SEMI E5-0600
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 March 2, 2000. Initially available on SEMI OnLine May 2000; to be published June 2000. Originally published in 1982; previously published February 2000.

CONTENTS

S2,F6 Service Program Load Data (SPD)
S2,F7 Service Program Run Send (CSS)
S2,F8 Service Program Run Acknowledge (CSA)
S2,F9 Service Program Results Request (SRR)
S2,F10 Service Program Results Data (SRD)
S2,F11 Service Program Directory Request (SDR)
S2,F12 Service Program Directory Data (SDD)
S2,F13 Equipment Constant Request (ECR)
S2,F14 Equipment Constant Data (ECD)
S2,F15 New Equipment Constant Send (ECS)
S2,F16 New Equipment Constant Acknowledge (ECA)
S2,F17 Date and Time Request (DTR)
S2,F18 Date and Time Data (DTD)
S2,F19 Reset/Initialize Send (RIS)
S2,F20 Reset Acknowledge (RIA)
S2,F21 Remote Command Send (RCS)
S2,F22 Remote Command Acknowledge (RCA)
S2,F23 Trace Initialize Send (TIS)
S2,F24 Trace Initialize Acknowledge (TIA)
S2,F25 Loopback Diagnostic Request (LDR)
S2,F26 Loopback Diagnostic Data (LDD)
S2,F27 Initiate Processing Request (IPR)
S2,F28 Initiate Processing Acknowledge (IPA)
S2,F29 Equipment Constant Namelist Request (ECNR)
S2,F30 Equipment Constant Namelist (ECN)
S2,F31 Date and Time Set Request (DTS)
S2,F32 Date and Time Set Acknowledge (DTA)
S2,F33 Define Report (DR)
S2,F34 Define Report Acknowledge (DRA)
S2,F35 Link Event Report (LER)
S2,F36 Link Event Report Acknowledge (LERA)
S2,F37 Enable/Disable Event Report (EDER)
S2,F38 Enable/Disable Event Report Acknowledge (EERA)
S2,F39 Multi-block Inquire (DMBI)
S2,F40 Multi-block Grant (DMBG)
S2,F41 Host Command Send (HCS)
S2,F42 Host Command Acknowledge (HCA)
S2,F43 Reset Spooling Streams and Functions (RSSF)
S2,F44 Reset Spooling Acknowledge (RSA)
S2,F45 Define Variable Limit Attributes (DVLA)
S2,F46 Variable Limit Attribute Acknowledge (VLAA)
S2,F47 Variable Limit Attribute Request (VLAR)
S2,F48 Variable Limit Attribute Send (VLAS)
S2,F49 Enhanced Remote Command
S2,F50 Enhanced Remote Command Acknowledge

7.7 Stream 3 Material Status

S3,F0 Abort Transaction (S3F0)
S3,F1 Material Status Request (MSR)
S3,F2 Material Status Data (MSD)
S3,F3 Time to Completion Request (TCR)
S3,F4 Time to Completion Data (TCD)
S3,F5 Material Found Send (MFS)
S3,F6 Material Found Acknowledge (MFA)
S3,F7 Material Lost Send (MLS)
S3,F8 Material Lost Acknowledge (MLA)
S3,F9 Material ID Equate Send (IES)
S3,F10 Material ID Equate Acknowledge (IEA)
S3,F11 Material ID Request (MIDR)
S3,F12 Material ID Request Acknowledge (MIRA)
S3,F13 Material ID Send (MIS)
S3,F14 Material ID Acknowledge (MIA)
S3,F15 Materials Multi-Block Inquire (MMBI)
S3,F16 Materials Multi-Block Grant (MMBG)
S3,F17 Carrier Action Request
S3,F18 Carrier Action Acknowledge
S3,F19 Cancel All Carrier Out Request
S3,F20 Cancel All Carrier Out Acknowledge
S3,F21 Port Group Definition
S3,F22 Port Group Definition Acknowledge
S3,F23 Port Group Action Request
S3,F24 Port Group Action Acknowledge
S3,F25 Port Action Request
S3,F26 Port Action Acknowledge

7.8 Stream 4 Material Control

S4,F0 Abort Transaction (S4F0)
S4,F1 Ready to Send Material (RSN)
S4,F2 Ready to Send Acknowledge (RSA)
S4,F3 Send Material (SMN)
S4,F4 Not Used
S4,F5 Handshake Complete (HCN)
S4,F6 Not Used
S4,F7 Not Ready to Send (ABN)
S4,F8 Not Used
S4,F9 Stuck in Sender (SSN)
S4,F10 Not Used
S4,F11 Stuck in Receiver (SRN)
S4,F12 Not Used
S4,F13 Send Incomplete Timeout (SIN)
S4,F14 Not Used
S4,F15 Material Received (MRN)
S4,F16 Not Used
S4,F17 Request to Receive (RTR)
S4,F18 Request to Receive Acknowledge (RRA)

Macro Level Messages

S4,F19 Transfer Job Create (TJ)
S4,F20 Transfer Job Acknowledge (TJA)
S4,F21 Transfer Job Command (TJC)
S4,F22 Transfer Command Acknowledge (TCA)
S4,F23 Transfer Job Alert (TJA)

S4,F24 Transfer Alert Acknowledge (TLA)
S4,F25 Multi-Block Inquire (MBI4)
S4,F26 Multi-Block Grant (MBG4)

Micro Level Messages

S4,F27 Handoff Ready (HR)
S4,F28 Not Used
S4,F29 Handoff Command (HC)
S4,F30 Not Used
S4,F31 Handoff Command Complete (HCC)
S4,F32 Not Used
S4,F33 Handoff Verify (HV)
S4,F34 Not Used
S4,F35 Handoff Cancel Ready (HCR)
S4,F36 Not Used
S4,F37 Handoff Cancel Ready Acknowledge (HCA)
S4,F38 Not Used
S4,F39 Handoff Halt (HH)
S4,F40 Not Used
S4,F41 Handoff Halt Acknowledge (HHA)
S4,F42 Not Used

7.9 Stream 5 Exception Handling

S5,F0 Abort Transaction (S5F0)
S5,F1 Alarm Report Send (ARS)
S5,F2 Alarm Report Acknowledge (ARA)
S5,F3 Enable/Disable Alarm Send (EAS)
S5,F4 Enable/Disable Alarm Acknowledge (EAA)
S5,F5 List Alarms Request (LAR)
S5,F6 List Alarm Data (LAD)
S5,F7 List Enabled Alarm Request (LEAR)
S5,F8 List Enabled Alarm Data (LEAD)
S5,F9 Exception Post Notify (EXPN)
S5,F10 Exception Post Confirm (EXPC)
S5,F11 Exception Clear Notify (EXCN)
S5,F12 Exception Clear Confirm (EXCC)
S5,F13 Exception Recover Request (EXRR)
S5,F14 Exception Recover Acknowledge (EXRA)
S5,F15 Exception Recovery Complete Notify (EXRCN)
S5,F16 Exception Recovery Complete Confirm (EXRCC)
S5,F17 Exception Recovery Abort Request (EXRAR)
S5,F18 Exception Recovery Abort Acknowledge (EXRAA)

7.10 Stream 6 Data Collection

S6,F0 Abort Transaction (S6F0)
S6,F1 Trace Data Send (TDS)
S6,F2 Trace Data Acknowledge (TDA)
S6,F3 Discrete Variable Data Send (DVS)
S6,F4 Discrete Variable Data Acknowledge (DVA)
S6,F5 Multi-block Data Send Inquire (MBI)
S6,F6 Multi-block Grant (MBG)
S6,F7 Data Transfer Request (DDR)
S6,F8 Data Transfer Data (DDD)
S6,F9 Formatted Variable Send (FVS)
S6,F10 Formatted Variable Acknowledge (FVA)
S6,F11 Event Report Send (ERS)
S6,F12 Event Report Acknowledge (ERA)
S6,F13 Annotated Event Report Send (AERS)
S6,F14 Annotated Event Report Acknowledge (AERA)
S6,F15 Event Report Request (ERR)
S6,F16 Event Report Data (ERD)
S6,F17 Annotated Event Report Request (AERR)
S6,F18 Annotated Event Report Data (AERD)
S6,F19 Individual Report Request (IRR)
S6,F20 Individual Report Data (IRD)
S6,F21 Annotated Individual Report (AIR)
S6,F22 Annotated Individual Report Data (AIRD)
S6,F23 Request Spooled Data (RSD)
S6,F24 Request Spooled Data Acknowledgement Send (RSDAS)
S6,F25 Notification Report Send
S6,F26 Notification Report Send Acknowledge
S6,F27 Trace Report Send (TRS)
S6,F28 Trace Report Send Acknowledge (TRSA)
S6,F29 Trace Report Request (TRR)
S6,F30 Trace Report Data (TRD)

7.11 Stream 7 Process Program Management

S7,F0 Abort Transaction (S7F0)
S7,F1 Process Program Load Inquire (PPI)
S7,F2 Process Program Load Grant (PPG)
S7,F3 Process Program Send (PPS)
S7,F4 Process Program Acknowledge (PPA)
S7,F5 Process Program Request (PPR)
S7,F6 Process Program Data (PPD)
S7,F7 Process Program ID Request (PIR)
S7,F8 Process Program ID Data (PID)
S7,F9 M/P M Request (MMR)
S7,F10 M/P M Data (MMD)
S7,F11 M/P M Update Send (UMS)
S7,F12 M/P M Update Acknowledge (UMA)
S7,F13 Delete M/P M Entry Send (DES)
S7,F14 Delete M/P M Entry Acknowledge (DEA)
S7,F15 Matrix Mode Select Send (MMS)
S7,F16 Matrix Mode Select Acknowledge (MMA)
S7,F17 Delete Process Program Send (DPS)
S7,F18 Delete Process Program Acknowledge (DPA)
S7,F19 Current EPPD Request (RER)

S7,F20 Current EPPD Data (RED)
S7,F21 Equipment Process Capabilities Request (PCR)
S7,F22 Equipment Process Capabilities Data (PCD)
S7,F23 Formatted Process Program Send (FPS)
S7,F24 Formatted Process Program Acknowledge (FPA)
S7,F25 Formatted Process Program Request (FPR)
S7,F26 Formatted Process Program Data (FPD)
S7,F27 Process Program Verification Send (PVS)
S7,F28 Process Program Verification Acknowledge (PVA)
S7,F29 Process program Verification Inquire (PVI)
S7,F30 Process Program Verification Grant (PVG)
S7,F31 Verification Request Send (VRS)
S7,F32 Verification Request Acknowledge (VRA)
S7,F33 Process Program Available Request (PAR)
S7,F34 Process Program Availability Data (PAD)
S7,F35 Process Program for MID Request (PPMR)
S7,F36 Process Program for MID Data (PPMD)

7.12 Stream 8 Control Program Transfer
S8,F0 Abort Transaction (S8F0)
S8,F1 Boot Program Request (BPR)
S8,F2 Boot Program Data (BPD)
S8,F3 Executive Program Request (EPR)
S8,F4 Executive Program Data (EPD)

7.13 Stream 9 System Errors
S9,F0 Abort Transaction (S9F0)
S9,F1 Unrecognized Device ID (UDN)
S9,F2 Not Used
S9,F3 Unrecognized Stream Type (USN)
S9,F4 Not Used
S9,F5 Unrecognized Function Type (UFN)
S9,F6 Not Used
S9,F7 Illegal Data (IDN)
S9,F8 Not Used
S9,F9 Transaction Timer Timeout (TTN)
S9,F10 Not Used
S9,F11 Data Too Long (DLN)
S9,F12 Not Used
S9,F13 Conversation Timeout (CTN)
S9,F14 Not Used

7.14 Stream 10 Terminal Services
S10,F0 Abort Transaction (S10F0)
S10,F1 Terminal Request (TRN)
S10,F2 Terminal Request Acknowledge (TRA)
S10,F3 Terminal Display, Single (VTN)
S10,F4 Terminal Display, Single Acknowledge (VTA)
S10,F5 Terminal Display, Multi-block (VTN)
S10,F6 Terminal Display, Multi-block Acknowledge (VMA)
S10,F7 Multi-block Not Allowed (MNN)
S10,F8 Not Used
S10,F9 Broadcast (BCN)
S10,F10 Broadcast Acknowledge (BCA)

7.15 Stream 11 Host File Services (Deleted)

7.16 Stream 12 Wafer Mapping
S12,F0 Abort Transaction (S12F0)
S12,F1 Map Set-Up Data Send (MSDS)
S12,F2 Map Set-up Data Acknowledge (MSDA)
S12,F3 Map Set-up Data Request (MSDR)
S12,F4 Map Set-up Data (MSD)
S12,F5 Map Transmit Inquire (MAPTI)
S12,F6 Map Transmit Grant (MAPTG)
S12,F7 Map Data Send Type 1 (MDS1)
S12,F8 Map Data Acknowledge Type 1 (MDA1)
S12,F9 Map Data Send Type 2 (MDS2)
S12,F10 Map Data Acknowledge Type 2 (MDA2)
S12,F11 Map Data Send Type 3 (MDS3)
S12,F12 Map Data Acknowledge 3 (MDA3)
S12,F13 Map Data Request Type 1 (MDR1)
S12,F14 Map Data Type 1 (MD1)
S12,F15 Map Data Request Type 2 (MDR2)
S12,F16 Map Data Type 2 (MD2)
S12,F17 Map Data Request Type 3 (MDR3)
S12,F18 Map Data Type 3 (MD3)
S12,F19 Map Error Report Send (MERS)
S12,F20 Not Used

7.17 Stream 13 Data Set Transfers
S13,F0 Abort Transaction (S13F0)
S13,F1 Send Data Set Send (DSSS)
S13,F2 Send Data Set Acknowledge (DSSA)
S13,F3 Open Data Set Request (DSOR)
S13,F4 Open Data Set Data (DSOD)
S13,F5 Read Data Set Request (DSRR)
S13,F6 Read Data Set Data (DSRD)
S13,F7 Close Data Set Send (DSCS)
S13,F8 Close Data Set Acknowledge (DSCA)
S13,F9 Reset Data Set Send (DSRS)
S13,F10 Reset Data Set Acknowledge (DSRA)
S13,F11 Data Set Object Multi-Block Inquire (DSOMGI)
S13,F12 Data Set Object Multi-Block Grant (DSOMBG)
S13,F13 Table Data Send (TDS)

S13,F14 Table Data Acknowledge (TDA)
S13,F15 Table Data Request (TDR)
S13,F16 Table Data (TD)

7.18 Stream 14 Object Services

S14,F0 Abort Transaction (S14F0)
S14,F1 GetAttr Request (GAR)
S14,F2 GetAttr Data (GAD)
S14,F3 SetAttr Request (SAR)
S14,F4 SetAttr Data (SAD)
S14,F5 GetType Request (GTR)
S14,F6 GetType Data (GTD)
S14,F7 GetAttrName Request (GANR)
S14,F8 GetAttrName Data (GAND)
S14,F9 Create Object Request (COR)
S14,F10 Create Object Acknowledge (CAO)
S14,F11 Delete Object Request
S14,F12 Delete Object Acknowledge (DOA)
S14,F13 Object Attach Request (OAR)
S14,F14 Object Attach Acknowledge (OAA)
S14,F15 Attached Object Action Request (AOAR)
S14,F16 Attached Object Action Acknowledge AOAA)
S14,F17 Supervised Object Action Request SOAR)
S14,F18 Supervised Object Action Acknowledge (SOAA)

7.19 Stream 15 Recipe Management

S15,F0 Abort Transaction (S15F0)
S15,F1 Recipe Management Multi-block Inquire
S15,F2 Recipe Management Multi-block Grant
S15,F3 Recipe Namespace Action Request
S15,F4 Recipe Namespace Action Acknowledge
S15,F5 Recipe Namespace Rename Request
S15,F6 Recipe Namespace Rename Acknowledge
S15,F7 Recipe Space Request
S15,F8 Recipe Space Data
S15,F9 Recipe Status Request
S15,F10 Recipe Status Data
S15,F11 Recipe Version Request
S15,F12 Recipe Version Data
S15,F13 Recipe Create Request
S15,F14 Recipe Create Acknowledge
S15,F15 Recipe Store Request
S15,F16 Recipe Store Acknowledge
S15,F17 Recipe Retrieve Request
S15,F18 Recipe Retrieve Data
S15,F19 Recipe Rename Request
S15,F20 Recipe Rename Acknowledge
S15,F21 Recipe Action Request
S15,F22 Recipe Action Acknowledge
S15,F23 Recipe Descriptor Request
S15,F24 Recipe Descriptor Data
S15,F25 Recipe Parameter Update Request
S15,F26 Recipe Parameter Update Acknowledge
S15,F27 Recipe Download Request
S15,F28 Recipe Download Acknowledge
S15,F29 Recipe Verify Request
S15,F30 Recipe Verify Acknowledge
S15,F31 Recipe Unload Request
S15,F32 Recipe Unload Data
S15,F33 Recipe Select Request
S15,F34 Recipe Select Acknowledge
S15,F35 Recipe Delete Request
S15,F36 Recipe Delete Acknowledge
S15,F37 DRNS Segment Approve Action Request
S15,F38 DRNS Segment Approve Action Acknowledge
S15,F39 DRNS Recorder Segment Request
S15,F40 DRNS Recorder Segment Acknowledge
S15,F41 DRNS Recorder Modify Request
S15,F42 DRNS Recorder Modify Acknowledge
S15,F43 DRNS Get Change Request
S15,F44 DRNS Get Change Request Data
S15,F45 DRNS Manager Segment Change Approval Request
S15,F46 DRNS Manager Segment Approval Acknowledge
S15,F47 DRNS Manager Rebuild Request
S15,F48 DRNS Manager Rebuild Acknowledge

7.20 Stream 16 Processing Management

S16,F0 Abort Transaction (S16F0)
S16,F1 Multi-block Process Job Data Inquire (PRJI)
S16,F2 Multi-block Process Job Data Grant (PRJG)
S16,F3 Process Job Create Request (PRJCR)
S16,F4 Process Job Create Acknowledge (PRJCA)
S16,F5 Process Job Command Request (PRJCMDR)
S16,F6 Process Job Command Acknowledge (PRJCMDA)
S16,F7 Process Job Alert Notify (PRJA)
S16,F8 Process Job Alert Confirm (PRJAC)
S16,F9 Process Job Event Notify (PRJE)
S16,F10 Process Job Event Confirm (PRJEC)
S16,F11 PRJobCreateEnh
S16,F12 PRJobCreateEnh Acknowledge
S16,F13 PRJobDuplicateCreate
S16,F14 PRJobDuplicateCreate Acknowledge
S16,F15 PRJobMultiCreate
S16,F16 PRJobMultiCreate Acknowledge

SEMI E5-0600
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 March 2, 2000. Initially available on SEMI OnLine April 2000; to be published June 2000. Originally published in 1982; previously published February 2000.

1 Introduction

1.1 *Intent* — 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 3.

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.

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

1.2.1 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.

1.2.2 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.

1.2.3 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.

1.3 *Application* — 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.

1.3.1 The minimum compliance to this standard involves meeting the few constraints outlined in Section 5. 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 7.

1.3.2 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.

1.4 *Applicable Documents*

1.4.1 *ANSI*[1]

1 ANSI, 1430 Broadway, New York, NY 10018

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

1.4.2 *IEEE*[2]

754 — Standard for Binary Floating Point Arithmetic

1.4.3 *SEMI Standards*[3]

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

SEMI E6 — Facilities Interface Specifications Guideline and Format

1.4.4 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 katakana characters in Japanese implementations of SECS-II.

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

NOTE 1: As listed or revised, all documents cited shall be the latest publications of adopted standards.

1.5 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 limitations prior to use.

2 Selected Definitions

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

2.1.1 *block* — a physical division of a message used by the message transfer protocol (see Section 3.3).

2.1.2 *conversation* — a sequence of related messages (see Section 5.4).

2.1.3 *conversation timeout* — an indication that a conversation has not completed properly (see Section 5.4.1).

2.1.4 *device ID* — a number between 0 and 32767 used in identifying the particular piece of equipment communicating with a host (see Section 3.4.1).

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

2.1.6 *function* — a specific message for a specific activity within a stream (see Section 4.2).

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

2.1.8 *interpreter* — the system that interprets a primary message and generates a reply when requested (see Section 3.2).

2.1.9 *item* — a data element within a message (see Section 6.2).

2.1.10 *item format* — a code used to identify the data type of an item (see Section 6.2).

2.1.11 *list* — a group of items (see Section 6.3).

2.1.12 *message* — a complete unit of communication (see Section 3.2).

2.1.13 *message header* — information about the message passed by the message transfer protocol (see Section 3.4).

2.1.14 *multi-block message* — a message sent in more than one block by the message transfer protocol (see Section 3.3.2).

2.1.15 *originator* — the creator of a primary message (see Section 3.2).

2.1.16 *packet* — a physical division of a message used by the message transfer protocol (see Section 3.3).

2.1.17 *primary message* — an odd numbered message. Also, the first message of a transaction (see Sections 3.2 and 4.2).

2.1.18 *reply* — the particular secondary message corresponding to a primary message (see Sections 3.2 and 4.2).

2.1.19 *secondary message* — an even-numbered message. Also the second message of a transaction (see Sections 3.2 and 4.2).

2.1.20 *single-block message* — a message sent in one block by the message transfer protocol (see Section 3.3.1).

2.1.21 *stream* — a category of messages (see Section 4.1).

2.1.22 *transaction* — a primary message and its associated secondary message, if any (see Section 5.2).

2.1.23 *transaction timeout* — an indication from the message transfer protocol that a transaction has not completed properly (see Section 3.5).

3 The Message Transfer Protocol

3.1 *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

2 IEEE Service Center, 445 Hoe Lane, Piscataway, NJ 08854

3 SEMI, 805 E. Middlefield Road, Mountain View, CA 94043

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.

3.2 *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 message at its destination and generates a reply when requested.

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

3.3.1 *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.

3.3.2 *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.

3.4 *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 2: In SECS-I, this information is contained in the 10-byte header of each block of a message.

3.4.1 *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.

3.4.2 *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.

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

3.5 *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 3: 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.

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

4 Streams and Functions

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

4.2 *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 7.4.

4.3 *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.

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

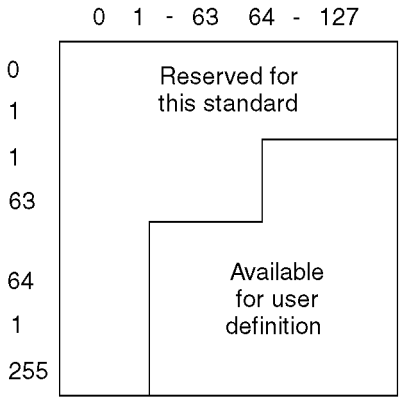


Figure 1
Stream and Function Allocation

4.3.2 The reserved codes assigned by this standard are listed in Section 7. 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 5.

5 Transaction and Conversation Protocols

5.1 *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.

5.2 *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.

5.3 *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 7.5.
2. For any received message that cannot be processed by the equipment, send the appropriate error message on Stream 9. As described in Section 7.13, S9-F1, F3, F5, F7, or F11 are possible.
3. Format any other supported messages according to Section 7.
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.

5.4 *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.

5.4.1 *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.

5.4.2 *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 7.17)

6. There is a conversation related to the handling of material between equipment. This conversation is described in detail in Stream 4. (See Section 7.8)

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.

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

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.

5.4.3 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.

6 Data Structures

6.1 *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.

6.2 *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.

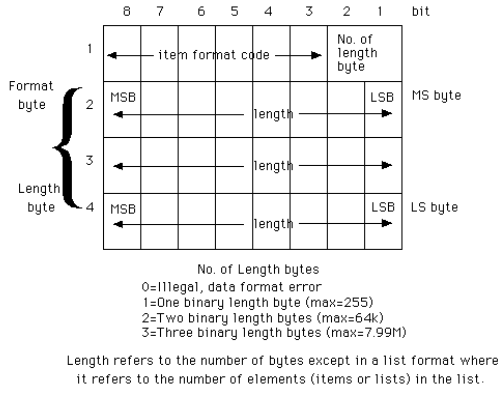


Figure 2
Item and List Header

6.2.1 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.

6.2.2 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, 15 are defined as shown in Table 1. Format code 0 is called a list and is defined in

E5 6.3. Format code 22 (octal) is called a localized string and is defined in SEMI E5 Section 6.4. 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. Signed integers 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.

6.3 *List* — A list is an ordered set of elements, where an element can be either an item (6.2) 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.

6.3.1 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.

Format Code		*Meaning*
Binary		*The data after the heading*
Bit 876543	*Octal*	*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.

6.4 *Localized Character String Items* — A localized character string is an item which is used for representing a string of multi-byte character. 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) .

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.

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

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	

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

6.5 *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)
00101110    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
```

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

6.6 *Data Item Dictionary* — This section defines the data items used in the standard SECS-II messages described in Section 7, 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.

ABS Format: 10

Any binary string

Where Used: S2F25, F26

ACDS Format: 32, 52

After Command Codes

```
Vector of all command codes which defined command must succeed within the
same block.
```

Where Used: S7F22

ACKA Format: 11

Indicates success of a request:

```
TRUE is successful else FALSE
```

Where Used: S5F14, F15, F18; S16F2, F4, F6, F12, F14, F16, F18, F24, F26, F28; S17F4, F8, F14

ACKC3 Format: 10

Acknowledge code, 1 byte

```
0   =   Accepted
>0  =   Error, not accepted
1-63 Reserved
```

Where Used; S3F6, F8, F10

ACKC5 Format: 10

Acknowledge code, 1 byte

```
0   =   Accepted
>0  =   Error, not accepted
1-63 Reserved
```

Where Used; S5F2, F4

ACKC6 Format: 10

Acknowledge code, 1 byte

```
0   =   Accepted
>0  =   Error, not accepted
1-63 Reserved
```

Where Used; S6F2, F4, F10, F12, F14

ACKC7 Format: 10

Acknowledge code, 1 byte

```
0    =   Accepted
1    =   Permission not granted
2    =   Length error
3    =   Matrix overflow
4    =   PPID not found
5    =   Mode unsupported
>5   =   Other error
6-63 Reserved
```

Where Used: S7F4, F12, F14, F16, F18, F24, F32

ACKC7A Format: 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
```

Where Used: S7F27

ACKC10 Format: 10

Acknowledge Code, 1 byte

```
0    =   Accepted for display
1    =   Message will not be displayed
2    =   Terminal not available
3-63 Reserved
```

Where Used: S10F2, F4, F6, F10

ACKC13 Format: 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
```

Where Used: S13F2, F4, F6, F8

AGENT Format: 20

Where Used: S15F11, 12, 21, 22, 25

ALCD Format: 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
```

Where Used: S5F1, F6

ALED Format: 10

Alarm enable/disable code, 1 byte

```
bit 8 = 1 means enable alarm
bit 8 = 0 means disable alarm
```

Where Used: S5F3

ALID Format: 3(), 5()

Alarm identification

Where Used: S5F1, F3, F5, F6

ALTX Format: 20

Alarm text limited to 40 characters

Where Used: S5F1, F6

ATTRDATA Format: 0, 10, 20, 3(), 4(), 5()

Contains a specific attribute value for a specific object.

Where Used: S1F20; S13F14, F16; S14F1, F2, F3, F4, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18

ATTRID Format: 20, 5()

Identifier for an attribute for a specific type of object.

Where Used: S1F19; S13F14, F16; S14F1, F2, F3, F4, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18; S18F1, F3

ATTRRELN Format: 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.
```

Where Used: S14F1

BCDS Format: 32, 52

Before Command Codes

```
Vector of all command codes which defined command must precede within the
same block.
```

Where Used: S7F22

BCEQU Format: 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.
```

Where Used: S12F3, F4

BINLT Format: 20, 51

The Bin List

```
Is an array of bin values. Format must be the same as used in NULBC and
BCEQU.
```

Where Used: S12F7, F9, F11, F14, F16, F18

BLKDEF Format: 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.

Where Used: S7F22

BPD Format: 10

Boot program Data

Where Used: S8F2

BYTMAX Format: 3(), 5()

Byte Maximum

Maximum length of process program. A value of zero indicates no limit. Negative values are invalid.

Where Used: S7F22

CAACK Format: 51

Carrier Action Acknowledge Code. One 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.
6-63		reserved.

Where Used: S3F18, F20, F22, F24, F26

CARRIERACTION Format: 20

Specifies the action requested for a carrier.

Where Used: S3F17

CARRIERID Format: 20

The identifier of a carrier.

CARRIERSPEC Format: 20

The object specifier for a carrier. Conforms to OBJSPEC.

Where Used: S3F17

CATTRDATA Format: 0, 10, 20, 3(), 4(), 5()

The value of a carrier attribute.

Where Used: S3F17

CATTRID Format: 20

The name of a carrier attribute.

Where Used: S3F17

CCODE Format: 32, 52

Command Code

```
Each command code corresponds to a unique process operation the machine
is capable of performing.
```

Where Used: S7F22, F23, F26, F31

CEED Format: 11

Collection event or trace enable/disable code, 1 byte

```
FALSE = Disable
TRUE = Enable
```

Where Used: S2F37; S17F5

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

Collected event ID

Where Used: S2F35, F37; S6F3, F8, F9, F11, F13, F15, F17; S17F5, F9, F10, F11, F12

CEPACK Format: 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
```

Where Used: S2F50

CEPVAL Format: 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
```

Where Used: S2F49

CKPNT Format: 54

Checkpoint as defined by the sending system

Where Used: S13F3, F6

CMDA Format: 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
```

Where Used: S2F22, F28

CMDMAX Format: 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.
```

Where Used: S7F22

CNAME Format: 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.
```

Where Used: S7F22

COLCT Format: 5()

Column count in die increments

Where Used: S12F1, F4

COLHDR Format: 20

Text description of contents of TBLELT. 1 - 20 characters

Where Used: S13F13, F16

COMMACK Format: 10

Establish Communications Acknowledge Code, 1 byte

```
0    =   Accepted
1    =   Denied, Try Again
2-63 Reserved
```

Where Used: S1F14

CPACK Format: 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
```

Where Used: S2F42

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

Command Parameter Name

Where Used: S2F41, F42, F49, F50; S4F21, F29; S16F5, F27

CPVAL Format: 10, 11, 20, 21, 3(), 5()

Command Parameter Value

Where Used: S2F41, F49; S4F21, F29; S16F5, F27; S18F13

CSAACK Format: 10

Equipment Acknowledgement code, 1 byte

```
0    =    Everything correct
1    =    Busy
2    =    Invalid SPID
3    =    Invalid data
>3   =    Equipment-specific error
4-63 Reserved
```

Where Used: S2F8

CTLJOBCMD Format: 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
```

Where used: S16F27

CTLJOBID Format: 20

Identifier for Control Job. Conforms to OBJID.

Where used: S16F27

DATA Format: 20

A vector or string of unformatted data.

Where Used: S18F6,F7

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

Data ID

Where Used: S2F33, F35, F39, F45, F49; S3F15, F17; S4F19, F25; S6F3, F5, F7, F8, F9, F11, F13, F16, F18, F27; S13F11, F13, F15; S15F27, F29, F33, F35, F37, F39, F41, F43, F45, F47; S16F1, F3, F5, F11, F13; S17F1, F5, F9

DATALENGTH Format: 3(), 5()

Total bytes to be sent

Where Used: S2F39; S3F15; S4F25; S6F5; S13F11; S16F1, F11; S18F5, F7

DATASEG Format: 20

Used to identify the data requested.

Where Used: S18F5,F7

DATASRC Format: 20

Object type for Data Source Objects

Where Used: S14F1, F3, F6, F7, F8; S17F1

DATLC Format: 51

Data location

```
Location of invalid data, represented in bytes, measured from the start of
the message in question, excluding all header bytes.
```

Where Used: S12F19

DRACK Format: 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
```

Where Used: S2F34

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

Data set ID

Where Used: S6F3, F8, F9

DSNAME Format: 20

The name of the Data Set

```
The minimum length is zero. The maximum is 50 characters.
```

Where Used: S13F1, F2, F3, F4

DSPER Format: 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.
```

Where Used: S2F23

DUTMS Format: 20

Die Units of Measure

```
Use units description per SEMI E5 Section 9.
```

Where Used: S12F1, F4

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

Data value name

Where Used: S6F3, F8

DVVAL Format: 0, 10, 11, 20, 21, 3(), 4(), 5()

Data value

Where Used: S6F3, F8, F9

EAC Format: 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
```

Where Used: S2F16

ECDEF Format: 10, 11, 20, 21, 3(), 4(), 5()

Equipment constant default value

Where Used: S2F30

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

Equipment Constant ID

Where Used: S2F13, F15, F29, F30

ECMAX Format: 10, 11, 20, 21, 3(), 4(), 5()

Equipment constant maximum value

Where Used: S2F30

ECMIN Format: 10, 11, 20, 21, 3(), 4(), 5()

Equipment constant minimum value

Where Used: S2F30

ECNAME Format: 20

Equipment constant name

Where Used: S2F30

ECV Format: 10, 11, 20, 21, 3(), 4(), 5()

Equipment Constant Value

Where Used: S2F14, F15

EDID Format: 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]
```

Where Used: S9F13

EMID Format: 10, 20

Equivalent material ID (16 bytes maximum)

Where Used: S3F9

EPD Format: 10

Executive program data

Where Used: S8F4

EQNAME Format: 20

A unique ASCII equipment identifier assigned by the factory to the equipment. Limited to a maximum of 80 characters.

Where Used: S4F29

ERACK Format: 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
```

Where Used: S2F38

ERRCODE Format: 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
 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
48-63 Reserved
```

Where Used: S1F20; S3F16; S4F20, F22, F23, F33, F35; S5F14, F15, F18; S13F14, F16; S14F2, F4, F6, F8, F10, F12, F14, F16, F18; S15F28, F18, F20, F22, F24, F26, F30, F32, F34, F36, F38, F40, F42, F44, F48; S16F12, F14, F16, F18, F24, F26, F28; S17F2, F4, F6, F8, F10, F12, F14

ERRTEXT Format: 20

Text string describing the error noted in the corresponding ERRCODE. Limited to 80 characters maximum.

Where Used: S1F20; S3F16; F18, F20, F22, F24, F26, S4F20, F22, F23, F33, F35; S5F14, F15, F18; S13F14, F16; S14F2, F4, F6, F8, F10, F12, F14, F16, F18; S15F28, F30, F32, F34, F36, F38, F40, F42, F44, F48; S16F12, F14, F16, F18, F24, F26, F28; S17F4, F8, F14

ERRW7 Format: 20

Text string describing error found in process program.

Where Used: S7F27

EVNTSRC Format: 20

Object type for Event Source Objects

Where Used: S17F5, F9, F10, F11, F12

EXID Format: 20

Unique identifier for the exception. Maximum length of 20 characters.

Where Used: S5F9, F11, F13, F14, F15, F17, F18

EXMESSAGE Format: 20

Text which describes the nature of the exception.

Where Used: S5F9, F11

EXRECVRA Format: 20

Text which specifies a recovery action for an exception. Maximum length of 40 bytes.

Where Used: S5F9, F13

EXTYPE Format: 20

Text which identifies the type of an exception. It is usually a single word of text.

```
"ALARM"
"ERROR"
```

Where Used: S5F9, F11; S14F1, F2, F8

FCNID Format: 51

Function Identification

Where Used: S2F43, F44

FFROT Format: 52

Film Frame Rotation

In degrees from the bottom CW. (Bottom equals zero degrees.) Zero length indicates not used.

Where Used: S12F1, F3

FILDAT Format: 10, 20

Data from the Data Set

The maximum length is the RECLEN from Open Data Set Data.

Where Used: S13F6

FNLOC Format: 52

Flat/Notch Location

In degrees from the bottom CW. (Bottom equals zero degrees.) Zero length indicates not used.

Where Used: S12F1, F3, F4

FRMLEN Format: 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.

Where Used: S7F34

GRANT Format: 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
```

Where Used: S2F2, F40; S3F16; S4F26; S13F12; S16F2, F12

GRANT6 Format: 10

Permission to send, 1 byte

```
0    =   Permission granted
1    =   Busy, try again
2    =   Not interested
>2   =   Other errors
3-63 Reserved
```

Where Used: S6F6

GRNT1 Format: 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
```

Where Used: S12F6

HANDLE Format: 3(), 5()

Logical unit or channel

Where Used: S13F3, F4, F5, F6, F7, F8

HCACK Format: 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
```

Where Used: S2F42, F50

HOACK Format: 11

Conveys whether the corresponding handoff activity succeeded (=True) or failed (=False)

Where Used: S4F31, F33

HOCANCELACK Format: 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
```

Where Used: S4F13

HOCMDNAME Format: 20

Identifier for the handoff command to be executed

Where Used: S4F31

HOHALTACK Format: 51

Tells whether the halt command was accepted or rejected

```
0    =    Halt Accepted
1    =    Atomic Transfer Unknown
2-63 Reserved
```

Where Used: S4F41

IACDS Format: 32, 52

Immediately After Command Codes

```
Vector of all command codes which defined command must immediately succeed
within the same block.
```

Where Used: S7F22

IBCDS Format: 32, 52

Immediately Before Command Codes

```
Vector of all command codes which defined command must immediately precede
within the same block.
```

Where Used: S7F22

IDTYP Format: 10

Id type

```
0    =    Wafer ID
1    =    Wafer Cassette ID
2    =    Film Frame ID
>2   =    Error
3-63 Reserved
```

Where Used: S12F1, F3, F4, F5, F7, F9, F11, F13, F14, F15, F16, F17, F18

LENGTH Format: 3(), 5()

Length of the service program or process program in bytes

Where Used: S2F1; S7F1, F29

LIMITACK Format: 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
```

Where Used: S2F46

LIMITID Format: 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.

Where Used: S2F45, F46, F48

LIMITMAX Format: 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.

Where Used: S2F48

LIMITMIN Format: 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.

Where Used: S2F48

LINKID Format: 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.

Where Used: S6F25; S15F22, F30

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

Lower limit for numeric value

Where Used: S7F22

LOC Format: 10

Machine material location code, 1 byte

Where Used: S2F27; S3F2

LOWERDB Format: 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.

Where Used: S2F45, F48

LRACK Format: 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
```

Where Used: S2F36

LVACK Format: 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
```

Where Used: S2F46

MAPER Format: 10

Map Error

```
0    =   ID not found
1    =   Invalid Data
2    =   Format Error
>2   =   Invalid error
3-63 Reserved
```

Where Used: S12F19

MAPFT Format: 10

Map data format type

```
0   =   Row format
1   =   Array format
2   =   Coordinate format
>2  =   Error
3-63 Reserved
```

Where Used: S12F3, F5

MCINDEX Format: 5()

Identifier used to link a handoff command message with its eventual completion message. Corresponding messages carry the same value for this data item.

Where Used: S4F31, F33

MDACK Format: 10

Map data acknowledge

```
0   =   Map received
1   =   Format error
2   =   No ID match
3   =   Abort/discard map
>3  =   Error
4-63 Reserved
```

Where Used: S12F8, F10, F12

MDLN Format: 20

Equipment Model Type, 6 bytes max

```
Same data as returned by S1,F2
```

Where Used: S1F2, F13, F14; S7F22, F23, F26, F31

MEXP Format: 20

Message expected in the form SxxFyy where x is stream and y is function.

Where Used: S9F13

MF Format: 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 9
```

Where Used: S3F2, F4, F5, F7 S16F11, F13, F15

MHEAD Format: 10

SECS message block header associated with message block in error

Where Used: S9F1, F3, F5, F7, F11

MID Format: 10, 20

Material ID

```
16 Characters maximum
```

Where Used: 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 Format: 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
```

Where Used: S3F14

MIDRA Format: 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
```

Where Used: S3F12

MLCL Format: 5()

Message length

```
Defined by message size in (bytes)
```

Where Used: S12F4, F5

MMODE Format: 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
```

Where Used: S7F15

NACDS Format: 32, 52

Not After Command Codes

```
Vector of all command codes which defined command may not succeed within the
same block.
```

Where Used: S7F22

NBCDS Format: 32, 52

Not Before Command Codes

```
Vector of all command codes which defined command may not precede within the
same block.
```

Where Used: S7F22

NULBC Format: 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.
```

Where Used: S12F1, F3, F4

OBJACK Format: 51

Acknowledge code:

```
0   =   Successful completion of requested data
1   =   Error
>1 Reserved
```

Where Used: S14F2, F4, F6, F8, F10, F12, F14, F16, F18

OBJCMD Format: 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 Format: 20, 5()

Identifier for an object

Where Used: S1F19; S14F1, F2, F3, F4

OBJSPEC Format: 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.

Where Used: S2F49; S13F11, F13, F15; S14F1, F3, F5, F7, F9, F10, F11, F13, F15, F16, F17; S15F43, F47

OBJTOKEN Format: 54

Token used for authorization.

Where Used: S14F14, F15; S15F37, F39, F41, F43

OBJTYPE Format: 20, 5()

Identifier for a group or class of objects. All objects of the same type must have the same set of attributes available.

Where Used: S1F19; S14F1, F3, F6, F7, F8

OFLACK Format: 10

Acknowledge code for OFF-LINE request.

```
0    =    OFF-LINE Acknowledge
1-63 Reserved
```

Where Used: S1F16

ONLACK Format: 10

Acknowledge code for ON-LINE request.

```
0    =    ON-LINE Accepted
1    =    ON-LINE Not Allowed
2    =    Equipment Already ON-LINE
3-63 Reserved
```

Where Used: S1F18

OPID Format: 5()

Operation ID. A unique integer generated by the requestor of an operation, used where multiple completion confirmations may occur.

Where Used: S6F25; S15F21, F29, F30, F37, F41, F44, F46

ORLOC Format: 10

Origin Location

```
implicit value of (0,0)
0    =    Center die of wafer
(Determined by:

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
```

Where Used: S12F1, F3, F4

PARAMNAME Format: 20

The name of a parameter in a request.

Where Used: S3F21,F23

PARAMVAL Format: 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.

Where Used: S3F21,F23

PDFLT Format: 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.
```

Where Used: S7F22

PFCD Format: 10

Predefined form code, 1 byte

Where Used: S6F9

PGRPACTION Format: 20

The action to be performed on a port group.

Where Used: S3F23

PMAX Format: 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
```

Where Used: S7F22

PNAME Format: 20

Parameter Name ≤ 16 characters

```
Text string identifying the parameter value expected by its parent process
command.
```

Where Used: S7F22

PORTACTION Format: 20

The action to be performed on a port.

Where Used: S3F25

PORTGRPNAME Format: 20

The identifier of a group of ports.

Where Used: S3F21,F23

PPARM Format: 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.
```

Where Used: S7F23, F26, F31

PPBODY Format: 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.
```

Where Used: S7F3, F6, F36

PPGNT Format: 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
```

Where Used: S7F2, F30

PPID Format: 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.
```

Where Used: S2F27; S7F1, F3, F5, F6, F8, F10, F11, F13, F17, F20, F23, F25, F26, F27, F31, F33, F34, F36; S9F13

PRAXI Format: 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
```

Where Used: S12F1, F3

PRCMDNAME Format: 20

Commands sent to a process job:

"START"
"STOP"
"PAUSE"
"RESUME"
"ABORT"
"CANCEL"

Where Used: S16F5

PRDCT Format: 5()

Process Die Count

```
Number of die to be processed or number of die which have been processed.
(Zero length indicates not used.)
```

Where Used: S12F1, F4

PREVENTID Format: 5()

Processing related event identification:

```
1 = Waiting for material
2 = Job state change
```

Where Used: S16F9

PRJOBID Format: 20

Text string which uniquely identifies a process job.

