SEMI E4-0699
SEMI EQUIPMENT COMMUNICATIONS STANDARD 1 MESSAGE TRANSFER (SECS-I)

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on February 28, 1999. Initially available at www.semi.org May 1999; to be published June 1999. Originally published in 1980; previously published January 1999.

1 Introduction

1.1 *Revision History* — This is the first major revision since the original release of SECS-I in 1980. Very little of the original intent of SECS-I has been altered, although there are a few significant additions. The changes are summarized in Appendix 1. This specification has been developed in cooperation with the Japan Electronic Industry Development Association Committee 12 on Equipment Communications.

1.2 *Scope* — The SECS-I standard defines a communication interface suitable for the exchange of messages between semiconductor processing equipment and a host. Semiconductor processing equipment includes equipment intended for wafer manufacturing, wafer processing, process measuring, assembly and packaging. A host is a computer or network of computers which exchange information with the equipment to accomplish manufacturing. This standard includes the description of the physical connector, signal levels, data rate and logical protocols required to exchange messages between the host and equipment over a serial point-to-point data path. This standard does not define the data contained within a message. The meaning of messages must be determined through some message content standard such as SEMI Equipment Communications Standard E5 (SECS-II).

1.3 *Intent* — This standard provides a means for independent manufacturers to produce equipment and/or hosts which can be connected without requiring specific knowledge of each other.

1.3.1 *Layered Protocol* — The SECS-I protocol can be thought of as a layered protocol used for point-to-point communication. The levels within SECS-I are the physical link, block transfer protocol, and message protocol. (See Related Information R1-1.1.)

1.3.2 *Speed* — It is not the intent of this standard to meet the communication needs of all possible applications. For example, the speed of RS-232 may be insufficient to meet the needs of transferring mass amounts of data or programs in a short period of time, such as might be required by high speed functional test applications.

1.3.3 *Network Support* — The method by which blocks of data are routed to a piece of equipment or find their way back to the proper host application is not specified by SECS-I. In a network, the roles of host and equipment might be assumed by any party in the network. In this situation, one end of the communications link must assume the role of the equipment and the other the role of the host.

1.4 *Applicable Documents*

1.4.1 *Electronics Industries Association Standards*[1]

EIA RS-232-C — Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange

EIA RS-269-B — Synchronous Signaling Rates for Data Transmission

EIA RS-334 — Signal Quality at Interface Between Data Processing Terminal Equipment and Synchronous Communication Equipment for Serial Data Transmission

EIA RS-422 — Electrical Characteristics of Balanced Voltage Digital Interface Circuits

EIA RS-423 — Electrical Characteristics of Unbalanced Voltage Digital Interface Circuits

1.4.2 *European Computer Manufacturing Association*[2]

ECMA/TC24/82/18 — "Network Layer Principles," Final Draft (April, 1982)

1.4.3 *Japanese Industrial Standards Committees*[3]

JIS C 6361 — "The Interface between Data Circuit Terminating Equipment (DCE) and Data Terminal Equipment (DTE) (25-pin Interface)"

1.4.4 *International Organization for Standardization*[4]

---

[1] *EIA Engineering Department, Standards Sales Office, 2001 Eye Street, N.W., Washington, D.C. 20006*

[2] *European Computer Manufacturing Association, 114 Rue du Rhone, 1204 Geneva, Switzerland*

[3] *Japanese Standards Association, 1-24, Akasaka 4 Chome, Minato-ku, Tokyo 107, Japan*

[4] *ANSI, 1430 Broadway, New York, NY 10018*

ISO 2110-1980 — Data Communications, Interface Connectors and Pin Assignment

1.4.5 *SEMI Specifications*

SEMI E5 — SEMI Equipment Communications Standard 2 — Message Content (SECS-II)

SEMI E6 — SEMI Facilities Interface Specification Format

1.5 *Overview of SECS-I* — The SECS-I standard defines point-to-point communication of data utilizing a subset of the international standard known in the U.S.A. as EIA RS-232-C and in Japan as JIS C 6361 for the connector and voltage levels. The actual transmission consists of 8-bit bytes sent serially with one start and one stop bit. The communication is bidirectional and asynchronous, but flows in one direction at a time. The direction is established by special characters and a handshake, after which the data itself is sent. Data is sent in blocks of 254 bytes or less. Each block consists of a 10-byte header followed by data. A message is a complete unit of communication in one direction and consists of 1 to 32,767 blocks. Each block header contains information for identifying the block as part of a specific message. Messages are paired by a request and its reply which together are called a transaction.

1.6 *Structure of Document* — This document is divided into sections which correspond to major aspects of the standard. The sections outline requirements as well as implications of the requirements. The standard may be implemented in a variety of ways, depending upon the computer environment where it is placed. Implementation is not part of the standard. Information which may be useful for implementation is included in the form of Related Information.

2 Terminology

2.1 The following brief definitions refer to sections providing further information.

2.1.1 *ACK* — "Correct Reception" handshake code. (See Section 5.2.)

2.1.2 *application software* — the software performing the specific task of the equipment or the host.

2.1.3 *block* — header plus up to 244 bytes of data. (See Sections 1.5, 6.7.)

2.1.4 *block length* — the number of bytes sent in the block transfer protocol. (See Section 5.6.)

2.1.5 *block number* — a 15-bit field in the header for numbering blocks in a message. (See Sections 6.7.)

2.1.6 *character* — a byte sent on the SECS-I serial line. (See Section 4.1.)

2.1.7 *checksum* — a 16-bit number used to detect transmission errors. (See Section 5.7.)

2.1.8 *communication failure* — a failure in the communication link resulting from a failed send. (See Section 5.4.)

2.1.9 *device ID* — a 15-bit field in the header used to identify the equipment. (See Section 6.3.)

2.1.10 *E-bit* — a bit in the header identifying the last block of a message. (See Section 6.6.)

2.1.11 *ENQ* — "Request to Send" handshake code. (See Section 5.2.)

2.1.12 *EOT* — "Ready to Receive" handshake code. (See Section 5.2.)

2.1.13 *equipment* — the intelligent system which communicates with a host.

2.1.14 *expected block* — the block of a message which is expected by the message protocol. (See Section 7.4.4.)

2.1.15 *header* — a 10-byte data element used by the message and transaction protocols. (See Section 6.)

2.1.16 *host* — the intelligent system which communicates with the equipment.

2.1.17 *length byte* — the character used to establish the block length during transmission. (See Section 5.6.)

2.1.18 *line control* — a portion of the block transfer protocol. (See Section 5.8.2.)

2.1.19 *master* — the block transfer designation for the equipment. (See Section 5.5.)

2.1.20 *message* — a complete unit of communication. (See Section 7.)

2.1.21 *message ID* — a 15-bit field in the header used in the process of message identification. (See Sections 6.5, 7.3.1.)

2.1.22 *multi-block message* — a message sent in more than one block. (See Sections 6.7, 7.2.2.)

2.1.23 *NAK* — "lncorrect Reception" handshake code. (See Section 5.2.)

2.1.24 *open message* — a multi-block message for which not all of the blocks have been received. (See Section 7.4.4.)

2.1.25 *open transaction* — a transaction in progress. (See Section 7.3.)

2.1.26 *primary message* — a message with an odd numbered message ID. Also the first message of a transaction. (See Section 6.5.)

2.1.27 *primary/secondary attribute* — the least significant bit of the lower message ID which indicates whether a block belongs to a primary or secondary message.

2.1.28 *R-bit* — a bit in the header signifying the direction of the message. (See Section 6.2.)

2.1.29 *receiver* — the end of the SECS-I link receiving a message. (See Section 5.8.4.)

2.1.30 *reply* — the particular secondary message corresponding to a primary message. (See Section 7.3.)

2.1.31 *reply linking* — the process of forming a transaction out of a primary and a secondary message. (See Section 7.3.1.)

2.1.32 *retry count* — the number of unsuccessful attempts to send a block in the block transfer protocol. (See Section 5.4.)

2.1.33 *RTY* — the retry limit or the number of times the block transfer protocol will attempt to retry sending a block before declaring a failed send. (See Section 5.4.)

2.1.34 *secondary message* — a message with an even numbered message ID. Also the second message of a transaction. (See Section 6.5.2.)

2.1.35 *sender* — the end of the SECS-I link sending message. (See Section 5.8.3.)

2.1.36 *slave* — the block transfer designation for the host (See Section 5.5.)

2.1.37 *system bytes* — a 4-byte field in the header used for message identification. (See Section 6.8.)

2.1.38 *T1* — receive inter-character timeout in the block transfer protocol. (See Section 5.3.1.)

2.1.39 *T2* — protocol timeout in the block transfer protocol. (See Section 5.3.2.)

2.1.40 *T3* — reply timeout in the message protocol. (See Sections 5, 7.3.2)

2.1.41 *T4* — inter-block timeout in the message protocol. (See Section 7.4.3.)

2.1.42 *transaction* — a primary message and its associated secondary message, if any. (See Section 7.3.)

