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.

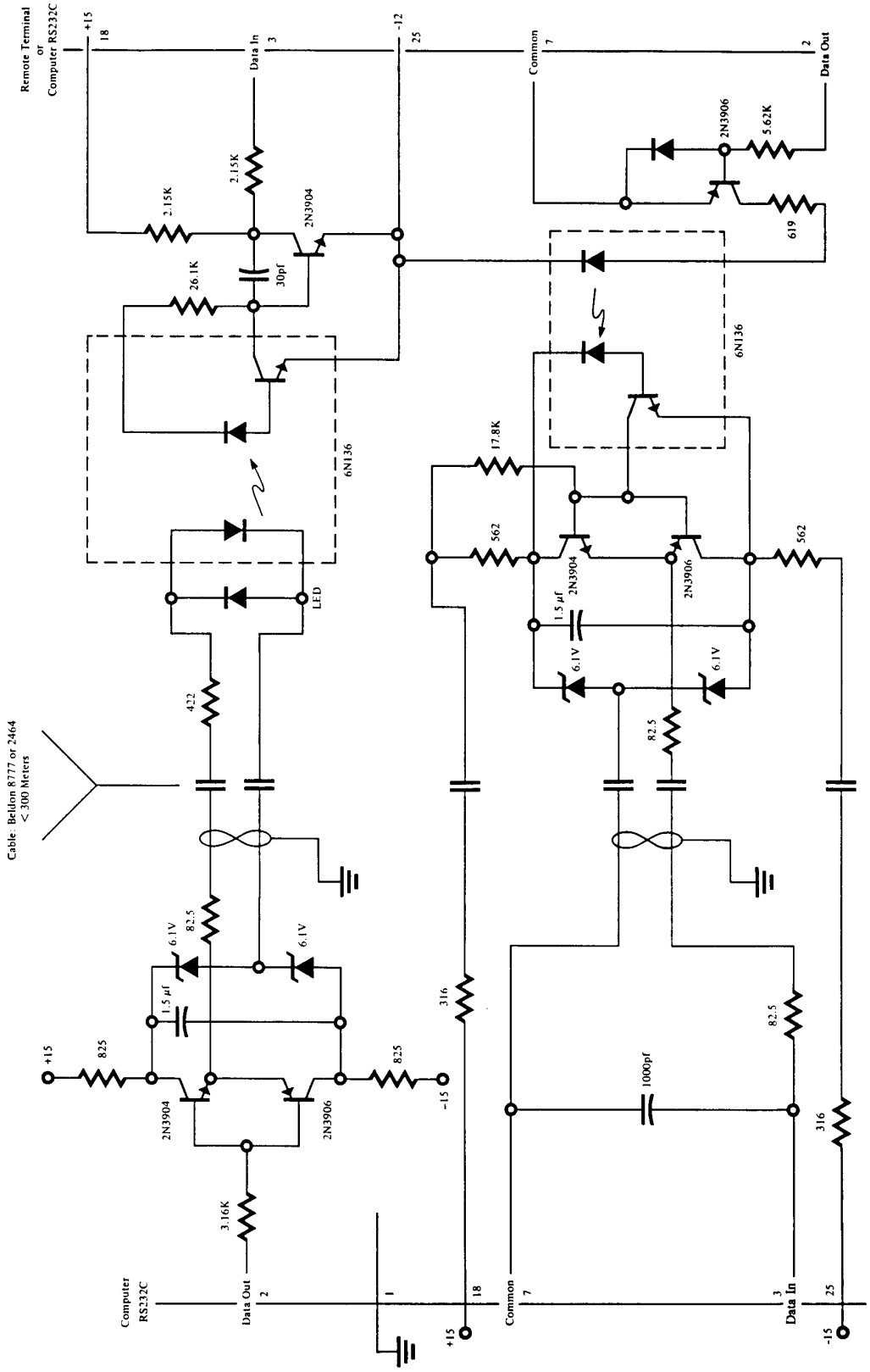

Figure R1-1

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.

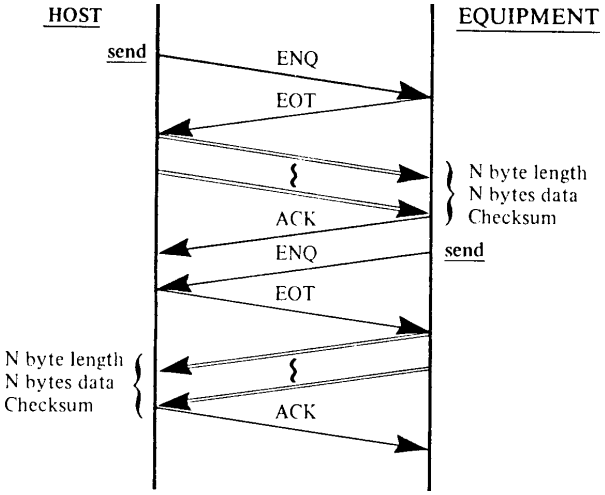

Figure R1-2

Ask for Status

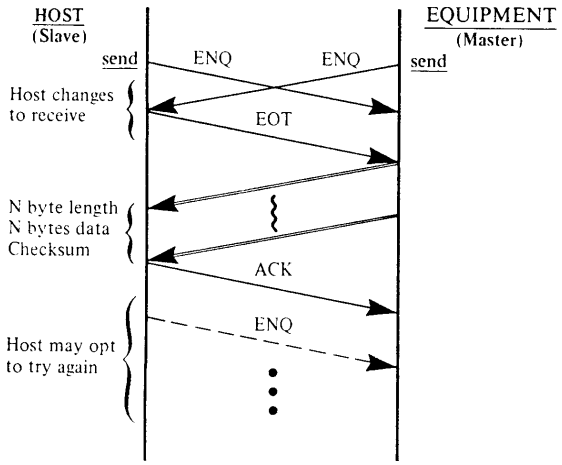

Figure R1-3

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.

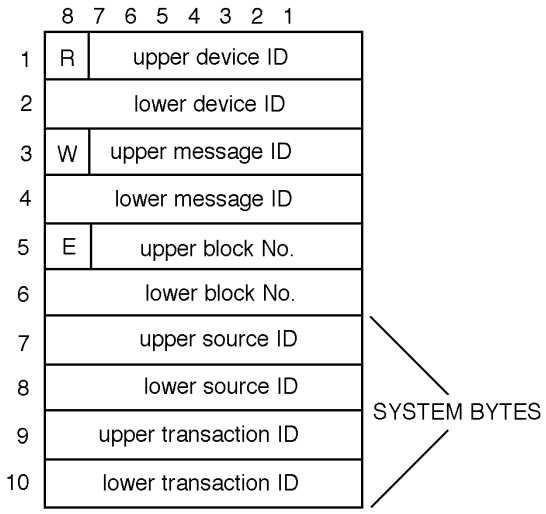

Figure R1-4

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.

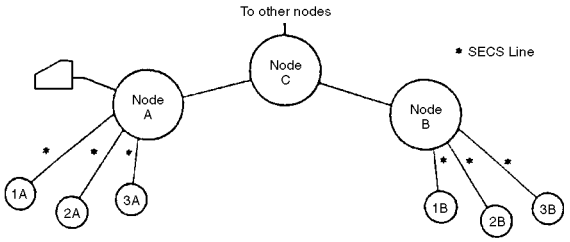

Figure R1-5

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.

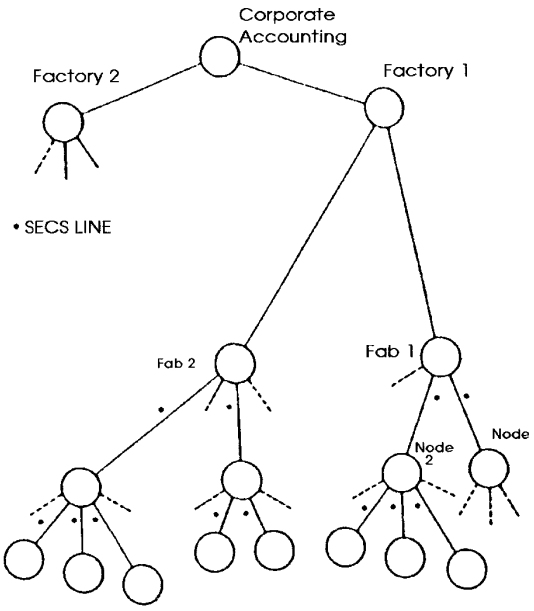

Figure R1-6

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.

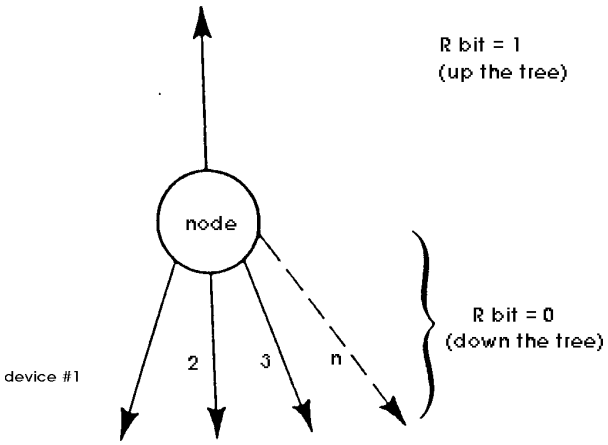

Figure R1-7

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.

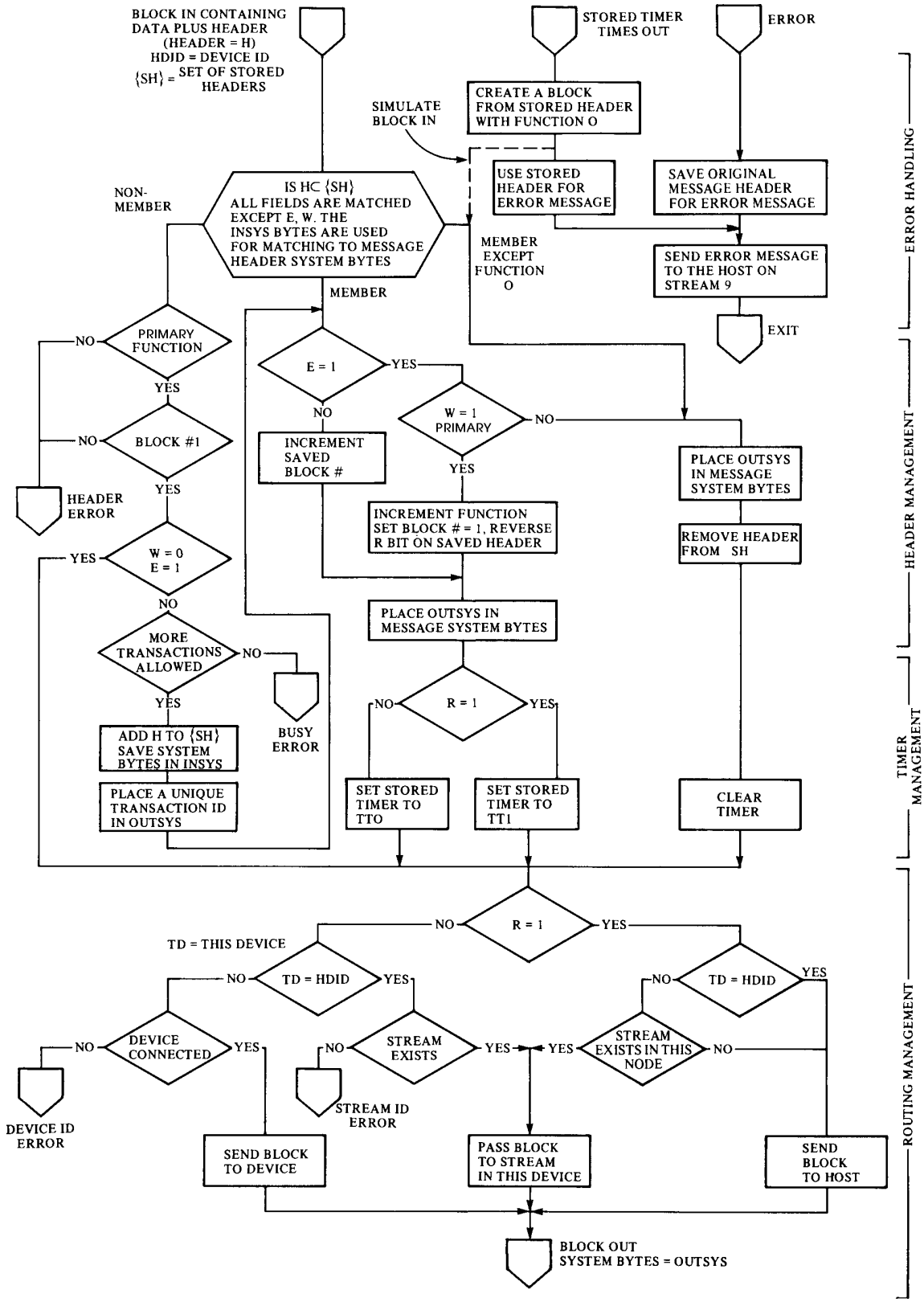

Figure R1-8

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.

