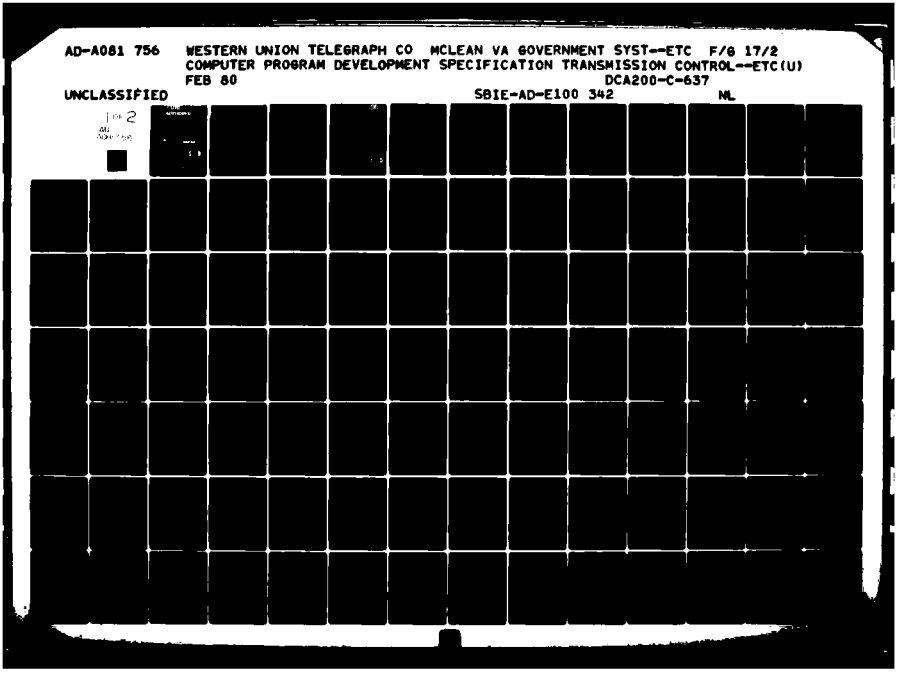
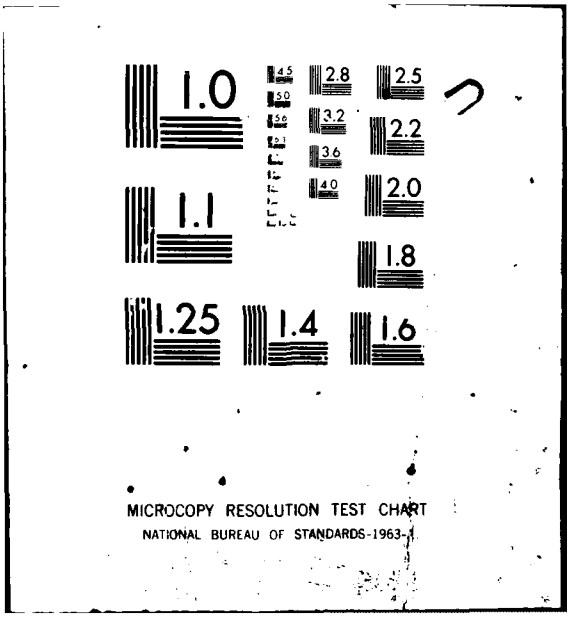
AD-A081 756 WESTERN UNION TELEGRAPH CO MCLEAN VA GOVERNMENT SYST--ETC F/G 17/2
COMPUTER PROGRAM DEVELOPMENT SPECIFICATION TRANSMISSION CONTROL--ETC(U)
FEB 80 DCA200-C-637

UNCLASSIFIED SBIE-AD-E100 342 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

AUTODIN II

See 1473 in back.

western union

Prepared Under Contract
DCA 200-C-637

WESTERN UNION TELEGRAPH COMPANY
GOVERNMENT SYSTEMS DIVISION
7916 Westpark Drive
McLean, Virginia 22102

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AD-E100 342

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: DCA 200-C-637-P005
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Computer Program Development Specification Transmission Control Program (TCP) • Revision 3.
5. TYPE OF REPORT & PERIOD COVERED: FINAL Sept.
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Western Union Telegraph Company
8. CONTRACT OR GRANT NUMBER(s): DCA 200-C-637
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Western Union Telegraph Company, Government Systems Division, 7916 Westpark Drive, McLean, Virginia 22102
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communications Agency, ATTN: Code 450, Washington, D.C. 20305
12. REPORT DATE: 8 Feb 80
13. NUMBER OF PAGES: 155
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Defense Communications Agency, ATTN: Code 450, Washington, D.C. 20305 SBIE
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A
16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distribution unlimited AD-E100 342
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): Approved for public release; distribution unlimited
18. SUPPLEMENTARY NOTES: Review relevance five years from submission date
19. KEY WORDS (Continue on reverse side if necessary and identify by block number): Telecommunications, Data Communications Protocol, Switching Systems, Telecommunications Systems, Data Communications System, AUTODIN II
20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

This document provides the functional specification of the Transmission Control Protocol (TCP) as used in the AUTODIN II packet switched network. TCP performs the mechanics of establishing, maintaining, and terminating a virtual connection through the AUTODIN II system. TCP controls the flow of data so that the users on either end of the virtual connection appear to be communicating via a dedicated circuit. This document also contains many examples of the TCP design and implementation in the AUTODIN II Multiple Channel Control Unit.

AD-A100829

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

AUTODIN II

(12) LEVEL II

FINAL
COMPUTER PROGRAM DEVELOPMENT
SPECIFICATION
TRANSMISSION CONTROL PROGRAM (TCP)

CDRL ITEM NO. B006

DOCUMENT CONTROL # DIN II-F-622-13B06-1124

PREPARED FOR:

DEFENSE COMMUNICATIONS AGENCY
DCA CONTRACT 200-C-637

MARCH 5, 1979

REVISION 1 - JULY 4, 1979
REVISION 2 - AUGUST 3, 1979
REVISION 3 - FEBRUARY 8, 1980

WESTERN UNION TELEGRAPH COMPANY
GOVERNMENT SYSTEMS DIVISION
7916 WESTPARK DRIVE
MCLEAN, VIRGINIA 22102

TCP SPECIFICATION
REVISION 1, JULY 4, 1979
LIST OF CHANGED PAGES

Title	A-10
ii	C-1
9	D-1
10	D-2
18	D-3
32	D-14
37	D-20
A-1	E-1
A-4	E-2
A-5	E-3
A-6	E-4

Attachment 1

TCP SPECIFICATION
REVISION 2, AUGUST 3, 1979
LIST OF CHANGED PAGES

Title

12

13

14

15

17

18

19

20

34

35

A-1

C-1

D-7

TCP SPECIFICATION

REVISION 3, FEBRUARY 8, 1980

LIST OF CHANGED PAGES

Title

10

18

30

B-1

C-1

C-44

D-4

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF TABLES

LIST OF FIGURES

SECTION 1 - GENERAL

1.1 SUMMARY

This document provides guidelines with which an AUTODIN II-standard Transmission Control Program (TCP) can be implemented. Included in this specification are:

- TCP Network Protocol (TCP-TCP)
- TCP-THP and TCP-SIP Interactions
- TCP Flow Control Mechanism
- TCP Connection Processing Mechanisms

There are certain concepts which are fundamental to the understanding of this specification: (1) AUTODIN II, (2) TCP's role in AUTODIN II, and, to a lesser extent, (3) THP's role in AUTODIN II. AUTODIN II provides a means by which diverse and geographically separate hosts (computers) and terminals may communicate. Users (hosts and terminals) need have no knowledge of physical characteristics (word size, line speed, link protocol, etc.) or limitations of another user to communicate with the other user; network components compensate for these differences. These AUTODIN II components fall into one of three general categories:

1. backbone components - Switch Control Module (SCM); Supervisory SCM (SSCM); Standby Processor (SB)

2. access components - Channel Control Unit (CCU); Terminal Access Controller (TAC); (note that using this and related specifications a user may develop his own access component)

3. Network Control Center (NCC)

The CCU and TAC are the components which provide general users access to AUTODIN II. User provided components, also in the access component category, could be a subscriber host or host front-end which supports the AUTODIN II interface protocols. The AUTODIN II interface protocols are:

- SIP (Segment Interface Protocol)
- TCP (Transmission Control Program)
- THP (Terminal-to-Host Protocol)

SIP is required by all users in order to access the Packet Switch (SCM). Users may employ a SIP to access the SCM and then develop any other levels of protocol (TCP and/or THP or equivalent or user defined protocols) to communicate via AUTODIN II within their own common-protocol user community. However, to communicate with users connected to AUTODIN II

via a TAC or CCU, other users must support SIP, TCP, and THP protocols that are fully compatible with those of AUTODIN II.

THP's role in AUTODIN II is to provide part of the compensatory service to users. THP, and other user-specific processes, such as Host Specific Interface (HSI) in a CCU or Terminal Handler (TH) in a TAC, know the specific characteristics of each user to be supported. THP, TH, and HSI use this knowledge to process data coming from and going to the user. The concept of NVT defines an AUTODIN II-network internal standard bidirectional, character-oriented device. THP is responsible for converting from user's to NVT format data going from his user to the network, and for converting from NVT to user's format data coming from the network to his user. (See Reference 9).

TCP provides the mechanism by which a process, such as THP, can interface with the network (via SIP) and, subsequently, other CCU/TAC TCPs and THPs. Essentially, TCP performs the mechanics of establishing, maintaining, and terminating an AUTODIN II "virtual connection." It is TCP's responsibility to control the flow of data so that the users on either end of the virtual connection appear to be communicating via a dedicated circuit.

The reader should be familiar with these concepts. More background information is available in the referenced documents (see Paragraph 1.2).

It should be noted that this document contains many examples which emphasize the MCCU design/environment. These are merely examples of one implementation of the TCP protocol. It is assumed that the reader is aware that many design decisions were based on the specific requirements for a CCU or TAC, and that the discussion of the TCP protocol itself consists of those portions of this document which relate to TCP-to-TCP communication. The THP-TCP and TCP-SIP interfaces shown here are specific to the CCU/TAC TCP implementation.

1.2 REFERENCES

The following documents provide additional information and detail on AUTODIN II, it's components and predecessors.

1. TELNET Protocol Specification, Network Information Center (NIC) 18639, latest issue (included in number 2, below)

2. ARPANET Protocol Handbook, NIC 7104, latest issue

3. Stanford Research Institute (SRI) Terminal-to-Host Protocol Specification, 15 July 1976

4. SRI Transmission Control Protocol Specification, 15 July 1976

5. Defense Communications Agency System Performance Specification (Type A) for AUTODIN II Phase I, latest issue

6. Western Union (WU) Proposal WU-301-1 AUTODIN II Phase I, April 1976, latest issue

7. WU AUTODIN II Design Plan, latest issue

8. AUTODIN II Design Plan Executive Summary, latest issue

9. AUTODIN II THP Transportable Specification, latest issue

10. AUTODIN II SIP Transportable Specification, latest issue

SECTION 2 - CONNECTION PROCESSING

2.1 ENVIRONMENT

For the CCU/TAC, THP is responsible for providing certain services to the user (human or process). These services include NVT conversion as discussed briefly in Paragraph 1.1 (see Reference 9). However, the mechanics of establishing, maintaining, and terminating a virtual connection are the responsibility of TCP. THP requests TCP's services on behalf of the user. TCP uses the services of SIP to send data to the SCM and, ultimately, to the TCP at the remote end of the virtual connection. TCP does all demultiplexing of the data stream coming from the SCM. That is, TCP determines to which connection the input segment is destined.

The subsequent paragraphs will describe TCP's environment in the MCCU, specifically, as well as in the AUTODIN II network, in general. These paragraphs are important for the understanding of the remainder of the document and the TCP protocol. In addition, Appendix E defines the basic units of transmission for AUTODIN II. The terminology defined in Appendix E is also important for the understanding of the remainder of this document.

2.1.1 Data Structures

Although data structures could be considered to be implementation dependent, the major TCP data structure is discussed here to point out the basic information about each connection required to perform TCP processing. There is only one major data structure which is used by TCP in the MCCU: the Transmission Control Block (TCB). There is one TCB per connection and it contains most of the information required to perform connection processing. The TCB is created when a valid open/listen request is received from THP (see Paragraph 2.2.1) and is deleted when close processing is completed (see Paragraph 2.4). There are other data structures used by CCU/TAC TCPs, which may be referenced in this document. These, however, are more implementation-oriented and are, therefore, not discussed in detail. The MCCU TCB is shown in its entirety in Appendix B. The SCCU and TAC TCBs are not shown, but are logically similar to that for the MCCU. That is, similar information is maintained for the connection, although the structures may differ.

2.1.2 TCP Processing

TCP is driven by the occurrence of significant events, such as, data coming in from the network, requests coming from THP, remote closing of the connection, and so forth. These significant situations are conveyed to TCP by either THP or SIP via inter/intra-process communications known as "events" in the CCUs and TACs. Events are processed in turn (first-in, first-out, by precedence) and carry enough information to complete the task required for the significant situation. For example, the situation of data arriving from the network is conveyed in the MCCU by a SIP-to-TCP Receive Data event. TCP events defined for the MCCU are described in Appendix C. These are conceptually similar to those used by the SCCU and TAC implementations, although the event format and the mechanics of inter/intra-process communication may differ. Any reference to an event in this document refers to an MCCU event.

In addition to the event mechanism, TCP is driven by the information in the T-segment header received from the remote TCP (see Appendix A). This information may be an acknowledgment for data sent, notice of a remote closing, or other TCP-TCP control information (see Paragraphs 2.1.4, 2.1.6, and 2.3.4).

2.1.3 TCP States

Based on TCP-TCP control information or events from THP and SIP, TCP changes the state of the virtual connection. This is reflected in the TCB for each connection. TCP "states" are defined here based on the MCCU implementation. It is conceivable that other implementations of TCP may not require as many, or may require more states. These states are used to keep track of one end of the virtual connection, i.e., they reflect the state of the local TCB. More extensive information as to the relationship between each state and the various possible stimuli (local close request, remote close request, data from the network, etc.) is given in Appendix D.

1. open - THP has issued a fully specified Open event, but the three-way handshake has not begun

2. listen - THP has issued a partially specified Open event (security, precedence, TCC, or destination address were not specified)

3. SYN sent - a SYN segment (request to synchronize send sequence numbers) has been sent, beginning the

three-way handshake

4. SYN sent/received - TCP has sent and received a SYN segment, but the acknowledgment has not been received for the SYN segment that was sent

5. established - the three-way handshake is complete; TCP has sent and received a SYN segment, and has received an acknowledgment for the SYN segment that was sent

6. local close received - TCP has received a Close event from the local THP, requesting a deferred (graceful) close of the connection, but has not sent the "FIN" segment; network data will not be delivered to the local THP (Receive events) for this connection once this state has been entered

7. remote close received - TCP has received a "status" control segment from the remote TCP, indicating that the remote TCP has begun close processing for the connection; TCP will continue to accept network data and pass it to THP (Receive events), but will accept no more THP letters destined for the network (Send events)

8. FIN sent - TCP has sent a FIN segment, beginning connection closure; no further data will be accepted from (Send events) or given to (Receive events) the local THP on this connection

9. FIN received - TCP has received FIN segment but is unable to send a FIN segment because all octets sent have not been accounted for as yet

10. FIN sent/received - TCP has sent and received a FIN segment, but has not received an acknowledgment for the FIN segment that was sent

11. closed - TCP has sent and received a FIN segment and has received an acknowledgment for the FIN segment that was sent, but TCP is waiting for all its queues to clear before deleting the TCB.

2.1.4 TCP Sequence Numbers

The use of sequence numbers is an integral part of TCP-TCP protocol; one is carried in every segment sent between TCPs. Each accountable octet (8 bits of user text or TCP control information) is assigned a sequence number. The se-

quence number in the send sequence number field of the T-segment header is that of the first octet in the segment. In segments containing data and accountable TCP control information the TCP control is represented by the first sequence number of that segment. Sequence numbers are assigned by the sending TCP and used by the receiving TCP for the following purposes:

1. reliability - With the segment sequence number and other T-segment header information, TCP can ensure that the segment received is a valid one for the indicated virtual connection. This is always important for data integrity and especially important in regard to error messages (see Paragraph 2.1.5) and other connection termination segments (FIN segments).

2. accountability - TCP acknowledges (to the sending TCP) for most segments received. The means by which this is accomplished is the sequence number of the segment, communicating the number of acknowledged octets.

3. order - Sequence numbers are used to order data being accumulated for transmission to the destination user. Each segment sent between source and destination TCPs may take a different path through the network, possibly arriving at the destination TCP before a segment which should logically precede it. TCP uses the sequence number of each segment (1) to order the data before it is sent to THP, or (2) to perform a requested function (e.g., flush) in the same relative order to the data as upon entry by the source user.

4. duplicate detection - TCP is able to detect and discard retransmitted segments based on the segment sequence number and the sequence numbers of previously received/acknowledged segments.

The send sequence number is a 31-bit value which is initialized for either side of the connection (each TCP's send side) during the three-way handshake of open processing (see Paragraph 2.2.2). CCU/TAC TCPs use the current value of the clock for the initial sequence number. This value represents the number of 10ths of a millisecond since the system (CCU/TAC) was initialized. The initial value is, therefore, a unique 31-bit value for each connection within the CCU/TAC.

The sequence number is incremented based on the amount of data (number of octets) or accountable control information sent on a connection. There are certain TCP-TCP control

segments which, when sent, will not cause the "next send sequence number" value to be incremented: ACK segments not carrying user text, WACK segments (acknowledgment for window open segments), status segments (see Paragraph 2.4), out-of-band interrupt segments (see Paragraph 2.3.3), and error message segments (see Paragraph 2.1.5). These are not subject to accountability.

All other segment types have unique sequence numbers: segments carrying user text, SYN segments, FIN segments, flush request segments, and WOPEN (window open) segments. When a sequence number is assigned to one of these segments, the "next send sequence number" value is incremented by the number of octets being sent (for data segments), including one for the control, or by one (for control-only segments).

Sequence number values for a virtual connection will wrap around eventually. That is, at some point in time when the next send sequence number is incremented, the value will go from a very large number (2,147,483,647) to zero. Therefore, although most of the time "lower" sequence numbers will be transmitted first and delivered first to the destination user, the possibility of wrap-around is there. TCP will take this possibility into account at all times when assigning, validating, and ordering sequence numbers. Neither the possibility, nor the contingency procedure, will be mentioned further in this document. Any discussion of sequence numbers and their relative position to one another (lower or higher) should be read assuming that wrap-around is being considered.

2.1.5 TCP-TCP Error Messages

There are five TCP error messages that may be sent/received which cause the receiving TCP to immediately consider the connection to be terminated. The normal connection termination procedures (see Paragraph 2.4) will not take place, in this case. There will be no acknowledgment sent to the TCP generating these messages. There will, however, be a validation of the receive segment sequence number to ensure that the error notice is legitimate. If the segment is valid, the connection will be closed and the local THP notified. The error messages are:

1. connection does not exist - This message is sent if there is no associated connection for a received segment, as determined during demultiplexing of the network data stream (see Paragraph 2.3.4).

2. security violation - This message is sent if the

security level of any incoming segment is determined to be inappropriate for the connection. There are three circumstances when this applies: (1) connection state is listen, the local user did not specify the security level in his listen request, and the security level of the incoming segment is higher than that authorized for use by the local user; (2) connection state is open, listen, or SYN sent, the local user did specify the security level in his open/listen request, and the security level of the incoming segment does not match the specified security; or (3) connection state is established and the security level of the incoming segment does not match the connection security level. The TCP generating the error message will remain in open, listen, SYN sent, or established state, as appropriate.

3. TCC violation - This message is sent (1) if the TCC of an incoming SYN segment does not match the TCC specified by a user in his open/listen request, or, (2) if the local user did not specify a TCC in his listen request and the TCC of an incoming SYN segment is not authorized for use by the local user. The TCP generating the error message will remain in or return to the open/listen state.

4. connection preempted - This message is sent if the established connection is being preempted (see Paragraph 2.4.3). The TCP generating the error message will continue close procedures for the preempted connection and, subsequently, establish the new connection (with another destination).

5. unacceptable SYN - This message is sent if a SYN segment is received (from same source as initial SYN) that is not a retransmission of the initial SYN segment when the state is SYN sent/received or established. The TCP generating the error message will also close the connection, notifying the local THP (via Close Return event in HCCU).

A TCP receiving an error message will ensure that the error is "believable" before terminating the connection. If the sequence number believability check fails, TCP will discard the error message segment. The validation consists of the following:

1. If the notification is "connection preempted," the sequence number contained in the acknowledgment field of the error message segment must be the preempting TCP's next send sequence number. The value, then, must be within the current receive

window for the TCP receiving the error message segment.

2. If any other error notification is received, the sequence number of the errored segment (e.g., the SYN segment) must be in the acknowledgment field of the error message segment. That value must, of course, be a sequence number for a segment sent by the TCP receiving the error message segment.

2.1.6 TCP-TCP Control Segments

TCP control segments are exchanged in support of the TCP-TCP protocol. These are defined below, as well as discussed in subsequent paragraphs and in Appendix D, TCP State Tables. It should be noted that those control segments which require acknowledgments always consume sequence number space (see Paragraph 2.1.4).

1. SYN segment - This control is used during open processing, specifically the three-way handshake (see Paragraph 2.2.2), to initialize the send sequence number for each TCP. The SYN segment must be acknowledged and may not carry user text. The SYN segment carries information (in the control data extension field) about the source user which is required to calculate the destination TCP's receive window and THP-TCP send window (see Paragraph 2.2.3). The approximate letter size, host-CCU or terminal-TAC line speed, and CCU- or TAC-SCH line speed are included in the SYN segment. If a non-CCU/TAC TCP does not use this procedure, i.e., does not include the user information in the control data extension field, default values (zero) for the letter size and line speed codes will be used in the window calculations.

2. FIN segment - This control is used during connection termination (see Paragraph 2.4) to ensure an orderly close of the connection. Final sequence number acknowledgment takes place at that time. The FIN segment must be acknowledged and represents the acknowledgment for all octets which have been accounted for by the destination TCP (sender of the acknowledgement). Although the TCP protocol allows data to be sent in a FIN segment, CCU/TAC TCPs will never send user text in this segment, but will accept text sent in a FIN segment by a non-CCU/TAC TCP. The FIN exchange is not always used to close the connection, as in the case of some TCP errors

or preemption. Error conditions, such as those reported in Send Data Return events or encountered in time out situations (see Appendix D), cause the connection to be closed using the FIN exchange. Other errors, such as security violation or preemption (see Appendix D), cause the error message to be sent and the connection to be closed without formal TCP-TCP protocol. In these cases, the TCP error message is sent (see Paragraph 2.1.5).

3. ACK segment - This control is used to acknowledge all accountable segments, including data segments and any control segment requiring accountability. The acknowledgment sequence number is contained in the segment and indicates the sequence number of the next octet expected on the connection. This implies that all octets with lower sequence numbers than the sequence number in the acknowledgment field have been accounted for by the TCP sending the ACK segment. A TCP receiving the ACK segment will ensure that the acknowledgment sequence number is indeed for octets sent. The value must be within a range from, and including, the "oldest" unacknowledged octet to, and excluding, the next send sequence number. The ACK segment itself is not acknowledged. However, if it carries data, which is permitted, the data is subject to accountability.

4. *OPEN segment - This control is used to notify source TCP that destination TCP's (sender of *OPEN segment) previously zero receive window is now non-zero. The segment must be sent to open the receive window, if the window is ever decreased to zero, and it may be sent into a zero window. That is, the TCP receiving a *OPEN will always process it. A *OPEN segment must be acknowledged by a *ACK segment and may never carry data. In addition, it should be noted that, once the *OPEN is received and a *ACK is sent to acknowledge it, data and/or other control segments may be sent on the connection. The destination TCP will be ready to accept segments once it has sent the *OPEN. The acknowledgment of the *OPEN is expected and the *OPEN will be retransmitted, as explained in Paragraph 2.3.2; however, this need not have taken place in order for the destination TCP to consider its receive window to be open.

5. *ACK segment - This control is used to acknowledge a *OPEN control and may not carry data. The *ACK itself is not subject to accountability.

6. Status segment - This control is used to inform the remote TCP that a local close is in process and to provide accountability for data received that cannot be delivered to THP due to the previous close request. The segment is sent to remote TCP after a local close request has been received, but before it is time to send a FIN segment (see Paragraph 2.4). The status segment is not acknowledged and may not carry data. The segment sequence number field will contain the sequence number plus one of the last octet successfully delivered to THP. The segment acknowledgment field will contain the sequence number of the next octet expected.

7. Error Message segments - The error message segment, as discussed in Paragraph 2.1.5, is used to notify the remote TCP that an error was detected in a segment received or that connection preemption has taken place. The segment is not acknowledged and may not carry data. The TCP receiving such a segment will, under most circumstances, close the connection (see Appendix D).