2.1.43 *W-bit* — a bit in the header signifying that a reply is expected. (See Section 6.4.)

3 Coupling

3.1 Coupling refers to the physical interface at the equipment. The host will provide compatible signals at this point. No restrictions are implied for any interface other than for equipment covered by this standard.

3.2 *Electrical Interface* — The connection will include a serial interface according to EIA Standard RS-232-C for interface Type E, full duplex communication, modified by the deletions, additions and exceptions described in this section.

3.2.1 *Connector* — Either the 9-pin or 25-pin connector described in the EIA RS232 may be used. In the case of the 25-pin connector a female connector will be mounted on the equipment and a male connector will be mounted on the cable from the host. In the case of the 9-pin connector the male connector will be mounted on the equipment and a female connector will be mounted on the cable. The connector on the equipment will have female 4-40 threaded jack screw locks.

NOTE: Suitable 25-pin connectors known as Type "D" are similar to Amphenol MIN RAC 17 series with jack screw locks. Suitable 9-pin connector is also Type "D" with jackscrew locks. It is the type commonly implemented on desktop and notebook PCs.

3.2.2 *Signal Pins* — Pins on the connector have functions as defined in Table 1. Pins 1, 2, 3, and 7 of the 25-pin connector or pins 3, 2, and 5 of the 9-pin connector are required for all equipment complying with SECS-I. When using a 25-pin connector, the two power supply pins, 18 and 25, are optional as indicated. Any other pins, if used, shall comply with the RS-232-C standard.

Table 1 Signal Connections

25-Pin	*9-Pin*	*RS-232-C Circuit*	*Circuit Description*
1	--	AA	Shield
2	3	BA	Data from Equipment
3	2	BB	Data to Equipment
7	5	AB	Signal Ground
18	--	--	+12 to +15 volts (opt for the 25-pin connector)
25	--	--	-12 to -15 volts (opt for the 25-pin connector)

3.2.3 *Logic Levels* — For the signal pins 2 and 3, the logic 1 level will be a voltage less than -3 volts and the logic 0 level will be a voltage greater than +3 volts. Voltages will never exceed ± 25 volts. These values correspond to those specified by the RS-232-C standard.

3.2.4 *Power Supplies* — When using a 25-pin connector, pins 18 and 25 are optional power supplies for driving external isolation circuits. When provided, both shall be present and must be able to supply at least 50 mA. (See Related Information R1-2 for example use.)

3.3 *Data Rate* — The supported data rates on signal pins shall be 9600, 4800, 2400, 1200, and 300 baud. The same data rate shall apply for data sent to and from the equipment. The data rate shall be controlled to better than 0.5%. (See RS-269-B and RS-334.) Optional rates of 19,200 and 150 baud may be supplied if desired.

3.4 *Physical Medium* — The connection with the host may involve any medium that provides the required RS-232-C quality, signal levels and data rate at the equipment connector. The quality of signal should be such that the effective bit error rate is less than 1×10^{-6}. This rate can be achieved easily with hardwired systems. The distance limits specified in RS-232-C apply only to systems using the wiring technique described in RS-232-C. Since any method may be used in SECS-I as long as RS-232-C signals are supplied at the connector, the distance and isolation is dependent upon the design of the physical medium which is external to the SECS-I standard. (See Related Information R1-2.)

4 Character Structure

4.1 *Characters* — Data will be transmitted or received in a serial bit stream of 10 bits per character at one of the specified data rates. The standard character has one start bit (0), 8 data bits and one stop bit (1). All bit transmissions are of the same duration. The 8 data bits are numbered from 1 to 8 in the order sent (see Figure 1). The timing between characters is asynchronous with respect to the data rate. The 8 data bits may be any arbitrary code. The eight data bits will hereafter be referred to as a byte.

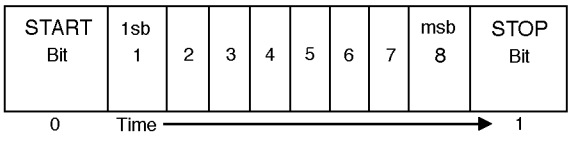

Figure 1

Character Structure

4.2 *Weighted Codes* — For bytes having weighted codes, bit one is the least significant and bit eight is the most significant. The most common weighted code is binary.

4.3 *Non-Weighted Codes* — For codes without numeric value such as ASCII, the bit numbers will be used as the entry into a standard code table for interpretation of the code. SECS-I performs no parity or other verification of the contents of individual bytes.

5 Block Transfer Protocol

5.1 The procedure used by the serial line to establish the direction of communication and provide the environment for passing message blocks is called the block transfer protocol. Most of the protocol is accomplished with a handshake of single bytes. When both ends of the line try to send at the same time, a condition known as line contention exists. The protocol resolves contention by forcing one end of the line, designated as the slave (always the host), to postpone its transmission and enter the receive mode. Retransmission of blocks is used to correct communication errors. The block transfer protocol is shown in flow chart form in Figure 2, and described below. Additional information is also contained in Related Information R1-3 and R1-4.

5.2 *Handshake Bytes* — The four standard handshake codes used in the block transfer protocol are shown in Table 2. The three letter names, ENQ, EOT, ACK, and NAK correspond to the ASCII code having the same pattern.

Table 2 Handshake Codes

Name	*Codeb8 b7.....b1*	*Function*
ENQ	00000101	Request to Send
EOT	00000100	Ready to Receive
ACK	00000110	Correct Reception
NAK	00010101	Incorrect Reception

5.3 *Timeout Parameters* — Timeouts are used to detect communications failures. A timeout occurs when the measured time between two events exceeds a predetermined limit. Generally, the length of time that must pass before it can be assumed that an error has occurred depends upon the particular systems involved. The time required in one situation might be excessively long in another. Thus, the timeout values must be "tuned" to meet the application. In the block transfer protocol, there are two situations requiring timeout values. The two timeout values are called parameters T1 and T2.

5.3.1 *Inter-Character Timeout, T1* — The inter-character timeout, T1, limits the time between receipt of characters within a block after the length byte has been received and until the receipt of the second checksum byte.

5.3.2 *Protocol Timeout, T2* — The protocol timeout, T2, limits the time between sending ENQ and receiving EOT, sending EOT and receiving the length byte, and sending the second checksum byte and receiving any character.

5.4 *Retry Limit, RTY* — The retry limit, RTY, is the maximum number of times the Block Transfer Protocol will attempt to retry sending a block before declaring a failed send. (See Section 5.8.2.)

5.5 *Master/Slave* — The master/slave parameter is used in the resolution of contention (see Section 5.8.2). The host is designated as the slave. The equipment is desig-nated as the master. This convention is based upon the assumption that the equipment is less able to store messages than the host.

5.6 *Block Lengths* — The unsigned integer value of the first byte sent after receipt of EOT is the length of the block being sent. The length includes all the bytes sent after the length byte, excluding the 2 bytes of the checksum. The maximum block length allowed by SECS-I is 254 bytes, and the minimum is 10 bytes.

5.7 *Checksum* — The checksum is calculated as the numeric sum of the unsigned binary values of all the bytes after the length byte and before the checksum in a single block. The checksum is sent as 16 bits in two bytes following the last byte of the block data. The high order eight bits of the checksum will be sent first, followed by the low order eight bits. The checksum is used by the receiver to check for transmission errors. The receiver performs the same checksum calculation on the received header and data.

5.8 *Algorithm* — The operation of the block transfer protocol is best understood by following the logic flow in Figure 2. This flow chart depicts the operation of the five states of the protocol - Receive, Idle, Send Line Control, and Completion. The flow chart shown in Figure 2 is not meant to imply that a particular implementation is required under this standard. However, any SECS-I block transfer protocol implementation must include all the logic described in Figure 2. The same algorithm is executed on each end of the SECS-I communications link.

5.8.1 *Idle State* — Both ends of the communications link are assumed to start in the Idle state. There are two primary activities of the protocol signified by the two exits from the Idle state. These are:

A. SEND — a message block is to be sent.

B. RECEIVE — the other end of the communications link has a message block to send

5.8.2 *Line Control* — The line control section establishes the transmission direction, resolves contention, and handles retries. When an ENQ is received in the Idle state, the Line Control responds with an EOT if the Block Transfer Protocol is ready to receive. The Block Transfer Protocol then goes to the Receive state. If a message block is to be sent, then an ENQ is sent. If an EOT is received in response to the ENQ within the time limit T2, the Block Transfer Protocol goes to the Send state.

5.8.2.1 If the slave receives an ENQ in response to the ENQ, contention has occurred. The slave postpones the send of its block until it receives a block from the master. The slave prepares to receive the incoming block and sends an EOT. When the block transfer protocol returns to the Idle state, the postponed block Send may be sent as if it were a new send request. After a master sends an ENQ, it can ignore all characters except an EOT. After a slave sends at ENQ, it can ignore all characters except an ENQ or EOT.

