)

Control character DEL included in the argument set

Fig 7. USASCII Character Assignments

control characters like ACK, NAK and ENQ in the header because some of the hardware may be sensitive to these. ASCII numerals should not be used as key characters, but may be used as special arguments. There is, however, no dearth of key characters. Additional key characters may be generated by using the ESC (Escape) character. An ESC followed by an ASCII graphic (octal 040 to 176) assumes the meaning of a new key character (ESC grph). Also it should be remembered that key characters are not control characters and should not be used for such a purpose. If a particular key character is not recognized by the computer software (or device hardware), it is to be ignored (together with the argument which may follow it) till the next recognizable key or control character appears.

For convenience the header control characters have been divided into three classes, representing communication requirements between different types of machines. For example, some small machines may recognize only a limited class of header control characters. Other small machines (e.g., satellite computers for display application) may require unbuffered transmission. Table 1 illustrates a tentative assignment of key characters and their meanings. This list is subject to revision as requirements are better understood.

The basic Class I header control characters include only the acknowledgment, identification, enquiry (not ASCII control characters) and a means to interrupt and quit. Messages containing identification will be acknowledged with that identification. Also a limitation on the maximum size of the message can be indicated. Such a bound may be necessary, as some machines may have limited buffers. (The number of characters in the message is equal to the sum of all characters between the leader and the trailer plus the receivable characters in the leader and the trailer, if any.)

Class II header control characters provide for unbuffered transmission between machines (i.e., machine A can send data to machine B directly into its proper location in B). This is particularly

Table I

Assignment of Key Characters

Class I

#	arg	argument is message identification
+	arg	acknowledge message #arg (message O.K.)
-	arg	negative acknowledge message #arg (message in error)
?	arg	repeat acknowledgment of message #arg (equivalent to ENK)
.	n	send transmissions no longer than size n.*
"		interrupt (similar to TTY break)
!		quit (similar to TTY quit) stops transmission of messages

Class II

:	arg	sender's status
;	arg	sender's interpretation of receiver's status
/	arg	number of characters in text of message
&	arg	argument is core memory address of incoming text
,	arg	binary checksum on header

Class III

=	n	make standard buffer size n
<	n	"= n" request rejected
>	n	"= n" request accepted
*	n	size n blocks of text only

* The character n in the above list stands for any numeral. A message of size n contains no more than 2**(6+n) characters, with n ranging between 0 and 9. This means that the range of allowable message sizes falls between 64 and 32,768 characters. Since messages can be of variable length, n serves primarily as an upper bound on the maximum size of the message.

important when a large time-sharing computer communicates with a small satellite computer. In general, satellite computers will have a limited amount of memory, and it is desirable not to have to provide a large message buffer in this machine. On the other hand, if a small message buffer is provided, it will take multiple calls on the time-shared computer to get a large message across, which is also undesirable. Direct transmission is desirable in this case. Note that the "where to" and "number of characters in this message" information must come before the text (i.e., in the header), and that it is probably desirable to send an error check on the header itself to ensure correct receipt before proceeding to store the text in the wrong location.

Class III header control characters are intended primarily to obviate the need for computer processor intervention after each message. It is inefficient to interrupt large multi-access computers every time a short message is received. This can be avoided by transmitting multiple messages in a single transmission and storing them in a receive buffer. Class III header control characters can be used to create the necessary receive buffers and indicate maximum size of a transmission.

The unassigned key characters (or ESC graphic characters) may be used to define supplementary header information as and when needed, but ASCII control characters should be assigned meaning only in accordance with ASA recommendations.

TEXT INFORMATION

The text of a message always consists of a string of ASCII characters, even if binary information is being sent. The whole text may be considered to consist of Files in which a Group of binary data can be transmitted by inserting into the text stream the Group Separator (GS) character (octal 035). The six least-significant bits of each ASCII character which follows will be interpreted as transmitted binary information and the most-significant bit will always be 'one'. In a

message, transmission is assumed to be a File of "pure" ASCII text until the first GS appears. The ASCII stream is then interpreted as a Group of binary data until the File Separator (FS) character (octal 034) appears which indicates that what follows is a new file of "pure" ASCII text and signals the leaving of binary mode. The positioning of GA and FS characters in text stream is completely independent of the message itself. Thus a single message may contain both binary and ASCII information, or any combination of the two.