8. Flush segment - This control requests that the remote TCP perform a flush function for the virtual connection (see Paragraph 2.3.3). The flush segment never carries data, but is subject to accountability.

9. Out-of-Band Interrupt segment - This control requests that the remote TCP perform the out-of-band interrupt function (see Paragraph 2.3.3). The out-of-band interrupt segment may not carry data and is not subject to accountability.

2.1.7 Virtual Connection Protocol Addressing

Virtual connection addressing is accomplished by TCP through a scheme based on "sockets." In general, a socket is the concatenation of an internetwork identifier, a TCP identifier, and a port identifier. This socket defines one end of a TCP virtual connection. A pair of sockets, one describing the source and one describing the destination, fully specifies a TCP virtual connection.

When a host process or terminal user makes a request to open a connection, all information required to fully specify the destination (foreign) and source (local) sockets must be given. When a destination TCP receives a request for a con-

nection, the received T-segment header will contain two fully specified socket addresses. For a connection to be established, a destination socket must exactly match one waiting for a connection (via a previously issued listen or open request at the destination). Once the connection is established, the two socket addresses, source and destination, will uniquely identify a TCP virtual connection.

Paragraph 2.2 discusses, in more detail, the procedures required for establishing a connection. Paragraph 2.3.4 describes, in detail, the demultiplexing of the network data stream, in which TCP uses the socket addresses to determine the appropriate connection. This paragraph is meant to define the concept of socket and the two categories of subscriber addressing: CCU addressing and TAC addressing.

CCU Addressing

Since the CCU is identified by one network subscriber address, the packet switch can route data to only the CCU. The CCU must determine to which host user (host-CCU channel) the connection refers. When a host user enters the open/listen request, he must include the complete local user identification. This includes both the local subscriber address and the local port ID. The subscriber address is the same for all users of any one CCU; the port ID is designed to identify a specific user. The port ID entered by the user becomes associated with the particular host-CCU channel. This is the concern of THP (Reference 9). It should be noted that every channel between a host and CCU can be associated with the same port ID, if it is desired. The unique address required to associate a connection to a specific user (channel) is provided by the fully specified socket, described above. When THP submits an open or listen request to TCP, a local connection name (LCN) is assigned to the connection. This is a shorthand notation for the fully specified socket pair and provides a means for THP and TCP to refer to the connection internally. To summarize: TCP receives a segment which contains both the source and destination user identification (the fully specified socket pair). TCP associates the segment with a particular connection using this socket pair. TCP relays the information (data or control) to THP using the LCN to describe which connection is concerned. THP correlates the LCN and the specific host-CCU channel.

The port ID is actually broken down into three parts: (1) function suffix, (2) user ID, and (3) static/dynamic port identifier. To the CCU (THP or TCP) the various portions of the port ID are recognized for only one purpose: TCP extracts the user ID to perform a security/precedence/TCC validation check to ensure that the particular user ID is authorized to use those values for security, precedence, and

TCC that were specified in his open/listen request. Under all other circumstances the port ID is not important as an entity. Of course, it is part of the unique fully specified socket pair. The port ID is used, rather, by the host to determine to which process the connection refers.

TAC Addressing

Each terminal attached to a TAC is represented by a unique subscriber address to the packet switch. The fully specified socket for each user will consist of only the unique subscriber address; no additional information (port ID) is required. TCP must still demultiplex the network data stream, determining the appropriate connection based on the socket pair in the received segment.

2.2 ESTABLISHING A CONNECTION

There are several steps in establishing a virtual connection between two CCU/TAC users (host process or terminal user). The steps, summarized here and detailed in subsequent paragraphs, are those taken by an MCCU THP/TCP, however, they are logically similar to those taken in an SCCU or TAC.

TCP becomes involved in open processing upon receipt of an Open event from THP. This event may have been generated as a result of a user request (THP open or listen command) or an auto-open/auto-listen function (see Reference 9). In any case, TCP must validate the open request and notify THP immediately of the results (Open Return event). The connection will not be established, however, until the TCP-TCP three-way handshake has taken place. At that time TCP will notify THP via the Open Complete event, and THP will notify the user, as required.

Paragraphs 2.2.1 and 2.2.2 will describe these two procedures, open request processing and the three-way handshake, concentrating on what is required to successfully establish a virtual connection, as well as the various conditions that can make an open attempt unsuccessful. Paragraph 2.2.3 describes the calculation of the THP-TCP send window and TCP-TCP receive window. Paragraph 2.2.4 provides a complete scenario for opening a connection. Appendix D, TCP State Tables, will aid in understanding and expand upon this discussion.

2.2.1 Open Request Processing

The Open event from THP to TCP defines the parameters required to establish and maintain a virtual connection. These parameters are:

1. precedence - A value from zero to fifteen representing the category and type of data to be sent by the local user on the new connection. There are four categories: I, II, III, and IV, with I being high (most expeditiously handled). For each category there are four subclasses. The relationship between Open event parameter values, category, and subclass, as well as a brief description of the type of data, is shown in Table 1. The terminology used in this table is the same as that used in the AUTODIN II System Performance Specification (Type A), Reference 5.

TABLE 1. AUTODIN II TRAFFIC CATEGORY MATRIX

value	category	criticality*	subclass	traffic type
15**	I	Y,W,Z	1	AUTODIN II control messages
14**	I	Y,W,Z	2	subscribing network control messages
13	I	Y,W,Z	3	Critic/ECP messages
12	I	Y,W,Z	4	subclasses A, B, C1, and C2
11	II	O	A	interactive traffic
10	II	O	B	query/response traffic
9	II	O	C1	narrative data
8	II	O	C2	bulk data
7	III	P	A	interactive traffic
6	III	P	B	query/response traffic
5	III	P	C1	narrative data
4	III	P	C2	bulk data
3	IV	R	A	interactive traffic
2	IV	R	B	query/response traffic
1	IV	R	C1	narrative data
0	IV	R	C2	bulk data

*criticality: Y,W = flash override
Z = flash
O = immediate
P = priority
R = routine

**category I, values 14 and 15, are for network use only and, as such, are not available to the subscriber

2. security - A value from zero to 15 representing the security level for all data to be sent by the local user on the new connection. Security levels represented are: unclassified, EFTO, restricted, confidential, secret, top secret, and MMMM (0 to 6, respectively), and PROG (15 - not available for use by subscribers). Seven through 14 are not assigned.

3. transmission control code (TCC) - A value defining the community of interest to which both the source and destination subscribers belong. There are 512 TCC codes for AUTODIN II. Users should contact DCA Code 530 for assignment.

4. destination subscriber address - A value defining the network address of the destination subscriber with which the connection is to be established. The value has a range from zero to 65535.

5. destination port ID - A value representing additional addressing information for the destination subscriber. The value is zero if the destination subscriber is a terminal connected to a TAC, and is, therefore, applicable only to CCU subscribers. The port ID defines the host-CCU channel for THP's use, and also contains user (process) identification for use in the host.

   This 16-bit value is actually divided into three fields: bit 0 (least significant bit) is the static/dynamic port indicator; bits 1-11 represent the user ID; and bits 12-15 represent the function suffix. (See Paragraph 2.1.7).

6. source port ID - A value representing additional addressing information for the local user. The value is zero if the local user is a terminal connected to a TAC, and is, therefore, applicable only to CCU subscribers. The purpose and format are identical to the destination port ID.

7. preemptability - A value (zero or one) indicating whether the new connection may be preempted (one) or not (zero). If this value is zero, TCP will not consider the connection to be a candidate for connection preemption. Resource preemption (see Paragraph 2.4.3) is not affected by this indicator, however.

To TCP the Open event may represent an open or listen request. If all the above parameters are specified in the Open event, TCP considers it to be a request for an open connection, and places the connection in the open state. If

any parameters are unspecified in the Open event, TCP considers it to be a listen request, and places the connection in the listen state. The only difference is that for a listening connection TCP will block any attempt by THP to send data (Send events) on the connection (see Paragraph 2.2.3). It should be noted that if a user were to enter a THP listen command with all parameters specified, the Open event would be fully specified and TCP would consider the connection "open." THP, however, considers the connection to be "listening" and would not send data to the network (see Reference 9).

TCP will ensure that the security, precedence, and TCC, if specified, are within the bounds authorized for this user. TCP has no way, however, to validate the destination subscriber address or destination port ID values. TCP also ensures that resources (buffer space) permits the establishment of another connection. If resources are critical, i.e., space for data structures and data buffers does not exist, TCP may deny the open request or may preempt lower precedence connections to acquire space for the new open request. An Open Return event will be sent to THP indicating acceptance or rejection of the open/listen request. If the request was valid, TCP will set the connection state to reflect the open or listen state, as appropriate.

2.2.2 TCP-TCP Three-way Handshake

TCP's "three-way handshake" is the mechanism used to establish the network virtual connection. The three-way handshake, from the viewpoint of each TCP, consists of sending a request for synchronization of the send sequence numbers, receiving the similar request from the remote TCP, sending an acknowledgment for the request that was received, and receiving an acknowledgment for the request that was sent. The request for synchronization is called a "SYN" segment.

The above opening procedures (Paragraph 2.2.1) are preliminary steps which leave the subscriber in a TCP state (open or listen) ready for a virtual connection. The three-way handshake will begin when the "opening" user enters data destined for the "listening" user. It is assumed for the purpose of this discussion that one user will enter a THP open command, causing a fully specified open state in TCP, and that the other user, connected to AUTODIN II through another access area (CCU/TAC), will enter a THP listen command, creating a partially specified open state in TCP.

When TCP receives a valid Send event from THP for a connection in the open state, TCP begins the three-way handshake, sending a SYN segment to the remote TCP. The remote TCP

will determine the connection (or local subscriber) associated with the incoming segment (see Paragraph 2.3.4) and will make the following additional checks before acknowledging the SYN segment and returning a SYN segment of its own. It should be noted that preemption is considered in determining the subscriber to be associated with the incoming segment. The security, precedence and TCC of the SYN segment are extracted from the BSL by SIP and passed to TCP for analysis, i.e., these are not in the TCP header.

1. connection security - If the connection security has been specified by the user in the open request, the security level of the SYN segment must match. If the security was not specified, the security level of the SYN segment will be used, but it must be authorized for use by the user.

2. connection precedence - If a precedence is specified in the opening/listening request, it will be the precedence level for that (local) user's send side of the virtual connection. The precedence specified in the SYN segment will be the precedence level for the local user's receive side. If the precedence was not specified in the local user's listen request, the local user's send precedence level will be set to that of the SYN segment or to the maximum authorized for use by this user, whichever is lower. There may be, therefore, two precedences associated with this connection: user's send precedence and user's receive precedence. If the connection becomes a candidate for preemption (see Paragraph 2.4.3), the higher of the two will be used in determining if preemption is allowed.

3. TCC - If the user specified a TCC in his open request, the SYN segment TCC must match. If the TCC was not specified, the SYN segment TCC will be used as long as it is authorized for use by the user.

4. foreign address - If the user specified a foreign subscriber address and foreign port ID, the SYN segment addresses must match. If the address was not specified, the SYN segment address is used.

If the above criteria are not satisfied, (e.g., an associated connection can not be found, a local user specified parameter is not "matched" in the SYN segment , or, for a parameter that was not specified by the local user, the SYN segment value is not authorized for use by the user), TCP will not complete the three-way handshake. The SYN segment will be discarded and TCP will return an error message to the TCP that sent the unacceptable SYN segment. That TCP will notify THP that the connection is closed (Send Return

and Close Return events).

If all the checks are successful, the three-way handshake will continue. Once the remote TCP sends an acknowledgment for the SYN that was received and initiates a SYN segment of its own, and the initiating TCP acknowledges the remote SYN segment, the connection is established. An Open Complete event is sent to THP, in the HCCU, to indicate that the connection has been established. A similar vehicle is used for the notification in the SCCU and TAC.

2.2.3 TCP Window Determination

Each TCP must determine two send windows, one to be used by the local THP in sending data to the local TCP (Send events) and one to be used by the remote TCP in sending data to the local TCP. The first window is called the THP-TCP send window and is expressed in number of letters which THP may have outstanding to TCP. In the HCCU, this represents the number of Send events which may be queued to TCP without receiving a Send Return event for the "oldest" Send event queued. The second window is this TCP's receive window, the remote TCP's send window. The value is conveyed in the T-segment header (see Appendix A) in every segment sent to the remote TCP, and represents the number of octets which may be sent to this TCP before waiting for acknowledgment.

For subclasses A and B of precedence categories II, III, and IV and subclasses 1 and 2 of precedence category I (see Table 1), the window determinations are quite simple. The nature of these traffic types permit certain assumptions about data exchange on a connection with a precedence in this group. Essentially, data flow will be one-way at a time rather than in a steady stream for either or both sides of the connection. In addition, the amount of data in each transmission will be within one THP letter. Therefore, TCP will establish each window with a value equivalent to one letter buffer. For the local THP send window the value will be one, indicating that one Send event may be sent at a time. For the remote TCP's send window the value will be equivalent to the letter size (conveyed in the SYN segment from the remote TCP) for that TCP's user. This window is conveyed in the T-segment header, in every segment sent by the destination TCP, as the destination TCP's receive window, and will be expressed in the number of octets (8-bit bytes) that the destination TCP is willing to receive. The THP-TCP send window value will not change throughout the life of a connection having a precedence in this group. The TCP-TCP receive window may go to zero, in the normal processing of a connection, but will not be increased to more

then the equivalent of one THP letter size. The TCP-TCP receive window will go to zero only if resources are needed to service a higher precedence connection.

The window determinations for the narrative and bulk data subclasses (see Table 1) are more complex. The THP send window is initially calculated during open request (Open event) processing. If the open request is partially specified, a listen request, (see Paragraph 2.2.1), the window value will be zero, blocking data from the user until a connection is established. If the open request is fully specified, the window value will be one, allowing one Send event. This send request will actually begin the three-way handshake (see Paragraph 2.2.2). The THP send window and TCP receive window will be recalculated during the three-way handshake and any time another connection terminates in the same CCU/TAC. Both windows are calculated using the information conveyed in the SYN segment during the three-way handshake, as well as, information about the local subscriber and CCU/TAC resources. The following items are considered in the window computations: source user's approximate letter size, speed of the line between source user (host or terminal) and access component (CCU/TAC), speed of the line between source access component and the source SCM (access line speed), access line speed between destination SCM and destination access component, line speed between destination access component and destination user, and local resource availability.

2.2.4 Connection Establishment Scenario

For clarification in reading this scenario certain terms must be defined. An "opening user" is one (host process or terminal user) which requests that a connection be opened to another user. A "listening user" is one which indicates that he is ready for a connection to be established. Opening THP and TCP are the access area component protocols which serve the opening user. Listening THP and TCP are those that serve the listening user.

This scenario will provide the overall view of the opening of a connection. Only the major participants are noted here. That is, HSI/TH and SIP are not considered as they provide only I/O driver/link protocol support. The scenario is also not explicit as to which access components are being used, although the events mentioned are those of the MCCU. The scenario presents a typical THP-TCP interface and TCP-TCP protocol. It should also be noted that the first three steps of the scenario may be completed any time before the opening TCP sends the SYN segment to the network (step

9), or more precisely, before the listening TCP receives that SYN segment.

1. User A enters a valid THP listen command sequence, indicating that a connection may be established with any requesting remote user. User A will now be referred to as the "listening user," and the THP and TCP serving user A will be referred to as "listening THP" and "listening TCP," respectively.

2. Listening THP validates user A's listen command sequence, determines that it is a valid request, and queues a partially specified Open event to TCP.

3. Listening TCP validates the Open event, creates a TCB for this user, puts the TCB in the listen state, and queues an Open Return event to THP.

4. Listening THP dequeues the Open Return event, notes that the request was successful, but takes no further action as the user is in the THP listening state.

5. User B enters a valid THP open command sequence, requesting that a connection be established with user A. User B will now be referred to as the "opening user," and the THP and TCP serving user B will be referred to as "opening THP" and "opening TCP," respectively.

6. Opening THP validates user B's open command sequence, determines that it is a valid request, and queues a fully specified Open event to TCP.

7. Opening TCP validates the Open event, creates a TCB for this user, puts the TCB in the open state, and queues an Open Return event to THP.

8. Opening THP dequeues the Open Return event, builds a characteristics option record (see Reference 9) in a to-network letter buffer, and queues a Send event to TCP, requesting that the letter be sent to the network. The user is in the THP opening state, waiting for characteristics processing to be completed.

9. Opening TCP dequeues the Send event from THP, determines that the send request is valid, and begins the three-way handshake to establish the connection. TCP sends a SYN segment to the network, addressed to user A and changes the TCB state to SYN sent.

10. Listening TCP receives the SYN segment, makes several checks to verify that the SYN segment is acceptable (see Paragraph 2.2.2), builds a SYN segment with an acknowledgment (ACK) for the received SYN segment, transmits the segment to the network, and changes the TCB state to SYN sent/received.

11. Opening TCP receives the SYN segment with an ACK for the SYN segment previously sent. TCP builds a data segment to be sent (to the listening TCP) which contains the initial THP letter and an ACK for the received SYN segment. The TCB state is changed to established and an Open Complete event and Send Window event (increasing THP-TCP send window) is queued to THP.

12. Listening TCP receives the ACK segment with data for the listening user, determines that the ACK is for the SYN segment previously sent, changes the TCB state to established, queues an Open Complete event to THP, and delivers the received data to THP (Receive event).

13. Listening THP dequeues the Open Complete event, notifies user A that the connection has been established, builds a characteristics option record in a to-network letter buffer, and queues a Send event to TCP as soon as the TCP send window opens (Send Window event). The THP state is active, waiting for characteristics processing to be completed.

14. Opening THP dequeues the Open Complete event, notifies user B that the connection has been established, but takes no further action as it is waiting for the characteristics record.

15. THPs will complete characteristics processing (see Reference 9) and then begin to exchange user data, but the virtual connection has been established from TCP's viewpoint at this time.

2.2.5 Move Connection

The move connection feature is primarily a THP function and is available in the MCCU only. It involves the relocation of an established virtual connection from one CCU user to another, essentially from one MCCU-host channel to another. TCP's role in this process is minor and involves performing a function similar to that for the Open event, thus the fea-

ture is discussed here.

It is conceivable that on an MCCU there would be one MCCU-host channel reserved for use by a host log-on process. An AUTODIN II user would request to be connected to that log-on subscriber/port ID in order to "sign on" the host. Once the log-on procedure is completed, the user would be transferred to another host process on another MCCU-host channel. The transfer would occur when the log-on process uses the THP move command. Because of the nature of this feature, there are certain restrictions. The reader should consult Reference 9 for details.

TCP would be asked via the Move Connection event to verify that the new user is authorized to receive the established connection. The security, precedence, and TCC check performed during open processing would be performed for the new user. The Move Connection Return event conveys the result of the validation. From TCP's viewpoint the connection will not have changed, i.e., THP will do all "mapping" of data to the new user. The log-on channel will then be available for a virtual connection with another subscriber/port ID.

It should be noted that the above description applies to the MCCU implementation. Other implementations of this function may handle the move differently, as required by specific user characteristics.

2.3 MAINTAINING A CONNECTION

2.3.1 Flow Control

Once the three-way handshake is complete and the virtual connection has been established, the TCB state will be set to established until close processing begins for some reason. TCP's major goal while maintaining a virtual connection is to sustain a rate of data transfer so as to facilitate a constant data flow from user to user. To accomplish this, a flow control mechanism has been implemented which involves not only the two TCPs but also other components in the two access areas (CCU or TAC), i.e., HSI, TH, and THP. The access area component included in this discussion is the MCCU. Similar mechanisms provide the SCCU and TAC flow control. TCP-TCP flow control is inherent in the protocol and does not vary across access areas. The reader can refer to Appendix E (Volume VII) of the AUTODIN II Design Plan (Reference 7) for details and the mathematical expression of the flow control algorithm. The terms source and destination indicate which access area is being discussed. That is, "source" refers to the CCU serving the originator of the data; "destination" refers to the CCU serving the receiver of the data.

user-to-network path

1. Source HSI will have up to a maximum number of From User events outstanding (From User Return events not received) to source THP for any one channel. The maximum number will be constant and will be based on the speed of the input line and other user specific information. When the maximum number is reached HSI will begin holding off the host, as appropriate for the link protocol. For example, a binary synchronous communications (BSC) protocol has a wait before transmission feature which would allow HSI to acknowledge for a transmission while asking the host not to start another transmission at this time.

2. Source THP will have up to a maximum number of Send events outstanding (Send Return events not received) to source TCP for any one connection. The maximum number, called the THP-TCP send window, will be dynamic and will be initialized (Open Return event) and changed (Send Window event) by TCP (see Paragraph 2.2.3). The initial value for an opening user in the MCCU will be one. A listening user will have a zero send window, initially. Source TCP will refine and increment the window, as

required, during the lifetime of a connection. When the current maximum send window is reached, THP will begin holding off HSI because From User events will not be processed and, therefore, From User Return events will not be queued to source HSI. This action (or lack thereof) will cause source HSI's hold-off of the host, as described above.

3. Source TCP will have up to a maximum number of octets outstanding (unacknowledged) to destination TCP. The maximum number, called the TCP-TCP window, will be dynamic and will be controlled by the destination TCP. The window will be conveyed on every segment transmitted by the destination TCP as the destination TCP's receive window. Source TCP will send up to that many octets before waiting for acknowledgment from destination TCP. When the maximum outstanding is reached and the destination TCP has stopped acknowledging for octets, source TCP, as a result, will be unable to send an acknowledgment to source THP (Send Return event). This, in turn, will hold off source THP because the number of THP's outstanding Send events will reach the maximum. The hold-off will then reach HSI and eventually the host.

network-to-user path
---

4. Destination TCP will have a maximum number of Receive events outstanding (Receive Return events not received) to destination THP. The maximum number (two in the MCCU) will not vary throughout the connection. Based on this maximum, destination TCP will regulate the source access area data flow. If Receive Return events are not sent from destination THP to TCP, destination TCP is unable to send TCP-TCP acknowledgments for octets received from source TCP. This situation will eventually stop source TCP from sending additional octets, and so forth (see number 3, above).

5. Destination THP will have up to a maximum number of To User events outstanding (To User Return events not received) to destination HSI. The maximum number will be constant and will be based on output line speed and other user specific information. When the maximum is reached THP will discontinue queueing Receive Return events to TCP. This will cause the acknowledgment rate by destination TCP to slow which will, in turn, slow source TCP's transmission rate.

6. Destination HSI will be regulated by the speed of the output line as well as the link protocol. To User Return events will not be queued to THP until protocol acknowledgements are received for corresponding data. If there is not a protocol acknowledgement used on the particular channel, such as for an asynchronous protocol, HSI will wait until the data has been transmitted on the link (I/O completion) before queueing the To User Return event to THP.

2.3.2 Processing Data for a Connection

Data coming from the user going to the network must be formatted as required by the packet switch (SCM). TCP is responsible for adding a TCP header to user data, THP letters in the case of a CCU or TAC, in preparation for the addition (by SIP) of the binary segment leader (BSL). This TCP segment, or T-segment, format is a requirement for all segments being exchanged among AUTODIN II-standard TCPs. The T-segment is pictured in Appendix E. The T-segment header, which carries TCP-TCP control information, is explained in detail in Appendix A.

TCP performs several functions in processing data for the network. These basic functions are summarized below as they apply to the MCCU. Similar methods of handling the functions exist in the SCCU and TAC implementations, as they should in any TCP implementation conforming to the AUTODIN II-standard protocol. The scenario assumes that a connection has been established.

1. validating THP send request

   THP will request that TCP send a THP letter to the destination user. In the MCCU this request will be in the form of a Send event. Whatever the vehicle, TCP must verify that the request is a valid one, ensuring that the data can be sent in the current connection state. If the current state is not established or open, TCP will reject the send request.

2. segmentizing the data

   The major TCP function in user-to-network processing is determining when the data or control information can be sent, building a T-segment header, and formatting the segment to be transmitted to the network. The first step is the major one, TCP

computes the number of octets which may be sent on the connection (available window). The available window will be based on the send window (destination TCP's receive window) and how many octets have been sent to, but not acknowledged by, the destination TCP. That is, the available window equals the send window minus the difference between the next sequence number to assign and the last sequence number acknowledged by destination TCP. If the available window is large enough to satisfy the next send request, the T-segment header is built and the segment is sent to the network, via SIP (Send Data event in the MCCU). If the available window is not large enough to send the entire letter and there are already segments on TCP's retransmission queue, TCP will not send the segment. If there are no segments on the retransmission queue, TCP will send the segment regardless of the send window.

Control information, e.g., acknowledgment (ACK) segments not carrying user text, out-of-band interrupt segments, status segments, WACK segments, and error message segments (see Appendix A and Paragraphs 2.1.4 and 2.1.6) do not consume sequence number space and, therefore, can be sent at any time, even if the available window is zero.