5.8.2.2 When the time between sending ENQ and receiving EOT exceeds T2, or the time between sending the second checksum byte and receiving any character exceeds T2, or a non-ACK character is received within time T2 after sending the second checksum byte, the Line Control will increment the retry count ("tries" in the flowchart). If the retry count does not exceed the value of the RTY parameter, then the Block Transfer Protocol will retry sending the block beginning with ENQ. If the retry count does exceed the value of the RTY parameter, then a failed send has occurred.

5.8.3 *Send* — Once a send state is established, the first byte sent is the length, N, of the data in the block. After N more bytes have been sent, the two bytes of the checksum are sent. The sender computes the checksum based on the data in the block and the block header, but not the length byte. When the sender receives the ACK before time T2, the block is deemed properly sent. However, if the sender receives a non-ACK character before time T2, or no character within time T2, it returns to the line control state for a possible retry. In the Send state, characters received prior to sending the last checksum byte can be ignored.

5.8.4 *Receive* — Once a receive state is established, the first byte received is the length byte, N. The receiver counts and saves the following N + 2 bytes. The last two bytes are the checksum. The receiver compares the two checksum bytes against its own computation of the checksum. In a good block, the computed and received checksum are the same.

5.8.5 *Receive Completion* — After a block is correctly received, an ACK character is sent and the message protocol is notified that a block has been received. If T2 is exceeded while waiting for the length character, or Tl is exceeded between characters being received, then an NAK is sent. If the length byte is invalid or if the received checksum does not agree with the computed checksum, the receiver continues to listen for characters to ensure that the sender is finished sending. This is detected when the inter-character time exceeds T1, at which point the receive is aborted and a NAK is sent. In

any case when an NAK is sent, the receiver may discard any data it has received for the block. The receiver returns immediately to the Idle state after sending an ACK or NAK.

5.8.6 *Send Completion* — After the second checksum byte is sent, if an ACK is received within time T2, the message protocol is notified that the send was successful. If a failed send occurs, the message protocol is informed of the failure. Application-level software must take the possibility of a send failure into account.

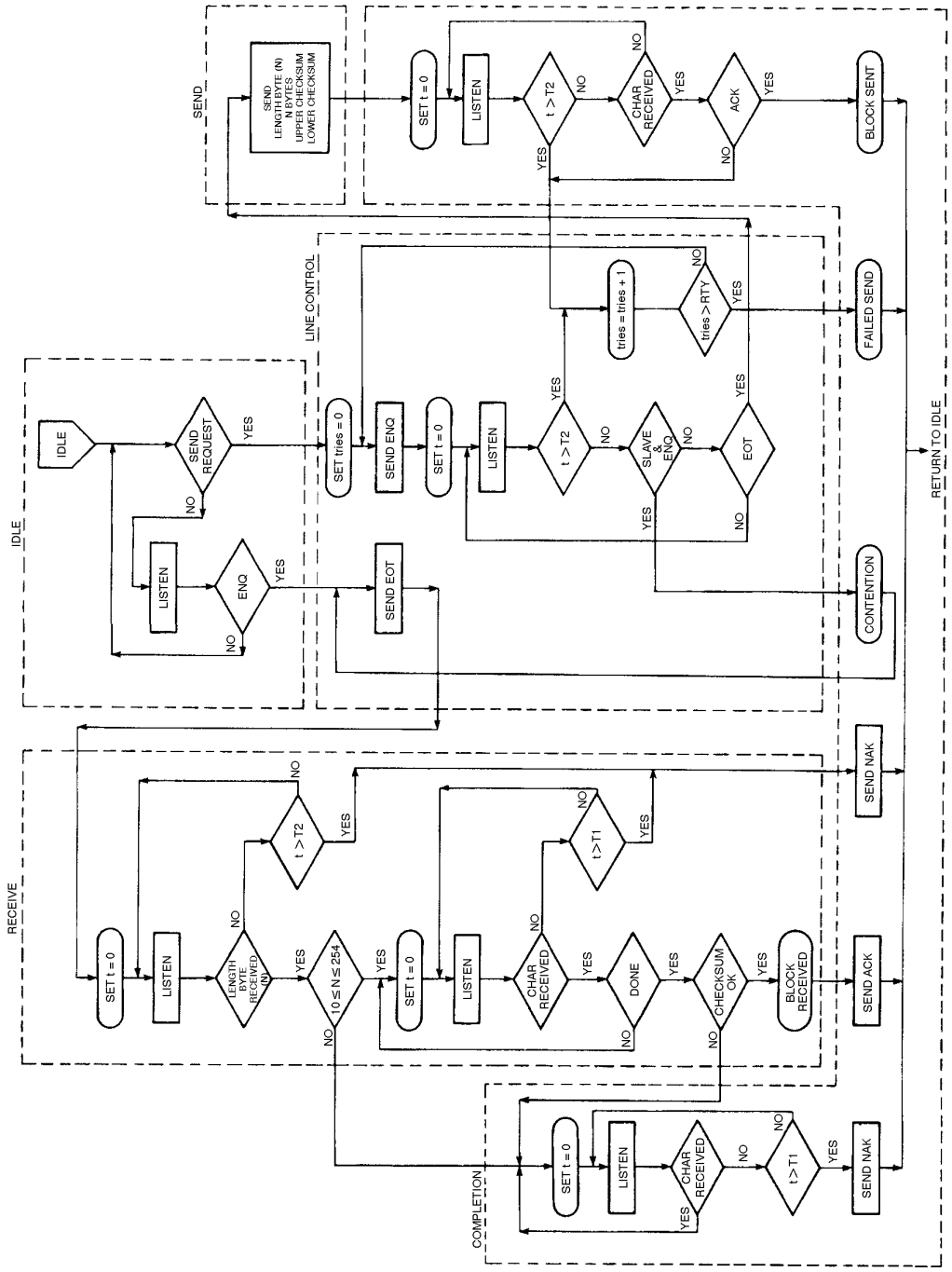

Figure 2

Block Transfer Protocol

6 Header Structure

6.1 The operation of all communications functions above the block transfer protocol is linked to information contained in a 10-byte data element called the header. The header is always the first 10 bytes of every block sent by the block transfer protocol. The information in the header is also used by the message protocol (see Section 7). The fixed format of the header is described in this section. The general header structure is shown in Figure 3.

NOTE: The header also contains information required by SECS-II.

6.2 *Reverse Bit (R-Bit)* — The reverse bit (R-bit) signifies the direction of a message. The R-bit is set to 0 for messages to the equipment and set to 1 for messages to the host. The R-bit is included in the header so that the direction of the message is contained in every block.

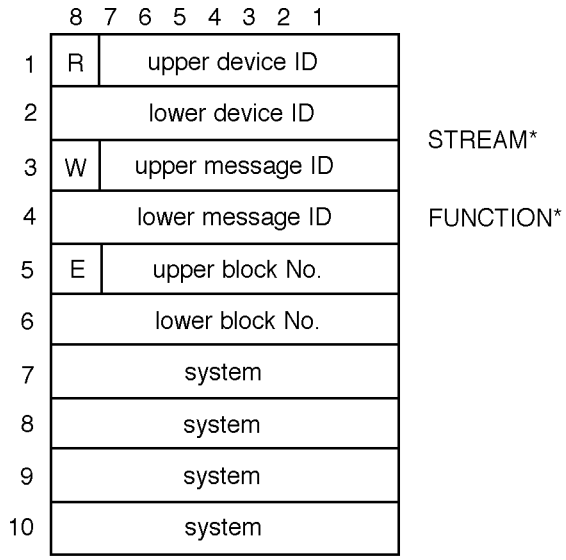

Figure 3

Block Header Structure

Table 3 Influence of the R-BIT

R-Bit	*Device ID*	*Message Direction*
0	Destination	Host to Equipment
1	Source	Equipment to Host

6.3 *Device ID* — The device ID defines the source or destination of the message depending upon the value of the R-bit as shown in Table 3. Device identification is a property of the equipment and must be settable according to Section 8. The host can view the device ID as a logical identifier connected with a physical device within the equipment. The host has no device ID.

6.4 *Wait Bit (W-Bit)* — The wait bit (W-bit) is used to indicate that the sender of a primary message expects a reply. A value of one in the W-bit means that a reply is expected. A value of zero in the W-bit means that no reply is expected. The W-bit must be set to zero in all secondary messages. For multi-block messages, the sender must ensure that the W-bit is the same in every block of the message.

6.5 *Message ID* — The message ID identifies the format and content of the message being sent (the particular message is one of many possible for the device in question). The exact message content is equipment-dependent. The upper message ID is the most significant portion of the ID.

6.5.1 *Primary Message* — A primary message is defined as any odd numbered message. An odd numbered message will have bit 1 of the lower message ID set to 1.

6.5.2 *Secondary Message* — A secondary message is defined as any even numbered message. An even-numbered message will have bit 1 of the lower message ID set to 0.

NOTE: In SECS-II, byte three of the header (excluding the W-bit) is known as the stream, and byte four of the header is known as the function. See SECS-II for more information.