Characters with 'zero' in their most significant bit may occur within binary mode of transmission and be interpreted usefully. These may be control characters or key characters. The ASCII control characters are to be used in accordance with ASA recommendations. The key characters occurring within binary mode may be used to further define the binary information if it is so desired. The meaning assigned to the key characters in text may be different from that assigned in the header. Thus key characters may be used to define the binary information that follows it and indicate what it is about.

TRANSPARENT TEXT

Many people have expressed a desire for a "Transparent Text" mode of operation, in which the communication equipment between the transmitting device and the ultimate addressee is insensitive to the contents of the text. The unrestricted coding of data permitted by a transparent text mode would allow transmission of full seven-bit binary data. Thus, messages containing EOT, ETX, ETB, ACK, NAK and similar control characters could be transmitted intact without affecting the transmission system. If the parity bit is also ignored in the transparent mode, then it would be possible to send full eight-bit binary data. The ability to transmit binary data is extremely important in computer communications.

The difficulty with transparent transmission is that there is no standard way to indicate the end of this mode. If all the 128 (256 if the parity bit is also used for information) codes are allowed for data, there can be no reserved "end transparent mode" code. For

transmitting seven-bit transparent text, ASA has suggested a technique which makes limited use of the eighth (parity) bit to achieve this control.[5]

IBM, whose machines are based on eight-bit characters, has a compelling motivation for providing an eight-bit binary "transparent mode." To accomplish this they have set aside the DLE code (with the normal odd parity) as a special character out of the possible 256.[6,7]

In the IBM technique, the sequence DLE STX initiates the transparent text mode and DLE ETX terminates it. If a bit pattern equivalent to DLE appears within the transparent data it is replaced by the sequence DLE DLE to permit transmission of DLE as data. In addition, other control sequences using DLE are available to provide active control characters within transparent text as required (see Fig. 8).

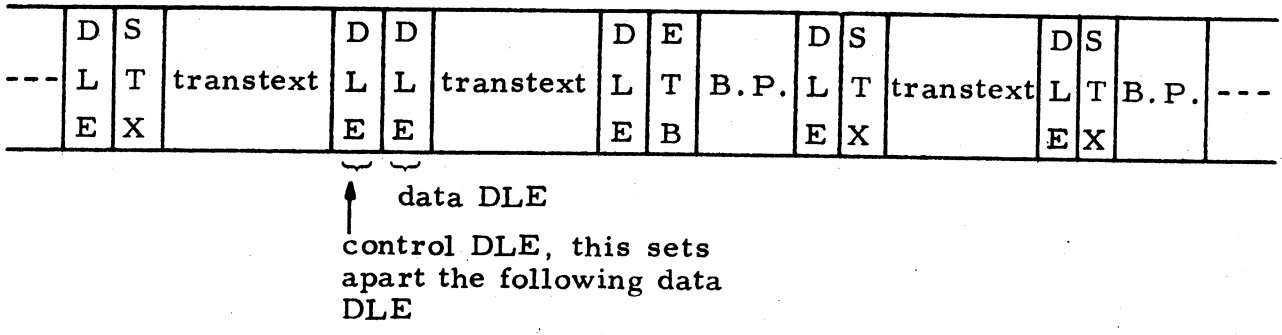


Fig. 8 IBM Method for Transmission of Transparent data

We see several difficulties with the IBM technique. First, all equipment handling the message must be built to conform to the technique, and it will be ineffective if any of the equipment in the transmission link (or computer network) is code sensitive. Second, if the character that indicates the end of "transparent mode" is in error

there will be no simple means for escaping from the "transparent mode." This is because standard ASCII control characters are ignored, once in "transparent mode." Third, the IBM method of transmitting eight-bit binary is sensitive to errors since it has no character parity checking (this also violates ASCII which requires the eighth bit to be character parity check). Finally, there is the price that has to be paid for inserting extra DLEs and extracting them (each character in the transmitted and received messages has to be examined as to whether it is or it is not a DLE).