Where Used: S16F4, F5, F6, F7, F9, F11, F12, F13, F14, F15, F16, F17, F20, F23, F25

PRJOBMILESTONE Format: 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
```

Where Used: S16F7

PRJOBSPACE Format: 52

The number of process jobs that can be created.

Where used: S16F22

PRPAUSEEVENT Format: 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.

Where used: S16F11, F13, F15

PTN Format: 10, 51

Material Port number, 1 byte

Where Used: S3F17, F21, F25; S4F1, F3, F5, F7, F9, F11, F13, F15, F17; S16F13, F17, F21

QUA Format: 10

Quantity in format, 1 byte

Where Used: S3F2, F4, F5, F7

RAC Format: 31, 51

Reset acknowledge, 1 byte

```
0    =   Reset to be done
1    =   Reset denied
>1   =   Other errors
2-63 Reserved
```

Where Used: S2F20

RCMD Format: 20, 31, 51

Remote command code or string

Where Used: S2F21, F41, F49

RCPATTRDATA Format: 0, 10, 11, 20, 3(), 4(), 5()

The contents (value) of a recipe attribute.

Where Used: S6F25; S15F13, F15, F18, F27, F28, F30, F32

RCPATTRID Format: 20

The name (identifier) of a non-identifier recipe attribute.

Where Used: S6F25; S15F13, F15, F18, F27, F28, F30, F32

RCPBODY Format: 10, 20, 3(), 5()

Recipe body

Where Used: S15F13, F15, F18, F27, F32

RCPCLASS Format: 20

Recipe class

Where Used: S15F11

RCPCMD Format: 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
```

Where Used: S15F21, F22

RCPDEL Format: 51

```
0   =   Delete
1   =   Deselect
>1 Reserved
```

Where Used: S15F35

RCPDESCLTH Format: 5()

The length in bytes of a recipe section.

Where Used: S15F24

RCPDESCNM Format: 20

Identifies a type of descriptor of a recipe: “ASDesc”, “BodyDesc”, “GenDesc”.

Where Used: S15F24

RCPDESCTIME Format: 20

The timestamp of a recipe section, in the format “YYYYMMDDhhmmsscc”.

Where Used: S15F24

RCPID Format: 20

Recipe identifier. Formatted text conforming to the requirements of OBJSPEC.

Where Used: S15F21, F23, F28, F29, F30, F33, F35, F37, F41, F44

RCPNAME Format: 20

Recipe name

Where Used: S15F11

RCPNEWID Format: 20

The new recipe identifier assigned as the result of a copy or rename operation.

Where Used: S15F19, F41, F44

RCPOWCODE Format: 11

Indicates whether any pre-existing recipe is to be overwritten (=TRUE) or not (=FALSE) on download.

Where Used: S15F27

RCPPARNM Format: 20

The name of a recipe variable parameter. Maximum length of 40 characters.

Where Used: S15F25, F33; S16F3, F11, F13, F15, F23

RCPPARRULE Format: 20

The restrictions applied to a recipe variable parameter setting. Maximum length of 80 characters.

Where Used: S15F25

RCPPARVAL Format: 10, 11, 20, 3(), 4(), 5()

The initial setting assigned to a recipe variable parameter. Text form restricted to maximum of 80 characters.

Where Used: S15F25, F33; S16F3, F11, F13, F15, F23

RCPRENAME Format: 11

Indicates whether a recipe is to be renamed (= TRUE) or copied (= FALSE).

Where Used: S15F19

RCPSECCODE Format: 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
```

Where Used: S15F15, F16, F17

RCPSECNM Format: 20

Recipe section name: “Generic”, “Body”, or “ASDS”.

Where Used: S15F15, F18

RCPSPEC Format: 20

Recipe specifier. The object specifier of a recipe.

Where Used: S15F1, F9, F13, F15, F17, F19, F27, F28, F31, F32, F45; S16F11, F13, F15, F23

RCPSTAT Format: 51

The status of a managed recipe

```
0   =   Does not exist
8   =   Unprotected
9   =   Protected
```

Where Used: S15F10

RCPUPDT Format: 11

Indicates if an existing recipe is to be updated (= True) or a new recipe is to be created (= False).

Where Used: S15F13

RCPVERS Format: 20

Recipe version

Where Used: S15F10, F12

READLN Format: 3(), 5()

Maximum length to read

Where Used: S13F5

RECLEN Format: 3(), 5()

Maximum length of a Discrete record

Where Used: S13F4

REFP Format: 3()

Reference Point

Where Used: S12F1, F4

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

Reporting group size

Where Used: S2F23; S17F5

RESC Format: 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
```

Where Used: S7F22

RESPEC Format: 20

Object specifier for the recipe executor

Where Used: S15F29, F33, F35

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

Resolution value for numeric data

```
If RESC=1, then RESV contains smallest increment allowed for parameter
```

Where Used: S7F22

RIC Format: 31, 51

Reset code, 1 byte

```
0   =   Not used
1   =   Power up reset
>1 Other reset conditions
2-63 Reserved
```

Where Used: S2F19

RMACK Format: 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
```

Where Used: S15F4, F6, F8, F10, F12, F14, F16, F18, F20, F22, F24, F25, F26, F28, F30, F32, F34, F36, F38, F40, F42, F44, F48

RMCHGSTAT Format: 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
```

Where Used: S15F25

RMCHGTYPE Format: 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
```

Where Used: S15F37, F41, F44, F46, F47, F48

RMDATASIZE Format: 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.

Where Used: S15F1

RMGRNT Format: 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
```

Where Used: S15F2, F37, F46

RMNEWNS Format: 20

New name (identifier) assigned to a recipe namespace

Where Used: S15F5

RMNSCMD Format: 51

Action to be performed on a recipe namespace

```
1   =   Created
5   =   Deleted
0, 2-4, 6-63 Reserved
```

Where Used: S15F3

RMNSSPEC Format: 20

The object specifier of a recipe namespace.

Where Used: S15F3, F5, F7, F11, F21, F23, F25, F47

RMRECSPEC Format: 20

The object specifier of a distributed recipe namespace recorder.

Where Used: S15F39, F41, F43

RMREQUESTOR Format: 11

Set to TRUE if initiator of change request was an attached segment. Set to FALSE otherwise.

Where Used: S15F41, F44, F46

RMSEGSPEC Format: 20

The object specifier of a distributed recipe namespace segment.

Where Used: S15F37, F39, F41, F44, F46, F47

RMSPACE Format: 5()

The amount of storage available for at least one recipe in a recipe namespace, in bytes.

Where Used: S15F8

ROWCT Format: 5()

Row count in die increments

Where Used: S12F1, F4

RPSEL Format: 51

Reference Point Select

```
Number of reference point from 0-n.
```

Where Used: S12F1, F4

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

Report ID

Where Used: S2F33, F35; S6F11, F13, F16, F11, F19, F21, F22; S17F1, F2, F3, F4, F5, F9, F11, F12

RPTOC Format: 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.

Where Used: S14F1, F2, F3, F4; S17F5

RQCMD Format: 11

Required Command

```
True = Command must be specified
False = Command is optional
```

Where Used: S7F22

RQPAR Format: 11

Required Parameter

```
True = Parameter must be specified
False = Parameter is optional
```

Where Used: S7F22

RRACK Format: 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
```

Where Used: S4F18

RSACK Format: 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
```

Where Used: S4F2

RSDA Format: 10

Request Spool Data Acknowledge

```
0    =   OK
1    =   Denied, busy try later
2    =   Denied, spooled data does not exist
3-63 Reserved
```

Where Used: S6F24

RSDC Format: 51

Request Spool Data Code

```
0    =   Transmit Spooled Messages
1    =   Purge Spooled Messages
2-63 Reserved
```

Where Used: S6F23

RSINF Format: 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.
```

Where Used: S12F7, F14

RSPACK Format: 10

Reset Spooling Acknowledge

```
0    =   Acknowledge, spooling setup accepted
1    =   Spooling setup rejected
2-63 Reserved
```

Where Used: S2F44

RTYPE Format: 3(), 5()

Type of record

```
0    =   Stream
1    =   Discrete
2-63 Reserved
```

Where Used: S13F4

SDACK Format: 10

Map set-up data acknowledge

```
0    =   Received data
>1   =   Error
1-63 Reserved
```

Where Used: S12F2

SDBIN Format: 10

Send bin information flag

```
0    =   Sent bin information
1    =   Don't send bin information
>1   =   Error
2-63 Reserved
```

Where Used: S12F17

SEQNUM Format: 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.
```

Where Used: S7F27

SFCD Format: 10

Status form code, 1 byte

Where Used: S1F5

SHEAD Format: 10

Stored header related to the transaction timer

Where Used: S9F9

SLOTID Format: 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.

Where used: S16F11, F13, F15

SMPLN Format: 3(), 5()

Sample number

Where Used: S6F1

SOFTREV Format: 20

Software revision code 6 bytes maximum

Where Used: S1F2, F13, F14; S7F22, F23, F26, F31

SPAACK Format: 10

Equipment acknowledgement code, 1 byte

```
0    = Everything correct
1    = Invalid data
>1   = Equipment-specific error
2-63 Reserved
```

Where Used: S2F4

SPD Format: 10

Service program data

Where Used: S2F3, F6

SPID Format: 20

Service program ID, 6 characters

Where Used: S2F1, F5, F7, F9, F12; S9F13

SPR Format: Device Dependent

Service program results

```
Device dependent
```

Where Used: S2F10

SSACK Format: 20

Indicates the success or failure of a requested action. Two characters.

Where Used: S18F2, F4, F6, F8, F10, F12, F14

SSCMD Format: 20

Indicates an action to be performed by the subsystem.

Where Used: S18F13

STATUS Format: 20

Provides status information for a subsystem component.

Where Used: S18F12

STEMP Format: 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.
```

Where Used: S7F22

STIME Format: 20

Sample time, 12 or 16 bytes

```
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
```

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.

Where Used: S6F1

STRACK Format: 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
```

Where Used: S2F44

STRID Format: 51

Stream Identification

Where Used: S2F43, F44

STRP Format: 3()

Starting position in die coordinate position. Must be in (X,Y) order.

Where Used: S12F9, F16

SV Format: 0, 10, 11, 20, 21, 3(), 4(), 5()

Status variable value

Where Used: S1F4; S6F1

SVID Format: 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.
```

Where Used: S1F3, F11, F12; S2F23

SVNAME Format: 20

Status Variable Name

Where Used: S1F12

TARGETID Format: 20

Description. Identifies where a request for action or data is to be applied. If text, conforms to OBJSPEC.

Where Used: S18F1,F3,F5,F7,F9,F11,F13

TARGETSPEC Format: 20

Object specifier of target object

Where Used: S14F17; S15F43

TBLACK Format: 51

Indicates success or failure

```
0   =   Success
1   =   Failure
```

Where Used: S13F14, F16

TBLCMD Format: 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
```

Where Used: S13F13, F15

TBLELT Format: 0, 10, 11, 20, 21, 3(), 4(), 5()

Table element. The first table element in a row is used to identify the row.

Where Used: S13F13, F15, F16

TBLID Format: 20

Table identifier. Text conforming to the requirements of OBJSPEC.

Where Used: S13F13, F15, F16

TBLTYP Format: 20

A reserved text string to denote the format and application of the table. Text conforming to the requirements of OBJSPEC.

Where Used: S13F13, F15, F16

TEXT Format: 10, 20, 22, 3(), 5()

A single line of characters

Where Used: S10F1, F3, F5, F9

TIAACK Format: 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
```

Where Used: S2F24

TIACK Format: 10

Time Acknowledge Code, 1 byte

```
0    =   OK
1    =   Error, not done
2-63 Reserved
```

Where Used: S2F32

TID Format: 10

Terminal number, 1 byte

```
0    =   Single or main terminal
>0   =   Additional terminals at the same equipment
```

Where Used: S10F1, F3, F5, F7

TIME Format: 20

Time of day, 12 or 16 bytes

```
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                 0000 to 9999
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
```

NOTE 5: 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.

Where Used: S2F18, F31

TIMESTAMP Format: 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
```

Where Used: S5F9, F11, F15; S15F41, F44; S16F5, F7, F9

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

Total samples to be made

Where Used: S2F23; S17F5

TRACK Format: 11

Tells whether the related transfer activity was successful (=True) or unsuccessful (=False).

Where Used: S4F20, F22, F23

TRATOMICID Format: 5()

Equipment assigned identifier for an atomic transfer

Where Used: S4F20

TRAUTOD Format: 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.

Where Used: S14F1, F2, F3, F4; S17F5

TRAUTOSTART Format: 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.

Where Used: S4F19

TRCMDNAME Format: 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)

Where Used: S4F21

TRDIR Format: 51

Direction of handoff.

```
1   =   Send material
2   =   Receive material
0, 3-63 Reserved
```

Where Used: S4F19, F29

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

Trace request ID

Where Used: S2F23; S6F1, F27, F28; S17F5, F6, F7, F8, F13, F14

TRJOBID Format: 10

Equipment assigned identifier for the transfer job.

Where Used: S4F20, F21

TRJOBMS Format: 51

Milestone for a transfer job (e.g., started or complete).

```
1   =   Transfer Job Started
2   =   Transfer Job Complete
0, 3-63 Reserved
```

Where Used: S4F23

TRJOBNAME Format: 20

Host assigned identifier for the transfer job. Limited to a maximum of 80 characters.

Where Used: S4F19, F23

TRLINK Format: 5()

Common identifier for the atomic transfer used by the transfer partners to confirm that they are working on the same host-defined task.

Where Used: S4F19, F29, F31, F33, F39

TRLOCATION Format: 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.

Where Used: S4F19, F29

TROBJNAME Format: 20

Identifier for the material (transfer object) to be transferred

Where Used: S4F19, F23, F29

TROBJTYPE Format: 5()

Type of object to be transferred

Where Used: S4F19, F29

TRPORT Format: 5()

Identifier of the equipment port to be used for the handoff

Where Used: S4F19, F29

TRPTNR Format: 20

Name of the equipment which will serve as the other transfer partner for this atomic transfer. This corresponds to EQNAME.

Where Used: S4F19, F29

TRPTPORT Format: 5()

Identifier of the transfer partner's port to be used for the transfer.

Where Used: S4F19, F29

TRRCP Format: 20

Name of the transfer recipe for this handoff. Limited to a maximum of 80 characters.

Where Used: S4F19

TRROLE Format: 51

Tells whether the equipment is to be the primary or secondary transfer partner.

```
1   =   Primary Transfer Partner
2   =   Secondary Transfer Partner
```

Where Used: S4F19, F29

TRSPER Format: 4()

A Trace Object attribute which holds the value for sampling interval time.

Where Used: S14F1, F2, F3, F4; S17F5

TRTYPE Format: 51

Tells whether the equipment is to be an active or passive participant in the transfer.

```
1   =   Active
2   =   Passive
```

Where Used: S4F19, F23, F29

TSIP Format: 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
```

Where Used: S1F10

TSOP Format: 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
```

Where Used: S1F10

TTC Format: 3(), 5()

Time to completion

Where Used: S3F4

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

Upper limit for numeric value

Where Used: S7F22

UNFLEN Format: 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.
```

Where Used: S7F34

UNITS Format: 20

Units Identifier

```
As allowed by SEMI E5 Section 9
```

Where Used: S1F12; S2F30, F48; S7F22

UPPERDB Format: 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.

Where Used: S2F45, F48

V Format: 0, 10, 11, 20, 21, 3(), 4(), 5()

Variable data

Where Used: S6F11, F13, F16, F20, F22

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

Variable ID

Where Used: S2F33, F45, F46, F47, F48; S6F13, F18, F22; S17F1

VLAACK Format: 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
```

Where Used: S2F46

XDIES Format: 4(), 5()

X-axis die size (index)

Where Used: S12F1, F4

XYPOS Format: 3()

X and Y Coordinate Position. Must be in (X,Y) order

Where Used: S12F11, F18

YDIES Format: 4(), 5()

Y-axis die size (index)

Where Used: S12F1, F4

6.7 *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 (SV's) always contain valid information, while data values (DVVAL's) may only be valid upon the occurrence of a particular event. All equipment constants (ECV's) 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.

AlarmID Format: 3(), 5()

Class: DVVAL

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.

AlarmsEnabled Format: 0

Class: SV

Contains the list of alarms (ALIDs) enabled for reporting (via Stream 5).

Structure: L,n n= # of alarms enabled

```
1.<ALID1>
.
.
n.<ALIDn>
```

AlarmsSet Format: 0

Class: SV

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.<ALID1>
.
.
n.<ALIDn>
```

ARAMSAccumReset Format: 20

Class: SV

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.

ARAMSInfo Format: 20

Class: SV

Text field set by the equipment to provide additional information concerning an ARAMS state change.

ARAMSState Format: 20

Class: SV

The ARAMS code corresponding to the current state/substate. Four characters.

ARAMSText Format: 20

Class: SV

Text describing the ARAMSState. 3-80 characters.

ARAMSTimeStamp Format: 20

Class: SV

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.

CLOCK Format: 20

Class: SV

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
```

ControlState Format: 10, 51

Class: SV

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
```

CycleCtr Format: 5()

Class: SV

The number of machine cycles during the lifetime of the equipment. Non-resettable.

DowntimeAlarm Format: 3(), 5()

Class: SV

Identifier of the last alarm or exception triggering an equipment-initiated transition to UNSCHEDULED DOWNTIME from the PRODUCTIVE or STANDBY states.

DowntimeAlarmText Format: 20

Class: SV

Text associated with DowntimeAlarm. 0-80 characters.

DowntimeData Format: 20

Class: SV

Equipment defined data associated with transitions to, or within, the SCHEDULED or UNSCHEDULED DOWNTIME states. For example, this may be used to carry fault information, the component serial number of a repaired component, or comments entered at the equipment's control panel. 0-256 characters.

EngTime Format: 5()

Class: ECV

Accumulation of time in ENGINEERING reported in minutes.

EqpModel Format: 20

Class: SV

Text string describing the equipment model. 1-80 characters.

EqpName Format: 20

Class: ECV

Text string containing a user-assigned name for equipment. 1-80 characters. Information in the data item EQNAME is a subset of EqpName.

EqpSerialNum Format: 20

Class: SV

Text string describing the product serial number assigned by the manufacturer. 1-80 characters. Information in the data item MDLN is a subset of EqpSerialNum.

EstablishCommunicationsTimeout Format: 52

Class: ECV

The length of time, in seconds, of the interval between attempts to send S1F13 when establishing communications.

EventsEnabled Format: 0

Class: SV

Contains the list of events (CEIDs) enabled for reporting (via Stream 6).

Structure: L,n n= # of events enabled

```
1.<CEID1>
.
.
n.<CEIDn>
```

EventLimit Format: 0, 10, 11, 20, 21, 3(), 4(), 5()

Class: DVVAL

Used with the Limits Monitoring capability, it contains the LIMITID of the limit reached or crossed by LimitVariable. Since multiple zone transitions for a variable may occur simultaneously (e.g., due to identical limit definitions or a slow data sampling rate), EventLimit has been defined to allow for a list of LIMITIDs.

InterruptionCtr Format: 5()

Class: ECV

The number of transitions to UNSCHEDULED DOWNTIME from PRODUCTIVE.

LastPowerdown Format: 20

Class: SV

Timestamp estimate of when the last powerdown or reset occurred. Uses format defined for Clock.

LimitVariable Format: 5()

Class: DVVAL

This variable contains the VID for the variable whose value changed monitoring zones.

MaxSpoolTransmit Format: 54

Class: ECV

The maximum number of messages which the equipment will transmit from the spool in response to an S6,F23 "Transmit Spooled Messages" request. If MaxSpoolTransmit is set to zero, no limit is placed on the messages sent from the spool. Multi-block inquire/grant messages are not counted in this total.

NSTime Format: 5()

Class: ECV

Accumulation of time in NON-SCHEDULED TIME, reported in minutes.

OperatorCommand Format: 5()

Class: DVVAL

This data variable is valid in the event the operator issues a command to the equipment. The codes for this variable are equipment-dependent.

OverWriteSpool Format: 11

Class: ECV

This Equipment Constant is used to indicate to the equipment either to overwrite data in the spool area or to stop spooling whenever the spool area limits are exceeded.

```
= TRUE to overwrite spooled data
= FALSE to stop spooling when limits exceeded
```

PowerdownTime Format: 20

Class: SV

This timestamp is periodically updated based on an interval set by the user. It is used to determine the approximate time that the equipment went down in the event of a power loss. Uses format defined for CLOCK.

PowerupState Format: 20

Class: SV

Specifies the powerup ARAMS state when powerdown occurs during manufacturing time. Single text digit: "2" = STANDBY, "5" = UNSCHEDULED DOWNTIME.

PPChangeName Format: 10, 20

Class: DVVAL

The PPID which was affected upon the event of the creation, editing, or deletion of a Process Program local to the equipment. If the PPID Data Item is also defined and implemented for the equipment, then the values for PPChangeName are subject to the same format restrictions defined for the PPID Data Item.

PPChangeStatus Format: 51

Class: DVVAL

The action taken on the Process Program named in PPChangeName. This variable is valid upon the event of the creation, editing, or deletion of a Process Program local to the equipment.

```
1    =    Credited
2    =    Edited
3    =    Deleted
4-63 Reserved
```

PPError Format: 20

Class: SV or DVVAL

Contains information about a failure to verify a text process program.

PPExecName Format: 0, 10, 20

Class: SV

The PPID(s) of the currently selected Process Program(s). The selection of a new Process Program updates this variable. If multiple Process Programs can be selected, then this variable is a list of PPIDs. If the PPID Data Item is also defined and implemented for the equipment, then the values for PPExecName are subject to the same format restrictions defined for the PPID Data Item.

PPFormat Format: 51

Class: SV

Indicates the type or types of process programs and recipes that are supported.

```
1    =    Unformatted process programs
2    =    Formatted process programs
3    =    Both unformatted and formatted process programs
4    =    Execution Recipes
>4 Reserved
```

PrdRecovery Format: 11

Class: ECV

A boolean value that enables (TRUE) or disables (FALSE) the equipment-initiated return to PRODUCTIVE from UNSCHEDULED DOWNTIME.

PrdState Format: 20

Class: SV

Default ARAMS Substate Code for automated transitions to PRODUCTIVE.

PrdTime Format: 5()

Class: ECV

Accumulation of time in PRODUCTIVE, reported in minutes.

PrevARAMSState Format: 20

Class: SV

The ARAMS code corresponding to the previous state/substate. Four characters.

PreviousProcessState Format: 51

Class: SV

The previous processing state of the equipment, before the most recent process state change.

```
0-63 Reserved
```

ProcessState Format: 51

Class: SV

The current processing state of the equipment.

```
0-63 Reserved
```

RcpChangeName Format: 20

Class: DVVAL

The identifier of the recipe affected upon the event of the creation, editing, or deletion of a recipe.

RcpChangeStatus Format: 51

Class: DVVAL

The type of change that occurred for the recipe indicated in RcpChangeName

```
0   =   No change
1   =   Created
2   =   Updated (modified)
3   =   Stored (new)
4   =   Replaced
5   =   Deleted
6   =   Copied
7   =   Renamed
8,9 Reserved
>10 Reserved
```

RcpExecName Format: 0, 20

Class: SV

The identifier, or a list of identifiers, of currently selected recipes. A zero-length item or list indicates no recipes are currently selected.

SbyRecovery Format: 11

Class: ECV

A boolean value that enables (TRUE) or disables (FALSE) the equipment-initiated return to STANDBY from UNSCHEDULED DOWNTIME.

SbyTime Format: 5()

Class: ECV

Accumulation of time in STANDBY, reported in minutes.

SDTime Format: 5()

Class: ECV

Accumulation of time in SCHEDULED DOWNTIME, reported in minutes.

SpoolCountActual Format: 5()

Class: SV

Used to keep a count of the messages actually contained in the equipment's spool area. Multi-block inquire/grant messages are not spooled and not included in this count.

SpoolCountTotal Format: 5()

Class: SV

Used to keep a count of the total number of primary messages directed to the spool, regardless of whether placed or retained in the spool. Multi-block inquire/grant messages are not spooled and not included in this count.

SpoolFullTime Format: 20

Class: SV

Contains the timestamp from the time the spool last became full. If the spool was not filled during the last spooling period, this will contain a time value prior to the current SpoolStartTime. Uses the same format as the CLOCK variable data item.

SpoolStartTime Format: 20

Class: SV

Contains the timestamp from the time spooling last became active. Uses the same format as the CLOCK variable data item.

SymptomID Format: 5()

Class: SV

A numeric code representing the symptom that initiated the user-initiated state change. A value of zero indicates "no symptom".

SymptomText Format: 20

Class: SV

Text describing the SymptomID. 0-80 characters.

TimeFormat Format: 5()

Class: ECV

NOTE 6: The setting of this ECV controls whether the equipment shall send the Data Items STIME and TIME in 12 or 16-byte format.

```
0   =   12-byte format
1   =   16-byte format
>1 Reserved
```

TransitionType Format: 10

Class: DVVAL

Used with the Limits Monitoring capability, it defines the direction of the zone transition which has occurred.

```
0   =   Transition from lower to upper zone.
1   =   Transition from upper to lower zone.
```

TRATID Format: 5()

Class: DVVAL

Contains the TRATOMICID of the atomic transfer referenced by the event.

TRJOBIDENT Format: 5()

Class: DVVAL

Contains the TRJOBID for the transfer job referenced by the event.

TRJOBNM Format: 20

Class: DVVAL

Contains the TRJOBNAME for the transfer job referenced by the event.

TRLNK Format: 5()

Class: DVVAL

Contains the TRLNK value for the atomic transfer referenced by the event.

UDTime Format: 5()

Class: ECV

Accumulation of time in UNSCHEDULED DOWNTIME, reported in minutes.

6.8 *Object Dictionary* — This section defines the public attributes of objects which are available through SECS-II messages.

The attributes of an object are defined in a table for each object in the following form:

Agent Attribute Definition

Attribute Name	*Definition*	*Access*	*Format*	*Value*
"ObjType"	Agent object type.	RO	20	"Agent"
"ObjID"	The agent's name, assigned by an *authorized user*.	RO	20	

Agent-Specific Dataset Object Attribute Definition

Attribute Name	*Definition*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"MRcpASDS"
"ObjID"	The object's identifier. Contains the value in AgentSpec Agent.	RO	20	
"AgentSpec_Agent"	The name of the *executing agent* to which the other attributes in the *dataset* apply. *Mandatory*.	RO	20	
"AgentSpec_AttrLength"	The length of the *agent-specific* attributes, in bytes. *Mandatory*.	RO	54	
"AgentSpec_ChgTime"	Timestamp of when an *agent-specific* attribute was last changed. *Mandatory*.	RO	20	
"AgentSpec_Comments"	Comments specific to the *agent* entered by the author.	RW	20	Maximum length is 80 characters.
"AgentSpec_LinkParam"	A list of *variable parameter definitions* modified from the list in LinkParam. Valid only for a *linked main* recipe. *Parameter name* and form may not be changed.	RO	00	List of structures composed of parameter name, value, and restrictions.
"Certified"	The certification level for the specific *agent*, assigned by an *authorized user*. Reset when AgentSpec LinkParam is modified. Required for *certification* support.	RW	52	
"AgentSpec_UD_"	Non-standard attribute defined by the supplier or user. Asterisk indicates the part of the attribute name that is provided in this definition. Must be preserved exactly except by the defining entity.	RO	10, 11 20, 3() 4(), 5()	Text form is limited to 80 characters.

Carrier Attribute Definition

Attribute Name	*Definition*	*Access*	*Format*	*Value*
"ObjType"	Object type.	RO	20	A[7] = "Carrier"
"ObjID"	Object identifier for Carrier. Same as CarrierID.	RO	20	A[1-80] (Conforms to restrictions of ObjID as specified in SEMI E39.1, Section 6.)
"Capacity"	Maximum number of substrates in carrier.	RO	51	Positive integer. Examples: 1, 13, and 25.
"CarrierIDStatus"	Current state of the carrier ID verification.	RO	51	Enumerated.
"ContentMap"	Ordered list of substrate identifiers, corresponding to slot 1, 2, …,n.	RO	0	L,n 1. <MID_1> ... n. <MID_n>
"CarrierProcessingStatus"	The current processing state of the material contained in the carrier.	RO	51	Enumerated.
"LocationID"	Identifier of current location.	RO	20	A[1-80] (Conforms to restrictions of ObjID as specified in SEMI E39.1, Section 6.)
"SlotMap"	List of slot statuses as read by	RO	0	List of 1 to n where n is less than or equal to

Attribute Name	Definition	Access	Format	Value
	the equipment.			the value of "Capacity": L,n 1. <Slot Status $_1$> . . n. <Slot Status $_n$> where "Slot Status" is enumerated as: 0 = UNDEFINED 1 = EMPTY 2 = NOT EMPTY 3 = NORMAL 4 = DOUBLE SLOTTED 5 = CROSS SLOTTED 6–128 Reserved.
"SlotMapStatus"	Current state of slot map verification.	RO	51	Enumerated.
"Usage"	The type of substrates contained in the carrier. (i.e., TEST, DUMMY, PRODUCT, FILLER, etc.).	RO	20	

Collection Event Object

Attribute Name	Description	Access	Format	Related Data Items	Value
"ObjType"	Collection Event Object type	RO	20	-	"COLLEVENT"
"ObjID"	Collection Event Identifiers	RO	20	CEID	-
"Enabled"	Boolean true means reporting is enabled for a specific CEID.	RW	11	CEED	-
"EventSource"	Object specifier for object which generates the event for a specific CEID.	RO	20	EVNTSRC	-
"DataReportList"	List of Report Identifiers linked to a specific CEID.	RO	20	-	(list of) RPTID

Data Report Object

Attribute Name	Description	Access	Format	Related Data Items	Value
"ObjType"	Data Report Object Type	RO	20	-	"DATARPT"
"ObjID"	Object identifier for a data report	RO	20	RPTID	-
"DataSource"	Source for the variable data, not writable for predefined provider reports	RO	20	DATASRC	-
"AttrList"	Returns the attribute (or variable) names that this report is requesting from the Data source.	RW	*	-	(list of) VID

Data Source Object

Attribute Name	Description	Access	Format	Related Data Items	Value
"ObjType"	Data Source Object Type	RO	20	-	"DataSource"
"ObjID"	Identifier of a specific Data Source Object	RO	20	DATASRC	-
"AttrList"	Name of attributes for a specific Data Source Object	RO	*	-	(list of) VID

Distributed Recipe Namespace Attribute Definition

Attribute Name	*Description*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"RNSD"
"ObjID"	Text.	RO	20	
"LockedRecipes"	A list of *identifiers* of all recipes with existing *change request records.*	RO	20	
"Recorder"	The *recorder specifier* of the attached *distributed recipe namespace recorder*.	RO	20	
"Segments"	A list of *specifiers* of the *distributed recipe namespace segments* attached to the *namespace.*	RO	20	

Distributed Recipe Namespace Manager Attribute Definition

Attribute Name	*Description*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"RNS_MgrD"
"ObjID"	The *manager's* name.	RO	20	

Distributed Recipe Namespace Recorder Attribute Definition

Attribute Name	*Description*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"RNSDRecorder"
"ObjID"	Text.	RO	20	
"LockedRecipes"	List of *identifiers* of recipes with existing *change request records.*	RO	00	
"Namespace"	Identifies the *namespace* to which the recorder is attached. May be set by the manager.	RO	20	
"NamespaceManager"	Identifies the *distributed recipe namespace manager*. May be set by the manager.	RO	20	
"Segments"	List of *specifiers* of currently attached segments.	RO	00	

Distributed Recipe Namespace Segment Attribute Definition

Attribute Name	*Description*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"RNSDSegment"
"ObjID"	The object name (identifier).	RO	20	
"Namespace"	The name (ObjID) of the *namespace* to which the *segment* belongs. May be set by the manager.	RO	20	
"NamespaceManager"	Identifies the *distributed recipe namespace manager*. May be set by the manager.	RO	20	
"RecipeReadOnlyLevel"	Used to track the corresponding attribute of the *namespace* to which the *segment* belongs. May be set by the manager.	RO	52	

Exception Attributes

Attribute Name	*Description*	*Access*	*Format*	*Related Data Items*	*Value*
"ObjType"	The object type	RO	20	-	"EXCEPTION"
"ObjID"	The identifier of a specific Exception	RO	20	EXID	-
"EXType"	Identifies the type of exception	RO	20	-	Select from set:

					“ALARM” “ERROR”
“EXMessage”	Text message describing the abnormal situation monitored.	RO	20	-	Max. length of 80 characters
“EXEnabled”	Indicates that reporting to the decision authority on the exception condition is enabled.	RW	11	-	Boolean; TRUE is enabled.
“EXRecoveryAction”	List of possible recovery actions (EXRecovery).	RO	20	-	list of text
“EXState”	Current state of an Exception Object. The Exception Object state is usually defined as a combination of substates and concurrent states.	RO	20	-	Composed from the set: “CLEARED” “SET” “NOTPOSTED” “POSTED” “NOTRECOVERING” “RECOVERING” “ABORTINGRECOVERY”

Execution Recipe Attribute Definition

Attribute Name	*Definition*	*Acc.*	*Rqmt*	*Form*	*Default Value*
Identification Attributes					
“ObjType”	The object type.	RO	Y	Text: “ERcp”	“ERcp”
“ObjID”	An identifier derived from Namespace, Class, Name, and Version.	RO	Y	Formatted text.	-
“Namespace”	The name of the *originating namespace*.	RO	Y	Text.	-
“Name”	A logical name assigned by the user when the recipe is *created*.	RO	Y	Text.	-
“Class”	The recipe’s class (e.g., “/PROCESS/” OR “/PROCESS/LOADER/”).	RO	Y	Formatted text: “CLASS/CLASS/.. /CLASS/”	-
“Version”	The version of the recipe.	RO	Y	Text.	-
Mandatory Attributes					
“ExecAttrLength”	The *length attribute* for the attributes of the *execution recipe*. Calculated when the recipe is *downloaded* and whenever an attribute changes. *Mandatory*.	RO	Y	Unsigned integer	-
“ExecChgTime”	The *timestamp* of a change to the attributes of the *execution recipe*. *Mandatory*.	RO	Y	Formatted text, *timestamp* format	-
“AttrLength”	Preserved. *Mandatory*.	RO	Y	Unsigned integer	-
“AttrChgTime”	Preserved. *Mandatory*.	RO	Y	Timestamp format	-
“EditTime”	Preserved unless recipe is modified. *Timestamp* of when the *body* was *created* or modified. *Mandatory*.	RO	Y	Formatted text *Timestamp* format	-
“BodyLength”	Preserved unless recipe is modified. Length of the recipe’s body, in bytes. *Mandatory*.	RO	Y	Unsigned integer	-
“BodyFormat”	Indicates the form and format of the recipe’s *body*.	RO	Y	Enumerated unsigned integer: 0 = *source*, 1 = *object*, > 1 reserved.	0
“Verified”	Indicates whether the recipe’s body is syntactically correct.	RO	Y	Boolean.	FALSE

Attribute Name	*Definition*	*Acc.*	*Rqmt*	*Form*	*Default Value*
"Linked"	Indicates whether the recipe is *linked*.	RO	Y	Boolean.	FALSE
"ChangedBody"	Set to TRUE if the recipe body has changed without a subsequent upload to the originating namespace. Note: this attribute is never updated to a namespace. Required only if recipe can be changed or created.	RO	Y	Boolean.	FALSE
"ExecChgCtl"	Preserved. Specifies change control requirements for recipe.	RO	Y	Binary. Bitwise: 1 - may change 2 - change notification required 4 - recipe may be selected after change, 8 - most recent parameter settings shall be saved. Any combination of these four bits is allowed.	0
Optional Attributes					
"AgentSpec_ Comments"	Copied from the original *agent-specific* attribute when downloaded. Set by the user.	RO	N	Text. Maximum length is 80 characters.	-
"ApprovalLevel"	Indicates the level of approval assigned by an *authorized* user.	RO	N	Unsigned integer	0
"Certified"	Preserved from the *agent-specific* attribute as downloaded. May be used as control for production-worthy recipes.	RO	N	Unsigned integer	0
"Comments"	User comments. Preserved from the *generic* attribute as downloaded.	RO	N	Text. Maximum length is 80 characters.	-
"EditedBy"	Preserved unless recipe is modified. The name of the person or *executing agent* who last modified the recipe.	RO	N	Text. Maximum length is 40 characters.	-
"EstRunTime"	The nominal or estimated execution (run) time of the recipe, in seconds. Used for scheduling purposes. Preserved from the *generic* attribute as downloaded.	RO	N	Unsigned integer	0
"ExecLinkParam"	Preserved unless last value is changed (Section 6.7.4). Contains the list of *parameter definitions* including any *agent-specific* modifications. Required for *variable parameter* support.	RO	N	Structure composed of parameter name, initial value, and restrictions.	NULL
"LinkList"	Preserved. A complete list of recipe *specifiers* for a *linked recipe set*. Required for multi-part recipe support.	RO	N	List of formatted text.	NULL
"SrcRcpID"	For a derived *object form* recipe, contains the recipe *identifier* of the original *source form* recipe. Required only for *derived object form* recipes.	RO	N	Formatted text.	NULL
"VerificationID"	Identification code used by the *verifier* of the recipe. May be used to determine out-of-date formats that need to be *re-verified*.	RO	N	Text. Maximum length is 40 characters.	NULL
Non-Standard Attributes					
AgentSpec_UD_*	Preserved from the original *agent-specific* attributed as downloaded.	RO	N	Defined by supplier or user. Text limited to 80 characters.	-

Attribute Name	*Definition*	*Acc.*	*Rqmt*	*Form*	*Default Value*
UD_*	Non-standard attribute defined by supplier or *user*. Asterisk indicates the part of the attribute name that is provided in this definition. Shall be preserved exactly except by the entity that defined it.	RO	N	Varies with definition. Text form is limited to 80 characters.	-

Managed Recipe Attribute Definition

Attribute Name	*Description*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"MRcp"
"ObjID"	An identifier derived from Class, Name, and Version. No part of a recipe's identifier shall be changed except through *renaming*.	RO	20	
(other)	Description of the information contained.	RO or RW	Varies with definition.	Varies with definition.
"Name"	A logical name assigned by the user when the recipe is *created* or *renamed*.	RO	20	
"Class"	The recipe's class (e.g., "/PROCESS/" or "/PROCESS/LOADER/").	RO	20	Formatted text: "CLASS/CLASS/../CLASS/"
"Version"	The version of the recipe.	RO	20	
"AttrLength"	The total length of the *generic* attributes, in bytes. *Mandatory*.	RO	5()	
"AttrChgTime"	Timestamp of the last change to a *generic* attribute. *Mandatory*.	RO	20	
"BodyLength"	Length of the recipe's body, in bytes. *Mandatory*.	RO	5()	
"EditTime"	Timestamp of when the *body* was *created* or last *updated*. *Mandatory*.	RO	20	*Timestamp* format: "YYYYMMDDhhmmsscc"
"BodyFormat"	Indicates the form and format of the recipe's *body*. *Default* is zero.	RO	52	0 = source, 1 = object, >1 reserved.
"Verified"	Indicates whether the recipe's body is syntactically correct. Reset when the recipe is *created* or *updated*. Default is FALSE.	RO	11	
"Linked"	Indicates whether the recipe is linked. Reset when the recipe is *originated*, *verified*, or *unlinked*. Default is FALSE.	RO	11	
"ApprovalLevel"	Indicates the level of approval assigned by an *authorized user*. *Default* is zero. Reset when the recipe is *originated* or *linked*. For a *linked* recipe, may not be higher than any of its *subrecipes*.	RW	52	
"Comments"	User comments.	RW	20	Maximum length is 80 characters.
"EditedBy"	The name of the person who last edited the recipe.	RO	20	Maximum length is 40 characters.
"EstRunTime"	The nominal or estimated execution (run) time of the recipe, in seconds. Reset when the recipe, is *created* or *updated*. Set when the recipe is *verified*. May be recalculated to total time for a *main* recipe when *linked*. Used for scheduling purposes. Algorithm for calculation shall be documented. Default is 0.	RW	54	

Attribute Name	*Description*	*Access*	*Format*	*Value*
"ExecChgCtrl"	Specifies change control requirements for recipe. Default is 0. Combinations of bits are used to indicate multiple permissions.	RW	10	Binary. Bitwise (MSB = 8): 1 - the recipe body may be changed 2 - change notification required 4 - recipe may be selected after change, 8 - most recent parameter settings shall be saved. Any combination of these four bits is allowed
"ExtRef"	A list of all recipe *specifiers* as referenced within the recipe. Explicit *versions* not required. Reset when the recipe is *created*, *updated*, and *verified*.	RO	00	List of items of format 20.
"LinkList"	A complete list of recipe *specifiers* found in the ExtRef attribute of a *main* recipe and all of its *subrecipe*s, with duplicates removed and all *versions* explicitly determined. Set for the *main* recipe when linked. Reset when the recipe is *originated* or *verified*. Required for multi-part recipe support.	RO	00	List of items of format 20.
"LinkParam"	A list of all variable parameter definitions contained in the Parameters attribute of a *main* recipe and all of its *subrecipes*, with duplicates removed. Reset when the recipe is *created*, *updated*, or *verifie*d. Set when the recipe is *linked*. Required for *variable parameter* support.	RO	00	List of parameter definition structures composed of parameter name, initial value, and restrictions.
"Parameters"	A list of variable parameter definitions contained in the recipe. Reset when the recipe is *created*, *updated*, and *verified*. Set when the recipe is *verified*. Required only for *variable parameter* support.	RO	00	List of parameter definition structures composed of parameter name, initial value, and restrictions.
"SrcRecID"	*Identifier* of the *source form* recipe from which a *derived object form* recipe is derived. Value determined by the *verifier* of the recipe. Required only for support of *derived object form* recipes.	RO	20	
"VerificationID"	Identification code set by the *verifier* of the recipe. May be used to determine out-of-date formats that need to be *re-verified*.	RO	20	Maximum length is 40 characters.
"UD_"	Non-standard attribute defined by supplier or *user*. Asterisk indicates the part of the attribute name that is provided in this definition. Shall be preserved exactly except by the entity that defined it.	RO	10, 11, 20, 3(), 4(), 5()	Text form is limited to a maximum of 80 characters.

Process Job Attributes

Attribute Name	*Description*	*Access*	*Format*	*Related Data Item*	*Value*
"ObjType"	Name of the Object Type	RO	20	-	"PROCESSJOB"
"ObjID"	Identifier of a Process Job	RO	20	PRJOBID	-
"PRMt1Type"	Type of material being processed	RO	20	-	allowed values: "css" "wfr"
"PRMt1NameList"	Process Material Name, identifies material being processed by a job, which could be more than one item.	RO	20	-	(list of) Text

Attribute Name	*Description*	*Access*	*Format*	*Related Data Item*	*Value*
"RecID"	Object Specifier of Recipe used by a Process Job, see SEMI E39 and SEMI E42	RO	20	-	-
"PRRecipeMethod"	Indicates any special handling for a Process Job's Recipe	RO	20	-	allowed values: "STANDARD" "USETUNING"
"PRJobState"	Indicates the current state of a Process Job. The state of a job may be a combination of sub-states and concurrent states.	RO	20	-	Composed from the set: "WAITINGFOR JOB" "JOBQUEUED" "JOB CANCELLED" "JOBACTIVE" "SETUP" "WAITINGFORSTART" "PROCESSING" "NOTPAUSED" "PAUSING" "PAUSED" "NOTSTOPPING" "STOPPING" "NOTABORTING" "ABORTING" "PROCESSCOMPLETE" "JOBCOMPLETE"
"PRProcessStart"	Processing should start automatically after Job is defined when this Boolean is set TRUE	RO	11	-	Boolean

Recipe Executor Attribute Definition

Attribute Name	*Description*	*Access*	*Rqmt*	*Form*
"ObjType"	The object type	RO	Y	Text = "RcpExec"
"ObjID"	Text	RO	Y	Text
"DefaultNamespace"	The name of an *executing agent's name-space* used for all hardware-dependent and other *agent-specific* recipes.	RW	Y	Text
"ProdApprove"	The minimum value of a recipe's *approval level* accepted during productive and standby states. Required for SEMI E10 support only.	RW	N	Unsigned integer
"ProdCertify"	The minimum value of a recipe's *certification level* accepted during productive and standby states. Required for SEMI E10 support only.	RW	N	Unsigned integer
"RunCycleUnit"	The process unit on which the calculation of the estimated value of the recipe *generic attribute* EstRunTime is based.	RO	N	Case-sensitive formatted text composed of a unit of measure and an optional numeric suffix. Compliant with SEMI E5, Section 9.
"RecipeSelectID"	A list of recipe *identifiers* for the currently selected recipes.	RO	Y	List of formatted text.
"RecipeSelect-Parameters"	A list of all *parameter definitions* in effect for the ith recipe *identifier* in RecipeSelectID. The maximum value for i is determined by the equipment supplier as the maximum number of recipes which may be *selected* at the same time. Required if variable parameters are supported.	RO	N	List of structures composed of parameter name, parameter value, parameter restriction.

Recipe Namespace Attribute Definition

Attribute Name	*Definition*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"RNS"
"ObjID"	The *name* of the *namespace*.	RO	20	A name of "Default" is prohibited.
"RecipeReadOnlyLevel"	The level of *approval* at which recipes are *read-only*.	RW	52	
"Members"	The *names* of *agents* capable of *verifying* and *executing* the recipes in the *namespace*.	RW	00	List of items of format 20.

Recipe Namespace Manager Attribute Definition

Attribute Name	*Definition*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"RNS_Mgr"
"ObjID"	The *manager's* name.	RO	20	
"NamespaceName"	The *name* of the *namespace* managed.	RO	20	

Table Attribute Definition

Attribute Name	*Definition*	*Access*	*Format*	*Value*
"ObjType"	The object type.	RO	20	"Table"
"ObjID"	The object's identifier.	RO	20	1-80 characters.
"NumCols"	Number of columns.	RO	5()	Non-zero.
"NumRows"	Number of rows.	RO	5()	Non-zero.
"TableLength"	Total number of bytes required to store the table elements, exclusive of any formatting required for storage.	RO	5()	Non-zero.

Trace Object

Attribute Name	*Description*	*Access*	*Format*	*Related Data Items*	*Value*
"ObjType"	Trace Report Object type	RO	20	-	"TRACE"
"ObjID"	Identifier of a specific Trace Report	RO	20	TRID	-
"Enabled"	Boolean true means the specific Trace Report is enabled.	RW	*	CEED	-
"ReportID"	List or report linked to this Trace Report	RO	20	-	(list of) RPTID
"SamplePeriod"	Time between report samples given in floating point seconds.	RW	4 ()	TRSPER	-
"TotalSamples"	The maximum number of samples that this Trace Report will perform.	RW	*	TOTSMP	-
"GroupSize"	Number of trace reports to be grouped before a report is sent.	RW	*	REPGSZ	-
"StartEventID"	Identifier of the event which starts trace reporting.	RW	20	CEID	-
"StartEvtSrcSpec"	Source for the start event	RW	20	EVNTSRC	-
"StopEventID"	Identifier of the event which stops trace reporting.	RW	20	CEID	-
"StopEvtSrcSpec"	Source for the stop event	RW	20	EVNTSRC	-
"AutoDelete"	Boolean true means this report is deleted when reporting is complete.	RW	11	TRAUTOD	-
"ReportChangeOnly"	Boolean, if true, then trace reports are sent only if at least one of the reported variables changes.	RW	11	RPTOC	-

attribute name — A reserved text string, of at most 40 characters, that is unique for that object.

description — A description of the attribute.

access — Indicates whether the attribute may be set through messages. Access is either read-only (RO) or read-write (RW).

format — Indicates the type of data (format code).

timestamp format — Text form indicating date and time in the format "YYYYMMDDhhmmsscc".

related data items — Indicates an explicit relationship with a corresponding data item.

value — Specifies any restrictions on the possible values. Examples of restrictions include exclusion of zero for format 5(), a maximum length for text, a format imposed on text, an order imposed on a list, or an enumerated set of valid values.

Requirements:

- The attributes "ObjType" and "ObjID" are required for all object definitions and shall use format 20.
- The attribute "ObjType" shall be assigned a fixed value for each object.
- The value of "ObjID" may not be changed by using SetAttr (S14F3).

The value of "ObJType" may be used for messages using the data item OBJTYPE. The value of the attribute "ObjID" may be used for messages using the data item OBJID.

The name of a public attribute may be used for messages using the data item ATTRID. The value of a public attribute may be used for messages using the data item ATTRDATA.

Variable data items defined in Section 6.7 may be regarded as attributes of the object type "Equipment", where SVs and DVVALs are RO and ECVs are RW.

6.9 With the use of Harel[4] state diagrams to describe the behavior of objects, an object's state must be describable as a combination of a set of sub-states and concurrent states. The rules for describing the state of an object are: (1) use the comma (',') to deliminate concurrent states, (2) use the foreslash ('/') to delimit a super-state and sub-state, (3) to deliver the set of lowest level concurrent states, and (4) optionally omit super-state names when there are no ambiguities in the names of the lowest level states.

Please refer to Figure 1 in order to follow the discussions for the notations. In Harel notation, 'pump' and 'vacuum' are concurrent states. The text to specify this relation in a response to a request for state is 'pump, vacuum'. The comma can be read as meaning 'and'. 'on' and 'off' are sub-states of 'pump'. 'vent', 'rough', and 'Hi-V' are sub-states of 'vacuum'. The sub-state syntax is 'state/sub-state' where the '/' can be read as 'is in sub-state'. So using the example in Figure 1, if the pump is off and the vacuum is vented, then the text message which conveys this is 'pump/off, vacuum/vent'. This message can be shortened to 'off,vent' because there is no ambiguity in doing so.

4 Harel D. "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming, 8, 1987, pp. 231-274. Elsevier Science, P.O. Box 945, New York, NY 10159-0945, http://www.elvesier.nl/homepage/browse.htt

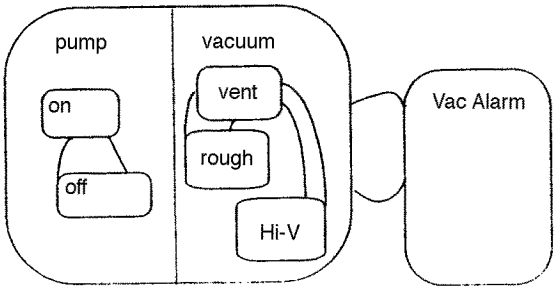


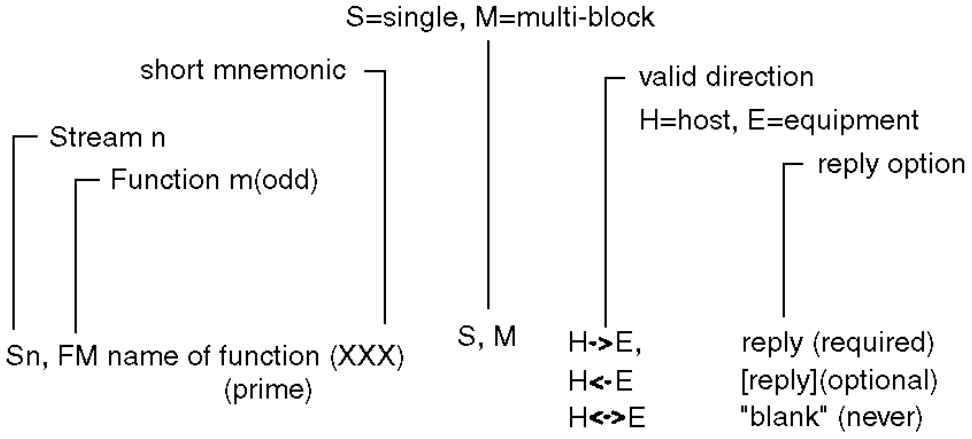
Figure 3
State Chart Example

7 Message Detail

7.1 *Intent* — This section defines a number of specific functions in different streams which can be used as a basis for communication between host and equipment. The functions are defined in the form of transaction message pairs according to the transaction level requirements specified in Section 5.

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

7.2.1 The abbreviations used in each transaction are as follows:



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

Structure:
                Detailed structure showing lists and defined items. Lists are
                denoted by a capital L followed by the length separated by a comma.
                The individual elements in the list are numbered on separate lines.
                Nested lists are indented to emphasize the structure. The detailed
                form of the items is given in the define section at the beginning
                of the transaction. The symbols "<" and ">" are used to enclose
                each item in the structure data and imply that there is an item
                header. A detailed description of each data item as well as a list of
                the allowable data formats can be found in the Data Item Dictionary.
Exception:
                Special cases in the structure that have a different meaning.
```