6.6 *End Bit (E-Bit)* — The end bit (E-bit) is used to determine if a block is the last block of a message. A value of one in the E-bit means that the block is the last block. A value of zero means that more blocks are to follow.

6.7 *Block Number* — A message sent as more than one block is called a multi-block message. The first block is given a block number of one, and the block number is incremented by one for each subsequent block until the entire message is sent. The blocks of a multi-block message are sent in order. In a single-block message, the block number must have a value of zero or one. The maximum block number is 32,767. The upper block number is the most significant portion of the block number. (See also 7.2.)

6.8 *System Bytes* — The system bytes in the header of each message for a given device ID must satisfy the following requirements. (For a further discussion of system bytes, see Related Information R1-5.)

6.8.1 *Distinction* — The system bytes of a primary message must be distinct from those of all currently open transactions initiated from the same end of the communications link. They must also be distinct from those of the most recently completed transaction. (See Section 7.3.) They must also be distinct from any system bytes of blocks that were not successfully sent since the last successful block send.

6.8.2 *Reply Message* — The system bytes of the reply message are required to be the same as the system bytes of the corresponding primary message.

6.8.3 *Multi-Block Messages* — The system bytes of all blocks of a multi-block message must be the same.

7 Message Protocol

7.1 A message is a complete unit of communication in one direction. The message protocol uses the services of the block transfer protocol to send and receive messages. The message consists of the message data together with the following information from the header — R-bit, device ID, W-bit, message ID, and system bytes.

7.2 *Message Send* — When a message is ready to be sent, the message send protocol performs the functions described below. A block send failure terminates the message protocol action on that message.

7.2.1 *Message Length* — The maximum data length in a single block of a message is 244 bytes. The maximum number of blocks that can be sent in a multi-block message is 32,767, and so the maximum data length allowed in one message is 244 × 32,767 bytes.

7.2.2 *Message Blocking* — Message blocking is the division of the message data into blocks to be sent to the Block Transfer Protocol. For best performance, it is recommended, but not required, that the sender fill all blocks of a multi-block message, except possibly the last block, with the maximum 254 bytes. The receiver of a multi-block message should be able to accept any block size from 11 to 254 bytes, and should not require consecutive blocks necessarily to be the same size.

7.2.2.1 Certain older implementations may impose application-specific requirements on block sizes for certain incoming messages. Beginning with the 1988 revision of the standard, new applications may not impose application-specific requirements on incoming block sizes. Applications implemented before 1988 may impose such requirements.

NOTE: In SECS-II, certain messages are defined as single-block messages and must be sent as a single block in SECS-I.

7.2.3 *Header* — The message protocol must establish the header in each block of the message according to the requirements of Section 6.

7.2.4 *Interleaving Messages* — This standard allows, but does not require, the support of more than one concurrent open transaction. This standard allows, but does not require, the support of interleaving the blocks of different multi-block messages. (See documentation requirements in 9.)

7.3 *Transactions* — A transaction is a primary message and an optional corresponding secondary message is called the reply. A transaction is opened when a primary message is ready to be sent. A transaction is closed when the last block of a primary message requesting no reply has been sent, or when the last block of the reply has been received.

7.3.1 *Reply Linking* — When a reply is expected for a primary message, the message protocol starts the reply timer for the transaction after the last block of the message is successfully sent. When a primary message is sent for which a reply is requested, an expected block is established for the message receive algorithm. (See Section 7.4.) The expected block will have the complement of the R-bit, will have the same device ID, will be the first block of a secondary message, and will have the same system bytes as those of the given primary message.

NOTE: In SECS-II, the reply will have the same upper message ID (stream), and the lower message ID (function) will either be one greater than that of the corresponding primary message, or it will be zero.

7.3.2 *Reply Timeout, T3* — The reply timeout, T3, is a limit on the length of time that the message protocol is willing to wait after the last block of a primary message has been sent and before the arrival of the first block of the reply. If the first block of the reply does not arrive within the T3 limit, the expected block is removed from the list of expected blocks, and the transaction is aborted. A timer, called the reply timer, is used to measure the time between the last block of the primary message and the first block of its reply. Each open transaction for which a reply is expected requires a separate reply timer.

7.4 *Message Receive* — Each block successfully received by the block transfer protocol is passed to the message protocol. It is the task of the message protocol to identify the blocks and assemble them into the proper message.

7.4.1 *Routing Error* — When a piece of equipment receives a block of data which has a device ID in the block header which does not match its own device ID and it has no other knowledge of this device ID, it can assume that the block was sent in error.

7.4.2 *Duplicate Block Detection* — A duplicate block is a block which is exactly the same as the previous block received by the Block Transfer Protocol. This may occur when the receiver has sent an ACK but, for some reason, the ACK did not arrive in time at the sender, causing a send retry. A duplicate block is detected by the SECS-I message protocol by comparing the full 10-byte header of a block currently received by the block transfer protocol with the header of the last block accepted as non-duplicate by the message protocol. If the headers are identical, the new block is a duplicate and should be discarded. If the headers are different, the new block is a non-duplicate. The header of the non-duplicate block saved for comparison with the next block passed on by the block transfer protocol, and the block containing the header is further processed by the message receive algorithm.

NOTE: Some implementations which follow the 1980 version of SECS-I may not provide the unique headers required for duplicate block detection. An option for disabling the duplicate block detection is required to be compatible with these systems.

7.4.3 *Inter-Block Timeout T4* — The time interval between the successful receipt of a block in a multiblock message, and the successful receipt of the subsequent block of the same message, is limited to time T4. If this time is exceeded, the message is cancelled and the transaction is aborted. A time called the inter-block timer is used to measure the time between block arrivals in the message protocol. There must be one inter-block timer for each open multi-block message currently being received by the protocol. As each successive block of a message is received, the corresponding inter-block timer is reset.

7.4.4 *Algorithm* — When a block arrives at the message protocol, a combination of all the bytes in the header is used to determine what to do with the block. The operation of the message receive algorithm is to be understood by following the logic flow in Figure 4. The flow chart shown in Figure 4 and the description of the algorithm below are not meant to imply that particular implementation is required under this standard. However, any SECS-I message protocol implementation must include all the logic shown in Figure 4 on page 11.

7.4.4.1 The description of the message protocol uses the concepts of an expected block. When a (properly routed and non-duplicate) block is received by the message protocol, the first determination is whether the block is one of the expected blocks or not. In order to determine if a block is one of the expected blocks, the header information is compared with the header information in a list of expected blocks.

7.4.4.2 If the block is not one of the expected blocks, it must be the first block of a primary message, otherwise the block has been sent in error and can be discarded. If the block is the first block of a primary message, and it is not the last block of the message (E-bit = 0), an inter-block timer for the given message is established and set, and the expected block for the given message is set to have the same R-bit, device ID, system bytes, W-bit, message ID, and a block number one greater than that of the block just received.

7.4.4.3 If the block is one of the expected blocks, then it is either the first block of a reply message or it is part of an open message.

7.4.4.4 If the block is the first block of a reply message, the reply timer for the given message is cancelled. For the first block of a reply message, the expected block will have block number one (or possibly zero if the reply is a single block message) and will be a secondary message, but the full message ID is undetermined. (See Section 7.3.1.)

7.4.4.5 If the block is the last block of the given message (E-bit = 1), the message is complete. If the message is a primary message for which a reply is requested (W-bit = 1), the system bytes are saved for sending with the reply message. (See Section 7.2.3.)

7.4.4.6 If the block is not the last block of the message (E-bit = 0), then the inter-block timer for the given message is reset, and the (next) expected block for the given message is set to have the same R-bit, device ID, system bytes, W-bit, message ID, and a block number one greater than that of the block just received.

8 Parameter Setting

8.1 The eight protocol parameters are listed in Table 4. The selection of baud rate should be based on system performance. The value of the device ID is determined by the particular system requirements and is generally unique within one factory. The values of the next five parameters are determined by the performance characteristics of the host system and the baud rate of the communication channel. The first seven parameters must be adjustable by the user. All parameters must be stored in such a manner that the settings will be retained if the power fails or if the system software is reloaded. The range and resolution of protocol parameters must be at least as shown in Table 4. The M/S parameter is set to master in the equipment and to slave in the host.

Table 4 Protocol Parameters

Symbol	*Parameter Name*	*Typical Function*	*Typical Value*	*Range*	*Resolution*
BAUD	Baud Rate	Sets serial line speed	9600	300 - 9600	see Section 3.3
DEVID	Device ID	Identifier assigned to the equipment	—	0 - 32767	1
T1	Inter-Character Timeout	Detects an interruption between characters	0.5 sec.	0.1-10 sec.	0.1 sec.
T2	Protocol Timeout	Detects a lack of protocol response	10 sec.	0.2-25 sec.	0.2 sec.
T3	Reply Timeout	Detects a lack of reply message	45 sec.	1-120 sec.	1 sec.
T4	Inter-Block Timeout	Detects an interruption in a multi-block message	45 sec.	1-120 sec.	1 sec.
RTY	Retry Limit	The maximum number of send retries allowed	3	0 - 31	1
M/S	Master Slave	Contention resolution	—	—	—

9 Documentation

9.1 For equipment or host to comply with SECS-I, a document is required containing the following information. (See also SEMI E6, Facilities Interface Specifications Guideline and Format.)