SEMI E5-1104
SEMI EQUIPMENT COMMUNICATIONS STANDARD 2 MESSAGE CONTENT (SECS-II)

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current changes approved by the North American Regional Standards Committee on August 16, 2004. Initially available at www.semi.org September 2004; to be published November 2004. Originally published in 1982; last published July 2004.

CONTENTS

1 Purpose

1.1 The SEMI Equipment Communications Standard Part 2 (SECS-II) defines the details of the interpretation of messages exchanged between intelligent equipment and a host. This specification has been developed in cooperation with the Japan Electronic Industry Development Association Committee 12 on Equipment Communications.

1.1.1 It is the intent of this standard to be fully compatible with SEMI Equipment Communications Standard E4 (SECS-I). It is also the intent to allow for compatibility with alternative message transfer protocols. The details of the message transfer protocol requirements are contained in Section 6.

1.1.2 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.

1.1.3 The messages defined in the standard support the most typical activities required for IC manufacturing. The standard also provides for the definition of equipment-specific messages to support those activities not covered by the standard messages. While certain activities can be handled by common software in the host, it is expected that equipment-specific host software may be required to support the full capabilities of the equipment.

2 Scope

2.1 SECS-II gives form and meaning to messages exchanged between equipment and host using a message transfer protocol, such as SECS-I.

2.1.1 SECS-II defines the method of conveying information between equipment and host in the form of messages. These messages are organized into categories of activities, called streams, which contain specific messages, called functions. A request for information and the corresponding data transmission is an example of such an activity.

2.1.2 SECS-II defines the structure of messages into entities called items and lists of items. This structure allows for a self-describing data format to guarantee proper interpretation of the message.

2.1.3 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 SECS-II implementation.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard to establish appropriate safety and health practices and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 SECS-II applies to equipment and hosts used in the manufacturing of semiconductor devices. Examples of the activities supported by the standard are: transfer of control programs, material movement information, measurement data, summarized test data, and alarms.

3.1.1 The minimum compliance to this standard involves meeting the few constraints outlined in Section 8. It is expected that a given piece of equipment will require only a subset of the 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 Section 10.

3.1.2 It is assumed that the equipment will define the messages used in a particular implementation of SECS-II. It is assumed the host will support equipment implementation.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E4 — SEMI Equipment Communications Standard 1 Message Transfer (SECS-I)

SEMI E6 — Guide for Semiconductor Equipment Installation Documentation

4.2 *ANSI Standard*[1]

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

4.3 *IEEE Standard*[2]

IEEE 754 — Standard for Binary Floating Point Arithmetic

4.4 The Japan Electronic Industry Development Association (JEIDA) has requested that the SECS-II standard incorporate support for the JIS-8 codes for data exchange. This code would allow support for

1 American National Standards Institute, Headquarters: 1819 L Street, NW, Washington, DC 20036, USA. Telephone: 202.293.8020; Fax: 202.293.9287, New York Office: 11 West 42nd Street, New York, NY 10036, USA. Telephone: 212.642.4900; Fax: 212.398.0023, Website: www.ansi.org.

2 Institute of Electrical and Electronics Engineers, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, New Jersey 08855-1331, USA. Telephone: 732.981.0060; Fax: 732.981.1721, Website: www.ieee.org

katakana characters in Japanese implementations of SECS-II.

JIS 8-bit Coded Character Set (JIS-6226) for information Interchange, Japanese Industrial Standards. [3]

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 *Definitions*

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

5.1.2 *block* — a physical division of a message used by the message transfer protocol (see Section 6.3).

5.1.3 *conversation* — a sequence of related messages (see Section 8.4).

5.1.4 *conversation timeout* — an indication that a conversation has not completed properly (see Section 8.4.1).

5.1.5 *device ID* — a number between 0 and 32767 used in identifying the particular piece of equipment communicating with a host (see Section 6.4.1).

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

5.1.7 *function* — a specific message for a specific activity within a stream (see Section 7.2).

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

5.1.9 *interpreter* — the system that interprets a primary message and generates a reply when requested (see Section 6.2).

5.1.10 *item* — a data element within a message (see Section 9.2).

5.1.11 *item format* — a code used to identify the data type of an item (see Section 9.2).

5.1.12 *list* — a group of items (see Section 9.3).

5.1.13 *message* — a complete unit of communication (see Section 6.2).

5.1.14 *message header* — information about the message passed by the message transfer protocol (see Section 6.4).

5.1.15 *multi-block message* — a message sent in more than one block by the message transfer protocol (see Section 6.3.2).

5.1.16 *originator* — the creator of a primary message (see Section 6.2).

5.1.17 *packet* — a physical division of a message used by the message transfer protocol (see Section 6.3).

5.1.18 *primary message* — an odd numbered message. Also, the first message of a transaction (see Sections 6.2 and 7.2).

5.1.19 *reply* — the particular secondary message corresponding to a primary message (see Sections 6.2 and 7.2).

5.1.20 *secondary message* — an even-numbered message. Also the second message of a transaction (see Sections 6.2 and 7.2).

5.1.21 *single-block message* — a message sent in one block by the message transfer protocol (see Section 6.3.1).

5.1.22 *stream* — a category of messages (see Section 7.1).

5.1.23 *transaction* — a primary message and its associated secondary message, if any (see Section 8.2).

5.1.24 *transaction timeout* — an indication from the message transfer protocol that a transaction has not completed properly (see Section 6.5).

6 The Message Transfer Protocol

6.1 *Intent* — SECS-II is fully compatible with the message transfer protocol defined by SECS-I. It is the intent of this standard to allow for compatibility with alternative message transfer protocols. The purpose of this section is to define the requirements of the interaction between an application using SECS-II and the message transfer protocol. The methods used to implement these requirements are not covered as a part of this standard. The terms used in this standard are those used by SECS-I. Equivalent terms may be different for other message transfer protocols.

6.2 *Messages* — The message transfer protocol is used to send messages between equipment and host. The message transfer protocol must be capable of sending a primary message, indicating whether a reply is requested; and, if a reply is requested, it must be capable of associating the corresponding secondary message or reply message with the original primary message. The term originator will refer to the creator of the original primary message. The term interpreter will refer to the entity that interprets the primary

3 Japanese Industrial Standards. Available through the Japanese Standards Association, 1-24, Akasaka 4-Chome, Minato-ku, Tokyo 107-8440, Japan. Telephone: 81.3.3583.8005; Fax: 81.3.3586.2014. Website: www.jsa.or.jp

message at its destination and generates a reply when requested.

6.3 *Blocking Requirements* — The message transfer protocol must support the following SECS-II message blocking requirements.

6.3.1 *Single-Block Messages* — SECS-II requires that certain messages be sent in a single block or single packet by the message transfer protocol. Those messages defined in this standard as single-block SECS-II messages must be sent in a single-block or packet. The method used by the application software to tell the message transfer protocol that a particular message must be sent as a single block is not covered as part of this standard. For compatibility with SECS-I, the maximum length allowed for a single-block SECS-II message is 244 bytes. The minimum requirement for the message transfer protocol is to be able to send single-block SECS-II messages.

6.3.2 *Multi-Block Messages* — For compatibility with SECS-I, SECS-II messages that are longer than 244 bytes are referred to as multi-block messages. Also, certain SECS-II messages are allowed to be multi-block messages even if they otherwise meet the single-block length requirements. 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.

6.4 *Message Header* — The message transfer protocol must provide the following information, called the message header, with every message. Only the content of the message header is defined by this standard. The exact format of the message header passed between the application and the message transfer protocol is not covered as part of this standard.

NOTE 1: In SECS-I, this information is contained in the 10-byte header of each block of a message.

6.4.1 *Device ID* — The message transfer protocol must be capable of identifying the device ID (0–32767) which indicates the source or destination of a message.

6.4.2 *Stream and Function* — The message transfer protocol must be capable of identifying to SECS-II a minimum15-bit message identification code. In SECS-II, messages are identified by a stream code (0–127, 7 bits) and a function code (0–255, 8 bits). Each combination of stream and function represents a distinct message identification.

6.4.3 *Reply Requested* — The message transfer protocol must be capable of identifying whether a reply is requested to a primary message.

6.5 *Transaction Timeout* — It is presumed that the message transfer protocol will notify SECS-II in the event of failure to receive an expected reply message within a specified transaction timeout period.

NOTE 2: In SECS-I, a transaction timeout occurs if either the reply timeout (T3) is exceeded before the first block of a reply message is received or if the inter-block timeout (T4) is exceeded before an expected block of a multi-block message is received.

6.6 *Multiple Open Transactions* — This standard allows, but does not require, the support of more than one concurrent open transaction.

7 Streams and Functions

7.1 *Streams* — A stream is a category of messages intended to support similar or related activities.

7.2 *Functions* — A function is a specific message for a specific activity within a stream. All the functions used in SECS-II will follow a numbering convention corresponding to primary and secondary message pairs. All primary messages will be given an odd-numbered function code. The reply message function code is determined by adding one to the primary message function code. The even-numbered function following a primary message which requests no reply is reserved and is not to be used. Function code 0 is reserved in all streams for aborting transactions as described in Section 10.4.

7.3 *Stream and Function Allocation* — Some of the stream and function code combinations are reserved for this standard, while others are available for user definition. The stream and function codes reserved for this standard are as follows:

In Stream 0, Functions 0–255.

In Streams 1–63, Functions 0–63.

In Streams 64–127, Function 0.

The stream and function codes available for user definition are as follows:

In Streams 1–63, Functions 64–255.

In Streams 64–127, Functions 1–255.

7.3.1 The stream and function code assignment can also be represented by the diagram shown in Figure 1.

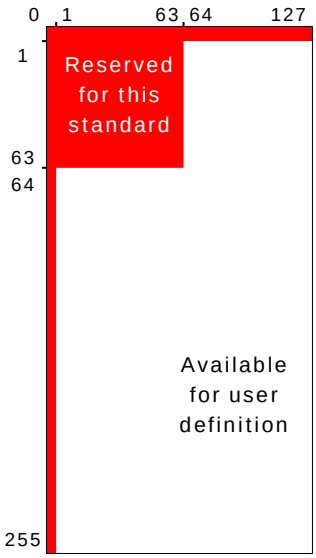

Figure 1
Stream and Function Allocation

7.3.2 The reserved codes assigned by this standard are listed in Section 10. It is recognized that there will be user needs beyond the specific definitions given in this standard. In these situations, the streams and functions reserved for user definition should be used subject to the guidelines for minimum compliance outlined in Section 8.

8 Transaction and Conversation Protocols

8.1 *Intent* — For an implementation to be in compliance with SECS-II, it must meet the minimum transaction requirements outlined in this section. The conversation protocols serve to further define the use and interaction between transactions.

8.2 *Transaction Definition* — A transaction forms the basis for all information exchanges in SECS-II. A transaction consists of either a primary message for which no reply is requested, or a primary message which requests a reply together with its corresponding secondary message. Secondary messages cannot request a reply.