Sn,Fm+1 Name of function (secondary) (same structure as above except never with reply)

7.3 *Message Usage* — This section discusses message features and where they may be used.

7.3.1 *Zero Length Items and Lists* — Certain message definitions may use zero length data items and zero length lists as a technique to convey specific information to the receiver of the message. For commands (i.e., "Do Something") and requests (i.e., "Return Some Data"), it may be used to mean "Use default values for the data item(s) which were not included". The default may be a specific value or a value chosen by the equipment.

7.3.1.1 For messages reporting data (either responses to requests or asynchronous reports), the technique may be used to indicate that the desired information is not available or not applicable. In some cases, the fact that data is unavailable may indicate success or failure of a command.

7.3.1.2 Certain message definitions may define a zero length data item or a zero length list to mean "the information is not supplied." The receiving party should react to this lack of information as it deems appropriate.

7.3.2 *Compliance to Message Definitions* — Any given standard SECS-II message shall comply to the format shown in the Message Definition for that Stream and Function. Specifically:

1. The message shall contain all Lists and Data Items shown as required in the Message Definition.
2. The message shall not contain any Lists or Data Items not shown in the Message Definition, unless the Message Definition specifically allows this.
3. The message shall not contain any List Item or Data Item with zero length unless the Message Definition specifically defines a meaning for such a zero length item.

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

7.4.1 Function 0 exists in all streams and has the same special meaning in each stream. A function 0 message closes a transaction, so that the originator will not have to wait for a transaction timeout to proceed. Function 0 is sent in lieu of the expected secondary message when the interpreter cannot, because of a transmission error or some other reason, respond with the expected reply. It is not a requirement that the interpreter send function 0 to close a transaction.

7.5 *Stream 1 Equipment Status* — This stream provides a means for exchanging information about the status of the equipment, including its current mode, depletion of various consumable items, and the status of transfer operations.

S1,F0 Abort Transaction (S1F0) S,H<->E

```
Description:    Used in lieu of an expected reply to abort a transaction. Function 0
                is defined in every stream and has the same meaning in every stream.

Structure:      Header only
```

S1,F1 Are You There Request (R) S,H<->E,reply

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

Structure: Header only

S1,F2 On Line Data (D) S,H<->E

Description: Data signifying that the equipment is alive.

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

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

S1,F3 Selected Equipment Status Request (SSR) S,H->E,reply

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

Structure: The following structure is approved for all item formats and should be used by all new implementations:

L,n
1. <$SVID_1$>
.
.
n. <$SVID_n$>

The following structure is included for compatibility with previous implementations and may only be used for items of format 3() and 5():

<$SVID_1$,...,$SVID_n$>

Exception: A zero-length list (structure 1) or item (structure 2) means report all SVIDs.

S1,F4 Selected Equipment Status Data (SSD) M,H<-E

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

Structure: L,n
1. <SV_1>
.
.
n. <SV_n>

Exceptions: A zero-length list item for SV_i means that $SVID_i$ does not exist.

S1,F5 Formatted Status Request (FSR) S,H->E,reply

Description: A request for the equipment to report the status according to a pre-defined fixed format.

Structure: <SFCD>

S1,F6 Formatted Status Data (FSD) M,H<-E

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

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

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

S1,F7 Fixed Form Request (FFR) S,H->E,reply

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

Structure: <SFCD>

S1,F8 Fixed Form Data (FFD) M,H<-E

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

Structure: Depends upon the form being specified.

Exception: A zero-length item means the form is unavailable.

S1,F9 Material Transfer Status Request (TSR) S,H->E,reply

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

Structure: Header only

S1,F10 Material Transfer Status Data (TSD) M,H<-E

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

Structure: L,2
1. <$TSIP_1$,...,$TSIP_n$>
2. <$TSOP_1$,...,$TSOP_n$>

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

S1,F11 Status Variable Namelist Request (SVNR) S,H->E,reply

Description: A request to the equipment to identify certain status variables.

Structure:
```
L,n
  1. <SVID1>
  .
  .
  n. <SVIDn>
```

Exception: A zero length means report all SVIDs.

S1,F12 Status Variable Namelist Reply (SVNRR) M,H<-E

Description: The equipment reports to the host the name and units of the requested SVs.

Structure:
```
L,n
  1. L,3
       1. <SVID1>
       2. <SVNAME1>
       3. <UNITS1>
  2. L,3
  .
  .
  n. L,3
       1. <SVIDn>
       2. <SVNAMEn>
       3. <UNITSn>
```

Exceptions: Zero-length ASCII items for both $SVNAME_i$ and $UNITS_i$ indicates that the SVID does not exist.

S1,F13 Establish Communications Request (CR) S,H<->E,reply

Description: The purpose of this message is to provide a formal means of initializing communications at a logical application level both on power-up and following a break in communications. It should be the following any period where host and Equipment SECS applications are unable to communicate. An attempt to send an Establish Communications Request (S1,F13) should be repeated at programmable intervals until an Establish Communications Acknowledge(S1,F14) is received within the transaction timeout period with an acknowledgement code accepting the establishment.

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

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

S1,F14 Establish Communications Request Acknowledge (CRA) S,H<->E

Description: Accept or deny Establish Communications Request (S1,F13). MDLN and SOFTREV are on-line data and are valid only if COMMACK = 0.

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

Exception: The host sends a zero-length list for item 2 to the equipment.

S1,F15 Request OFF-LINE (ROFL) S,H->E,reply

Description: The host requests that the equipment transition to the OFF-LINE state.

Structure: Header only

S1,F16 OFF-LINE Acknowledge (OFLA) S,H<-E

Description: Acknowledge or error

Structure: <OFLACK>.

S1,F17 Request ON-LINE (RONL) S,H->E,reply

Description: The host requests that the equipment transition to the ON-LINE state.

Structure: Header only

S1,F18 ON-LINE Acknowledge (ONLA) S,H<-E

Description: Acknowledge or error

Structure: <ONLACK>.

S1,F19 Get Attribute (GA) S,H<—>E,reply[5]

Description: Request for attribute data relating to the specified object or entity within the equipment.

Structure:

```
L,3
  1.<OBJTYPE>
  2.L,m          [m=number of objects for which attributes requested]
      1.<OBJID1>
      .
      .
      m.<OBJIDm>
  3.L,n          [n=number of attributes requested for each object]
      1.<ATTRID1>
      .
      .
      n.<ATTRIDn>
```

Exception: A zero-length list (m=0) is a request for attributes of all objects of the specified type.
A zero-length list (n=0) is a request for all attributes of the object(s)to be returned in a predefined order.

5 Material Movement Management used only the Host to Equipment direction for this message. However, both directions are included for future compatibility with Recipe Management and other future services.

S1,F20 Attribute Data (AD) M,H<—>E

Description: This message is used to transfer the requested set of object attributes. The order of requested objects and attributes is retained from the primary message.

Structure:
```
L,2
  1. L,m        [m=number of objects for which data is sent]
       1.L,n      [n= number of attributes returned for OBJID1]
            1.<ATTRDATA1>
            .
            .
            n.<ATTRDATAn>
       .
       .
       m.L,n      [n= number of attributes returned for OBJIDm]
            1.<ATTRDATA1>
            .
            .
            n.<ATTRDATAn>
  2. L,p        [p=# errors reported]
       1.L,2
            1.<ERRCODE1>
            2.<ERRTEXT1>
            .
            .
       p.L,2
            1.<ERRCODEp>
            2.<ERRTEXTp>
```

Exception: If m=0, it indicates that the specified OBJTYPE is unknown. If any n=0, it indicates that the corresponding object was not found. If any ATTRDATA item is reported as a zero-length item, it indicates that the specified attribute does not exist. If no errors were found, p=0.

7.6 *Stream 2 Equipment Control and Diagnostics* — Messages which deal with control of the equipment from the host. This includes all remote operations and equipment self-diagnostics and calibration but specifically excludes the control operations which are associated with material transfer (see Stream 4), loading of executive and boot programs (Stream 8), and all file and operating system calls (Streams 10, 13). See also continuations in Stream 17.

S2,F0 Abort Transaction (S2F0) S,H<->E

Description: Same form as S1,F0

S2,F1 Service Program Load Inquire (SPI) S,H<->E,reply

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

Structure:
```
L,2
  1. <SPID>
  2. <LENGTH>
```

S2,F2 Service Program Load Grant (SPG) S,H<->E

Description: Provides permission to load

Structure: <GRANT>

S2,F3 Service Program Send (SPS) M,H<->E, reply

Description: The data associated with the S2,F1 inquire is sent. If S2,F3 is multi-block, it must be preceded by the S2,F1/S2,F2 Inquire/Grant transaction.

Structure: <SPD>

S2,F4 Service Program Send Acknowledge (SPA) S,H<->E

Description: Acknowledge or error

Structure: <SPAACK>

S2,F5 Service Program Load Request (SPR) S,H<->E,reply

Description: A service program is requested.

Structure: <SPID>

S2,F6 Service Program Load Data (SPD) M,H<->E

Description: A service program is sent.

Structure: <SPD>

Exception: A zero-length item means that the requested program cannot be returned.

S2,F7 Service Program Run Send (CSS) S,H->E,reply

Description: Start the requested program

Structure: <SPID>

S2,F8 Service Program Run Acknowledge (CSA) S,H<-E

Description: Acknowledge or error

Structure: <CSAACK>

S2,F9 Service Program Results Request (SRR) S,H->E,reply

Description: Ask for results of service program

Structure: <SPID>

S2,F10 Service Program Results Data (SRD) M,H<-E

Description: Get the results back

Structure: <SPR>

Exception: A zero-length item means SPR does not exist.

S2,F11 Service Program Directory Request (SDR) S,H<->E,reply

Description: There may be more than one service program.

Structure: Header only

S2,F12 Service Program Directory Data (SDD) S,H<->E

Description: A list of service program names.

Structure: L,n
1. <$SPID_1$>
.
.
n. <$SPID_n$>

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

S2,F13 Equipment Constant Request (ECR) S,H->E,reply

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

Structure: The following structure is approved for all item formats and should be used by all new implementations:

L,n
1. <$ECID_1$>
.
.
n. <$ECID_n$>

The following structure is included for compatibility with previous implementations and may only be used for items of format 3() and 5():
<$ECID_1$, . . . , $ECID_n$>

Exception: A zero-length list (structure1) or item (structure2) means report all ECV's according to a predefined order.

S2,F14 Equipment Constant Data (ECD) M,H<-E

Description: Data Response to S2,F13 in the order requested.

Structure:
L,n
1. <ECV_1>
2. <ECV_2>
.
.
n. <ECV_n>

Exceptions: A zero-length list item for ECV_i means that $ECID_i$ does not exist. The list format for this data item is not allowed, except in this case.

S2,F15 New Equipment Constant Send (ECS) S,H->E,reply

Description: Change one or more equipment constants.

Structure:
L,n
1. L,2
   1. <$ECID_1$>
   2. <ECV_1>
2. L,2
.
.
n. L,2
   1. <$ECID_n$>
   2. <ECV_n>

S2,F16 New Equipment Constant Acknowledge (ECA) S,H<-E

Description: Acknowledge or error If EAC contains a non-zero error code, the equipment should not change any of the ECIDs specified in S2F15.

Structure: <EAC>

S2,F17 Date and Time Request (DTR) S,H<->E,reply

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

Structure: Header only

S2,F18 Date and Time Data (DTD) S,H<->E

Description: Actual time data

Structure: <TIME>

Exception: A zero-length item means no time exists.

S2,F19 Reset/Initialize Send (RIS) S,H->E,reply

Description: Causes equipment to reach one of several predetermined initialized conditions.

Structure: <RIC>

S2,F20 Reset Acknowledge (RIA) S,H<-E

Description: Acknowledge or error

Structure: <RAC>

S2,F21 Remote Command Send (RCS) S,H->E,[reply]

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

Structure: <RCMD>

S2,F22 Remote Command Acknowledge (RCA) S,H<-E

Description: Acknowledge or error

Structure: <CMDA>

S2,F23 Trace Initialize Send (TIS) M,H->E,reply

Description: Status variables exist at all times. This function provides a way to sample a subset of those status variables as a function of time. The trace data is returned on S6,F1 and is related to the original request by the TRID Multiple trace requests may be made to that equipment allowing it. If equipment receives S2,F23 with the same TRID as a trace function that is currently in progress, the equipment should terminate the old trace and then initiate the new trace. A trace function currently in progress may be terminated by S2,F23 with TRID of that trace and TOTSMP=0.

If S2,F23 is multi-block, it must be preceded by the S2,F39/S2,F40 Inquire/Grant transaction. Some equipment may support only single-Block S6,F1, and may refuse a S2,F23 message which would cause a multi-block S6,F1.

Each equipment shall document its trace performance limits. The Host Computer shall not send an S2,F23 which exceeds the equipment's performance limits, or the equipment may operate incorrectly.

Structure: The following structure is approved for all item formats and should be used by all new implementations:

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

The following structure is included for compatibility with previous implementations and may only be used for items whose SVID is format 3() and 5():

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

S2,F24 Trace Initialize Acknowledge (TIA) S,H<-E

Description: Acknowledge or error

Structure: <TIAACK>

S2,F25 Loopback Diagnostic Request (LDR) S,H<->E,reply

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

Structure: <ABS>

S2,F26 Loopback Diagnostic Data (LDD) S,H<->E

Description: The echoed binary string

Structure: <ABS>

S2,F27 Initiate Processing Request (IPR) S,H->E,reply

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

Structure:
L,3
1. <LOC>
2. <PPID>
3. L,n
1. <MID_1>
.
.
n. <MID_n>

Exception: A zero-length PPID indicates no process program is being specified and the equipment is to take whatever action is appropriate for it to determine the proper program to use. A zero-length MID list indicates no MID is to be associated with the material to be processed.

S2,F28 Initiate Processing Acknowledge (IPA) S,H<-E

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

Structure: <CMDA>

S2,F29 Equipment Constant Namelist Request (ECNR) S,H->E,reply

Description: This function allows the host to retrieve basic information about what equipment constants are available in the equipment.

Structure:
L,n
1. <$ECID_1$>
.
.
n. <$ECID_n$>

Exception: A zero-length list means send information for all ECIDs.

S2,F30 Equipment Constant Namelist (ECN) M,H<-E

Description: Data Response

Structure: L,n (number of equipment constants)
1. L,6
    1. <$ECID_1$>
    2. <$ECNAME_1$>
    3. <$ECMIN_1$>
    4. <$ECMAX_1$>
    5. <$ECDEF_1$>
    6. <$UNITS_1$>
2. L,6
.
.
n. L,6
    1. <$ECID_n$>
    2. <$ECNAME_n$>
    3. <$ECMIN_n$>
    4. <$ECMAX_n$>
    5. <$ECDEF_n$>
    6. <$UNITS_n$>

Exceptions: Zero-length ASCII items for $ECNAME_i$, $ECMIN_i$, $ECMAX_i$, $ECDEF_i$, and $UNITS_i$ indicates that the ECID does not exist.

S2,F31 Date and Time Set Request (DTS) S,H->E,reply

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

Structure: <TIME>

S2,F32 Date and Time Set Acknowledge (DTA) S,H<-E

Description: Acknowledge the receipt of time and date.

Structure: <TIACK>

S2,F33 Define Report (DR) M,H->E,reply

Description: The purpose of this message is for the host to define a group of reports for the equipment.

The type of report to be transmitted is designated by a Boolean "Equipment Constant." An "Equipment Constant Value" of "False" means that an "Event Report"(S6,F11) will be sent, and a value of "True" means that an "Annotated Event Report"(S6,F13) will be sent.
If S2,F33 is Multi-block, it must be preceded by the S2,F39/S2,F40 Inquire/Grant transaction.

Structure:

```
L,2
  1. <DATAID>
  2. L,a                                        # reports
       1. L,2                                     report 1
            1. <RPTID1>
            2. L,b                          #VIDs this report
                 1. <VID1>
                 .
                 .
                 b.<VIDb>
       a. L,2                                     report a
            1. <RPTIDa>
            2. L,c                          #VIDs this report
                 1. <VID1>
                 .
                 .
                 c. <VIDc>
```

Exceptions:
1. A list of zero-length following <DATAID> deletes all report definitions and associated links. See S2,F35 (Link Event/Report).
2. A list of zero-length following <RPTID> deletes report type RPTID. All CEID links to this RPTID are also deleted.

S2,F34 Define Report Acknowledge (DRA) S,H<-E

Description: Acknowledge or error If an error condition is detected the entire message is rejected (i.e., partial changes are not allowed).

Structure: <DRACK>

S2,F35 Link Event Report (LER) M,H->E,reply

Description: The purpose of this message is for the host to link n reports to an event (CEID). These linked event reports will default to 'disabled' upon linking. That is, the occurrence of an event would not cause the report to be sent until enabled. See S2,F37 for enabling reports.

If S2,F35 is Multi-block, it must be preceded by the S2,F39/S2,F40 Inquire/Grant transaction.

Structure:

```
L,2
  1. <DATAID>
  2. L,a                                          # events
       1. L,2                                       event 1
            1. <CEID_1>
            2. L,b
                 1. <RPTID_1>
                 .
                 .
                 b. <RPTID_b>
       .
       .
       a. L,2                                       event a
            1. <CEID_a>                  # RPTIDS this event
            2.L,c
                1.<RPTID_1>
                .
                .
                c. <RPTID_c>
```

Exception: A list of zero length following CEID deletes all report links to that event.

S2,F36 Link Event Report Acknowledge (LERA) S,H<-E

Description: Acknowledge or error If an error condition is detected the entire message is rejected (i.e., partial changes are not allowed).

Structure: <LRACK>

S2,F37 Enable/Disable Event Report (EDER) S,H- >E,reply

Description: The purpose of this message is for the host to enable or disable reporting for a group of events (CEIDs).

Structure:

```
L,2
  1. <CEED>                                       enable/disable
  2. L,n   #CEIDs
       1. <CEID_1>
       .
       .
       n. <CEID_n>
```

Exception: A list of zero length following <CEED> means all CEIDs.

S2,F38 Enable/Disable Event Report Acknowledge (EERA) S,H<-E

Description: Acknowledge or error If an error condition is detected the entire message is rejected, i.e., partial changes are not allowed.

Structure: <ERACK>

S2,F39 Multi-block Inquire (DMBI) S,H->E,reply

Description: If a S2,F23 S2,F33, S2,F35, S2,F45, or S2,F49 message is more than one block, this transaction must precede the message.

Structure: L,2
1. <DATAID>
2. <DATALENGTH>

S2,F40 Multi-block Grant (DMBG) S,H<-E

Description: Grant permission to send multi-block message.

Structure: <GRANT>

S2,F41 Host Command Send (HCS) S,H->E,reply

Description: The Host requests the Equipment perform the specified remote command with the associated parameters.

Structure: L,2
1. <RCMD>
2. L,n # of parameters
   1. L,2
      1. $<CPNAME_1>$ parameter 1 name
      2. $<CPVAL_1>$ parameter 1 value
   .
   .
   n. L,2
      1. $<CPNAME_n>$parameter n name
      2. $<CPVAL_n>$parameter n value

S2,F42 Host Command Acknowledge (HCA) S,H<-E

Description: Acknowledge Host command or error. If command is not accepted due to one or more invalid parameters (i.e., HCACK=3), then a list of invalid parameters will be returned containing the parameter name and reason for being invalid.

Structure:
```
L,2
  1. <HCACK>
  2. L,n # of parameters
     1. L,2
          1. <CPNAME_1> parameter 1 name
          2. <CPACK_1> parameter 1 reason
     .
     .
     n. L,2
          1. <CPNAME_n>parameter n name
          2. <CPACK_n>parameter n reason
```

Exception: If there are no invalid parameters, then a list of zero length will be sent for item 2.

S2,F43 Reset Spooling Streams and Functions (RSSF) S,H->E,reply

Description: This message allows the host to select specific streams and functions to be spooled whenever spooling is active.

Structure:
```
L,m
  1. L,2
       1.<STRID_1>
       2. L,n
            1. <FCNID_1>
            .
            .
            n. <FCNID_n>
       .
       .
       m. L,2
            1. <STRID_m>
            2. L,n
               1. <FCNID_1>
               .
               .
               . <FCNID_n>
```

Exceptions:
1. A zero-length list, m=0, turns off spooling for all streams and functions.
2. A zero-length list, n=0, turns on spooling for all functions for the associated stream.

Notes:
1. Turning off spooling for all functions for a specific stream is achieved by omitting reference to the stream from this message.
2. Spooling for Stream 1 is not allowed.
3. Equipment must allow host to spool all primary messages for a stream (except Stream 1).
4. A defined list of functions for a stream in this message will replace any previously selected functions.

S2,F44 Reset Spooling Acknowledge (RSA) M,H<-E

Description: Acknowledge or error

Structure:

```
L,2
  1. <RSPACK>                        (accept or reject)
  2. L,m                             (m = number of streams with errors)
       1. L,3
            1. <STRID_1>
            2. <STRACK_1>            (error in stream)
            3. L,n                   (n = number of functions in error)
                 1. <FCNID_1>
                 .
                 .
                 n. <FCNID_n>
       .
       .
       m. L,3
            1. <STRID_m>
            2. <STRACK_m>            (error in stream)
            3. L,n                   (n = number of functions in error)
                 1. <FCNID_1>
                 .
                 .
                 n. <FCNID_n>
```

Exceptions:

1. If RSPACK=0, a zero-length list, m=0, is given, indicating no streams or functions in error.
2. A zero-length list, n=0, indicates no functions in error for specified stream.

S2,F45 Define Variable Limit Attributes (DVLA) M,H->E,reply

Structure:

```
L,2
1. <DATAID>
2. L,m (m=# of variables in this definition)
     1. L,2
          1. <VID1>
          2. L,n (n=# of limits being defined/changed for VID1)
               1. L,2
                    1. <LIMITID1>
                    2. L,p (p={0,2})
                         1. <UPPERDB1>
                         2. <LOWERDB1>
                         .
                         .
                         n. L,2
                              1. <LIMITIDn>
                              2. L,p (p={0,2})
                                   1. <UPPERDBn>
                                   2. <LOWERDBn>
.
.
m.L,2
    1. <VIDm>
    2. L,n (n=# of limits being defined/changed for VIDm)
         1. L,2
              1. <LIMITID1>
              2. L,p (p={0,2})
                   1. <UPPERDB1>
                   2. <LOWERDB1>
              .
              .
              n. L,2
                   1. <LIMITIDn>
                   2. L,p (p={0,2})
                        1. <UPPERDBn>
                        2. <LOWERDBn>
```

Exceptions:

1. A zero-length list, m=0, sets all limit values for all monitored VIDs to "undefined."
2. A zero-length list, n=0, sets all limits values for that VID to "undefined."
3. A zero-length list, p=0, sets that limit to "undefined."

S2,F46 Variable Limit Attribute Acknowledge (VLAA) M,H<-E

Description: Acknowledge definition of variable limit attributes or report error. If DVLA is not accepted due to one or more invalid parameters (e.g., LIMITACK=3), then a list of invalid parameters is returned containing the variable limit attribute and reason for rejection. If an error condition is detected, the entire message is rejected (i.e., partial changes are not allowed).

Structure:

```
L,2
  1. <VLAACK>
  2. L,m                              (m=number of invalid parameters)
       1. L,3
            1. <VID1>                 (VID with error)
            2.<LVACKp>                (reason)
            3.L,n {n=0,2}
                1. <LIMITID1>         (1st limit in error for VIDp)
                2. <LIMITACK1>        (reason)
       .
       .
       m. L,3
            1. <VIDm>                 (VID with error)
            2. <LVACKm>               (reason)
            3. L,n {n=0,2}
                 1. <LIMITID1>        (1st limit in error for VIDx)
                 2. <LIMITACK1>       (reason)
```

Exceptions:
1. A zero-length list, m=0 indicates no invalid variable limit attributes.
2. A zero-length list, n=0 indicates no invalid limit values for that VID.

S2,F47 Variable Limit Attribute Request (VLAR) S,H->E,reply

Description: This message allows the host to query the equipment for current variable limit attribute definitions.

Structure:

```
L,m                                   (m=# of VIDs this request)
  1. <VID1>
  .
  .
  m. <VIDm>
```

Exception: A zero-length list, m=0, requests a list of all VID values that can have variable limit attributes.

S2,F48 Variable Limit Attributes Send (VLAS) M,H<-E

Description: Equipment sends values of requested variable limit attribute definitions in the order requested.

Structure:

```
L,m                                  (m=# of VIDs this request)
  1.L,2
      1. <VID1>
      2. L,p {p=0,4}
           1. <UNITS1>
           2. <LIMITMIN1>
           3. <LIMITMAX1>
           4. L,n                    (n=# of limits defined for this VID)
                1. L,3
                     1. <LIMITID1>
                     2. <UPPERDB1>
                     3. <LOWERDB1>
                .
                .
                n. L,3
                     1.<LIMITIDn>
                     2.<UPPERDBn>
                     3.<LOWERDBn>
  .
  .
  m.L,2
      1.<VIDm>
      2. L,p {p=0,4}
           1.<UNITSm>
           2.<LIMITMINm>
           3.<LIMITMAXm>
           4. L,n                    (n=# of limits defined for this VID)
                1. L,3
                     1.<LIMITID1>
                     2.<UPPERDB1>
                     3.<LOWERDB1>
                .
                .
                n. L,3
                     1.<LIMITIDn>
                     2.<UPPERDBn>
                     3.<LOWERDBn>
```

Exceptions:

1. A zero-length list, p=0, indicates that limits are not supported for the VID.
2. A zero-length list, n=0, means no limits are currently defined for the specified variable.

Description: The host requests an object to perform the specified remote command with its associated parameters. If multi-block, it shall be preceded by the S2,F39/S2,F40 Multi-Block Inquire/Grant transaction.

Structure:

```
L,4
  1. <DATAID>
  2. <OBJSPEC>
  3. <RCMD>
  4. L,m # of parameter groups
       1. L,2
            1. <CPNAME1> command parameter 1 name
            2. <CEPVAL1> command-enhanced parameter 1 value
       2. L,2
            1. <CPNAME2> command parameter 2 name
            2. <CEPVAL2> command-enhanced parameter 2 value
       .
       .
       .
       .
       m. L,2
            1. <CPNAMEm> command parameter m name
            2. <CEPVALm> command enhanced parameter m value
```

If a specific value of CPNAME is defined to have a CEPVAL defined as a LIST, it shall always be a LIST. If the CEPVAL that is associated to that specific value of CPNAME is defined to be anything other than LIST, it will result in a format error.

Exception: A zero length list, m = 0, indicates that no parameter groups are sent with the command. OBJSPEC can be a null length item.

Notes: 1. If CEPVAL is a LIST, the items that make up that list shall take on one of the following forms: (1) a list of items with an identical format, (2) a LIST of CPNAME, CEPVAL pairs, as illustrated below.

```
A) L,2
     1. <CPNAMEa>
     2. L,m
          1. <CPVALa1>
          2. <CPVALa2>
          .
          .
          .
          .
          .
          .
          m. <CPVALam>
```

```
B) L,2
     1. <CPNAMEb>
     2. L,n
          1. L,2
               1. <CPNAMEb1>
               2. <CEPVALb1>
          .
          .
          n. L,2
               1. <CPNAMEbn>
               2. <CEPVALbn>
```

S2,F50 Enhanced Remote Command Acknowledge M,H<-E

Description: The equipment acknowledges Enhanced Remote Command or reports any error(s). If the command is not accepted due to one or more invalid parameters, (i.e. HCACK = 3), then a list of invalid parameters will be returned containing the parameter name and reason for being invalid.

Structure:
```
L,2
  1. <HCACK>
  2. L,n            # of parameter groups
       1. L,2
            1. <CPNAME1>
            2. <CEPACK1>
       .
       .
       n. L,2
            1. <CPNAMEn>
            2. <CEPACKn>
```

7.7 *Stream 3 Materials Status* — The functions of the material status stream are used to communicate information and actions related to material, including carriers and material-in-process, time-to-completion information, and extraordinary material occurrences.

S3,F0 Abort Transaction (S3F0) S,H<->E

Description: Same form as S1,F0.

S3,F1 Material Status Request (MSR) S,H->E,reply

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

Structure: Header only

S3,F2 Material Status Data (MSD) M,H<-E

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

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

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

S3,F3 Time to Completion Request (TCR) S,H->E,reply

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

Structure: Header only

S3,F4 Time to Completion Data (TCD) M,H<-E

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

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

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

S3,F5 Material Found Send (MFS) S,H<-E,[reply]

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

Structure:
```
L,2
  1. <MF>
  2. <QUA>
```

S3,F6 Material Found Acknowledge (MFA) S,H->E

Description: Acknowledge or error

Structure: <ACKC3>

S3,F7 Material Lost Send (MLS) S,H<-E,[reply]

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

Structure:
```
L,3
  1. <MF>
  2. <QUA>
  3. <MID>
```

S3,F8 Material Lost Acknowledge (MLA) S,H->E

Description: Acknowledge or error

Structure: <ACKC3>

S3,F9 Material ID Equate Send (IES) S,H<-E,reply

Description: Provide an alternative name to be used as equivalent to the original material ID.

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

S3,F10 Material ID Equate Acknowledge (IEA) S,H->E

Description: Acknowledge or error

Structure: <ACKC3>

S3,F11 Material ID Request (MIDR) S,H<-E,reply

Description: The equipment requests the Material ID of the material at the specified port.

Structure: <PTN>

S3,F12 Material ID Request Acknowledge (MIRA) S,H->E

Description: The host acknowledges the request for the Material ID. If the use of a request/acknowledge/send/acknowledge conversation is required, it indicated by the acknowledge code MIDRA=2. In this case, the send/ acknowledge transaction is S3,F13, S3,Fl4. A timeout when electing S3,F13 is indicated by S9,F13 or a restart of the conversation, with S3,F11.

Structure: L,3
1. <PTN>
2. <MIDRA>
3. <MID>

Note: For all cases except MIDRA=0 (accepted, <MID> follows), the <MID> will be ignored by the receiver of message S3,F12. When MIDRA=0, a zero-length MID indicates that no MID is available.

S3,F13 Material ID Send (MIS) S,H->E,reply

Description: The host sends the Material ID of the material at the specified port.

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

Note: A zero-length MID indicates that no MID is available.

S3,F14 Material ID Acknowledge (MIA) S,H<-E

Description: Acknowledge or error

Structure: <MIDAC>

S3,F15 Materials Multi-Block Inquire (MMBI) S,H->E,reply

Description: This message requests permission to send a multi-block message based upon a maximum length of the total message. It must be sent prior to sending any multi-block primary message in Stream 3.

Structure:
```
L,2
  1.  <DATAID>
  2.  <DATALENGTH>
```

S3,F16 Materials Multi-Block Grant (MMBG) S,H< - E

Description: This message grants or denies permission to send a multi-block primary message in Stream 3.

Structure:
```
<GRANT>
```

S3,F17 Carrier Action Request M,H->E,reply

Description: This message requests an action to be performed for a specified carrier. If multi-block, this message must be preceded by the S3,F11/F12 transaction.

Structure:
```
L,5
  1.  <DATAID>
  2.  <CARRIERACTION>
  3.  <CARRIERSPEC>
  4.  <PTN>
  5.  L,n                          n = number of carrier attributes
        1.  L,2
              1.  <CATTRID1>
              2.  <CATTRDATA1>
        .
        .
        n.  L,2
              1.  <CATTRIDn>
              2.  <CATTRDATAn>
```

Exception: If n = 0, then no carrier attributes are included. If CARRIERSPEC is not a zero-length item, then PTN may be omitted (a zero-length item).

S3,F18 Carrier Action Acknowledge S,H<-E

Description: This message acknowledges the carrier action request.

Structure:
```
L,2
  1.  <CAACK>
  2.  L,n
        1.  L,2
              1.  <ERRCODE1>
              2.  <ERRTEXT1>
        .
        .
        n.  L,2
              1.  <ERRCODEn>
              2.  <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S3,F19 Cancel All Carrier Out Request S,H->E, reply

Description: This message is used to cancel all pending carrier out requests.

Structure: Header only.

S3,F20 Cancel All Carrier Out Acknowledge S,H<-E

Description: This message acknowledges the Cancel Carrier Out request.

Structure:
```
L,2
  1.  <CAACK>
  2.  L,n
        1.  L,2
              1.  <ERRCODE1>
              2.  <ERRTEXT1>
            .
            .
            n.  L,2
                  1.  <ERRCODEn>
                  2.  <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S3,F21 Port Group Definition S,H->E, reply

Description: This message defines the port in a port group and provides the initial port access.

Structure:
```
L,3
  1.  <PORTGRPNAME>
  2.  <PORTACCESS>
  3.  L,n
        1.  <PTN1>
        .
        .
        n.  <PTNn>
```

S3,F22 Port Group Definition Acknowledge S,H<-E

Description: This message acknowledges the port group definition.

Structure:
```
L,2
  1.  <CAACK>
  2.  L,n
        1.  L,2
              1.  <ERRCODE1>
              2.  <ERRTEXT1>
            .
            .
            n.  L,2
                  1.  <ERRCODEn>
                  2.  <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S3,F23 Port Group Action Request S,H->E, reply

Description: This message requests an action be performed for a port group. The access mode may be changed or the port group may be deleted.

Structure:
```
L,3
  1.  <PGRPACTION>
  2.  <PORTGRPNAME>
  3.  L,m
        1.  L,2
              1.  <PARAMNAME1>
              2.  <PARAMVAL1>
        .
        .
        m.  L,2
              3.  <PARAMNAME1>
              4.  <PARAMVAL1>
```

Exception: If m = 0, then no parameters are provided.

S3,F24 Port Group Action Acknowledge S,H<-E

Description: This message acknowledges the port group action.

Structure:
```
L,2
  1.  <CAACK>
  2.  L,n
        1.  L,2
              1.  <ERRCODE1>
              2.  <ERRTEXT1>
        .
        .
        n.  L,2
              1.  <ERRCODEn>
              2.  <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S3,F25 Port Action Request S,H->E, reply

Description: This message requests an action be performed for a port.