3. acknowledgments for segments sent

There are two types of acknowledgments which are of interest to TCP. The first is the acknowledgment from SIP indicating that the Send Data event has been processed. In the MCCU this is conveyed by a Send Data Return event and "processed" means that the segment has been sent to and acknowledged by (at the link level) the source SCM. If an ADCCP error occurred or network nondelivery notice, (e.g., destination subscriber down), upon transmission of the segment to the SCM, the Send Data Return event will be marked accordingly. TCP will attempt to retransmit segments falling into this category (see below). There will be a fixed number of retries before the connection will be terminated. Other errors returned in the Send Data Return event are invalid security, precedence, TCC, or destination address. These errors are fatal and cause immediate connection closure.

The second acknowledgment of importance to TCP is that sent by the destination TCP for octets delivered to the destination user (THP). This acknowledgment comes in the T-segment header of a segment

received from the destination TCP. If the acknowledgment is not received in a predetermined time, TCP will retransmit the segment (see below). Again, only a set number of retries will be attempted before the connection is closed.

When an acknowledgment is received from the destination TCP, source TCP validates it, ensuring that the acknowledgment sequence number value is between the oldest unacknowledged sequence number (inclusive) and the next send sequence number (exclusive). If it is not, the ACK control is ignored. If the value is valid, TCP will remove all acknowledged octets from the retransmission queue. The retransmission queue contains information concerning all segments that have been sent to the network but have not been acknowledged by the remote (destination) TCP. If a partial segment is acknowledged, i.e., the acknowledgment sequence number is lower than the sequence number of the last octet included in the segment, TCP will recompute the start of data in the segment so that only unacknowledged octets will be retransmitted. As complete TMP letters are acknowledged, TCP queues Send Return events to TMP for corresponding Send events.

4. Retransmitting segments

As discussed above, TCP will retransmit segments under two conditions: (1) when notified via Send Data Return event that transmission to the network was unsuccessful, and (2) when a predetermined amount of time elapses after successful transmission to the source SCH, with no acknowledgment from destination TCP for the octets sent. The current TCP send window (at the time retransmission is being considered) is taken into account. If the sequence number of the the first unacknowledged data byte of the segment lies within the current TCP send window, the segment (or remainder thereof) will be transmitted. A retransmission timer will be started. The length of the timer may vary, depending on timer type, but the retransmission mechanism remains the same. If the time elapses, TCP will resend the appropriate segment, as permitted by the send window. When maximum number of retransmissions have been sent, the connection will be closed. It should be noted that closure as a result of network transmission failure will not include a FIN exchange with or sending an error message to the remote TCP. Essentially, the close processing will be local.

The timeout value for the network transmission failure retry will be initially set to one second. The timeout value for waiting for destination TCP acknowledgement will be based on the line speed of the destination user CCU-host channel for data segments, and will be approximately 10 seconds for certain control only segments (SYN, FIN, and flush request).

There are several types of control segments which are not considered for retransmission. These are: ACK, status, out-of-band interrupt, HACK, and TCP error message segments. The information carried by the ACK and status segments will be sent again as additional segments are received from the remote TCP. There is no TCP accountability of these types of segments. In addition, TCP error messages denote that there is a catastrophic error in the system at the TCP protocol or network protocol level, requiring that the connection be closed. Relying on the other TCP to acknowledge for these messages is questionable.

2.3.3 Interrupt/Flush Feature

The TCP interrupt/flush feature is actually two separate functions, as defined for AUTODIN II users. Both functions are performed at the request of THP.

A request for a "flush" causes TCP to discard all not yet segmentized send requests (Send events) and send a control segment with the flush indicator set to the remote TCP. The remote TCP will discard all segments having sequence numbers lower than the sequence number of the flush control segment and pass the flush request to THP (Interrupt Return event in the MCCU). THP will then discard all data not yet queued to HSI for delivery to the user. In the CCUs and TAC, all requests for the TCP interrupt/flush feature are generated by THP interrupt functions (see Reference 9). TCP also uses the flush feature for an immediate close of the connection (see Paragraph 2.4.2).

A request for an "interrupt" causes TCP to send a control segment with the out-of-band control indicator set to the remote TCP. The remote TCP will notify the remote THP, via the Interrupt Return event, that an out-of-band interrupt has been requested. This feature is referred to as interrupt function 10 (IF10) in the THP specification (Reference 9).

2.3.4 Processing Data from the Network

Although SIP provides CCU-SCM network protocol support, it does not process the network data stream other than to analyze and remove the binary segment leader (BSL) from each segment. The received segments are passed to TCP via Receive Data events in the MCCU. TCP will do all the processing necessary to associate the segments with and deliver the segments to the appropriate virtual connection. This processing can be broadly categorized under three general areas, as discussed below:

1. demultiplexing the network data stream

   There are two types of segments received by TCP: those with data for the user (data segments or data/control segments) and those without data for the user (control only segments). For each received segment there are certain steps in attempting to associate the segment with a local user and his active connection, if any, and in determining the processing requirements for the segment. These steps involve the addresses associated with local subscribers and with the received segment.

   For each local TAC user there is a unique network subscriber address. For each CCU user there is a common network subscriber address (for the CCU) and a subscriber port ID, which is not necessarily unique (see Paragraph 2.1.7). If a user has entered an open or listen request, via the THP Open event, TCP will be aware of the user, that is, will be aware of the subscriber address/port ID. If the user's open or listen request specified a destination subscriber, the local address will be associated with the foreign address (subscriber address/port ID of the destination user). These two addresses make up a "fully specified" address (socket pair) which is associated with the virtual connection once it is established. Each fully specified address is unique within any one CCU or TAC. If the user's listen request did not specify a destination subscriber, the local address will not be associated with a foreign address and is, from TCP's viewpoint, a "partially specified" address.

   When a segment is received from the network, TCP extracts the fully specified address from the segment and attempts to match this with one of the fully specified addresses associated with that CCU or TAC.

If a fully specified match is not found, TCP attempts to match the partially specified address (destination address of the received segment, actually the address of the local user) with one of the partially specified addresses of which TCP is aware for that CCU or TAC.

If a partially specified match is still not found, TCP uses the partially specified (local) address from the segment, ignoring the foreign address of the received segment again, to match on any similar local address. The local address may be associated with another foreign address, in this case. If a match is found, TCP will analyze the associated TCB for possible preemption (see Paragraph 2.4.3).

If a match was not found on any of the above attempts, or if preemption was not possible, TCP will return a TCP-TCP error message segment to the sender of the segment, with "connection does not exist" indicated.

If a match is found, TCP will perform processing as required by the received segment type and the current state reflected in the TCB. Specific information can be found in Appendix D, THP State Tables.

2. reassembly queue

Because segments are either data segments or TCP-TCP control segments or data segments with control information, one of the major TCP functions for the network-to-user path is assembling the data to be given to THP. Due to the nature of the packet switch network, segments do not necessarily arrive at the destination in the order sent from the source TCP. For this reason, TCP must accumulate and reorder segments, based on the segment sequence numbers, before passing the data on to THP. Destination TCP will not send acknowledgment segments to source TCP for an octet until all octets with "lower" sequence numbers have been received, essentially, until the data has been acknowledged by a Receive Return event from THP. TCP always sends a complete THP letter to THP. The end-of-letter indicator is part of the T-segment header control information.

In performing the reordering function, TCP ensures that duplicate segments are discarded and that this TCP's receive window has been honored. To perform these checks, TCP considers the beginning sequence number (one in segment sequence number field of

T-segment header) and the ending sequence number (beginning sequence number plus text length minus one) for the incoming segment and the current receive acceptability range. The receive acceptability range is determined by the sequence number of the last acknowledged octet plus one (left edge of receive range) and the sequence number of the last octet which will be accepted (left edge plus receive window minus one). The following rules apply:

1. If the starting sequence number lies to the right of the receive acceptability range or the ending sequence number lies to the left of the range, the segment will be discarded.

2. If the starting sequence number lies to the left of the receive acceptability range and the ending sequence number lies within the range, the start of the segment shall be adjusted to coincide with the current left edge. The octets preceding that have already been accepted, processed, and acknowledged.

3. If the starting sequence number lies within the range and the ending sequence number lies to the right of the range, the entire segment will be accepted, if, in addition, no segments are currently on the reassembly queue. If there are other segments on the reassembly queue, this segment will be discarded.

4. If the starting sequence number lies to the left of the range and the ending sequence number lies to the right of the range, the segment will be discarded.

5. If the starting and ending sequence numbers lie within the range, the segment will be inserted, in order, in the reassembly queue. When an entire letter has been accumulated and if there can be another letter sent to THP (see Paragraph 2.3.1), TCP will send the letter to THP.

3. sending acknowledgment (ACK) segments

The ACK segment is issued by the destination TCP to provide accountability and receive window information to the source TCP. When the ACK control is sent, the acknowledgment field contains the sequence number of the next octet expected by the destination TCP. This sequence number implies that

all octets with sequence numbers lower than the acknowledgment value have been delivered to and acknowledged by destination THP. The ACK segment (with or without user text) will be sent to the source TCP under five conditions:

1. A Receive Return event has been received from the destination THP which causes the value of the next octet expected to increase by the letter size. This ACK segment will acknowledge for received/delivered octets.

2. Immediately, if a data segment has been received from source TCP, the sequence numbers (for all octets) of which do not lie within the acceptable sequence number range. This ACK segment is sent to iterate the destination TCP's receive window and the value of the previously ackknowledged sequence number.

3. Immediately, if a valid SYN segment is received and processed. This ACK segment will complete the three-way handshake (see Paragraph 2.2.2).

4. Immediately, if a valid flush request segment is received and processed (see Paragraph 2.3.3).

5. As soon as all octets with sequence numbers lower than a received FIN segment have been accounted for. This ACK segment completes the FIN exchange sequence of connection termination (see Paragraph 2.4).

4. control handling

Control information, carried in the T-segment header, is processed according to the current state reflected in the TCB. Specifics of these relationships can be found in Appendix D. It should be iterated, however, that some TCP-TCP control information is not acknowledged. Only SYN, FIN, flush and wOPEN control segments are subject to accountability. As discussed in Paragraph 2.3.2, the nature of the unaccountable information eliminates the need for retransmission or accounting of some TCP-TCP control only segments. If the unaccountable control information is carried in a data segment, however, the segment will be subject to the accountability of the data segment.

2.4 TERMINATING A CONNECTION

From TCP's viewpoint a connection can be closed either locally, by TCP itself or at THP's request (Close event), or remotely, by the remote THP or TCP. If the close is remote, TCP will not be aware of the reason for the close unless it was due to a reported error. In that case a TCP-TCP error message segment would be received. Exact processing for all closes can be traced using the state tables in Appendix D. The relationship between each close type and each possible connection state are many. There is a general procedure, however, which can be summarized here.

1. Initial stimulus - The initial stimulus for terminating a connection may be a THP close request, a TCP detected error or timeout, or TCP preemption of the connection. Each case is different in that the close processing may be deferred, waiting until delivery of user's data, or immediate.

   If the close processing is to be gradual (deferred) and the remote TCP is actively sending data segments, TCP will notify the remote TCP that close processing has begun. The status segment is used for this purpose. This notification informs the remote TCP that no more data should be sent on the connection. The local TCP will continue sending his user's data (that preceded the close request) until all data has been delivered to the destination THP. The closing TCP will not send the status segment unless a data segment is received from the remote TCP after close processing has begun.

   If the close processing is to be immediate, or once the deferred delivery is complete, as discussed above, TCP will begin the next step, the connection termination "handshake." The handshake is not accomplished if the termination is due to TCP error or preemption (see Paragraph 2.4).

2. FIN sequence - The FIN sequence is similar to the SYN segment exchange of the threeway handshake for establishing a connection. It consists of each TCP sending a FIN segment, receiving a FIN segment from the other TCP, acknowledging the FIN segment that was received (ACK segment) and receiving the acknowledgement for the FIN segment that was sent. These acknowledgments will, coincidently, acknowledge all octets received, as the FIN segment sequence number will be "higher" than sequence numbers for previous data segments sent.

These general procedures apply to most closes. Specific in-

formation on the different types of connection closures is given in subsequent paragraphs.

2.4.1 Local Graceful Close

The local graceful close type is also referred to as a deferred or gradual close. The stimulus is a request by the user to close the connection. In the MCCU this request is entered via the THP close command and conveyed to TCP via the Close event with the deferred indicator set. It is assumed that when the user enters the close command he wants all data previously entered, which may still be in the network, to be delivered to the destination user. He does not, however, want to receive any more data on the connection. Considering this and if data is actively being sent to the closing user (see Paragraph 2.4) by the remote TCP, the closing TCP will notify the remote TCP, via the status segment, that close processing is beginning. The remote TCP will then be in the remote close received state. Once all previously sent data (octets) have been acknowledged, the closing TCP will begin the FIN sequence, described in Paragraph 2.4.

2.4.2 Local Immediate Close

The local immediate close type is also referred to as an abort. One stimulus for this type of a close is a request by the user that the connection be aborted. In the MCCU this request is entered via the THP abort command and conveyed to TCP via the Close event with the flush indicator set. It is assumed that the user does not want any more data to be delivered in either direction on the connection. Another stimulus is a second Close event for the connection, regardless of the deferred/flush indicator. It is assumed that a subsequent close request (Close event) was generated under conditions which require immediate action for closure. A third stimulus for the immediate close is a local THP detected protocol error (see Reference 9), which is conveyed to TCP in the same way as a user abort request. Still another stimulus for the immediate close is a TCP detected error or timeout condition.

In the first three cases, the closing TCP will begin the FIN sequence, described in Paragraph 2.4, as soon as the stimulus is received. All data currently "in" TCP, i.e., in the reassembly queue or waiting to be segmentized, will be flushed. TCP detected errors do not result in the FIN ex-

change. Only the error message segment is sent. The closing TCP then considers the connection to be closed. A TCP receiving an error message segment performs certain verification of the error message segment (see Paragraph 2.1.5) and, if the segment is reliable, considers the connection to be terminated and notifies the local THP.

2.4.3 Connection Preemption

A connection is subject to preemption either by a "higher" precedence request for connection or because of critical buffer space. The subject of this discussion will primarily be for connection preemption. Resource preemption is subject to implementation requirements. In the MCCU, connections will be preempted for resource depletion only if there are no alternative solutions. TCP will always preempt lower category (IV) connections first, preempting categories II and III only if the situation dictates. Category I connections will not be preempted for any reason.

As discussed in Paragraph 2.3.4, an attempt is made to match a received segment with a local subscriber and a corresponding connection, if possible. If a SYN segment is received that is not associated with an established connection or an outstanding listen, TCP will try to find an already established connection for the addressee of the SYN segment. If a match is found that connection is a candidate for preemption. The following criteria must be satisfied completely, however, for the preemption to occur.

1. the segment must be a valid SYN segment for the user, that is, the security, precedence, and TCC must be authorized for use by the local subscriber/port ID

2. the subscriber/port ID must be in an established connection, i.e., the connection can not be in any state other than established

3. the foreign subscriber/port ID in the SYN segment (sender of the segment) must be different than the one in the fully specified address of the established connection

4. the connection must be marked preemptable; this indicator is carried in the open request (Open event in MCCU) and saved by TCP for such contingency checks

5. the SYN segment precedence must be in a higher ca-

tegory (I, II, III, IV) then both the send and receive the precedence of the established connection (see Paragraph 2.2.2)

6. neither precedence level of the established connection may be in category I

If a decision is made to preempt the established connection, TCP will send an error message segment ("connection preempted") to the remote TCP and complete establishment of the new connection. The following steps are taken in an MCCU regarding preemption processing:

1. evaluate preemptability, as described above, determining that preemption is possible

2. send "connection preempted" message segment to remote TCP on the preempted (old) connection

3. perform close processing on the old connection, including sending a Close Return event with connection preemption reason code to THP

4. create new TCB for new connection

5. process SYN segment for new connection and continue three-way handshake for new connection

6. once three-way handshake is complete, send Preempt event to THP, indicating that "open complete" processing is required for new connection

7. calculate and relay (via Send Window event) the THP-TCP send window for the new connection

INDEX

Index

APPENDIX A

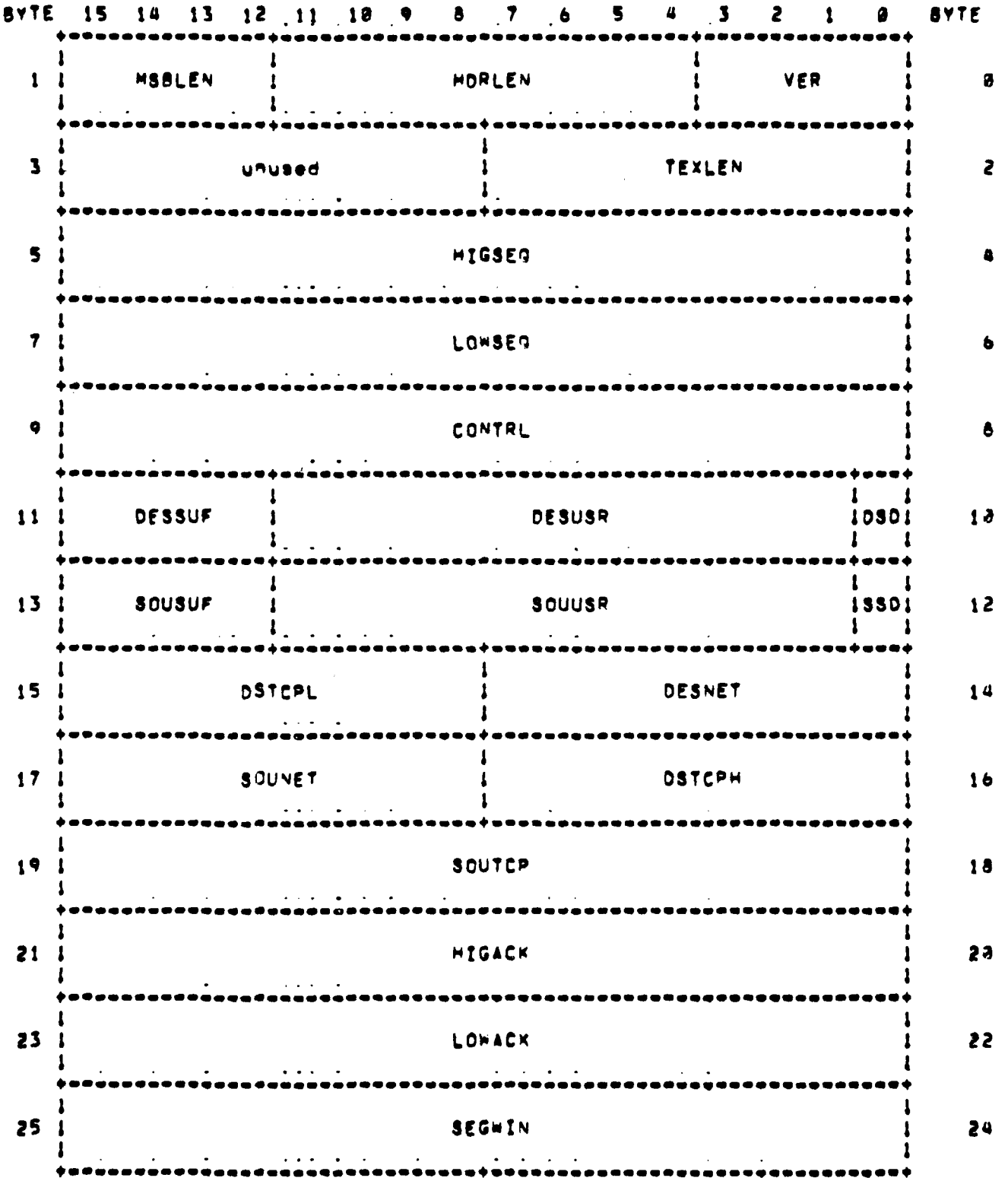
T-SEGMENT HEADER

This appendix describes the TCP-TCP T-segment header format to be used by a TCP desiring to interface with CCU/TAC TCPs on the AUTODIN II network. There is no variation to the format as described herein.

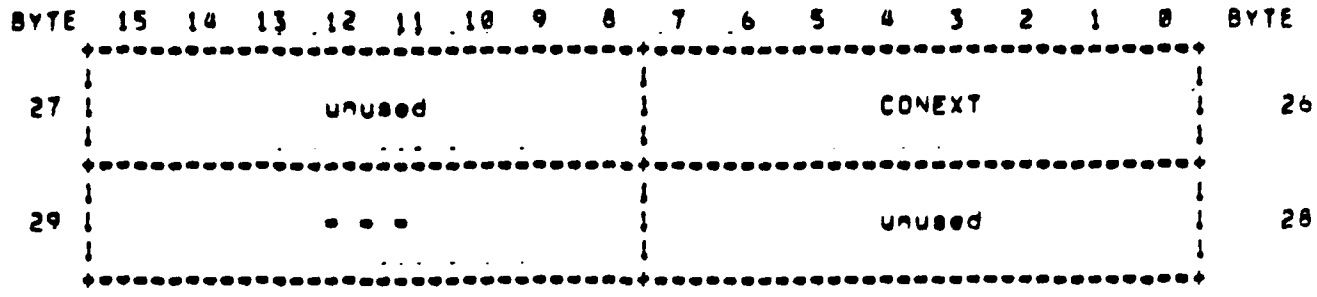
The memory image shown in this appendix shows 30 bytes (0 thru 29) for the T-segment header. The last byte (29), represented by dashes (- - -), is not part of the T-segment header. The T-segment header built by TCP for each segment sent to the network is 29 bytes long. Each field in the 29-byte T-segment header is defined in the documentation following the memory image. The fields that are described as "unused" are always zero.

The format is shown in standard DEC PDP-11 memory image format, with the least significant byte on the right (even number) and the most significant byte on the left (odd number) of a 16-bit word. The bits are labeled 0 to 15 (least significant to most significant) right to left in the word. The direction of transmission is from the least significant byte/least significant bit (0) to the most significant byte/most significant bit (15) for each word.

T-SEGMENT HEADER
26-FEB-79



T-SEGMENT HEADER
26-FEB-79



T-SEGMENT HEADER
30-JUN-79

MNEMONIC	# OF BITS	DESCRIPTION
VER	4	TCP Version Number, Value representing the version of TCP being used by the sender of the segment.
HDRLEN	8	Header Length, Value specifying the length of the T-segment header for this segment. The value is currently 29 bytes.
MSBLEN	4	Text Length (MSB), Most significant 4 bits of the value representing the length (in bytes) of the text carried in this segment.
TEXLEN	8	Text Length (LSB), Least significant 8 bits of the value representing the length in bytes) of the text carried in this segment.
unused	8	This 8-bit field is not used.
HIGSEQ	16	Segment Sequence Number (MSB), Most significant 15 bits (bit 15 is not used) of sequence number for this segment. If status segment, most significant 15 bits of sequence number of last consecutive octet received by the sender of this segment.
LOWSEQ	16	Segment Sequence Number (LSB), Least significant 16 bits of sequence number of this segment. If status segment, least significant 16 bits of sequence number of last consecutive octet received by the sender of this segment.
CONTRL	16	Control Information, TCP-TCP control indicators: bits 0-2 = specific TCP function; bits 3-5 are unused; bit 6 = WACK; bit 7 = WOPEN; bits 8-9 are unused; bit 10 = flush; bit 11 = end-of-letter; bit 12 is unused; bit 13 = FIN; bit 14 = ACK; bit 15 = SYN, See also T-Segment Header Control Field Expansion.
DSD	1	Static/Dynamic Indicator, Bit specifying static or dynamic destination port ID for destination user.
DESUSR	11	Destination User ID, Value representing

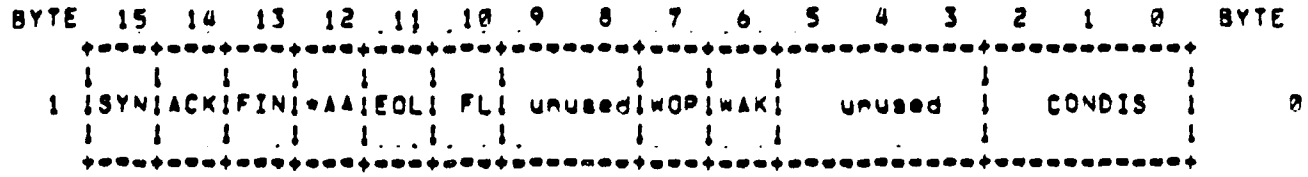
T-SEGMENT HEADER
30-JUN-79

MNEMONIC	# OF BITS	DESCRIPTION
		user ID portion of destination port ID.
DESSUF	4	Destination Function Suffix. Value representing suffix portion of destination port ID.
SSD	1	Static/Dynamic Indicator. Bit specifying static or dynamic source port ID for source user.
SOUUSR	11	Source User ID. Value representing user ID portion of source port ID.
SOUSUF	4	Source Function Suffix. Value representing function suffix portion of source port ID.
DESNET	8	Destination Network. Value representing the destination network of destination user; currently set to zero.
DSTCPL	8	Destination Subscriber Address (LSB). Least significant 8 bits of the destination subscriber address.
DSTCPH	8	Destination Subscriber Address (MSB). Most significant 8 bits of the destination subscriber address.
SOUNET	8	Source Network. Value representing the source network of source user; currently set to zero.
SOUTCP	16	Source Subscriber Address. Value representing network address for source subscriber.
HIGACK	16	Acknowledgment Field (MSB). Most significant 15 bits (bit 15 is not used) of sequence number of next octet expected by sender of this segment. If status segment, most significant 15 bits of last octet delivered to THP by sender of this segment.
LOWACK	16	Acknowledgement Field (LSB). Least significant 16 bits of sequence number of next octet expected by sender of this segment. If status segment, least significant 16 bits of last octet delivered to THP by sender of this segment.

