



E5-85

SEMI EQUIPMENT COMMUNICATIONS STANDARD 2 TRANSACTION (SECS-II) (PROPOSED)

CONTENTS

1. Introduction	S2,F3	Service Program Send (SPS)
1.1 Intent	S2,F4	Service Program Send Acknowledge (SPA)
1.2 Overview	S2,F5	Service Program Load Request (SPR)
1.3 Application	S2,F6	Service Program Load Data (SPD)
1.4 Applicable Documents	S2,F7	Service Program Run Send (CSS)
2. Message Structure	S2,F8	Service Program Run Acknowledge (CSA)
2.1 Intent	S2,F9	Service Program Results Request (SRR)
2.2 Message and Blocks	S2,F10	Service Program Results Data (SRD)
2.3 Block Length and Message Length	S2,F11	Service Program Directory Request (SDR)
2.4 Stream and Function Definition	S2,F12	Service Program Directory Data (SDD)
2.5 Stream and Function Allocation	S2,F13	Equipment Constant Request (ECR)
2.6 Understanding the Message	S2,F14	Equipment Constant Data (ECD)
3. Message Transaction and Conversation Protocol	S2,F15	New Equipment Constant Send (ECS)
3.1 Intent	S2,F16	New Equipment Constant Acknowledge (ECA)
3.2 Transaction Definition	S2,F17	Date and Time Request (DTR)
3.3 Function Code Restrictions	S2,F18	Date and Time Data (DTD)
3.4 Transaction Protocol Requirements	S2,F19	Reset/Initialize Send (RIS)
3.5 Conversation Protocols	S2,F20	Reset Acknowledge (RIA)
4. Data Structures	S2,F21	Remote Command Send (RCS)
4.1 Intent	S2,F22	Remote Command Acknowledge (RCA)
4.2 Item	S2,F23	Trace Initialize Send (TIS)
4.3 List	S2,F24	Trace Initialize Acknowledge (TIA)
4.4 Example Data Structures	S2,F25	Loopback Diagnostic Request (LDR)
5. Message Detail	S2,F26	Loopback Diagnostic Data (LDD)
5.1-5.3 Intent	S2,F27	Initiate Processing Request (IPR)
5.4 Stream 0 and Function 0	S2,F28	Initiate Processing Acknowledge (IPA)
5.5 Stream 1 Equipment Status	5.7 Stream 3 Material Status	
S1,F0 Abort Transaction	S3,F0	Abort Transaction
S1,F1 Are You There Request (R)	S3,F1	Material Status Request (MSR)
S1,F2 On Line Data (D)	S3,F2	Material Status Data (MSD)
S1,F3 Selected Equipment Status Request (SSR)	S3,F3	Time to Completion Request (TCR)
S1,F4 Selected Equipment Status Data (SSD)	S3,F4	Time to Completion Data (TCD)
S1,F5 Formatted Status Request (FSR)	S3,F5	Material Found Send (MFS)
S1,F6 Formatted Status Data (FSD)	S3,F6	Material Found Acknowledge (MFA)
S1,F7 Fixed Form Request (FFR)	S3,F7	Material Lost Send (MLS)
S1,F8 Fixed Form Data (FFD)	S3,F8	Material Lost Acknowledge (MLA)
S1,F9 Material Transfer Status Request (TSR)	S3,F9	Material ID Equate Send (IES)
S1,F10 Material Transfer Status Data (TSD)	S3,F10	Material ID Equate Acknowledge (IEA)
5.6 Stream 2 Equipment Control and Diagnostics	5.8 Stream 4 Material Control	
S2,F0 Abort Transaction		
S2,F1 Service Program Load Inquire (SPI)		
S2,F2 Service Program Load Grant (SPG)		



S4,F0	Abort Transaction	S7,F14	Delete M/P M Entry Acknowledge (DEA)
S4,F1	Ready to Send Material (RSN)	S7,F15	Matrix Mode Select Send (MMS)
S4,F2	Not Used	S7,F16	Matrix Mode Select Acknowledge (MMA)
S4,F3	Send Material (SMN)	S7,F17	Delete Process Program Send (DPS)
S4,F4	Not Used	S7,F18	Delete Process Program Acknowledge (DPA)
S4,F5	Handshake Complete (HCN)	S7,F19	Current EPPD Request (RER)
S4,F6	Not Used	S7,F20	Current EPPD Data (RED)
S4,F7	Not Ready to Send (ABN)	S7,F21	Equipment Process Capabilities Request (PCR)
S4,F8	Not Used	S7,F22	Equipment Process Capabilities Data (PCD)
S4,F9	Stuck in Sender (SSN)	S7,F23	Formatted Process Program Send (FPS)
S4,F10	Not Used	S7,F24	Formatted Process Program Acknowledge (FPA)
S4,F11	Stuck in Receiver (SRN)	S7,F25	Formatted Process Program Request (FPR)
S4,F12	Not Used	S7,F26	Formatted Process Program Data (FPD)
S4,F13	Send Incomplete Timeout (SIN)	S7,F27	Process Program Verification Send (PVS)
S4,F14	Not Used	S7,F28	Process Program Verification Acknowledge (PVA)
S4,F15	Material Received (MRN)	S7,F29	Process Program Verification Inquire (PVI)
S4,F16	Not Used	S7,F30	Process Program Verification Grant (PVG)
5.9 Stream 5 Exception Reporting		S7,F31	Verification Request Send (VRS)
S5,F0	Abort Transaction	S7,F32	Verification Request Acknowledge (VRA)
S5,F1	Alarm Report Send (ARS)	S7,F33	Process Program Available Request (PAR)
S5,F2	Alarm Report Acknowledge (ARA)	S7,F34	Process Program Availability Data (PAD)
S5,F3	Enable/Disable Alarm Send (EAS)	5.12 Stream 8 Control Program Transfer	
S5,F4	Enable/Disable Alarm Acknowledge (EAA)	S8,F0	Abort Transaction
S5,F5	List Alarms Request (LAR)	S8,F1	Boot Program Request (BPR)
S5,F6	List Alarm Data (LAD)	S8,F2	Boot Program Data (BPD)
5.10 Stream 6 Data Collection		S8,F3	Executive Program Request (EPR)
S6,F0	Abort Transaction	S8,F4	Executive Program Data (EPD)
S6,F1	Trace Data Send (TDS)	5.13 Stream 9 System Errors	
S6,F2	Trace Data Acknowledge (TDA)	S9,F0	Abort Transaction
S6,F3	Discrete Variable Data Send (DVS)	S9,F1	Unrecognized Device ID (UDN)
S6,F4	Discrete Variable Data Acknowledge (DVA)	S9,F2	Not Used
S6,F5	Multi-block Data Send Inquire (MBI)	S9,F3	Unrecognized Stream Type (USN)
S6,F6	Multi-block Grant (MBG)	S9,F4	Not Used
S6,F7	Data Transfer Request (DDR)	S9,F5	Unrecognized Function Type (UFN)
S6,F8	Data Transfer Data (DDD)	S9,F6	Not Used
S6,F9	Formatted Variable Send (FVS)	S9,F7	Illegal Data (IDN)
S6,F10	Formatted Variable Acknowledge (FVA)	S9,F8	Not Used
5.11 Stream 7 Process Program Management		S9,F9	Transaction Timer Timeout (TTN)
S7,F0	Abort Transaction	S9,F10	Not Used
S7,F1	Process Program Load Inquire (PPI)		
S7,F2	Process Program Load Grant (POG)		
S7,F3	Process Program Send (PPS)		
S7,F4	Process Program Acknowledge (PPA)		
S7,F5	Process Program Request (PPR)		
S7,F6	Process Program Data (PPD)		
S7,F7	Process Program ID Request (PIR)		
S7,F8	Process Program ID Data (PID)		
S7,F9	M/P M Request (MMR)		
S7,F10	M/P M Data (MMD)		
S7,F11	M/P M Update Send (UMS)		
S7,F12	M/P M Update Acknowledge (UMA)		
S7,F13	Delete M/P M Entry Send (DES)		