8.3 *Transaction Level Requirements* — The following are the requirements to comply with the SECS-II protocol at the transaction level:

1. Respond to S1F1 with S1F2 as described in Section 10.5.
2. For any received message that cannot be processed by the equipment, send the appropriate error message on Stream 9. As described in Section 10.13, S9F1, F3, F5, F7, or F11 are possible.
3. Format any other supported messages according to Section 10.
4. Upon detection of a transaction timeout at the equipment, send S9F9 to the host.
5. Upon receipt of function 0 as a reply to a primary message, terminate the related transaction. No error message should be sent to the host by the equipment.

8.4 *Conversation Protocols* — A conversation is a series of one or more related transactions used to complete a specific task. A conversation should include all transactions necessary to accomplish the task and leave both the originator and interpreter free of resource commitments at its conclusion.

8.4.1 *Conversation Timeout* — A conversation timeout is used to indicate that a conversation has not completed properly. A conversation timeout is application-dependent, and the methods used for detecting conversation timeouts are not covered as part of this standard. A conversation timeout will terminate further action on the conversation, and will allow for the clearing of any committed resources. Upon detection of a conversation timeout at the equipment, S9F13 should be sent to the host.

8.4.2 *Types of Conversations* — There are seven types of conversations which characterize all information exchanges in SECS-II:

1. A primary message with no reply is the simplest conversation. This message must be a single-block SECS-II message. 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 primary message and the data returned to the originator as a reply message. It is assumed that the originator requesting the data is prepared to receive the amount of data returned. This is the request/data conversation.
3. If the originator wishes to send data in a single-block SECS-II message to the interpreter, then the originator sends the data and expects an acknowledgment from the interpreter. This is the send/acknowledge conversation.
4. If the originator has a multi-block SECS-II message to send for a particular exchange, then the originator must receive permission from the interpreter prior to sending the data. The first transaction requests permission to send, and the interpreter either grants or denies permission. If

permission is granted, the originator sends the data and the interpreter replies appropriately. This is the inquire/grant/send/acknowledge conversation. Between the inquire 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 dependent upon the application, at which time the interpreter will free its resources and send an S9,F13 error message to the originator. Note that under the definition of S9,F13 in this standard, only the equipment should generate an error message to the host under these conditions.

5. There is a conversation related to the transfer of unformatted data sets between equipment and host. This conversation is described in detail in Stream 13. (See Section 10.17)

6. There is a conversation related to the handling of material between equipment. This conversation is described in detail in Stream 4. (See Section 10.8)

7. The originator may request information from the interpreter which requires some time to obtain (e.g., operator input is required). The first transaction requests the information and the interpreter responds in one of three ways: 1) the information is returned, 2) the interpreter indicates that the information cannot or will not be obtained, or 3) the interpreter indicates that the information will be obtained and returned in a subsequent transaction, as specified for this conversation. For case number 3, the interpreter will initiate the subsequent transaction when the information is available.

8.4.2.1 Case 3 is the request/acknowledge/send/acknowledge transaction.

8.4.2.2 The originator of the request/acknowledge/send/acknowledge conversation may commit some resources in anticipation of the send/acknowledge transaction. Consequently, a conversation timeout may be set by the originator at a time dependent on the application. On timeout, the originator will free its resources and restart the conversation with the 'request', or send an S9,F13 error message. Note that under the definition of S9,F13 in this standard, only the equipment should generate an error message to the host under these conditions.

8.4.3 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.

9 Data Structures

9.1 *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 transfer protocol. They are intended to provide a self-describing internal structure to messages passed between equipment and host.

9.2 *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.

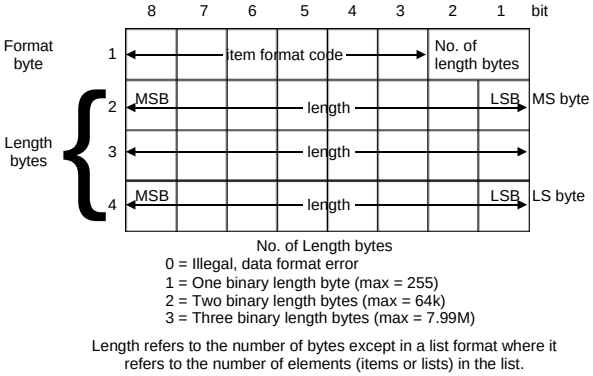

Figure 2
Item and List Header

9.2.1 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.

9.2.2 Bits 3 through 8 of the format byte of the item header define the format of the data which follows. Of the 64 possible formats, 16 are defined as shown in Table 1. Format code 0 is called a list and is defined in Section 9.3. Format code 22 (octal) is called a localized string and is defined in Section 9.4. The remaining 14 item formats define unspecified binary, code 10 (octal); Boolean, code 11 (octal); ASCII character strings, code 20 (octal); JIS-8 character strings, code 21 (octal)

signed integer, codes 30, 31, 32, 34 (octal); floating point, codes 40, 44 (octal); and unsigned integer, codes 50, 51, 52, 54 (octal). These formats are used for groups of data which have the same representation in order to save repeated item headers. In signed integers, negative values will be two's complement values. Floating point numbers will conform to the IEEE standard 754. Boolean values will be byte quantities, with zero being equivalent to false, and non-zero being equivalent to true.

9.3 *List* — A list is an ordered set of elements, where an element can be either an item (Section 9.2) 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.

9.3.1 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, which is defined in the detailed message definitions.

Table 1 Item Format Codes

Format Code (Bit 876543)		*Meaning*
Binary	*Octal*	*Data after the heading has the following form*
000000	00	LIST (length in elements)
001000	10	Binary
001001	11	Boolean
010000	20	ASCII[1]
010001	21	JIS-8
010010	22	2-byte character[2, 4]
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 754. The byte containing the sign bit is sent first.

[4] The code for Multi-byte character must be specified in the data in the first 2 bytes of the TEXT item.

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

9.4 *Localized Character String Items* — A localized character string is an item which is used for representing a string of multi-byte characters. Because there are many different encoding schemes and the information could be in any one of a number of languages, these characteristics must also be included in the item. Thus for localized character strings which use item format code 22 (octal), there is an additional localized string header (LSH) .

9.4.1 This localized string header follows the item header and precedes the string. The localized string header is part of the item data, thus the length of the header (2 bytes) is included in the length in the item header. The length of the localized string itself is the number of bytes that it occupies, regardless of the number of characters that represents the string. The localized string header followed by the string together comprise the localized string item. For example, a 2 byte localized string (which may represent a single character), because of the 2 byte length of the localized string header, will have a 4 byte length in the item header.

9.4.2 The LSH is a 16 bit number which specifies the encoding method used for the string. Defined values for the encoding are as follows:

Table 2

Encoding Code (Decimal)	*Encoding Scheme*	*Notes*
0	none	reserved
1	ISO 10646 UCS-2	Unicode 2.0
2	UTF-8	Transformation of ISO 10646 UCS-2
3	ISO 646-1991	ASCII, 7-bit
4	ISO 8859-1	ISO Latin-1, Western Europe
5	ISO 8859-11 (proposed)	Thai
6	TIS 620	Thai (will be supported by ISO 8859-11)
7	IS 13194 (1991)	ISCII
8	Shift JIS	
9	Japanese EUC-JP	
10	Korean EUC-KR	
11	Simplified Chinese GB	
12	Simplified Chinese EUC-CN	
13	Traditional Chinese Big5	
14	Traditional Chinese EUC-TW	

9.4.3 Encoding Codes from 15 to 32767 are reserved for future expansion. Encoding codes from 32768 to 65535 are available for custom purposes.

9.5 *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, binary, 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 signed integer form.

```
01101001     Item, 2-byte integers
00000110     6 bytes total (6/2 = 3 integers)
xxxxxxxx     MSByte number x
xxxxxxxx     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 754
ffffffff
```

e. A message is sent from device 66 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 (e.g., 17). The third item is a string of text giving a brief description of the alarm (e.g., "T1 HIGH.") No reply is requested. The complete message including the header is as follows:

```
10000000     R = 1 (to the host)
01000010     Device Code = 66
00000101     Stream 5, W = 0
00000001     Function 1
10000000     E = 1
00000001     Block 1
00000000
00000000     System bytes = 0
00000000
00000000
00000001     List
00000011     3 Elements
00100001     Binary Item next byte length
00000001     1 byte long
00000100     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 1
00100000     ASCII space
01001000     ASCII H
01001001     ASCII I
01000111     ASCII G
01001000     ASCII H
```

9.5.1 Using E4 SECS-I transmission, the entire message contains 1-byte length (not shown), 10 bytes of header, 17 bytes of data, and 2 bytes checksum (not shown) for a total of 30 bytes. At 9600 baud transmission, the message would be sent in 31 milliseconds.

9.6 *Data Item Dictionary* — This section defines the data items used in the standard SECS-II messages described in Section 10, Message Detail.

Name: A unique mnemonic name for this data item. This name is used in message definitions.

Format: The allowable item format codes which can be used for this standard data item. Item format codes are shown in octal, as described in Table 1, Item Format Codes. The notation "3()" indicates any of the signed integer formats (30, 31, 32, 34). The notation "4()" indicates any of the floating point formats (40, 44). The notation "5()" indicates any of the unsigned integer formats (50, 51, 52, 54). The notation "0" indicates that a list with user-defined structure may be used. Where more than one format is shown, a given implementation can use any of the formats specified.

Description: A description of the data item, with the meanings of specific values.

Where Used: The standard messages in which this data item appears.

Table 3 Data Item Dictionary

Name	*Format*	*Description*	*Values*	*Where Used*
ABS	10	Any binary string		S2F25, F26
ACCESSMODE	51	Load Port Access Mode. Possible values are:	0 = Manual 1 = Auto	S3F27

Name	*Format*	*Description*	*Values*	*Where Used*
ACDS	32, 52	After Command Codes	Vector of all command codes which defined command must succeed within the same block.	S7F22
ACKA	11	Indicates success of a request:	TRUE is successful else FALSE	S5F14, F15, F18; S16F2, F4, F6, F12, F14, F16, F18, F24, F26, F28, F30; S17F4, F8, F14
ACKC3	10	Acknowledge code, 1 byte	0 = Accepted >0 = Error, not accepted 1-63 Reserved	S3F6, F8, F10
ACKC5	10	Acknowledge code, 1 byte	0 = Accepted >0 = Error, not accepted 1-63 Reserved	S5F2, F4
ACKC6	10	Acknowledge code, 1 byte	0 = Accepted >0 = Error, not accepted 1-63 Reserved	S6F2, F4, F10, F12, F14
ACKC7	10	Acknowledge code, 1 byte	0 = Accepted 1 = Permission not granted 2 = Length error 3 = Matrix overflow 4 = PPID not found 5 = Mode unsupported 6 = Command will be performed with completion signaled later >6 = Other error 7-63 Reserved	S7F4, F12, F14, F16, F18, F24, F32, S7F38, F40, F42, F44
ACKC7A	31, 51	Acknowledge Code, 1 byte	0 = Accepted 1 = MDLN is inconsistent 2 = SOFTREV is inconsistent 3 = Invalid CCODE 4 = Invalid PPARM value 5 = Other error (described by ERRW7) 6-63 Reserved	S7F27
ACKC10	10	Acknowledge Code, 1 byte	0 = Accepted for display 1 = Message will not be displayed 2 = Terminal not available 3-63 Reserved	S10F2, F4, F6, F10
ACKC13	10	Return code for secondary messages 1 byte	0 = O.K. 1 = ERROR: Try Later 2 = ERROR: Unknown Data Set name 3 = ERROR: Illegal Checkpoint value 4 = ERROR: Too many open Data Sets 5 = ERROR: Data set open too many times 6 = ERROR: No open Data Set 7 = ERROR: Cannot continue 8 = ERROR: End of Data 9 = ERROR: Handle in Use >10 = ERROR: Pending Transaction 11-127 Reserved	S13F2, F4, F6, F8