T-SEGMENT HEADER
30-JUN-79

MNEMONIC	# OF BITS	DESCRIPTION
SEGWIN	16	Receive window, Value of destination TCP's (sender of this segment) receive window, expressed in number of octets which destination TCP is willing to accept from source TCP.
CONEXT	8	Control Data Extension, Field containing various information to be used by TCP, See Control Data Extension expansion.
unused	8	This field is reserved for the most significant 8 bits of the TCP checksum, however, CCU/TAC/NCC TCPs do not use the field.
unused	8	This field is reserved for the lease significant 8 bits of the TCP checksum, however, CCU/TAC/NCC TCPs do not use the field.

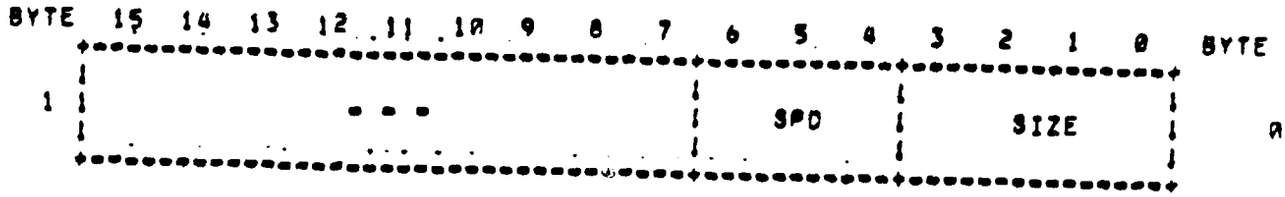
T-SEGMENT HEADER CONTROL FIELD (CONTRL)
26-FEB-79



T-SEGMENT HEADER CONTROL FIELD (CONTRL)
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
CONDIS	3	TCP Function Indicator. Three-bit code for function carried by this segment; 0 = no TCP function; 1 = control extension field (CONEXT) contains error code for segment received; 3 = out-of-band interrupt (OBI) required; 4 = status control; 2, 5, 6, and 7 are not defined.
unused	3	This 3-bit field is not used.
WAK	1	WACK Indicator. When sent, indicates the segment is an acknowledgement for a WOPEN segment.
WOP	1	WOPEN Indicator. When set, indicates the segment contains WOPEN control information; new TCP receive window.
unused	2	This 2-bit field is not used.
FL	1	Flush Indicator. When set, indicates the segment is requesting that the receiving TCP perform a TCP flush function.
EOL	1	End-of-Letter Indicator. When set, indicates the segment contains the last byte in a THP letter.
*AA = unused	1	This 1-bit field is not used.
FIN	1	FIN Indicator. When set, indicates the segment contains final acknowledgement information and that the sender of the segment will send no additional data or control on the connection and will deliver no additional data to the local THP.
ACK	1	ACK Indicator. When set, indicates the acknowledgement field (HIGACK and LOWACK) contains an acknowledgement for octets received by destination TCP (sender of this segment).
SYN	1	SYN Indicator. When set, indicates that the segment contains SYN control information.

T=SEGMENT HEADER CONTROL DATA EXTENSION
26-FEB-79



T-SEGMENT HEADER CONTROL DATA EXTENSION
30-JUN-79

MNEMONIC	# OF BITS	DESCRIPTION
SIZE	4	Size/Error. If error message (CONDIS=1), contains code for error detected; (octal values) 1=connection does not exist; 2=bad SYN control; 3=half open; 10=security error; 11=BSL subscriber address error; 12=TCC error; 13=T-segment header subscriber address error. If SYN control indicated, contains approximate number of grains (grain = 64 bytes) to be sent in a THP letter.
SPD	3	Input Line Speed. 3-bit code for the speed of the line (in bps) between source user and source CCU/TAC: 0 = 150 or less; 1 = 300; 2 = 600; 3 = 1200; 4 = 2400; 5 = 4800; 6 = 9600; 7 = 19200 or more. Field is used only if SYN control is indicated.

APPENDIX B

TCP MCCU DATA STRUCTURE

This appendix describes the major TCP data structure of the MCCU: the Transmission Control Block (TCB). This structure contains all the information required to perform TCP connection processing for an MCCU.

The exact format for the TCB is unique to the MCCU, however, similar information concerning the user's connection must be maintained by any TCP.

The format is in standard DEC PDP-11 memory image format, with the least significant byte on the right (even number) and the most significant byte on the left (odd number) of a 16-bit word. The bits are labeled 0 to 15 (least significant to most significant) right to left in the word.

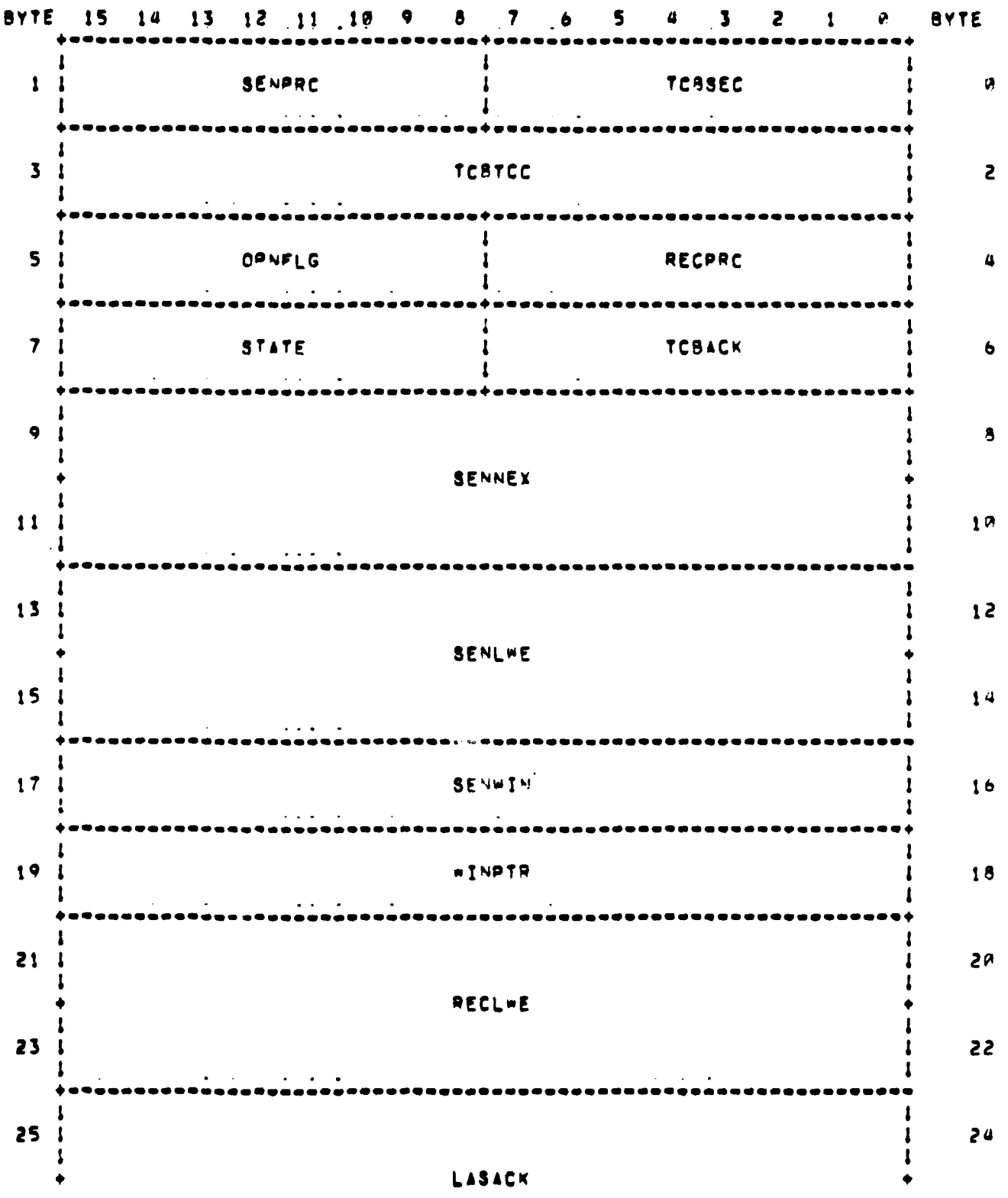
The memory image for each structure shown in this appendix is described by an accompanying narrative. The size (byte, word, etc.) of each unit within the structure may cause "extra" bytes within or at the end of the structure. These extra bytes are not used and may be marked as such or not defined in the narrative. They will be represented by a series of dashes (- - -) in the memory image and are always zero. If the extra byte(s) is within the structure, it is, of course, counted in the length of the structure. If the extra byte(s) is at the end of the structure, it will not be counted in the length of the structure.

TCP MCCU DATA STRUCTURE
Transmission Control Block (TCB)

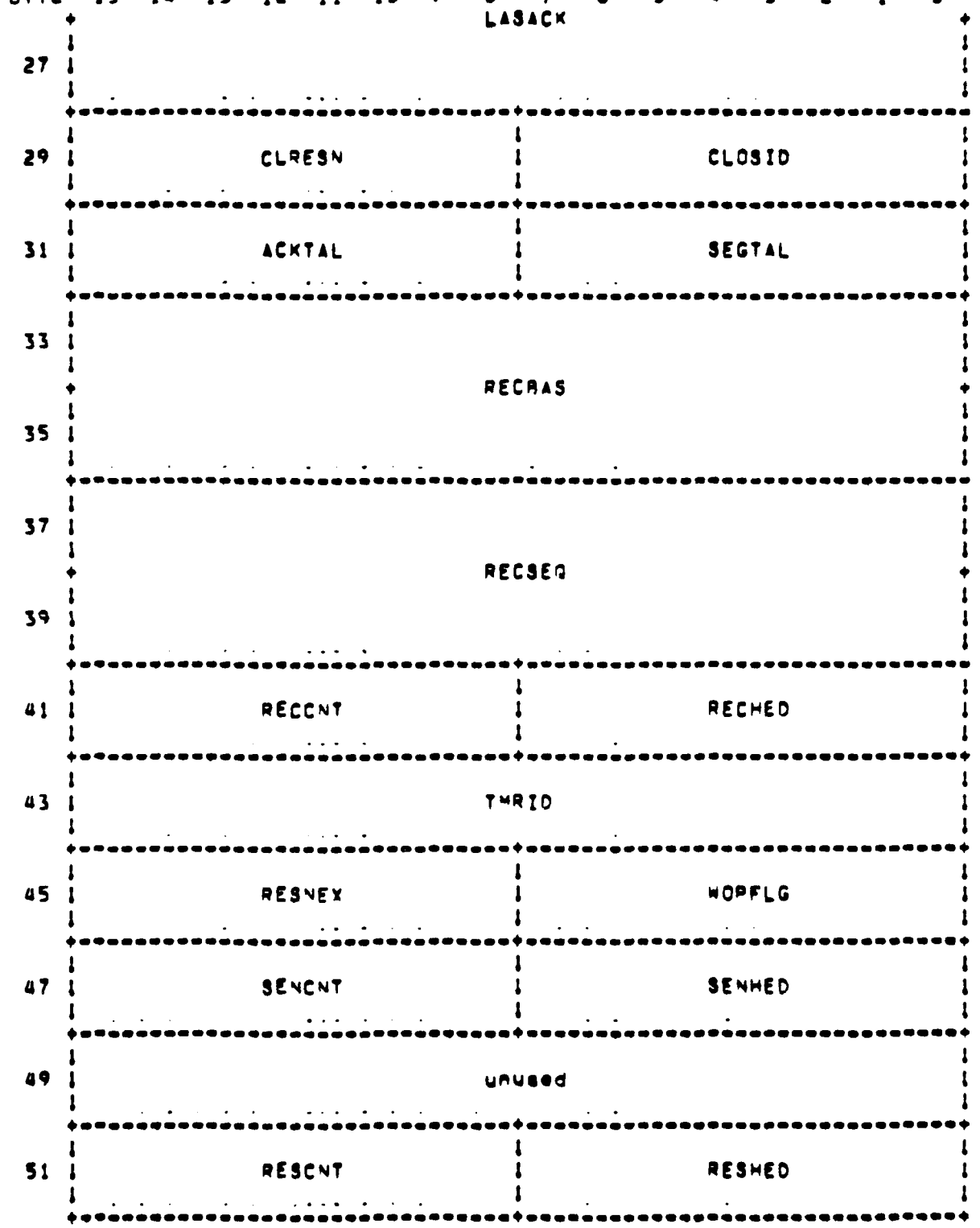
TRANSMISSION CONTROL BLOCK (TCB)

The TCB contains most of the information required by TCP to perform connection processing. There is one TCB per connection. The TCB is created in the MCCU when a valid Open event is received from THP, and deleted once connection closure is completed. There are other minor data structures used by TCP, however, these are implementation-oriented and are not discussed in this appendix.

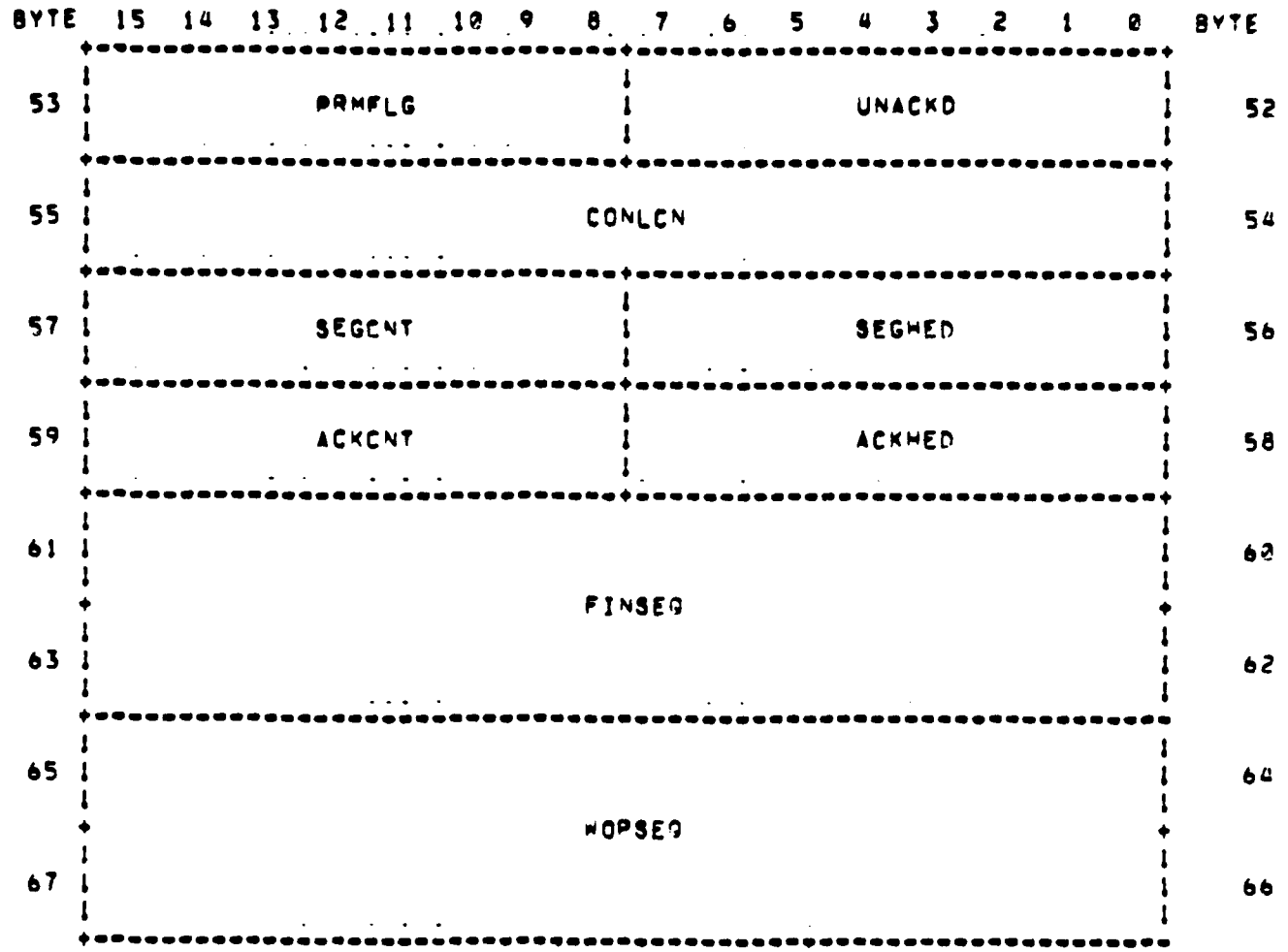
TRANSMISSION CONTROL BLOCK
26-FEB-79



TRANSMISSION CONTROL BLOCK
26-FEB-79



TRANSMISSION CONTROL BLOCK
26-FEB-79



TRANSMISSION CONTROL BLOCK
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
TCBSEC	8	Security. Security level specified in Open event; if none specified in Open event, contains maximum security level authorized for use by this user.
SENPRC	8	Send Precedence. Precedence level specified in Open event; if none specified in Open event, contains maximum precedence level authorized for use by this user.
TCBTCC	16	TCC. Binary value for TCC specified in Open event; if none specified in Open event, contains zero.
RECPRC	8	Receive Precedence. Precedence level of the "network-to-user" side of the connection, to be used by the remote TCP in sending data to local user; value is used in connection preemption processing.
OPNFLG	8	Open Flag. Flag from Open event, indicating which open request parameters were not specified by user (set if parameter is unspecified) bit 0 = security; bit 1 = send precedence; bit 2 = TCC; bit 3 = destination address.
TCBACK	8	TCB Acknowledgement Flag. Field used to govern transmission of ACK, status, OBI, WOPEN, and WACK control segments: status = -1; ACK = 1; WOPEN = 2; WACK = 3; OBI = 4.
STATE	8	Connection State. Field designating current state of the connection; 0 = listen; 1 = open; 2 = SYN sent; 4 = SYN sent/received; 8 = established; 16 = local close received; 18 = FIN sent; 1A = FIN sent/received; 22 = FIN received; 24 = remote close received; 32 = closed.
SENNEX	32	Next Sequence Number. Value to be assigned to next segment sent.
SENLWE	32	Send Left Window Edge. Value of sequence number for "oldest" unacknowledged octet.

TRANSMISSION CONTROL BLOCK
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
SENWIN	16	Send Window, Number of octets which may be sent to remote TCP.
WINPTR	16	Window Pointer, Pointer to window array entry for connection; contains information for each connection such as current receive window, maximum receive window ever issued, input and output line speeds for connection, and input and output approximate letter sizes.
RECLWE	32	Receive Left Window Edge, Value of next sequence number expected from remote TCP.
LASACK	32	Last Acknowledged, Value of sequence number of last octet successfully delivered to THP; used in TCP close accountability processing.
CLOSID	8	Close Event ID, Transaction ID of THP Close event or zero if none received.
CLRESN	8	Close Reason, Code for close reason: 0 = local user issued a close and no errors occurred; non-zero = error condition is causing close.
SEGTAL	8	Retransmission Tail, Pointer to next available entry in retransmission queue.
ACKTAL	8	Acknowledgement Tail, Pointer to next available entry in acknowledgement queue.
RECBAS	32	Receive Base, Value of sequence number of first data byte received on connection from remote TCP.
RECSEQ	32	Receive Left Edge, Value, relative to sequence number of Receive Base, of receive left edge.
RECHED	8	Receive Queue, Pointer to head of receive queue.
RECCNT	8	Receive Count, Number of elements on receive queue.
TMRID	16	Timer ID, Timer identification of Retry

TRANSMISSION CONTROL BLOCK
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
		event queued because of elements on retransmission queue or WOPEN control outstanding.
WOPFLG	8	WOPEN Flag. Flag indicating number of transmissions have been made of WOPEN control which have not been acknowledged by WACK control segment.
RESNEX	8	Reassembly Pointer. Pointer to next available free space in reassembly queue.
SENHED	8	Send Queue Head. Pointer to head of send queue.
SENCNT	8	Send Queue Count. Number of elements on send queue.
unused	16	This 16-bit field is not used.
RESHED	8	Reassembly Head. Pointer to head of reassembly queue.
RESCNT	8	Reassembly Count. Number of elements on reassembly queue.
UNACKD	8	Retransmission Acknowledgement Pointer. Pointer to first element in retransmission queue which has not been acknowledged.
PRMFLG	8	Preemption Flag. Preemption flag from Open event, indicating whether the connection is available for preemption (value is one) or not (value is zero).
CONLCN	16	Preempted LCN. Contains Local Connection Name for preempted connection.
SEGHED	8	Retransmission Head. Pointer to head of retransmission queue.
SEGCNT	8	Retransmission Count. Number of elements on retransmission queue.
ACKHED	8	ACK Queue Head. Pointer to head of ACK queue.
ACKCNT	8	ACK Count. Number of elements on ACK queue.

TRANSMISSION CONTROL BLOCK
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
FINSEQ	32	Received FIN Sequence Number. Value of sequence number assigned to FIN segment received from remote TCP; value is negative if FIN segment has not been received as yet.
WOPSEQ	32	WOPEN Sequence Number. Value of sequence number of outstanding WOPEN segment sent to remote TCP.

APPENDIX C

TCP EVENTS FOR MCCJ

This appendix describes the events "received" and "sent" by TCP in the MCCU. For the network-to-user processing path, TCP receives events from SIP and sends events to THP. For the user-to-network processing path, TCP receives events from THP and queues events to SIP.

These events are implementation-oriented and, in fact, represent one MCCU implementation. This appendix is not intended to provide documentation for the MCCU events; that is provided under seperate cover. It is, rather, intended to suggest the type of information that must be passed between a TCP and THP or SIP in an implementation similar to the MCCJ. In the MCCJ the network protocol processing function of the SIP is handled by the Network Interface Protocol (NIP). Events between the TCP function and this function in the MCCJ are shown with the SENDER or RECEIVER of the event being NIP, as appropriate.

The format is shown in standard DEC PDP-11 memory image format, with the least significant byte on the right (even number) and the most significant byte on the left (odd number) of a 16-bit word. The bits are labeled 0 to 15 (least significant to most significant) right to left in the word.

The memory image for each structure shown in this appendix is described by an accompanying narrative. The size (byte, word, etc.) of each unit within the structure may cause "extra" bytes within or at the end of the structure. These extra bytes are not used and may be marked as such or not defined in the narrative. They will be represented by a series of dashes (- - -) in the memory image and are always zero. If the extra byte(s) is within the structure, it is, of course, counted in the length of the structure. If the extra byte(s) is at the end of the structure, it will not be counted in the length of the structure.

EVENT SPECIFICATION

EVENT NAME: CLOSE EVENT
MNEMONIC: OCLOSE

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event allows THP to request that TCP close a virtual connection.

REASON:

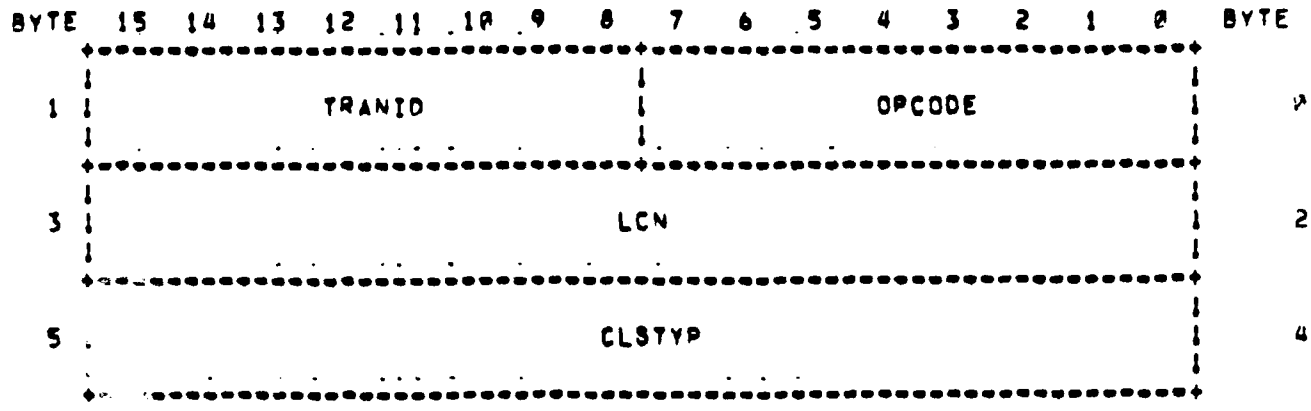
THP issues a Close event in response to a close command from the user or as a result of a protocol error encountered in processing for the referenced connection.