1. Method for setting all the parameters in Table 4.
2. Range allowed and resolution for each parameter in Table 4.
3. Compatibility with duplicate block detection and the method for enabling and disabling the same if present (see Section 7.4.2).
4. Maximum expected inter-character, protocol, reply, and inter-block delays generated under normal operating conditions.
5. Whether multi-block messages are supported as a receiver.
6. Whether multi-block messages are used as a sender.
7. Whether there is a limit to the size of a message received and, if so, what the limit is.
8. Maximum expected size of a message being sent.
9. Whether message interleaving is supported as a receiver.
10. Whether message interleaving is used as a sender.
11. Number of device ID's supported on the port.
12. Maximum number of supported concurrent open transactions.

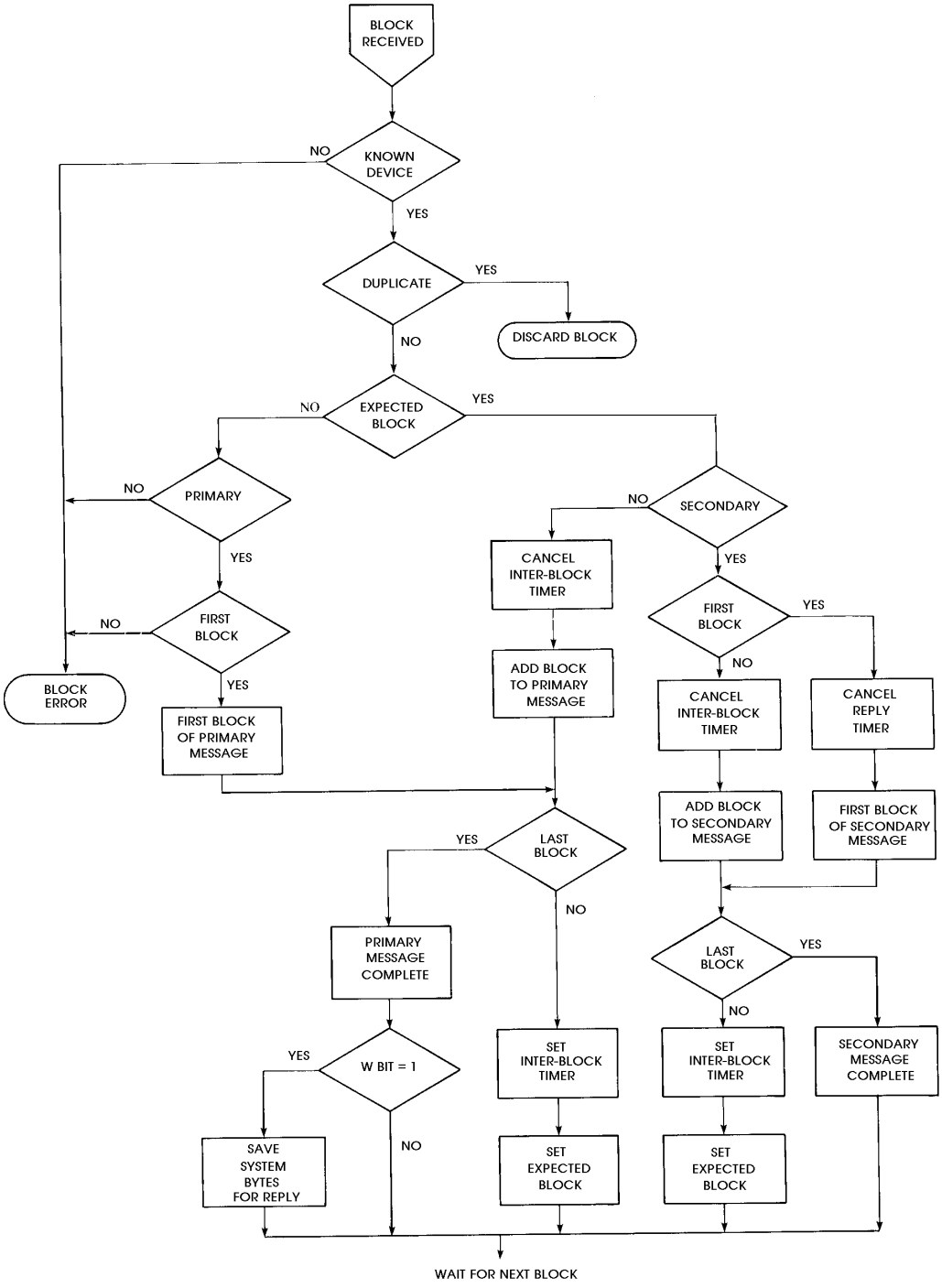

Figure 4

Message Receive Algorithm

APPENDIX 1

A1-1 Differences from SECS-I 1980

This appendix describes the major differences between this version of the standard and the version originally adopted in 1980.

A1-1.1 *Signal Connections* — The required voltage levels for pins 18 and 25 on the 25 pin "D" connector are now optional.

A1-1.2 *Data Rate* — The 150 baud data rate has been made optional.

A1-1.3 *Timeout Parameters* — The T4 timeout, which limits the inter-block arrival time of multi-block messages, has been added. Use of the T3 timeout has been clarified.

A1-1.4 *Time Before Length Byte* — The protocol time with limit T2 is now used while waiting for the length byte after sending an EOT.

A1-1.5 *Block Send Acknowledgment* — Any character other than an ACK received after sending the second checksum byte is treated as an NAK.

A1-1.6 *Illegal Block Lengths and Bad Checksums* — An NAK code is sent for an illegal block length or for a bad checksum, but only after waiting for the sender to stop sending by forcing an inter-character (T1) timeout.

A1-1.7 *Block Number* — The wording describing the block number has been clarified to say that the value, zero, is allowed only for single block messages.

A1-1.8 *System Bytes* — The handling of the system bytes for secondary messages sent by a host is now the same as for the equipment. Also, specific functional requirements have been added for the content of the system bytes field.

A1-1.9 *Duplicate Blocks* — A mechanism for the detection of duplicate blocks has been added.

A1-1.10 *Messages* — The discussion of message assembly from blocks has been clarified and expanded.

A1-1.11 *Transactions* — The discussion of transaction handling and reply linking has been moved from SECS-II and expanded.

A1-1.12 *Appendix* — Parts of the Appendix have been moved to a new section called Related Information to distinguish the content from the standard itself. The general node transaction flow chart has been moved from the Appendix in SECS-II to the Related Information in SECS-I.

A1-1.13 *Titles* — The title of SECS-I has been changed from "Data Link" to "Message Transfer." This phrase more accurately covers the content of the standard and avoids confusion with other uses of the term "data link." The title of the "Data Link" section of SECS-I has been changed to "Block Transfer." The "Data Link Control" portion of the Block Transfer Protocol has been retitled "Line Control."

A1-1.14 *W-Bit* — The W-bit is now required to be set consistently in all blocks.

A1-1.15 *Documentation* — A section has been added on the documentation required for compliance with the standard.

RELATED INFORMATION

NOTICE — The material contained in this Related Information is not an official part of SEMI E4 (SECS-I) and is not intended to modify or supercede the official standard. Rather, this information describes possible methods for implementing the protocol described by the standard and are included as reference material. The standard should be referred to in all cases. SEMI makes no warranties or representations as to the suitability of the material set forth herein for any particular application. The determination of the suitability of the material is solely the responsibility of the user.

R1-1 Miscellaneous Notes on SECS-I

R1-1.1 *Layered Protocol (see Section 1.3.1)* — The lnternational Standards Organization (ISO) has published a model for Open Systems lnterconnection (OSI). The SECS-I protocol both predates the ISO/OSI model and is not a true "open system" and, therefore, does not correspond exactly to the ISO/OSI model. The SECS-I protocol is a communications interface rather than a network protocol. However, the SECS-I levels can be roughly compared to layers 1 through part of layer 5 of the ISO/OSI model. ISO/OSI layer 1, the physical link layer, corresponds to the SECS-I physical link. ISO/OSI layer 2, the data link layer, corresponds to the SECS-I block transfer protocol. lSO/OSI layer 3, the network layer, is a function of the host and is not defined in SECS-I beyond the provision for a bi-directional flow (see Section 6.2). Similarly, network management is assumed to be the responsbility of the host. ISO/OSI layer 4, the transport layer, is covered by the SECS-I block transfer protocol, duplicate block detection, and the message protocol. ISO/OSI layer 5, the session layer, is partially covered by the SECS-I message protocol.

R1-1.2 *Single Timer for T1 and T2 (see Section 5.3)* — A single timer can be used for both the inter-character timer and the protocol timer, since both are the time between receiving successive characters and both limits are never in effect at the same time.