Name	*Format*	*Description*	*Values*	*Where Used*
ACKC15	10	Return code for secondary messages, 1 byte	0 = Accepted 1 = command will be performed with completion signaled later 2 = DSNAME not found 3 = Permission not granted 4 = other error 5-63 Reserved	S15F50,F52
AGENT	20			S15F11, F12, F21, F22, F25
ALCD	10	Alarm code byte	bit 8 = 1 means alarm set bit 8 = 0 means alarm cleared bit 7-1 is alarm category 0 = Not used 1 = Personal safety 2 = Equipment safety 3 = Parameter control warning 4 = Parameter control error 5 = Irrecoverable error 6 = Equipment status warning 7 = Attention flags 8 = Data integrity >8 = Other categories 9-63 Reserved	S5F1, F6
ALED	10	Alarm enable/disable code, 1 byte	bit 8 = 1 means enable alarm bit 8 = 0 means disable alarm	S5F3
ALID	3(), 5()	Alarm identification		S5F1, F3, F5, F6
ALTX	20	Alarm text limited to 40 characters		S5F1, F6
ATTRDATA	0, 10, 20, 3(), 4(), 5(), 11	Contains a specific attribute value for a specific object		S1F20; S3F17, F18; S13F14, F16; S14F1, F2, F3, F4, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18; S18F2, F3
ATTRID	20, 5()	Identifier for an attribute for a specific type of object		S1F19; S3F17, F18; S13F14, F16; S14F1, F2, F3, F4, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18; S18F1, F3

Name	*Format*	*Description*	*Values*	*Where Used*
ATTRRELN	51	The relationship that a specified qualifying value has to the value of an attribute of an object instance (the value of interest):	0 = The qualifying value is equal to the value of interest, 1 = The qualifying value is not equal to the value of interest, 2 = The qualifying value is less than the value of interest, 3 = The qualifying value is less than or equal to the value of interest, 4 = The qualifying value is greater than the value of interest, 5 = The qualifying value is greater than or equal to the value of interest, 6 = The qualifying value is present (contained in the set of) the value of interest, 7 = The qualifying value is absent (not contained in the set of) the value of interest, >7 = Reserved.	S14F1
BCDS	32, 52	Before Command Codes	Vector of all command codes which defined command must precede within the same block.	S7F22
BCEQU	20, 51	Bin code equivalents	Array of all codes that are to be processed. Must be the same format as BINLT and NULBC. Zero length indicates all codes be sent.	S12F3, F4
BINLT	20, 51	The Bin List	Is an array of bin values. Format must be the same as used in NULBC and BCEQU.	S12F7, F9, F11, F14, F16, F18
BLKDEF	31, 51	Block Definition	Blocks define the range for before/after code checking (specified by BCDS, IBCDS, NBCDS, ACDS, IACDS, and NACDS). BLKDEF specifies whether the command being defined starts a block (+1), terminates a block (-1), or is within the body of a block (0). All other values are invalid. The outermost block of a process program is implicit, and is bounded by the first and last command of the process program. Before/after checks for a particular command are performed with all other commands in the same block and at the same nesting level. For the purpose of before/after checking, a set of commands making up a block is considered to be a single body command of its containing block. This command has the before/after restrictions of the start block command which begins the block.	S7F22
BPD	10	Boot program Data		S8F2

Name	*Format*	*Description*	*Values*	*Where Used*
BYTMAX	3(), 5()	Byte Maximum	Maximum length of process program. A value of zero indicates no limit. Negative values are invalid.	S7F22
CAACK	51	Carrier Action Acknowledge Code, 1 byte	0 = Acknowledge, command has been performed. 1 = Invalid command 2 = Can not perform now 3 = Invalid data or argument 4 = Acknowledge, request will be performed with completion signaled later by an event. 5 = Rejected. Invalid state. 6 = Command performed with errors. 7-63 Reserved.	S3F18, F20, F22, F24, F26, F30, F32
CARRIERACTION	20	Specifies the action requested for a carrier		S3F17
CARRIERID	20	The identifier of a carrier		S3F17
CARRIERSPEC	20	The object specifier for a carrier. Conforms to OBJSPEC.		F29, F31
CATTRDATA	0, 10, 20, 3(), 4(), 5(), 11	The value of a carrier attribute		S3F17
CATTRID	20	The name of a carrier attribute		S3F17
CCODE	20, 32, 34, 52, 54	Command Code	Each command code corresponds to a unique process operation the machine is capable of performing.	S7F22, F23, F26, F31, F39, F43
CEED	11	Collection event or trace enable/disable code, 1 byte	FALSE = Disable TRUE = Enable	S2F37; S17F5
CEID	20, 3(), 5()	Collected event ID		S2F35, F37; S6F3, F8, F9, F11, F13, F15, F17; S17F5, F9, F10, F11, F12
CEPACK	0, 51	Command Enhanced Parameter Acknowledge. If a specific value of CPNAME is defined to have a CEPVAL that is a LIST, then CEPACK shall have the same structure as the corresponding list format of CEPVAL as used in S2,F49. Otherwise CEPACK will be a 1 byte integer. Enumerated:	0 = No error 1 = Parameter name (CPNAME) does not exist 2 = Illegal value specified for CEPVAL 3 = Illegal format specified for CEPVAL 4 = Parameter name (CPNAME) not valid as used 5-63 Reserved	S2F50

Name	*Format*	*Description*	*Values*	*Where Used*
CEPVAL	0, 10, 11, 20, 21, 3(), 4(), 5()	Command Enhanced Parameter Value. A specific application of CEPVAL shall always be identified with a specific value of CPNAME. A CEPVAL has the following forms: a single (non-list) value (e.g., CPVAL), a list of single items of identical format and type, or a list of items of the form.	L,2 1. CPNAME 2. CEPVAL	S2F49
CKPNT	54	Checkpoint as defined by the sending system		S13F3, F6
CMDA	31, 51	Command acknowledge code	0 = Completed or done 1 = Command does not exist 2 = Cannot perform now >2 = Other equipment-specific error 3-63 Reserved	S2F22, F28
CMDMAX	3(), 5()	Command Maximum	Maximum number of commands to be allowed in a process program. A value of zero indicates no limit. Negative values are invalid.	S7F22
CNAME	20	Command Name = 16 characters	Text string unique among other CNAMEs in PCD which describes the processing done by the equipment for the corresponding CCODE.	S7F22
COLCT	5()	Column count in die increments		S12F1, F4
COLHDR	20	Text description of contents of TBLELT. 1–20 characters		S13F13, F16
COMMACK	10	Establish Communications Acknowledge Code, 1 byte	0 = Accepted 1 = Denied, Try Again 2-63 Reserved	S1F14
CONDITION	20	Provides condition information for a subsystem component. Used in the data item in the CONDITIONLIST.		See CONDITION-LIST
CONDITIONLIST	0	A list of CONDITION data sent in a fixed order. CONDITIONLIST has the following form:	L,s 1.<CONDITION1> . s.<CONDITIONs>	S18F16
CPACK	10	Command Parameter Acknowledge Code, 1 byte	1 = Parameter Name (CPNAME) does not exist 2 = Illegal Value specified for CPVAL 3 = Illegal Format specified for CPVAL >3 = Other equipment-specific error 4-63 Reserved	S2F42

Name	*Format*	*Description*	*Values*	*Where Used*
CPNAME	20, 3(), 5()	Command Parameter Name		S2F41, F42, F49, F50; S4F21, F29; S16F5, F27
CPVAL	10, 11, 20, 21, 3(), 5()	Command Parameter Value		S2F41, F49; S4F21, F29; S16F5, F27; S18F13
CSAACK	10	Equipment Acknowledgement code, 1 byte	0 = Everything correct 1 = Busy 2 = Invalid SPID 3 = Invalid data >3 = Equipment-specific error 4-63 Reserved	S2F8
CTLJOBCMD	51	Control Job command codes are assigned as follows:	1 = CJStart 2 = CJPause 3 = CJResume 4 = CJCancel 5 = CJDeselect 6 = CJStop 7 = CJAbort 8 = CJHOQ	S16F27
CTLJOBID	20	Identifier for Control Job. Conforms to OBJID.		S16F27
DATA	20	A vector or string of unformatted data		S3F29, F31; S18F6, F7
DATAACK	10	Acknowledge code for data	0 = Acknowledge 1 = Unknown DATAID 2 = At least parameter is invalid 3–63 Reserved	S14F22
DATAID	20, 3(), 5()	Data ID		S2F33, F35, F39, F45, F49; S3F15, F17; S4F19, F25; S6F3, F5, F7, F8, F9, F11, F13, F16, F18, F27; S13F11, F13, F15; S14F19, F21, F23; S15F27, F29, F33, F35, F37, F39, F41, F43, F45, F47, F49; S16F1, F3, F5, F11, F13; S17F1, F5, F9
DATALENGTH	3(), 5()	Total bytes to be sent		S2F39; S3F15, F29, F31; S4F25; S6F5; S13F11; S14F23; S16F1, F11; S18F5, F7

Name	*Format*	*Description*	*Values*	*Where Used*
DATASEG	20	Used to identify the data requested		S3F29, F31; S18F5, F7
DATASRC	20	Object type for Data Source Objects		S14F1, F3, F6, F7, F8; S17F1
DATLC	51	Data location	Location of invalid data, represented in bytes, measured from the start of the message in question, excluding all header bytes.	S12F19
DRACK	10	Define Report Acknowledge Code, 1 byte	0 = Accept 1 = Denied. Insufficient space 2 = Denied. Invalid format 3 = Denied. At least one RPTID already defined 4 = Denied. At least VID does not exist >4 = Other errors 5-63 Reserved	S2F34
DSID	20, 3(), 5()	Data set ID		S6F3, F8, F9
DSNAME	20	The name of the Data Set	The minimum length is zero. The maximum is 200 characters.	S13F1, F2, F3, F4, S7F37, F39, F41, F43; S15F49, F51
DSPER	20	Data sample period. DSPER has two allowable formats:	Format 1: hhmmss, 6 bytes Format 2: hhmmsscc, 8 bytes Where “hh” is hours, “mm” is minutes, “ss” is seconds’ and “cc” is centiseconds. Equipment shall either (1) support only Format 1, or (2) support both Format 1 and Format 2. Equipment shall document which formats it accepts. Equipment which supports Format 2 need not necessarily support a minimum DSPER of 1 centisecond, nor a trace resolution of 1 centisecond, but equipment suppliers shall document its trace performance limits.	S2F23
DUTMS	20	Die Units of Measure	Use units description per SEMI E5 Section 12.	S12F1, F4
DVNAME	3(), 20, 5()	Data value name		S6F3, F8
DVVAL	0, 10, 11, 20, 21, 3(), 4(), 5()	Data value		S6F3, F8, F9

Name	*Format*	*Description*	*Values*	*Where Used*
EAC	10	Equipment acknowledge code, 1 byte	0 = Acknowledge 1 = Denied. At least one constant does not exist 2 = Denied. Busy 3 = Denied. At least one constant out of range >3 = Other equipment-specific error 4-63 Reserved	S2F16
ECDEF	10, 11, 20, 21, 3(), 4(), 5()	Equipment constant default value		S2F30
ECID	3(), 20, 5()	Equipment Constant ID		S2F13, F15, F29, F30
ECMAX	10, 11, 20, 21, 3(), 4(), 5()	Equipment constant maximum value		S2F30
ECMIN	10, 11, 20, 21, 3(), 4(), 5()	Equipment constant minimum value		S2F30
ECNAME	20	Equipment constant name		S2F30
ECV	10, 11, 20, 21, 3(), 4(), 5()	Equipment Constant Value		S2F14, F15
EDID	10, 20, 3(), 5()	Expected data Identification	Three possible responses. MEXP EDID EDID S02F03, <SPID> A[6] S03Fl3, <PTN> B[1] S07F03, <PPID> A[80], B[80]	S9F13
EMID	10, 20	Equivalent material ID (16 bytes maximum)		S3F9
EPD	10	Executive program data		S8F4
EQNAME	20	A unique ASCII equipment identifier assigned by the factory to the equipment. Limited to a maximum of 80 characters.		S4F29
ERACK	10	Enable/Disable Event Report Acknowledge Code, 1 byte	0 = Accepted 1 = Denied. At least one CEID does not exist >1 = Other Errors 2-63 Reserved	S2F38
ERRCODE	5()	Code identifying an error	0 = No error 1 = Unknown object in Object Specifier 2 = Unknown target object type 3 = Unknown object instance 4 = Unknown attribute name 5 = Read-only attribute - access denied 6 = Unknown object type 7 = Invalid attribute value 8 = Syntax error	S1F20; S3F16, F30, F32; S4F20, F22, F23, F33, F35; S5F14, F15, F18; S13F14, F16; S14F2, F4, F6, F8, F10, F12, F14, F16, F18,