COMMENTS:

If TCP receives a second consecutive Close event for a connection, i.e., a Close event received following another Close event, but before close processing is complete, TCP will send a Close Return event for the first Close event, and treat the second as a "flushing" close request.

VERSION: 29-SEP-78

CLOSE EVENT
29-SEP-78



CLOSE EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Close event = 67.
TRANID	8	Transaction ID. Sequence number for this event, for coordination of return event, with range 1-255.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
CLSTYP	16	Type of Close. Indicates what type of close should be performed by TCP; 0 = deferred, all Send events issued prior to Close event will be delivered to destination, if possible; 1 = immediate, all Send events not yet segmentized will be returned and TCP close processing done immediately; in either case, no further Receive events will be queued to THP.

EVENT SPECIFICATION

EVENT NAME: CLOSE RETURN EVENT
MNEMONIC: OCLSRT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event notifies THP that TCP close processing has been completed for a connection.

REASON:

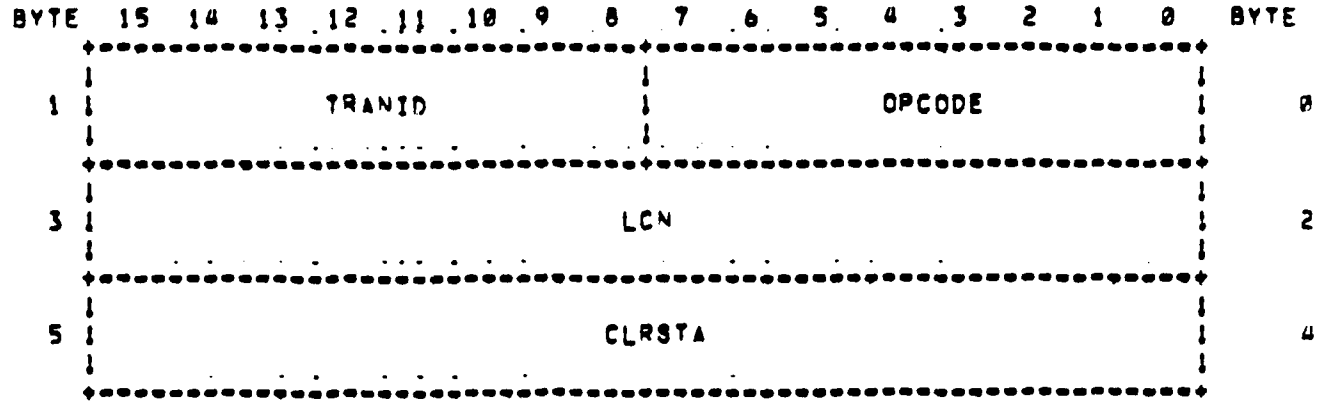
The event will be issued in response to a previous Close event or for several exception conditions encountered by TCP, such as, remote user closing, S/P/T errors, destination subscriber down, etc.

COMMENTS:

If TCP sends an unsolicited Close Return event, i.e., one not caused by previous Close event, the Transaction ID of the return event will be zero.

VERSION: 29-SEP-78

CLOSE RETURN EVENT
29-SEP-78



CLOSE RETURN EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Close Return event # 43.
TRANID	8	Transaction ID, Sequence number for previous Close event, if any, with range 1-255; this byte will be zero if the return event is unsolicited.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.
CLRSTA	16	Close Status, 0=successful local; 1=nonexistent connection; 11=remote close; 16=SCM security error; 17=SCM precedence error; 18=SCM TCC error; 19=SCM address error; 20=destination subscriber down; 21=destination access circuit down; 22=destination busy; 30=remote TCP security error; 31=remote TCP TCC error; 32=remote preempt; 33=space preempt; 34=half open; 35=no ack rcvd; 36=line error

EVENT SPECIFICATION

EVENT NAME: INITIALIZE TCP EVENT
MNEMONIC: OINITT

CPCI: MCCU

SENDER: BEC

RECEIVER: TCP

PURPOSE:

This event informs TCP that the MCCU has been initialized and relays the subscriber address for this MCCU.

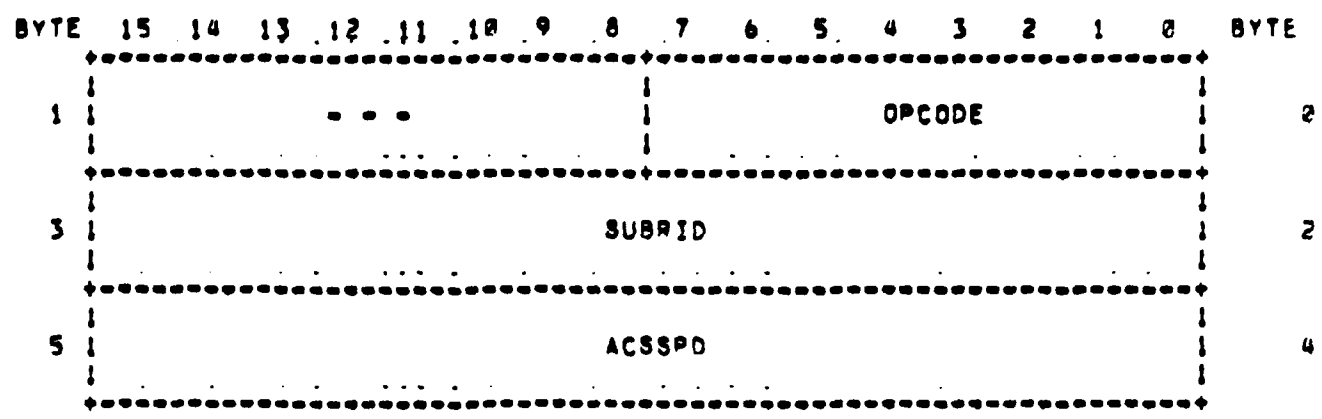
REASON:

TCP must initialize several of its internal tables prior to execution and also must be informed of the local subscriber address for this MCCU in order to send and receive segments.

COMMENTS:

VERSION: 13-SEP-78

INITIALIZE TCP EVENT
13-SEP-78



INITIALIZE TCP EVENT
13-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Initialize TCP event = 73.
SUBRID	16	Local Subscriber ID.
ACSSPD	16	Access Line Speed. Value representing speed of access line between MCCU and SCM.

EVENT SPECIFICATION

EVENT NAME: INTERRUPT EVENT
MNEMONIC: OINTPT

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event requests that TCP perform either an interrupt or flush function for data being sent from the source (originator of Interrupt event) to the destination on the referenced connection.

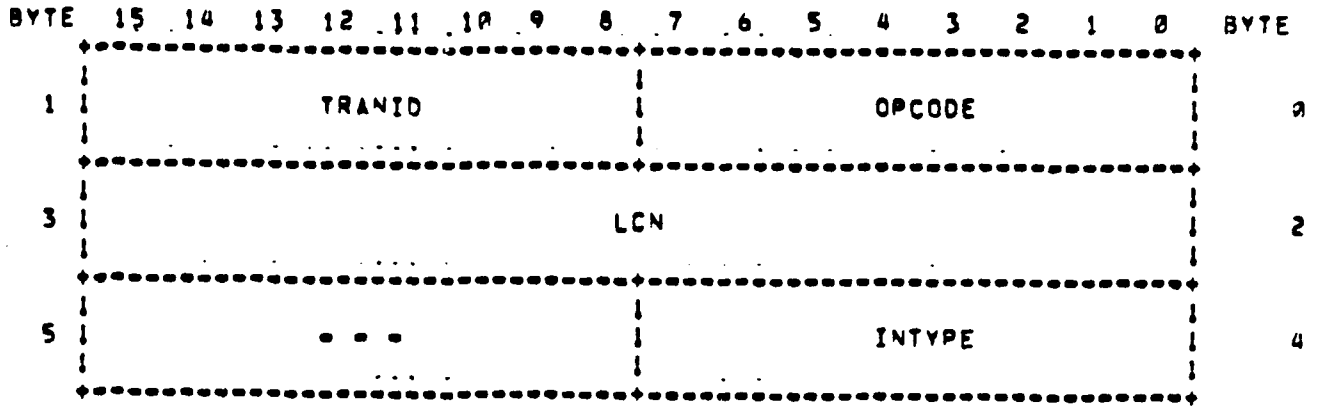
REASON:

The user is given 19 interrupt function "keys" which cause 10 different interrupt functions. Of these one causes an out-of-band TCP interrupt function and several cause TCP flush functions. These TCP interrupt/flush functions are requested via the Interrupt event.

COMMENTS:

VERSION: 29-SEP-78

INTERRUPT EVENT
29-SEP-78



INTERRUPT EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Interrupt event = 69.
TRANID	8	Transaction ID, Sequence number for this event, for coordination of return event, with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.
INTYPE	8	Function Type, Value indicating type of request: 0 = out-of-band interrupt; 1 = flush.

EVENT SPECIFICATION

EVENT NAME: INTERRUPT RETURN EVENT
MNEMONIC: OINTRT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event informs THP that the interrupt/flush function requested in a previous Interrupt event has been completed. If there was not a previous Interrupt event (Transaction ID is zero), this event informs THP that an interrupt/flush function was requested by the remote THP.

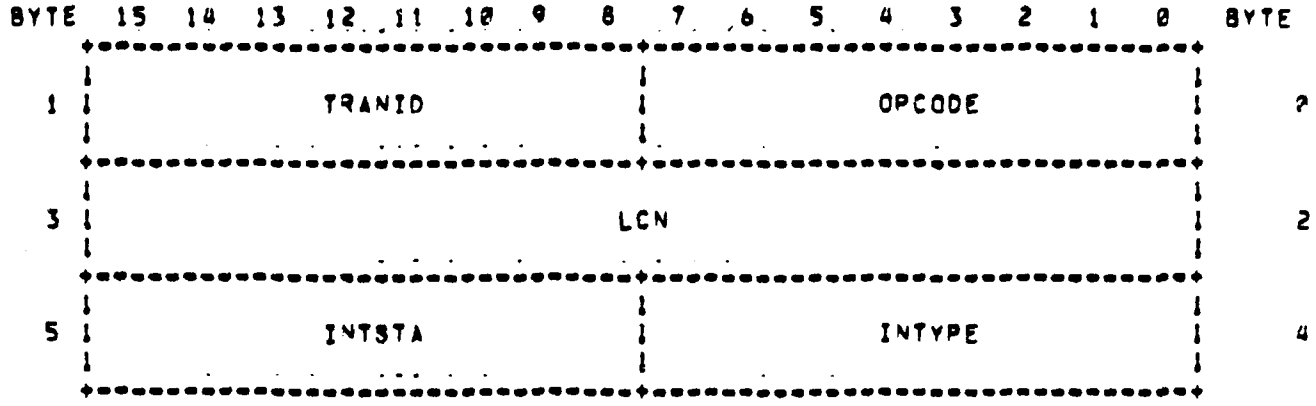
REASON:

Source THP has requested an interrupt/flush and both source and destination THPs must be made aware that the function has been performed so that any THP follow-up processing may be completed.

COMMENTS:

VERSION: 12-OCT-78

INTERRUPT RETURN EVENT
12-OCT-78



INTERRUPT RETURN EVENT
12-OCT-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Interrupt Return event = 39.
TRANID	8	Transaction ID. Sequence number for corresponding Interrupt event with range 1-255; zero if this is unsolicited Interrupt Return event.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
INTYPE	8	Function Type. Value indicating type of function: 0 = out-of-band interrupt; 1 = flush.
INTSTA	8	Return Status. 0 = successful; 1 = connection does not exist; 2 = function not legal for current TCP state; 7 = function not completed (undelivered).

EVENT SPECIFICATION

EVENT NAME: MOVE CONNECTION EVENT
MNEMONIC: OMOVE

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event requests that TCP verify a user as authorized to receive an already established connection.

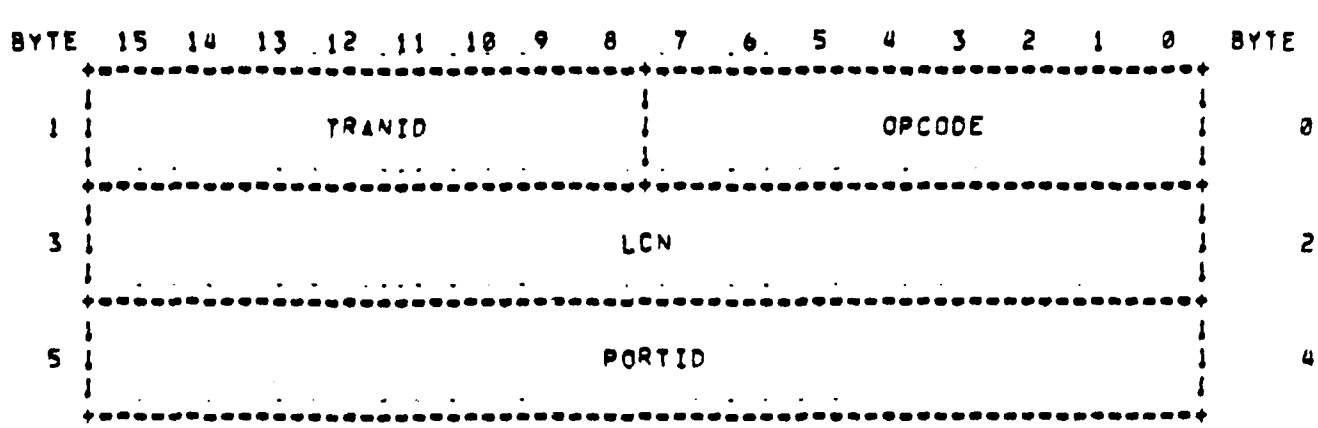
REASON:

THP has received a request on an active connection, via the move command, that the connection be "moved" to another user on this MCCU. THP must ensure that the user is authorized the security, precedence, and TCC of the established connection. To acquire this authorization, THP enlists TCP.

COMMENTS:

VERSION: 29-SEP-78

MOVE CONNECTION EVENT
29-SEP-78



MOVE CONNECTION EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Move Connection event = 74.
TRANID	8	Transaction ID. Sequence number of this event, for coordination with return event, with range 1-255.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
PORTID	16	Local Port ID. Value for local port ID for new users: bits 1-11 = user ID; bits 12-15 = function suffix; bit 0 = static/dynamic port indicator.

EVENT SPECIFICATION

EVENT NAME: MOVE CONNECTION RETURN EVENT
MNEMONIC: OMOVRT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event responds to THP's Move Connection event, indicating whether the new user is authorized to receive the established connection.

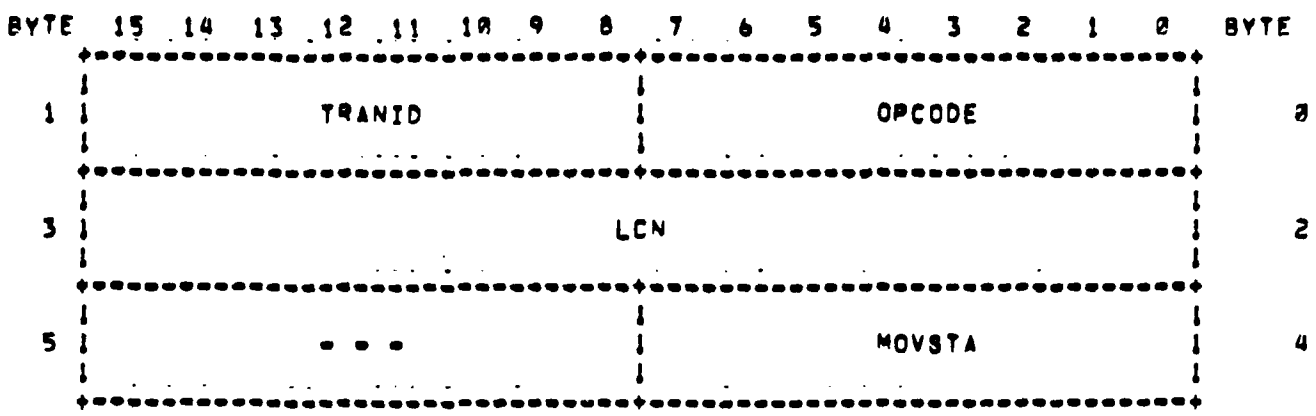
REASON:

TCP received a Move event from THP requesting that a verification be made as to the authorization of a new user to use an established connection.

COMMENTS:

VERSION: 29-SEP-78

MOVE CONNECTION RETURN EVENT
29-SEP-78



MOVE CONNECTION RETURN EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Move Connection Return event = 48.
TRANID	8	Transaction ID. Sequence number of the corresponding Move Connection event with range 1-255.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
MOVSTA	8	Status of Move event processing; 0 = new user is authorized to receive referenced connection; 1 = new user is not authorized to receive referenced connection.

EVENT SPECIFICATION

EVENT NAME: OPEN COMPLETE EVENT
MNEMONIC: OOPCMP

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event notifies THP of the security, precedence, TCC, and foreign address of the connection. These fields may not have been specified in the original open event or in the case of precedence may have been changed by TCP.

REASON:

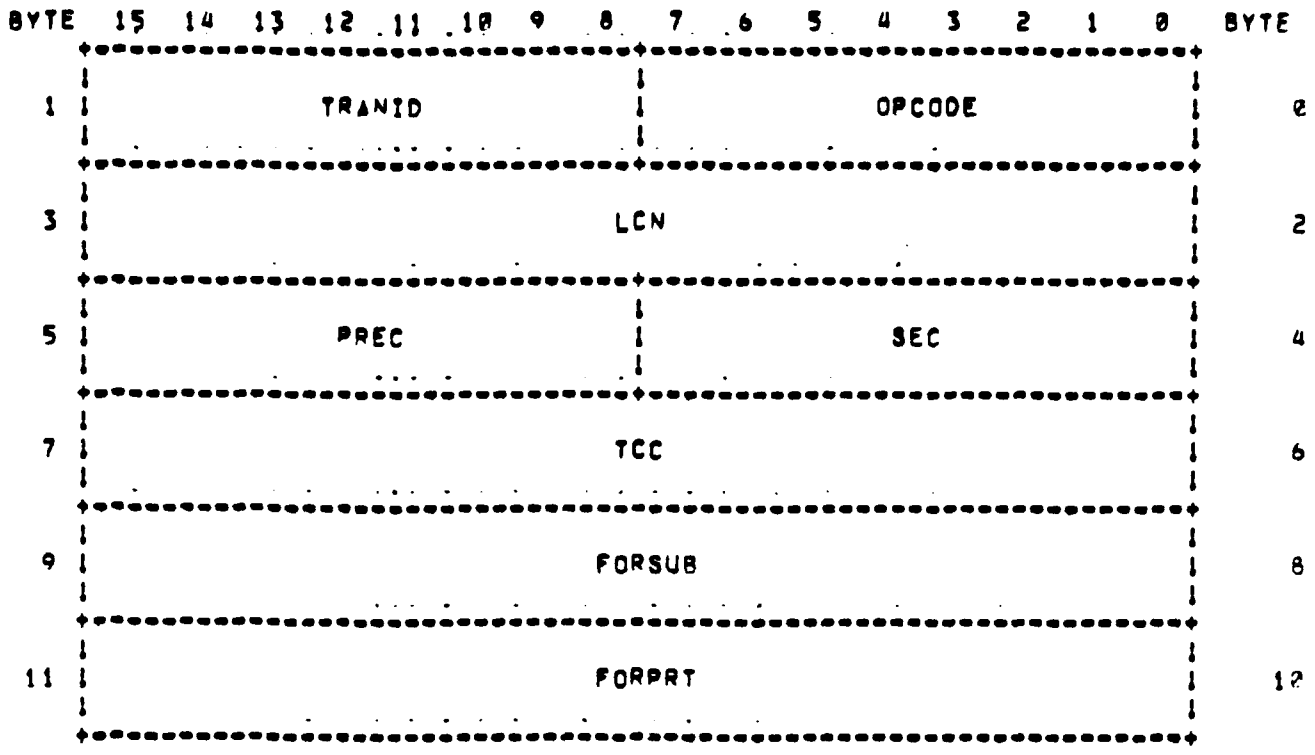
This event is issued after the first 'SYN' segment is received from the remote TCP for the connection. At that time all unspecified parameters of the open request become bound by the values of the incoming segment.

COMMENTS:

The local user must be authorized the higher precedence and all values associated with the new connection that had been previously unspecified in his open request.

VERSION: 26-FEB-79

OPEN COMPLETE EVENT
26-FEB-79



OPEN COMPLETE EVENT
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Open Complete event = 45.
TRANID	8	Transaction ID. Sequence number of the corresponding Open event with range 1-255.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
SEC	8	Connection Security Level. 15 is high; 0 is low.
PREC	8	Connection Precedence Level. 15 is high; 0 is low.
TCC	16	Connection TCC. Binary value with range 0-512.
FORSUB	16	Foreign Subscriber ID.
FORPRT	16	Foreign Port ID.

EVENT SPECIFICATION

EVENT NAME: OPEN EVENT
MNEMONIC: OOPEN

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event allows THP to specify the parameters of a new connection.

REASON:

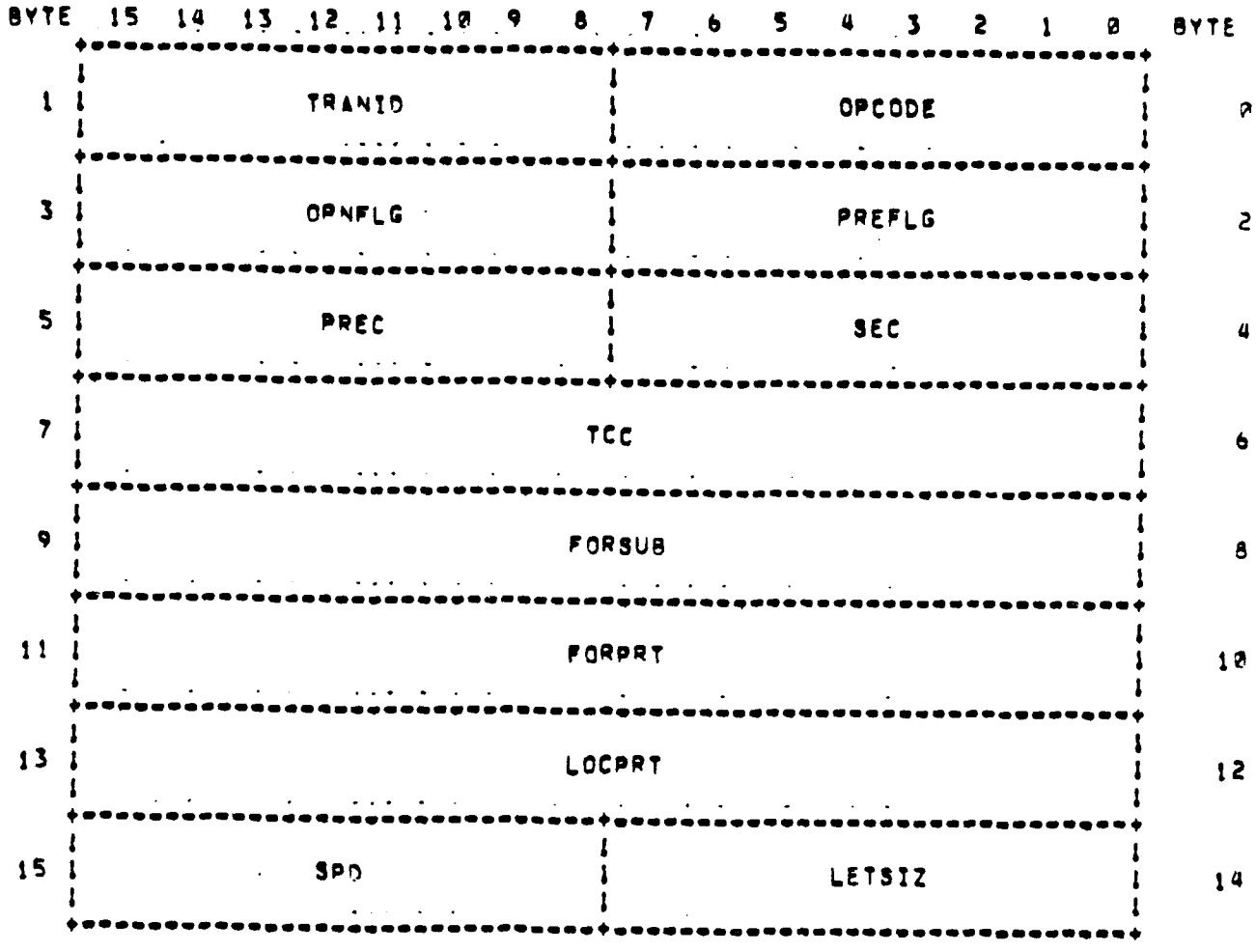
THP builds an Open event when it receives an open or listen command from the user when a connection does not already exist, or when it performs an auto-listen or auto-open function. In the case of a listen request, the destination address, security, precedence, or TCC may be unspecified.