R1-1.3 *Stalling (see Section 5.8.2)* — The line control portion of the block transfer protocol has the ability to delay the acceptance of a data block by not responding with an EOT immediately after receiving an ENQ. Such an action by the receiver is called "Stalling." If the delay exceeds the sender's T2 value, the sender will send another ENQ. This can be continued depending upon the sender's setting of T2 and RTY. Such a delay should be an occasional convenience to accommodate random short delays in the receiver's ability to accept a new block and should not be counted upon routinely to make up for poor response. In particular, the block transfer protocol should probably have at least two buffers available for storing incoming blocks. This allows one block to be inspected by the message protocol while the next block is being received, thus allowing a reasonably continuous reception of data. If both buffers are full and the message protocol is slow in freeing the buffer for the block transfer protocol, then the block transfer protocol will start stalling the sender. If the sender is stalled long enough, it will declare a send error, which is probably the correct thing to do. The sender cannot distinguish between a reluctance to receive and failure in the communications. In either situation, since no block gets through in a predetermined time limit, the communications link is effectively broken. Arbitrarily long delays are not acceptable in SECS-I.

R1-1.4 *Determining the Cause of an NAK (see Section 5.8.5)* —The block transfer protocol does not include a way for a sender to determine the cause of an NAK. If such information is useful for application purposes, it must be collected at the receiving end.

R1-1.5 *Master Sending a Long Message (see Section 5.8)* —When the master is sending a long message, the slave may be unable to send a block, which may result in timeouts. It is good practice for the master to introduce enough delay between blocks so that the slave has a chance to send a block every few seconds.

R1-1.6 *Device Identification (see Section 6.3)* — Although the 15 bits of the device ID can identify 32,767 different devices, the host may find it more convenient to use the upper seven bits to identify the type of device such as a spinner or diffusion furnace, and to use the lower eight bits to identify the specific device of that type.

R1-1.7 *Sending Multiple Open Messages (see Section 7.2.4)* —The message protocol algorithms defined in the standard are capable of inter-leaving the blocks of any number of open multi-block messages. Since the receiving protocol is sensitive to the time between blocks of each message being received, it is proper procedure for the sending algorithm to alternate between messages when sending blocks from interleaved multi-block messages.

R1-1.8 *Single Timer for T3 and T4 (see Sections 7.3.2 and 7.4.3)* — For a given transaction, a single timer can be used for both the inter-block timer and the reply timer, since both are the time between receiving successive blocks of a message, and both limits are never in effect at the same time.

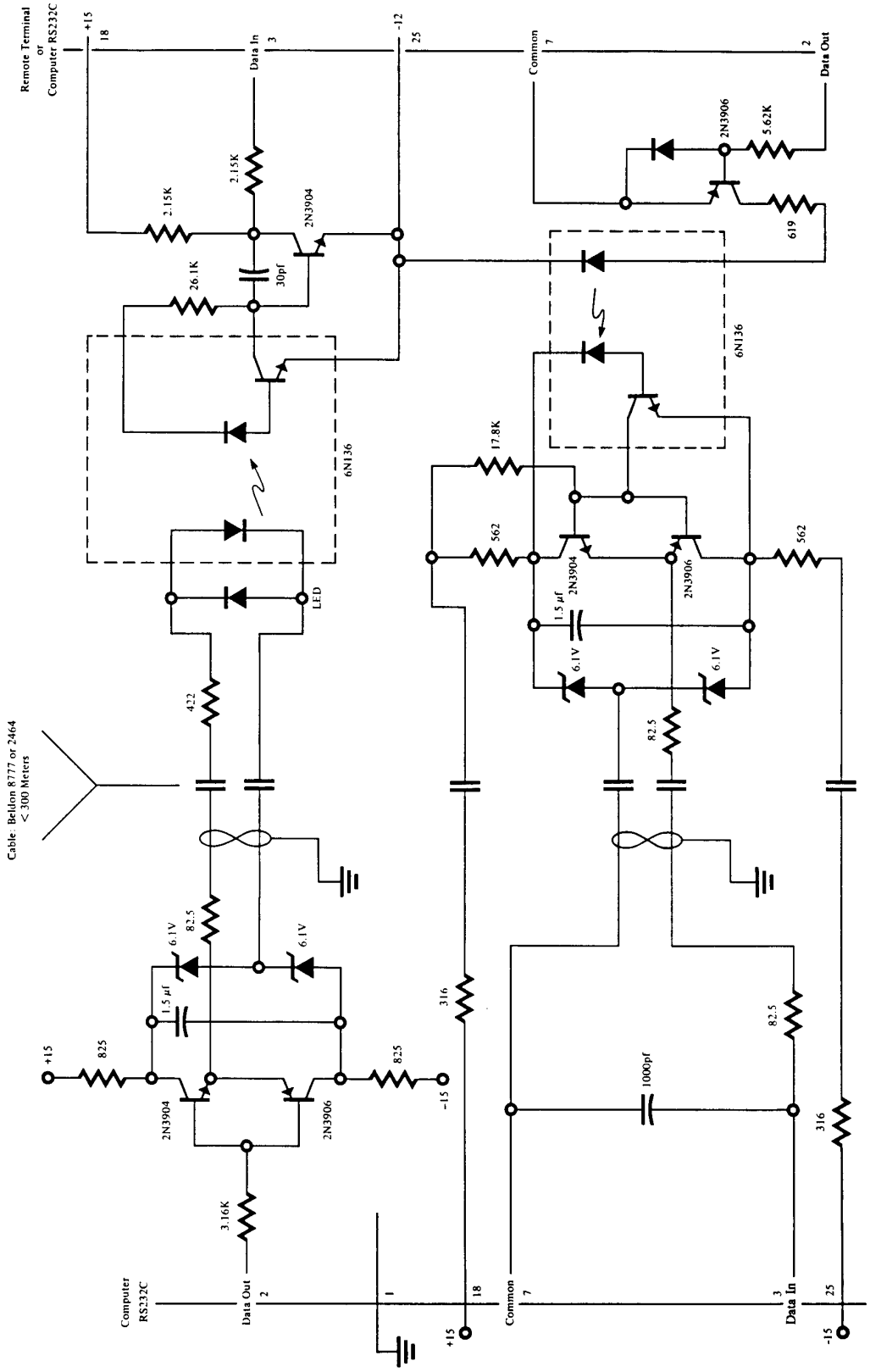

Figure R1-1

RS-232 Isolation Example

R1-2 Isolation

R1-2.1 As mentioned in Section 3.4, the standard RS-232-C line does not provide isolation between the grounds of remote equipment and a central computer. It is often desirable to provide isolation to minimize communications errors and to protect equipment. The following example is provided here to illustrate how such an isolation might be provided between two devices using RS-232-C, while at the same time providing the capability to drive longer lines. This description is provided for information only and is not part of the standard.

R1-2.2 Figure R1-1 illustrates a possible circuit for isolation which provides line isolation and line driving capability. The two wires used to provide power to the remote isolator are required since any local power supply would have the same isolation problems as the data wires. The power wires also eliminate extra supplies at every terminal or remote device. There are many other possible circuits. This example has been used successfully and works quite well at 9600 baud for lines less than 300 meters and illustrates the use of the voltages on pins 18 and 25 as mentioned in 3.2.4.

R1-3 Examples of Block Transfer Protocol

R1-3.1 Figures R1-2 and R1-3 illustrate some simple message interactions between the host and the equipment. Figure R1-2 shows the handshake sequence possible to acquire the status of the equipment. Fig-ure R1-3 shows the sequence of events when both the host and the equipment try to send at the same time.

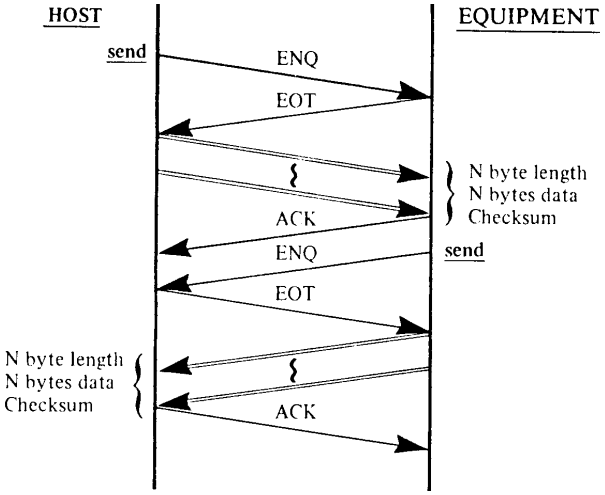

Figure R1-2

Ask for Status

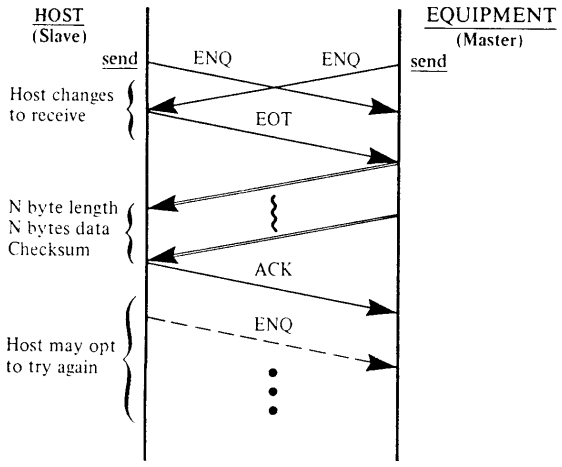

Figure R1-3

Contention Resolution

R1-4 Use of Buffers in the Block Transfer Protocol

R1-4.1 In the flow chart of the Block Transfer Protocol (Figure 2), the flow chart entry "LISTEN" implies that characters are recognized after entering the given state and not before. If the particular implementation uses buffers and stores all of the characters received before the "LISTEN" state is achieved, care must be taken in the proper handling of the data in the buffer to avoid problems due to "timing windows" between sending and receiving.