Structure:
```
L,2
  1. <PORTACTION>
  2. <PTN>
  4.  L,m
        2.  L,2
              5.  <PARAMNAME1>
              6.  <PARAMVAL1>
        .
        .
        m.  L,2
              7.  <PARAMNAME1>
              8.  <PARAMVAL1>
```

Exception: If m = 0, then no parameters are provided.

S3,F26 Port Action Acknowledge S,H<-E

```
Description:   This message acknowledges the port action request.

Structure:     L,2
                 1. <CAACK>
                 2. L,n
                      1. L,2
                           3. <ERRCODE1>
                           4. <ERRTEXT1>
                      .
                      .
                      n. L,2
                           1. <ERRCODEn>
                           2. <ERRTEXTn>

Exception:     If n = 0, no errors exist.
```

7.8 *Stream 4 Material Control* — The material control stream contains the original material control protocol and the newer protocol which supports SEMI E32.

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

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

7.8.1.2 Figure S4.1 shows six possible handshake situations between the sender and the receiver. There are two normal handshakes. Figure S4.1(a) shows the normal three-message exchange when material is passed between equipment. The host is informed of a complete transfer of material. Figure S4.1(b) shows an alternative message exchange where the sender changes its mind and decides not to send the material. Figures S4.1(c) and (d) show two situations where the material gets stuck during the transfer. In each situation an error message is issued to the host from the equipment where the material is stuck. The other equipment terminates normally. When material is stuck, manual intervention is required to move the material towards the equipment which indicates the stuck condition. The manual intervention has two possible outcomes. One, the material can be moved to a position where the handshake can resume or, two, the material is broken or lost from the transfer. Lost material causes a lost material error message to be sent to the host prior to resuming the operation. The specific details of recovering front stuck material are equipment-dependent. The stuck material condition is determined by the amount of time the material transfer mechanism is turned on. The sender claims stuck material if the material is not clear of its sensor before a time t1. The receiver claims stuck material if the material is not received before time t2. Figures S4.1(e) and (f) show the possible error conditions in the unlikely event that for some reason a handshake message is lost. Figure S4.1(e) shows that time t3 is the longest that the sender will wait for material received message. Times t2 and t3 set an upper limit on the amount of time either material transport mechanism will operate.

7.8.1.3 Figure S4.2 summarizes the interaction of the timers, handshake messages, and the error messages in the form of a flow chart. It also identifies specific states for the sender and the receiver. These states are referred to in the messages.

The ranges of timer values are as follows:

tl — time to leave sender

tl+10<= t2<= 60sec.---time to receive

t2+10<= t3<= 70sec.---time to complete send

Default values, tl= 10 sec., t2= 60 sec., t3= 70 sec.

NOTE 7: t1, t2, t3 defined for Stream 4 are not to be confused with timeouts T1, T2, T3, and T4 defined in SEMI E4 (SECS-I).

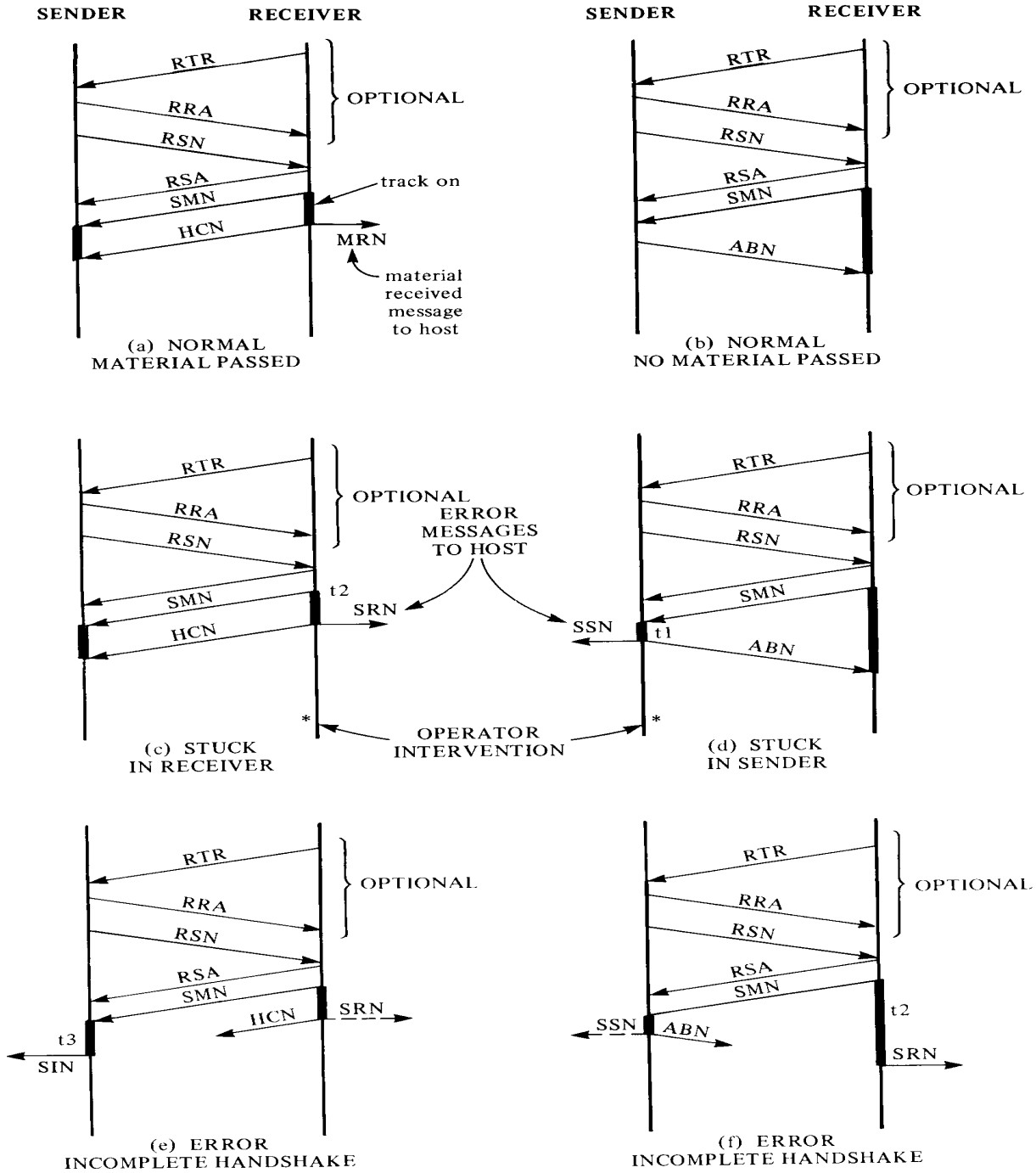


Figure S4.1
The Six Possible Handshakes

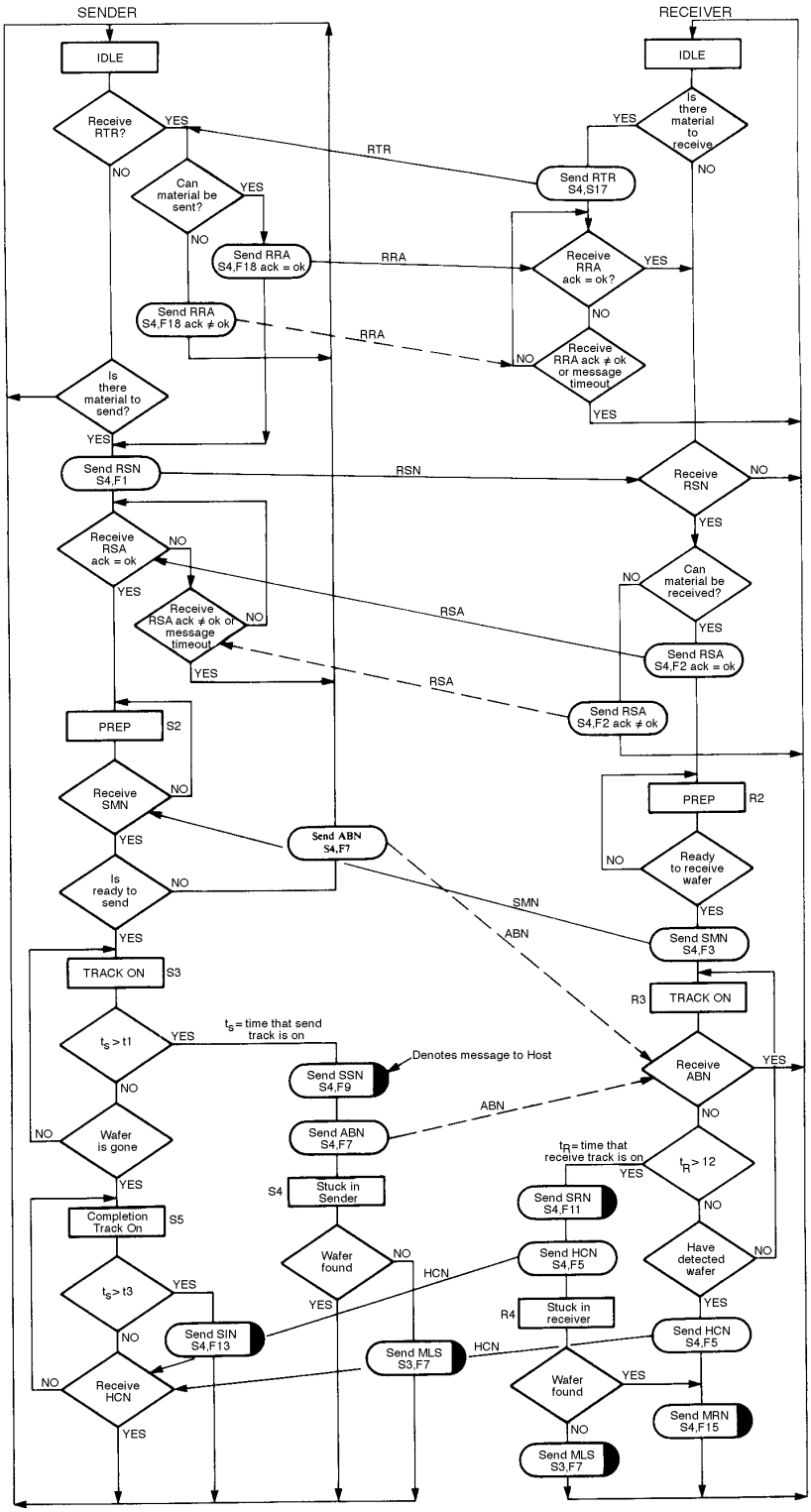


Figure S4.2
Material Control-Handshake Flowchart

S4,F0 Abort Transaction (S4F0) S,H->E

```
Description:   Same form as S1,F0.
```

S4,F1 Ready to Send Materials (RSN) S,H<->E,reply

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

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

S4,F2 Ready to Send Acknowledge (RSA) S,H<->E

```
Description:   Acknowledge or error

Structure:     <RSACK>
```

S4,F3 Send Material (SMN) S,H<->E

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

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

S4,F4 Not Used

S4,F5 Handshake Complete (HCN) S,H<->E

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

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

S4,F6 Not Used

S4,F7 Not Ready to Send (ABN) S,H<->E

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

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

S4,F8 Not Used

S4,F9 Stuck in Sender (SSN) S,H<-E

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

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

S4,F10 Not Used

S4,F11 Stuck in Receiver (SRN) S,H<-E

Description: An error from the receiver to the host. The time between Send Material (SMN) and detection of the material at the receiver exceeds the receiver's t2 timeout. The receiver goes to a hold state until the disposition of material is determined.

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

S4,F12 Not Used

S4,F13 Send Incomplete Timeout (SIN) S,H<-E

Description: An error from the sender to the host. The time between the receipt of the Send Material (SMN) and the receipt of Handshake-Complete (HCN) exceeds the sender's t3 timeout. There has been an error in the handshake and the transfer mechanism is turned off.

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

S4,F14 Not Used

S4,F15 Material Received (MRN) S,H<-E

Description: A message from the receiver to the host. Material has been transferred to the receiver.

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

S4,F16 Not Used

S4,F17 Request to Receive (RTR) S,H<->E,reply

```
Description:   Receiver requests the sender initiate a conversation to send the
               specified material to the specified port.

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

Exceptions:    A zero-length MID means equipment doesn't know MID.
```

S4,F18 Request to Receive Acknowledge (RRA) S,H<->E

```
Description:   Acknowledge or error

Structure:     <RRACK>
```

7.8.2 *Support for Material Movement Management Services* — The following messages were defined to support SEMI E32.

7.8.2.1 *Macro Level Messages* — The following messages support the host supervised macro level of material movement as defined in SEMI E32. Stream 1 Macro Level Messages can be found in Section 7.5: S1F19, Get Attribute (GA); S1F20, Attribute Data (AD).

S4,F19 Transfer Job Create (TJ) M,H—>E,reply

Description: The host requests that the equipment undertake one or more discrete (or atomic) transfers to achieve a host defined objective. The host provides the transfer specifications for each atomic transfer. Atomic transfers on separate ports on the equipment are allowed to execute in parallel. Atomic transfers for a port must be executed sequentially or in some cases concurrently. Both equipment transfer partners for each atomic transfer must receive appropriate Transfer Job Request messages in order to execute a transfer. If S4,F20 is multi-block, it must be preceded by the S4,F25/S4,F26 Inquire/Grant transaction.

Structure:

```
L,2
  1.<DATAID>
  2.L,2
      1.<TRJOBNAME>
      2.L,n                    [n=#atomic xfrs defined for this job]
          1.L,12               [Specification for first atomic xfr]
               1.<TRLINK>      [Atomic transfer identifier]
               2.<TRPORT>      [Port to be used for transfer]
               3.<TROBJNAME>   [Transfer object identifier]
               4.<TROBJTYPE>   [Object type-what form is the material
                               in]
               5.<TRROLE>      [Role in transfer-primary/secondary]
               6.<TRRCP>       [Transfer recipe identifier]
               7.<TRPTNR>      [Identifier of transfer partner]
               8.<TRPTPORT>    [Partner's Port to be Used]
               9.<TRDIR>       [Transfer direction-send or receive]
              10.<TRTYPE>      [Active or Passive]
              11.<TRLOCATION>  [Location to send/receive mtl]
              12.<TRAUTOSTART> [Does eqp await host start command
                               after setup?]
          .
          .
          n.L,12               [Specification for nth atomic xfr]
               1.<TRLINK]
               ↓
              12.<TRAUTOSTART>
```

S4,F20 Transfer Job Acknowledge (TJA) S,H<—E

Description: The equipment informs the host of its acceptance or rejection of the Transfer Job Request.

Structure:
```
L,3
  1.<TRJOBID>
  2. L,m          [m=number of atomic transfers in the transfer job.]
       1.<TRATOMICID1>
       .
       .
       m.<TRATOMICIDm>
  3. L,2
       1.<TRACK>  [Accepted or rejected]
       2. L,n     [n=# errors reported]
            1. L,2
                 1.<ERRCODE1>
                 2.<ERRTEXT1>
            .
            .
            n. L,2
                 1.<ERRCODEn>
                 2.<ERRTEXTn>
```

Exception: A zero-length list (m=0) is sent if the transfer job is rejected. A zero-length list (n=0) is sent if the transfer job is accepted.

S4,F21 Transfer Job Command (TC) S,H—>E,reply

Description: This message is used by the host to modify a current transfer job on an equipment.

Structure:
```
L,3
  1.<TRJOBID>
  2.<TRCMDNAME>              [identifier of the transfer command]
  3.L,n                      [n=number of parameters=0 if none]
      1.L,2
          1.<CPNAME1>        [transfer parameter name]
          2.<CPVAL1>         [transfer parameter value]
      .
      .
      n. L,2
           1.<CPNAMEn>
           2.<CPVALn>
```

S4,F22 Transfer Command Acknowledge (TCA) S,H<—E

Description: Equipment accepts or rejects the transfer command.

Structure:
```
L,2
  1.<TRACK>                  [Accepted or rejected]
  2. L,n                     [n=# errors reported]
       1. L,2
            1.<ERRCODE1>
            2.<ERRTEXT1>
       .
       .
       n. L,2
            1.<ERRCODEn>
            2.<ERRTEXTn>
```

Exception: If the command is accepted, n=0.

S4,F23 Transfer Job Alert (TJA) S,H<—E,[reply]

Description: Equipment informs the host that a transfer job milestone has been reached (e.g., job started or job complete). If complete, all equipment resources originally reserved for the transfer have been released.

Structure:
```
L,4
  1.<TRJOBID>
  2.<TRJOBNAME>
  3.<TRJOBMS>
  4. L,2
       1.<TRACK>             [success or failure]
       2. L,n                [n=# errors reported]
            1. L,2
                 1.<ERRCODE1>
                 2.<ERRTEXT1>
            .
            .
            n. L,2
                      1.<ERRCODEn>
                      2.<ERRTEXTn>
```

Exception: If the transfer job is completed successfully, n=0.

S4,F24 Transfer Alert Acknowledge (TLA) S,H—>E

Description: Acknowledge receipt of the S4,F23 message.

Structure: Header Only

S4,F25 Multi-block Inquire (MB14) S,H—>E,reply

```
Description:   If a Stream 4 host-initiated message is more than a single block in
               length, this transaction must precede the message.

Structure:     L,2
                 1.<DATAID>
                 2.<DATALENGTH>
```

S4,F26 Multi-block Grant (MBG4) S,H<—E

```
Description:   Grant (or deny) permission to send multi-block message.

Structure:     <GRANT>
```

7.8.2.2 *Micro Level Messages* — The following messages support the equipment-to-equipment micro level handoff of material as defined in SEMI E32.

7.8.2.2.1 The messages which support the micro level are passed directly between the equipment. For the purpose of the communication link, one of the equipment must be designated the host and the other the equipment. The choice is up to the implementer. Equipment which are configurable to act as either host or equipment are suggested for ease of installation.

7.8.2.2.2 The two equipment involved in a micro level transfer assume different roles. One equipment is designated the "Primary Transfer Partner," and the other is the "Secondary Transfer Partner" (see SEMI E32 for more definition). While some consistency of roles is expected, this designation is fluid and may change from one transfer to the next. The Primary Transfer Partner has more responsibility and thus initiates messages which the Secondary does not.

7.8.2.2.3 The selection of "Host" and "Equipment" for the communication link is not related to the fluid relationship of Primary and Secondary Transfer Partner. However, it is the designation of Primary or Secondary which determines the originator of certain messages. It is for this reason that the designation P = Primary and S = Secondary Transfer Partner.

Micro Level Messages

S4,F27 Handoff Ready (HR) S,P<—>S

```
Description:   Each transfer partner informs the other when they are ready to
               perform a specified atomic transfer. The TRLINK values from the two
               partners must match. The values contained in the atomic transfer
               specification pertain to the sender of the message (except where
               specified).

Structure:     L,2
                 1.<EQNAME>
                 2.L,11                [Specification for atomic xfr]
                      1.<TRLINK>       [Atomic transfer identifier]
                      2.<TRPORT>       [Port to be used for transfer]
                      3.<TROBJNAME>    [Transfer object identifier]
                      4.<TROBJTYPE>    [Object type-what form the material is in]
                      5.<TRROLE>       [Role in transfer-primary/secondary]
                      7.<TRPTNR>       [Identifier of transfer partner]
                      8.<TRPTPORT>     [Partner's Port to be Used]
                      9.<TRDIR>        [Transfer direction-send or receive]
                     10.<TRTYPE>       [Active or Passive]
                     11.<TRLOCATION>   [Location to send/receive mtl]
```

S4,F28 Not Used

S4,F29 Handoff Command (HC) S,P—>S

Description: Command issued by the primary to the secondary transfer partner to achieve some physical action.

Structure:
```
L,4
  1.<TRLINK>                [Atomic Transfer identifier]
  2.<MCINDEX>               [Identified this specific Micro Cmd request]
  3.<HOCMDNAME>             [Requested Micro Cmd]
  4.L,n                     [n=number of parameters]
      1.L,2
          1.<CPNAME1>       [Micro Cmd parameter name]
          2.<CPVAL1>        [Micro Cmd parameter value]
      .
      .
      n.L,2
          1.<CPNAMEn>
          2.<CPVALn>
```

Exception: n=0 if no parameters are used.

S4,F30 Not Used

S4,F31 Handoff Command Complete (HCC) S,P<—S

Description: Completion status of the micro command. This is sent from the secondary to the primary transfer partner when the command is completed or terminated.

Structure:
```
L,3
  1.<TRLINK>                [Atomic Transfer identifier]
  2.<MCINDEX>               [Links to specific micro command (S4,F31)]
  3.L,2
      1.<HOACK>             [success or failure]
      2.L,n                 [n=# errors reported]
          1.L,2
              1.<ERRCODE1>
              2.<ERRTEXT1>
          .
          .
          n.L,2
              1.<ERRCODEn>
              2.<ERRTEXTn>
```

S4,F32 Not Used

S4,F33 Handoff Verified (HV) P<—>S

Description: Sent by the primary transfer partner to inform the secondary that no more micro commands will be issued for this atomic transfer and to request a verification that the transfer is complete and successful. Also sent by the secondary partner following the receipt of this message to verify that the transfer is complete and successful (or to report problems).

Structure:
```
L,2
  1.<TRLINK>
  2.L,2
      1.<HOACK>                    [success or failure]
      2.L,n                        [n=# errors reported]
          1.L,2
              1.<ERRCODE1>
              2.<ERRTEXT1>
          .
          .
          n.L,2
              1.<ERRCODEn>
              2.<ERRTEXTn>
```

S4,F34 Not Used

S4,F35 Handoff Cancel Ready (HCR) P<—>S

Description: Sent by either transfer partner to cancel a previous Handoff Ready message. This message is valid only before the handoff has begun.

Structure: <TRLINK>

S4,F36 Not Used

S4,F37 Handoff Cancel Ready Acknowledge (HCA) P<—>S

Description: Sent by the receiver of the Handoff Cancel Ready message to accept or deny the cancel. The cancel request is denied if the handoff process has begun.

Structure:
```
L,2
  1.<TRLINK>
  2.<HOCANCELACK>
```

S4,F38 Not Used

S4,F39 Handoff Halt (HH) P<—>S

Description: Sent by either transfer partner to cause all transfer related activity of the other to cease immediately. It is used when the equipment or material is at risk of damage.

Structure: <TRLINK>

S4,F40 Not Used

S4,F41 Handoff Halt Acknowledge (HHA) P<—>S

```
Description:   Sent to equipment's transfer partner following completion of halt
               activities resulting from a previously received S4,F39.

Structure:     L,2
                 1.<TRLINK>
                 2.<HOHALTACK>
```

S4,F42 Not Used

7.9 *Stream 5 Exception Handling* — This stream contains messages regarding binary and analog equipment exceptions. Exceptions are classified into two categories: errors and alarms. Messages S5,F1 through S5,F8 of this section provide basic alarm messages. The messages S5,F9 through S5,F18 provide extended capabilities for Exception Handling. When using messages F1 - F8, alarms may be divided into categories as follows:

1. *personal safety* — Condition may be dangerous to people.
2. *equipment safety* — Condition may harm equipment.
3. *parameter control warning* — Parameter variation outside of preset limits — may harm product.
4. *parameter control error* — Parameter variation outside of reasonable control limits — may indicate an equipment malfunction.
5. *irrecoverable error* — Intervention required before normal use of equipment can resume.
6. *equipment status warning* — An unexpected condition has occurred, but operation can continue.
7. *attention flags* — A signal from a process program indicating that a particular step has been reached.
8. *data integrity* — A condition which may cause loss of data; usually related to Stream 6.

7.9.1 For messages F1 through F8, it will be the equipment's responsibility to categorize the alarm. Some alarm conditions may cause more than one type of alarm to be issued. For example, a parameter control error on over temperature may also trip a protective device that makes the alarm irrecoverable without some intervention.

S5,F0 Abort Transaction (S5F0) S,H<->E

```
Description:   Same form as S1,F0.
```

S5,F1 Alarm Report Send (ARS) S,H<-E,[reply]

```
Description:   This message reports a change in or presence of an alarm condition.
               One message will be issued when the alarm is set and one message will
               be issued when the alarm is cleared. Irrecoverable errors and
               attention flags may not have a corresponding clear message.

Structure:     L,3
                 1. <ALCD>
                 2. <ALID>
                 3. <ALTX>
```

S5,F2 Alarm Report Acknowledge (ARA) S,H->E

Description: Acknowledge or error

Structure: <ACKC5>

S5,F3 Enable/Disable Alarm Send (EAS) S,H->E,[reply]

Description: This message will change the state of the enable bit in the equipment. The enable bit determines if the alarm will be sent to the host. Alarms which are not controllable in this way are unaffected by this message.

Structure: L,2
1. <ALED>
2. <ALID>

Exception: A zero-length item for ALID means all alarms.

S5,F4 Enable/Disable Alarm Acknowledge (EAA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC5>

S5,F5 List Alarms Request (LAR) S,H->E,reply

Description: This message requests the equipment to send binary and analog alarm information to the host.

Structure: $<ALID_1, . . . , ALID_n>$

Exception: A zero-length item means send all possible alarms regardless of the state of ALED.

S5,F6 List Alarm Data (LAD) M,H<-E

Description: This message contains the alarm data known to the equipment. There are "m" alarms in the list.

Structure: L,m
1. L,3
    1. $<ALCD_1>$
    2. $<ALID_1>$
    3. $<ALTX_1>$
2. L,3
.
.
m. L,3
    1. $<ALCD_m>$
    2. $<ALID_m>$
    3. $<ALTX_m>$

Exception: If m=0, no response can be made. A zero-length item returned for $ALCD_i$ or $ALTX_i$ means that value does not exist.

S5,F7 List Enabled Alarm Request (LEAR) S,H->E,reply

Description: List alarms which are enabled.

Structure: Header only

S5,F8 List Enabled Alarm Data (LEAD) M,H<-E

Description: This message is similar to S5,F6 except that it lists only alarms which are enabled.

Structure: Same as S5,F6

S5,F9 Exception Post Notify (EXPN) S,H<-E,[reply]

Description: This message provides the means to inform a host system that an exception condition is 'set'. Optionally, recovery actions for the exception may be sent.

Structure:
L,5
1. <TIMESTAMP>
2. <EXID>
3. <EXTYPE>
4. <EXMESSAGE>
5. L,n
    1. <EXRECVRA$_1$>
    .
    .
    .
    n. <EXRECVRA$_n$>

Exception: A zero-length list (n = 0) shall be sent when there are no possible recovery actions.

Exception: This is a single block message. The text in each of the EXRECVRA data items may need to be restricted in length to meet the single block requirement.

S5,F10 Exception Post Confirm (EXPC) S,H->E

Description: Host confirms receipt of S5,F9 message from the equipment.

Structure: Header only

S5,F11 Exception Clear Notify (EXCN) S,H <-E,[reply]

Description: This message provides the means to inform a host system that an exception/alarm condition is no longer active (set).

Structure:
L,4
1. <TIMESTAMP>
2. <EXID>
3. <EXTYPE>
4. <EXMESSAGE>

Exception: EXMESSAGE can be used to provide the reason that the exception cleared.

S5,F12 Exception Clear Confirm (EXCC) S,H->E

Description: Host confirms receipt of S5,F11 message from the equipment.

Structure: Header only

S5,F13 Exception Recover Request (EXRR) S,H->E,reply

Description: Request that the entity which is experiencing an error execute a recovery action.

Structure:
```
L,2
  1. <EXID>
  2. <EXRECVRA>
```

S5,F14 Exception Recover Acknowledge (EXRA) S,H<-E

Description: The entity indicates a response to the recovery request.

Structure:
```
L,2
  1. <EXID>
  2. L,2
       1. <ACKA>
       2. L,m (m = {0,2})
            1. <ERRCODE>
            2. <ERRTEXT>
```

Exception: The list m can be zero length, if the recovery request was accepted.

S5,F15 Exception Recovery Complete Notify (EXRCN) S,H<-E,[reply]

Description: Allows the service provider to inform the controller/host that the recovery operation completed on a specific exception and an error code if the recovery terminated abnormally.

Structure:
```
L,3
  1. <TIMESTAMP>
  2. <EXID>
  3. L,2
       1. <ACKA>
       2. L,m (m = {0,2})
            1. <ERRCODE>
            2. <ERRTEXT>
```

Exception: This list m can be of zero length if the recovery was successful.

S5,F16 Exception Recovery Complete Confirm (EXRCC) S,H->E

Description: Host confirms receipt of S5,F15 message from the equipment.

Structure: Header only

S5,F17 Exception Recovery Abort Request (EXRAR) S,H->E,reply

Description: Stop the recovery procedure on a specific exception.

Structure:
```
1. <EXID>
```

S5,F18 Exception Recovery Abort Acknowledge (EXRAA) S,H<-E

Description: Indicate the success of the request for Recovery Abort.

Structure:
```
L,2
  1. <EXID>
  2. L,2
       1. <ACKA>
       2. L,m (m = {0,2})
            1. <ERRCODE>
            2. <ERRTEXT>
```

Exception: The list m can be of zero length if the abort was successful.

7.10 *Stream 6 Data Collection* — This stream is intended to cover the needs of in-process measurements and equipment monitoring.

S6,F0 Abort Transaction (S6F0) S,H<->E

Description: Same form as S1,F0.

S6,F1 Trace Data Send (TDS) M,H<-E,[reply]

Description: This function sends samples to the host according to the trace setup done by S2,F23. Trace is a time-driven form of equipment status.

Even if S6,F1 is multi-block, it is not preceded by an Inquire/Grant transaction, because the Host S2,F23 is an implicit grant. Some equipment may support only single-block S6,F1, and may refuse an S2,F23 (Trace Initiate Send) message which would cause a multi-block S6,F1.

Structure:
```
L,4
  1. <TRID>
  2. <SMPLN>
  3. <STIME>
  4. L,n
       1. <SV1>
       2. <SV2>
       .
       .
       n. <SVn>
```

Exception: A zero-length STIME means no value is given and that the time is to be derived from SMPLN along with knowledge of the request.

S6,F2 Trace Data Acknowledge (TDA) S,H->E

Description: Acknowledge or error

Structure: <ACKC6>

S6,F3 Discrete Variable Data Send (DVS) M,H<-E,[reply]

Description: Any data report which is initiated by an event, such as the completion of a measurement, rather than passage of time is called a discrete variable. Some equipment may have several possible events on which to send the data. S2,F15 is used to select the desired reporting events. Reports requiring only one block of data may report directly to the host with this message. If S6,F3 is multi-block, it must be preceded by the S6,F5/S6,F6 Inquire/Grant transaction.

Structure:

```
L,3
  1.<DATAID>
  2.<CEID>
  3. L,n
       1. L,2
            1. <DSID1>
            2. L,m
                 1. L,2
                      1. <DVNAME1>
                      2. <DVVAL1>
                 2. L,2
                 .
                 .
                 m. L,2
                      1. <DVNAMEm>
                      2. <DVVALm>
       2. L,2
       .
       .
       n. L,2
            1. <DSIDn>
            2. etc.
```

S6,F4 Discrete Variable Data Acknowledge (DVA) S,H->E

Description: Acknowledge or error

Structure: <ACKC6>

S6,F5 Multi-block Data Send Inquire (MBI) S,H<-E,reply

Description: If the discrete data report S6F3, F9, F11, F13 can involve more than one block, this transaction must precede the transmission.

Structure:

```
L,2
  1. <DATAID>
  2. <DATALENGTH>
```

S6,F6 Multi-block Grant (MBG) S,H->E

Description: Grant permission to send

Structure: <GRANT6>

S6,F7 Data Transfer Request (DDR) S,H->E,reply

Description: The host may initiate a data transfer of specified data stored in the equipment with this function.

Structure: <DATAID>

S6,F8 Data Transfer Data (DDD) M,H<-E

Description: Equipment sends data to the host.

Structure: Similar to the structure of S6,F3

Exception: A zero-length item returned means the requested data cannot be sent.

S6,F9 Formatted Variable Send (FVS) M,H<-E,[reply]

Description: The same function as S6,F3 except that the DVNAMEs are supplied from a predefined form that is known to the host. Thus, the data are more compact. If S6,F9 is multi-block, it must be preceded by the S6,F5/S6, F6 Inquire/Grant transaction.

Structure:

```
L,4
  1.<PFCD>
  2.<DATAID>
  3.<CEID>
  4. L,n
       1. L,2
            1. <DSID1_1>
            2. L,m
                 1. <DVVAL_1>
                 .
                 .
                 m. <DVVAL_m>
       2. L,2
       .
       .
       n. L,2
            1. <DSID_n>
            2. etc.
```

S6,F10 Formatted Variable Acknowledge (FVA) S,H->E

Description: Acknowledge or error

Structure: <ACKC6>

S6,F11 Event Report Send (ERS) M,H<-E, reply

Description: The purpose of this message is for the equipment to send a defined, linked, and enabled group of reports to the host upon the occurrence of an event (CEID).

If S6,F11 is Multi-block, it must be preceded by the S6,F5/S6,F6 Inquire/Grant transaction.

Structure:

```
L,3
  1.<DATAID>
  2.<CEID>
  3.L,a
      1. L,2
           1. <RPTID1>
           2. L,b
                1.<V1>
                .
                .
                b.<Vb>
         .
         .
     a.L,2                  report a
         1. <RPTIDa>
         2. L,c#Vs          this report
              1. <V1>
              .
              .
              c.<Vc>
```

Exceptions: If there are no reports linked to the event a 'null' report is assumed. A zero-length list for # of reports means there are no reports linked to the given CEID.

S6,F12 Event Report Acknowledge (ERA) S,H->E

Description: Acknowledge or error

Structure: <ACKC6>

S6,F13 Annotated Event Report Send (AERS) M,H<-E,reply

Description: This message is the same as S6F11 with the exception that VID's are sent with data.

If S6,F13 is Multi-block, it must be preceded by the S6,F5/S6,F6 Inquire/Grant transaction.

Structure:

```
L,3
  1.<DATAID>
  2.<CEID>
  3. L,a
       1. L,2
            1. <RPTID1>
            2. L,b
                 1. L,2
                      1.<VID1>
                 .
                 .
                 b. L,2
                      1.<VIDb>
                      b.<Vb>
       .
       .
       a. L,2
            1. <RPTIDa>
            2. L,c
                 1. L,2
                      1.<VID1>
                      2.<V1>
                    .
                    .
                      c. L,2
                           1.<VIDc>
                           2.<Vc>
```

Exception: If there are no reports linked to the event a 'null' report is assumed. A zero-length list for # of reports means there are no reports linked to the given CEID.

S6,F14 Annotated Event Report Acknowledge (AERA) S,H->E

Description: Acknowledge or error

Structure: <ACKC6>

S6,F15 Event Report Request (ERR) S,H->E, reply

Description: The purpose of this message is for the host to demand a given report group from the equipment.

Structure: <CEID>

S6,F16 Event Report Data (ERD) M,H<-E

Description: Equipment sends reports linked to given CEID to host.

Structure: Identical to structure of S6,F11.

Exceptions: A zero-length item means there are no reports linked to the given CEID.

S6,F17 Annotated Event Report Request (AERR) S,H->E,reply

Description: Same as S6,F15, but requests annotated reports.

Structure: <CEID>

S6,F18 Annotated Event Report Data (AERD) M,H<-E

Description: Equipment sends annotated reports linked to given CEID.

Structure: Same as S6,F13.

Exceptions: A zero-length item means there are no reports linked to the given CEID.

S6,F19 Individual Report Request (IRR) S,H->E,reply

Description: The purpose of this message is for the host to request a defined report from the equipment.

Structure: <RPTID>

S6,F20 Individual Report Data (IRD) M,H<-E

Description: Equipment sends variable data defined for the given RPTID to the host.

Structure: L,n # of variable data items
1. $<V_1>$
.
.
n. $<V_n>$

Exceptions: A zero length list means RPTID is not defined.

S6,F21 Annotated Individual Report Request (AIRR) S,H->E,reply

Description: The purpose of this message is for the host to request an annotated defined report from the equipment.

Structure: <RPTID>

S6,F22 Annotated Individual Report Data (AIRD) M,H<-E

Description: Equipment sends annotated variable data defined for the given RPTID to the host.

Structure:
L,n # of variable data items
1.L,2
1. <VID_1>
2. <V_1>
.
.
n. L,2
1. <VID_n>
2. <V_n>

Exceptions: A zero-length list for # of variable data items means RPTID is not defined.

S6,F23 Request Spooled Data (RSD) S,H->E,reply

Description: The purpose of this message is for the host to request transmission or deletion of the messages currently spooled by the equipment.

Structure: <RSDC>

S6,F24 Request Spooled Data Acknowledgement Send (RSDAS) S,H<-E

Description: The purpose of this message is to acknowledge the receipt of the Request Spooled Data (S6,F23) and to respond with an appropriate acknowledge code.

Structure: <RSDA>

S6,F25 Notification Report Send M, H<->E,[reply]

Description: This message is used for change notifications or confirmation reports. A change notification is a report of an internal action and is not associated with a prior action requested by the host.

A confirmation report is always associated with an earlier request for action. A confirmation report is sent to the initial requestor of a delayed action at the time the action is completed. A delayed action is an action that is any action not performed before the response to the initial request is sent. OPID contains the value of OPID in the initial request. LINKID is set to a non-zero value if and only if additional completion reports with the same OPID will be sent. If S6,F25 is multiblock, it must be preceded by the S6,F5/S6,F6 Inquire Grant transaction.

Structure:

```
L,7
  1. <DATAID>
  2. <OPID>
  3. <LINKID>
  4. <RCPSPEC>
  5. <RMCHGSTAT>
  6. L,m
       1. L,2
            1. <RCPATTRID_1>
            2. <RCPATTRDATA_1>
       .
       .
       m. L,2
            1. <RCPATTRID_m>
            2. <RCPATTRDATA_m>
  7. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE_1>
                 2. <ERRTEXT_1>
            .
            .
            p. L,2
                 1. <ERRCODE_p>
                 2. <ERRTEXT_p>
```

Exception: OPID and LINKID are zero-length items when and only when S6,F25 is sent as a change notification rather than as a confirmation report. p = 0 if and only if RMACK indicates no errors.

S6,F26 Notification Report Send Acknowledge S, H<->E

Description: This message is used to acknowledge the confirmation report. It is defined for completeness and as an aid to the user in identifying problems.

Structure: <ACKC6>

Exception: None

S6,F27 Trace Report Send (TRS) M,H <-E,[reply]

Description: The equipment sends a completed Trace Report to the host.

Structure:

```
L,3
  1. <DATAID>
  2. <TRID>
  3. L,n          (n cannot exceed group size specified by S2F53)
       1. L,p     (p is the number of reports for each trace sample)
            1. L,2
                 1. <RPTID1>
                 2. L,m (number of items in this data report)
                      1. <V1>
                      .
                      .
                      m. <Vm>
                          .
                          .
                         p. L,2
                                1. <RPTIDp>
                                2. L,m
                                       1. <V1>
                                       .
                                       .
                                       m. <Vm>
                             .
                             .
       n. L,p     (p is the number of reports for each trace sample)
            1. L,2
                 1. <RPTID1>
                 2. L,m (number of items in this data report)
                      1. <V1>
                      .
                      .
                      m. <Vm>
            .
            .
            p. L,2
                 1. <RPTIDp>
                 2. L,m
                      1. <V1>
                      .
                      .
                      m. <Vm>
```

Exception: The lists of variables associated with a unique RPTID are also unique. This structure illustrates the form of the message, so in general, V_1 for $RPTID_a$ and V_1 for $RPTID_b$ do not reference the same variable.

S6,F28 Trace Report Send Acknowledge S,H ->E

Description: The host Acknowledges receipt of the Trace Report.

Structure: <TRID>

S6,F29 Trace Report Request (TRR) S,H->E

Description: Request that the data reports assigned to the trace report be sampled and returned to the host.

Structure: <TRID>

S6,F30 Trace Report Data (TRD) M,H<-E

Description: Message containing the requested data reports associated with the TRID of trace data report definition.

Structure:
```
L,3
  1. <TRID>
  2. L,n        (n = number data reports defined for this TRID)
       1. L,2
            1. <RPTID1>
            2. L,m      (m = number of items in this RPTID)
                 1. <V1>
                 .
                 m. <Vm>
       .
       .
       n. L,2
            1. <RPTIDn>
            2. L,m      (m = number of items in this RPTID)
                 1. <V1>
                 .
                 m. <Vm>
            3. <ERRCODE>
```

Exception: If TRID is unknown, a zero-length list (n = 0) shall be sent. Item 3 (ERRCODE) shall be set to zero length when there is no error.

7.11 *Stream 7 Process Program Management* — The functions in this stream are used to manage and transfer process programs. Process programs are the equipment-specific descriptions that determine the procedure to be conducted on the material by a single piece of equipment. Methods are provided to transfer programs as well as establish the link between the process program and the material to be processed with that program.

S7,F0 Abort Transaction (S7F0) S,H<->E

Description: Same form as S1,F0

S7,F1 Process Program Load Inquire (PPI) S,H<->E,reply

Description: This message is used to initiate the transfer of a process program or to select from stored programs. The message may be used to initiate the transfer of an unformatted process program (S7,F3/S7,F4) or a formatted process program (S7,F23/S7,F24),(S7,F31/S7,F32).

Structure:
```
L,2
  1. <PPID>
  2. <LENGTH>
```

S7,F2 Process Program Load Grant (PPG) S,H<->E

```
Description:    This message gives permission for the process program to be loaded.

Structure:      <PPGNT>
```

S7,F3 Process Program Send (PPS) M,H<->E,reply

```
Description:    The program is sent. If S7,F3 is multi-block, it must be preceded by
                the S7,F1/S7,F2 Inquire/Grant transaction.

Structure:      L,2
                  1. <PPID>
                  2. <PPBODY>
```

S7,F4 Process Program Acknowledge (PPA) S,H<->E

```
Description:   Acknowledge or error

Structure:     <ACKC7>
```

S7,F5 Process Program Request (PPR) S,H<->E,reply

```
Description:   This message is used to request the transfer of a process program.

Structure:     <PPID>
```

S7,F6 Process Program Data (PPD) M,H<->E

```
Description:   This message is used to transfer a process program.

Structure:     L,2
                 1. <PPID>
                 2. <PPBODY>

Exception:     A zero-length list means request denied.
```

NOTE 8: The equipment-to-host transfer of the process program, denoted by the R bit in the header (R=1), provides the mechanism for the host computer to receive process programs created on the equipment. This allows use of the equipment without having process program generation capabilities on the host.

S7,F7 Process Program ID Request (PIR) S,H<-E,reply

```
Description:   This message is used to request the PPID for use on the material
               identified.

Structure:     <MID>
```

S7,F8 Process Program ID Data (PID) S,H->E

Description: This message is used to transmit a single matrix entry in response to S7,F7.

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

Exception: A zero-length list returned means no such MID or other error.

S7,F9 M/P M Request (MMR) S,H<->E,reply

Description: This message is used to request the transmission of the material/process matrix. If the message is from the host, the response will be the current matrix in the equipment. If the message is from the equipment, the response will be a new matrix to initialize the equipment.