Name	Format	Description	Values	Where Used
			9 = Verification error 10 = Validation error 11 = Object identifier in use 12 = Parameters improperly specified 13 = Insufficient parameters specified 14 = Unsupported option requested 15 = Busy 16 = Not available for processing 17 = Command not valid for current state 18 = No material altered 19 = Material partially processed 20 = All material processed 21 = Recipe specification related error 22 = Failed during processing 23 = Failed while not processing 24 = Failed due to lack of material 25 = Job aborted 26 = Job stopped 27 = Job cancelled 28 = Cannot change selected recipe 29 = Unknown event 30 = Duplicate report ID 31 = Unknown data report 32 = Data report not linked 33 = Unknown trace report 34 = Duplicate trace ID 35 = Too many data reports 36 = Sample period out of range 37 = Group size to large 38 = Recovery action currently invalid 39 = Busy with another recovery currently unable to perform the recovery 40 = No active recovery action 41 = Exception recovery failed 42 = Exception recovery aborted 43 = Invalid table element 44 = Unknown table element 45 = Cannot delete predefined 46 = Invalid token 47 = Invalid parameter 48 = Load port does not exist 49 = Load port already in use 50 = Missing Carrier 51-63 = Reserved (data formats 51, 52, 54, or 50 must be used) 64-32767 = User defined (data formats 52, 54, or 50 must be used) 32768 = Action will be performed at earliest opportunity	F26, F28; S15F28, F18, F20, F22, F24, F26, F30, F32, F34, F36, F38, F40, F42, F44, F48, F53; S16F12, F14, F16, F18, F24, F26, F28; S17F2, F4, F6, F8, F10, F12, F14

Name	*Format*	*Description*	*Values*	*Where Used*
			32769 = Action can not be performed now 32770 = Action failed due to errors 32771 = Invalid command 32772 = Client Already Connected 32773 = Duplicate ClientID 32774 = Invalid ClientType 32775 = IncompatibleVersions 32776 = Unrecognized ClientID (Client not currently connected) 32777 = Failed (Completed Unsuccessfully) 32778 = Failed (Unsafe) – External intervention required 32779 = Sensor-Detected Obstacle 32780 = Material Not Sent 32781 = Material Not Received 32782 = Material Lost 32783 = Hardware Failure 32784 = Transfer Cancelled 32785–32792 reserved for future use by SEMI E127 service requests. 32793-65335 Reserved (data formats 52, 54, or 50 must be used) 65536 or above = User defined (data formats 54 or 50 must be used)	
ERRTEXT	20	Text string describing the error noted in the corresponding ERRCODE. Limited to 80 characters maximum.		S1F20; S3F16, F18, F20, F22, F24, F26, F30, F32; S4F20, F22, F23, F33, F35; S5F14, F15, F18; S13F14, F16; S14F2, F4, F6, F8, F10, F12, F14, F16, F18, F26, F28; S15F28, F30, F32, F34, F36, F38, F40, F42, F44, F48, F53; S16F12, F14, F16, F18, F24, F26, F28; S17F4, F8, F14
ERRW7	20	Text string describing error found in process program.		S7F27
EVNTSRC	20	Object type for Event Source Objects		S17F5, F9, F10, F11, F12
EXID	20	Unique identifier for the exception. Maximum length of 20 characters.		S5F9, F11, F13, F14, F15, F17, F18

Name	*Format*	*Description*	*Values*	*Where Used*
EXMESSAGE	20	Text which describes the nature of the exception.		S5F9, F11
EXRECVRA	20	Text which specifies a recovery action for an exception. Maximum length of 40 bytes.		S5F9, F13
EXTYPE	20	Text which identifies the type of an exception. It is usually a single word of text.	"ALARM" "ERROR"	S5F9, F11; S14F1, F2, F8
FCNID	51	Function Identification		S2F43, F44
FFROT	52	Film Frame Rotation	In degrees from the bottom CW. (Bottom equals zero degrees.) Zero length indicates not used.	S12F1, F3
FILDAT	10, 20	Data from the Data Set	The maximum length is the RECLEN from Open Data Set Data.	S13F6
FNLOC	52	Flat/Notch Location	In degrees from the bottom CW. (Bottom equals zero degrees.) Zero length indicates not used.	S12F1, F3, F4
FRMLEN	3(), 5()	Formatted Process Program Length	If greater than zero, indicates PPID is available as a formatted process program and its length in bytes. If zero, the PPID is not available as a formatted program. Negative values are invalid.	S7F34
GRANT	10	Grant code, 1 byte	0 = Permission Granted 1 = Busy, Try Again 2 = No Space Available 3 = Duplicate DATAID >3 = Equipment Specific Error Code 4-63 Reserved	S2F2, F40; S3F16; S4F26; S13F12; S14F24; S16F2, F12
GRANT6	10	Permission to send, 1 byte	0 = Permission granted 1 = Busy, try again 2 = Not interested >2 = Other errors 3-63 Reserved	S6F6
GRNT1	10	Grant code, 1 byte	0 = Positive response, transfer ok 1 = Busy, try again 2 = No space 3 = Map too large 4 = Duplicate ID 5 = Material ID not found 6 = Unknown map format >6 = Error 7-63 Reserved	S12F6
HANDLE	3(), 5()	Logical unit or channel		S13F3, F4, F5, F6, F7, F8

Name	*Format*	*Description*	*Values*	*Where Used*
HCACK	10	Host Command Parameter Acknowledge Code, 1 byte	0 = Acknowledge, command has been performed 1 = Command does not exist 2 = Cannot perform now 3 = At least one parameter is invalid 4 = Acknowledge, command will be performed with completion signaled later by an event 5 = Rejected, Already in Desired Condition 6 = No such object exists 7-63 Reserved	S2F42, F50
HOACK	11	Conveys whether the corresponding handoff activity succeeded (= True) or failed (= False)		S4F31, F33
HOCANCELACK	51	Tells whether the cancel ready message was accepted or rejected	0 = Cancel Ready Accepted 1 = Atomic Transfer Unknown 2 = Cancel Ready Rejected - Handoff Begun	S4F13
HOCMDNAME	20	Identifier for the handoff command to be executed		S4F31
HOHALTACK	51	Tells whether the halt command was accepted or rejected	0 = Halt Accepted 1 = Atomic Transfer Unknown 2-63 Reserved	S4F41
IACDS	32, 52	Immediately After Command Codes	Vector of all command codes which defined command must immediately succeed within the same block.	S7F22
IBCDS	32, 52	Immediately Before Command Codes	Vector of all command codes which defined command must immediately precede within the same block.	S7F22
IDTYP	10	Id type	0 = Wafer ID 1 = Wafer Cassette ID 2 = Film Frame ID >2 = Error 3-63 Reserved	S12F1, F3, F4, F5, F7, F9, F11, F13, F14, F15, F16, F17, F18
INPTN	10, 51	A specialized version of PTN indicating the InputPort.		S3F35
JOBACTION	20	Specifies the action for a ReticleTransferJob		S3F35
LENGTH	3(), 5()	Length of the service program or process program in bytes		S2F1; S7F1, F29
LIMITACK	10	Acknowledgment code for variable limit attribute set, 1 byte	1 = LIMITID does not exist 2 = UPPERDB > LIMITMAX 3 = LOWERDB < LIMITMIN 4 = UPPERDB < LOWERDB 5 = Illegal format specified for UPPERDB or LOWERDB 6 = ASCII value cannot be translated to numeric 7 = Duplicate limit definition for this variable >7 = Other equipment-specific error 8-63 Reserved	S2F46

Name	*Format*	*Description*	*Values*	*Where Used*
LIMITID	10	The identifier of a specific limit in the set of limits (as defined by UPPERDB and LOWERDB) for a variable to which the corresponding limit attributes refer, 1 byte.		S2F45, F46, F48
LIMITMAX	11, 20, 3(), 4(), 5()	The maximum allowed value for the limit values of a specific variable. The equipment manufacturer should specify this value, which would typically coincide with the maximum value of the variable being monitored. The format must match that of the referenced variable.		S2F48
LIMITMIN	11, 20, 3(), 4(), 5()	The minimum allowed value for the limit values of a specific variable. The equipment manufacturer should specify this value, which would typically coincide with the minimum value of the variable being monitored. The format must match that of the referenced variable.		S2F48
LINKID	54	Used to link a completion message with a request that an operation be performed. LINKID is set to the value of RMOPID in the initial request except for the last completion message to be sent, where it is set to zero.		S6F25; S14F20, F21; S15F22, F30
LLIM	3(), 4(), 5()	Lower limit for numeric value		S7F22
LOC	10	Machine material location code, 1 byte		S2F27; S3F2
LOCID	20	The logical identifier of a material location.		S3F29, F31
LOWERDB	11, 20, 3(), 4(), 5()	A variable limit attribute which defines the lower boundary of the deadband of a limit. The value applies to a single limit (LIMITID) for a specified VID. Thus, UPPERDB and LOWERDB as a pair define a limit.		S2F45, F48

Name	*Format*	*Description*	*Values*	*Where Used*
LRACK	10	Link Report Acknowledge Code, 1 byte	0 = Accepted 1 = Denied. Insufficient space 2 = Denied. Invalid format 3 = Denied. At least one CEID link already defined 4 = Denied. At least one CEID does not exist 5 = Denied. At least one RPTID does not exist >5 = Other errors 6-63 Reserved	S2F36
LVACK	10	Variable limit definition acknowledge code, 1 byte. Defines the error with the limit attributes for the referenceVID.	1 = Variable does not exist 2 = Variable has no limits capability 3 = Variable repeated in message 4 = Limit value error as described in LIMITACK 5-63 Reserved	S2F46
MAPER	10	Map Error	0 = ID not found 1 = Invalid Data 2 = Format Error >2 = Invalid error 3-63 Reserved	S12F19
MAPFT	10	Map data format type	0 = Row format 1 = Array format 2 = Coordinate format >2 = Error 3-63 Reserved	S12F3, F5
MCINDEX	5()	Identifier used to link a handoff command message with its eventual completion message. Corresponding messages carry the same value for this data item.		S4F31, F33
MDACK	10	Map data acknowledge	0 = Map received 1 = Format error 2 = No ID match 3 = Abort/discard map >3 = Error 4-63 Reserved	S12F8, F10, F12
MDLN	20	Equipment Model Type, 6 bytes max	Same data as returned by S1,F2	S1F2, F13, F14; S7F22, F23, F26, F31, F39, F43
MEXP	20	Message expected in the form SxxFyy where x is stream and y is function		S9F13