E5-85

SEMI EQUIPMENT COMMUNICATIONS STANDARD 2 TRANSACTION (SECS-II) (PROPOSED)

1. Introduction

1.1 *Intent* – The SEMI Equipment Communications Standard Part 2 (SECS-II) is intended to be used in conjunction with SEMI Standard E4 (SECS-I) and does not stand alone. SECS-II defines the details of the language to be used in transactions between intelligent equipment and its host computer. The need for this definition was identified in Appendix 3 to SECS-I.

1.1.1 It is the intent of this standard to define messages to such a level of detail that some consistent host software may be constructed with only minimal knowledge of individual equipment. The equipment, in turn, may be constructed with only minimal knowledge of the host.

1.1.2 The set of messages chosen includes, in the opinion of the standards committee, the most typical activities required for IC manufacturing. It should be possible to perform basic functions of processing material without adding to the host software. Full equipment capabilities are expected to require some equipment specific software in the host.

1.1.3 SECS-II is intended to be compatible with a multiple node tree network as described in Appendix 4 to SECS-I.

1.2 *Overview* – SECS-I provides the basic definition of a protocol for exchanging blocks of data where each block is 256 bytes or less. The blocks themselves are similar to packets in a packet switching network. No form or meaning for the data is included in SECS-I. Both form and meaning are provided by SECS-II.

1.2.1. SECS-II defines, by means of general categories (streams) which contain detailed messages (functions), the methods of conveying information between intelligent processing equipment and its host computer. A request for information and the corresponding data transmission is an example of a function. The content of a message is structured as Items and Lists. This structure allows the data to describe itself.

1.2.2 The interchange of messages is governed by a set of rules for handling messages called the transaction protocol. The transaction protocol places some minimum requirements on any device connected to a SECS-II interface. For simple equipment, the requirements are few. For equipment having multiple SECS ports and device identifications (ID's), such as the host, a discussion is included in Appendix 1 to aid in the design of a system which is consistent with the

constraints defined by the transaction protocol.

1.3 *Application* – SECS-II applies to equipment used in the manufacturing of semiconductor devices which is attached to a host computer network. This standard applies to all types of transmissions between the computer and the processing equipment. Examples of the data transmission are: control programs, material movement information, measurement data, summarized test data and alarms.

1.3.1 The minimum compliance to this standard involves meeting the few constraints outlined in 3. It is expected that a given piece of equipment will require only a subset of the possible functions described in this standard. The number of functions and the selection of functions will depend upon the equipment capabilities and requirements. For each piece of equipment, the exact format for each function provided must be documented according to the form outlined in 6.

1.4 *Applicable Documents*

ANSI X3.4-1977 Code for Information Interchange (ASCII)

E4. SEMI Equipment Communications Standard 1 Data Link (SECS-I)

IEEE P754 Proposed Standard for Binary Floating Point Arithmetic (*Computer Magazine*, March 1981)

2. Message Structure

2.1 *Intent* – A message in SECS-II is like a sentence in English; it contains a complete idea. It is assumed that processing equipment, intermediate nodes, and a host are all intelligent machines connected in a network by point to point links following SECS-I. A message is passed between machines over these links in coded form. This section provides clarification of the term message and restrictions on the message ID in respect to SECS-I.

2.2 *Message and Blocks* – Since in a network a message may pass through several nodes between its source and destination, some additional terms are useful to avoid confusion in the following discussion. The term originator will refer to the software that creates the original message. The term interpreter will refer to the software that "understands" the message at the destination. These terms are distinct from the two ends of a SECS link which are called the sender



Documents

Document Sharing Hub

E5

© SEMI 1982, 1985



S9,F11	Data Too Long (DLN)	S11,F7	File Close (FCN)
S9,F12	Not Used	S11,F8	Not Used
S9,F13	Conversation Timeout (CTN)	S11,F9	File Delete (FDN)
S9,F14	Not Used	S11,F10	Not Used
5.14 Stream 10 Terminal Services		S11,F11	File Directory Request (FDR)
S10,F0	Abort Transaction	S11,F12	File Directory Data (FDD)
S10,F1	Terminal Request (TRN)	S11,F13	File Inquire Request (FIR)
S10,F2	Not Used	S11,F14	File Inquire Data (FID)
S10,F3	Terminal Display, Single (VTN)	6. Message Documentation	
S10,F4	Not Used	6.1 Intent	
S10,F5	Terminal Display, Multi-block (VMN)	6.2 Standard Form SECS-II Document	
S10,F6	Not Used	7. Units of Measure	
S10,F7	Multi-block Not Allowed (MNN)	7.1 Intent	
S10,F8	Not Used	7.2 Units Symbol	
S10,F9	Broadcast (BCN)	7.3 Compliance	
S10,F10	Not Used	7.4 SECS-II Units of Measure Identifiers	
5.15 Stream 11 Host File Services		Application Notes	
S11,F0	Abort Transaction	A.1 The General Node Transaction Protocol	
S11,F1	File Access Inquire (FAI)	A.2 Some Suggested Message Usage	
S11,F2	File Access Grant (FAG)	A.3 Notes on SECS-II Data Transfers	
S11,F3	File Write Data Send (FWS)	A.4 Process Programs	
S11,F4	File Write Acknowledge (FWA)	A.5 Baseline Implimentation of SECS	
S11,F5	File Read Request (FRR)		
S11,F6	File Read Data (FRD)		



or receiver depending upon the direction of the message being sent.

2.2.1 A long message may be broken into smaller pieces, called blocks, to facilitate handling. Blocks are established by the originator and remain intact until received by the interpreter. Each block of the message is identified as part of the same message by a 10 byte header described in E4, 5.1. Each block of a given message has the same device ID and message ID. Individual blocks of the message are handled on a single link according to the data link protocol described in E4, 4.

2.2.2 The interpreter of the message can reconstruct the original message from the individual blocks by stringing together the data between the header and the check bytes from each sequential block. Only one header is required by the interpreter in order to understand the message. The headers on other blocks are required to identify blocks of different messages that may be intermixed during sending. Thus, although each block sent contains a header, a logical model of the message contains only one header plus the data from each block as shown in Figure 1. This model is used for the message forms given in 5.