Structure: Header only

NOTE 9: M/PM defines the Material/Process Matrix. The Material/Process Matrix is a table which links the material to the process program to be used in processing the material.

Description: In the response to S7,F9, the equipment will transmit the current matrix it contains. The matrix will be the sum of all matrix updates transmitted since initialization less the completed material whose linkages have been deleted. Programs with no pending material will be deleted from the matrix but not from the equipment program directory.

Structure:

```
L,n number of process programs
  1. L,2
       1. <PPID1>
       2. L,a (number of MID for this PPID)
            1. <MID1>
            .
            .
            a. <MIDa>
  2. L,2
  .
  .
  n. L,2
       1. <PPIDn>
       2. L,b
            1. <MID1>
            .
            .
            b. <MIDb>
```

Exception: a=0 indicates that this PPID will be used for all material processed. The last default transmitted will be the one used; all other entries will be deleted from the active matrix. A zero-length list returned means no such matrix.

Function 10 Example 2 process programs (1 and 3 MID, respectively)

```
L,2
  L,2
    1. <PPID1>
    2. L,1
         1. <MIDa>
  L,2
    1. <PPID2>
    2. L,3
         1. <MIDb>
         2. <MIDc>
         3. <MIDd>
```

S7,F11 M/P M Update Send(UMS) S,H->E,[reply]

Description: This message is used by the host to add to the M/PM in the equipment.

Structure:

```
L,n (number of process programs)
  1. L,2
       1. <PPID1>
       2. L,a (number of MID's using PPID1)
            1. <MID1>
            .
            .
            a. <MIDa>
  2. L,2
  .
  .
   n. L,2
        1. <PPIDn>
        2. L,b
             1. <MID1>
             .
             .
             b. <MIDb>
```

Exception: If a=0, then the preceding PPID is to be used for all material processed. All other entries will be deleted from the active matrix.

S7,F12 M/P M Update Acknowledge (UMA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC7>

S7,F13 Delete M/P M Entry Send (DES) S,H->E,[reply]

Description: This message is used to delete program to material linkages in the M/PM of the equipment.

Structure:

```
L,n (number of process programs)
  1. L,2
       1. <PPID1>
       L,a (number of MID's using PPID)
         1. <MID1>
         .
         .
         a. <MIDa>
  2. L,2
  .
  .
  n. L,2
       1. <PPIDn>
       2. L,b
            1. <MID1>
            .
            .
            b. <MIDb>
```

Exception: A delete consisting of a zero-length means delete all entries and generate a S7,F9 request to initialize matrix.

S7,F14 Delete M/P M Entry Acknowledge (DEA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC7>

S7,F15 Matrix Mode Select Send (MMS) S,H->E,reply

Description: This message is used by the host to change the method of process program selection in the equipment which might not support all modes.

Structure: <MMODE>

S7,F16 Matrix Mode Select Acknowledge (MMA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC7>

NOTE 10: The matrix structure allows the program linkages to be established for each MID or the multi-MID production plans for an extended period of time. The host system makes the choice of operating mode. By continuous updates to the equipment matrix, automatic system backup is achieved.

S7,F17 Delete Process Program Send (DPS) S,H->E,reply

Description: This message is used by the host to request the equipment to delete process programs from equipment storage.

Structure: L,n (Number of process programs to be deleted)
1. <$PPID_1$>
.
.
n. <$PPID_n$>

Exception: If n=0, then delete all.

S7,F18 Delete Process Program Acknowledge (DPA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC7>

S7,F19 Current EPPD Request (RER) S,H->E,reply

Description: This message is used to request the transmission of the current equipment process program directory (EPPD). This is a list of all the PPIDs of the process programs stored in the equipment.

Structure: Header only

S7,F20 Current EPPD Data (RED) M,H<-E

Description: This message is used to transmit the current EPPD.

Structure: L,n (number of process programs in the directory)
1. <PPID$_1$>
.
.
n. <PPID$_n$>

S7,F21 Equipment Process Capabilities Request (PCR) S,H->E,reply

Description: This message is used to request the Equipment Process Capabilities Data(PCD).

Structure: Header only

S7,F22 Equipment Process Capabilities Data (PCD) M,H<-E

Description: This equipment message provides the information necessary for the host to create and partially verify the contents of a new process program or display the object of a process program previously generated by a host or equipment. The PCD defines the process program content acceptable to the originating equipment.

Structure: L,5
1. <MDLN>
2. <SOFTREV>
3. <CMDMAX>
4. <BYTMAX>
5. L,c (c=Number of Possible Commands)
1. L,11
1. <CCODE>
2. <CNAME>
3. <RQCMD>
4. <BLKDEF>
5. <BCDS>
6. <IBCDS>
7. <NBCDS>
8. <ACDS>
9. <IACDS>
10. <NACDS>
11. L,p (p=Number of Parameters)
1. (parameter specification)
(see below)
.
.
p. (parameter specification)
2. L,11
.
.
c. L,11

Parameter specifications depend on the data type of each parameter. The structure of each of the possible three groups is as follows:

Numeric Data	*String Data*	*Boolean Data*
L,9	L,5	L,4
1. <PNAME>	1. <PNAME>	1. <PNAME>
2. <RQPAR>	2. <RQPAR>	2. <RQPAR>
3. <PDFLT>	3. <PDFLT>	3. <PDFLT>
4. <PMAX>	4. <PMAX>	4. <PMAX>
5. <LLIM>	5. L,s	
6. <ULIM>	1. <STEMP$_1$>	
7. <UNITS>	.	
8. <RESC>	.	
9. <RESV>	s. <STEMP$_s$>	

S7,F23 Formatted Process Program Send (FPS) M,H<->E,reply

Description: This message allows movement of formatted process programs between a piece of equipment and its host system. The values of MDLN and SOFTREV are obtained from the PCD used to generate the process program. If S7,F23 is multi-block, it must be preceded by the S7F1/F2 Inquire/Grant transaction.

Structure:
```
L,4
  1. <PPID>
  2. <MDLN>
  3. <SOFTREV>
  4. L,c (c=Number of Process Commands)
       1. L,2
            1. <CCODE>
            2. L,p (p=Number of Parameters)
                 1. <PPARM1>
                 .
                 .
                 p. <PPARMp>
       2. L,2
       .
       .
       c. L,2
```

S7,F24 Formatted Process Program Acknowledge (FPA) S,H<->E

Description: Acknowledges reception of a formatted process program at its destination and whether the process program was accepted by the interpreter. A returned status of "accepted" by the interpreter means only that the message is understood. The validity of the contents of the process program is determined through a separate transaction (S7,F27/S7,F28).

Structure: <ACKC7>

S7,F25 Formatted Process Program Request (FPR) S,H<->E,reply

Description: This message is used by either equipment or host to request a particular process program from the other.

Structure: <PPID>

S7,F26 Formatted Process Program Data (FPD) M,H<->E

Description: This message transfers a process program in response to a request for the PPID. The values of MDLN and SOFTREV are obtained from the PCD used to generate the process program.

Structure:
```
L,4
  1. <PPID>
  2. <MDLN>
  3. <SOFTREV>
  4. L,c (c=Number of Process Commands)
       1. L,2
            1. <CCODE>
            2. L,p (p=Number of Parameters)
                 1. <PPARM1>
                 .
                 .
                 p. <PPARMp>
       2. L,2
       .
       .
       c. L,2
```

Exception: A zero length list indicates the request was denied.

S7,F27 Process Program Verification Send (PVS) S,H<-E,reply

Description: This message indicates to the host that a process program has been received and checked by the equipment. The result of the check is specified by the list of errors. An empty error list (list of zero-length) or a one-element list with ACKC7A having a value of zero (0) indicates no errors were found in the process program. The equipment may report as many errors as it seems appropriate. The equipment is responsible for sending a single copy of this message to the host after any reception of a formatted process program (S7,F23 or S7,F26 or S7,F31). If S7,F27 is multi-block, it must be preceded by the S7,F29/S7,F30 Inquire/Grant Transaction.

Structure:
```
L,2
  1. <PPID>
  2. L,n (n=number of errors being reported)
       1. L,3
            1. <ACKC7A>
            2. <SEQNUM>
            3. <ERRW7>
       2. L,3
       .
       .
       n. L,3
```

S7,F28 Process Program Verification Acknowledge (PVA) S,H->E

Description: Reply by host to equipment acknowledging reception of Process Program Verification Send (PVS).

Structure: Header only

S7,F29 Process Program Verification Inquire (PVI) S,H<-E,reply

Description: This message allows a piece of equipment to ask a host for permission to send a multi-block PVS.

Structure: <LENGTH>

S7,F30 Process Program Verification Grant (PVG) S,H->E

Description: Reply by host to equipment providing response to Process Program Verification Inquire (PVI).

Structure: <PPGNT>

S7,F31 Verification Request Send (VRS) M,H->E,reply

Description: This message requests the interpreting equipment to check the contents of the provided process program and inform the host whether or not the process program is acceptable for processing at the machine. The values of MDLN and SOFTREV are obtained from the PCD used to generate the process program. If S7,F31 is multi-block, it must be preceded by the S7,F1/S7,F2 Inquire/Grant transaction.

Structure:

```
L,4
  1. <PPID>
  2. <MDLN>
  3. <SOFTREV>
  4. L,c (c=Number of Process Commands)
       1. L,2
            1. <CCODE>
            2. L,p (p=Number of Parameters)
                 1. <PPARM1>
                 .
                 .
                 p. <PPARMp>
       2. L,2
       .
       .
       c. L,2
```

S7,F32 Verification Request Acknowledge (VRA) S,H<-E

Description: Acknowledges reception of a formatted process program verification request at its destination and whether the process program was accepted by the equipment. A returned status of accepted by the interpreter means only that the message is understood. The validity of the contents of the process program is specified through a separate transaction (S7,F27/S7,F28).

Structure: <ACKC7>

S7,F33 Process Program Available Request (PAR) S,H<->E,reply

```
Description:   This message requests the interpreting host or equipment to check its
               process program library and tell the requester if the PPID will be
               supplied if requested.

Structure:     <PPID>
```

S7,F34 Process Program Availability Data (PAD) S,H<->E

```
Description:   This message allows originator to tell requester whether it can
               provide the specified process program and whether it can provide it
               formatted, unformatted, or both.

Structure:     L,3
                 1. <PPID>
                 2. <UNFLEN>
                 3. <FRMLEN>
```

S7,F35 Process Program for MID Request (PPMR) S,H<->E,reply

```
Description:   This message is used to request the transfer of the process program
               to be used for the material identified.

Structure:     <MID>
```

S7,F36 Process Program for MID Data (PPMD) M,H<->E

```
Description:   This message is used to transfer the process program for the material
               identified.

Structure:     L,3
                 1. <MID>
                 2. <PPID>
                 3. <PPBODY>

Exception:     A zero-length list returned means no such MID or other error.
```

7.12 *Stream 8 Control Program Transfer* — The purpose of this stream is to provide the method for transmitting the programs used in the equipment to perform the control function or to execute the transmitted process program.

S8,F0 Abort Transaction (S8F0) S,H<->E

```
Description:   Same form as S1,F0.
```

S8,F1 Boot Program Request (BPR) S,H<->E,reply

```
Description:   This message is used to request the transmission of the boot program.
               It is assumed that there is only one boot program associated with any
               given equipment.

Structure:     Header only
```

S8,F2 Boot Program Data (BPD) M,H<->E

Description: The boot program is required by some systems as a precursor to loading an operating system or executive program.

Structure: <BPD>

Exception: A zero-length item means no boot.

S8,F3 Executive Program Request (EPR) S,H<->E,reply

Description: This message is used to request the executive program. It is assumed that there is only one executive program associated with any given equipment.

Structure: Header only

S8,F4 Executive Program Data (EPD) M,H<->E

Description: The executive program is the master control program of the equipment. The executive may contain all the program required or it may contain the information required to request the rest of the program it needs on Stream 13.

Structure: <EPD>

7.13 *Stream 9 System Errors* — This stream provides a method of informing the host that a message block has been received which cannot be handled or that a timeout on a transaction (receive) timer has occurred. The messages indicate either a Message Fault or a Communications Fault has occurred but do not indicate a Communications Failure has occurred.

7.13.1 *Communications Failure* — A Communications Failure occurs in a SECS-I environment when, and only when, the RTY limit is exceeded.

NOTE 11: In the event of a Communications Failure, no Stream 9 message is sent.

7.13.2 *Communications Fault* — A Communications Fault occurs when the equipment does not receive an expected message (when a transaction timer or a conversation timer has expired).

7.13.3 *Message Fault* — A Message Fault occurs when the equipment receives a message which it cannot process because of a fault that arises from the content, context, or length of the message.

S9,F0 Abort Transaction (S9F0) S,H<->E

Description: Same form as S1,F0.

S9,F1 Unrecognized Device ID (UDN) S,H<-E

Description: The device ID in the message block header did not correspond to any known device ID in the node detecting the error.

Structure: <MHEAD>

S9,F2 Not used

S9,F3 Unrecognized Stream Type (USN) S,H<-E

Description: The equipment does not recognize the stream type in the message block header.

Structure: <MHEAD>

S9,F4 Not Used

S9,F5 Unrecognized Function Type (UFN) S,H<-E

Description: This message indicates that the function in the message ID is not recognized by the receiver.

Structure: <MHEAD>

S9,F6 Not Used

S9,F7 Illegal Data (IDN) S,H<-E

Description: This message indicates that the stream and function were recognized, but the associated data format could not be interpreted.

Structure: <MHEAD>

S9,F8 Not Used

S9,F9 Transaction Timer Timeout (TTN) S,H<-E

Description: This message indicates that a transaction (receive) timer has timed out and that the corresponding transaction has been aborted. It is up to the host to respond to this error in an appropriate manner to keep the system operational.

Structure: <SHEAD>

S9,F10 Not Used

S9,F11 Data Too Long (DLN) S,H<-E

Description: This message to the host indicates that the equipment has been sent more data than it can handle.

Structure: <MHEAD>

S9,F12 Not Used

S9,F13 Conversation Timeout (CTN) S,H<-E

```
Description:   Data were expected but none were received within a reasonable length
               of time. Resources have been cleared.

Structure:     L,2
                 1. <MEXP>
                 2. <EDID>
```

S9,F14 Not Used

7.14 *Stream 10 Terminal Services* — The functions of this stream is to pass textual messages between operator terminals attached to processing and/or testing equipment and the host. The equipment makes no attempt to interpret the text of the message, but merely passes it from terminal keyboard to the host or from the host to the display of the terminal. Management of human response times to information displayed on terminals is the responsibility of the host.

S10,F0 Abort Transaction (S10F0) S,H<->E

```
Description:   Same form as S1,F0.
```

S10,F1 Terminal Request (TRN) S,H<-E,[reply]

```
Description:   A terminal text message to the host.

Structure:     L,2
                 1. <TID>
                 2. <TEXT>
```

S10,F2 Terminal Request Acknowledge (TRA) S,H->E

```
Description:   Acknowledge or error

Structure:     <ACKC10>
```

S10,F3 Terminal Display, Single (VTN) S,H->E, [reply]

```
Description:   Data to be displayed.

Structure:     L,2
                 1. <TID>
                 2. <TEXT>
```

S10,F4 Terminal Display, Single Acknowledge (VTA) S,H<-E

```
Description:   Acknowledge or error

Structure:     <ACKC10>
```

S10,F5 Terminal Display, Multi-Block (VTN) M,H->E,[reply]

Description: Data to be displayed on the equipment's terminal.

Structure:
```
L,2
  1. <TID>
  2. L,N
       1. <TEXT1>
       .
       .
       n.<TEXTn>
```

S10,F6 Terminal Display, Multi-block Acknowledge (VMA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC10>

S10,F7 Multi-block Not Allowed (MNN) S,H<-E

Description: An error message from a terminal that cannot handle a multi-block message from S10,F5.

Structure: <TID>

S10,F8 Not Used

S10,F9 Broadcast (BCN) S,H->E,[reply]

Description: This function is generally the same as S10,F3 except that specific TID in each equipment need not be specified. Instead, the text is directed to each terminal in the equipment when the function is received. This function assumes that this feature exists on all equipment, otherwise repeated S10,F3 messages should be used.

Structure: <TEXT>

S10,F10 Broadcast Acknowledge (BCA) S,H<-E

Description: Acknowledge or error

Structure: <ACKC10>

7.15 Stream 11 has been deleted and will not appear again in this publication.

It is the consensus of the Communications Committee that Stream 11 is obsolete. Its use is discouraged, and it has been removed from the 1989 edition of the standard. The reasons for removal are three-fold:

1. The purpose of this stream, as it was originally envisioned, is perceived to be of little use and can best be accomplished by other means beyond the scope of this standard;
2. The functions in this stream have many technical problems that severely limit their use;
3. There is a noticeable lack of implementations of this standard that utilize Stream11 in its originally intended form.

NOTE 12: Applications that need to transfer unformatted data between the host and equipment should use the facilities of Stream 13.

7.16 *Stream 12 Wafer Mapping* — Messages which deal with coordinate positions and data associated with those positions. This includes functions such as wafer mapping with coordinates of die on a wafer and the associated binning information.

7.16.1 *Structure* — Functions 1 through 20 address the variations required by semiconductor equipment manufacturers in transmitting wafer maps to and from the process equipment (wafer probe through die attach). The functions include three basic formats. The three formats developed are:

1. Row/column format where a coordinate row starting position is given with die count in the row and starting direction. The respective binning information follows for each die.
2. Array format is structured such that a matrix array captures all or part of a wafer with the associated binning information.
3. Coordinate format provides an X/Y location and bin code for die on the wafer.

7.16.2 *Definitions and Descriptions* — The following information is required to perform map association to the physical wafer as it relates to the archival use and transmission of wafer maps.

1. Flat/Notch Location
2. Frame Rotation
3. Row Count
4. Column Count
5. Die Units of Measure
6. Die Size
7. Process Die Count
8. Reference Points
9. Bin Code Equivalents
10. Process Axis
11. Null Bin Code Value
12. ID Type

7.16.2.1 *Flat/Notch Location* — The position in degrees that the flat or notch are oriented during processing relative to a "normal" position of zero degrees. See Figure S12.1.

7.16.2.2 *Frame Rotation* — The orientation of a film frame relative to a "normal" position of zero degrees. See Figure S12.2.

7.16.2.3 *Row/Column Count* — The row and column counts are the total number of rows and columns, respectively, on a wafer which must be correlated directly with the wafer map. These numbers will always be greater than zero.

7.16.2.4 *Die Sizes* — The die size is given in standard units as specified by the die unit of measure item DUTMS, and will also be greater than zero. The value of the die size is determined by measuring the distance from a point on one die to the same point on the next die, often referred to as an index. This is depicted in the lower portion of Figure 7, Section B in the General Rules Section.

7.16.2.5 *Process Die Count* — The process die count item is used by equipment that is being map driven to make determinations about how much material to prepare. For example, a die attach will epoxy lead frames in advance of the attach process. By knowing the total number of die to be processed within a wafer map, the equipment can stop epoxying lead frames equivalent to the last die to be attached. This item is also used by the equipment to tell the host how many total die it processed for that map. For example, a die attach would use PRDCT to report the total die actually attached from a particular wafer.

7.16.2.6 *Reference Points* — Reference points provide a means of relating a map to the physical wafer. The total number of these points, and the method for assigning and detecting them, is the responsibility of the equipment. This standard only provides a means for transmitting them.

7.16.2.7 *Origin* — The origin is in one of five locations which is specified by the equipment when generating a wafer map. The origin is on an array structure having dimensional values equal to those

specified by the row and column count. The origin then lies on one of the four corners of that array or in a center location determined by the following formula:

$$\left| \frac{\text{row} \sim \text{or} \sim \text{column} + 1}{2} \right|_{\text{truncated}}$$

7.16.2.7.1 It is implicit in determining the center location that the upper-left-hand corner of the area, in the normal position, be counted as the first row and column position. An equipment requesting a map provides the origin location that it wants the map to be based on before transmission. If the equipment does not provide an origin, the host must provide a default value. An equipment transmitting a map must provide the origin with the map setup data.

7.16.2.8 *Bin Code Equivalents* — Bin code equivalents is a list of bin codes that the receiving equipment will process. (i.e., if a map contains codes 1 through 10 and the good die are bins 1 and 2, then bin code equivalent list could indicate 1 and 2 if only the good die categories were needed. These are the only bin codes to which an equivalent will drive for its respective process function.) In the case of X/Y coordinate format, the locations transmitted will be only those with the bin codes stated in the Bin Code Equivalent list, unless the length byte is set to zero, in which all of the bin codes in the map will be transmitted.

7.16.2.9 *Process Axis* — The process axis is the axis, either rows or columns, increasing or decreasing, and the side of the map, (top, bottom, left, or right, respectively) that the map data will originate from. This is based on the coordinate system as described under the General Rules section of this document.

7.16.2.10 *ID Type* — ID type indicates the appropriate material ID type (i.e., wafer, cassette, or film frame).

7.16.3 *General Rules*

7.16.3.1 *Map Data Size* — Stream 12 provides for the transmission of a complete map regardless of size. Equipment requiring segmented maps for transmission or reception will not be able to use the Stream 12 functions to handle the complete conversation.

7.16.3.2 *Orientation Conventions* — The orientation of a wafer presented for processing will differ from equipment to equipment. Stream 12 specifies conventions for expressing wafer orientation so that a map can be translated from one geometric representation to another.

7.16.3.2.1 The bottom of the wafer is the notch or the line of the major flat. The orientation of a wafer is measured in positive degrees clockwise (CW) from the "normal" position. The "normal" position is where the bottom of the wafer is closest to you when the wafer is lying horizontally in front of you with the die side facing up. The "normal" position has an orientation of zero degrees. See Figure S12.1 for graphic representation of wafer orientations.

7.16.3.2.2 The bottom of a film frame is also the notch or the line of notches. Its orientation and "normal" position are measured in the same manner as for wafers. See Figure S12.2 for examples of bottoms of film frames.

7.16.3.2.3 The orientation of an unmounted wafer presented for processing is given by the parameter FNLOC, Flat/Notch LOCation.

7.16.3.2.4 The ultimate orientation of a wafer presented for processing after it has been mounted on a film frame is the cumulative rotation of the wafer from the "normal" position on the film frame and the rotation of film frame as it is presented to the equipment. This is determined by the sum of the parameters FNLOC and FFROT, Film Frame ROTation. It is possible for an application to represent the ultimate orientation of a wafer in one of these parameters only and pass the other parameter as zero length.

7.16.3.2.5 Figure 6 shows wafers oriented at 270 degrees with respect to the bottoms of a metal and round film frame. If one of these film frames were presented to an equipment rotated 90 degrees clockwise (CW), (bottom facing the left edge of the page), the ultimate orientation of the wafer would be zero degrees.

7.16.3.2.6 In the case where either FNLOC or FFROT are unknown or irrelevant information, a zero-length data item is transmitted, and the item will be ignored by the application. One of the items must exist.

7.16.3.3 *Coordinate Axis System* — The coordinate axis orientation is shown in Figure S12.3, Section A. The assumption is that the "X" or "column" coordinates increase to the right of the "Y-axis" and the "Y" or "row" coordinates increase above the "X-axis." In describing the physical wafer it is also given that the coordinate axis orientation never rotates. The wafer moves or rotates within the coordinate axis system. The origin within the array describing the wafer's coordinate system must be in one of five locations on that array (the center, upper-left, lower-left, upper-right, or lower-right corner of the array).

7.16.3.3.1 Figures S12.4 and S12.5 summarize the conversation protocol in the form of a flow chart. Since a single transmit inquire/grant can be used for one of three message function pairs, the application is required to examine MAPFT, is received as part of the map setup data to determine the appropriate function to follow. If the appropriate function is not transmitted, the conversation is aborted and the error is reported using the appropriate error reporting stream and function.

S12,F0 Abort Transaction (S12F0) S,H<->E

Description: Same form as S1F0.

S12,F1 Map Set-up Data Send (MSDS) S,H<-E,reply

Description: Used to send all of the map set-up data common to all formats and required to link the data map with the physical wafer.

Structure: L,15
1. <MID>
2. <IDTYP>
3. <FNLOC>
4. <FFROT>
5. <ORLOC>
6. <RPSEL>
7. L,n
1. <$REFP_x$ $REFP_y$>
.
.
n. <$REFP_x$ $REFP_y$>
8. <DUTMS>
9. <XDIES>
10. <YDIES>
11. <ROWCT>
12. <COLCT>
13. <NULBC>
14. <PRDCT>
15. <PRAXI>

S12,F2 Map Set-up Data Acknowledge (MSDA) S,H->E

Description: Acknowledgment of receipt of complete set of map set-up parameters.

Structure: <SDACK>

S12,F3 Map Set-up Data Request (MSDR) S,H<-E,reply

Description: Used to request set-up data from the host for the product ready to be processed at the equipment (common to all formats).

Structure: L,9
1. <MID>
2. <IDTYP>
3. <MAPFT>
4. <FNLOC>
5. <FFROT>
6. <ORLOC>
7. <PRAXI>
8. <BCEQU...>
9. <NULBC>

S12,F4 Map Set-up Data (MSD) S,H->E

Description: Used to send all of the map set-up data required to link the data map with the physical wafer.

Structure:
```
L,15
   1.   <MID>
   2.   <IDTYP>
   3.   <FNLOC>
   4.   <ORLOC>
   5.   <RPSEL>
   6.   L,n
          1. <REFP_x REFP_y>
          .
          .
          n. <REFP_x REFP_y>
   7.   <DUTMS>
   8.   <XDIES>
   9.   <YDIES>
  10.   <ROWCT>
  11.   <COLCT>
  12.   <PRDCT>
  13.   <BCEQU>
  14.   <NULBC>
  15.   <MLCL>
```

Exception: A zero-length list returned means no such MID.

S12,F5 Map Transmit Inquire (MAPTI) S,H<-E,reply

Description: Used to prepare the host for map transmission. S12,F5 must precede all S12,F7-8,F9-10, & F11-12 transactions.

Structure:
```
L,4
  1. <MID>
  2. <IDTYP>
  3. <MAPFT>
  4. <MLCL>
```

S12,F6 Map Transmit Grant (MAPTG) S,H->E

Description: Provides permission to transfer.

Structure: <GRNT1>

S12,F7 Map Data Send Type 1 (MDS1) M,H<-E,reply

Description: Used to send map data from the equipment to the host in row or column compressed format. If S12,F7 is multi-block, it must be preceded by the S12,F5/S12,F6 Inquire/Grant transaction.

Structure:
```
L,3
  1. <MID>
  2. <IDTYP>
  3. L,n
       1. L,2
            1. <RSINF1>
            2. <BINLT1>
       2. L,2
       .
       .
       n. L,2
            1. <RSINFn>
            2. <BINLTn>
```

S12,F8 Map Data Acknowledge Type 1 (MDA1) S,H->E

Description: Acknowledge or error

Structure: <MDACK>

S12,F9 Map Data Send Type 2 (MDS2) M,H<-E,reply

Description: Used to send map data from the equipment in array format. If S12,F9 is multi-block, it must be preceded by the S12,F5/S12,F6 Inquire/Grant transaction.

Structure:
```
L,4
  1. <MID>
  2. <IDTYP>
  3. <STRPxSTRPy>
  4. <BINLT...>
```

S12,F10 Map Data Acknowledge Type 2 (MDA2) S,H->E

Description: Acknowledge or error

Structure: <MDACK>

S12,F11 Map Data Send Type 3 (MDS3) M,H<-E,reply

Description: Used to send map data from the equipment in cartesian coordinate format. Bin values may or may not be included in the message. If S12F11 is multi-block, it must be preceded by the S12,F5/S12,F6 Inquire/Grant transaction.

Structure:
```
L,3
  1. <MID>
  2. <IDTYP>
  3. L,n
       1. L,2
            1. <XYPOS1x XYPOS1y>
            2. <BINLT1...>
       2. L,2
       .
       .
       n. L,2
            1. <XYPOSn>
            2. <BINLTn>
```

S12,F12 Map Data Acknowledge Type 3 (MDA3) S,H->E

Description: Acknowledge or error

Structure: <MDACK>

S12,F13 Map Data Request Type 1 (MDR1) S,H<-E,reply

Description: Used to request map data for product at equipment process station in row or column format.

Structure:
```
L,2
  1. <MID>
  2. <IDTYP>
```

S12,F14 Map Data Type 1 (MD1) M,H->E

Description: Used to send map data from the host to the equipment in row or column format.

Structure:
```
L,3
  1. <MID>
  2. <IDTYP>
  3. L,n
       1. L,2
            1. <RSINFx1 RSINFy1RSINd>
            2. <BINLT...>
       2. L,2
       .
       .
       n. L,2
            1. <RSINFn>
            2. <BINLTn>
```

Exception: A zero-length list returned means no such MID.

S12,F15 Map Data Request Type 2 (MDR2) S,H<-E,reply

Description: Used to request map data for product at an equipment process station, in array format.

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

S12,F16 Map Data Type 2 (MD2) M,H->E

Description: Used to send map data from the host to the equipment in array format.

Structure: L,4
1. <MID>
2. <IDTYP>
3. <$STRP_x$ $STRP_y$>
4. <BINLT...>

Exception: A zero-length list returned means no such MID.

S12,F17 Map Data Request Type 3 (MDR3) S,H<-E,reply

Description: Used to request map data for product at an equipment process station in cartesian coordinate format.

Structure: L,3
1. <MID>
2. <IDTYP>
3. <SDBIN>

S12,F18 Map Data Type 3 (MD3) M,H->E

Description: Used to send map data from the host to the equipment in cartesian coordinate format. Bin values may or may not be included.

Structure: L,3
1. <MID>
2. <IDTYP>
3. L,n
    1. L,2
        1. <$XYPOS_{x1}$ $XYPOS_{y1}$>
        2. <$BINLT_1$...>
    2. L,2
    .
    .
    n. L,2
        1. <$XYPOS_n$>
        2. <$BINLT_n$>

Exception: A zero-length list returned means no such MID.

S12,F19 Map Error Report Send (MERS) S,H<->E

```
Description:   Used to transmit map related errors.

Structure:     L,2
                 1. <MAPER>
                 2. <DATLC>
```

S12,F20 Not Used

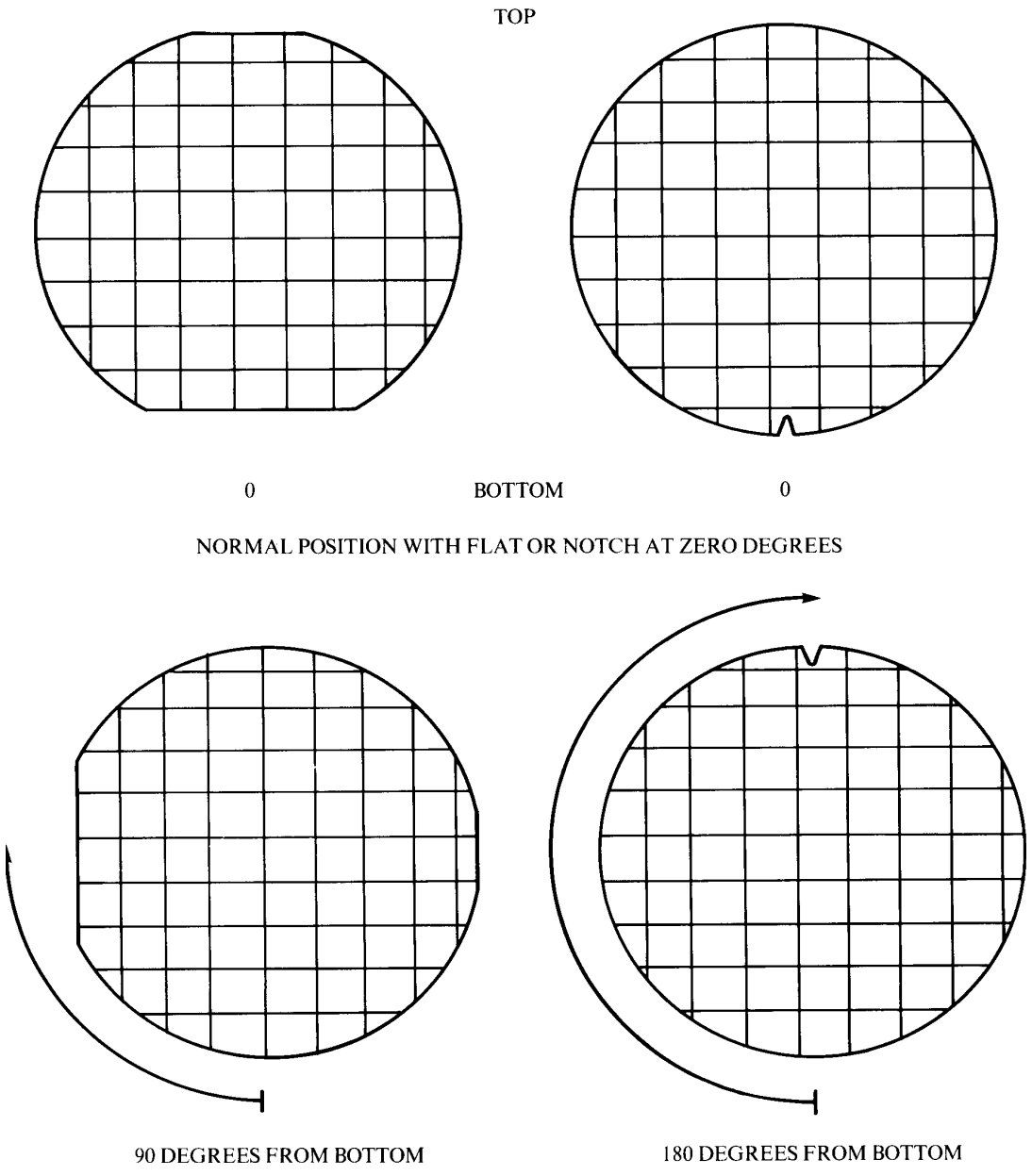


Figure S12.1
Wafer Rotation Position in Degrees

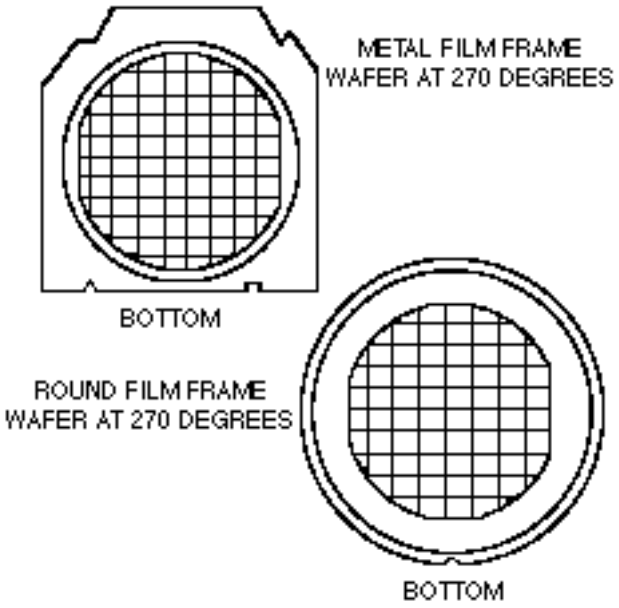


Figure S12.2
Wafer Rotation on Film Frame

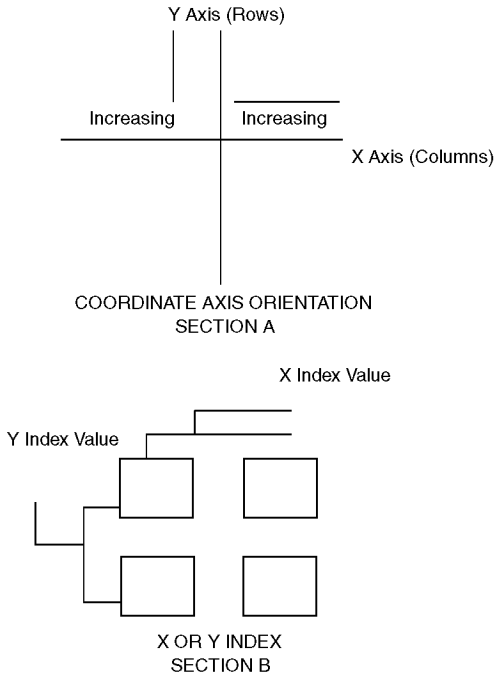


Figure S12.3
Orientation Reference and Index Determination

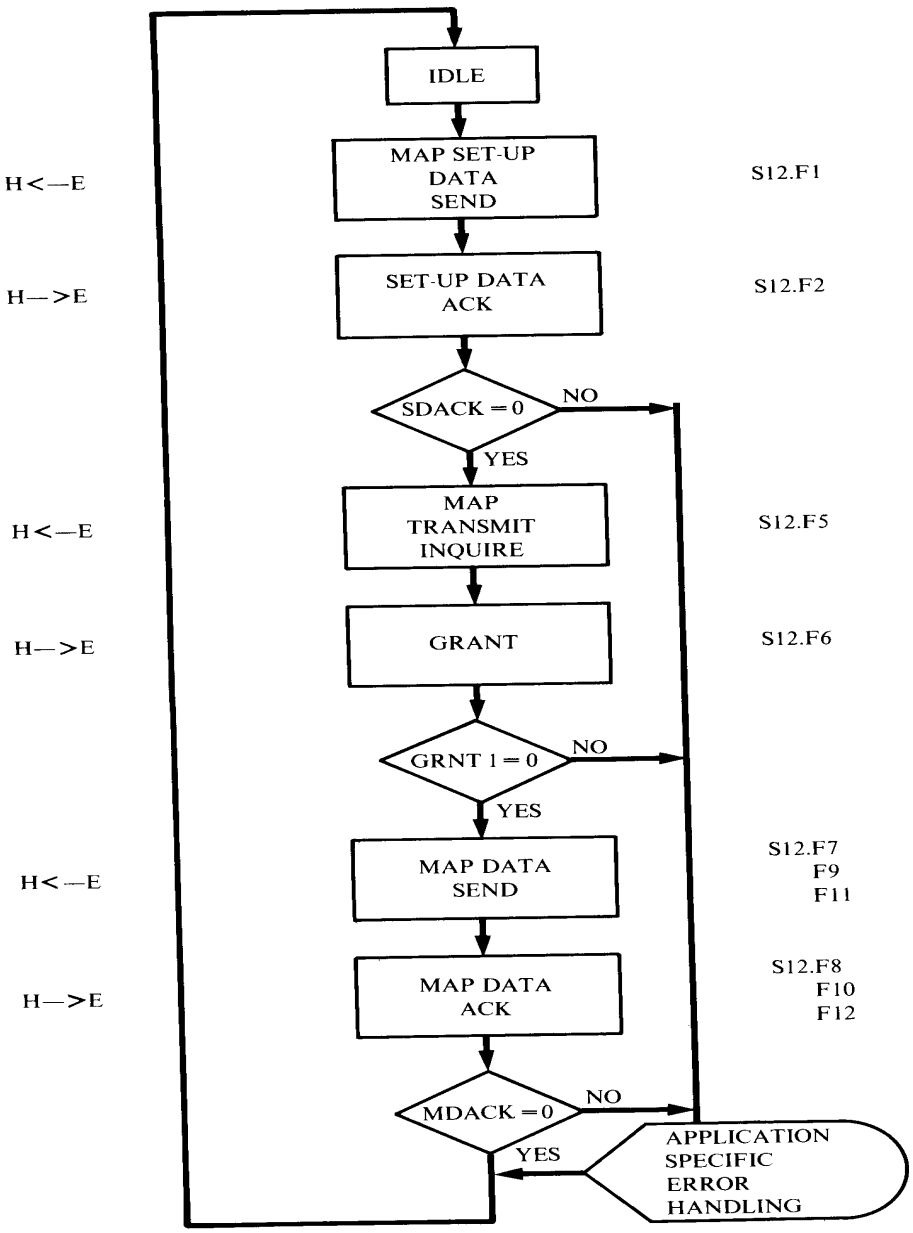


Figure S12.4
Wafer Map Transmitted by Equipment

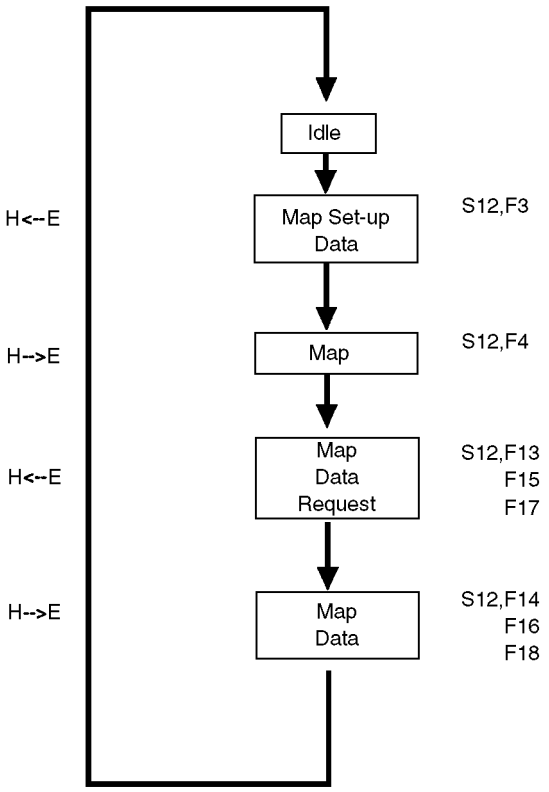


Figure S12.5
Wafer Map Received by Equipment

7.17 *Stream 13 Data Set Transfers* — This stream provides protocols to transfer data sets between systems. It is not intended to provide a general file access mechanism.

7.17.1 *Data Set Characteristics* — The data set may reside on the host or the equipment.

The term *data set* is used in a very general sense. A data set may represent a file, a data structure in memory, a collection of sensor values, or high density wafer profile data. The protocols define only the way data is sent from one system to another and do not define how the data set is stored by either the host or equipment.

The *sending* system is defined to be the system that has the data set. The *receiving* system is defined to be the system to which the data set is being transferred. The host or the equipment may assume either role.

7.17.2 *Unformatted Data Set Protocol* — The protocol for transferring unformatted data sets has the following characteristics:

1. Information about the record structure of the data set may be available.
2. ASCII records are transferred without the record terminating "noise" characters used by some operating systems.
3. Data sets do not need to be transferred in a single message.
4. No arbitrary limits are imposed on the length of one message. The maximum amount of data sent in each message is determined by both the sending and receiving systems, so there can be no data overruns.
5. There is a method of restarting a transfer in the event of an interruption.

7.17.2.1 *Data Set Name* — An ASCII string (format 20) which performs a function similar to the Message ID (Stream and Function) of the SECS-I protocol. This is a logical name which has meaning to both the equipment and the host. Neither the equipment nor the host is

required to use the *Data Set Name* for the same information in any other context. For example, maintenance data may be stored in the equipment in the file "WIDGET.DAT" and in the host as records in a database, but the *Data Set Name* may be "S11,F2".

7.17.2.2 *Records* — The Record Type determines the way the data set is divided into messages for transfer to the receiving system. There are two types of records: Discrete and Stream.

1. A data set with *Discrete* records has a traditional record structure, such as ASCII text. *RecordLength* is the length of the longest record. Zero-length records are allowed. Each record from the data set is sent as a single item in a message.
2. If *Record Type* is *Stream,* then the data set has no internal structure which can be communicated with this protocol.