Name	*Format*	*Description*	*Values*	*Where Used*
MF	10, 20	Material format code 1 byte by Format 10	Items with format 10 will be encoded as follows: 1 = Quantities in wafers 2 = Quantities in cassette 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 waterframes 13 = Quantities in carriers 14 = Quantities in substrates 15-63 Reserved Items with format 20 will be a unit identifier for one of the special SECS generic units, as specified in Section 12.	S3F2, F4, F5, F7; S16F11, F13, F15
MHEAD	10	SECS message block header associated with message block in error		S9F1, F3, F5, F7, F11
MID	10, 20	Material ID	80 Characters maximum	S2F27; S3F2, F4, F7, F9, F12, F13; S4F1, F3, F5, F7, F9, F11, F13, F15, F17; S7F7, F8, F10, F11, F13, F35, F36; S12F1, F3, F4, F5, F7, F9, F11, F13, F14, F15, F16, F17, F18; S16F11, F13, F15; S18F10, F11
MIDAC	10	Material ID Acknowledge Code, 1 byte	0 = Accepted 1 = Invalid port number 2 = Material is not present at identified port >2 = Error 3-63 Reserved	S3F14
MIDRA	10	Material ID Acknowledge Code, 1 byte	0 = Acknowledge, MID follows 1 = Acknowledge, will not send MID 2 = Acknowledge, will send MID later in S3F13 3-63 Reserved	S3F12
MLCL	5()	Message length	Defined by message size in (bytes)	S12F4, F5

Name	*Format*	*Description*	*Values*	*Where Used*
MMODE	10	Matrix mode select, 1 byte	1 = Host source mode Use S7F7 and F8 to define the process program to be used. 2 = Local source mode Use the matrix defined by S7F9, F10, F11 and F13 to define the process program to be used. The equipment will initialize to local source mode at power up. The equipment will default to local source mode from host source mode in the event of loss of communication with the host. 3 = Host immediate mode Use the current process program for all material unless changed by S7F1-F4. The timing of the mode change is equipment-specific.	S7F15
NACDS	32, 52	Not After Command Codes	Vector of all command codes which defined command may not succeed within the same block.	S7F22
NBCDS	32, 52	Not Before Command Codes	Vector of all command codes which defined command may not precede within the same block.	S7F22
NULBC	20, 51	Null bin code value	This value is the bin code value that is used for no die at a location. (For X/Y coordinate format the ASCII value is a value with "n" length. For other map formats, ASCII is a single byte per bin with "n" length per item; thus, the total number of bins is the length, i.e., length "n" = 10 for ASCII format is 10 single byte bin codes.) The format used must be the same as the one used for BINLT and BCEQU. Zero length indicates not used.	S12F1, F3, F4
OBJACK	51	Acknowledge code:	0 = Successful completion of requested data 1 = Error >1 Reserved	S14F2, F4, F6, F8, F10, F12, F14, F16, F18, F26, F28
OBJCMD	51	Specifies an action to be performed by an object:	0 = Reserved 1 = Attach to requestor 2 = Detach from requestor (requires authorization token) 3 = Reattach to requestor 4 = Set attributes (requires authorization token) >4 Reserved	
OBJID	20, 5()	Identifier for an object		S1F19; S14F1, F2, F3, F4

Name	*Format*	*Description*	*Values*	*Where Used*
OBJSPEC	20	A text string that has an internal format and that is used to point to a specific object instance. The string is formed out of a sequence of formatted substrings, each specifying an object's type and identifier. The substring format has the following four fields: object type, colon character ":", object identifier, greater-than symbol ">" where the colon character ":" is used to terminate an object type and the "greater than" symbol ">" is used to terminate an identifier field. The object type field may be omitted where it may be otherwise determined. The final ">" is optional.		S2F49; S13F11, F13, F15; S14F1, F3, F5, F7, F9, F10, F11, F13, F15, F16, F17, F19, F25, F27; S15F43, F47
OBJTOKEN	54	Token used for authorization		S14F14, F15; S15F37, F39, F41, F43
OBJTYPE	20, 5()	Identifier for a group or class of objects. All objects of the same type must have the same set of attributes available.		S1F19; S14F1, F3, F6, F7, F8, F25, F26, F27
OFLACK	10	Acknowledge code for OFF-LINE request	0 = OFF-LINE Acknowledge 1-63 Reserved	S1F16
ONLACK	10	Acknowledge code for ON-LINE request	0 = ON-LINE Accepted 1 = ON-LINE Not Allowed 2 = Equipment Already ON-LINE 3-63 Reserved	S1F18
OPID	5()	Operation ID. A unique integer generated by the requestor of an operation, used where multiple completion confirmations may occur.		S6F25; S14F19, F21; S15F21, F29, F30, F37, F41, F44, F46
ORLOC	10	Origin Location	implicit value of (0,0) 0 = Center die of wafer (Determined by: $\frac{\text{row or column count} + 1}{2}$ truncated) 1 = Upper right 2 = Upper left 3 = Lower left 4 = Lower right >4 = Error 5-63 Reserved Zero length indicates not available	S12F1, F3, F4

Name	*Format*	*Description*	*Values*	*Where Used*
OUTPTN	10, 51	A specialized version of PTN indicating the OutPutPort		S3F35
PARAMNAME	20	The name of a parameter in a request		S3F21, F23
PARAMVAL	1, 10, 11, 20, 3(), 4(), 5()	The value of the parameter named in PARAMNAME. Values that are lists are restricted to lists of single items of the same format type.		S3F21, F23
PDFLT	11, 20, 3(), 4(), 5()	Parameter Default Value	Specifies default value and data type of parameter. If no defaults are provided, item data length will be zero. For numeric or Boolean data, the default item may be a multi-varied vector if the parameter itself can be a vector. If RQPAR is false: Position of data in a default item is significant. When obtaining a default value to be used in the Nth position of a vector parameter, the value is obtained from the Nth position of the default item. If the default item has L entries, no default value will be provided for the L+1,...,PMAX parameter entries. If RQPAR is true: The length of the default vector (L) specifies the minimum number of entries which must be entered for the parameter. If > PMAX, only PMAX entries are required.	S7F22
PFCD	10	Predefined form code, 1 byte		S6F9
PGRPACTION	20	The action to be performed on a port group		S3F23

Name	*Format*	*Description*	*Values*	*Where Used*
PMAX	3(), 5()	Parameter Count Maximum	Maximum amount of data to be accepted by the host for this parameter. When a conflict arises between value of PMAX and length of PDFLT, PMAX takes precedence. For numeric and Boolean parameters: PMAX < 0 invalid PMAX = 0 specifies there is no upper bound PMAX = 1 specified a single value is expected PMAX > 1 specifies a vector of values is expected with a maximum of PMAX entries For string parameters: PMAX < 0 invalid PMAX = 0 specifies there is no upper bound PMAX > 0 maximum length of parameter string	S7F22
PNAME	20	Parameter Name ≤ 16 characters	Text string identifying the parameter value expected by its parent process command.	S7F22
PORTACTION	20	The action to be performed on a port		S3F25
PORTGRPNAME	20	The identifier of a group of ports		S3F21,F23
PPARM	11, 20, 3(), 4(), 5()	Process Parameter	Numeric or Boolean SECS data item, single or multiple value, or text string which provides information required to complete the process command to which the parameter refers.	S7F23, F26, F31, F39, F43
PPBODY	10, 20, 3(), 5()	Process program body	The process program describes to the equipment, in its own language, the actions to be taken in processing the material it receives.	S7F3, F6, F36, F37, F41
PPGNT	10	Process program grant status, 1 byte	0 = OK 1 = Already have 2 = No space 3 = Invalid PPID 4 = Busy, try later 5 = Will not accept >5 = Other error 6-63 Reserved	S7F2, F30
PPID	10, 20	Process program ID	Limited to a maximum of 80 bytes. The format used in the PPID will be host-dependent. For internal use of the equipment, the PPID can be treated as a unique binary pattern. If the local equipment is not prepared to display the transmitted code, the display should be in hexadecimal form.	S2F27; S7F1, F3, F5, F6, F8, F10, F11, F13, F17, F20, F23, F25, F26, F27, F31, F33, F34, F36, F53; S9F13

Name	*Format*	*Description*	*Values*	*Where Used*
PRAXI	10	Process axis	0 = Rows (X-axis), top, increasing 1 = Rows (X-axis), top, decreasing 2 = Rows (X-axis), bottom, increasing 3 = Rows (X-axis), bottom, decreasing 4 = Columns (Y-axis), left, increasing 5 = Columns (Y-axis), left, decreasing 6 = Columns (Y-axis), right, increasing 7 = Columns (Y-axis), right, decreasing >7 = Error 8-63 Reserved	S12F1, F3
PRCMDNAME	20	Commands sent to a process job:	"START" "STOP" "PAUSE" "RESUME" "ABORT" "CANCEL"	S16F5
PRDCT	5()	Process Die Count	Number of die to be processed or number of die which have been processed. (Zero length indicates not used.)	S12F1, F4
PREVENTID	5()	Processing related event identification:	1 = Waiting for material 2 = Job state change	S16F9
PRJOBID	20	Text string which uniquely identifies a process job		S16F4, F5, F6, F7, F9, F11, F12, F13, F14, F15, F16, F17, F20, F23, F25
PRJOBMILESTONE	5()	Notification of Processing status shall have one of the following values:	1 = Job Setup 2 = Job Processing 3 = Job Processing Complete 4 = Job Complete 5 = Job Waiting for Start	S16F7
PRJOBSPACE	52	The number of process jobs that can be created		S16F22
PRMTRLORDER	51	Defines the order by which material in the process jobs material list will be processed. Possible values are assigned as follows:	1 = ARRIVAL – process whichever material first arrives 2 = OPTIMIZE – process in an order that maximizes throughput 3 = LIST – follow the order in the list	S16F29
PRPAUSEEVENT	00	The list of event identifiers, which may be sent as an attribute value to a process job. When a process job encounters one of these events it will pause, until it receives the PRJobCommand RESUME.		S16F11, F13, F15
PRPROCESSSTART	11	Indicates that the process resource start processing immediately when ready:	TRUE = Automatic Start FALSE = Manual Start	S16F11, F13, F15, F25

Name	*Format*	*Description*	*Values*	*Where Used*
PRRECIPEMETHOD	51	Indicates the recipe specification type, whether tuning is applied and which method is used:	1 – Recipe only 2 – Recipe with variable tuning	S16F11, F13, F15
PRSTATE	51	Enumerated value, 1 byte		S16F20
PTN	10, 51	Material Port number, 1 byte		S3F17, F21, F25; S4F1, F3, F5, F7, F9, F11, F13, F15, F17; S16F13, F17, F21
QUA	10	Quantity in format, 1 byte		S3F2, F4, F5, F7
RAC	31, 51	Reset acknowledge, 1 byte	0 = Reset to be done 1 = Reset denied >1 = Other errors 2-63 Reserved	S2F20
RCMD	20, 31, 51	Remote command code or string		S2F21, F41, F49
RCPATTRDATA	0, 10, 11, 20, 3(), 4(), 5()	The contents (value) of a recipe attribute		S6F25; S15F13, F15, F18, F27, F28, F30, F32, F49, F51
RCPATTRID	20	The name (identifier) of a non-identifier recipe attribute		S6F25; S15F13, F15, F18, F27, F28, F30, F32, F49, F51
RCPBODY	10, 20, 3(), 5()	Recipe body		S15F13, F15, F18, F27, F32, F49, F51
RCPCLASS	20	Recipe class		S15F11
RCPCMD	51	Indicates an action to be performed on a recipe	5 = Delete 8 = Unprotect 9 = Protect 10 = Verify 11 = Link 12 = Unlink 13 = Certify 14 = De-certify 15 = Download 16 = Upload 0-4, 6-7, 17-63 Reserved	S15F21, F22
RCPDEL	51		0 = Delete 1 = Deselect >1 Reserved	S15F35
RCPDESCLTH	5()	The length in bytes of a recipe section		S15F24
RCPDESCNM	20	Identifies a type of descriptor of a recipe: “ASDesc”, “BodyDesc”, “GenDesc”		S15F24
RCPDESCTIME	20	The timestamp of a recipe section, in the format “YYYYMMDDhhmmsscc”		S15F24