COMMENTS:

The immediate return event (Open Return event) merely signifies that the parameters specified in the Open event were valid for the user. The connection, however, is not established until TCP completes the three-way handshake. This will be caused by a Send event by either the local or remote THP.

VERSION: 26-FEB-79

OPEN EVENT
26-FEB-79



OPEN EVENT
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for an Open event = 64.
TRANID	8	Transaction ID. Sequence number for this event, for coordination of return events, with range 1-255.
PREFLG	8	Preemption Flag. Flag indicating if this connection can be preempted: 0 = preemption not allowed; 1 = connection can be preempted.
OPNFLG	8	Open Flag Byte. Flags indicating the unspecified parameters of the open request. If set, bit 0 = no security; bit 1 = no precedence; bit 2 = no TCC; bit 3 = no foreign subscriber address.
SEC	8	Security. Value for security level for the new connection; 15 is high; 0 is low.
PREC	8	Send Precedence. Value for send precedence level for the new connection; 15 is high; 0 is low.
TCC	16	TCC. Value for TCC of the new connection with range 0-512.
FORSUB	16	Foreign Subscriber ID.
FORPRT	16	Foreign Port ID. Value for foreign port ID for new connection; bits 1-11 = user ID; bits 12-15 = function suffix; bit 0 = static/dynamic port indicator.
LOCPRT	16	Local Port ID. Value for local port ID for new connection; bits 1-11 = user ID; bits 12-15 = function suffix; bit 0 = static/dynamic port indicator.
LETSIZ	8	Letter Size. Approximate number of grains (one grain = 64 bytes) to be sent in a THP letter on this connection.
SPD	8	Line Speed. Code for input line speed (bps) for host-CCU channel for this user: 0 = 150 or less; 1 = 300; 2 = 600; 3 = 1200; 4 = 2400; 5 = 4800; 6 = 9600; 7 = 19200 and more.

EVENT SPECIFICATION

EVENT NAME: OPEN RETURN EVENT
MNEMONIC: OOPNRT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event informs THP of the local connection name (LCN) for the possible connection, if the Open event was valid, or returns an error status, if the Open event was rejected by TCP.

REASON:

THP must be informed if the open request was invalid so that the user can be informed of an error, or, in the case of a successful open request, the characteristics option can begin. This option is performed only if the user open command or auto-open feature caused the open request to TCP.

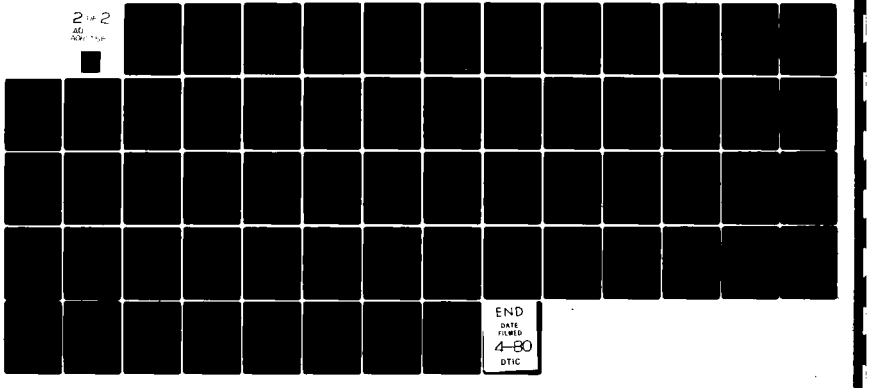
COMMENTS:

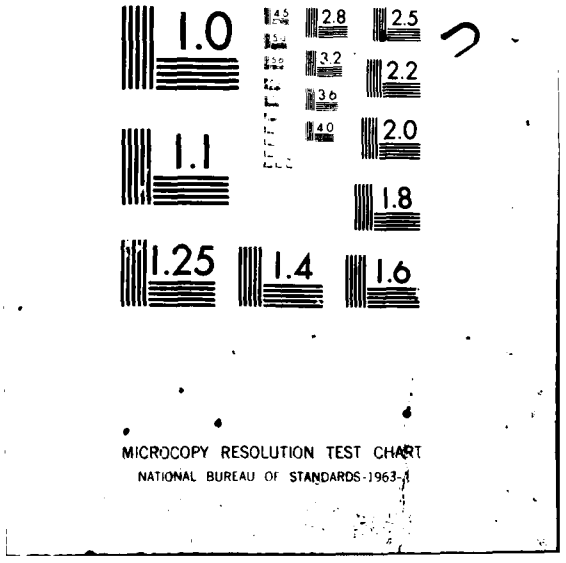
This event does not mean that the connection has been established. In fact, the three-way handshake will not take place until the first Send event from THP. The Send event queued as a result of the Open Return event (characteristics option record) will be the first Send event and, thus, will cause the three-way handshake.

VERSION: 26-FEB-79

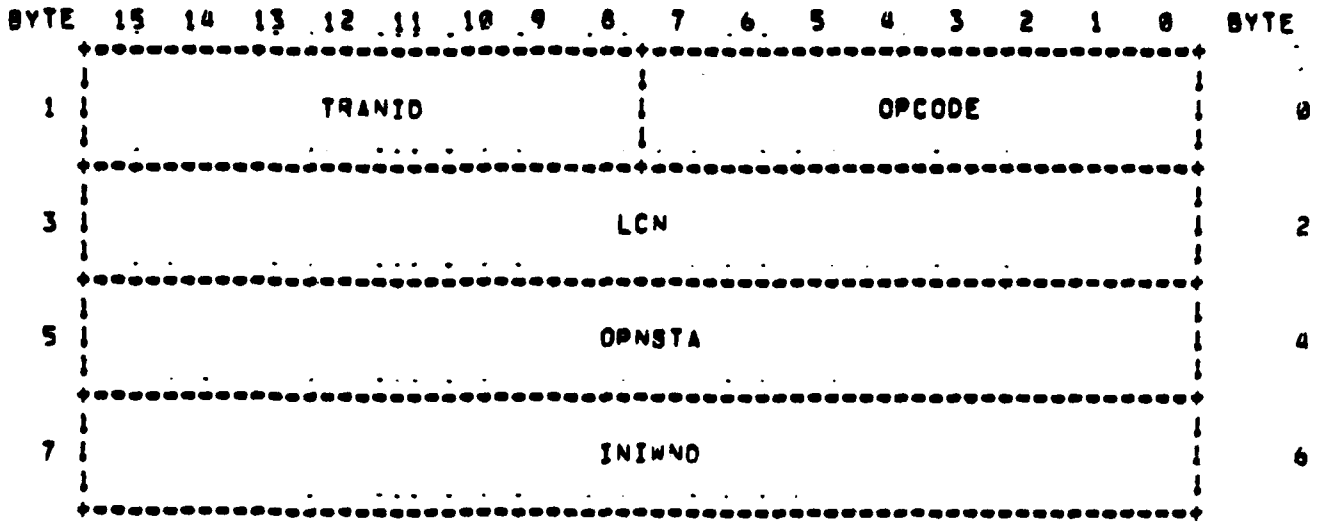
AD-A081 756 WESTERN UNION TELEGRAPH CO MCLEAN VA GOVERNMENT SYST--ETC F/G 17/2
COMPUTER PROGRAM DEVELOPMENT SPECIFICATION TRANSMISSION CONTROL--ETC(U)
FEB 80 DCA200-C-637

UNCLASSIFIED SBIE-AD-E100 342 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

OPEN RETURN EVENT
26-FEB-79



OPEN RETURN EVENT
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Open Return event = 40.
TRANID	8	Transaction ID, Sequence number of the corresponding Open event with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.
OPNSTA	16	Status of Open event processing: 0=no error; 2=connection specified in open event exists; 3=unknown user; 6=insufficient resources for new connection; 8=security error; 9=precedence error; 10=TCC error.
INIWND	16	Initial Send Window, Value which provides the initial THP letter ceiling in number of letters which may be outstanding to TCP (Send Return events not received). Possible values are: 0 = Open event was partially specified (listen request), THP Send events are blocked; 1 = Open event was fully specified (open request), THP may send one letter; initial value will not be more than 1.

EVENT SPECIFICATION

EVENT NAME: PREEMPT EVENT
MNEMONIC: OPRMPT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event notifies THP that a new connection is being established in place of a preempted one, which has been closed. THP will have received an unsolicited Close Return event with preemption as the close status.

REASON:

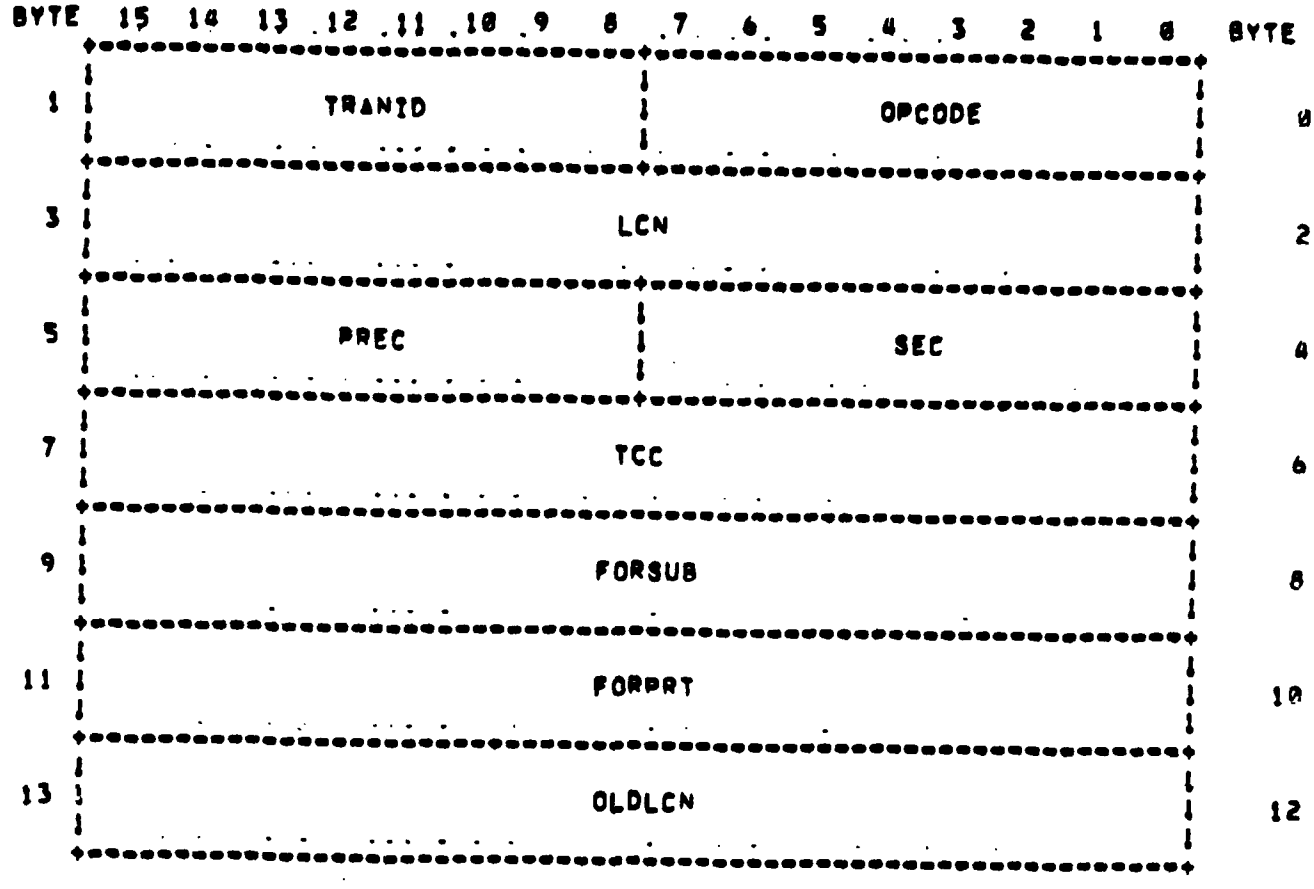
The Preemot event acts as an Open Complete event for THP, i.e., upon receipt of the Preempt event, THP will send the characteristics option record. THP reacts the same way to an Open Complete event when the connection is "listening".

COMMENTS:

THP requires the LCN of the preempted connection for coordination purposes in finishing the close processing for that connection.

VERSION: 08-FEB-79

PREEMPT EVENT
08-FEB-79



PREEMPT EVENT
08-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Preempt event = 47.
TRANID	8	Transaction ID. This byte is always zero for the Preempt event.
LCN	16	Local Connection Name. Internal THP-TCP connection identification with range 1-32. This field contains LCN for new connection.
SEC	8	Connection Security Level. 15 is high; 0 is low.
PREC	8	Connection Precedence Level. 15 is high; 0 is low.
TCC	16	Connection TCC. Binary value with range 0-512.
FORSUB	16	Foreign Subscriber ID.
FORPRT	16	Foreign Port ID.
OLDLCN	16	Old Local Connection Name. LCN of the preempted connection.

EVENT SPECIFICATION

EVENT NAME: RECEIVE DATA EVENT
MNEMONIC: ORCDAT

CPCI: MCCU

SENDER: NIP

RECEIVER: TCP

PURPOSE:

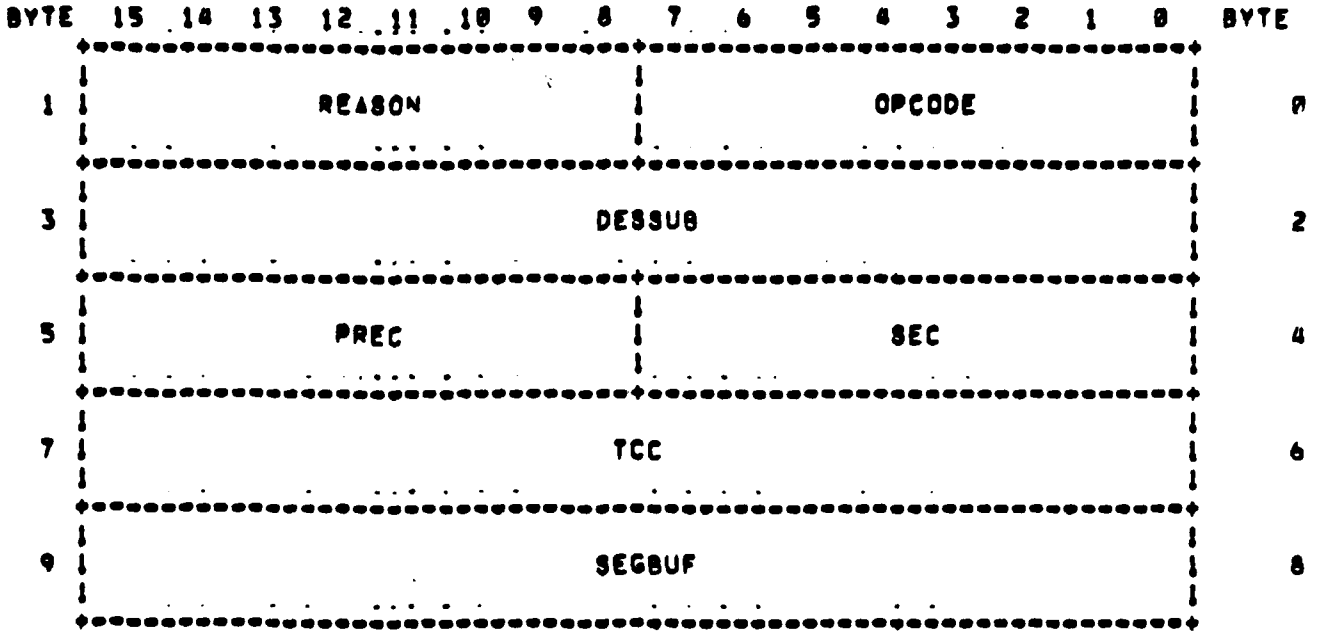
This event notifies TCP that a segment has arrived from the network to be processed.

REASON:

COMMENTS:

VERSION: 01-SEP-78

RECEIVE DATA EVENT
01-SEP-78



RECEIVE DATA EVENT
01-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Receive Data event = 71.
REASON	8	SIP specific field unused by TCP.
DESSUB	16	Destination Subscriber ID.
SEC	8	Security. Value for security level of the incoming segment: 15 is high; 0 is low.
PREC	8	Precedence. Value for precedence level of the incoming segment: 15 is high; 0 is low.
TCC	16	TCC. Binary value for the TCC of the incoming segment with range 0-512.
SEGBUF	16	Buffer ID of the incoming segment.

EVENT SPECIFICATION

EVENT NAME: RECEIVE EVENT
MNEMONIC: ORCEIV

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event passes a letter of data to THP for output to the user on the specified connection.

REASON:

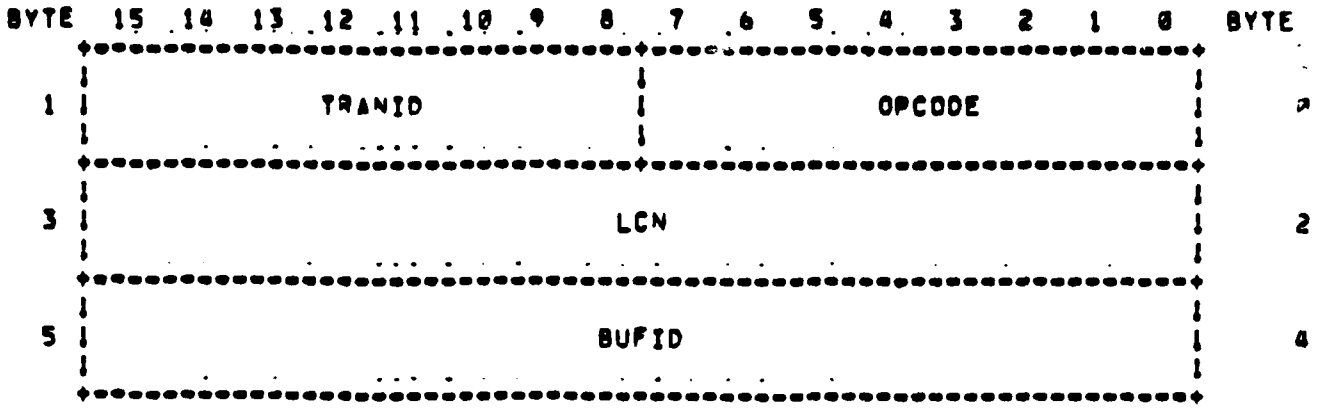
TCP has received data from the network that must be sent to the user.

COMMENTS:

The triplet buffer, passed via this event, contains buffer IDs for each incoming segment that makes up the THP letter. Each group of 3 words represents one segment with the first word in the triplet buffer being the number of segments in the letter and the second word in the triplet buffer being reserved for future use.

VERSION: 29-SEP-78

RECEIVE EVENT
29-SEP-78



RECEIVE EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Receive event = 42.
TRANID	8	Transaction ID. Sequence number for this event, for coordination of return event, with range 1-255.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
BUFID	16	Buffer ID. Tag used to map to the triplet buffer containing information about segments to be processed: word 0 = number of segments; word 1 = reserved for future use; words 2-n, in groups of 3-word entries: first word = segment buffer ID; second word = offset to first data byte; third word = number of bytes of data to be processed in segment.

EVENT SPECIFICATION

EVENT NAME: RECEIVE RETURN EVENT
MNEMONIC: ORCVRT

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event notifies TCP that a letter has been processed for delivery to the user on the connection referenced in the corresponding Receive event.

REASON:

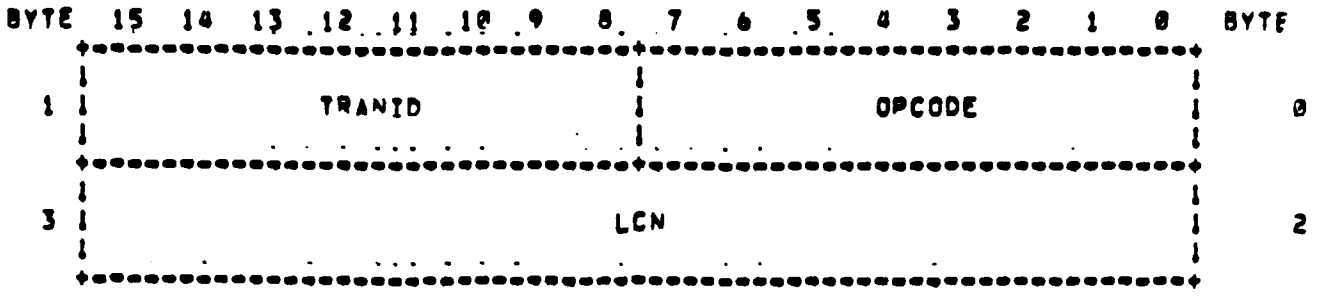
THP has dequeued a Receive event and has processed and moved the data in the THP letter to a "to-user" buffer. The to-user buffer may or may not have been sent to HSI (To User event) for actual output to the user. The triplet and all segment buffers have been processed and released, however.

COMMENTS:

The Receive Return event is used by TCP to determine when an acknowledgement may be sent to remote TCP for data received and delivered to THP.

VERSION: 26-FEB-79

RECEIVE RETURN EVENT
26-FEB-79



RECEIVE RETURN EVENT
26-FEB-79

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Receive Return event = 66.
TRANID	8	Transaction ID, Sequence number of the corresponding Receive event with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.

EVENT SPECIFICATION

EVENT NAME: RETRANSMIT EVENT
MNEMONIC: ORETRY

CPCI: MCCU

SENDER: TCP

RECEIVER: TCP

PURPOSE:

This event causes TCP to scan its retransmission queue for a connection to find segment(s) that should be retransmitted.

REASON:

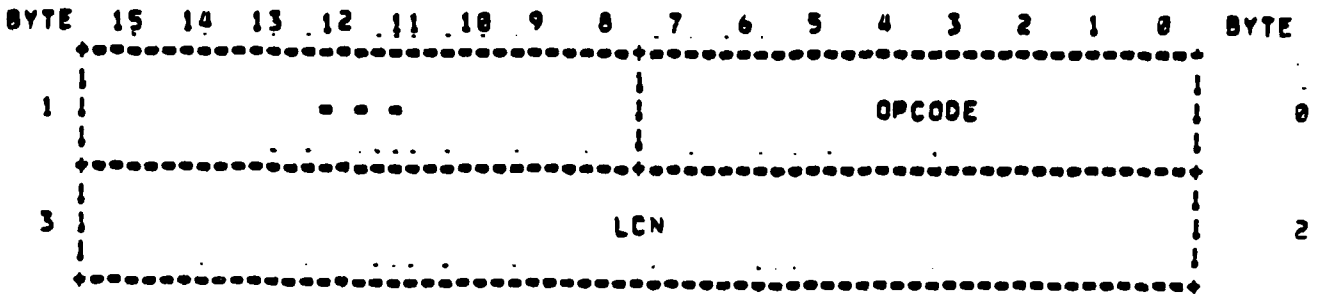
Effectively, TCP queues this event to itself through the BEC timer function (SETMR) which places the event in the TCP dispatch queue after the specified time period.

COMMENTS:

TCP maintains one timer per connection. If the connection has any elements on its retransmission queue, when the timer expires, TCP will decrement the timer count for each segment eligible for retransmission and if any count reaches zero, will retransmit that segment. TCP will reissue the timer event unless the connection closes or the retransmission queue becomes empty.

VERSION: A1-FES-83

RETRANSMIT EVENT
29-SEP-78



RETRANSMIT EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Retransmit event = 70.
LCN	16	Local Connection Name, Internal TCP connection identification value with range 1-32.

EVENT SPECIFICATION

EVENT NAME: SEND DATA RETURN EVENT
MNEMONIC: OSDTRT

CPCI: MCCU

SENDER: NIP

RECEIVER: TCP

PURPOSE:

This event notifies TCP of SIP processing status and network error status in reference to corresponding Send Data event.