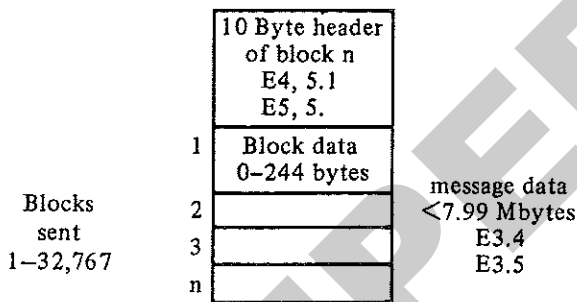


Figure 1
Message Structure Model

NOTE 1 – According to E4, the block count, bytes 5 and 6 of the header, is the only difference between the headers of a multiblock message. The first block has the count equal to 1 and the E bit equal to 0. Subsequent blocks up to and including the second to the last have counts incremented by one and the E bit equal to 0. The last block has the E bit set to 1.

2.3 *Block Length and Message Length* – The first character in each block refers to the length of the associated block as described in E4, 4.3.4. The message data in each block is limited to 244 bytes. A multi-block message may contain 32,767 blocks for a total of 7.99 Mbytes in one message. It is assumed that messages will be confined to as few blocks as is reasonable, but that there is no need to fill every block with 244 bytes should more blocks follow. The

length of the message is described by the message data itself (see 3.). Therefore, message lengths are not standardized. However, the data items are subject to length constraints as outlined in 5.

NOTE 2 – The length and content of the message are conceptually independent of the division of the message into multiple blocks of transmission. However, the interpretation of the message may be more efficient if it is divided at distinct points in the data. Natural breakpoints in the data can be chosen in places that limit the amount of message buffer required in the receiver or that allow immediate execution such as displaying one line of a CRT display. This simplification is achieved only if the originator and the interpreter follow the same breakpoint rules.

2.4 *Stream and Function Definition* – Bits 1 through 7 in byte 3 of the header are defined to be the Stream code. Bits 1 through 8 in byte 4 of the header are defined to be the Function code. Each combination of stream and function represents a distinct message identification.

NOTE 3 – All functions with the same stream code are intended to have similar or related activity with respect to the process of manufacturing. It is assumed that these functions are handled by a single software program or module of software. These assumptions are the basis for selecting the grouping of functions detailed in 5. The execution of the defined streams and functions is likely to be most efficient if this relationship between stream and software modules is true.

2.5 *Stream and Function Allocation* – The stream and function codes are assigned in the following manner:

Stream Code (Header byte 3)	Use
0-63	Definition reserved for this standard and its extensions.
64-127	Available for user definition.
Function Code (Header byte 4)	Use
0-63	Reserved for this standard and its extensions in all streams.
64-255	Available for user definition in all streams.

The reserved codes assigned by this standard are listed in 5. It is recognized that inevitably there will be user needs beyond the specific definitions given here. In these situations, the streams and functions reserved for user definition should be used subject to the guidelines for minimum compliance outlined in 3.

2.6 *Understanding the Message* – When a piece of equipment having a single SECS port receives a



sage, the device ID must match its own ID switch settings in order for the message to be accepted. The equipment then assumes that the message has been formatted according to its own understanding of stream and function. Even though a standard message follows the list and item structures, the specific data it contains will normally be equipment dependent. The equipment then attempts to fit the data received into its own internal form for handling or interpretation.

2.6.1 When a host-like device having multiple SECS ports receives a message, it must not only look at the stream and function, but also the device ID in order to determine the meaning of the message since it may receive the same stream and function from different types of equipment. The host has the burden of matching the device ID with the equipment type. This task can be simplified by judicious choice of device ID's such that the host can easily decode the equipment type. The equipment device ID is settable according to SECS so that the device ID selection can be made to aid the host.

3. Message Transaction and Conversation Protocols

3.1 *Intent* – Equipment in compliance with SECS-II must meet the minimum transaction requirements outlined in this section. The conversation protocols serve to further define the use of and interaction between transactions.

3.2 *Transaction Definition* – A transaction forms the basis for all information exchanges in SECS-II. A transaction consists of two related messages. The first message of a transaction is called the prime message. If the prime message requests some information or needs an acknowledgement, then the answer is called the secondary message. Secondary messages must have a corresponding prime message. Secondary messages may not ask for a reply. A prime message may stand alone as a transaction if it does not ask for a reply; however, space is still reserved for a secondary message; see 3.3.

3.3 *Function Code Restrictions* – All the functions used in SECS-II will follow a numbering convention corresponding to prime and secondary pairs. All prime messages will be given an odd numbered function and the corresponding secondary message will be given the next even function. Thus, the secondary message function code may be determined by adding one to the prime message function code.

3.3.1 The even numbered function following a single message transaction is not used.

3.3.2 Function code 0 is reserved in all streams for aborting transactions as described in 5.4.

3.4 *Transaction Protocol Requirements* – A protocol for handling data composed of individual characters is described in E4, 4. This section describes some requirements for handling data consisting of entire blocks which are parts of messages. The complexity of the requirements depends upon constraints in the received messages. Equipment which has the following transaction constraints:

- (a) Only one SECS port,
- (b) Only one device ID,
- (c) Receives only single block prime messages,

has the following minimum requirements in order to comply with SECS-II transaction protocol:

1. Send system bytes in the header of all secondary message blocks that are copies of the system bytes from the corresponding prime message as described in E4, 5.8.2. If the equipment originates a message which requires a reply, then the equipment may use the system bytes for its own matching of replies by placing a code there. It will be assumed that the host system will return this code with the reply.
2. Send an error message to the host on stream 9 for each block received that cannot be accepted. In E5, 5.13, S9-F1, F3, F5, F7, or F11 are possible.
3. Respond to S1,F1 with S1,F2 as described in 5.5.
4. Format other applicable messages according to 5.
5. The time between receipt of the prime message and the first block of the response or between blocks of a secondary message being sent must be less than 10 seconds.
- 3.4.1 Removing the constraint of receiving only single block prime messages adds several requirements to the protocol. These requirements involve the use of a software settable timer called the transaction timeout timer.
6. Transaction timeout must be remotely settable as an equipment constant from 1 to 120 seconds with a resolution of 1 second. See S2,F15 in 5.6. The default transaction timeout will be 10 seconds.
7. Incomplete received messages either due to the lack of receiving a requested secondary message or lack of receiving the last block in a multi-block message within the current transaction timeout limit must cause an error S9,F9 (see 5.13) to be sent to the host. Any committed resources can then be released.
8. Terminate related transaction after receiving function 0 and be prepared to receive a new prime message. The related transaction is the



terminated by the contents of the header on the function 0 message. No error message is sent to the host.

3.4.2 For equipment having more than one SECS port or more than one device ID, such as in the host, the handling of the protocol to meet the transaction requirements can become quite complicated. As an aid in the implementation of such systems, an algorithm is presented in Appendix 1 which illustrates some of the considerations which must be given to timeout and error recovery. These ideas emphasize the philosophy behind the transaction requirements presented in this section.