Also, transmitted text would not contain DLE pattern.

Our approach for transmitting transparent or binary text is to provide a six-bit binary mode within ordinary text by using the Group and File separator characters, as suggested earlier. This has the advantage that it is uniform, ASCII conforming, easy to implement, and keeps aside the standard control character set for error control and recovery protocol. Further, it is independent of code-sensitive equipment, and does not require insertion and deletion of DLEs in the transmitted and received texts. The price paid is the reduced effective bandwidth, since only six bits out of each eight-bit character carry information. However transmission of six-bit binary data is desirable for those machines which have word lengths based on six-bit characters.

If a transparent mode for seven-bit characters is required, a modification of the A.S.A. and IBM approaches may be taken without violating ASCII. In this, DLE STX defines the start of "transparent text," and ETX defines the end. All ASCII control characters (bit 7, 6 = 00) appearing within the transparent text are detected by transmitting hardware (or software) and are replaced by a two-character sequence consisting of DLE followed by the character detected, with its bit 7 converted from a 0 to 1. Thus the only control character ever seen by the communication equipment is DLE. The receiver will recover the original text by deleting the DLE and reconverting bit 7 of the next character from 1 to the original 0. ETX marks the end

of the message as with any other message. This approach is independent of code-sensitive equipment and keeps aside the standard control character set for error control and recovery protocol.

MESSAGE PROTOCOL

Message protocol here refers to the uniform, agreed-upon manner of exchanging messages between computing machines. This includes link establishment, link relinquishing and acknowledgment procedure for messages.

The following control characters will be used for message protocol. The first group contains control characters with standard ASCII meanings; the second contains our assignment of additional controls for protocol purposes.

ASCII Code Value octal	ASCII Character	Meaning Assigned
005	ENQ (Enquiry)	Request response from remote station
006	ACK (Acknowledge)	Last message ok. Ready to receive next message
026	NAK (Negative Acknowledge)	Last message in error. Repeat transmission
020	DLE (Data Link Escape)	Modifies the following character for communication control purpose
030	CAN (Cancel)	Disregard the message

ASCII Character	Mnemonic	
DLE ACK	WBT (Wait Before Transmit)	Last message ok. Wait for ACK before transmitting
DLE NAK	BSY (Busy)	Not ready
DLE CAN	INT (Interrupt)	Cause a higher level interrupt (similar to teletype break)
DLE !	QIT (Quit)	Cause the program to quit (similar to a teletype quit)
DLE EOT	HNG (Hangup)	Disconnects the line

To start transmission, a number of SYN characters (the exact number being under program control) are sent to establish the correct character and bit synchronization. The originating station will then send an enquiry to request a response from the remote station. The enquiry may be just a single ENQ, or an ENQ followed by an EOT, or a message included within an SOH and an EOT including an enquiry and indicating station identification, station status and a check sum as desired. The response by the receiving station may be ACK, NAK DLE ACK, or DLE NAK depending on the state of the receiving station. These may or may not be followed by an EOT. The response could also be a message within an SOH and an EOT including the acknowledgment and indicating station identification, station status and a check sum as desired.

Again there cannot be any single standard protocol format and procedure because of the varying needs of different systems. The exact protocol chosen in particular cases will depend on the type of operation, the level of communication and the hardware characteristics. Operation could be half duplex, or full duplex, synchronous or asynchronous, point-to-point or multi-point, centralized (with one station as master station always) or decentralized (no one station as master station), and over private line or a switched network. In many of the above cases the protocol needs would differ, and hence the protocol procedures chosen would vary. The level of communication would essentially dictate the particular protocol format rather than the protocol procedures. Depending on the level of communication, the acknowledgments may or may not be formatted within a header. Finally, the protocol format would depend on the communication control character that puts the transmitting station into receive mode (i.e., turns the line around, and initiates action at the receiver). This function may be under program control or it may be built into the hardware. In computing machines of various manufacture, any one or more of the characters EOT, ENQ, ACK, NAK, ETX or ETB may

cause a transmission reversal. If ETX and ETB are allowed to cause a transmission reversal (and acknowledgments to be sent), then it is not possible to transmit multiple messages. Allowing transmission reversals on ENQ, ACK and NAK would limit the use of these characters considerably. Therefore it is our opinion that only an EOT should be used to cause a transmission reversal, and initiate proper action at the receiver. When such is the case, acknowledgments and enquiry must be followed by an EOT.