R1-4.2 If the buffer is cleared before sending the last character prior to the "LISTEN" (last checksum byte in sending a message, ENQ or EOT in line control), there is still a time, albeit small, where spurious characters could be received into the buffer.

R1-4.2.1 In addition, during the sending of the message, a delay between the checksum bytes while the buffer is cleared may cause an inter-character timeout by the receiver. If the buffer is cleared after the last character sent before the "LISTEN," there is a possibility that the desired character will be cleared from the buffer.

R1-4.3 One solution is to look at the buffer after entering the "LISTEN" state and examine the last character in the buffer, if any are there. This works while listening for an EOT after sending an ENQ. In the case where the sender is listening for an ACK after sending the second checksum byte, it is possible that the receiver also has a block to send, and follows the

ACK immediately with an ENQ. In this case the last character in the buffer might be an ENQ, and so the last two characters should be examined. If an ACK precedes the ENQ, then it should be assumed that the block was sent successfully, and the ENQ should be saved for establishing line control (i.e., send an EOT, or an ENQ if there is contention).

R1-4.4 The handling of the buffer when listening for the length byte is not improved by this suggestion. The buffer should be cleared after the ENQ is received and before the EOT is sent. Spurious characters received before the length byte will cause a send failure.

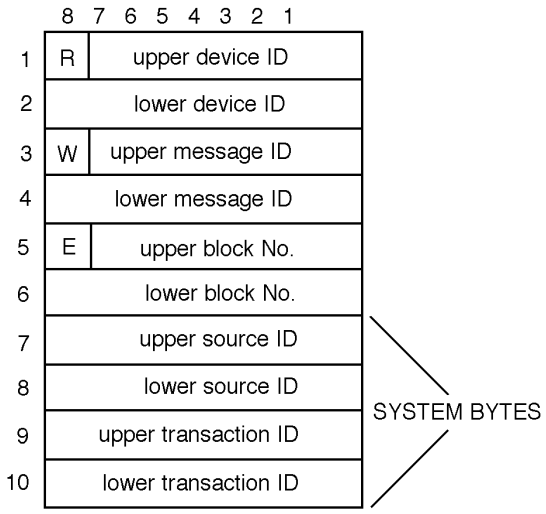

Figure R1-4

Possible Bock Header Structure

R1-5 System Bytes

R1-5.1 This section presents a sample scheme for generating and managing the system bytes field of the block header. There are many possible implementations which meet the requirements of SECS-I. This section describes one of those implementations.

R1-5.2 *Source ID and Transaction ID* — The system bytes are divided into two parts, a source ID and a transaction ID, as shown in Figure R1-4.

R1-5.3 Distinct source IDs are assigned to each application level originator of primary messages in a host or equipment. This allows a secondary message to be routed to the source of the corresponding primary message of the transaction. The transaction ID is an integer that is incremented for each primary message sent by the SECS-I interface. It is the same for all blocks of a multi-block message.

R1-5.4 *Actions* — When a SECS-II message is passed to the SECS-I protocol, the least significant bit of the lower message ID is examined to determine whether the message is primary (bit 1 = 1) or secondary (bit 1 = 0) message. If the message is primary, transaction ID is generated and placed in the transaction ID portion of the system bytes. One method for generating the transaction ID is to start with a value of 0 upon initialization, and increment by 1 for each new transaction, starting with 1 (not 0) when the largest transaction ID value has been used. At the same time, the application sender's source ID is placed in the source ID portion of the system bytes. If the message is secondary, the system bytes are the same as those of the corresponding primary message. It is assumed that the application generating the reply can identify the reply to the message protocol, which has saved the system bytes from the primary message.

R1-5.5 *Block Send Failures (see Sections 5.8.2 and 6.8.1)* — Since it is a requirement that the system bytes be distinct from those used in blocks that were not successfully sent since the last successful block send, the use of two bytes for the transaction ID would effectively satisfy the requirement by allowing for up to 65,536 consecutive block send failures.

R1-6 Using SECS in a Network

R1-6.1 The SECS standard can be used to control the flow of data within a network of computers. A network is an interconnection of computers or intelligent controllers communicating with one another over communications lines. Each intelligent entity is a node of the network. Each node may have a number of connections to other members of the network.

R1-6.2 One connection of equipment and computers is in the form of a tree network with the processing equipment at the ends of branches as shown in Figure R1-5. Intermediate Nodes A and B service like pieces of equipment; i.e., all the furnaces to Node A, all masking to Node B, and so on, for more nodes.

R1-6.3 Node A monitors the three stations 1A, 2A, and 3A. Node A handles most of the normal requirements of coordinating 1A, 2A, and 3A. Node A receives instructions from Node C regarding the behavior desired from 1A, 2A, and 3A. A CRT or operator interface connected to Node A is shared by 1A, 2A, and 3A. Any node can direct messages up or down the tree. A message such as an alarm can be sent from 1A to Node A and from Node A to Node C which might be considered the host of the network. Node A might be part of a system supplied by a company which has its own communications scheme for devices tied to Node A. As long as the company provides one connection to Node A which is consistent with this standard, then

Node A and its attached equipment can be connected to the host, Node C, and collectively be called one node.

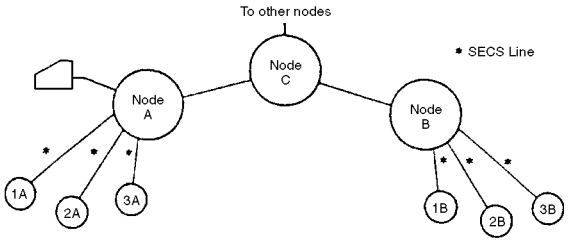

Figure R1-5

Tree Network

R1-6.4 This tree structure can be extended even further as shown in Figure R1-6 to include multiple manufacturing plants at remote sites. However, the primary intent of this standard is to address the communication between processing equipment and a host. At higher levels of the tree, the message is combined with more system level messages and more computer file manipulations. At these levels, the communications may require more complex or higher speed standards. Thus, the SECS line is intended to be used within one integrated circuit manufacturing area rather that between general purpose computer centers, although its only limitations for such a use would be speed.

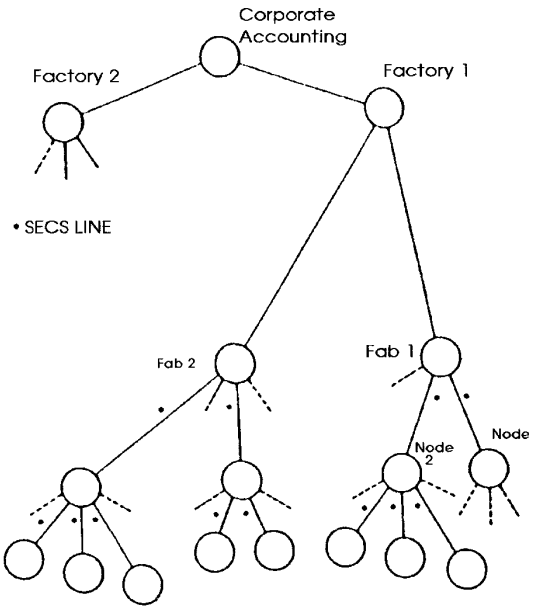

Figure R1-6

Extended Tree Network for a Large Organization

R1-6.5 The tree network is characterized by the fact that there is only one possible path between any two nodes in the network. This feature makes handling messages in the tree network relatively easy when compared to networks where more than one communication path exists between nodes. In these latter networks, the nodes must make some judgment about which path to take, perhaps based on the conditions of the lines or on the destination of the message.

R1-6.6 The device ID and the R (Reply) bit play an important role in directing messages in the network. Each node of the three maintains a table of the device below it. From the device code, it knows on which branch to send the data. When a message is going from the central node toward the device, the R-bit is set to 0. When a message is being sent from the device up the tree, the R-bit is set to 1. When a node sees the R-bit set to 1, the message must take the one communication direction that goes up the tree structure. Thus, the R-bit serves to direct messages up the tree or down the tree, depending on its value. At any given node, this behavior may be explained in Figure R1-7.