These kind of data set might be, for example, a dump of main memory, SECS-II structured data, or data which has implicit record boundaries. RecordLength has no meaning for this kind of data set. Items containing data from the data set have no relationship to the structure of the data set.

7.17.2.3 *Transactions* — The basic data transfer is performed by the OPEN, READ, and CLOSE transactions. There is no explicit write transaction. A write is performed indirectly using the SEND transaction. A RESET transaction is provided to allow a graceful recovery after a crash. This protocol describes the transactions over the communications channel only. No assumption is made about the implementation of the transactions. For example, the OPEN transaction on a data set which is stored on a disk file does not necessarily cause the sending system to open the file.

7.17.2.3.1 The OPEN, READ, and CLOSE transactions are initiated by the receiving system. The SEND transaction is initiated by the sending system. The RESET transaction is initiated by either system. The usual transaction timer operates between the primary and secondary messages of each transaction. The time between transactions, especially between READ transactions, is application-specific so no additional timer is defined.

7.17.2.3.2 Internally, the protocol uses a *Handle* to keep track of multiple open data sets, and a *Checkpoint* which aids in error recovery. A value called *ReadLength* is used to negotiate the amount of data sent at one time.

7.17.2.4 *Handle* — Between the sending and receiving systems, more than one data set may be open at a time, or one data set may be opened many times. The Handle is a short name used to keep track of the state of a particular data set and instance of OPEN between the sending and receiving systems (see Figure S13.1). This *Handle* may be thought of as a name for a single application level connection from the sending to the receiving system. Its value is assigned in the primary message of the OPEN transaction.

7.17.2.4.1 The value used for the *Handle* must not be used in another OPEN by the receiving system to the same sending system until it is used in a CLOSE to that sending system, or the RESET transaction is sent by either system. For example, assume a host system opens a data set on equipment 255 using *Handle 1*. The host may not issue another OPEN to equipment 255 using *Handle 1* until it closes 1 on 255. However, the host may use *Handle 1* to open a data set on another piece of equipment, and equipment 255 may use *Handle 1* to open a data set on the host.

7.17.2.4.2 The number of data sets which may be open at one time and the number of times one data set may be opened is not specified by this standard. Error codes are defined for situations where the limits are exceeded. It must be possible to have one outstanding transaction (i.e., a primary message for which there has not been a reply) for each open *Handle*. If the sending system receives a primary message for a Handle which already has an outstanding transaction, then the error code for Pending Transaction is returned in the secondary message (see Figure 10).

7.17.2.5 *Checkpoint* — The response to each READ transaction contains the data and a new Checkpoint value. The *Checkpoint* is defined by and has meaning only for the sending system. Its purpose is to allow a data transfer to be restarted from the point of the last complete message after some communication interruption. The exact nature of the *Checkpoint* is not specified. It could be the byte index in the data set, a record counter (for Discrete records), or some other system-dependent value.

NOTE 13: Checkpoint and the SECS-II transaction timer define a performance requirement for the sending system. The sending system must be able to get data from any checkpoint location within a data set between the receipt of the OPEN primary message and the time for reply to the first READ.

7.17.2.5.1 The value of *Checkpoint* must conform to several rules:

1. The Checkpoint value is exactly four bytes long.
2. The beginning of the data set has Checkpoint value with all bits reset.
3. A Checkpoint with all bits set is illegal.
4. A Checkpoint supplied by the sending system which does not have all bits set is usable in an OPEN

transaction to restart a data transfer without any lost data or duplicated data.

7.17.2.5.2 The receiving system defines the initial *Checkpoint* in the primary message of the OPEN transaction. The sending system returns the next *Checkpoint* in the response to each READ.

7.17.2.6 *Read Length* — The *Read Length* must be supplied by the receiving system with each READ transaction. It specifies the maximum number of data bytes which that system is prepared to process at one time. The sending system may supply less if it has limited resources. The sending system may supply more if *ReadLength* is zero, or is smaller than *RecordLength*.

7.17.2.7 *Reading a Data Set* — The basic data transfer is initiated with the OPEN transaction and completed with the CLOSE transaction. Information is sent from the sending to the receiving systems by a series of READ transactions.

7.17.2.8 *OPEN Transaction* — The receiving system sends a primary message containing the *DataSet Name* of the desired data set, the *Handle* to be used, and the *Checkpoint* of the initial READ transaction. The response from the sending system is a secondary message with a return code and the *RecordType* and *RecordLength* of the data set. If the return code is one of the error codes, then no data set was opened and the values of *RecordType* and *RecordLength* are undefined. If the *RecordType* is *Stream,* then the value of *RecordLength* is undefined. Notice that the undefined items will still appear in the secondary message.

7.17.2.8.1 The return code in the secondary message is one of the following:

OK.

ERROR: Unknown Data Set ID.

ERROR: Try later (i.e., the data set is in use).

ERROR: Too many open data sets.

ERROR: Data set open too many times.

ERROR: Handle in use.

ERROR: Pending Transaction.

7.17.2.9 *The READ Transaction* — The receiving system sends a primary message which contains the *Handle*, and the *ReadLength*. The sending system responds with a secondary message which has a return code, the next Checkpoint, and zero or more items with data. At least one data item must be supplied unless there is an error. The return codes are:

OK.

ERROR: End of Data.

ERROR: No open Data Set (i.e., incorrect Handle).

ERROR: Cannot continue (i.e., a disk read error on the sending system).

ERROR: Pending Transaction.

7.17.2.9.1 Any READ transaction which follows a READ which returned an error, except "Pending Transaction," will generate the same error. The value of *Checkpoint* must be illegal (i.e., all bits must be set) when the "End of Data," "No open Data Set," or "Pending Transaction" error is returned. The value of *Checkpoint* error must be a value from which recovery may be attempted without duplicating data when the "Cannot continue" error is returned. Recovery may be attempted by issuing a CLOSE, followed by an OPEN with the last value of *Checkpoint*, and then another READ.

7.17.2.9.2 Each secondary message for the READ transaction must contain a whole number of *Discrete* records. A record may be sent as an ASCII or a binary item. *Stream* data sets are broken into pieces by the READ transaction without regard to internal structure. Each piece would be sent as a single binary item. The number of items which contain data depends on the *RecordType, Record Length,* and *Read Length*. The algorithm is designed so that the maximum length of the secondary message is deterministic. It gives the receiving system the ability to control the amount of resources (such as SECS-I buffers) which it must allocate. The sending system may send less data than the maximum if it has limited resources. The performance (i.e., the packing of records into a message of some maximum size) should be very good for the case where records are all nearly the maximum length. This is assumed to be the usual case. The efficiency in pathological cases (e.g., many short records) will not be good, but the algorithm is robust enough to accommodate this without exceeding the maximum message size.

NOTE 14: If the *RecordType* is *Stream,* then there is exactly one item with a binary format whose length is not more than *ReadLength*. If the *RecordType* is *Discrete,* then the maximum number of items, MaxItems, is calculated by the formula:

$$\text{MaxItems} = \max\left(1, \text{int}\left(\frac{\text{ReadLength}}{\text{RecordLength}}\right)\right)$$

7.17.2.9.3 The size of the secondary message may be less because of limited resources in the sending system.

7.17.2.9.4 For data sets with *Discrete* records, the format of each item is either ASCII (format 20) or binary (format 10). There is no requirement that all records be in the same format, but mixed record formats are not encouraged. Items with ASCII format should have only

data characters. Characters which the sending system uses for control information (e.g., newline for a record terminator) should not appear. If an application finds it necessary to include these characters, then format 10, or Stream should be used.

7.17.2.10 *CLOSE Transaction* — This transaction terminates a data transfer and frees the *Handle* for future use. The primary message is sent by the receiving system and contains only the *Handle*. The sending system responds with a secondary message which has a return code.

7.17.2.10.1 The return code is one of the following:

OK.

ERROR: No open Data Set (i.e., incorrect *Handle*).

ERROR: Pending Transaction.

7.17.2.11 *Sending a Data Set* — Writing a data set is performed by requesting that the receiving system read it. The sending system initiates the SEND transaction to request that a data set be read. The receiving system is expected to perform the OPEN, READ, and CLOSE transactions to transfer the data set if it accepts the request. The time between the secondary message of the SEND and the primary message of the OPEN depends on the application.

7.17.2.12 *The SEND Transaction* — The primary message sent from the sending system to the receiving system contains the *Data Set Name*.

7.17.2.12.1 The secondary message contains the *Data Set Name* and a return code which is one of the following:

OK.

ERROR: Unknown Data Set Name.

ERROR: Try later (i.e., the system is busy).

7.17.2.13 *Error Recovery* — The receiving system may crash while a data set is open but no READ transaction is pending. The sending system will not be able to tell that this has happened because there is no timeout value defined between READ transactions. When the receiving system is restarted it may have forgotten which data sets were open. States in the two systems are now inconsistent.

7.17.2.14 *RESET Transaction* — The RESET transaction offers a way to resynchronize the two systems. When one system issues the primary message of the RESET transaction, it is informing the other system that any data sets which may have been open are to be closed. This applies to all data sets open between both systems. It is not necessary to issue CLOSE transactions for each individual data set because the RESET transaction is a global close.

7.17.2.15 Any equipment which uses Stream 13 must issue the RESET transaction as part of its initialization or bootstrap procedure. A host system must issue a RESET to equipment which uses Stream 13 during the initialization for that equipment.

7.17.2.16 *SECS-II Protocol Definition* — Figure 10 shows the state diagram for the sending system while a data set is being transferred. Each circle shows a possible state of the sending system. The names of these states are for reference only. They are not meant to suggest an implementation. The arrows show transitions due to SECS-II messages received or sent by the sending system.

7.17.2.16.1 In the initial state (Idle) handle X is not open. The states marked with an asterisk are those in which a transaction is outstanding. If the sending system receives any primary message from the receiving system with handle X during the time it is in these states, then the secondary message for that transaction will contain the "Pending Transaction" error code, but the original transaction for handle X will not be affected. Some states, especially the error states, may take zero time in some implementations. In these cases, the "Pending Transaction" error code would not be returned from those states.

7.17.3 *Formatted Data Sets* — Formatted data sets are data sets transferred in a standard format. Stream 13 provides a method for transferring data sets in a table format. A table has both attributes and content. The attributes of the table provide information about the data set as a whole, such as the date and time that it was last modified, its size, etc. The content of the table consists of column headers and rows. A row is an ordered list of table elements. A column refers to all table elements at a specific position within all rows, where each column is identified by a corresponding text string as a column header. The table elements in the 1st column position are used as an identifier for the row.

S13,F0 Abort Transaction (S13F0) S,H<->E

```
Description:   Same form as S1,F0
```

S13,F1 Send Data Set Send (DSSS) S,H<->E,reply

```
Description:   Sent by the sending system to request that the other system read a
               dataset.

Structure:     L,1
                 1. <DSNAME>
```

S13,F2 Send Data Set Acknowledge (DSSA) S,H<->E

```
Description:   Sent by the receiving system in response to Send Data Set Send.

               1. <DSNAME>
               2. <ACKC13>

Exceptions:    The possible ACKC13 codes for this message are:

               0=O.K.
               1=ERROR:Try later.
               2=ERROR:Unknown Data Set Name.
```

S13,F3 Open Data Set Request (DSOR) S,H<->E,reply

```
Description:   Sent by the receiving system to open a data set for reading.

Structure:     L,3
                 1. <HANDLE>
                 2. <DSNAME>
                 3. <CKPNT>
```

S13,F4 Open Data Set Data (DSOD) S,H<->E

```
Description:   Sent by the sending system in response to Open Data Set Request.

Structure:     L,5
                 1. <HANDLE>
                 2. <DSNAME>
                 3. <ACKC13>
                 4. <RTYPE>
                 5. <RECLEN>

Exceptions:    The possible ACKC13 codes for this message are:

               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.
               9=ERROR:Handle in Use.
              10=ERROR:Pending Transaction.
```

S13,F5 Read Data Set Request (DSRR) S,H<->E,reply

Description: Sent by the receiving system to read data from an open data set.

Structure:
```
L,2
  1. <HANDLE>
  2. <READLN>
```

S13,F6 Read Data Set Data (DSRD) M,H<->E

Description: Sent by the sending system in response to Read Data Set Request.

Structure:
```
L,4
  1. <HANDLE>
  2. <ACKC13>
  3. <CKPNT>
  4. L,n
       1. <FILDAT>
       .
       .
       n. <FILDAT>
```

Exceptions: The possible item formats, number of items (n), and length of each FILDAT item (|th) are given by the following table. MaxItems is defined in Section 7.17.10.2.

RTYPE	0 (Stream)	1 (Discrete)
Item Format	10 (binary)	10 (binary) or 20 (ASCII)
Maximum n	1	MaxItems
Maximum n	1 (ACKC13=0) 0(any error)	1 (ACKC13=0) 0(any error)
Maximum |th	READLN	RECLEN
Maximum |th	0	0

The possible ACKC13 codes for this message are:
0 = O.K.
6 = ERROR: No open Data Set
7 = ERROR: Cannot Continue
8 = ERROR: End of Data
10= ERROR: Pending Transaction

S13,F7 Close Data Set Send (DSCS) S,H<->E,reply

Description: Sent by the receiving system to close an open data set.

Structure:
```
L,1
  1. <HANDLE>
```

S13,F8 Close Data Set Acknowledge (DSCA) S,H<->E

```
Description:   Sent by the sending system in response to Close Data Set Send (DSCS).

Structure:     L,2
                 1. <HANDLE>
                 2. <ACKC13>

The possible ACKC13 codes for this message are:
               0 = O.K.
               6 = ERROR:No open Data Set.
               10= ERROR:Pending Transaction.
```

S13,F9 Reset Data Set Send (DSRS) S,H<->E,reply

```
Description:   Sent by either system to close all open data sets.

Structure:     Header only
```

S13,F10 Reset Data Set Acknowledge (DSRA) S,H<->E

```
Description:   Sent in response to Reset Data Set Send.

Structure:     Header only
```

S13,F11 Data Set Object Multi-Block Inquire (DSOMGI) S,H<->E,reply

```
Description:   This message requests permission to send a multi-block data set. If
               the receiving system does not grant permission in the reply, the
               multi-block data set may not be sent. OBJSPEC is used to identify the
               data set object type and identifier and may include a destination.
               DATALENGTH represents the total message length, not the length of the
               data set.

Structure:     L,3
                 1. <DATAID>
                 2. <OBJSPEC>
                 3. <DATALENGTH>
```

S13,F12 Data Set Object Multi-Block Grant (DSOMBG) S,H<->E

```
Description:   This message grants or denies permission to send a multi-block data
               set.

Structure:     <GRANT>
```

S13,F13 Table Data Send (TDS) M,H<->E,reply

Description: This message allows the host and the equipment to exchange predefined datasets in a tabular format. The first element of every row is used to reference that row for all other elements. If S13,F13 is Multi-block, it must be preceded by the S13,F11/S13,F12 Inquire/Grant transaction.

Structure:

```
L,8
  1. <DATAID>
  2. <OBJSPEC>
  3. <TBLTYP>
  4. <TBLID>
  5. <TBLCMD>
  6. L,n                        # of table attributes
       1. L,2
            1. <ATTRID_1>
            2. <ATTRDATA_1>
       .
       .
       n. L,2
            1. <ATTRID_n>
            2. <ATTRDATA_n>
  7. L,c                        # of column definitions
       1. <COLHDR_1>            1st column element description
       .
       c. <COLHDR_c>            cth column element description
  8. L,r                        # of row definitions
       1. L,c_1                 # of entries per definition
            1. <TBLELT_11>      1st table element, 1st row
            .
            m. <TBLELT_1c1>     mth table element, 1st row
       .
       r. L,c_r                 rth row definition
            1. <TBLELT_r1>      1st table element, rth row
           .
            m. <TBLELT_rcr>     mth table element, rth row
```

Exception: If OBJSPEC is a zero-length item, then the owner of the table is the receiver of the message. If r is zero, any existing table definition of the given type and id is to be deleted. Otherwise, c_1 may not be zero, and the value of c_1 shall be less than or equal to the value of c.

S13,F14 Table Data Acknowledge (TDA) S,H<->E

Description: This message is used to acknowledge the receipt of a table and to indicate any errors.

Structure:
```
L,2
  1. <TBLACK>
  2. L,p
       1. L,2
            1. <ERRCODE_1>
            2. <ERRTEXT_1>
       .
       .
       P. L,2
            1. <ERRCODE_p>
            2. <ERRTEXT_p>
```

Exception: p = 0 if and only if TBLACK indicates no errors.

S13,F15 Table Data Request (TDR) M,H<->E,reply

Description: This message allows the host or the equipment to request part or all of a specific table. Either specific columns or specific rows may be requested, but not both at the same time. If S13,F15 is Multi-block, it must be preceded by the S13,F11/S13,F12 Inquire/Grant transaction.

Structure:
```
L,7
  1. <DATAID>
  2. <OBJSPEC>
  3. <TBLTYP>
  4. <TBLID>
  5. <TBLCMD>
  6. L,p                    # of column definitions
       1. <COLHDR_1>        1st column element description
       .
       p. <COLHDR_p>        pth column element description
  7. L,q
       1. <TBLELT_1>        1st row identifier
       .
       .
       q. <TBLELT_q>
```

Exception: If OBJSPEC is a zero-length item, then the owner of the table is the receiver of the message. Either p or q, or both, must be zero. If p = 0 and q = 0, all rows are requested; otherwise, only the specified columns, or the rows referenced by TBLELT, are requested.

Description: This message is used to return data from the requested table.

Structure:

```
L,6
  1. <TBLTYP>
  2. <TBLID>
  3. L,n                        # of table attributes
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       n. L,2
            1. <ATTRIDn>
            2. <ATTRDATAn>
  4. L,c                        # of column definitions
       1. <COLHDR1>             1st column element description
       .
       c. <COLHDRc>             cth column element description
  5. L,r                        # of row definitions
       1. L,c1                  # of entries per definition
            1. <TBLELT11>       1st table element, 1st row
            .
            m. <TBLELT1c1>      last table element, 1st row
       .
       r. L,cr                  rth row definition
            1. <TBLELTr1>       1st table element, rth row
            .
            m. <TBLELTrcr>      mth table element, rth row
  6. L,2
       1. <TBLACK>
       2. L,p
            1. L,2
                 1. <ERRCODE11>
                 2. <ERRTEXT12>
            .
            P. L,2
                 1. <ERRCODEp1>
                 2. <ERRTEXTp2>
```

Exceptions: p = 0 if, and only if, TBLACK indicates no errors. The length c_{11} of a table row may not exceed the value of c.

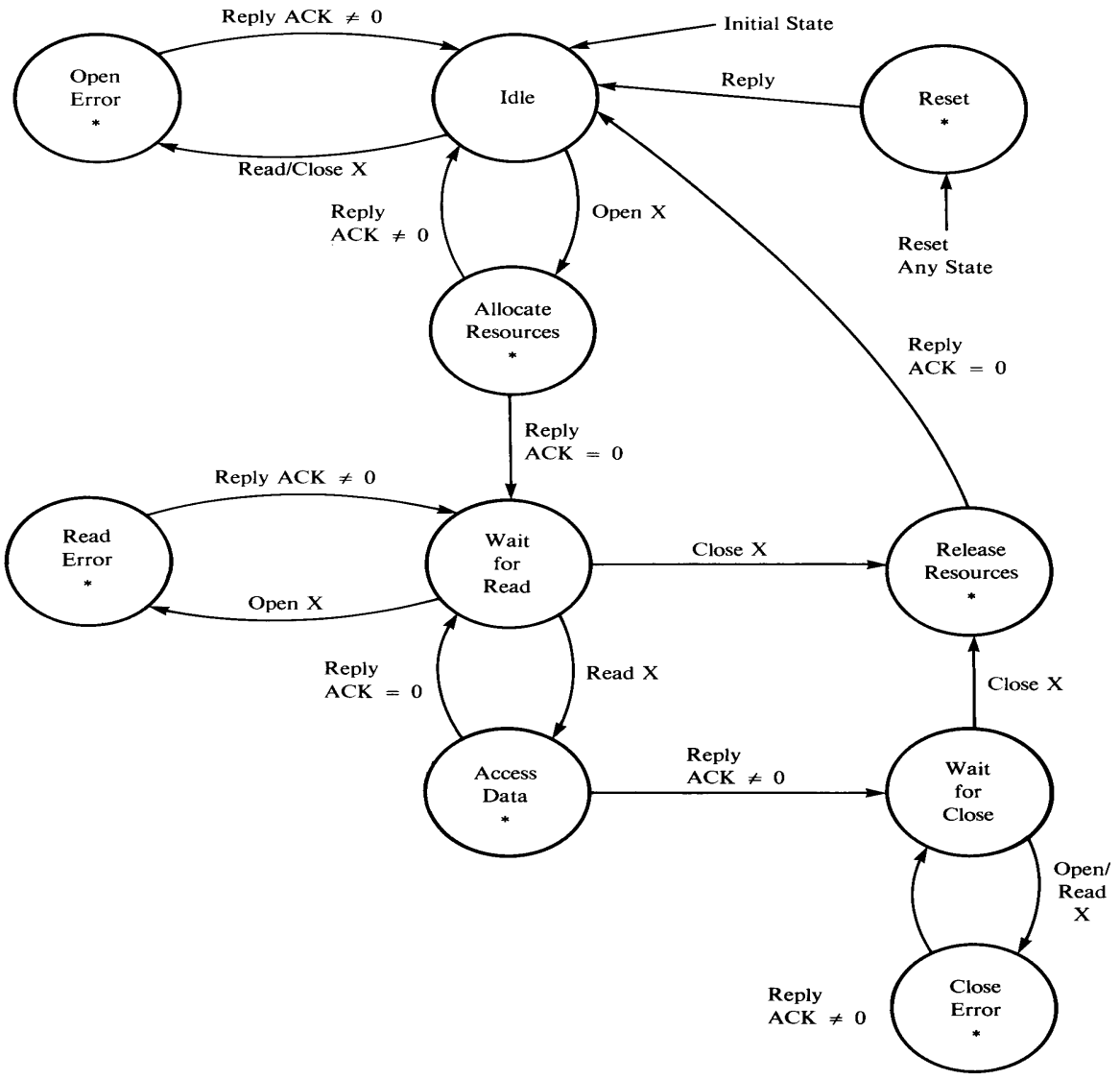


Figure S13.1
The Sending System's State Diagram During a Data Set Transfer
An Asterisk (*) marks the states where a primary message which uses handle X would result in a secondary message with the error code for "Pending Transaction."

7.18 *Stream 14 Object Services* — The functions in this stream are used for generic functions concerning objects, including obtaining information about objects and setting values for an object.

S14,F0 Abort Transaction (S14F0) S,H<->E

Description: Same form as S1,F0.

S14,F1 GetAttr Request (GAR) S,H<->E,reply

Description: This message is used to request a set of specified attributes for one or more objects. It consists of an "object specifier" for the owner of the target objects (the objects of interest), the target object type, a list of identifiers of the target objects, a filter (a list of qualifying relationships) that limits the target objects of interest to those that meet all of the qualifications in and the specific attributes whose values are requested.

The object specifier provides a specification of the owner of the target object(s). It contains a sequence of hierarchical object relationships. Each element of the object specifier identifies a specific object instance that is the superior of the following object instance in the sequence. The last object instance in the sequence is in a hierarchical relationship to the target objects.

The target object type designates the type of the target object, and the list of object identifiers indicates the specific instance of that type that are of interest. The target type may be omitted only if object identifiers are unique across all object types and the list of identifiers is not empty.

The object filter is an optional list of qualifications, each of which provides a condition to be applied to the object instances of interest. Each qualification objects of interest are those that meet all of the specified qualifications.

The attribute relationship quantifier is a logical binary relationship $ATTRRELN_i$ that the specified qualifying value $ATTRDATA_i$ has to the corresponding attribute of each instance of the desired object type(s). The objects that are to be qualified with this filter have an attribute value V_i such that the statement "$ATTRDATA_i$ $ATTRRELN_i$ V_i" is TRUE. If $ATTRRELN_i$ is omitted, the relationship of equality is intended.

For ASCII attribute values $ATTRDATA_i$, the characters for question mark "?" and asterisk "*" are used as "wild characters" to provide filtering for certain object types. The character "?" may be used in any attribute or key attribute value with an ASCII format to represent "any single character" and may be repeated. The asterisk character "*" may be similarly used to represent a variable-length string, including a null string. The string "*x" represents a string of any length that ends in "x", the string "x*" represents any string that begins with "x", and the string "*" represents any string of any non-zero length. The comparison for text characters is case insensitive.

Equipment is not required to support wild characters in particular, or attribute filters in general.

Structure: L,5

1. <OBJSPEC>
2. <OBJTYPE>
3. L,i i = identifiers of the object instances requested
    1. <$OBJID_1$>
    .
    .
    i. <$OBJID_i$>
4. L,q q = # object qualifiers to match
    1. L,3
        1. <$ATTRID_1$>
        2. <$ATTRDATA_1$>
        3. <$ATTRRELN_1$>
    .
    .
    q. L,3
        1. <$ATTRID_q$>
        2. <$ATTRDATA_q$>
        3. <$ATTRRELN_q$>
5. L,a a = # attributes requested
    1. <$ATTRID_1$>
    .
    .
    a. <$ATTRID_a$>

Exception: If OBJSPEC is a zero-length item, no object specifier is provided. If i = 0,only the filter is to be applied. If q = 0, no filter is specified. If both i and q = 0, information for all instances of the objects are requested. If a = 0, all attributes are requested.

Description: This message is used to transfer the set of requested attributes for the specified object(s). The order of attributes is retained from the primary message.

Structure:

```
L,2
  1. L,n                  n = number of objects
       1. L,2
            1. <OBJID1>
            2. L,a        a = number of attributes
                 1. L,2
                      1. <ATTRID1>
                      2. <ATTRDATA1>
                 .
                 .
                 a. L,2
                      1. <ATTRIDa>
                      2. <ATTRDATAa>
                      .
                      .
       n. L,2
            1. <OBJIDn>
            2. L,b        b = number of attributes
                 1. L,2
                      1. <ATTRID1>
                      2. <ATTRDATA1>
                 .
                 .
                 b. L,2
                      1. <ATTRIDb>
                      2. <ATTRDATAb>
  2. L,2
       1. <OBJACK>
       2. L,p             p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If OBJSPEC is a zero-length item, no object specifier is provided. If n = 0, no objects matched the specified filter. If p = 0, no errors were detected.

S14,F3 SetAttr Request (SAR) S,H<->E, reply

Description: This message is used to request that a given set of attributes be assigned specified values for all objects of the specified type and exactly matching the specified attribute requirements. Certain attributes may not be changed through the interface. For a description of filters, see S14,F1.

Structure:

```
L,4
  1. <OBJSPEC>
  2. <OBJTYPE>
  3. L,i                 i = number of object instances requested
       1. <OBJID1>
       .
       .
       i. <OBJIDi>
  4. L,n                 n = # attribute settings
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       n. L,2
            1. <ATTRIDn>
            2. <ATTRDATAn>
```

Exception: If OBJSPEC is a zero-length item, no object specifier is provided.

S14,F4 SetAttr Data (SAD) M,H<->E

Description: This message is used to acknowledge that the attributes for the specified objects have been set as requested or to indicate an error for each attribute value that was not set as requested. The order of attributes is retained from the primary message.

Structure:

```
L,2
  1. L,i                  i = number of objects requested
       1. L,2
            1. <OBJID1>
            2. L,n        n = number of attributes set.
                 1. L,2
                      1. <ATTRID1>
                      2. <ATTRDATA1>
                 .
                 .
                 n. L,2
                      1. <ATTRIDn>
                      2. <ATTRDATAn>
       .
       .
       i. L,2
            1. <OBJIDi>
            2. L,n
                 1. L,2
                      1. <ATTRID1>
                      2. <ATTRDATA1>
                 .
                 .
                 n. L,2
                      1. <ATTRIDn>
                      2. <ATTRDATAn>
  2. L,2
       1. <OBJACK>
       2. L,p             p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If n = 0 for any object, the object was not found.
If p = 0, no errors were detected.

S14,F5 GetType Request (GTR) S,H<->E,reply

Description: This message is used to request the types of objects owned by an object. This is an operation performed on an object type rather than on object instances. Wild characters "?" and "*" may be used as a filter for object types. Equipment is not required to support wild characters.

Structure: <OBJSPEC>

Exception: If OBJSPEC is a zero-length item, no object specifier is provided.

S14,F6 GetType Data (GTD)

```
Structure:     L,2
                 1. L,n              n = number of object types
                      1. <OBJTYP1>
                      .
                      .
                      n. <OBJTYPn>
                 2. L,2
                      1. <OBJACK>
                      2. L,p         p = number of errors reported
                           1. L,2
                                1. <ERRCODE1>
                                2. <ERRTEXT1>
                           .
                           .
                           p. L,2
                                1. <ERRCODEp>
                                2. <ERRTEXTp>
```

Exception: If n = 0, there are no owned object types. If p = 0, no errors were detected.

S14,F7 GetAttrName Request (GANR) S,H<->E,reply

Description: This message is used to request the names of the attributes of specified types of owned objects. This is an operation performed on an object type rather than on object instances. Wild characters "?" and "*" may be used as a filter for object types. Equipment is not required to support wild characters.

```
Structure:     L,2
                 1. <OBJSPEC>
                 2. L,n              n = # of object types
                      1. <OBJTYP1>
                      .
                      .
                      n. <OBJTYPn>
```

Exception: If OBJSPEC is a zero-length item, no object specifier is provided.

S14,F8 GetAttrName Data (GAND) M,H<->E

Description: This message contains the names of the attributes of the requested objects.

Structure:

```
L,2
  1. L,n                n = number of object types
       1. L,2
            1. <OBJTYP1>
            2. L,a      a = number of attributes
                 1. <ATTRID1>
                 .
                 .
                 a. <ATTRIDa>
       .
       .
       n. L,2
            1. <OBJTYPn>
            2. L,b      b = number of attributes
                 1. <ATTRID1>
                 .
                 .
                 b. <ATTRIDb>
  2. L,2
       1. <OBJACK>
       2. L,p           p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If OBJSPEC is a zero-length item, no objects matched the specified filter.
If p = 0, no errors were detected.

S14,F9 Create Object Request (COR) M,H<->E,reply

Description: This message is used to request an object owner to create an object instance. OBJSPEC specifies the object owner.

Structure:

```
L,3
  1. <OBJSPEC>
  2. <OBJTYPE>
  3. L,a                a = # attributes requested
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       a. L,2
            1. <ATTRIDa>
            2. <ATTRDATAa>
```

Exception: If OBJSPEC is a null-length item, no object specifier is provided. If a = 0, no specific attribute settings are requested for the new object.

S14,F10 Create Object Acknowledge (CAO) M,H<->E

Description: This message is used to acknowledge the success or failure of creating the new object specified. If successful, OBJSPEC is the object specifier of the new object. The list of attributes returned is dependent upon the type of object specified.

Structure:

```
L,3
  1. <OBJSPEC>
  2. L,b                b = number of attributes returned
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       b. L,2
            1. <ATTRIDb>
            2. <ATTRDATAb>
  3. L,2
       1. <OBJACK>
       2. L,p           p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If OBJSPEC is a null-length item, no object was created. If b = 0, no attributes of the new object are returned. If p = 0, no errors were detected.

S14,F11 Delete Object Request S,H<->E,reply

Description: This message is used to request that the object specified in OBJSPEC be deleted. The list of attribute settings depends upon the type of object to be deleted.

Structure:
```
L,2
  1. <OBJSPEC>
  2. L,a                n = # attribute settings
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       a. L,2
            1. <ATTRIDa>
            2. <ATTRDATAa>
```

Exception: If n = 0, no attribute settings are provided.

S14,F12 Delete Object Acknowledge (DOA) M,H<->E

Description: This message is used to acknowledge the success or failure of deleting the object specified. The list of attributes returned is dependent upon the type of object to be deleted.

Structure:
```
L,2
  1. L,b                n = number of attributes returned
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       b. L,2
            1. <ATTRIDb>
            2. <ATTRDATAb>
  2. L,2
       1. <OBJACK>
       2. L,p           p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If n = 0, no attribute values are returned. If p = 0, no errors were detected.

S14,F13 Object Attach Request (OAR) M,H<->E,reply

Description: This message is sent by a supervisor to request the object specified in OBJSPEC to attach or reattach itself to the requestor.

Structure:

```
L,2
  1. <OBJSPEC>
  2. L,a                a = # attribute settings
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       a. L,2
            1. <ATTRIDa>
            2. <ATTRDATAa>
```

Exception: If a = 0, no attribute settings are provided.

S14,F14 Object Attach Acknowledge (OAA) M,H<->E

Description: This message is used to acknowledge the success or failure of the requested attachment. If successful, a non-zero token shall be returned for the supervisor's use in subsequent communications with the attached object.

Structure:

```
L,3
  1. <OBJTOKEN>
  2. L,b                b = number of attributes
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       b. L,2
            1. <ATTRIDb>
            2. <ATTRDATAb>
  3. L,2
       1. <OBJACK>
       2. L,p           p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: OBJTOKEN is zero if and only if p is non-zero. If b = 0, no attribute values are returned. If p = 0, no errors were detected.

S14,F15 Attached Object Action Request (AOAR) M,H<->E,reply

Description: This message is used by a supervisor (only) to request an attached object to perform an action.

Structure:
```
L,4
  1. <OBJSPEC>
  2. <OBJCMD>
  3. <OBJTOKEN>
  4. L,a                a = # attribute settings
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       a. L,2
            1. <ATTRIDa>
            2. <ATTRDATAa>
```

Exception: If a = 0, no attribute settings are provided.

S14,F16 Attached Object Action Acknowledge (AOAA) M,H<->E

Description: This message is used to acknowledge the success or failure of an action requested by a supervisor.

Structure:
```
L,2
  1. L,b                b = number of attributes
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       b. L,2
            1. <ATTRIDb>
            2. <ATTRDATAb>
  2. L,2
       1. <OBJACK>
       2. L,p           p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If p = 0, no errors were detected.

S14,F17 Supervised Object Action Request (SOAR) S,H<->E,reply

Description: This message is used to request a supervisor to have a supervised attached object perform an action. OBJSPEC specifies the supervisor, and TARGETSPEC specifies the attached object.

Structure:

```
L,4
  1. <OBJSPEC>
  2. <OBJCMD>
  3. <TARGETSPEC>
  4. L,a                a = number of attribute settings
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       a. L,2
            1. <ATTRIDa>
            2. <ATTRDATAa>
```

Exception: If a = 0, no attribute settings are provided.

S14,F18 Supervised Object Action Acknowledge (SOAA) M,H<->E

Description: This message is used to acknowledge the success or failure of an action requested of a supervisor.

Structure:

```
L,2
  1. L,b              b = number of attributes
       1. L,2
            1. <ATTRID1>
            2. <ATTRDATA1>
       .
       .
       b. L,2
            1. <ATTRIDb>
            2. <ATTRDATAb>
  2. L,2
       1. <OBJACK>
       2. L,p         p = number of errors reported
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If b = 0, no attributes are returned. If p = 0, no errors were detected.

7.19 *Stream 15 Recipe Management* — The functions in this stream are used requesting information and operations concerning recipes, recipe namespaces, and recipe executors. A recipe is an object that is transferred in sections, where a section consists of either recipe attributes, agent-specific dataset attributes, or the body of the recipe. An attribute is information concerning the recipe body, the recipe as a whole, or the application of the recipe. An attribute consists of an attribute name/attribute value pair.

S15,F0 Abort Transaction (S15F0) S,H<->E

```
Description:   Same form as S1,F0.
```

S15,F1 Recipe Management Multi-block Inquire S,H<->E, reply

```
Description:   This message requests permission to send a multi-block message based
               upon a maximum length of the total message.

Structure:     L,3
                 1. <DATAID>
                 2. <RCPSPEC>
                 3. <RMDATASIZE>

Exception:     If RCPSPEC is zero-length, the multi-block message for which
               permission to send is requested does not contain a recipe.
```

S15,F2 Recipe Management Multi-block Grant S,H<->E

```
Description:   This message grants or denies permission to send a multi-block
               message.

Structure:     <RMGRNT>
```

S15,F3 Recipe Namespace Action Request S,H<->E, reply

```
Description:   This message requests that a recipe namespace be created or deleted.

Structure:     L,2
                 1. <RMNSSPEC>
                 2. <RMNSCMD>
```

S15,F4 Recipe Namespace Action Acknowledge M,H<->E

```
Description:   This message is used to confirm whether the requested action was
               completed successfully or to provide error information otherwise.

Structure:     L,2
                 1. <RMACK>
                 2. L,p
                      1. L,2
                           1. <ERRCODE_1>
                           2. <ERRTEXT_1>
                      .
                      .
                      p. L,2
                           1. <ERRCODE_p>
                           2. <ERRTEXT_p>

Exception:     p = 0 if and only if RMACK indicates no errors.
```

S15,F5 Recipe Namespace Rename Request S,H<->E, reply

Description: A request is made for a recipe namespace to be renamed.

Structure: L,2
1. <RMNSSPEC>
2. <RMNEWNS>

S15,F6 Recipe Namespace Rename Acknowledge M,H<->E

Description: This message is used to acknowledge or deny a request to rename a recipe namespace.

Structure: L,2
1. <RMACK>
2. L,p
1. L,2
1. <ERRCODE$_1$>
2. <ERRTEXT$_1$>
.
.
p. L,2
1. <ERRCODE$_p$>
2. <ERRTEXT$_p$>

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F7 Recipe Space Request S,H<->E, reply

Description: This message requests the amount of recipe storage available in the storage of a recipe namespace or recipe executor, as indicated by its object specifier OBJSPEC.

Structure: <OBJSPEC>

Exception: None.

S15,F8 Recipe Space Data M,H<->E

Description: This message contains the amount of storage available for recipes.

Structure: L,2
1. <RMSPACE>
2. L,2
1. <RMACK>
2. L,p
1. L,2
1. <ERRCODE$_1$>
2. <ERRTEXT$_1$>
.
.
p. L,2
1. <ERRCODE$_p$>
2. <ERRTEXT$_p$>

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F9 Recipe Status Request S,H<->E, reply

Description: This message is used to request the status of a recipe and the next available numeric version for that recipe class and name.

Structure: <RCPSPEC>

Exception: None.

S15,F10 Recipe Status Data M,H<->E

Description: This message contains the protected status of the recipe and the next available version number for that recipe class and name.

Structure:
```
L,3
  1. <RCPSTAT>
  2. <RCPVERS>
  3. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE_1>
                 2. <ERRTEXT_1>
            .
            .
            p. L,2
                 1. <ERRCODE_p>
                 2. <ERRTEXT_p>
```

Exception: RCPVERS is a zero-length item if and only if the recipe does not exist. p = 0 if and only if RMACK indicates no errors.

S15,F11 Recipe Version Request S,H<->E, reply

Description: This message is used to request the best version of a recipe for the specified agent.

Structure:
```
L,4
  1. <RMNSSPEC>
  2. <RCPCLASS>
  3. <RCPNAME>
  4. <AGENT>
```

Exception: If item 2 is zero length, the recipe class PROCESS is indicated. If item 4 is a zero-length item, no agent is specified.

S15,F12 Recipe Version Data M,H<->E

Description: This message contains the recommended version.

Structure:
```
L,3
  1. <AGENT>
  2. <RCPVERS>
  3. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE_1>
                 2. <ERRTEXT_1>
            .
            .
            p. L,2
                 1. <ERRCODE_p>
                 2. <ERRTEXT_p>
```

Exception: If AGENT is a zero-length item in the request, it shall also be a zero-length item in the reply. If it is not zero-length in the request, and it is of zero-length in the reply, then no qualifying recipe was found specific to that equipment. p = 0 if and only if RMACK indicates no errors.

S15,F13 Recipe Create Request M,H<->E, reply

Description: This message is used to create or modify a recipe body. If multi-block, it shall be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:
```
L,5
  1. <DATAID>
  2. <RCPUPDT>
  3. <RCPSPEC>
  4. L,m
       1. L,2
            1. <RCPATTRID_1>
            2. <RCPATTRDATA_1>
       .
       .
       m. L,2
            1. <RCPATTRID_m>
            2. <RCPATTRDATA_m>
  5. <RCPBODY>
```

Exception: RCPBODY may be of zero length.

Description: This message is used to acknowledge that a recipe has been created or updated with the body sent in the request.

Structure:
```
L,2
  1. <RMACK>
  2. L,2
       1. L,2
            1. <ERRCODE_1>
            2. <ERRTEXT_1>
       .
       .
       p. L,2
            1. <ERRCODE_p>
            2. <ERRTEXT_p>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F15 Recipe Store Request M,H<->E, reply

Description: This message is used to send a recipe, or one or more recipe sections, to a recipe namespace. If multi-block, it shall be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:

```
L,4
  1. <DATAID>
  2. <RCPSPEC>
  3. <RCPSECCODE>
  4. L,q                (q = 1,2,3)
       1. L,r           (r = 0 or 2)
            1. <RCPSECNM>
            2. L,g      (g = # generic attributes)
                 1. L,2
                      1. <RCPATTRID1>
                      2. <RCPATTRDATA1>
                 .
                 .
                 g. L,2
                      1. <RCPATTRIDg>
                      2. <RCPATTRDATAg>
       2. <RCPBODY>
       3. L,m           (m = # agent-specific datasets)
            1. L,2
                 1. <RCPSECNM1>
                 2. L,a
                      1. L,2
                           1. <RCPATTRID11>
                           2. <RCPATTRDATA11>
                      .
                      .
                      a. L,2
                           1. <RCPATTRID1a>
                           2. <RCPATTRDATA1a>
            .
            .
            m. L,2
                 1. <RCPSECNMm>
                 2. L,b
                      1. L,2
                           1. <RCPATTRIDm1>
                           2. <RCPATTRDATAm1>
                      .
                      .
                      b. L,2
                           1. <RCPATTRIDmb>
                           2. <RCPATTRDATAmb>
```

Exception: RCPBODY is a zero-length item when the body is omitted. If g = 0, no generic attributes are transferred and RCPBODY shall be a zero-length item. If m = 0, no agent-specific datasets are transferred.

S15,F16 Recipe Store Acknowledge M,H<->E

Description: This message is used to acknowledge that the specified recipe has been stored as requested or to indicate the error(s).

Structure:
```
L,2
  1. <RECPSECCODE>
  2. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE_1>
                 2. <ERRTEXT_1>
            .
            .
            p. L,2
                 1. <ERRCODE_p>
                 2. <ERRTEXT_p>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F17 Recipe Retrieve Request S,H<->E, reply

Description: This message is used to get a recipe, or one or more recipe sections, from a recipe namespace.

Structure:
```
L,2
  1. <RCPSPEC>
  2. <RCPSECCODE>
```

Exception: None.

Description: This message is used to acknowledge that the specified recipe, or recipe sections, have been set as requested, or to indicate the error(s).