Name	*Format*	*Description*	*Values*	*Where Used*
RCPID	20	Recipe identifier. Formatted text conforming to the requirements of OBJSPEC.		S15F21, F23, F28, F29, F30, F33, F35, F37, F41, F44, F53
RCPNAME	20	Recipe name		S15F11
RCPNEWID	20	The new recipe identifier assigned as the result of a copy or rename operation.		S15F19, F41, F44
RCPOWCODE	11	Indicates whether any pre-existing recipe is to be overwritten (= TRUE) or not (= FALSE) on download		S15F27, F49
RCPPARNM	20	The name of a recipe variable parameter. Maximum length of 256 characters.		S15F25, F33; S16F3, F11, F13, F15, F23
RCPPARRULE	20	The restrictions applied to a recipe variable parameter setting. Maximum length of 80 characters.		S15F25
RCPPARVAL	10, 11, 20, 3(), 4(), 5()	The initial setting assigned to a recipe variable parameter. Text form restricted to maximum of 80 characters.		S15F25, F33; S16F3, F11, F13, F15, F23
RCPRENAME	11	Indicates whether a recipe is to be renamed (= TRUE) or copied (= FALSE)		S15F19
RCPSECCODE	10	Indicates the sections of a recipe requested for transfer or being transferred:	1 = Generic attributes only 3 = Generic attributes and body 4 = All agent-specific datasets (zero or more) 7 = Generic attributes, body, and all agent-specific datasets 8 = Single agent-specific dataset (zero or one) 11 = Generic attributes, body, and single agent-specific datasets All other values reserved	S15F15, F16, F17
RCPSECNM	20	Recipe section name: “Generic”, “Body”, or “ASDS”		S15F15, F18
RCPSPEC	20	Recipe specifier. The object specifier of a recipe.		S15F1, F9, F13, F15, F17, F19, F27, F28, F31, F32, F45, F49, F51, F53; S16F11, F13, F15, F23
RCPSTAT	51	The status of a managed recipe	0 = Does not exist 8 = Unprotected 9 = Protected	S15F10
RCPUPDT	11	Indicates if an existing recipe is to be updated (= True) or a new recipe is to be created (= False)		S15F13
RCPVERS	20	Recipe version		S15F10, F12
READLN	3(), 5()	Maximum length to read		S13F5

Name	*Format*	*Description*	*Values*	*Where Used*
RECLEN	3(), 5()	Maximum length of a Discrete record		S13F4
REFP	3()	Reference Point		S12F1, F4
REPGSZ	20, 3(), 5()	Reporting group size		S2F23; S17F5
RESC	31, 51	Resolution code for numeric data	1 = Absolute. Value may be specified to nearest increment of RESV 2 = Significant Digits. Value may be specified with no more significant digits than RESV allows	S7F22
RESPEC	20	Object specifier for the recipe executor		S15F29, F33, F35
RESV	3(), 4(), 5()	Resolution value for numeric data	If RESC=1, then RESV contains smallest increment allowed for parameter	S7F22
RETICLEID	20	The object identifier for a reticle. Conforms to OBJSPEC.		S3F35
RETPLACEINSTR	51	Instructions to indicate which pod slots will have reticles placed. Possible values for ReticlePlacementInstruction are:	0 = PLACE 1 = PASS BY 2 = CURRENTLY OCCUPIED	S3F35
RETREMOVEINSTR	51	Instructions to indicate which pod slots will have reticles removed	0 = REMOVE 1 = PASS BY	S3F35
RIC	31, 51	Reset code, 1 byte	0 = Not used 1 = Power up reset >1 Other reset conditions 2-63 Reserved	S2F19
RMACK	51	Conveys whether a requested action was successfully completed, denied, completed with errors, or will be completed with notification to the requestor	0 = Successful completion 1 = Cannot perform action 2 = Completed with errors 3 = Action will be completed and notification sent 4 = No request for this action exists	S15F4, F6, F8, F10, F12, F14, F16, F18, F20, F22, F24, F25, F26, F28, F30, F32, F34, F36, F38, F40, F42, F44, F48, F53
RMCHGSTAT	5()	Indicates the change that occurred for an object	0 = No change 1 = Created 2 = Updated 3 = Stored (new) 4 = Replaced 5 = Deleted 6 = Copied (new object) 7 = Renamed 8 = Unprotected 9 = Protected 10 = Verified 11 = Linked 12 = Unlinked 13 = Certified 14 = De-certified 15 = Selected 16 = Deselected	S15F25

Name	*Format*	*Description*	*Values*	*Where Used*
RMCHGTYPE	5()	Indicates the type of change for a recipe	0 = No change 1 = Create 2 = Update 5 = Delete 6 = Copy (new object) 7 = Rename 8 = Unprotect 9 = Product 10 = Verify 11 = Link 12 = Unlink 13 = Certify 14 = De-certify 15 = Change generic attribute 16 = Change agent-specific attribute 17 = Change both generic and agent-specific attributes	S15F37, F41, F44, F46, F47, F48
RMDATASIZE	5()	The maximum total length, in bytes, of a multi-block message, used by the receiver to determine if the anticipated message exceeds the receiver's capacity		S15F1
RMGRNT	10	Grant code, used to grant or deny a request. 1 byte.	0 = Permission granted 1 = Cannot accept now, try again 2 = No space 3 = Request is on hold 4-64 Reserved	S15F2, F37, F46
RMNEWNS	20	New name (identifier) assigned to a recipe namespace		S15F5
RMNSCMD	51	Action to be performed on a recipe namespace	1 = Created 5 = Deleted 0, 2-4, 6-63 Reserved	S15F3
RMNSSPEC	20	The object specifier of a recipe namespace		S15F3, F5, F7, F11, F21, F23, F25, F47
RMRECSPEC	20	The object specifier of a distributed recipe namespace recorder		S15F39, F41, F43
RMREQUESTOR	11	Set to TRUE if initiator of change request was an attached segment. Set to FALSE otherwise.		S15F41, F44, F46
RMSEGSPEC	20	The object specifier of a distributed recipe namespace segment		S15F37, F39, F41, F44, F46, F47
RMSPACE	5()	The amount of storage available for at least one recipe in a recipe namespace, in bytes		S15F8
ROWCT	5()	Row count in die increments		S12F1, F4

Name	*Format*	*Description*	*Values*	*Where Used*
RPMACK	51	Reticle Pod management service acknowledge code. One byte.	0 = Acknowledge, service has been performed 1 = Service does not exist 2 = Can not perform now 3 = At least parameter does not exist 4 = Acknowledge, request will be performed with completion signaled later by an event 5 = Service is not completed or prohibited 6 = No such object exists 7-63 Reserved	
RPMDESTLOC	20	The LocationID towards which a reticle must be moved. Conforms to OBJID		S14F19
RPMSOURLOC		The LocationID of the location from which to pick-up a reticle for moving it to another location. Conforms to OBJID.		S14F19
RPSEL	51	Reference Point Select	Number of reference point from 0-n.	S12F1, F4
RPTID	20, 3(), 5()	Report ID		S2F33, F35; S6F11, F13, F16, F11, F19, F21, F22; S17F1, F2, F3, F4, F5, F9, F11, F12
RPTOC	11	A Trace Object attribute for a flag which, if set TRUE, causes only variables which have changed during the sample period to be included in a report.		S14F1, F2, F3, F4; S17F5
RQCMD	11	Required Command	True = Command must be specified False = Command is optional	S7F22
RQPAR	11	Required Parameter	True = Parameter must be specified False = Parameter is optional	S7F22
RRACK	10	Request to Receive Acknowledge code, 1 byte	0 = Acknowledge, OK (note that 'OK' differs from 'ready') 1 = Invalid port number 2 = Requested material is not at identified port 3 = Busy. Try again 4 = Sender does not have permission to perform this operation 5-63 Reserved	S4F18

Name	*Format*	*Description*	*Values*	*Where Used*
RSACK	10	Ready to Send Acknowledge code, 1 byte	0 = Acknowledge, OK (note that 'OK' differs from 'ready') 1 = Invalid port number 2 = Port is already occupied 3 = Busy, unable to move material at this time. Try again 4 = Receiver does not have permission to perform this operation 5-63 Reserved	S4F2
RSDA	10	Request Spool Data Acknowledge	0 = OK 1 = Denied, busy try later 2 = Denied, spooled data does not exist 3-63 Reserved	S6F24
RSDC	51	Request Spool Data Code	0 = Transmit Spooled Messages 1 = Purge Spooled Messages 2-63 Reserved	S6F23
RSINF	3()	Starting location for row or column. This item consists of 3 values (x,y,direction). If direction value is negative, it equals decreasing direction. If the value is positive, it equals increasing direction. Direction must be a non-zero value.		S12F7, F14
RSPACK	10	Reset Spooling Acknowledge	0 = Acknowledge, spooling setup accepted 1 = Spooling setup rejected 2-63 Reserved	S2F44
RTYPE	3(), 5()	Type of record	0 = Stream 1 = Discrete 2-63 Reserved	S13F4
SDACK	10	Map set-up data acknowledge	0 = Received data >1 = Error 1-63 Reserved	S12F2
SDBIN	10	Send bin information flag	0 = Sent bin information 1 = Don't send bin information >1 = Error 2-63 Reserved	S12F17
SEQNUM	3(), 5()	Command Number	Value which identifies a unique process program command by its position in the list of commands relative to the first. For the first command of the process program, SEQNUM is l.	S7F27
SFCD	10	Status form code, 1 byte		S1F5
SHEAD	10	Stored header related to the transaction timer		S9F9
SLOTID	51	Used to reference material by slot (a position that holds material/substrates) in a carrier. This item may be implemented as an array in some messages.		S16F11, F13, F15
SMPLN	3(), 5()	Sample number		S6F1

Name	*Format*	*Description*	*Values*	*Where Used*
SOFTREV	20	Software revision code 6 bytes maximum		S1F2, F13, F14; S7F22, F23, F26, F31, F39, F43
SPAACK	10	Equipment acknowledgement code, 1 byte	0 = Everything correct 1 = Invalid data >1 = Equipment-specific error 2-63 Reserved	S2F4
SPD	10	Service program data		S2F3, F6
SPID	20	Service program ID, 6 characters		S2F1, F5, F7, F9, F12; S9F13
SPNAME	20	Service parameter name defined in specific standard. If service parameter is defined as an object attribute, this is completely the same as ATTRID except format restrictions above.		S14F19, F20, F21, F28
SPR	Device Depend-ent	Service program results	Device dependent	S2F10
SPVAL	0, 10, 11, 20, 21, 3(), 4(), 5()	Service parameter value, corresponding to SPNAME. If service parameter is defined as an object attribute, this is completely the same as ATTRDATA except format restrictions for the attribute.		S14F19, F20, F21
SSACK	20	Indicates the success or failure of a requested action. Two characters.		S18F2, F4, F6, F8, F10, F12, F14
SSCMD	20	Indicates an action to be performed by the subsystem.		S18F13
STATUS	20	Provides status information for a subsystem component. Used in the data item STATUSLIST.		See STATUSLIST
STATUSLIST	0	A list of STATUS data sent in a fixed order. STATUSLIST has the following form:	L,s 1. <$STATUS_1$> . . s. <$STATUS_s$>	S18F4, F8, F10, F12, F14, F16
STEMP	20	String template. ASCII text string acceptable to equipment as a parameter value. A data string matches a template string if the data string is at least as long as the template and each character of the data string matches the corresponding character of the template. A null list indicates all user data is acceptable to the machine.		S7F22