3.5 Conversation Protocols – A conversation is an exchange of information between an originator and an interpreter that leaves both ends of the conversation free of timeouts or resource commitments at its conclusion. Most conversations in SECS-II are completed in one transaction. There are six types of conversations which characterize all information exchanges in SECS-II:

1. A single primary message with no reply is the simplest conversation. This message must fit in one block. The originator must assume that the interpreter responds to the message. This conversation is used where the originator can do nothing if the message is rejected.
2. If the interpreter has data that the originator wants, the data are requested with a prime message and the data returned to the originator as a secondary message. It is assumed that if the originator requests data that it is prepared to receive the amount of data returned. This is the request/data conversation.
3. If the originator wishes to send only one block of data to the interpreter, then the originator sends the data and expects an acknowledgement from the interpreter. This is the send/ack conversation.
4. If the originator has multiple blocks of data for a particular exchange, then the originator must receive permission from the interpreter prior to sending the data. Two transactions are involved. The first transaction inquires if room is available and the interpreter grants the room. In the second transaction, the originator sends the data and the interpreter acknowledges the reception. This is the inquire/grant/send/ack conversation. Between the grant and the send, the interpreter may commit some resources in preparation for the data. Consequently, a conversation timeout may be set by the interpreter at a time much longer than the transaction timeout, at which

time the interpreter will free its resources and send an error message to the host.

5. The file conversation is the last dealing with data. In principle, it is the same as the inquire/grant/send/ack conversation except that after the initial inquire/grant transaction, any number of send/ack or request/data transactions may follow which are connected to the initial inquire/grant transaction by a code called the logical unit number. A file conversation is completed by a single message transaction called close file. There are no timeouts usually associated with a file conversation, but the host will certainly have some means of closing files that are left open beyond some point.
6. The last conversation in SECS-II is related to material movement. Wafers are passed between two pieces of equipment using single message transactions in a handshake. This conversation is essentially continuous as long as the two pieces of equipment handle material in this way. The conversation is described in detail in stream 4 (5.8).

The key words, request, data, send, acknowledge, inquire, and grant are used in the function names as an aid to understanding the relationship between the messages and the conversation. Single message transactions do not use these words.

4. Data Structures

4.1 Intent – All information transmitted according to this standard will be formatted using two data structures, items and lists. These data structures define the logical divisions of the message, as distinct from the physical divisions of the message which are defined by the data link protocol (E4, 4.) and the message protocol (E4, 5.). They are intended to provide a self-describing internal structure to messages passed between two machines to aid in understanding by the interpreter.

4.2 Item – An item is an information packet which has a length and format defined by the first 2, 3, or 4 bytes of the item. These first bytes are called the item header (IH). The item header consists of the format byte and the length byte(s) as shown in Figure 2. Bits one and two of the item header tell how many of the following bytes refer to the length of the item. This feature allows for long items without requiring the byte overhead for shorter items. The item length refers to the number of bytes following the item header, called the item body (IB), which is the actual data of the item. The item length refers only to the item body not including the item header, so the

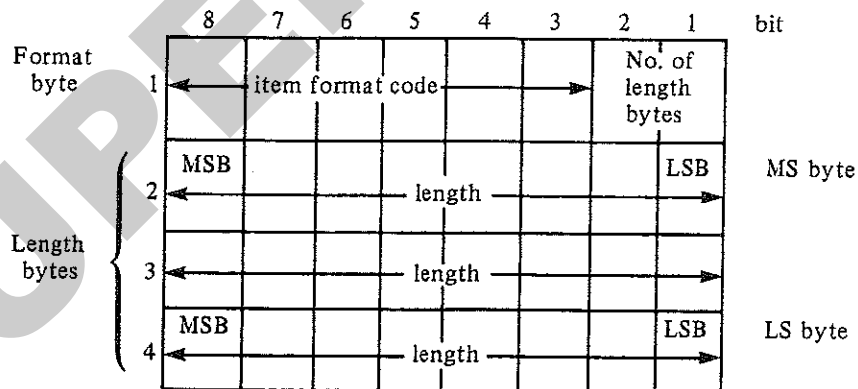
actual number of bytes in the message for one item is the item length plus 2, 3, or 4 bytes for the item header. All bytes in the item body are in the format specified in the format byte.

4.2.1 A zero length in the format byte is illegal and produces an error. A zero length in the item length bytes has a special meaning as defined in the detailed message definitions.

4.2.2 Bits 3 through 8 of the format byte of the item header define the format of the data which follows. Of the 63 possible formats, fourteen are defined as shown in Table 1. Format code 0 is called a list and is defined in E5 4.3. The remaining 13 item formats define unspecified binary, code 10 (octal); Boolean, code 11 (octal); ASCII character strings, code 20 (octal); signed integer, codes 30, 31, 32, 34 (octal); floating point, codes 40, 44 (octal); and un-

signed integer, codes 50, 51, 52, 54 (octal). These formats are used for groups of data that have the same representation in order to save repeated item headers. Signed integers will be two's complement values. Floating point numbers will conform to the IEEE standard IEEE task P754, draft 8.0. Boolean values will be byte quantities with zero being equivalent to false and non-zero being equivalent to true.

4.3 *List* – A list is an ordered set of elements, where an element can be either an item (4.2) or a list. The list header (LH) has the same form as an item header with format type 0. However, the length bytes refer to the number of elements in the list rather than to the number of bytes. The list structure allows grouping items of related information which may have different formats into a useful structure.



No. of Length bytes
 0 = illegal, data format error
 1 = One binary length byte (max = 255)
 2 = Two binary length byte (max = 64K)
 3 = Three binary length bytes (max = 7.99M)

Length refers to the number of bytes except in a list format where it refers to the number of elements (items or lists) in the list.

Figure 2
Item and List Header



Table 1
Item Format Codes

Format Code		Meaning
Binary bit-876543	octal	The data after the heading has the following form.
000000	00	LIST (length in elements)
001000	10	Binary
001001	11	Boolean
010000	20	ASCII ¹
011000	30	8-byte integer (signed) ²
011001	31	1-byte integer (signed)
011010	32	2-byte integer (signed) ²
011100	34	4-byte integer (signed) ²
100000	40	8-byte floating point ³
100100	44	4-byte floating point ³
101000	50	8-byte integer (unsigned) ²
101001	51	1-byte integer (unsigned)
101010	52	2-byte integer (unsigned) ²
101100	54	4-byte integer (unsigned) ²

¹ Non-printing characters are equipment specific.

² Most significant byte sent first.

³ IEEE P754. The byte containing the sign bit is sent first.

NOTE — Changes in integer format codes may conflict with earlier implementations.

4.3.1 A zero length in the format byte is illegal and produces an error. A zero length in the list length bytes has a special meaning as defined in the detailed message definitions.

4.4 *Example Data Structures* — The data structures for different types of items are illustrated in the following examples.