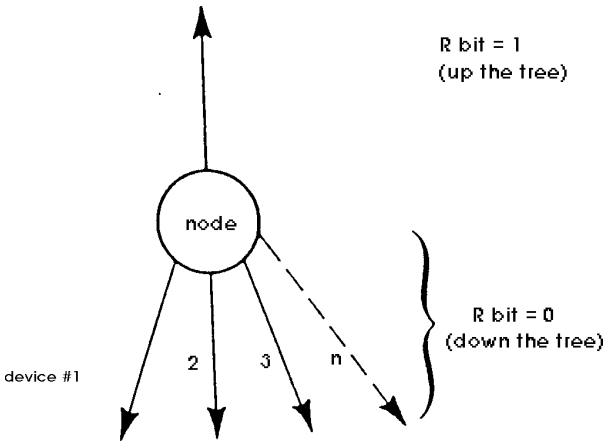

Figure R1-7

Message Handling at a Node by Using the R-Bit

R1-7 The General Node Transaction Protocol

R1-7.1 The transaction protocol has three primary functions. First, it coordinates the receiving of messages that arrive in multiple blocks. Second, it provides a mechanism for matching primary and secondary messages. Third, it does error detection on incomplete messages whether they are due to a break during multiple blocks or a lack of reply messages. The third function is vital to the maintenance of a working system, as can be seen from the following discussion. Once a transaction is begun with the first block of a primary message, the receiving machine will have some memory allocated to buffering or pointing to interpreting modules. If the transaction remains incomplete for some reasonable length of time, it may

be assumed that an error has occurred and the receiving machine should release any memory it may be holding for the transaction. If memory were to remain allocated to incomplete transactions, conceivably the receiving machine could run out of available memory and would be unable to function properly. For this reason, the error detection of incomplete transactions is an essential part of the message transaction protocol in SECS-I.

R1-7.2 The message transaction protocol is based on a system of timers called transaction timers. These timers are maintained in software and are associated with a given transaction. Once a transaction is begun, there will exist a timer on the time interval between all blocks of the transaction independent of the direction of the message. The machine that expects to receive a block has the burden of timing and error detection. Timers are required in any machine that (1) makes a request and expects to receive data back; (2) must receive multiple block requests; or (3) sends data and expects an acknowledgement. A machine may require several transaction timers if it can conduct multiple transactions at one time on one port or has multiple ports. A message timer will exist for each link involved in the transaction. This allows the error detection to identify the particular link which has failed.

R1-7.3 A general algorithm can be constructed to handle an arbitrary number of SECS ports. Such an algorithm is presented in the form of a flow chart in Figure R1-8. There is one such algorithm for each device ID. The algorithm handles all SECS data blocks that enter, leave or pass the device ID. A new block causes the procedure to be executed starting from the point marked "block in" and continues until the point marked "block out." The block of data will then be directed to the proper destination.

R1-7.4 This algorithm makes use of a stored version of the message header for each transaction being handled. The stored header may be visualized as shown in Figure R1-9. The stored header has two system areas and an associated transaction timer. The two system areas are required to keep the incoming and the outgoing system areas independent in coding, yet related in meaning. The two system areas are referred to by the state of the R-bit in the message block header. One area is related to R = 0 and the other R = l. When a prime message is received that requires a reply, the system bytes from the block are stored in the system area corresponding to the opposite state of the R bit on the message. When the reply is sent, the system bytes are used from the store system bytes corresponding to the same R-bit as the reply. The system area corresponding to the same R-bit as the message is called OUTSYS while the system area corresponding to the opposite R-bit is called INSYS.

R1-7.5 When a block is presented to this algorithm, the first task is to determine if the block is part of an ongoing transaction, the first block of a new transaction, or some other unknown block. This test is accomplished by comparing the header of the block with the stored headers. If a match is not found, then a check is made to see if the block is the first of a primary message. If this is true, then a new header is added to the stored header list. If the primary message is only one block long and requires no reply, then no stored header is created. When a match is found, the algorithm modifies the stored header so that it will look like the next block of the transaction to be expected. On all blocks requiring a stored header, except the last, a transaction timer is set to a timeout value prior to sending the block to its destination. The last part of the algorithm directs the block to its destination or detects an error in the destination.

R1-7.6 Error recovery is an important part of the algorithm to ensure that the system will remain operational in spite of bad data blocks. For any block that does not pass the tests described in the algorithm, the header of the message is saved and sent in the host direction on a Stream 9 error message. This information can be useful in tracking down the source of the error. Another type of error occurs when one of the transaction timers times out. Such an error message means that the transaction has been interrupted. An error message is formed which sends the stored header in the host direction on Stream 9. The stored header and the timer are then released. This procedure aids in keeping software cleared and ready in case of difficulties in the network.

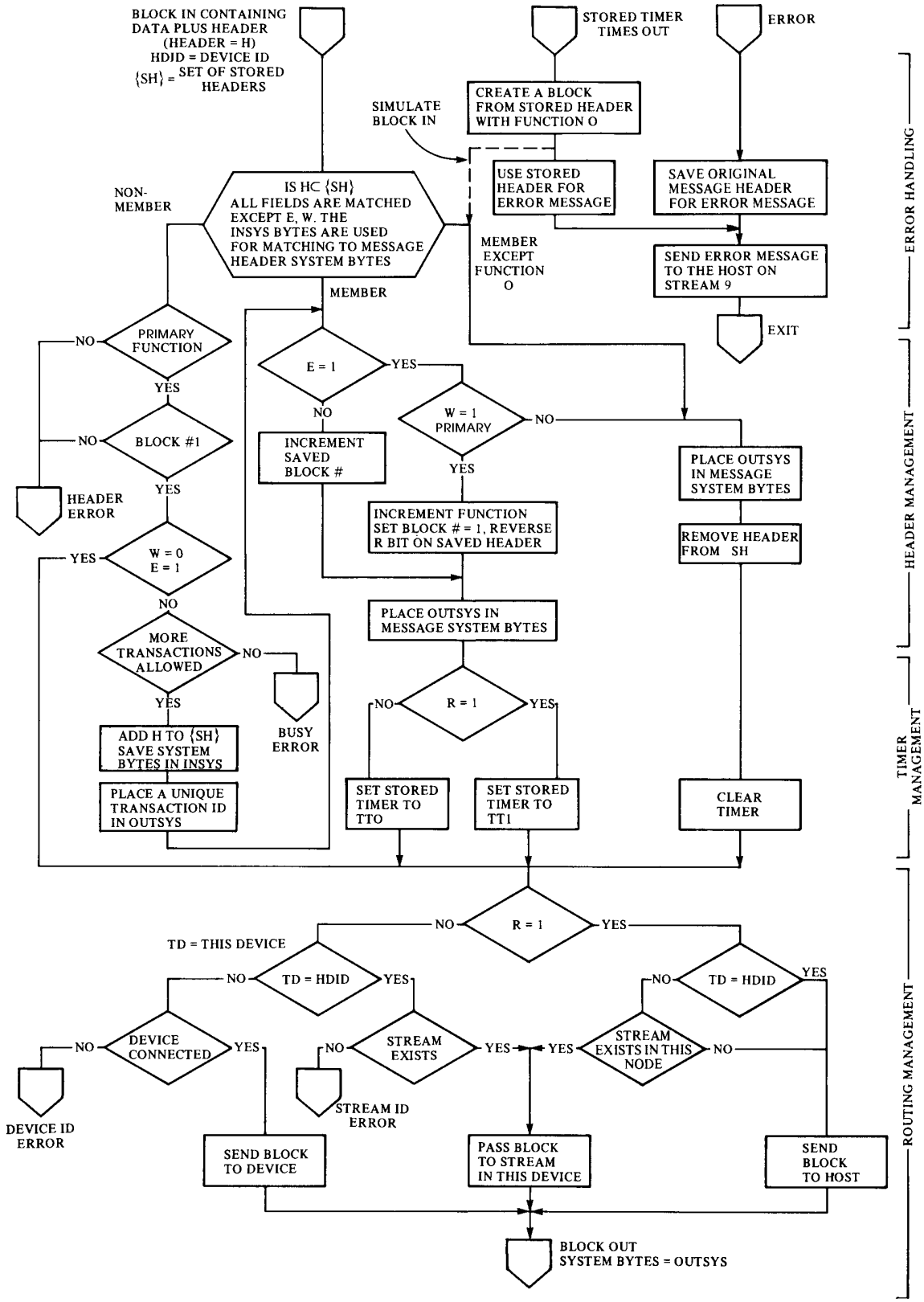

Figure R1-8

General Node Transaction Algorithm

R	Device ID
	Device ID
W	STREAM
	FUNCTION
E	Block No.
	Block No.
	S1(R=0)
	S2(R=0)
	S3(R=0)
	S4(R=0)
	S1(R=1)
	S2(R=1)
	S3(R=1)
	S4(R=1)
Transaction Timer	

Figure R1-9

Stored Header Model

NOTICE: These standards do not purport to address safety issues, if any, associated with their use. It is the responsibility of the user of these standards to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use. SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.