Structure:

```
L,2
  1. L,q                  (q = 1,2,3)
       1. L,r             (r = 0 or 2)
            1. <RCPSECNM>
            2. L,g        (g = # generic attributes)
                 1. L,2
                      1. <RCPATTRID1>
                      2. <RCPATTRDATA1>
                 .
                 g. L,2
                      1. <RCPATTRIDg>
                      2. <RCPATTRDATAg>
       2. <RCPBODY>
       3. L,m             (m = # agent-specific datasets)
            1. L,2
                 1. <RCPSECNM1>
                 2. L,a
                      1. L,2
                           1. <RCPATTRID11>
                           2. <RCPATTRDATA11>
                      .
                      .
                      a. L,2
                           1. <RCPATTRID1a>
                           2. <RCPATTRDATA1a>
            .
            .
            m. L,2
                 1. <RCPSECNMm>
                 2. L,b
                      1. L,2
                           1. <RCPATTRIDm1>
                           2. <RCPATTRDATAm1>
                      .
                      .
                      b. L,2
                           1. <RCPATTRIDmb>
                           2. <RCPATTRDATAmb>
            2. L,2
                 1. <RMACK>
                 2. L,p
                      1. L,2
                           1. <ERROCODE1>
                           2. <ERRTEXT1>
                      .
                      .
                      p. L,2
                           1. <ERRCODEp>
                           2. <ERRTEXTp>
```

Exception: If r = 0, no generic attributes are transferred and RCPBODY shall be a zero-length item. If m = 0, no agent-specific datasets are transferred. p = 0 if and only if RMACK indicates no errors.

S15,F19 Recipe Rename Request S,H<->E, reply

Description: This message is used to request that a recipe be copied to, or renamed to, a recipe with a new identifier.

Structure:
```
L,3
  1. <RCPSPEC>
  2. <RCPRENAME>
  3. <RCPNEWID>
```

Exception: None.

S15,F20 Recipe Rename Acknowledge M,H<->E

Description: This message acknowledges the request to copy or rename a recipe and indicates whether the action was successfully performed or errors that occurred.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE_1>
            2. <ERRTEXT_1>
       .
       .
       p. L,2
            1. <ERRCODE_p>
            2. <ERRTEXT_p>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F21 Recipe Action Request M,H<->E, reply

Description: This message is used to acknowledge the request to perform an action on one or more recipes within a namespace.

Structure:
```
L,6
  1. <DATAID>
  2. <RCPCMD>
  3. <RMNSSPEC>
  4. <OPID>
  5. <AGENT>
  6. L,n
       1. <RCPID_1>
       .
       .
       n. <RCPID_n>
```

Exception: AGENT may be a zero-length item except for requests for certify, de-certify, download, and upload.

S15,F22 Recipe Action Acknowledge M,H<->E

Description: This message is used to acknowledge the request to originate a new recipe.

Structure:
```
L,4
  1. <AGENT>
  2. <LINKID>
  3. <RCPCMD>
  4. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE_1>
                 2. <ERRTEXT_1>
            .
            .
            p. L,2
                 1. <ERRCODE_p>
                 2. <ERRTEXT_p>
```

Exception: LINKID is zero if and only if all requested actions have been completed. p = 0 if and only if RMACK indicates no errors.

S15,F23 Recipe Descriptor Request M,H<->E, reply

Description: This message is used to request the descriptors of a list of the specified recipes. If multi-block, it must be preceded by the S15,F1/F2 inquire/grant transaction. OBJSPEC is the object specifier of either a recipe namespace of a recipe executor.

Structure:
```
L,3
  1. <DATAID>
  2. <OBJSPEC>
  3. L,n
       1. <RCPID_1>
       .
       .
       n. <RCPID_n>
```

Exception: None.

Description: This message returns the requested descriptors in the same order as requested.

Structure:

```
L,2
  1. L,n                (n = number of recipes from request)
       1. L,a           (descriptors for recipe #1)
            1. L,r      (r = 0 or 3) (1st component descriptor)
                 1. <RCPDESCNM11>
                 2. <RCPDESCTIME11>
                 3. <RCPDESCLTH11>
            .
            .
            a. L,r      (r = 0 or 3)
                 1. <RCPDESCNM1a>
                 2. <RCPDESCTIME1a>
                 3. <RCPDESCLTH1a>
       .
       .
       n. L,b           (descriptors for recipe #n)
            1. L,r      (r = 0 or 3) (1st component descriptor)
                 1. <RCPDESCNMn1>
                 2. <RCPDESCTIMEn1>
                 3. <RCPDESCLTHn1>
            .
            .
            b. L,r      (r = 0 or 3)
                 1. <RCPDESCNMnb>
                 2. <RCPDESCTIMEnb>
                 3. <RCPDESCLTHnb>
  2. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: A zero-length recipe descriptor (r = 0) means that the specified recipe does not exist (could not be located). p = 0 if and only if RMACK indicates no errors.

S15,F25 Recipe Parameter Update Request M,H<->E, reply

Description: This message is used to update the variable parameter definitions for a specific agent. If multi-block, it must be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:
```
L,4
  1. <DATAID>
  2. <RMNSSPEC>
  3. <AGENT>
  4. L,n
       1. L,3
            1. <RCPPARNM1>
            2. <RCPPARVAL1>
            3. <RCPPARRULE1>
       .
       .
       n. L,3
            1. <RCPPARNMn>
            2. <RCPPARVALn>
            3. <RCPPARRULEn>
```

Exception: None.

S15,F26 Recipe Parameter Update Acknowledge M, H<->E

Description: This message indicates the successful performance of the request or otherwise indicates the nature of error(s) that occurred.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE1>
            2. <ERRTEXT1>
       .
       .
       p. L,2
            1. <ERRCODEp>
            2. <ERRTEXTp>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F27 Recipe Download Request M,H->E,reply

Description: This message is used to send a recipe to a recipe executor. If multi-block, it shall be preceded by the S15,F1/S15,F2 inquire/grant transaction.

Structure:
```
L,5
  1. <DATAID>
  2. <RCPOWCODE>
  3. <RCPSPEC>
  4. L,m
       1. L,2
            1. <RCPATTRID1>
            2. <RCPATTRDATA1>
       .
       .
       m. L,2
            1. <RCPATTRIDm>
            2. <RCPATTRDATAm>
  5. <RCPBODY>
```

Exception: None.

S15,F28 Recipe Download Acknowledge M,H<-E

Description: This message is used to acknowledge that a recipe has been received by the recipe executor. If the recipe was successfully verified, the results are returned to the sender. RCPID contains the identifier of a derived object form recipe if created during verification.

Structure:
```
L,3
  1. <RCPID>
  2. L,n (n = # of attributes)
       1. L,2
            1. <RCPATTRID1>
            2. <RCPATTRDATA1>
          .
          .
       n. L,2
            1. <RCPATTRIDn>
            2. <RCPATTRDATAn>
  3. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If item is a zero length item, no derived object form recipe was originated. n = 0 if and only if the recipe was not verified or failed verification. p = 0 if and only if RMACK indicates no errors.

S15,F29 Recipe Verify Request M,H->E, reply

Description: This message is used to request verification of one or more recipes by a recipe executor. If multi-block, it shall be preceded by the S15F1, F2 inquire/grant transaction. The operation identifier OPID, used where multiple verification requests may be outstanding, may be zero if no further verifications will be requested before all current verification requests are completed by the recipe executor. Otherwise, OPID is generated to be unique for the requestor. RESPEC is the object specifier for the recipe executor.

Structure:
L,4
1. <DATAID>
2. <OPID>
3. <RESPEC>
4. L,m
1. <$RCPID_1$>
.
.
m. <$RCPID_m$>

Exception: If RESPEC is a zero length item, the target is the recipient of the message.

S15,F30 Recipe Verify Acknowledge M,H<-E

Description: This message is used to acknowledge the request to verify one or more recipes. If a single recipe verification was requested and the recipe was successfully verified, the results are returned to the sender in this message, and RCPID contains the identifier of a derived object form recipe if created during verification. If multiple recipe verifications were requested, then LINKID shall be non-zero.

Structure:

```
L,5
  1. <OPID>
  2. <LINKID>
  3. <RCPID>
  4. L,n (n = # attributes)
       1. L,2
            1. <RCPATTRID1>
            2. <RCPATTRDATA1>
       .
       .
       n. L,2
            1. <RCPATTRIDn>
            2. <RCPATTRDATAn>
  5. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: LINKID is zero if and only if a single recipe verification was requested and has been completed. If item 3 is zero length item, no derived object form recipe was originated. n = 0 if and only if the recipe was not verified or failed verification. p = 0 if and only if RMACK indicates no errors.

S15,F31 Recipe Upload Request S,H->E,reply

Description: This message is used to request an execution recipe from a recipe executor.

Structure: <RCPSPEC>

Exception: None.

Description: This message is used to send an execution recipe from a recipe executor.

Structure:

```
L,4
  1. <RCPSPEC>
  2. L,m (m = # attributes)
       1. L,2
            1. <RCPATTRID1>
            2. <RCPATTRDATA1>
       .
       .
       m. L,2
            1. <RCPATTRIDm>
            2. <RCPATTRDATAm>
  3. <RCPBODY>
  4. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F33 Recipe Select Request M,H->E,reply

Description: This message is used to request the selection of one or more execution recipes. If multi-block, it shall be preceded by the S15,F1/S15,F2 inquire/grant transaction.

Structure:
```
L,3
1. <DATAID>
2. <RESPEC>
3. L,r            (r = # selections)
   1. L,2
      1. <RCPID1> (1st recipe selection)
      2. L,p      (p = # parameter settings for 1st recipe)
         1. L,2
            1. <RCPPARNM11>
            2. <RCPPARVAL11>
         .
         .
        p. L,2
            1. <RCPPARNM1p>
            2. <RCPPARVAL1p>
   .
   .
   r. L,2
      1. <RCPIDr> (rth recipe selection)
      2. L,s      (s = # parameter settings for rth recipe)
         1. L,2
            1. <RCPPARNMr1>
            2. <RCPPARVALr1>
         .
         .
         s. L,2
            1. <RCPPARNMrs>
            2. <RCPPARVALrs>
```

Exception: If the list of parameter settings for a recipe selection is of zero length, then no parameter settings are specified for the corresponding recipe.

S15,F34 Recipe Select Acknowledge M,H<-E

Description: This message is used to acknowledge the request for recipe selection.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE1>
            2. <ERRTEXT1>
       .
       .
       p. L,2
            1. <ERRCODEp>
            2. <ERRTEXTp>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F35 Recipe Delete Request M,H->E,reply

Description: This message is used to request that one or more recipes be deleted or deselected. If multi-block, it shall be preceded by the S15,F1/ S15,F2 inquire/grant transaction.

Structure:
```
L,4
  1. <DATAID>
  2. <RESPEC>
  3. <RCPDEL>
  4. L,n (n = # recipes deselected)
       1. <RCPID1>
       .
       .
       n. <RCPIDn>
```

Exception: If n = 0 and recipes are to be deselected (RCPDEL = 1), then all currently-selected recipes are indicated.

S15,F36 Recipe Delete Acknowledge M,H<-E

Description: This message is used to acknowledge the request that recipes be deleted or deselected.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE1>
            2. <ERRTEXT1>
       .
       .
       p. L,2
            1. <ERRCODEp>
            2. <ERRTEXTp>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F37 DRNS Segment Approve Action Request S,H<->E,reply

Description: This message is sent by a distributed recipe namespace manager to an attached distributed recipe namespace segment to approve an action previously requested by the segment. If multi-block, it shall be preceded by the S15,F1/S15,F2 inquire/grant transaction.

Structure:
```
L,6
  1. <RMSEGSPEC>
  2. <OBJTOKEN>
  3. <RMGRNT>
  4. <OPID>
  5. <RCPID>
  6. <RMCHGTYPE>
```

Exception: None.

S15,F38 DRNS Segment Approve Action Acknowledge M,H<->E

Description: This message is used to acknowledge or deny the approve action request.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE_1>
            2. <ERRTEXT_1>
       .
       .
       p. L,2
            1. <ERRCODE_p>
            2. <ERRTEXT_p>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F39 DRNS Recorder Segment Request M,H<->E,reply

Description: This message is used by the distributed recipe namespace manager to request that an attached recorder create or delete a segment specifier record. If multi-block, it shall be preceded by the S15,F1/ S15,F2 inquire/grant transaction.

Structure:
```
L,5
  1. <DATAID>
  2. <RMNSCMD>
  3. <RMRECSPEC>
  4. <RMSEGSPEC>
  5. <OBJTOKEN>
```

Exception: None.

S15,F40 DRNS Recorder Segment Acknowledge M,H<->E

Description: This message is used to acknowledge the request to add or delete a segment specifier record.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE_1>
            2. <ERRTEXT_1>
       .
       .
       p. L,2
            1. <ERRCODE_p>
            2. <ERRTEXT_p>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F41 DRNS Recorder Modify Request M,H<->E,reply

Description: This message is used by a distributed recipe namespace manager to a recorder to store or delete a change request record. If multi-block, it shall be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:
```
L,5
  1. <DATAID>
  2. <RMRECSPEC>
  3. <OBJTOKEN>
  4. <RMNSCMD>
  5. L,c      (c = 1 or 7)
       1. <RCPID>
       2. <RCPNEWID>
       3. <RMSEGSPEC>
       4. <RMCHGTYPE>
       5. <OPID>
       6. <TIMESTAMP>
       7. <RMREQUESTOR>
```

Exception: If RMNSCMD = create, then c = 7, otherwise c = 1.

S15,F42 DRNS Recorder Modify Acknowledge M,H<->E

Description: This message is used to acknowledge a request to store or delete a change request.

Structure:
```
L,2
  1. <RMACK>
  2. L,p
       1. L,2
            1. <ERRCODE_1>
            2. <ERRTEXT_1>
            .
            .
       p. L,2
            1. <ERRCODE_p>
            2. <ERRTEXT_p>
```

Exception: p = 0 if and only if RMACK indicates no errors.

S15,F43 DRNS Get Change Request M,H<->E,reply

Description: This message is used to request a distributed recipe namespace recorder or manager to return change requests records for a specific recipe or assigned to a specific segment. If multi-block, it shall be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:
```
L,3
  1. <DATAID>
  2. <OBJSPEC>
  3. <TARGETSPEC>
```

Exception: If TARGETSPEC is omitted, OBJSPEC identifies a recipe.

S15,F44 DRNS Get Change Request Data M,H<->E

Description: This message is used to return the specified change request records.

Structure:
```
L,2
  1. L,n      n = # change requests
       1. L,7
            1. <RCPID1>
            2. <RCPNEWID1>
            3. <RMSEGSPEC1>
            4. <RMCHGTYPE1>
            5. <OPID1>
            6. <TIMESTAMP1>
            7. <RMREQUESTOR1>
       .
       .
       n. L,7
            1. <RCPIDn>
            2. <RCPNEWIDn>
            3. <RMSEGSPECn>
            4. <RMCHGTYPEn>
            5. <OPIDn>
            6. <TIMESTAMPn>
            7. <RMREQUESTORn>
  2. L,2
       1. <RMACK>
       2. L,p
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            p. L,2
                 1. <ERRCODEp>
                 2. <ERRTEXTp>
```

Exception: If n = 0, no change records were found matching the specification. p = 0 if and only if RMACK indicates no errors.

S15,F45 DRNS Manager Segment Change Approval Request M,H<->E,reply

Description: This message is sent to a distributed recipe namespace manager by an attached distributed recipe namespace segment to request approval for a specific type of change to a recipe. If multi-block, it shall be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:
```
L,4
  1. <DATAID>
  2. <RCPSPEC>
  3. <RCPNEWID>
  4. <RMCHGTYPE>
```

Exception: RCPNEWID is a zero-length item except where RMCHGTYPE specifies a copy or rename change.

S15,F46 DRNS Manager Segment Approval Acknowledge S,H<->E

Description: This message is used to acknowledge the request to change a recipe.

Structure:
```
L,3
  1. <RMCHGTYPE>
  2. <RMGRNT>
  3. <OPID>
```

Exception: OPID is zero if and only if RMGRNT indicates the change is denied.

S15,F47 DRNS Manager Rebuild Request M,H<->E, reply

Description: This message requests a distributed recipe namespace manager specified in OBJSPEC to rebuild a distributed recipe namespace. Either a distributed recipe namespace recorder or a list of distributed recipe namespace segment specifiers shall be provided.If multi-block, it shall be preceded by the S15,F1/F2 inquire/grant transaction.

Structure:
```
L,5
  1. <DATAID>
  2. <OBJSPEC>
  3. <RMNSSPEC>
  4. <RMRECSPEC>
  5. L,n
       1. <RMSEGSPEC1>
       .
       .
       n. <RMSEGSPECn>
```

Exception: If RMRECSPEC is a non-zero length item, then n is zero. If RMRECSPEC is a zero length item, then n is non-zero.

S15,F48 DRNS Manager Rebuild Acknowledge M,H<->E

Description: This message is used to acknowledge the request to rebuild a distributed recipe namespace.

Structure:
```
L,2
  1. <RMACK>
  2. L,P
       1. L,2
            1. <ERRCODE1>
            2. <ERRTEXT1>
       .
       .
       p. L,2
            1. <ERRCODEp>
            2. <ERRTEXTp>
```

Exception: p = 0 if and only if RMACK indicates no errors.

7.20 *Steam 16 Processing Management* — This stream provides protocol for a set of messages that enable the control of material processing at equipment and equipment resources. Control is implemented by supporting two job types; the control job and the process job. A process job is a single unit of work that ensures that the appropriate processing is applied to a particular material by a processing resource. The Process Job provides a widely applicable supervisory control capability for automated processing of material in equipment, irrespective of the particular process being used. The Process Job creates a transient link between the three elements of the manufacturing process; the first is the material to be processed. The second is the equipment on which the process will occur. The third is the process specification, a Process Recipe. When a Process Job has completed, it ceases to exist; its Process Job ID is no longer valid. The control job is used to group a set of related process jobs. The group is logically related from the host's viewpoint. For instance; if a carrier contains multiple lots, then the process jobs for each lot (in the carrier) could be included in the control job specification. Control jobs also provide mechanisms for specifying the destination for processed material.

S16,F0 Abort Transaction (S16F0) S,H<->E

```
Description:   Same form as S1F0.
```

S16,F1 Multi-block Process Job Data Inquire (PRJI) S,H->E,reply

```
Description:   If any of Processing Management messages are larger than one block,
               then this transaction must precede that message.

Structure:     L,2
                 1. <DATAID>
                 2. <DATALENGTH>
```

S16,F2 Multi-block Process Job Data Grant (PRJG) S,H<-E

```
Description:   Message to indicate if permission is granted to transmit a multi-
               block Job Data message.

Structure:   <GRANT>
```

S16,F3 Process Job Create Request (PRJCR) M,H->E,reply

Description: The purpose of this message is to request material to be processed on a Process Module.

Structure:
```
L,5
  1. <DATAID>
  2. <MF>
  3. L,n
       1. <MID1>
       .
       .
       n. <MIDn>
  4. L,3
       1. <PRRECIPEMETHOD>
       2. <RCPSPEC>
       3. L,m (m = {c,2})
            1. L,2
                 1. <RCPPARNM1>
                 2. <RCPPARVAL1>
            .
            .
            .
            m. L,2
                 1. <RCPPARNMm>
                 2. <RCPPARVALm>
  5. <PRPROCESSSTART>
```

Exception: For the m length list m = 0 may be allowed value depending on the value of PRRECIPEMETHOD.

S16,F4 Process Job Create Acknowledge (PRJCA) S,H<-E

Description: Acknowledge or report error in the creation of a Process Job.

Structure:
```
L,2
  1. <PRJOBID>
  2. L,2
       1. <ACKA>
       2. L,n
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

Exception: This list may be zero length, generally the case when ACKA indicates success. When ACKA indicates a create failure, the equipment may supply one or more ERRCODE's.

S16,F5 Process Job Command Request (PRJCMDR) M,H->E,reply

Description: Send a job control command to a processing job.

Structure:
```
L,4
  1. <DATAID>
  2. <PRJOBID>
  3. <PRCMDNAME>
  4. L,n
       1. L,2
            1. <CPNAME1>
            2. <CPVAL1>
       .
       .
       .
       n. L,2
            1. <CPNAMEn>
            2. <CPVALn>
```

Exception: The CPNAME, CPVAL pairs are command parameter identifiers and values; n = 0 is valid for some commands (PRCMDNAME).

S16,F6 Process Job Command Acknowledge (PRJCMDA) S,H<-E

Description: The processing service sends its confirmation for receipt of a command request.

Structure:
```
L,2
  1. <PRJOBID>
  2. L,2
       1. <ACKA>
       2. L,n          (n = {0,n})
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

Exception: This list n may be zero length.

S16,F7 Process Job Alert Notify (PRJA) S,H<-E,[reply]

Description: The processing service may notify the controlling entity of important events. The Process Job Milestones only assume small number of different values. However, the conditions under which a process job meets one of these milestones may vary. For instance, a Job may reach Job Complete because the Process was Aborted. By using item 4, the status of the Alert (PRJOBMILESTONE) can be indicated. See the list of Error Codes for Processing in Data Item Dictionary.

Structure:
```
L,4
  1. <TIMESTAMP>
  2. <PRJOBID>
  3. <PRJOBMILESTONE>
  4. L,2
       1. <ACKA>
       2. L,n            (n = {0,n})
            1. L,2
                 1. <ERRCODE_1>
                 2. <ERRTEXT_1>
            .
            .
            .
            n. L,2
                 1. <ERRCODE_n>
                 2. <ERRTEXT_n>
```

Exception: The list n may be zero length.

S16,F8 Process Job Alert Confirm (PRJAC) S,H->E

Description: Host confirms receipt of Process Job Alert message from the equipment.

Structure: Header only

S16,F9 Process Job Event Notify (PRJE) S,H<-E,[reply]

Description: Send Processing Job related event to the controlling entity.

Structure:
```
L,4
  1. <PREVENTID>
  2. <TIMESTAMP>
  3. <PRJOBID>
  4. L,n
       1. L,2
            1. <VID_1>
            2. <V_1>
       .
       .
       .
       n. L,2
            1. <VID_n>
            2. <V_n>
```

Exception: The VID, V pairs are variable data identifiers and values; exceptions n = 0 is valid for some events (PREVENTID).

S16,F10 Process Job Event Confirm (PRJEC) S,H->E

Description: Host confirms receipt of S16,F9 message to equipment.

Structure: Header only.

S16,F11 PRJobCreateEnh M,H->E,reply

Description: Request equipment to create a Process Job with the given PRJOBID. If multi-block, this message must be preceded by the S16,F1/F2 transaction.

Structure:

```
L,7
  1. <DATAID>
  2. <PRJOBID>
  3. <MF>
  4a. L,n                    [MF = carrier, n = # of carriers]
        1. L,2
             1. <CARRIERID1>
             2. L,j          [j = # of slots, may be implemented as an
                                  array]
                  1. <SLOTID1>
                  2. <SLOTID2>
                  .
                  j. <SLOTIDj>
        .
        .
        n. L,2
             1. <CARRIERIDn>
             2. L,j          [j = # of slots, may be implemented as an
                                  array]
                  1. <SLOTID1>
                  2. <SLOTID2>
                  .
                  j. <SLOTIDj>

  4b. L,n                    [MF = substrate]
        1. <MID1>
        .
        .
        n. <MIDn>
  5.  L,3
        1. <PRRECIPEMETHOD>
        2. <RCPSPEC>
        3. L,m               [m = # recipe parameters]
             1. L,2
                  1. <RCPPARNM1>
                  2. <RCPPARVAL1>
             .
             m. L,2
                  1. <RCPPARNMm>
                  2. <RCPPARVALm>
  6. <PRPROCESSSTART>
  7. <PRPAUSEEVENT>
```

Exception: The list for specifying material (item 4a and 4b) is empty (L,0 instead of L,n), when no material is specified for the process job. The form of data item 4(a or b) depends on the value in MF.

S16,F12 PRJobCreateEnh Acknowledge S,H<-E

Description: This message acknowledges the request and reports any errors in the creation of a process job.

Structure:
```
L,2
  1. <PRJOBID>
  2. L,2
       1. <ACKA>
       2. L,n
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S16,F13 PRJobDuplicateCreate M,H->E,reply

Description: This function creates multiple process jobs. The same recipe and value of PRProcessStart are applied to each process job created. If multi-block, this message must be preceded by the S16,F1/F2 transaction.

Structure:
```
L,5
  1. <DATAID>
  2. L,p                    [p = # of process jobs being created]
       1. L,3
            1. <PRJOBID1>
            2. <MF1>
            3a. L,n         [MF = carrier, n = # of carriers]
                  1. L,2
                       1. <CARRIERID1>
                       2. L,j   [j = # of slots, may be implemented
                                    as an array]
                            1. <SLOTID1>
                            2. <SLOTID2>
                            .
                            j. <SLOTIDj>
                  .
                  .
                  n. L,2
                       1. <CARRIERIDn>
                       2. L,j   [j = # of slots, may be implemented
                                    as an array]
                            1. <SLOTID1>
                            2. <SLOTID2>
                            .
                            j. <SLOTIDj>
            3b. L,n             [MF = substrate, n = # of MID]
                  1. <MID1>
                  .
                  n. <MIDn>
       .
       .
```

```
                    p. L,3
                         1. <PRJOBID_p>
                         2. <MF_p>
                         3a. L,n              [MF = carrier, n = # of carriers]
                               1. L,2
                                    1. <CARRIERID_1>
                                    2. L,j    [j = # of slots, may be implemented
                                                 as an array]
                                         1. <SLOTID_1>
                                         2. <SLOTID_2>
                                         .
                                         j. <SLOTID_j>
                               .
                               .
                               n. L,2
                                    1. <CARRIERID_n>
                                    2. L,j    [j = # of slots, may be implemented
                                                 as an array]
                                         1. <SLOTID_1>
                                         2. <SLOTID_2>
                                         .
                                         j. <SLOTID_j>
                         3b. L,n              [MF = substrate, n = # of MID]
                               1. <MID_1>
                               .
                               n. <MID_n>
               3. L,3
                    1. <PRRECIPEMETHOD>
                    2. <RCPSPEC>
                    3. L,m                    [m = # recipe parameters]
                         1. L,2
                              1. <RCPPARNM_1>
                              2. <RCPPARVAL_1>
                         .
                         m. L,2
                              1. <RCPPARNM_m>
                              2. <RCPPARVAL_m>
               4. <PRPROCESSSTART>
               5. <PRPAUSEEVENT>
```

Exception: The list for specifying material (item 3a and 3b) is empty (L,0 instead of L,n), when no material is specified for the process job. The form of data item 3(a or b) depends on the value in MF.

S16,F14 PRJobDuplicateCreate Acknowledge S,H<-E

Description: This message acknowledges the request and reports any errors in the creation of a process job. ERRTEXT contains the identifier of process jobs that were not created.

Structure:
```
L,2
  1. L,m                          [m = # of jobs created]
       1. <PRJOBID1>
       .
       m. <PRJOBIDm>
  2. L,2
       1. <ACKA>
       2. L,n
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S16,F15 PRJobMultiCreate M,H->E,reply

Description: Use this single message to Create Multiple Process Jobs, each of which may be unique in its association of material to process recipe. If multi-block, this message must be preceded by the S16,F1/F2 transaction.

Structure:
```
L,2
  1. <DATAID>
  2. L,p                          [p = # of process jobs being created]
       1. L,6
            1. <PRJOBID1>
            2. <MF1>
            3a. L,n               [MF = carrier, n = # of carriers]
                  1. L,2
                       1. <CARRIERID1>
                       2. L,j     [j = # of slots, may be implemented
                                      as an array]
                            1. <SLOTID1>
                            2. <SLOTID2>
                            .
                            j. <SLOTIDj>
                  .
                  .
                  n. L,2
                       1. <CARRIERIDn>
                       2. L,j     [j = # of slots, may be implemented
                                      as an array]
                            1. <SLOTID1>
                            2. <SLOTID2>
                            .
                            j. <SLOTIDj>
```

```
      3b. L,n              [MF = substrate, n = # of MID]
            1. <MID1>
            .
            n. <MIDn>
      4. L,3
           1. <PRRECIPEMETHOD1>
           2. <RCPSPEC1>
           3. L,m          [m = # recipe parameters]
                1. L,2
                     1. <RCPPARNM1>
                     2. <RCPPARVAL1>
                .
                m. L,2
                     1. <RCPPARNMm>
                     2. <RCPPARVALm>
      5. <PRPROCESSSTART1>
      6. <PRPAUSEEVENT1>
.
.
p. L,6
      1. <PRJOBIDp>
      2. <MFp>
      3a. L,n              [MF = carrier, n = # of carriers]
            1. L,2
                 1. <CARRIERID1>
                 2. L,j    [j = # of slots, may be implemented
                              as an array]
                      1. <SLOTID1>
                      2. <SLOTID2>
                      .
                      j. <SLOTIDj>
            .
            .
            n. L,2
                 1. <CARRIERIDn>
                 2. L,j    [j = # of slots, may be implemented
                              as an array]
                      1. <SLOTID1>
                      2. <SLOTID2>
                      .
                      j. <SLOTIDj>
      3b. L,n              [MF = substrate, n = # of MID]
            1. <MID1>
            .
            n. <MIDn>
      4. L,3
           1. <PRRECIPEMETHODp>
           2. <RCPSPECp>
           3. L,m          [m = # recipe parameters]
                1. L,2
                     1. <RCPPARNM1>
                     2. <RCPPARVAL1>
                .
                m. L,2
                     1. <RCPPARNMm>
                     2. <RCPPARVALm>
      5. <PRPROCESSSTARTp>
      6. <PRPAUSEEVENTp>
```

Exception: The list for specifying material (item 3a and 3b) is empty (L,0 instead of L,n), when no material is specified for the process job. The form of data item 3(a or b) depends on the value in MF.

S16,F16 PRJobMultiCreate Acknowledge S,H<-E

Description: This message acknowledges the request and reports any errors in the creation of a process job. ERRTEXT contains the identifier of process jobs that were not created.

Structure:
```
L,2
  1. L,m                          [m = # jobs created]
       1. <PRJOBID1>
       .
       m. <PRJOBIDm>
  2. L,2
       1. <ACKA>
       2. L,n
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S16,F17 PRJobDequeue S,H->E,reply

Description: Used to remove process jobs from the equipment for jobs that have not begun processing.

Structure:
```
L,m                               [m = # jobs to remove]
  1. <PRJOBID1>
  .
  m. <PRJOBIDm>
```

Exception: If m = 0, then de-queue all.

S16,F18 PRJobDequeue Acknowledge S,H<-E

Description: Acknowledge the request to de-queue and report any errors. ERRTEXT will contain the identifier of any jobs that were not de-queued.

Structure:
```
L,2
  1. L,m                          [m = # jobs removed]
       1. <PRJOBID1>
       .
       m. <PRJOBIDm>
  2. L,2
       1. <ACKA>
       2. L,n
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S16,F19 PRGetAllJobs S,H->E

Description: Requests the equipment to return a list of process jobs which have not completed. They may be running or waiting to run.

Structure: header only

S16,F20 PRGetAllJobs Send S,H<-E

Description: Returns the requested list of process jobs.

Structure:
```
L,m                               [m = # jobs in the list]
  1. L,2
       1. <PRJOBID1>
       2. <PRSTATE1>
  .
  m. L,2
       1. <PRJOBIDm>
       2. <PRSTATEm>
```

Exception: If m = 0, then no process jobs are running or waiting to run.

S16,F21 PRGetSpace S,H->E

Description: Requests the equipment to return the number of process jobs it has space to create.

Structure: header only

S16,F22 PRGetSpace Send S,H<-E

Description: Sends the host the number of process jobs which can be created.

Structure: <PRJOBSPACE>

S16,F23 PRJobSetRecipeVariable S,H->E

Description: Reset the value of recipe variable parameters for a specific process job.

Structure:
```
L,2
  1. <PRJOBID>
  2. L,m                          [m = # recipe variables]
       1. L,2
            1. <RCPPARNM1>
            2. <RCPPARVAL1>
       .
       .
       m. L,2
            1. <RCPPARNMm>
            2. <RCPPARVALm>
```

S16,F24 PRJobSetRecipeVariable Acknowledge S,H<-E

Description: Indicate the status of the request to set recipe variables. ERRTEXT will contain the RCPPARNM value for parameters that could not be reset.

Structure:
```
L,2
  1. <ACKA>
  2. L,n
       1. L,2
            1. <ERRCODE1>
            2. <ERRTEXT1>
       .
       .
       n. L,2
            1. <ERRCODEn>
            2. <ERRTEXTn>
```

Exception: If n = 0, no errors exist.

S16,F25 PRJobSetStartMethod S,H->E

Description: Used to request to change the start method (USERSTART or AUTO) for one or more process jobs.

Structure:
```
L,2
  1. L,m                          [m = # of jobs]
       1. <PRJOBID1>
       .
       .
       m. <PRJOBIDm>
  2. <PRPROCESSSTART>
```

S16,F26 PRJobSetStartMethod Acknowledge S,H<-E

Description: Acknowledges request to set job start method and indicates any errors. ERRTEXT will contain the identifiers of any process jobs that did not accept the new process start method.

Structure:
```
L,2
  1. L,m                          [m = # of jobs]
       1. <PRJOBID1>
       .
       m. <PRJOBIDm>
  L,2
       1. <ACKA>
       2. L,n
            1. L,2
                 1. <ERRCODE1>
                 2. <ERRTEXT1>
            .
            .
            n. L,2
                 1. <ERRCODEn>
                 2. <ERRTEXTn>
```

S16,F27 Control Job Command Request S,H->E

Description: Send a control job command to a control job.

Structure:
```
L,3
  1. <CTLJOBID>
  2. <CTLJOBCMD>
  3. L,2
       1. <CPNAME>
       2. <CPVAL>
```

Exception: 3. L,2 IS L,0 for commands that do not need parameters.

S16,F28 Control Job Command Acknowledge S,H<-E

Description: Indicates success or failure of command request to a control job. If applicable ERRTEXT shall contain information on specific command parameter names or values that caused the error.

Structure:
```
L,2
  1. <ACKA>
  2. L,2
       1. <ERRCODE>
       2. <ERRTEXT>
```

Exception: 2. L,2 IS L,0 if no errors.

7.21 *Stream 17 Equipment Control and Diagnostics* — This stream is a continuation of Stream 2.

S17,F0 Abort Transaction (S17F0) S,H<->E

Description: Same form as S1F0.

S17,F1 Data Report Create Request (DRC) M,H->E,reply

Description: Create a Data Report definition. This function allows the referencing of a Data Source for the items (variables or attributes) specified in the data report.

Structure:
```
L,4
  1. <DATAID>
  2. <RPTID>
  3. <DATASRC>
  4. L,n
       1. <VID1>
       2. <VID2>
       .
       .
       .
       n. <VIDn>
```

Exception: DATAID is a zero length item when the request can be sent in a single block. If RPTID is a zero length item, then the equipment shall return a value in RPTID by which the host can then reference the report efinition. If RPTID has a value, then the equipment shall retain this value by which the host can then reference the report definition.

S17,F2 Data Report Create Acknowledge (DRCA) S,H<-E

Description: Equipment confirms creation of a Data Report and returns RPTID.

Structure:
```
L,2
  1. <RPTID>
  2. <ERRCODE>
```

Exception: If ERRCODE is a zero length item, then no error occurred.

S17,F3 Data Report Delete Request (DRD) S,H->E,reply

Description: Delete one or more data reports. This shall cause those reports to be unlinked from any Event Reports to which they were linked. This shall cause the report to be excluded from any Trace Reports for in which it had originally been included.

Structure:
```
L,n
  1. <RPTID1>
  2. <RPTID2>
  .
  .
  .
  n. <RPTIDn>
```

Exception: If this message is sent with a zero length list, then all reports shall be deleted.

S17,F4 Data Report Delete Acknowledge (DRDA) S,H<-E

Description: Equipment confirms or indicates any errors on the request to delete Data Reports. All Data Reports which could be deleted shall be listed in the response and the associated error code shall be included in the list.

Structure:
```
L,2
  1. <ACKA>
  2. L,m
       1. L,3
            1. <RPTID1>
            2. <ERRCODE1>
            3. <ERRTEXT1>
       .
       .
       m. L,3
            1. <RPTIDm>
            2. <ERRCODEm>
            3. <ERRTEXTm>
```

Exception: If ACKA is TRUE, then no errors were encountered, meaning all report requests were completed successfully and a zero-length list (m = 0) shall be sent.

Exception: If some reports could not be deleted, then their RPTID's shall be given in a space separated list in ERRTEXT.

S17,F5 Trace Create Request (TRC) M,H->E,reply

Description: Establish a Trace Report definition.

Structure:
```
L,6
  1. <DATAID>
  2. <TRID>
  3. <CEED>
  4. L,n
       1. <RPTID1>
       2. <RPTID2>
       .
       .
       n. <RPTIDn>
  5. <TRSPER>
  6. L,m                (m = {0,8})
       1. <TOTSMP>
       2. <REPGSZ>
       3. <EVNTSRC>     (Defines source for start Event)
       4. <CEID>        (Defines ID of the start Event)
       5. <EVNTSRC>     (Defines source for stop Event)
       6. <CEID>        (Defines ID of the stop Event)
       7. <TRAUTOD>
       8. <RPTOC>
```

Exception: The list *m* can be zero-length, or it can contain all eight data items. Since specifying values for each item is optional, each of the eight items can be zero-length. If the item is zero-length, the format of the item shall be the same format used in other instances of the S17,F5 message where the value is not zero-length.

S17,F6 Trace Create Acknowledge (TRCA) S,H<-E

Description: Equipment confirms creation of an Event Report and returns a TRID.

Structure: L,2
1. <TRID>
2. <ERRCODE>

Exception: If ERRCODE is a zero length item, then no error occurred.

S17,F7 Trace Delete Request (TRD) S,H->E,reply

Description: The host requests to delete one or more Trace Reports.

Structure: L,n
1. <$TRID_1$>
2. <$TRID_2$>
.
.
n. <$TRID_n$>

S17,F8 Trace Delete Acknowledge (TRDA) S,H<-E

Description: This message is required to inform the host when a Trace Report could not be deleted. This message does not need to be sent to confirm the successful deletion of a Trace Report. If the report is sent for a successfully deleted Trace Report, then the ERRCODE item length shall be set to zero.

Structure:
```
L,2
  1. <ACKA>
  2. L,m
       1. L,3
            1. <TRID1>
            2. <ERRCODE1>
            3. <ERRTEXT1>
       .
       .
       m. L,3
            1. <TRIDm>
            2. <ERRCODEm>
            3. <ERRTEXTm>
```

Exception: If ACKA is TRUE, then no errors were encountered, meaning all report requests were completed successfully and a zero-length list (m = 0) shall be sent.

Exception: If some reports could not be deleted, then their TRID's shall be provided in a space separated list in ERRTEXT.

S17,F9 Collection Event Link Request (CELR) M,H->E,reply

Description: Establish a Collection Event Report definition with respect to a specific Event Source.

Structure:
```
L,4
  1. <DATAID>
  2. <EVNTSRC>
  3. <CEID>
  4. L,n                (n is the number of Reports to be linked)
       1. <RPTID1>
       2. <RPTID2>
       .
       .
       n. <RPTIDn>
```

S17,F10 Collection Event Link Acknowledge (CELA) S,H<-E

Description: Indicate the success or failure of a Collection Event Link Request.

Structure:
```
L,3
  1. <EVNTSRC>
  2. <CEID>
  3. <ERRCODE>
```

Exception: Item 3 should be set to zero length to indicate success.

S17,F11 Collection Event Unlink Request (CEUR) S,H->E,reply

Description: Request to unlink a specific Data Report from a Collection Event Report.

Structure: L,3
1. <EVNTSRC>
2. <CEID>
3. <RPTID>

Exception: Item one can be zero length, in which case the default event source is assumed.

S17,F12 Collection Event Unlink Acknowledge (CEUA) S,H<-E

Description: Indicates success or failure of a requested Unlink.

Structure: L,4
1. <EVNTSRC>
2. <CEID>
3. <RPTID>
4. <ERRCODE>

Exception: Item one can be zero length to indicate the default event source. Item 4 is set to zero length if the primary request was successful.

S17,F13 Trace Reset Request (TRR) S,H->E,reply

Description: The Host requests the equipment to clear the data and reset the specified trace reports. If n = 0, then all defined Trace Objects will be reset.

Structure: L,n
1. <$TRID_1$>
2. <$TRID_2$>
.
.
n. <$TRID_n$>

S17,F14 Trace Report Reset Acknowledge (TRRA) S,H<-E

Description: This list in item 1 contains the identifiers of all the Trace Objects which were reset. If all Trace Objects are successfully reset, then ACKA shall be set to TRUE.

Structure:
```
L,2
  1. <ACKA>
  2. L,m
       1. L,3
            1. <TRID1>
            2. <ERRCODE1>
            3. <ERRTEXT1>
       .
       .
       m. L,3
            1. <TRIDm>
            2. <ERRCODEm>
            3. <ERRTEXTm>
```

Exception: If ACKA is TRUE, then no errors were encountered, meaning all report requests were completed successfully and a zero-length list (m = 0) shall be sent.

Exception: If some reports could not be reset, then their TRID's shall be given in a space separated list in ERRTEXT.

7.22 *Stream 18 Subsystem Control and Data* — Messages exchanged between component subsystems and higher level controllers. Compared to similar messages exchanged between equipment and host, subsystem messages are less complex.

S18,F1 Read Attribute Request (RAR) S,H->E,reply

Description: This message requests the current values of specified attributes of the subsystem component indicated in TARGETID.

Structure:
```
L,2
  1. <TARGETID>
  2. L,n                n = # attribute identifiers
       1. <ATTRID1>
       .
       n. <ATTRIDn>
```

Exceptions: If n = 0, then all attributes of the target component are requested.

S18,F2 Read Attribute Data (RAD) S,H<-E

Description: This message returns the current values of requested attributes and the current status of the requested component indicated in TARGETID. Attributes are returned in the order requested.

Structure:
```
L,4
  1. <TARGETID>
  2. <SSACK>
  3, L,n
       1. <ATTRVAL1>
       .
       n. <ATTRVALn>
  3. L,s
       1. <STATUS1>
       .
       s. <STATUSS>
```

Exceptions: Both n = 0 and s = 0 if the target component is unknown.

S18,F3 Write Attribute Request (WAR) S,H->E,reply

Description: This message requests the subsystem to set the value of read/write attributes of the component specified in TARGETID.

Structure:
```
L,2
  1. <TARGETID>
  2, L,n
       1. L,2
            1. <ATTRID1>
            2. <ATTRVAL1>
       .
       n. L,2
            1. <ATTRIDn>
            2. <ATTRVALn>
```

S18,F4 Write Attribute Acknowledge (WAA) S,H<-E

Description: This message acknowledges the success of failure of the request to write attribute data to the subsystem indicated in TARGETID.

Structure:
```
L, 3
   1. <TARGETID>
   2. <SSACK>
   3. L,s
        1. <STATUS1>
        .
        s. <STATUSS>
```

Exceptions: s = 0 if the target component is unknown.

S18,F5 Read Request (RR) S,H->E,reply

Description: The host requests the subsystem indicated in TARGETID to read information. DATASEG may be used to indicate a specific section of data to be read. DATALENGTH is used to limit the amount of data for that section.

Structure: L,3
1. <TARGETID>
2. <DATASEG>
3. <DATALENGTH>

Exceptions: If DATASEG and DATALENGTH are both omitted (are zero length items) then all data is requested. If DATALENGTH only is omitted, then all data within the indicated section is requested.

S18,F6 Read Data (RD) S,H<-E

Description: This message is used to return requested information from the subsystem indicated in TARGETID or to acknowledge the results of the request.