- (a) An item contains one binary code 10101010.
bit
87654321
- 00100001 Item with 1 length byte
00000001 1 byte long
10101010 data byte

- (b) An item contains three ASCII characters ABC.
- 01000001 Item ASCII, 1 length byte
00000011 Three bytes long
01000001 ASCII A
01000010 ASCII B
01000011 ASCII C

- (c) An item contains three binary numbers in 2-byte form.
- 01101001 Item, 2-byte integers
00000110 6 bytes total (6/2=3 integers)
xxxxxxx MSByte number x
xxxxxxx LSByte number x
yyyyyyyy MSByte number y
yyyyyyyy LSByte number y
zzzzzzzz MSByte number z
zzzzzzzz LSByte number z

- (d) An item contains one 4-byte IEEE floating point number.
- 10010001 Item, 4-byte floating point
00000100 4 bytes (4/4=1 number)
ffffffff
ffffffff Floating point number
ffffffff in IEEE P754
ffffffff

- (e) A message is sent from device 42 telling the host that the temperature at point T1 has exceeded a preset process limit. The message ID is stream 5, function 1, and the data consists of a list of three items. The first item is a code for the alarm set and the alarm category code. The second item is the equipment-specific alarm number for this alarm, say 17. The third item is a string of text giving a brief description of the alarm, say "T1 high." No reply is requested. The complete message including the header is as follows:

10000000 R=1 (to the host)—
00101010 Device Code=42
00000101 Stream 5, W=0 10 byte
00000001 Function 1 Header
10000000 E=1
00000001 Block 1
00000000
00000000 System bytes = 0
00000000
00000000
00000001 List



```

00000011 3 Elements
00100001 Item, 1-byte, next byte
length
00000001 1 byte long
10000100 Alarm set, category 4
01100101 Item, 1-byte integer, next byte
length
00000001 1 byte long
00010001 Alarm 17
01000001 Item, ASCII, next byte length
00000111 7 characters
01010100 ASCII T
00110001 ASCII I
00100000 ASCII space
01001000 ASCII h
01001001 ASCII i
01000111 ASCII g
01001000 ASCII h

```

The entire message contains 17 bytes of data, 10 bytes of header, 1 byte length (not shown), and 2 bytes checksum (not shown) for a total of 30 bytes.

At 9600 baud transmission, the message would be sent in 31 milliseconds.

5. Message Detail

5.1 *Intent* – This section defines a number of specific functions in different streams which can be used as a basis for communication between a host computer and a piece of processing equipment. The functions are defined in the form of transaction message pairs according to the transaction protocol requirements specified in 3.

5.2 The functions are described in a standard form which involves specification of the number, name, single or multiple block, direction of communication, nature of reply required, description, variable definition, and the detailed structure of the message in terms of lists and items. Double lines separate streams and single lines separate transactions to aid readability.

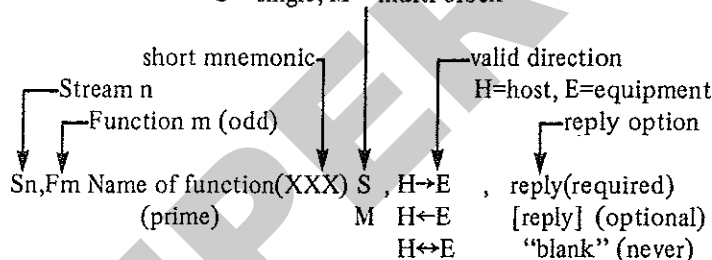
5.3 The abbreviations used in each transaction are as follows:

Define: Variables to be defined

Required formats (octal)

3() means all formats starting with 3

S = single, M = multi-block



Description: A description of the action generated by the function.

Structure: Detailed structure showing lists and defined items. Lists are denoted by a capital L followed by the list length separated by a comma. The individual elements in the list are numbered on separate lines. Nested lists are indented to emphasize the structure. The detailed form of the items is given in the define section at the beginning of the transaction. Items used several times in one stream are only defined once where first used. The symbols < and > are used to enclose each item in the structure definition and implies that there is an item header.

Exceptions: Special cases in the structure that have a different meaning.

Sn, Fm+1 Name of function
(secondary)

(same structure as above except never with reply)

5.4 *Stream 0 and Function 0* – Stream 0 is always defined as not used since a 0 is the most likely error. No functions are defined in stream 0.

5.4.1 Function 0 exists in all streams and has the same special meaning in each stream. Function 0

means that the current transaction is aborted due to a transmission error or some other system requirement. An application program receiving either a multiblock message or a secondary message must respond to receiving a function 0 as part of the message.



ping the transaction and cleaning up the application program. Part of this cleanup may involve closing files, releasing buffers, or sending a notice to a local terminal. No error messages need be generated to the host.

5.4.2 One situation which may generate function code 0 is a timeout of one of the receive timers (transaction timers, see Appendix 1). The function 0 is generated where the timeout first occurs and is sent as a reply or a message block to signify that no

more blocks can be expected. The transaction is aborted at this point. The device detecting the timeout will send an error message, S9,F9 to the host which contains the device number of that device in the header. The host can then determine where the error occurred while the F0 message is used to terminate the transaction elsewhere in the system. Equipment with only the host port is not burdened with generating F0 on a receive timeout. This task is reserved for multi-port nodes. See Appendix 1.

5.5 *Stream 1 Equipment status* – This stream provides a means for exchanging information about the status of the equipment, including its current mode,

depletion of various consumable items, and the status of transfer operations.

S1,F0 Abort Transaction (S1F0)

S,H↔E

Description: May occur in a multi-block message or in reply to a primary message. Terminates further work on this transaction. A similar function exists in each stream.

Structure: Header only. The header is constructed from the stored header for the transaction in progress by changing the function code to 0 and the E-bit to a 1. (See Appendix 1 for discussion)

Define: MDLN Equipment model type 6 bytes max.

Format: 20

SOFTREV Software revision code 6 bytes max.

Format: 20

S1,F1 Are You There Request (R)

S,H↔E, reply

Description: Establishes if the equipment is on line. A function 0 response to this function means the communication is inoperative. In the equipment, a function 0 is equivalent to a timeout on the receive timer after issuing S1,F1 to the host.

Structure: Header only

S1,F2 On Line Data (D)

S,H↔E

Description: Data signifying that the equipment is alive.

Structure: L,2
1. <MDLN>
2. <SOFTREV>

Exceptions: The host sends a zero length list to the equipment.

Define: SVID Status variable ID

Format: 20,3(),5()

Status variables may include any parameter that can be sampled in time such as temperature or quantity of a consumable.

SV Status variable data

Format: 20,3(),4(),5()

S1,F3 Selected Equipment Status Request (SSR)

S,H→E, reply

Description: A request to the equipment to report selected values of its status.

Structure: <SVID1, . . . ,SVIDn>

Exceptions: A zero length item means report all SVID's.



S1,F4 Selected Equipment Status Data (SSD)

M,H←E

Description: The equipment reports the value of each SVID requested in the order requested. The host remembers the names of values requested.

Structure: L,n
1. <SV1>
:
n. <SVn>

Exceptions: If n=0, no response can be made. A zero length returned for SVi means that SVIDi does not exist.

Define: SFCD Status form code, 1 byte

Format: 10

S1,F5 Formatted Status Request (FSR)

S,H→E, reply

Description: A request for the equipment to report the status according to a predefined fixed format.

Structure: <SFCD>

S1,F6 Formatted Status Data (FSD)

M,H←E

Description: The equipment reports the value of status variables according to the SFCD.

Structure: Depends upon the structure specified by the status form.