The only difference between synchronous and asynchronous operation, so far as the message format and protocol are concerned, is in the transmission and treatment of SYN characters. Full duplex allows simultaneous communication of messages in both directions and thus obviates the need of transmission reversals. It also completely avoids the issue of line priority and control, and possible confusion arising thereof (i.e., both communicators trying to transmit at the same time over a single line). In half-duplex transmission, however, suitable protocol procedures must be adopted to assign line priority to the stations and avoid such confusion and "hung" conditions. In point-to-point communcations it is not necessary to establish station identification and priority, but multi-point operation over private lines would require suitable polling and selection procedures in the protocol. We recommend using a conversational mode in which acknowledgments, identification and other information are included in a properly formatted header. When operating in a switched network, the point-to-point connection can be established either manually or automatically; the manually dialed connections could either be operated as point-to-point or multi-point with polling and selection. When automatic calling and answering is employed, both stations must first properly identify themselves, transmit messages and disconnect after they have completed their message transmissions.

In point-to-point operation, receipt of an ACK after an enquiry is sent by the originating station establishes the link. Messages can now be exchanged between the computing machines. To avoid "hung" conditions

half-duplex operation, the transmitter is always responsible for getting the message through and acknowledged. If the expected acknowledgment is not received within a few seconds (exact time may be fixed by each communicator depending on his requirements), an enquiry is sent by the originating station to repeat last acknowledgment or indicate station status. Similarly, if the ready condition (ACK) is not received within a specified time after the "Wait Before Transmit" (WBT) signal, an enquiry is sent to determine if the ready was lost.

When multiple messages are being transmitted, they may be all individually acknowledged in a single transmission (between an SOH and and EOT) by referring to their identification. This acknowledgment may also be made part of the header of a return message, as shown in Fig. 9 (the letters a, b, c.. are the message identifiers).

- - -	SOH	Ackge	a	Ackge	b	Ackge	c	Header contd	STX	TEXT	ETX	B.P.	- - -

Fig. 9 Acknowledgement of Multiple Messages

If the called computer is busy and cannot accept a message, it sends a BSY (DLE NAK) signal to indicate this condition. BSY does not disconnect the link.

The control character HNG (DLE EOT) is reserved for breaking the link. The link must then be re-established if communication is to be resumed. To avoid unwanted hangup by an incorrect transmission of DLE EOT, additional error checking mechanisms may be employed. For example, the HNG can be transmitted as part of a Header message with parity check, or one may require two or more HNGs in succession to mean hangup.

The need for other control characters such as cancel, interrupt and quit is envisaged. The character CAN appearing anywhere in the message would cause the entire message to be disregarded. INT (DLE CAN)

may be used for a higher level interrupt at the receiving computer. QIT (DLE!) may be used to stop the execution and transmission of messages. The link is not relinquished.

REFERENCES

1. "Proposed Revised American Standard Code for Information Interchange", Communications of the ACM, Vol.8, No. 4, April 1965, pp. 207-214

2. "Control Procedures for Data Communications - An ASA Progress Report", Communications of the ACM, Vol. 9, No. 2, February 1966, pp. 100-107

3. "Code Extension in ASCII (An ASA Tutorial)", Communications of the ACM, Vol. 9, No. 10, October 1966, pp. 758-762

4. IBM Systems Reference Library Form A27-3004-0, General Information - Binary Synchronous Communication

5. IBM Technical Newsletter No. N27-3011, Re: Form A22-6468-1 on SDA - II

6. "Character Structure and Character Parity Sense for Serial-by-Bit Data Communication in the American Standard Code for Information Interchange", Communications of the ACM, Vol. 8, No. 9, September 1965, pp. 553-556

7. "Transparent-Mode Procedures for Data Communication; Using the American Standard Code for Information Interchange - A Tutorial", Communications of the ACM, Vol. 8, No. 4, April 1965, pp. 203-206