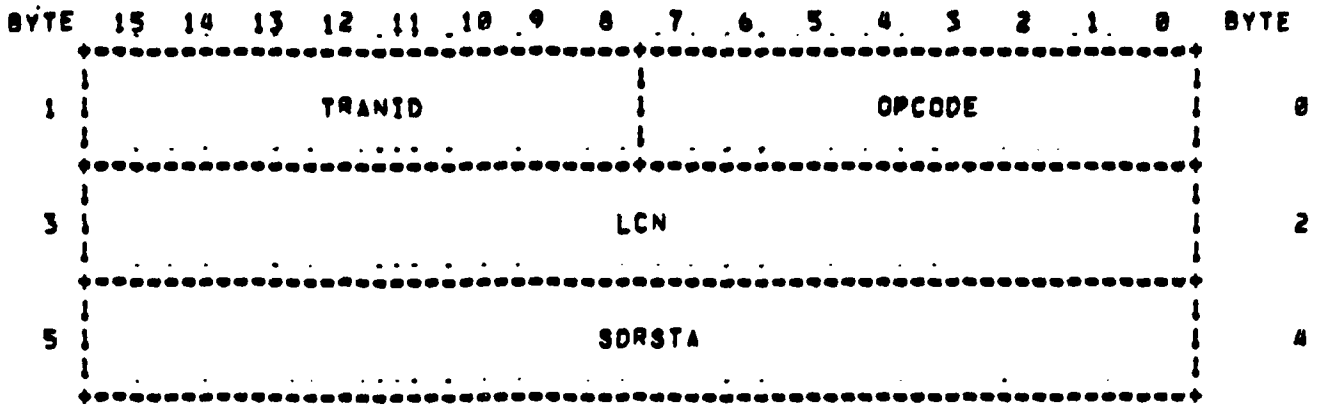
REASON:

This event is normally sent when SIP completes transmission of the segment to the source switch. Either transmission successful or unsuccessful is returned in this case. The event is also sent if SIP receives a network delivery status indicator from its source switch that indicates a fatal condition to TCP. These status conditions are shown below.

COMMENTS:

VERSION: 29-SEP-78

SEND DATA RETURN EVENT
29-SEP-78



SEND DATA RETURN EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Send Data Return event = 72.
TRANID	8	Transaction ID. Location of segment referenced by the Send Data Return event. This address is word, not byte, oriented.
LCN	16	Local Connection Name. Internal SIP-TCP connection identification value with range 1-32.
SDRSTA	16	Status: 0=successful to source SCM; 2=SCM flow reject; 16=security error; 17=precedence error; 18=TCC error; 19=invalid address; 20=destination subscriber down; 21=destination access circuit down; 22=destination subscriber busy; 36=line error.

EVENT SPECIFICATION

EVENT NAME: SEND EVENT
MNEMONIC: OSEND

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event allows THP to send a letter to the network for the referenced connection.

REASON:

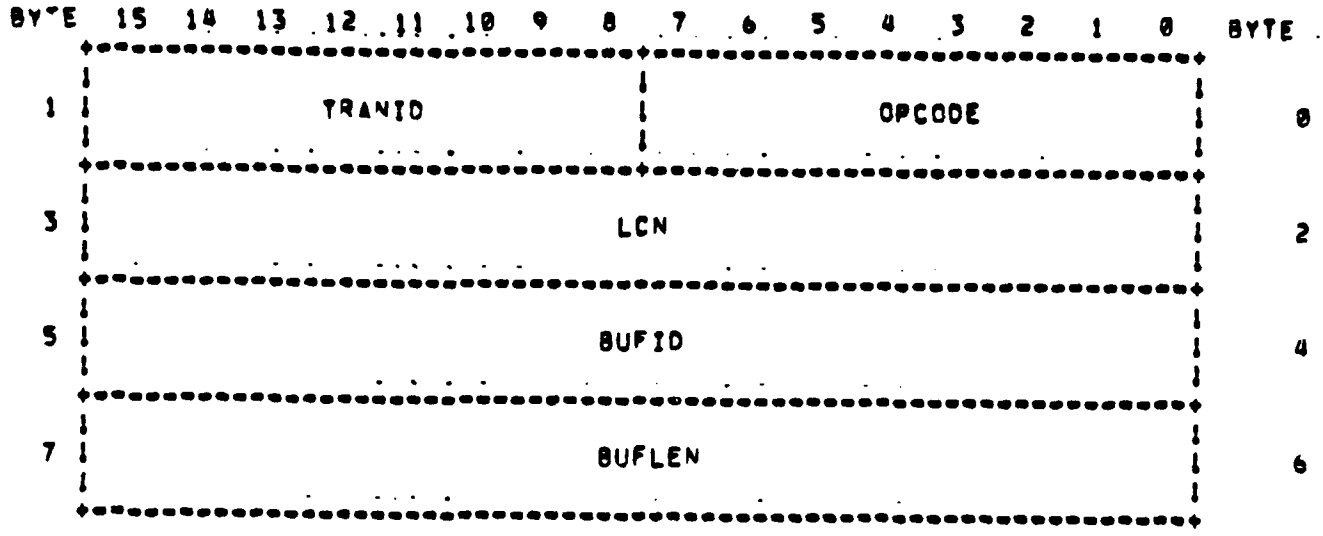
THP builds a Send event when it has received enough data from HSI to satisfy the packet release mechanism for this user.

COMMENTS:

TCP will release the letter buffer when Send event processing is complete.

VERSION: 29-SEP-78

SEND EVENT
29-SEP-78



SEND EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Send event = 65.
TRANID	8	Transaction ID, Sequence number of this event, for coordination with return event, with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.
BUFID	16	Buffer ID, Tag used to map to the buffer containing the letter to be sent to the network.
BUFLEN	16	Buffer Length, Size, in bytes, of the letter to be sent to the network.

EVENT SPECIFICATION

EVENT NAME: SEND DATA EVENT
MNEMONIC: OSNDAT

CPCI: MCCU

SENDER: TCP

RECEIVER: NIP

PURPOSE:

This event allows TCP to request that SIP transmit a segment to the SCM.

REASON:

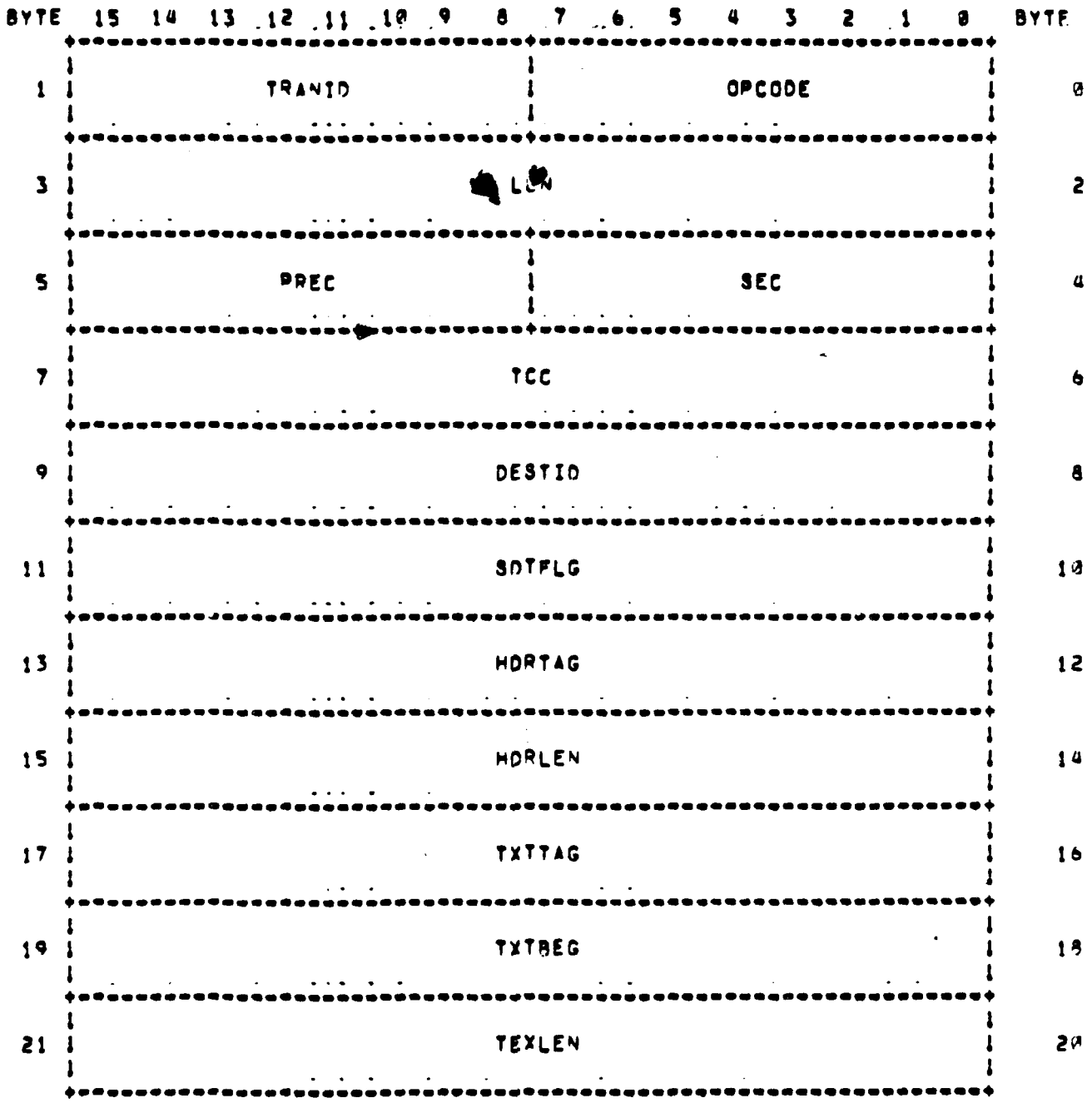
SIP must build and "attach" the Binary Segment Leader to the segment being sent to the network by TCP.

COMMENTS:

TCP sends both segments that contain data and segments that contain only control information.

VERSION: 29-SEP-78

SEND DATA EVENT
29-SEP-78



SEND DATA EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Send Data event = 96.
TRANID	8	Transaction ID. Location of segment referenced by the Send Data event. This address is word, not byte, oriented.
LCN	16	Local Connection Name. Internal SIP-TCP connection identification value with range 1-32.
SEC	8	Security to assign to the segment; 15 is high; 0 is low.
PREC	8	Precedence to assign to the segment; 15 is high; 0 is low.
TCC	16	TCC to assign to the segment with range 0-512.
DESTID	16	Destination subscriber ID to assign to the segment.
SDTFLG	16	Send Data Flag Word. Values currently undefined for the MCCU.
HDRTAG	16	Buffer ID of the T-segment header.
HDRLEN	16	Length, in bytes, of the T-segment header.
TXTTAG	16	Buffer ID of the data portion of the segment; zero if no data being sent.
TXTREG	16	Offset into the data buffer to the first byte of data to be transmitted; value is relative to one, not zero.
TEXLEN	16	Number of bytes to be transmitted.

EVENT SPECIFICATION

EVENT NAME: SEND RETURN EVENT
MNEMONIC: OSNDRT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event provides THP the status of the corresponding Send event processing.

REASON:

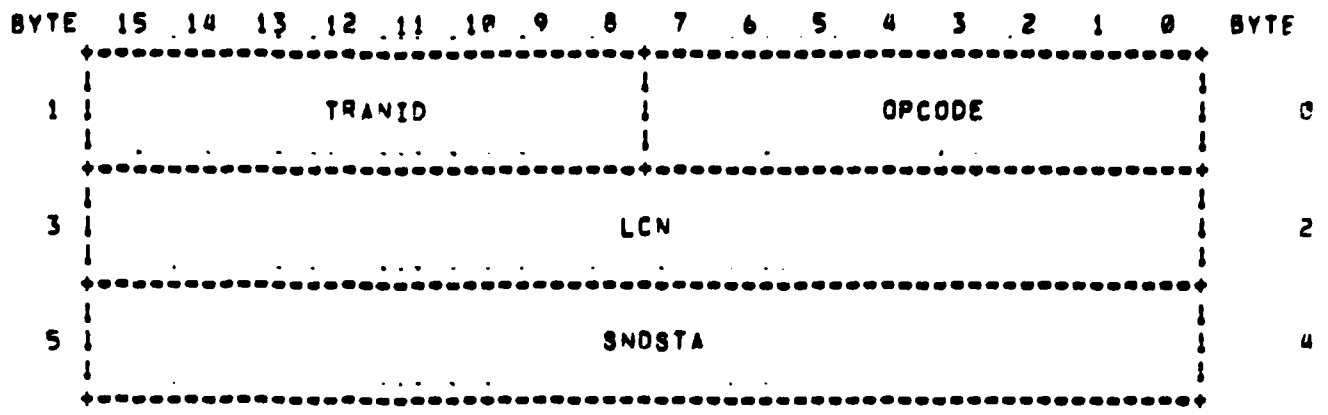
This event is returned if TCP finds an error in the preliminary verification of the Send event, if the data referenced in the event is delivered, or if the data is not delivered. THP must be aware of Send event processing completion as the number of outstanding Send events is limited as part of the flow control mechanism for the MCCU.

COMMENTS:

If an "undelivered" status is returned, the connection will be closing. The particular reason for the close will be returned in a Close Return event.

VERSION: 29-SEP-78

SEND RETURN EVENT
29-SEP-78



SEND RETURN EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Send Return event = 41.
TRANID	8	Transaction ID, Sequence number of the corresponding Send event with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.
SNDSTA	16	Status of Send event processing: 0=data delivered; 1=nonexistent connection; 4=connection closing when Send event received; 5=send illegal in listen state; 7=data undelivered (connection will be closed).

EVENT SPECIFICATION

EVENT NAME: SEND WINDOW EVENT
MNEMONIC: OSNWND

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event notifies THP of the new value to be used as the send window on the user-to-network data path. The value represents the maximum number of Send events that THP may have outstanding to TCP at any one time. That is, the number of Send events that have not been acknowledged by TCP with corresponding Send Return events.

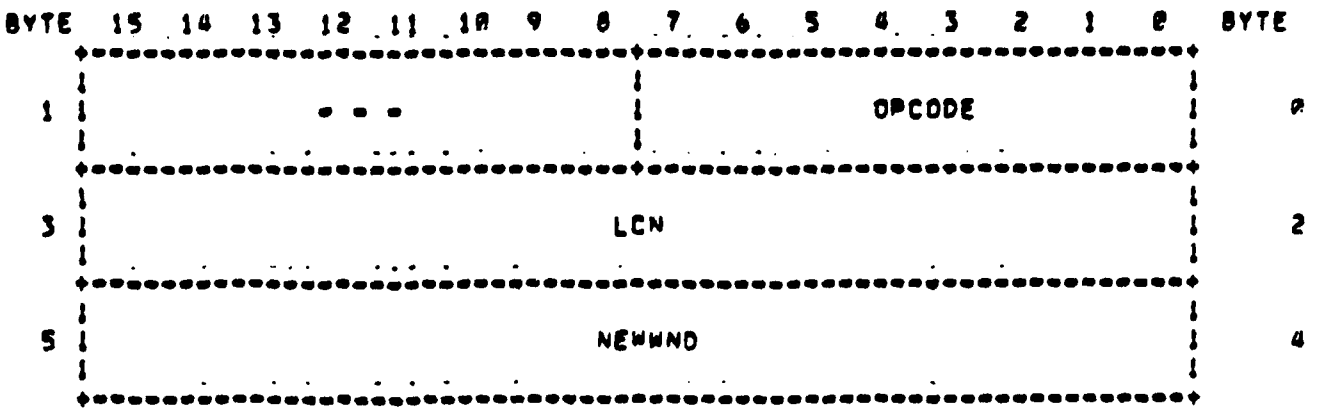
REASON:

The send window is part of the flow control used by TCP to ensure that the virtual connection sustains a steady flow of data in either direction with no appreciable backups in any area. If the send window "closes," THP will eventually stop sending From User Return events to HSI, and HSI will eventually begin holding off the host.

COMMENTS:

VERSION: 29-SEP-78

SEND WINDOW EVENT
29-SEP-78



SEND WINDOW EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier. Value for Send Window event = 46.
LCN	16	Local Connection Name. Internal THP-TCP connection identification value with range 1-32.
NE-WND	16	New Send Window. New value for send window, indicating the number of letters which may be outstanding to TCP (Send Return events not received). Value must be greater than or equal to the current send window value.

EVENT SPECIFICATION

EVENT NAME: STATUS EVENT
MNEMONIC: OSTATS

CPCI: MCCU

SENDER: THP

RECEIVER: TCP

PURPOSE:

This event requests that TCP return the current status of the specified connection as relates to the local TCP.

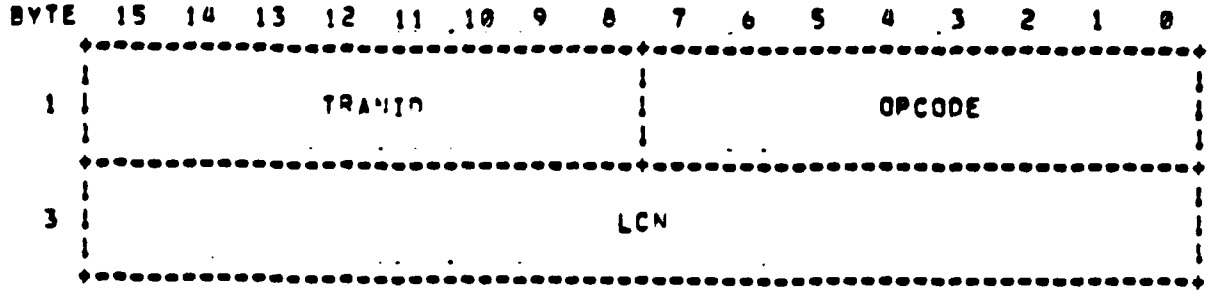
REASON:

The user has entered a THP status command and THP believes there is an active connection that is not being held off in THP itself. THP first requests the TCP status of the connection. If that status is "normal," THP will attempt to communicate with the remote THP via a THP status request record.

COMMENTS:

VERSION: 05-NOV-78

STATUS EVENT
05-NOV-78



STATUS EVENT
05-NOV-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Status event = 68.
TRANID	8	Transaction ID, Sequence number for this event, for coordination of the return event, with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.

EVENT SPECIFICATION

EVENT NAME: STATUS RETURN EVENT
MNEMONIC: OSTSRT

CPCI: MCCU

SENDER: TCP

RECEIVER: THP

PURPOSE:

This event notifies THP of the current status of the specified connection as relates to TCP.

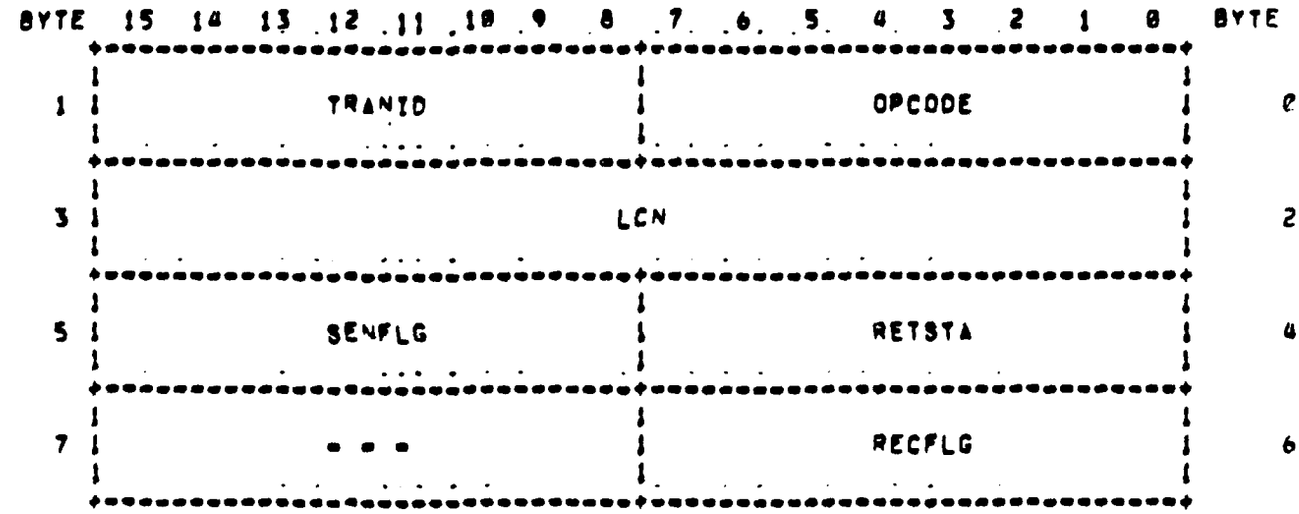
REASON:

This event is sent in response to a Status event.

COMMENTS:

VERSION: 29-SEP-78

STATUS RETURN EVENT
29-SEP-78



STATUS RETURN EVENT
29-SEP-78

MNEMONIC	# OF BITS	DESCRIPTION
OPCODE	8	Event Identifier, Value for Status Return event = 44.
TRANID	8	Transaction ID, Sequence number of the corresponding Status event with range 1-255.
LCN	16	Local Connection Name, Internal THP-TCP connection identification value with range 1-32.
RETSTA	8	Return Status, 0 = connection established; 1 = connection does not exist; 4 = connection is closing.
SENFLG	8	Send Flag, Value indicating current TCP status of the send path; 0 = open; 1 = blocked.
RECFLG	8	Receive Flag, Value indicating current TCP status of the receive path; 0 = open; 1 = blocked, window closed; 2 = blocked, reassembly queue.

APPENDIX D

TCP STATE TABLES

This appendix contains tables which describe the action taken by an MCCU TCP depending on the current state of the connection, reflected in the TCB, and the stimuli received. There are many different stimuli received by TCP, e.g., Send events from THP, Receive Data events from SIP, etc. The processing required by each stimulus is a function of the state of the connection at the time of receipt.

It should be noted that these state tables are relevant only if a TCB exists, i.e., TCP is aware of the local user via an open or listen request. If a TCB does not exist, TCP will perform one of the following actions:

1. If stimulus is an event, "connection does not exist" status will be returned in appropriate return event.

2. If stimulus is an incoming non-error segment, an error segment with control data extension set to "connection does not exist" will be sent to the source TCP.

3. If stimulus is an incoming error segment, TCP will discard the segment. Returning another error segment would result in an endless loop of error messages between the two TCPs.

These tables do not address preemption. They represent the processing accomplished by an MCCU TCP once an appropriate TCB is found. The reader's attention is directed to the TCP Specification, Paragraph 2.3.4 (demultiplexing function) and Paragraph 2.4.3 (preemption).

Definitions for TCP states are summarized here with definitions of other terms to aid the reader in interpretation of these tables.

TCP states

1. open - THP has issued a fully specified Open event, but the three-way handshake has not begun

2. listen - THP has issued a partially specified Open event (security, precedence, TCC, or destination address were not specified)

3. SYN sent - a SYN segment (request to synchronize send sequence numbers) has been sent, beginning the three-way handshake

4. SYN sent/received - TCP has sent and received a SYN segment, but the acknowledgement has not been received for the SYN segment that was sent

5. established - the three-way handshake is complete; TCP has sent and received a SYN segment, and has received an acknowledgement for the SYN segment that was sent

6. local close received - TCP has received a Close event from the local THP, requesting a deferred (graceful) close of the connection, but has not sent the "FIN" segment; network data will not be delivered to the local THP (Receive events) for this connection once this state has been entered

7. remote close received - TCP has received a "status" control segment from the remote TCP, indicating that the remote TCP has begun close processing for the connection; TCP will continue to accept network data and pass it to THP (Receive events), but will accept no more THP letters destined for the network (Send events)

8. FIN sent - TCP has sent a FIN segment, beginning connection closure; no further data will be accepted from (Send events) or given to (Receive events) the local THP on this connection

9. FIN received - TCP has received a FIN segment but is unable to send a FIN segment because all octets sent have not been accounted for as yet

10. FIN sent/received - TCP has sent and received a FIN segment, but has not received an acknowledgment for the FIN segment that was sent

11. closed - TCP has sent and received a FIN segment and has received an acknowledgement for the FIN segment that was sent, but TCP is waiting for all its queues to clear before deleting the TCB.

other terms

1. retransmission queue - an internal TCP queue which holds all unacknowledged segments; segments may be retransmitted if not acknowledged by the remote TCP within a certain amount of time or if a line error indication was received from SIP

2. segment sequence number - 31-bit value representing the number of the first octet (data character) of the segment (see also Paragraph 2.1.4 of the TCP specification)

3. acknowledgement sequence number - 31-bit value representing the number of the next expected (by the sender of this segment) octet; the acknowledgement indicates that all octets with a lower sequence number have been delivered to THP

4. send left window edge - 31-bit value representing the number of the first unacknowledged octet sent to the remote TCP on the virtual connection

5. receive left window edge - 31-bit value representing the number of the next octet expected to be received from the remote TCP on the connection

6. receive window - 16-bit value representing the number of octets which will be accepted from the remote TCP on the connection; value is sent with every segment

7. send window - 16-bit value representing the number of octets which may be sent to the remote TCP; corresponds to remote TCP's receive window.