Exceptions: A zero length item means that no report can be made.

S1,F7 Fixed Form Request (FFR)

S,H→E, reply

Description: A request for the form used in S1,F6.

Structure: <SFCD>

S1,F8 Fixed Form Data (FFD)

M,H←E

Description: The form is returned with the name of each value and the corresponding data format item having a zero length as a two element list in the place of each single item to be returned in S1,F6.

Format: Depends upon the form being specified.

Exceptions: A zero length item means the form is unavailable.

Define: TSIP Transfer status of input port, 1 byte

Format: 10

- 1 = Idle state
- 2 = Prep state
- 3 = Track on state
- 4 = Stuck in Receiver state
- 5-63 reserved.

TSOP Transfer status of output port, 1 byte

Format: 10

- 1 = Idle state
- 2 = Prep state
- 3 = Track on state
- 4 = Stuck in Sender state
- 5 = Completion state
- 6-63 reserved.

S1,F9 Material Transfer Status Request (TSR)

S,H→E, reply

Description: A request to report the status of all material ports to the host.

Structure: Header only



Documents
Document Sharing Hub



S1,F10 Material Transfer Status Data (TSD)

M,H←E

Description: The equipment reports to the host the transfer status of all material ports.

Structure: L,2

1. <TSIP1, . . . ,TSIPn>
2. <TSOP1, . . . ,TSOPn>.

Exceptions: A zero length item means there are no such ports.
 A zero length list means there are no ports.

5.6 Stream 2 Equipment control and diagnostics –
 Messages which deal with control of the equipment from the host. This includes all remote operations and equipment self-diagnostics and calibration but specifically excludes the control operations which

are associated with material transfer (see stream 4), loading of executive and boot programs (stream 8), and all file and operating system calls (streams 10, 11).

S2,F0 Abort Transaction (S2F0)

S,H↔E

Description: Same form as S1F0

Define: GRANT Grant code, 1 byte

0 = positive response, load OK

1 = busy, try again

2 = no space

3 = duplicate name

< 3 = equipment specific error code.

4-63 reserved.

LENGTH Length of the service program in bytes.

SPID Service program ID, 6 char.

Format: 10

Format: 3(),5()

Format: 20

S2,F1 Service Program Load Inquire (SPI)

S,H↔E,reply

Description: Either the host or equipment wants to send the specified program.

Structure: L,2

1. <SPID>

2. <LENGTH>

S2,F2 Service Program Load Grant (SPG)

S,H↔E

Description: Provides permission to load.

Structure: <GRANT>

Define: SPD Service program data

ACKC2 Acknowledge code, 1 byte

0 = everything correct

> 0 = equipment specific error

1-63 reserved.

Format: 10

Format: 10

S2,F3 Service Program Send (SPS)

M,H↔E,reply

Description: The data associated with the S2,F1 inquire is sent.

Structure: <SPD>

S2,F4 Service Program Send Acknowledge (SPA)

S,H↔E

Description: Acknowledge or error.

Structure: <ACKC2>



Documents

Document Sharing Hub



S2,F5 Service Program Load Request (SPR) *S,H↔E,reply*

Description: A service program is requested.

Structure: <SPID>

S2,F6 Service Program Load Data (SPD) *M,H↔E*

Description: A service program is sent.

Structure: <SPD>

Exceptions: A zero length item means that the requested program can not be returned.

S2,F7 Service Program Run Send (CSS) *S,H→E,reply*

Description: Start the requested program.

Structure: <SPID>

S2,F8 Service Program Run Acknowledge (CSA) *S,H←E*

Description: Acknowledge or error.

Structure: <ACKC2>

Define: SPR Service program results Format: Device dependent

S2,F9 Service Program Results Request (SRR) *S,H→E,reply*

Description: Ask for results of service program.

Structure: <SPID>

S2,F10 Service Program Results Data (SRD) *M,H←E*

Description: Get the results back.

Structure: <SPR>

Exceptions: A zero length item means SPR does not exist.

S2,F11 Service Program Directory Request (SDR) *S,H↔E,reply*

Description: There may be more than one service program.

Structure: Header only.

S2,F12 Service Program Directory Data (SDD) *S,H↔E*

Description: A list of service program names.

Structure: L,n
 1. <SPID1>
 ⋮
 n. <SPIDn>

Exceptions: If n = 0, there are no service programs.

Define: ECID Equipment Constant ID Format: 3(),20,51

1 = Receiver timer, 1 byte integer with value of 1-120
 2-63 Reserved.

ECV Equipment Constant Value Format: 3(),20,5()



S2,F13 Equipment Constant Request (ECR)

S,H→E,reply

Description: Constants such as for calibration, servo gain, alarm limits, data collection mode and other values that are changed infrequently can be obtained using this message.

Structure: <ECID1, . . . ,ECIDn>

Exceptions: A zero length item requests all constants according to a pre-defined format.

S2,F14 Equipment Constant Data (ECD)

M,H←E

Description: Data response to F13 in the order requested.

Structure: L,n
1. <ECV1>
2. <ECV2>
.
n. <ECVn>

Exceptions: If n = 0, then no response exists. A zero length ECVi means that ECIDi does not exist.

Define: EAC Equipment acknowledge code, 1 byte
0 = Acknowledge
1 = Constant does not exist
2 = Can not change constants now
> 2 = Other equipment-specific error
3-63 reserved.

Format: 10

S2,F15 New Equipment Constant Send (ECS)

S,H→E,reply

Description: Change one or more equipment constants.

Structure: L,n
1.L,2
1.<ECID1>
2.<ECV1>
2.L,2
.
n.L,2
1.<ECIDn>
2.<ECVn>

S2,F16 New Equipment Constant Acknowledge (ECA)

S,H←E

Description: Acknowledge or error.

Structure: <EAC>

Define: TIME time of day, 12 bytes yymmddhhmmss

Format:20

S2,F17 Date and Time Request (DTR)

S,H↔E,reply

Description: Useful to check equipment time base or for equipment to synchronize with the host time base.

Structure: Header only

S2,F18 Date and Time Data (DTD)

S,H↔E

Description: Actual time data.

Structure: <TIME>

Exceptions: A zero length item means no time exists.



Define:

RIC

Reset code, 1 byte

0 = not used

1 = power up reset

> 1 = other reset conditions

2-63 reserved.

Format: 31,51

RAC

Reset acknowledge, 1 byte

0 = reset to be done

1 = reset denied

> 1 = other errors

2-63 reserved.

Format: 31,51

S2,F19 Reset/Initialize Send (RIS)

S,H→E,reply

Description:

Causes equipment to reach one of several predetermined initialized conditions.

Structure:

<RIC>

S2,F20 Reset Acknowledge (RIA)

S,H←E

Description:

Acknowledge or error.

Structure:

<RAC>

Define:

RCMD

Remote command code or string

Format: 31,20,51

CMDA

Command acknowledge code

0 = completed or done

1 = command does not exist

2 = can not perform now

> 2 = other equipment specific error

3-63 reserved.

Format: 31,51

S2,F21 Remote Command Send (RCS)

S,H→E,[reply]

Description:

Similar to pressing buttons on the front panel or causes some equipment activity to commence or to cease.