Name	*Format*	*Description*	*Values*	*Where Used*
STIME	20	Sample time, 12 or 16 bytes. NOTE 3: The 16-byte format is currently optional. After January 1, 1998, the 16-byte format shall be required on new and updated implementations. Support for the 12-byte format shall be supported as a configurable option using the equipment constant TimeFormat. This is a format requirement only and does not imply either precision or accuracy.	If 12 bytes the format is YYMMDDhhmmss YY = Year 00 to 99 MM = Month 01 to 12 DD = Day 01 to 31 hh = Hour 00 to 23 mm = Minute 00 to 59 ss = Second 00 to 59 If 16 bytes the format is YYYYMMDDhhmmsscc YYYY = Year 0000 to 9999 MM = Month 01 to 12 DD = Day 01 to 31 hh = Hour 00 to 23 mm = Minute 00 to 59 ss = Second 00 to 59 cc = Centisecond 00 to 99	S6F1
STRACK	10	Spool Stream Acknowledge	1 = Spooling not allowed for stream (i.e., Stream 1) 2 = Stream unknown 3 = Unknown function specified for this stream 4 = Secondary function specified for this stream	S2F44
STRID	51	Stream Identification		S2F43, F44
STRP	3()	Starting position in die coordinate position. Must be in (X,Y) order.		S12F9, F16
SV	0, 10, 11, 20, 21, 3(), 4(), 5()	Status variable value		S1F4; S6F1
SVCACK	10	Service acceptance acknowledge code, 1 byte	0 = Acknowledge, service has been performed 1 = Service does not exist 2 = Cannot perform now 3 = At least parameter is invalid 4 = Acknowledge, service will be performed with completion notified later with parameters for response 5 = Service is not completed or prohibited 6 = No such object exists 7-63 Reserved	S14F20
SVCNAME	20	Service name provided on specified object asking by the host		S14F19, F26, F27, F28
SVID	20, 3(), 5()	Status variable ID	Status variables may include any parameter that can be sampled in time such as temperature or quantity of a consumable.	S1F3, F11, F12; S2F23
SVNAME	20	Status Variable Name		S1F12
TARGETID	20	Identifies where a request for action or data is to be applied. If text, conforms to OBJSPEC.		S18F1, F3, F5, F7, F9, F11, F13

Name	*Format*	*Description*	*Values*	*Where Used*
TARGETSPEC	20	Object specifier of target object		S14F17; S15F43
TBLACK	51	Indicates success or failure	0 = Success 1 = Failure	S13F14, F16
TBLCMD	51	Provides information about the table or parts of the table being transferred or requested. Enumerated:	0 = Complete Table 1 = New rows (add) 2 = New columns (append) 3 = Replace existing rows 4 = Replace existing columns	S13F13, F15
TBLELT	0, 10, 11, 20, 21, 3(), 4(), 5()	Table element. The first table element in a row is used to identify the row.		S13F13, F15, F16
TBLID	20	Table identifier. Text conforming to the requirements of OBJSPEC.		S13F13, F15, F16
TBLTYP	20	A reserved text string to denote the format and application of the table. Text conforming to the requirements of OBJSPEC.		S13F13, F15, F16
TEXT	10, 20, 22, 3(), 5()	A single line of characters		S10F1, F3, F5, F9
TIAACK	10	Equipment acknowledgement code, 1 byte	0 = Everything correct 1 = Too many SVIDs 2 = No more traces allowed 3 = Invalid period >3 = Equipment-specified error 4-63 Reserved	S2F24
TIACK	10	Time Acknowledge Code, 1 byte	0 = OK 1 = Error, not done 2-63 Reserved	S2F32
TID	10	Terminal number, 1 byte	0 = Single or main terminal >0 = Additional terminals at the same equipment	S10F1, F3, F5, F7
TIME	20	Time of day, 12 or 16 bytes NOTE 4: The 16-byte format is currently optional. After January 1, 1998, the 16-byte format shall be required on new and updated implementations. Support for the 12-byte format shall be supported as a configurable option using the equipment constant TimeFormat. This is a format requirement only and does not imply either precision or accuracy.	If 12 bytes the format is YYMMDDhhmmss YY = Year 00 to 99 MM = Month 01 to 12 DD = Day 01 to 31 hh = Hour 00 to 23 mm = Minute 00 to 59 ss = Second 00 to 59 If 16 bytes the format is YYYYMMDDhhmmsscc YYYY = Year 0000 to 9999 MM = Month 01 to 12 DD = Day 01 to 31 hh = Hour 00 to 23 mm = Minute 00 to 59 ss = Second 00 to 59 cc = Centisecond 00 to 99	S2F18, F31

Name	*Format*	*Description*	*Values*	*Where Used*
TIMESTAMP	20	A text string indicating the time of an event, which encodes time in the following format:	YYYYMMDDhhmmsscc YYYY = year 0000 to 9999 MM = month 01 to 12 DD = day 01 to 31 hh = hour 00 to 23 mm = minute 00 to 59 ss = second 00 to 59 cc = centisecond 00 to 99	S5F9, F11, F15; S15F41, F44; S16F5, F7, F9
TOTSMP	20, 3(), 5()	Total samples to be made		S2F23; S17F5
TRACK	11	Tells whether the related transfer activity was successful (= True) or unsuccessful (= False)		S4F20, F22, F23
TRATOMICID	5()	Equipment assigned identifier for an atomic transfer		S4F20
TRAUTOD	11	A Trace Object attribute for a control flag which, if set TRUE, causes the Trace Object to delete itself when it has completed a report		S14F1, F2, F3, F4; S17F5
TRAUTOSTART	11	For each atomic transfer, this data item tells the equipment if it should automatically start the handoff when ready (= TRUE) or await the host's "StartHandoff" command (= FALSE) following setup. This data item only affects the primary transfer partner.		S4F19
TRCMDNAME	20	Identifier of the transfer job-related command to be executed. Possible values:	"CANCEL" "PAUSE" "RESUME" "ABORT" "STOP" "STARTHANDOFF" (requires a TRATOMICID as a parameter)	S4F21
TRDIR	51	Direction of handoff.	1 = Send material 2 = Receive material 0, 3-63 Reserved	S4F19, F29
TRID	20, 3(), 5()	Trace request ID		S2F23; S6F1, F27, F28; S17F5, F6, F7, F8, F13, F14
TRJOBID	10	Equipment assigned identifier for the transfer job.		S4F20, F21
TRJOBMS	51	Milestone for a transfer job (e.g., started or complete)	1 = Transfer Job Started 2 = Transfer Job Complete 0, 3-63 Reserved	S4F23
TRJOBNAME	20	Host assigned identifier for the transfer job. Limited to a maximum of 80 characters.		S4F19, F23
TRLINK	5()	Common identifier for the atomic transfer used by the transfer partners to confirm that they are working on the same host-defined task.		S4F19, F29, F31, F33, F39

Name	*Format*	*Description*	*Values*	*Where Used*
TRLOCATION	5()	Identifier of the material location involved with the transfer. For one transfer partner, this will represent the designated source location for the material to be sent. For the other transfer partner, it will represent the designated destination location for the material to be received.		S4F19, F29
TROBJNAME	20	Identifier for the material (transfer object) to be transferred		S4F19, F23, F29
TROBJTYPE	5()	Type of object to be transferred		S4F19, F29
TRPORT	5()	Identifier of the equipment port to be used for the handoff		S4F19, F29
TRPTNR	20	Name of the equipment which will serve as the other transfer partner for this atomic transfer. This corresponds to EQNAME.		S4F19, F29
TRPTPORT	5()	Identifier of the transfer partner's port to be used for the transfer		S4F19, F29
TRRCP	20	Name of the transfer recipe for this handoff. Limited to a maximum of 80 characters.		S4F19
TRROLE	51	Tells whether the equipment is to be the primary or secondary transfer partner	1 = Primary Transfer Partner 2 = Secondary Transfer Partner	S4F19, F29
TRSPER	4()	A Trace Object attribute which holds the value for sampling interval time		S14F1, F2, F3, F4; S17F5
TRTYPE	51	Tells whether the equipment is to be an active or passive participant in the transfer	1 = Active 2 = Passive	S4F19, F23, F29
TSIP	10	Transfer status of input port, 1 byte	1 = Idle state 2 = Prep state 3 = Track on state 4 = Stuck in Receiver state 5-63 Reserved	S1F10
TSOP	10	Transfer status of output port, 1 byte	1 = Idle state 2 = Prep state 3 = Track on state 4 = Stuck in Sender state 5 = Completion state 6-63 Reserved	S1F10
TTC	3(), 5()	Time to completion		S3F4
ULIM	3(), 4(), 5()	Upper limit for numeric value		S7F22

Name	*Format*	*Description*	*Values*	*Where Used*
UNFLEN	3(), 5()	Unformatted Process Program Length	If greater than zero, indicates PPID is available as an unformatted process program and its length in bytes. If zero, the PPID is not available as an unformatted process program. Negative values are invalid.	S7F34
UNITS	20	Units Identifier	As allowed by SEMI E5 Section 12.	S1F12; S2F30, F48; S7F22
UPPERDB	11, 20, 3(), 4(), 5()	A variable limit attribute which defines the upper boundary of the deadband of a limit. The value applies to a single limit (LIMITID) for a specified VID. Thus, UPPERDB and LOWERDB as a pair define a limit.		S2F45, F48
V	0, 10, 11, 20, 21, 3(), 4(), 5()	Variable data		S6F11, F13, F16, F20, F22
VID	20, 3(), 5()	Variable ID		S2F33, F45, F46, F47, F48; S6F13, F18, F22; S17F1
VLAACK	10	Variable Limit Attribute Acknowledge Code, 1 byte	0 = Acknowledge, command will be performed 1 = Limit attribute definition error 2 = Cannot perform now >2 = Other equipment-specific error 3-63 Reserved	S2F46
XDIES	4(), 5()	X-axis die size (index)		S12F1, F4
XYPOS	3()	X and Y Coordinate Position. Must be in (X,Y) order.		S12F11, F18
YDIES	4(), 5()	Y-axis die size (index)		S12F1, F4

9.7 *Variable Item Dictionary* — This section defines variable data items which are available to the Host for data collection purposes.

Name: A unique mnemonic name for this variable data item. This name is provided for reference only.

Class: The data type classification (SV, ECV, or DVVAL) of the item. Status values (SVs) always contain valid information, while data values (DVVALs) may only be valid upon the occurrence of a particular event. All equipment constants (ECVs) are settable by the Host.

Format: The allowable item format codes which can be used for this variable data item. ...as in Data Item Dictionary...

Description: A description of the variable data item, with the meanings of specific values. Also, specify validity for item of class DVVAL.

Table 4 Variable Item Dictionary

Name	*Format*	*Description*	*Values*	*Class*
AlarmID	3(), 5()	This variable is valid only upon the setting or clearing of an alarm condition and contains the current alarm identification (ALID), regardless of whether that alarm is enabled for reporting		DVVAL
AlarmsEnabled	0	Contains the list of alarms (ALIDs) enabled for reporting (via Stream 5)	Structure: L,n n = # of alarms enabled 1.<ALID$_1$> . . n.<ALID$_n$>	SV
AlarmsSet	0	Contents of this variable is a list of alarms (ALIDs) currently in the UNSAFE (alarm set) state, regardless of whether the alarms are enabled for reporting	Structure: L,n n = # of alarms set 1.<ALID$_1$> . . n.<ALID$_n$>	SV
ARAMSAccumReset	20	The timestamp of when the set of accumulators EngTime, InterruptionCtr, PrdTime, NSTime, SbyTime, SDTime, and UDTime were reset to zero. Uses format defined for CLOCK.		SV
ARAMSInfo	20	Text field set by the equipment to provide additional information concerning an ARAMS state change.		SV
ARAMSState	20	The ARAMS code corresponding to the current state/substate. Four characters.		SV
ARAMSText	20	Text describing the ARAMSState. 3–80 characters.		SV
ARAMSTimeStamp	20	The timestamp of the last ARAM state change. This is a format requirement only and does not imply precision or accuracy. Uses format defined for CLOCK.		SV
CLOCK	20	The value of the equipment's internal clock. This is a format requirement only and does not imply precision or accuracy.	Format: YYYYMMDDhhmmsscc YYYY = Year 0000 to 9999 MM = Month 01 to 12 DD = Day 01 to 31 hh = Hour 00 to 23 mm = Minute 00 to 59 ss = Second 00 to 59 cc = centisecond 00 to 99	SV
ControlState	10, 51	This status variable contains the code which identifies the current control state of the equipment. When reported related to a control state transition, its value should represent the state current after the transition.	1 = OFF-LINE/EQUIPMENT OFF-LINE 2 = OFF-LINE/ATTEMPT ON-LINE 3 = OFF-LINE/HOST OFF-LINE 4 = ON-LINE/LOCAL 5 = ON-LINE/REMOTE 6-63 Reserved	SV
CycleCtr	5()	The number of machine cycles during the lifetime of the equipment. Non-resettable.		SV