STIMULUS: SYN segment received from network*

CURRENT STATE	PROCESSING REQUIRED
open	set value of receive left window edge to that of SYN segment sequence number plus one; send a SYN segment; acknowledge the SYN segment received; move to SYN sent/received state; record value of sequence number of SYN segment received
SYN sent	perform same processing as for open state
SYN sent/received	using value of the sequence number of original SYN segment received, verify that SYN segment received is a retransmission of original SYN received; if not, send an "unacceptable SYN" message to remote TCP, return all Send events to THP, send a Close Return event to THP, and move to closed state; if retransmission, discard the segment
established	using value for sequence number of original SYN segment received, verify that this is a retransmission of original SYN; if not, send an "unacceptable SYN" message to remote TCP, return all not yet segmentized Send events to THP with undelivered status, send a FIN segment, and move to FIN sent state; if retransmission, discard the segment
all other states	discard the segment

*NOTE: This table does not address preemption. It represents the processing accomplished by an MCCJ TCP once an appropriate TCB has been found. The reader's attention is directed to the TCP Specification, Paragraph 2.3.4 (demultiplexing function) and Paragraph 2.4.3 (preemption) for additional details.

STIMULUS: ACK segment received from network

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
SYN sent	check if this segment acknowledges the SYN segment that was sent and also carries a SYN; if both are true, acknowledge for SYN received, send Open Complete event to THP, and move to established state; if this ACK segment is not an acknowledgement for SYN segment that was sent, discard the segment
SYN sent/received	check if this segment acknowledges the SYN segment that was sent; if so, send Open Complete event to THP and move to established state; if not, discard the segment
established	check if this segment acknowledges something on the retransmission queue; if not, discard the segment; if so, update value of send left window edge with the sequence number in the acknowledgement field, delete all segments from the retransmission queue with a sequence number lower than the ACK segment, and send a Send_Return event with successful delivery status for each Send event carrying acknowledged data
local close received	perform same processing as for established state
remote close received	perform same processing as for established state
FIN sent	perform same processing as for established state
FIN received	perform same processing as for established state; if all data sent has been accounted for, send a FIN segment and move to FIN sent/received state
FIN sent/received	perform same processing as for established state; if the new send left window edge equals the value of the next send sequence number, move to closed state

TCP State Tables
,TIMULUS: Data received from network

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
SYN sent	discard the segment
SYN sent/received	discard the segment
established	check for valid sequence number; if valid, order the data segment on the reassembly queue; as holes in the reassembly queue are filled or if segments arrive in order, deliver data to THP (Receive event)
local close received	check for valid sequence number; if valid, order the data segment on the reassembly queue; as holes in the reassembly queue are filled or if segments arrive in order, update the value of the receive left window edge, send a status control segment to the remote TCP, and delete those segments on the reassembly queue that are being acknowledged
remote close received	perform same processing as for established state
FIN sent	perform same processing as for local close received state
FIN received	perform same processing as for established state; if the data received causes the receive left window edge to equal the value plus one of the FIN segment that was received, send a FIN segment with ACK for the FIN segment that was received, and move to FIN sent/received state
FIN sent/received	perform same processing as for local close received state; if send left window edge equals the value of the next send sequence number, and the receive left window edge equals the value plus one of the FIN segment that was received, send a status control segment to acknowledge the FIN segment that was received, and move to closed state

TCP State Tables PAGE D-7
STIMULUS: FIN (no flush) segment received from network

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
SYN sent	discard the segment
SYN sent/received	discard the segment
established	return all not yet segmentized Send events to TMP with undelivered status, record the sequence number of the FIN segment that was received and move to FIN received state; if the sequence number of the FIN segment that was received equals the value of the current receive left window edge, increment the received left window edge, and send an ACK segment for the FIN segment that was received, send a FIN segment and move to FIN sent/received state
local close received	perform same processing as for established state; it should be noted that this situation changes the graceful local close to an immediate close (abort)
remote close received	record the sequence number of the FIN segment that was received and move to FIN received state; if the sequence number of the FIN segment that was received equals the current receive left window edge, increment the value of the receive left window edge, send a FIN segment with ACK control, and move to FIN sent/received state
FIN sent	record the sequence number of the FIN segment that was received and move to FIN sent/received state; if the sequence number of the FIN segment that was received equals the current receive left window edge, increment the receive left window edge, send a status control segment for the FIN segment that was received and stay in the FIN sent/received state; if the send left window edge equals the next send sequence number and the receive left window edge equals the value plus one of the sequence number of the FIN segment that was received, move to closed state
all other states	discard the segment

STIMULUS: FIN (flush) segment received from network

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
SYN sent	discard the segment
SYN sent/received	discard the segment
established	return all not yet segmentized Send events to THP with undelivered status, purge the reassembly queue, set the receive left window edge to the sequence number plus one of the FIN segment that was received, send a FIN segment with an ACK for the FIN segment that was received, and move to FIN sent/received state
local close received	perform same processing as for established state; it should be noted that this situation changes the graceful local close to an immediate close (abort)
remote close received	purge the reassembly queue, set the receive left window edge to the sequence number plus one of the FIN segment that was received, record the sequence number of the FIN segment that was received, send a FIN segment, send an ACK segment for the FIN segment that was received, and move to FIN sent/received state
FIN sent	perform same processing as for remote close received state, however, if the value of the next send sequence number equals the send left window edge, move to closed state as opposed to FIN sent/received state
all other states	discard the segment

JTIMULUS: Status segment received from network

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
SYN sent	discard the segment
SYN sent/received	discard the segment
established	return all not yet segmentized Send events to THP with undelivered status, and move to remote close received state; if sequence number of the last octet delivered to remote THP acknowledges items on the retransmission queue, update value of send left window edge, delete all segments acknowledged from the retransmission queue, and build Send Return events with successful status for each Send event carrying acknowledged data; if the sequence number of the last octet undelivered but accounted for references a segment on the retransmission queue, update the value of the send left window edge, delete all segments with lower sequence numbers from the retransmission queue, and build a Send Return event with undelivered status for each Send event referenced
local close received	this implies that the remote TCP is also in the process of closing); return all not yet segmentized Send events to THP, send a FIN segment, and move to the FIN sent state; it should be noted that this situation changes the graceful local close to an immediate close (abort); perform same processing as for established state with respect to sequence numbers
remote close received	perform same processing as for established state with respect to sequence numbers
FIN sent	perform same processing as for established state with respect to sequence numbers
FIN received	perform same processing as for established state with respect to sequence numbers
FIN sent/received	perform same processing as for established state with respect to sequence numbers; if the value of current send left window edge matches the value of next send sequence number, move to closed state

TCP State Tables
STIMULUS: TCP timeout (Retry event) occurred

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
all other states	for all segments on the retransmission queue that are within the current send window and have received a Send Data Return event from SIP, decrement count of number of seconds until retransmission of the segment; if the seconds count for a segment reaches zero and there have been less than three transmission attempts for the segment, retransmit the segment and mark it "Send Data Return event outstanding;" if the transmission count reaches three, close the connection

"connection does not exist"

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
all other states	if the sequence number in the acknowledgement field references a value between the current send left window edge and next send sequence number, perform immediate close processing for the connection; if not, discard the segment

"security violation"

CURRENT STATE	PROCESSING REQUIRED
open,closed	discard the segment
all other states	if the sequence number in the acknowledgement field references a value between the current send left window edge and next send sequence number, perform immediate close processing for the connection; if not, discard the segment

"TCC violation"

CURRENT STATE	PROCESSING
open,closed	discard the segment
all other states	if the sequence number in the acknowledgement field references a value between the current send left window edge and next send sequence number, perform immediate close processing for the connection; if not, discard the segment

"connection preempted"

CURRENT STATE	PROCESSING
open,closed, SYN sent, and SYN sent/received	discard the segment
all other states	if the sequence number in the acknowledgement field references a value between the current send left window edge and next send sequence number, perform immediate close processing for the connection; if not, discard the segment

TCP State Tables
STIMULUS: TCP messages received from remote TCP

"unacceptable SYN"

CURRENT STATE	PROCESSING
SYN sent	if the sequence number in the acknowledgement field references a value between the current send left window edge and next send sequence number, perform immediate close processing for the connection; if not, discard the segment
all other states	discard the segment

TCP State Tables PAGE D-13
STIMULUS: Send event received from local THP

CURRENT STATE	PROCESSING REQUIRED
open	queue the Send event for the segmentizer and initiate the threeway handshake by sending SYN segment to remote TCP and moving to SYN sent state
SYN sent	queue the Send event for the segmentizer
SYN sent/received	queue the Send event for the segmentizer
established	queue the Send event for the segmentizer
local close received	return a Send Return event to THP with connection closing status
remote close received	return a Send Return event to THP with connection closing status
FIN sent	return a Send Return event to THP with connection closing status
FIN received	return a Send Return event to THP with connection closing status
FIN sent/received	return a Send Return event to THP with connection closing status
closed	return a Send Return event with connection does not exist status

TCP State Tables
STIMULUS: Receive Return event received from local THP

CURRENT STATE	PROCESSING REQUIRED
open,closed	ignore the event
SYN sent	ignore the event
SYN sent/received	ignore the event
established	update the receive left window edge by the number of octets acknowledged in THP's Receive Return event and sent an ACK segment to remote TCP; check reassembly queue for a complete letter, the beginning sequence number of which equals the new receive left window edge; if one is found send letter to THP via Receive event
local close received	ignore the event
remote close received	perform same processing as for established state
FIN sent	ignore the event
FIN received	ignore the event
FIN sent/received	ignore the event

TCP State Tables PAGE D-15
STIMULUS: Send Data Return event for retransmission queue element

NOTE: Data, SYN, FIN, flush, and WOPEN segments are retransmission queue elements.

CURRENT STATE	PROCESSING REQUIRED
open	discard the segment
all other states except closed	locate the segment referenced by the Send Data Return event on the retransmission queue and mark the segment "Send Data Return event received;" if the return status is successful, increment the count of successful transmissions of the segment and set a retransmission timer interval; if Type 1 error status (nondelivery notice = flow control and ANCCP line error) was received, increment the count of unsuccessful transmissions of the segment and, if the transmission count has not reached three, set a one second timer for retransmission of the segment; if the count has reached three, move to the closed state, delete all segments from retransmission queue with "Send Data Return event received" status, sending Send Return events to THP, as appropriate, purge all other queues for the connection, and send a Close Return event to THP with appropriate status as soon as all segments have been deleted; if Type 2 or 3 error status (nondelivery notice = undeliverable or validation reject, respectively) was received, move to closed state and initiate connection closure as described for Type 1, eliminating the retransmission attempts
closed	locate the segment on the retransmission queue and mark the segment "Send Data Return event received;" scan the retransmission queue and delete all segments with Send Data Return event received status and issue appropriate Send Return events to THP; once all segments on the retransmission queue have been deleted, issue a Close Return event to THP

TCP State Tables
STIMULUS: Send Data Return event for ACK queue element

NOTE: ACK, STA, WACK, and OBI segments are ACK queue elements.

CURRENT STATE	PROCESSING REQUIRED
open, SYN sent	discard the segment
SYN sent/received, established, local close received, remote close received, FIN sent, and FIN received	locate the segment referenced in the Send Data Return event and delete it from the ACK queue; if Type 2 or 3 error status (nondelivery = undeliverable or validation reject, respectively) was returned, move to closed state, delete all segments from the retransmission queue with "Send Data Return event received" status, purge all other connection queues, and, if all queues are empty, send a Close Return event to THP
FIN sent/received	perform same processing as described above; if Type 3 error is not received, check to see if the Send Data Return event is for the acknowledgement sent for the FIN segment that was received (current send left window edge is equal to next send sequence number); if the ACK segment was for the FIN segment that was received, move to closed state and, if all connection queues are empty, send a Close Return event to THP
closed	locate the segment referenced in the Send Data Return event and delete it from the ACK queue; if all connection queues are empty, send a Close Return event to THP

TCP State Tables
STIMULUS: Interrupt event received from local THP

CURRENT STATE	PROCESSING REQUIRED
open,closed	reject request by sending Interrupt Return event with connection does not exist reason code
SYN sent	reject request by sending Interrupt Return event with connection does not exist reason code
SYN sent/received	reject request by sending Interrupt Return event with connection does not exist reason code
established	if flush indicator is set in request, return all not yet segmentized Send events to THP with undelivered status and send flush control segment to remote TCP (NOTE: flush control segment is subject to accountability and, therefore, will be put on the retransmission); if interrupt indicator is sent in request, send an out-of-band interrupt (OBI) control segment to remote TCP (NOTE: OBI control segments are not subject to accountability and, therefore, will be put on the ACK queue)
all other states	reject request by sending Interrupt Return event with connection closing reason code

TCP State Tables PAGE D-1A
STIMULUS: Close event received from local THP

CURRENT STATE	PROCESSING REQUIRED
open	purge all connection queues and send THP Close Return event with successful local close status
SYN sent	return all Send events to THP with undelivered status, record the transaction ID of the Close event, and move to closed state; once all segments already transmitted are accounted for, send THP a Close Return event with successful local close status
SYN sent/received	perform same processing as for SYN sent state
established	record the current value of the receive left window edge for possible use later in status control segment; if deferred (graceful) close is specified, queue the Close event for the segmentizer; if no entries are on the send queue, send a FIN segment immediately and move to FIN sent state; if an immediate close is specified, return all not yet segmentized Send events to THP with undelivered status, send a FIN (flush) segment, and move to FIN sent state
local close received	assume this is an abort request (second Close event received) from THP; return Close Return event for first Close event with close flushed status, return all not yet segmentized Send events to THP with undelivered status, send a FIN segment, move to FIN sent state, and record transaction ID of new Close event
remote close received	record the current value of the receive left window edge, send a FIN segment, and move to FIN sent state
FIN sent	send THP a Close Return event with close flushed status for first Close event Close event
FIN received	perform same processing as for established state; set the value of the receive left window edge to that of the value of the sequence number plus one of the FIN segment that was received, send a FIN segment, send a status control segment for the FIN segment that was received, and move to FIN sent/received state

STIMULUS: Close event received from local THP

FIN sent/received	perform same processing as for established state; it should be noted that this situation changes the graceful local close to an immediate close (abort)
closed	send Close Return event with connection does not exist status

INDEX

APPENDIX E

AUTODIN II TRANSMISSION UNITS

There are standard units of transmission for each component of a CCU/TAC, i.e., HSI or TH, THP, TCP, and SIP. It is important to understand the terminology used to describe each, the contents of each, and the relationship of all the units of transmission. The units are discussed below beginning with the "smallest" unit (user text) to the "largest" (SIP or AUTODIN II segment). A visual conception of the relationship is shown in Figure E-1. Figure E-2 shows a "direction of transmission" schematic.

1. user text - This unit of transmission is the smallest for a CCU/TAC. User text is received by HSI or TH on the host-to-CCU or terminal-to-TAC link, respectively. The text may come in a character at a time or in blocks of several characters. HSI/TH accumulates text and passes it to THP via the From User event in the MCCU, and by an equivalent vehicle in the SCCU and TAC. Each event may bring one or many characters, depending on the particular type of user and the link protocol being used. For the purpose of this discussion, and from THP's viewpoint, the unit of transmission called "user text" should be considered as one or more user text characters which, as a unit, have meaning to THP. That is, some characters have special meaning to THP, such as, interrupt function characters, erase line characters, carriage return, linefeed, etc. Other characters have no special meaning to THP, and are accumulated as data to be transmitted to the network in a unit called the THP data record.

2. THP record - This unit of transmission is the means by which THPs communicate. That is, each THP record contains information instructing THP as to the required processing. The most common THP record, the data record, consists of user text characters preceded by an 8-bit record mark, an 8-bit record type, and a 16-bit length field. Other THP records consist of an 8-bit record mark, an 8-bit record type, and optional parameter bytes. All THP record types and formats are defined in Appendices A and B of the THP Transportable Specification.

3. THP letter - This unit of transmission is associated with THP and consists of one or more THP records. If only user text characters are being transmitted on the connection, e.g., in binary mode, requiring no other THP control records, there would be one THP data record in the THP letter. If other information is being transmitted, e.g., user text characters followed by an option request, the letter would consist of a THP data record followed by a THP option record. The size of the letter is determined by the various packet release mechanisms of THP (see Reference 9). Letters are passed to TCP via the Send event in the MCCU, and by an equivalent vehicle in an SCCU or TAC.

4. T-segment - This unit of transmission, also known as TCP segment, is associated with TCP and consists of none, one or more, but never partial THP letters, preceded by a T-segment header. The T-segment header is the means by which TCPs communicate, and conveys send/receive flow control information, TCP control information, and addressing information (see Appendix A of the TCP Transportable Specification). The T-segment is passed to SIP for transmission to the SCM via a Send Data event in the MCCU, and the equivalent in an SCCU or TAC.

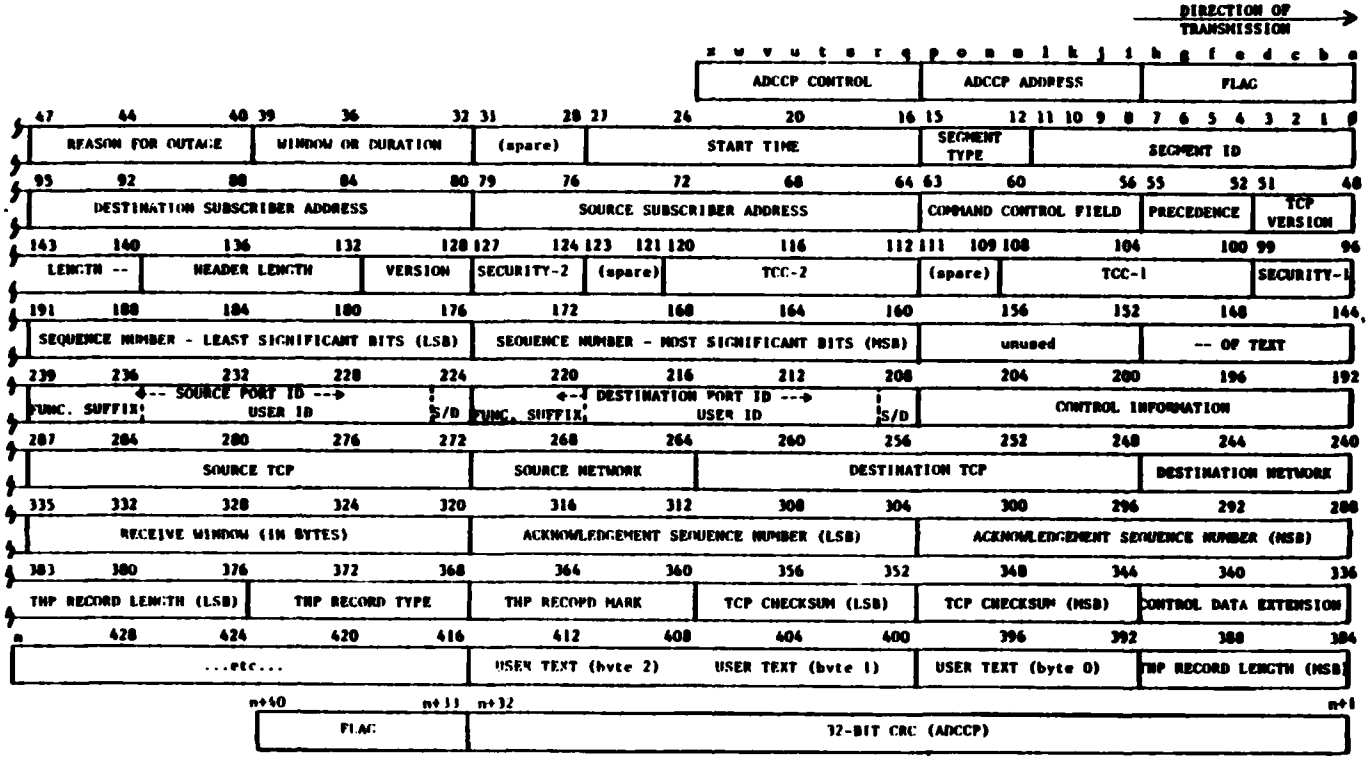
5. S-segment - This unit of transmission, also known as SIP segment or AUTODIN II segment, is associated with SIP. It is the unit transferred between the CCU or TAC and SCM. The S-segment consists of the T-segment preceded by a binary segment leader (BSL). The BSL contains security, precedence, TCC, addressing, and SIP-SCM control information concerning the S-segment being sent.

PDP-11 MEMORY IMAGE AUTODIN II SEGMENT FORMAT

BYTE	15.14.13.12.11.10.9.8.7.6.5.4.3.2.1.0.	BYTE	WORD	
1	SEGMENT TYPE / SEGMENT ID	0	0	BSI
3	(spare) / START TIME	2	1	
5	REASON FOR OUTAGE / WINDOW OR DURATION	4	2	
7	COMMAND CONTROL FIELD / PRECEDENCE / TCP VERSION	6	3	
9	SOURCE SUBSCRIBER ADDRESS	8	4	
11	DESTINATION SUBSCRIBER ADDRESS	10	5	
13	(spare) / TCC-1 / SECURITY-1	12	6	
15	SECURITY-2 / (spare) / TCC-2	14	7	
17	LENGTH -- / HEADER LENGTH / VERSION	16	8	TCP HEADER
19	unused / -- OF TEXT	18	9	
21	SEQUENCE NUMBER - MOST SIGNIFICANT BITS (MSB)	20	10	
23	SEQUENCE NUMBER - LEAST SIGNIFICANT BITS (LSB)	22	11	
25	CONTROL INFORMATION	24	12	
27	<-- DESTINATION PORT ID --> / FUNC. SUFFIX; USER ID ;S/D	26	13	
29	<-- SOURCE PORT ID --> / FUNC. SUFFIX; USER ID ;S/D	28	14	
31	DESTINATION TCP (LSB) / DESTINATION NETWORK	30	15	
33	SOURCE NETWORK / DESTINATION TCP (MSB)	32	16	
35	SOURCE TCP	34	17	
37	ACKNOWLEDGEMENT SEQUENCE NUMBER (MSB)	36	18	
39	ACKNOWLEDGEMENT SEQUENCE NUMBER (LSB)	38	19	
41	RECEIVE WINDOW (IN BYTES)	40	20	
43	TCP CHECKSUM (MSB) / CONTROL DATA EXTENSION	42	21	
	TCP CHECKSUM (LSB)	44	22	
45	THP RECORD MARK			
47	THP RECORD LENGTH (LSB) / THP RECORD TYPE	46	23	
49	USER TEXT (byte 0) / THP RECORD LENGTH (MSB)	48	24	
51	USER TEXT (byte 2) / USER TEXT (byte 1)	50	25	
:	etc.	:	:	

NOTE: TCP checksum field (bytes 43 and 44) is not used in the CCU/TAC/NCC TCP implementation.

Figure E-1. AUTODIN II Segment Format



- Binary Segment Leader (BSL) - bits ∅-127
- TCP Header (Note: TAC, CCU, and NCC implementations of TCP do not use checksum field (bits 344 thru 359)) - bits 128-359
- Data (Note: This example illustrates a THP record present in the data stream). - bits 360-n
- ADCCP - 24-bit leader (a thru x)
  - 40-bit trailer (n+1 thru n+40)

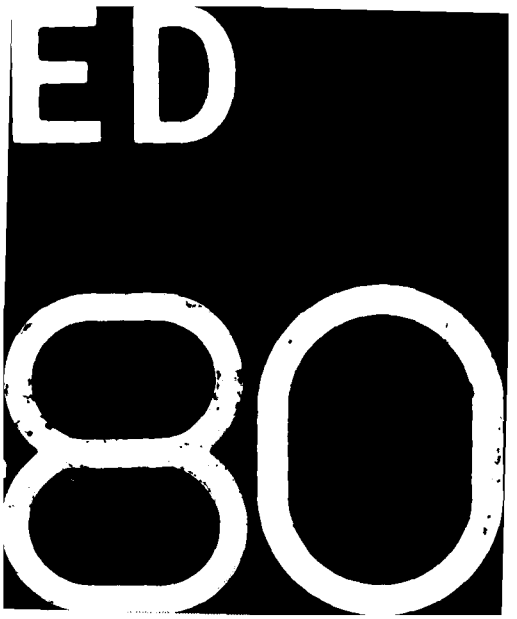
Data transmission begins with bit ∅ and ends with bit n. (Note: Link control, e.g., ADCCP, is shown outside of bits ∅ to n). Within the individual fields, the data is sent from least to most significant bit. For example, if BSL precedence = 8_{10}, the bits would be sent as follows: first bit = ∅, second bit = ∅, third bit = ∅, fourth bit = 1, ..., i.e., precedence = $1∅∅∅_2$. Therefore, 1000 -----> direction of transmission.

Figure E-2. AUTODIN II Segment Transmission Schematic

11-JUL-79 TR0044930 **UNCLASSIFIED**

AUTODIN II TRANSMISSION UNITS
Figure E-2. AUTODIN II Segment Transmission Schematic

11-JUL-79 TR0044930 **UNCLASSIFIED**


ED
80