Structure:

<RCMD>

S2,F22 Remote Command Acknowledge (RCA)

S,H←E

Description:

Acknowledge or error.

Structure:

<CMDA>

Define:

TRID

Trace request ID

Format: 20,3(),5()

DSPER

Data sample period

hhmmss, 6 bytes

Format: 20

TOTSMP

Total samples to be made

Format: 20,3(),5()

REPGSZ

Reporting group size

Format: 20,3(),5()

SVID

Status variable ID

Format: 20,3(),5()



S2,F23 Trace Initialize Send (TIS)

S,H→E,reply

Description: Status variables exist at all times. This function provides a way to sample a subset of those status variables as a function of time. The trace data is returned on S6,F1 and is related to the original request by the trace request ID. Multiple trace requests may be made to that equipment allowing it.

Structure: L,5
1. <TRID>
2. <DSPER>
3. <TOTSMP>
4. <REPGSZ>
5. <SVID1, . . . ,SVIDn>

S2,F24 Trace Initialize Acknowledge (TIA)

S,H←E

Description: Acknowledge or error.

Structure: <ACKC2>

Define: ABS Any binary string

Format: 10

S2,F25 Loopback Diagnostic Request (LDR)

S,H↔E,reply

Description: A diagnostic message for checkout of protocol and communication circuits. The binary string sent is echoed back.

Structure: <ABS>

S2,F26 Loopback Diagnostic Data (LDD)

S,H↔E

Description: The echoed binary string.

Structure: <ABS>

Define: LOC Machine material location code, 1 byte

Format: 10

MID Material ID, 16 characters maximum

Format: 10,20

PPID Process program ID

Format: 10,20

S2,F27 Initiate Processing Request (IPR)

S,H→E,reply

Description: Host requests equipment to initiate processing of the identified material at the specified location in the machine using the specified process program.

Structure: List 3
1. <LOC>
2. <PPID>
3. List n
1. <MID1>
.
n. <MIDn>

Exception: A zero length PPID indicates no process program is being specified and the equipment is to take whatever action is appropriate for it to determine the proper program to use.

A zero length MID list indicates no MID is to be associated with the material to be processed.

S2,F28 Initiate Processing Acknowledge (IPA)

S,H←E

Description: Response by equipment to Initiate Processing Request. Returned status indicates whether or not the request was honored by the equipment.

Structure: <CMDA>



Documents

Document Sharing Hub



5.7 *Stream 3 Material Status* – The functions of the material status stream are used to communicate material-in-process and time-to-completion informa-

tion to the host, as well as to inform the host of extraordinary material occurrences.

S3,F0 Abort Transaction (S3F0)

S,H↔E

Description: Same form as S1F0.

Define:	MF	Material format code 1 byte for Format 10 Items with format 10 will be encoded as follows: 1 = quantities in wafers 2 = quantities in cassettes 3 = quantities in die or chips 4 = quantities in boats 5 = quantities in ingots 6 = quantities in leadframes 7 = quantities in lots 8 = quantities in magazines 9 = quantities in packages 10 = quantities in plates 11 = quantities in tubes 12 = quantities in waferframes 13-63 reserved. Items with format 20 will be a unit identifier for one of the special SECS generic units, as specified in Section 7.4.	Format: 10,20
	LOC	Material location code, 1 byte	Format: 10
	QUA	Quantity in format, 1 byte	Format: 10
	MID	Material ID, 16 characters max.	Format: 20,10

S3,F1 Material Status Request (MSR)

S,H→E,reply

Description: Host requests the device to send the status of all material in process.

Structure: Header only

S3,F2 Material Status Data (MSD)

M,H←E

Description: Material in process information is sent from the equipment to the host. There are m locations.

Structure: L,2
 1. <MF>
 2. L,m
 1. L,3
 1. <LOC1>
 2. <QUA1>
 3. <MID1>
 2. L,3
 ⋮
 m. L,3
 1. <LOCm>
 2. <QUAm>
 3. <MIDm>

Exceptions: A zero length list returned means no such data exists.



Define: TTC Time to completion

Format: 5()

S3,F3 Time to Completion Request (TCR)

S,H→E,reply

Description: Host requests the equipment to send the time to completion of operations on all material in possession.

Structure: Header only

S3,F4 Time to Completion Data (TCD)

M,H←E

Description: Time to completion information is sent by the equipment to the host.

Structure: L,2

1. <MF>
2. L,m
 1. L,3
 1. <TTC1>
 2. <QUA1>
 3. <MID1>
 2. L,3
 - ...
 - m. L,3
 1. <TTCm>
 2. <QUAm>
 3. <MIDm>

Exceptions: A zero length list header returned means no such data exists.

Define: ACKC3 Acknowledge code, 1 byte
 0 = accepted
 > 0 = error, not accepted
 1-63 reserved.

Format: 10

S3,F5 Material Found Send (MFS)

S,H←E,[reply]

Description: The equipment advised the host that unsolicited material has appeared at one of its sensors.

Structure: L,2

1. <MF>
2. <QUA>

S3,F6 Material Found Acknowledge (MFA)

S,H→E

Description: Acknowledge or error.

Structure: <ACKC3>

S3,F7 Material Lost Send (MLS)

S,H←E,[reply]

Description: The equipment advised the host that material has disappeared from its sensors.

Structure: L,3

1. <MF>
2. <QUA>
3. <MID>

S3,F8 Material Lost Acknowledge (MLA)

S,H→E

Description: Acknowledge or error.

Structure: <ACKC3>



Define:	EMID Equivalent material ID (16 bytes)	Format: 10,20
	<i>S3,F9 Material ID Equate Send (IES)</i>	<i>S,H←E,reply</i>
Description:	Provide an alternative name to be used as equivalent to the original material ID.	
Structure:	L,2	
	1. <MID>	
	2. <EMID>	
	<i>S3,F10 Material ID Equate Acknowledge (IEA)</i>	<i>S,H→E</i>
Description:	Acknowledge or error.	
Structure:	<ACKC3>	

5.8 *Stream 4 Material Control* – The functions in the material control stream are used to effect the automatic transfer of material between equipment. A simple handshake is achieved which provides for a variety of error conditions which gracefully terminate the handshake. Separate messages advise the host of errors and completed material transfers.

5.8.1 Since the handshake and host messages are separate, the handshake may be achieved transparently through the host or by direct connection between equipment. The host completes the handshake by relaying messages between the equipment. Only a single port is required on the equipment and the equipment has a simple message handling requirement. When a direct connection is desired, at least three ports are required, the receiving equipment must look like the host with respect to the sending equipment, and message handling in the equipment is significantly more complicated than in the host only connection. Nevertheless, the direct connection may still be chosen in an attempt to provide operation without a host. Since the host is reasonably transparent in the material handshake, a simple explanation of the handshake may be achieved by just considering the exchange of messages between the sender, the equipment wanting to get rid of material, and the receiver, the equipment able to accept the material.

5.8.2 Figure 3 shows six possible handshake situations between the sender and the receiver. There are two normal handshakes. Figure 3(a) shows the normal three message exchange when material is passed between equipment. The host is informed of a complete transfer of material. Figure 3(b) shows an alternative message exchange where the sender changes its mind and decides not to send the material. Figures 3(c) and (d) show two situations where the

material gets stuck during the transfer. In each situation an error message is issued to the host from the equipment where the material is stuck. The other equipment terminates normally. When material is stuck, manual intervention is required to move the material towards the equipment which indicates the stuck condition. The manual intervention has two possible outcomes. One, the material can be moved to a position where the handshake can resume or, two, the material is broken or lost from the transfer. Lost material causes a lost material error message to be sent to the host prior to resuming the operation. The specific details of recovering from stuck material are equipment-dependent. The stuck material condition is determined by the amount of time the material transfer mechanism is turned on. The sender claims stuck material if the material is not clear of its sensor before a time t_1 . The receiver claims stuck material if the material is not received before time t_2 . Figures 3(e) and (f) show the possible error conditions in the unlikely event that for some reason a handshake message is lost. Figure 3(e) shows that time t_3 is the longest that the sender will wait for material-received message. Times t_2 and t_3 set an upper limit on the amount of time either material transport mechanism will operate.

5.8.3 Figure 4 summarizes the interaction of the timers, handshake messages and the error messages in the form of a flow chart. It also identifies specific states for the sender and the receiver. These states are referred to in the messages.

5.8.4 The ranges of timer values are as follows:

t_1 ——— time to leave sender
 $t_1+10 \leq t_2 \leq 60$ sec. ——— time to receive
 $t_2+10 \leq t_3 \leq 70$ sec. ——— time to complete send

Default values, $t_1=10$ sec., $t_2=60$ sec., $t_3=70$ sec.

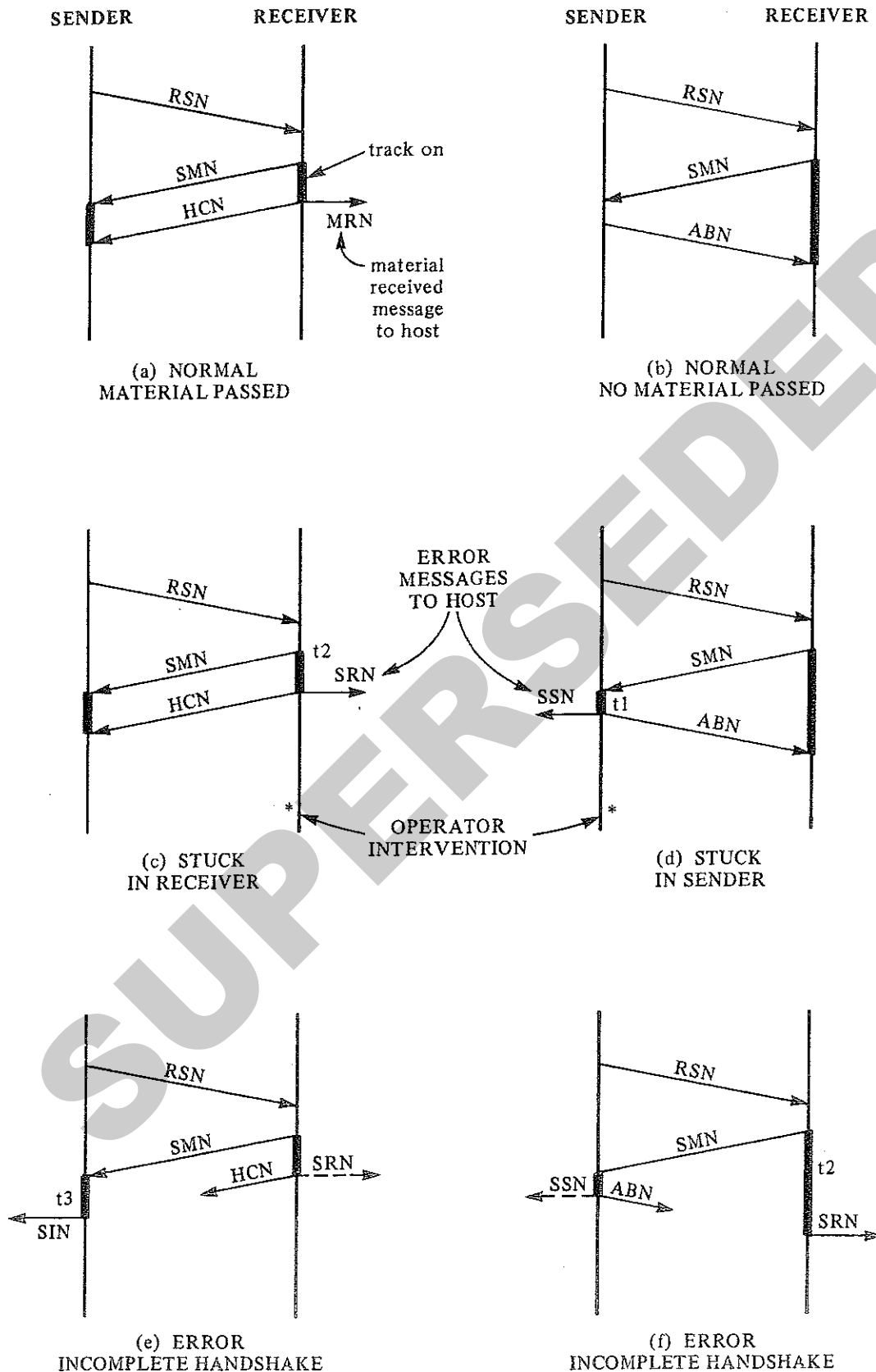
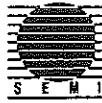


Figure 3
The Six Possible Handshakes



S4,F0 Abort Transaction (S4F0)

S,H↔E

Description: Same form as S1F0.

Define: MID Material ID, 16 char. max.
PTN Material Port number, 1 byte

Format: 10,20

Format: 10

S4,F1 Ready to Send Material (RSN)

S,H↔E

Description: The sender advises the receiver that some material is awaiting transfer.

Structure: L,2
1. <PTN>
2. <MID>

S4,F2 Not used

S4,F3 Send Material (SMN)

S,H↔E

Description: The receiver advises the sender that it is ready to receive material and that its transfer mechanism is running.

Structure: L,2
1. <PTN>
2. <MID>

S4,F4 Not used

S4,F5 Handshake Complete (HCN)

S,H↔E

Description: Receiver advises sender that the handshake is complete. The sender may now stop its transfer mechanism.

Structure: L,2
1. <PTN>
2. <MID>

S4,F6 Not used

S4,F7 Not Ready to Send (ABN)

S,H↔E

Description: Sender advises receiver that no material is being sent. The receiver may now stop its transfer mechanism.

Structure: L,2
1. <PTN>
2. <MID>

S4,F8 Not Used

S4,F9 Stuck in Sender (SSN)

S,H↔E

Description: An error from the sender to the host. The time between the receipt of Send Material (SMN) and the material leaving the sender's sensor exceeds the sender's t1 timeout. The sender goes to a hold state until the disposition of the stuck material is determined.

Structure: L,2
1. <PTN>
2. <MID>

S4,F10 Not Used