Structure: L,3
1. <TARGETID>
2. <SSACK>
3. <DATA>

Exceptions: If TARGETID is unknown, then DATA is zero length.

S18,F7 Write Data Request (WDR) S,H->E,reply

Description: This message requests to write data to the subsystem component indicated in TARGETID. DATASEG may be used to indicate a specific section of data to be written or overwritten.

Structure: L,4
1. <TARGETID>
2. <DATASEG>
3. <DATALENGTH>
4. <DATA>

Exceptions: If DATASEG and DATALENGTH are both omitted (are zero length items) then all data is to be overwritten. If only DATALENGTH is omitted or if DATALENGTH has a value of zero, then all data within the indicated section is to be written.

S18,F8 Write Data Acknowledge (WDA) S,H<-E

Description: This message acknowledges the success or failure of writing data to the subsystem indicated in TARGETID.

Structure:
```
L,3
  1. <TARGETID>
  2.  <SSACK>
  3. L,s
       1. <STATUS1>
       .
       s. <STATUSS>
```

Exceptions: s = 0 if and only if TARGETID is unknown.

S18,F9 Read ID Request (RIR) S,H->E,reply

Description: This message is used to request the subsystem indicated by TARGETID to read an identifier.

Structure: <TARGETID>

Exceptions: None.

S18,F10 Read ID Data (RID) S,H<-E

Description: This message returns a requested material identifier MID as read by the subsystem indicated in TARGETID.

Structure:
```
L,4
  1. <TARGETID>
  2. <SSACK>
  3. <MID>
  4. L,s
       1. <STATUS1>
       .
       s. <STATUSS>
```

Exceptions: s = 0 if and only if TARGETID is unknown.

S18,F11 Write ID Request (WIR) S,H->E,reply

Description: This message is used to request the subsystem indicated by TARGETID to write an identifier.

Structure:
```
L,2
  1. <TARGETID>
  2. <MID>
```

Exceptions: None.

S18,F12 Write ID Acknowledge (WIA) S,H<-E

```
Description:   This message acknowledges the success or failure of the subsystem
               specified in TARGETID in writing the ID.

Structure:     L,3
                 1. <TARGETID>
                 2. <SSACK>
                 3. L,s
                      1. <STATUS1>
                      .
                      s. <STATUSS>

Exceptions:    s = 0 if and only if TARGETID is unknown.
```

S18,F13 Subsystem Command Request (SCR) S,H->E,reply

```
Description:   This message is used to request the subsystem indicated in TARGETID
               to perform a specific action.

Structure:     L,3
                 1. <TARGETID>
                 2. <SSCMD>
                 3.  L,n
                       1. <CPVAL1>
                       .
                       n.  <CPVALn>

Exceptions:    If n = 0, no parameters are provided.
```

S18,F14 Subsystem Command Acknowledge (SCA) S,H<-E

```
Description:   This message reports the results from the subsystem specified in
               TARGETID for the requested action.

Structure:     L,3
                 1. <TARGETID>
                 2. <SSACK>
                 3. L,s
                      1. <STATUS1>
                      .
                      s. <STATUSS>

Exceptions:    s = 0 if and only if TARGETID is unknown.
```

8 Message Documentation

8.1 *Intent* — Equipment makers using SECS-II messages must communicate the equipment-specific details of each message to the host designer in order for the host to properly adapt to the equipment. The details are communicated in a document which will follow a standard form in order to convey most clearly the information required. The following form is presented here to act as a guide for organizing the equipment-specific details.

8.2 *Standard Form SECS-II Document* — The standard form will contain three clearly labeled parts as follows.

Part I — General Information

Part II — Message Summary

Part III — Message Detail

8.2.1 Part I will contain general information on the following:

Manufacturer and product number

General description of equipment function

Intended function of interface

Software revision code

Changes from previous versions

8.2.2 Part II will contain two lists of all messages understood and all messages sent by the equipment in terms of their stream and function codes. The first list will have pairs of columns: the first for the message received and understood and the second for the message sent in response. The second list will also have two columns: the first for the message sent and the second for the response understood. The message will be identified using the format "SxxFyy," where xx is the stream number and yy is the function number. Each transaction will be on a separate line. A "-" will indicate that one of a pair is not included. All messages not listed on the received side are implied to cause an error message to the host. All messages not listed on the sent side are assumed never to be sent from the equipment. Since some messages can be sent in two directions, the same message pair may appear in each list with the sent and received orders interchanged. A transaction listed in the standard as being allowed in two directions does not have to be implemented in both directions. This list will indicate which directions are implemented.

8.2.3 Part III will contain the details for every message listed in Part II. Messages that appear on both the sent and received sides must be detailed separately. The details shall include the following information on the data in each message:

1. For each fixed item, all the values or strings either understood or possible to send are listed along with their meaning to the equipment.
2. For each variable item, the restriction on or possible range of value or length of string.
3. Any other special interpretation of the message.

8.2.4 Each message so detailed must be clearly labeled with its stream and function code.

9 Units of Measure

9.1 *Intent* — Certain SECS-II transactions require specification of units of measure for data items passed between equipment and host. The concept of units of measure has been included as part of the SECS-II standard to enhance the ability of the host system to prompt its human operators for proper information when generating process programs, and also to facilitate automated handling of process programs by host systems and automated handling of data reported to a host by equipment.

9.2 *Units Symbols* — Under SECS-II, a units symbol is a text string of unspecified length which specifies the physical significance of a numeric value. Units of measure symbols under SECS-II may be either a SECS-II recognized unit identifier, a SECS-II unit identifier with prefix and/or suffix symbols, or an arithmetic expression of SECS-II identifiers.

9.2.1 A SECS-II units identifier is a text string which may be the full name, an abbreviation of the full name, or a special character which is unique for a specific unit of weight or measure. Identifier strings may consist of upper or lower case alphabetic characters and numerals or special characters of the ASCII character set. The first character of an identifier may not be a numeral. The case of alphabetic characters is significant (e.g., G and g, the units symbols for Gauss and gram, respectively).

9.2.2 A unit identifier may be nationally or internationally recognized, may be unique to the semiconductor industry, or, due to the special requirements of SECS-II, may be unique to this standard. Section 9.4 lists all units identifiers recognized by SECS-II. For each identifier defined in Section 9.4, six pieces of information are provided. They are:

1. Unit — Full name of the unit of measure in question.
2. Unit Identifier — SECS-II-recognized identifier for the unit.
3. Prefix Allowed — Specifies whether or not the unit identifier may be combined with a prefix symbol to generate a unit identifier which is a decimal multiple or submultiple of the base unit. Metric (or SI) units are usually capable of accepting a prefix symbol while English units may not.
4. Suffix Allowed — Specifies whether or not the unit identifier may be concatenated with a numeric suffix which provides additional information to the meaning of the associated unit symbol. The numeric suffix is composed of the ASCII digits 0 through 9 and represents a decimal value. This meaning of the numeric value is symbol-dependent and must be specified in the description section of the unit symbol's definition.
5. Equivalence — In those cases where a unit can be expressed as an arithmetic expression (of simpler units), this column will contain the expression of simpler units. For those units which are non-standard to either of the standard systems of units of measure (English or SI), this column will contain an

expression which relates the non-standard unit to the equivalent unit of the standard units system. In either case, the expression provided in this column may be substituted for the corresponding SECS-II units identifier whenever required.

6. Description — Additional information as may be required to uniquely define the unit of measure in question.

9.2.3 Any SECS-II identifier which Section 9.4 indicates as being capable of taking on a prefix symbol may be appended to one of the prefix symbols shown in Table 2, forming a new unit which is a decimal multiple or submultiple of the base unit. A prefix symbol may not be used alone. It must appear concatenated to one of the identifiers in Section 9.4. Finally, only one prefix symbol may appear before any identifier. A units symbol such as "mus" (micromillisecond) is not allowed. The proper symbol is "ns" (nanosecond).

Prefix Name	*Multiplicative Factor*	*Prefix Symbol*
exa	10^18	E
peta	10^15	P
Tera	10^12	T
giga	10^9	G
mega	10^6	M
kilo	10^3	k
hecto	10^2	h
deka (deca)	10^1	da
deci	10^-1	d
centi	10^-2	c
milli	10^-3	m
micro	10^-6	u
nano	10^-9	n
pico	10^-12	p
femto	10^-15	f
atto	10^-18	a

9.2.4 Any SECS-II identifier which Section 9.4 indicates as being capable of taking on a suffix value may have a numeric string appended to it. This decimal value allows the user to identify one of a family of symbol names with only the generic symbol name of the family being defined in Section 9.4. The meaning of a numeric suffix is dependent on the particular symbol with which it is being used and must be defined in the description section of the symbol definition.

9.2.5 Arithmetic expressions of units of measure identifiers are recognized by SECS-II as units symbols if they are formed by the following rules:

1. All units identifiers in the expression are SECS-II units identifiers defined in Section 9.4 or SECS-II prefixed units identifiers as defined above.
2. Exponentiation is denoted by a circumflex (^) between the identifier to be operated on and the exponent. Exponents may be positive or negative values. A negative value is denoted by a unary minus sign (-) between the circumflex and exponent. For positive values, the exponent will immediately follow the circumflex (A^2 or A^-2).
3. Multiplication of units identifiers is expressed by an asterisk (*) positioned between the factors to be multiplied (A*B).
4. Division of units identifiers is expressed by a slash (/) positioned between the dividend and divisor. Division may also be expressed as the product of the dividend and the divisor with a negative exponent (A/B or A*B^-1).
5. Parentheses may be used to specify the order in which the arithmetic operations will be performed.
6. Within expressions or sub-expressions where parentheses do not specify the order of operations, exponentiation will be carried out first, followed by left-to-right evaluation of all multiplication and division that is (A*B^-2*30*C^2) is equivalent to (((A/(B^2))*30)*(C^2)).

9.3 *Compliance* — For the units of measure information to have any value and to be in compliance with SECS-II, equipment and host system manufacturers must ensure that only units symbols allowed by SECS-II are used by their systems. In those instances where SECS-II does not provide a units symbol required for a particular application, the manufacturer requiring the new symbol may submit a proposal to the SEMI Communications Subcommittee requesting the enhancement. A proposal must include all the information provided by each entry of Section 9.4 as described above.

9.3.1 A proposal must undergo the full approval cycle as prescribed by SECS-II for amending a standard (acceptance by committee, balloting, etc.). As a result, the proposal should be submitted as soon as possible, so that sufficient time is available to complete the standard amendment process and to notify all interested parties of the change before the product requiring the new symbol becomes available for use in a manufacturing facility.

9.4 *SECS-II Units of Measure Identifiers* — All units of measure symbols recognized by SECS-II are defined in this section or are compound symbols based on the identifiers defined here and formed by the rules specified in Section 9.2. Portions of the information provided below have been obtained from ANSI/IEEE 260-1978, ANSI X3.5-1976, ISO 2955-1974(E), Webster's New World Collegiate Dictionary (copyright 1977), and the

CRC Handbook of Chemistry and Physics (52nd edition for 1971-1972).

Unit — Non-dimensional quantities (pure numbers)
Unit Identifier — null string
Prefix Allowed — No
Equivalence — None
Suffix Allowed — No
Description — For all quantities which have no associated unit of measure, a zero length (null) text string is the appropriate 'identifier' to use when units of measure information is required.

Unit — ampere
Prefix Allowed — Yes
Unit Identifier — A
Suffix Allowed — No
Equivalence — None
Description — SI unit of electric current.

Unit — ampere (turn)
Prefix Allowed — Yes
Unit Identifier — AT
Suffix Allowed — No
Equivalence — None
Description — SI unit of magnetomotive force.

Unit — angstrom
Unit Identifier — Ang
Prefix Allowed — Yes
Equivalence — m*10^-10
Suffix Allowed — No
Description — Unit of length used when measuring wavelength of light.

Unit — atmosphere, standard
Unit Identifier — atm
Prefix Allowed — No
Equivalence — 101325*Pa
Suffix Allowed — No
Description — A unit of pressure.

Unit — atmosphere, technical
Unit Identifier — at
Prefix Allowed — No
Equivalence — kgf/cm^2
Suffix Allowed — No
Description — A unit of pressure.

Unit — atomic mass unit (unified)
Unit Identifier — u
Prefix Allowed — No
Equivalence — 1.660531*10^-27*kg
Suffix Allowed — No
Description — One twelfth the mass of an atom of carbon 12 nuclide.

Unit — bar
Unit Identifier — bar
Prefix Allowed — Yes
Equivalence — 100*kPa
Suffix Allowed — No
Description — CGS unit of pressure.

Unit — barn
Unit Identifier — barn
Prefix Allowed — Yes
Equivalence — 10^-28*m^2
Suffix Allowed — No
Description — Unit for measuring capture cross sections of elements.

Unit — barrel (petroleum)
Unit Identifier — bbl
Prefix Allowed — No
Equivalence — 42*gal or 158.99*1
Suffix Allowed — No
Description — A unit of volume.

Unit — baud
Unit Identifier — Bd Prefix Allowed — Yes
Equivalence — bit/s Suffix Allowed — No
Description — Telecommunications measure of data transfer rate equivalent to one bit of information transferred per second.

Unit — bel
Unit Identifier — B Prefix Allowed — Yes
Equivalence — None Suffix Allowed — No
Description — The logarithm of the ratio of two power signals.

Unit — Becquerel
Unit Identifier — Bq Prefix Allowed — Yes
Equivalence — None Suffix Allowed — No
Description — SI unit of activity of a radionuclide.

Unit — bit
Unit Identifier — bit Prefix Allowed — Yes
Equivalence — None Suffix Allowed — No
Description — A unit of computer information equivalent to the choice between two alternatives (as yes or no, on or off).

Unit — boat
Unit Identifier — boat Prefix Allowed — No
Equivalence — None Suffix Allowed — Yes
Description — Special SECS generic unit corresponding to a holder of wafers or packages with discrete positions. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.

Unit — British thermal unit
Unit Identifier — Btu Prefix Allowed — No
Equivalence — 1054.35*J Suffix Allowed — No
Description — The quantity of heat required to raise the temperature of one pound of water one degree Fahrenheit at or near 39.2˚F.

Unit — byte
Unit Identifier — byte Prefix Allowed — Yes
Equivalence — 8*bit Suffix Allowed — No
Description — Unit of storage for computer memory.

Unit — calorie (International Table)
Unit Identifier — callIT Prefix Allowed — Yes
Equivalence — 4.1868*J Suffix Allowed — No
Description — Defined by the 1929 International Stream Table Conference to be 1/860 international joules or l/859.858 joules.

Unit — calorie (thermochemical)
Unit Identifier — cal Prefix Allowed — Yes
Equivalence — 4.l840*J Suffix Allowed — No
Description — Unit of energy defined by the NBS to be 4.184 joules. Also, called the gram calorie.

Unit — candela
Unit Identifier — cd Prefix Allowed — Yes
Equivalence — None Suffix Allowed — No
Description — SI unit of luminous intensity.

Unit — candle
Unit Identifier — cd
Equivalence — None
Description — Alternate name for candela.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — carrier
Unit Identifier — carrier
Equivalence — None
Description — Special SECS generic unit corresponding to a holder for substrates, wafers or wafer frames. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.
Prefix Allowed — No
Suffix Allowed — Yes

Unit — cassette
Unit Identifier — css
Equivalence — None
Description — Special SECS generic unit corresponding to a holder for wafers or wafer frames. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.
Prefix Allowed — No
Suffix Allowed — Yes

Unit — Coulomb
Unit Identifier — C
Equivalence — A*s
Description — SI unit of electric charge.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — curie
Unit Identifier — Ci
Equivalence — 3.7*l0^l0*Bq
Description — A unit of activity of radionuclide.
Prefix Allowed — No
Suffix Allowed — No

Unit — cycle
Unit Identifier — c
Equivalence — None
Description — Unit equivalent to one complete performance of a periodic process.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — darcy
Unit Identifier — D
Equivalence — cP*(cm/s)(cm/atm) or 0.986923*um^2
Description — Unit of permeability of a porous medium. By traditional definition, a permeability of one darcy will permit a flow of 1 cm^3/s of fluid of 1 cP viscosity through an area of 1 cm^2 under a pressure gradient of 1 atm/cm.
Prefix Allowed — No
Suffix Allowed — No

Unit — day (mean solar)
Unit Identifier — d
Equivalence — 24*h
Description — The period required for the Earth to complete one rotation about its axis.
Prefix Allowed — No
Suffix Allowed — No

Unit — degree (plane angle)
Unit Identifier — deg
Equivalence — pi/180*rad
Description — One three hundred sixtieth part of the circumference of a circle.
Prefix Allowed — No
Suffix Allowed — No

Unit — degree Celsius
Unit Identifier — degC
Equivalence — None
Prefix Allowed — No
Suffix Allowed — No

Description — Unit of temperature where 0˚C corresponds to the freezing point of water and l00˚C corresponds to the boiling point at standard atmospheric conditions.

Unit — degree Fahrenheit
Unit Identifier — degF
Prefix Allowed — No
Equivalence — None
Suffix Allowed — No
Description — Unit of temperature where 32˚F corresponds to the freezing point of water and 212˚F corresponds to the boiling point at standard atmospheric conditions.

Unit — degree Kelvin
Unit Identifier — K
Prefix Allowed — No
Equivalence — None
Suffix Allowed — No
Description — SI unit of temperature.

Unit — die
Unit Identifier — die
Prefix Allowed — No
Equivalence — None
Suffix Allowed — No
Description — Special SECS generic unit corresponding to an individual integrated circuit both on a wafer and after wafer separation. Also referred to as a bar or chip.

Unit — dyne
Unit Identifier — dyn
Prefix Allowed — Yes
Equivalence — 10^-5*N
Suffix Allowed — No
Description — Unit of force in the cgs system. One dyne is the force required to provide a one grain mass with an acceleration of 1 cm/s^2.

Unit — electronvolt
Unit Identifier — eV
Prefix Allowed — Yes
Equivalence — l.60209*10^-19*J
Suffix Allowed — No
Description — Energy acquired by a small particle carrying a unit electronic charge when it falls through a potential difference of one volt.

Unit — erg
Unit Identifier — erg
Prefix Allowed — Yes
Equivalence — l0^-7*J
Suffix Allowed — No
Description — Unit of work or energy in the cgs system. One erg is equal to the work done or energy expended to exert a force of one dyne through a distance of 1 cm.

Unit — farad
Unit Identifier — F
Prefix Allowed — Yes
Equivalence — A*s/V
Suffix Allowed — No
Description — SI unit of capacitance.

Unit — foot
Unit Identifier — ft
Prefix Allowed — No
Equivalence — 12*in
Suffix Allowed — No
Description — English unit of length.

Unit — footcandle
Unit Identifier — Fc
Prefix Allowed — No
Equivalence — lm/ft^2
Suffix Allowed — No
Description — Unit of illuminance. Also called lumen per square foot.

Unit — footlambert
Unit Identifier — FL
Prefix Allowed — No
Equivalence — (1/pi)*cd/ft^2
Suffix Allowed — No
Description — A unit of luminance. One lumen per square foot leaves a surface whose luminance is one footlambert in all directions within a hemisphere.

Unit — gal
Unit Identifier — Gal
Prefix Allowed — Yes
Equivalence — cm/s^2
Suffix Allowed — No
Description — A unit of acceleration used especially for values of gravity.

Unit — gallon (US)
Unit Identifier — gal
Prefix Allowed — No
Equivalence — 23l*in^3 or 4*qt or 3.7854*1
Suffix Allowed — No
Description — United States version of English system unit of volume.

Unit — gallon (UK)
Unit Identifier — galUK
Prefix Allowed — No
Equivalence — 4.546l*l
Suffix Allowed — No
Description — United Kingdom version of English system unit of volume.

Unit — gauss
Unit Identifier — G
Prefix Allowed — Yes
Equivalence — Mx/cm^2
Suffix Allowed — No
Description — Electromagnetic CGS unit of magnetic flux density.

Unit — gilbert
Unit Identifier — Gb
Prefix Allowed — Yes
Equivalence — l0/(4*pi)*AT
Suffix Allowed — No
Description — Electromagnetic CGS unit of magnetomotive force.

Unit — grain
Unit Identifier — gr
Prefix Allowed — No
Equivalence — .0022857l43*oz
Suffix Allowed — No
Description — English unit of weight.

Unit — gram
Unit Identifier — g
Prefix Allowed — Yes
Equivalence — None
Suffix Allowed — No
Description — One thousandth of the SI unit of mass.

Unit — gram-force
Unit Identifier — gf
Prefix Allowed — Yes
Equivalence — 9.80665*N*10^-3
Suffix Allowed — No
Description — The weight of a gram mass when subjected to the mean gravitational attraction of the Earth.

Unit — gray
Unit Identifier — Gy
Prefix Allowed — Yes
Equivalence — Unknown
Suffix Allowed — No
Description — SI unit of absorbed dose in the field of radiation dosimetry.

Unit — henry
Unit Identifier — H
Equivalence — V*s/A
Description — SI unit of inductance.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — hertz
Unit Identifier — Hz
Equivalence — c/s
Description — SI unit of frequency.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — horsepower (electric)
Unit Identifier — hp
Equivalence — 746*W
Description — Archaic unit of power.
Prefix Allowed — No
Suffix Allowed — No

Unit — hour
Unit Identifier — h
Equivalence — 60*min
Description — Derived unit of time.
Prefix Allowed — No
Suffix Allowed — No

Unit — inch
Unit Identifier — in
Equivalence — 2.54*cm
Description — English unit of length.
Prefix Allowed — No
Suffix Allowed — No

Unit — conventional inch of mercury
Unit Identifier — inHg
Equivalence — 3386.4*Pa
Description — Unit equivalent to the pressure required to balance a one inch high column of mercury in a manometer at 32˚F.
Prefix Allowed — No
Suffix Allowed — No

Unit — conventional inch of water
Unit Identifier — inH_2O
Equivalence — 249.09*Pa
Description — Unit equivalent to the pressure required to balance a one inch high column of water in a manometer at 4˚C.
Prefix Allowed — No
Suffix Allowed — No

Unit — ingot
Unit Identifier — ing
Equivalence — None
Description — Special SECS generic unit corresponding to the entity of semiconductor manufacture from which wafers are made.
Prefix Allowed — No
Suffix Allowed — No

Unit — ion
Unit Identifier — ion
Equivalence — None
Description — SECS II unique symbol equivalent to an atom that carries an electric charge as a result of losing or gaining electrons.
Prefix Allowed — No
Suffix Allowed — No

Unit — joule
Unit Identifier — J
Equivalence — N*m
Description — SI unit of energy, work, and quantity of heat.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — kelvin
Unit Identifier — K Prefix Allowed — No
Equivalence — None Suffix Allowed — No
Description — SI unit of temperature. Also referred to as degree Kelvin.

Unit — kilopound force
Unit Identifier — klbf Prefix Allowed — No
Equivalence — l000*lbf Suffix Allowed — No
Description — A multiple of the English unit of force or weight.

Unit — knot
Unit Identifier — kn Prefix Allowed — No
Equivalence — nmi/h Suffix Allowed — No
Description — Unit of velocity expressed in nautical miles per hour.

Unit — lambert
Unit Identifier — L Prefix allowed — Yes
Equivalence — (l/pi)*cd/cm^2 Suffix allowed — No
Description — CGS unit of luminance. One lumen per square centimeter leaves a surface whose luminance is one lambert in all directions within a hemisphere.

Unit — leadframe
Unit Identifier — ldfr Prefix Allowed — No
Equivalence — None Suffix Allowed — Yes
Description — Special SECS generic unit corresponding to a structure for leads which is removed after packaging. The structure may be fixed length or a reel. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.

Unit — liter
Unit Identifier — l Prefix Allowed — Yes
Equivalence — 10^-3*m^3 Suffix Allowed — No
Description — A metric unit of volume.

Unit — lot
Unit Identifier — lot Prefix Allowed — No
Equivalence — None Suffix Allowed — No
Description — Special SECS generic unit corresponding to a grouping of material which is undergoing the same processing operations. The amount of material represented by "1 lot" is situation-dependent.

Unit — lumen
Unit Identifier — lm Prefix Allowed — Yes
Equivalence — cd*sr Suffix Allowed — No
Description — SI unit of luminous flux.

Unit — lux
Unit Identifier — lx Prefix Allowed — Yes
Equivalence — lm/m^2 Suffix Allowed — No
Description — SI unit of illuminance.

Unit — magazine
Unit Identifier — mgz Prefix Allowed — No
Equivalence — None Suffix Allowed — Yes
Description — Special, SECS generic unit corresponding to a holder of fixed length leadframes. The unit capacity is specified by the symbols suffix, if provided. Otherwise, the capacity is situation-dependent.

Unit — maxwell
Unit Identifier — Mx
Equivalence — 10^-8*Wb
Description — Electromagnetic CGS unit of magnetic flux.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — meter
Unit Identifier — m
Equivalence — None
Description — SI unit of length.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — metric ton
Unit Identifier — t
Equivalence — 10^3*kgf
Description — Unit of weight of force.
Prefix Allowed — No
Suffix Allowed — No

Unit — mho
Unit Identifier — mho
Equivalence — S
Description — Previous name for the SI unit siemens.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — micron
Unit Identifier — um
Equivalence — l0^-6*m
Description — Alternate name for a micrometer.
Prefix Allowed — No
Suffix Allowed — No

Unit — conventional micron of mercury
Unit Identifier — umHg
Equivalence — 133.32*Pa*10^-3
Description — Unit of pressure.
Prefix Allowed — No
Suffix Allowed — No

Unit — mil
Unit Identifier — mil
Equivalence — 10^-3*in
Description — English unit of length.
Prefix Allowed — No
Suffix Allowed — No

Unit — mile
Unit Identifier — mi
Equivalence — 5280*ft
Description — English unit of length.
Prefix Allowed — No
Suffix Allowed — No

Unit — conventional millimeter of mercury
Unit Identifier — mmHg
Equivalence — 133.322*Pa
Description — Unit of pressure.
Prefix Allowed — No
Suffix Allowed — No

Unit — millimicron
Unit Identifier — nm
Equivalence — 10^-9*m
Description — Alternate name for nanometer.
Prefix Allowed — No
Suffix Allowed — No

Unit — minute (plane angle)
Unit Identifier — mins
Prefix Allowed — No

Equivalence — deg/60 Suffix Allowed — No
Description — One sixtieth of a degree (plane angle).

Unit — minute(time)
Unit Identifier — min Prefix Allowed — No
Equivalence — 60*s Suffix Allowed — No
Description — Unit of time.

Unit — mole
Unit Identifier — mol Prefix Allowed — No
Equivalence — 6.02252*l0^23 Suffix Allowed — No
Description — SI unit of number of entities within a substance.

Unit — month
Unit Identifier — mo Prefix Allowed — No
Equivalence — None Suffix Allowed — No
Description — Unit of time.

Unit — nautical mile
Unit Identifier — nmi Prefix Allowed — No
Equivalence — 1852*m Suffix Allowed — No
Description — English unit of measurement.

Unit — neper
Unit Identifier — Np Prefix Allowed — Yes
Equivalence — 0.1l51*dB Suffix Allowed — No
Description — Unit for expressing ratios of power levels.

Unit — newton
Unit Identifier — N Prefix Allowed — Yes
Equivalence — kg*m/s^2 Suffix Allowed — No
Description — SI unit of force.

Unit — nit
Unit Identifier — nt Prefix Allowed — Yes
Equivalence — cd/m^2 Suffix Allowed — No
Description — Alternate name for the SI unit of luminance, candela per square meter.

Unit — oersted
Unit Identifier — Oe Prefix Allowed — Yes
Equivalence — 79.577472*A/m Suffix Allowed — No
Description — Electromagnetic CGS unit of magnetic field strength.

Unit — ohm
Unit Identifier — ohm Prefix Allowed — Yes
Equivalence — V/A Suffix Allowed — No
Description — SI unit of resistance.

Unit — ounce (avoirdupois)
Unit Identifier — oz Prefix Allowed — No
Equivalence — lbf/16 Suffix Allowed — No
Description — English unit of weight.

Unit — package
Unit Identifier — pkg
Equivalence — None
Prefix Allowed — No
Suffix Allowed — No
Description — Special SECS generic unit corresponding to an individual entity both as a place for the die to reside and as a completed unit.

Unit — pascal
Unit Identifier — Pa
Equivalence — N/m^2
Prefix Allowed — Yes
Suffix Allowed — No
Description — SI unit of pressure or stress.

Unit — percent
Unit Identifier — %
Equivalence — 1/100
Prefix Allowed — No
Suffix Allowed — No
Description — Ratio of parts per hundred.

Unit — phot
Unit Identifier — ph
Equivalence — lm/cm^2
Prefix Allowed — Yes
Suffix Allowed — No
Description — CGS unit of illuminance.

Unit — pH
Unit Identifier — pH
Equivalence — 1
Prefix Allowed — No
Suffix Allowed — No
Description — Normalized measure of acidity or alkalinity.

Unit — pint (UK)
Unit Identifier — ptUK
Equivalence — 0.56826*1
Prefix Allowed — No
Suffix Allowed — No
Description — United Kingdom version of English unit of capacity.

Unit — pint (US dry)
Unit Identifier — ptUS
Equivalence — 0.55061*1
Prefix Allowed — No
Suffix Allowed — No
Description — United States version of English unit of dry capacity.

Unit — pint (US liquid)
Unit Identifier — pt
Equivalence — 0.47318*1
Prefix Allowed — No
Suffix Allowed — No
Description — United States version of English unit of liquid capacity.

Unit — plate
Unit Identifier — plt
Equivalence — None
Prefix Allowed — No
Suffix Allowed — Yes
Description — Special SECS generic unit corresponding to a temporary fixture used to hold die during assembly operations. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.

Unit — poise
Unit Identifier — P
Equivalence — 36*N*s/m^2, or 36*kg/(m*s)
Prefix Allowed — Yes
Suffix Allowed — No
Description — A CGS unit of viscosity equal to the viscosity of a fluid that would require a shearing force of one dyne to move a square centimeter area of either of two parallel layers of fluid one centimeter apart with a velocity of one centimeter per second relative to the other layer, with the space between the layers being filled with fluid.

Unit — pound
Unit Identifier — lb
Equivalence — 0.0310810*slug
Description — English unit of mass.
Prefix Allowed — No
Suffix Allowed — No

Unit — pound-force
Unit Identifier — lbf
Equivalence — 4.4482217*N
Description — English unit of force or weight.
Prefix Allowed — No
Suffix Allowed — No

Unit — poundal
Unit Identifier — pdl
Equivalence — 0.0310810*lbf
Description — Force required to accelerate a one pound mass at one ft/s^2.
Prefix Allowed — No
Suffix Allowed — No

Unit — parts per million
Unit Identifier — ppm
Equivalence — 1/I0^6
Description — Ratio of parts per million.
Prefix Allowed — No
Suffix Allowed — No

Unit — quart (UK)
Unit Identifier — qtUK
Equivalence — 1.1365*1
Description — United Kingdom version of an English unit of capacity.
Prefix Allowed — No
Suffix Allowed — No

Unit — quart (US dry)
Unit Identifier — qtUS
Equivalence — 1.1012*1
Description — United States version of an English unit of dry capacity.
Prefix Allowed — No
Suffix Allowed — No

Unit — quart (US liquid)
Unit Identifier — qt
Equivalence — 0.94635*1
Description — United States version of an English unit of liquid capacity.
Prefix Allowed — No
Suffix Allowed — No

Unit — rad
Unit Identifier — rd
Equivalence — 10^-2*Gy
Description — A unit of absorbed dose in the field of radiation dosimetry.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — radian
Unit Identifier — rad
Equivalence — None
Description — SI unit of plane angle.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — rem
Unit Identifier — rem
Equivalence — 10^-2*Sv
Description — A unit of dose equivalent in the field of radiation dosimetry.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — revolution
Unit Identifier — r
Equivalence — c
Prefix Allowed — No
Suffix Allowed — No
Description — One complete cycle of a rotating body.

Unit — roentgen
Unit Identifier — R
Equivalence — Unknown
Prefix Allowed — No
Suffix Allowed — No
Description — A unit of exposure in the field of radiation dosimetry.

Unit — second (plane angle)
Unit Identifier — sec
Equivalence — mins/60
Prefix Allowed — No
Suffix Allowed — No
Description — One sixtieth of a minute of a degree.

Unit — second (time)
Unit Identifier — s
Equivalence — None
Prefix Allowed — Yes
Suffix Allowed — No
Description — SI unit of time.

Unit — siemens
Unit Identifier — S
Equivalence — 1/ohm
Prefix Allowed — Yes
Suffix Allowed — No
Description — SI unit of conductance.

Unit — sievert
Unit Identifier — Sv
Equivalence — Unknown
Prefix Allowed — Yes
Suffix Allowed — No
Description — SI unit of dose equivalent in the field of radiation dosimetry.

Unit — slug
Unit Identifier — slug
Equivalence — 145939*kg
Prefix Allowed — No
Suffix Allowed — No
Description — English unit of mass.

Unit — standard cubic centimeter per minute
Unit Identifier — sccm
Equivalence — cc/min
Prefix Allowed — No
Suffix Allowed — No
Description — A unit of flow equivalent to one cubic centimeter of a gas at standard temperature and pressure flowing past a point in one minute.

Unit — standard liter per minute
Unit Identifier — slpm
Equivalence — 1/min
Prefix Allowed — No
Suffix Allowed — No
Description — A unit of flow equivalent to one liter of a gas at standard temperature and pressure flowing past a point in one minute.

Unit — steradian
Unit Identifier — Sr
Equivalence — Unknown
Prefix Allowed — Yes
Suffix Allowed — No
Description — SI unit of solid angle.

Unit — stilb
Unit Identifier — sb
Equivalence — cd/cm^2
Description — A CGS unit of luminance.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — stokes
Unit Identifier — St
Equivalence — P*cm^3/g
Description — A CGS unit of kinematic viscosity.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — substrate
Unit Identifier — substrate
Equivalence — None
Description — Special SECS generic unit corresponding to the entity of material being operated on, processed or fabricated.
Prefix Allowed — No
Suffix Allowed — No

Unit — tesla
Unit Identifier — T
Equivalence — N/(A*m) or Wb/m^2
Description — SI unit of magnetic flux density (magnetic induction).
Prefix Allowed — Yes
Suffix Allowed — No

Unit — therm
Unit Identifier — thm
Equivalence — 10^5*Btu
Description — An English unit of energy.
Prefix Allowed — No
Suffix Allowed — No

Unit — ton (short)
Unit Identifier — ton
Equivalence — 2000*lbf
Description — English unit of weight.
Prefix Allowed — No
Suffix Allowed — No

Unit — torr
Unit Identifier — torr
Equivalence — mmHg
Description — Pressure unit. Alternative name for millimeters of mercury.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — tube
Unit Identifier — tube
Equivalence — None
Description — Special SECS generic unit corresponding to a holder of packages arranged in a flow. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.
Prefix Allowed — No
Suffix Allowed — Yes

Unit — var
Unit Identifier — var
Equivalence — Unknown
Description — SI unit for reactive power.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — volt
Unit Identifier — V
Equivalence — W/A
Description — SI unit of voltage.
Prefix Allowed — Yes
Suffix Allowed — No

Unit — wafer

Unit Identifier — wfr
Prefix Allowed — No
Equivalence — None
Suffix Allowed — No
Description — Special SECS generic unit corresponding to the entity of material on which semiconductor devices are fabricated.

Unit — waferframe
Unit Identifier — wffr
Prefix Allowed — No
Equivalence — None
Suffix Allowed — Yes
Description — Special SECS generic unit corresponding to a temporary fixture for wafers. The unit capacity is specified by the symbol's suffix, if provided. Otherwise, the capacity is situation-dependent.

Unit — watt
Unit Identifier — W
Prefix Allowed — Yes
Equivalence — J/s
Suffix Allowed — No
Description — SI unit of power.

Unit — watthour
Unit Identifier — Wh
Prefix Allowed — Yes
Equivalence — 3600*J
Suffix Allowed — No
Description — Unit of energy.

Unit — weber
Unit Identifier — Wb
Prefix Allowed — Yes
Equivalence — V*s
Suffix Allowed — No
Description — SI unit of magnetic flux.

Unit — year
Unit Identifier — yr
Prefix Allowed — No
Equivalence — None
Suffix Allowed — No
Description — Unit of time.

APPLICATION NOTES

NOTICE: The material contained in these Application Notes is not an official part of SEMI E5 (SECS-II) and is not intended to modify or supersede the official standard. Rather, these notes describe 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.

A1 The General Node Transaction Protocol

A1.1 This application note has been moved to follow SEMI E4 (SECS-I) as Application Note A7.

A2 Some Suggested Message Usage

A2.1 The number of messages implemented and the choice of messages are greatly influenced by the actual function of the equipment. To illustrate which messages might be appropriate, the following suggestions are offered for a variety of different types of equipment capabilities. It is assumed that the minimum message sets S1,F1; F2 and S9,F1; F3; F5; F7 are always implemented.

A2.2 For equipment which makes nondestructive in-process measurements using a fixed measurement procedure, it may be necessary only to implement S6, F9 to send the data according to a fixed format upon measurement. Optional remote control can be added with S2,F21 to start a measurement.

A2.3 If the equipment has a variety of measurement routines, it might be desirable to respond to S1,F5 with S1,F6, which would give the host a brief report of the test being made. The test can be thought of as a process program. Accordingly, S7,F1 and S7,F2 could be used for the host to select the program. The same messages in conjunction with S7,F3 and S7,F4 could load a new test procedure. S7,F19 could be used by the host to find out what tests were available.

A2.4 Some equipment which automatically processes wafers in a batch might make more extensive use of S1,F5 or S1,F3 and might include some error reporting on S5,F1. More sophisticated equipment may include some trace features with S2,F23 and S6,F1 or some control loop tuning by S2,F15.

A2.5 Equipment using in-line wafer movement could utilize Stream 4, S1,F9, and Stream 3 to keep track of wafers.

A2.6 Stream 7,F9 through F19 can be used to manage a local backup of process programs should the host fail for a short while.

A2.7 Microprocessor equipment can benefit from features such as provided by Stream 8 and S2,F1 through F12 which allow managing and servicing the software routines.

A2.8 Equipment, including a CRT, might elect to make it available to the host by including Stream 10 messages.

A2.9 Some equipment, such as functional testers, might have sufficient need to undertake remote file usage such as provided in Stream 13.

A2.10 These brief suggestions serve to illustrate that the final choice of the messages included in a given equipment depends upon its function. The messages can be viewed as interface features in the same way that other parts of the equipment are viewed as processing features or wafer handling features.

A3 Notes on SECS-II Data Transfers

A3.1 *Introduction*

A3.1.1 There are two primary ways to send and ask for data in SECS-II. One of these is to use the trace feature and the other is to use the event reporting method. The purpose of this note is to describe the intended operation of the messages described in the existing standard. Discussion of completeness or need for other reporting methods is left for task force and committee work.

A3.2 *Trace Data Collection and Reporting*

A3.2.1 This method of collecting data is intended for engineering and developmental use rather than routine data collection for production. The features included allow the collection of relatively large amounts of real time data over a finite amount of time. The data is generated at regular time intervals as determined by a timing generator in the equipment. The function of the host is to set up the trace and then to subsequently store the data as it is received from the equipment. It is assumed that some host resident applications will exist to analyze the data either as it is received or at some later time.

A3.2.2 The trace feature will only exist in equipment which implements it.

A3.2.3 The host sets up the trace with the S2,F23-24 transaction. At this time, the host assigns several important parameters. TRID is the trace request ID and is used later when the equipment sends back the data. Every trace data reply includes the TRID corresponding to the request that set up the trace. Several traces can

theoretically be done at the same time if the equipment allows it and the TRID keeps the data for each trace distinct from other trace data. DSPER is the data sample period and is used to indicate how often the specified parameters should be sampled (that is, have their values saved). TOTSMP is the total number of samples to be made. Since TOTSMP is finite and the number of parameters is specified in this transaction, the host can reserve adequate file space for the reported data if required. The REPGSZ is the reporting group size and corresponds to the number of time samples that should be combined into one message prior to transmission. Thus, if it is desired to sample one or two parameters every second but only send those samples to the host once a minute, the reporting group size would be 60. Having the reporting group size parameter allows the host to have some control over how often it may be interrupted to handle the trace data. However, as presently defined in the standard, the trace data is reported as a single block message (S6,F1), which restricts both the number of status variables or the number of samples which can be combined into one message. The equipment may be able to accommodate this in several ways, to be described shortly.

The last element in the trace initialize request is a list of status variable IDs. The trace command only allows tracing variables that have been declared and are known to the equipment as status variables. It is assumed that the equipment will report the variable values in the same order as specified in the trace request. This will allow the host to identify the values returned.

A3.2.4 The trace data send message, S6,F1-2, sends the trace data as a single block message to minimize the overhead in reporting data. The TRID is the first item and identifies the request that asked for the data. The next item is SMPLN, the sample number of the last sample in this message, should more than one sample be combined. The next item is STIME which is the time of the last sample in this message. These three items are followed by the list of values. If five (5) values were requested with a reporting group size of 5, then 25 values would be in this list, each group of 5 in the same order as requested and in the time order sampled. Some flexibility is allowed in how the equipment chooses to report the data to the host when the reporting group size exceeds one block of data. The equipment can send the data when it has a complete block or it can reject the request when it is set up.

A3.3 *Event Driven Data Reporting*

A3.3.1 The second major type of data reporting is initiated by some event in the equipment. Data reporting is often desired after some event such as the completion of a measurement, the completion of a lot, the completion of a wafer, the occurrence of a special event command in the recipe, or some other action which is determined by the equipment. The two aspects involved in event driven data reporting are the control of which events cause data to be sent to the host and the formatting of the data sent to the host.

A3.3.2 It is assumed that a set of events has been established for a particular piece of equipment and that each event can produce a report of some sort. It is further assumed that a set of equipment constants exist in the machine such that they have control over the optional reporting of the events. For example, a boolean constant may exist for each possible reporting event, and when the host sets the constant to a logical 1, the corresponding event will cause a report to be sent to the host, when the host sets the constant to logical 0, the event will not send a report. S2,F15-16 in the equipment constant send transaction can be used to control the event reporting.

A3.3.3 When an event causes data to be sent to the host, there are several possible conversations, depending upon the length of the data and the complexity of the formatting. S6,F3-4 is the basic data transaction, which has a very general format. The parameters provide for an overall name, DATAID, for the type of data; a collection event identification, CEID, should there be more than one event that could generate the same type of data; and a list of data sets. This structure allows reporting such data as the measurements taken on each of the wafers in a lot. The measurements on each wafer make up one data set, and the list of data sets is the whole lot. The collection event would be the lot completion, and the data ID might be film thickness measurements. Other types of organizations are possible, depending upon the type of data being sent. The same type of data might be produced by a different CEID, such as the forced termination of the lot. This collection ID would indicate that the data is incomplete for the lot. Within each data set, each data value is reported as a pair of items, one item being the name of the value and the other being the value.

A3.3.4 Since many simple measuring devices have only a very few types of data sets, an alternative data format is provided in S6,F9-10, which has the same form as S6,F3-4 but does not require the value name in the data set. Instead, the order of the values is fixed format for the particular DSID in that particular equipment.

A3.3.5 When either of the above data messages is long enough to require multiple blocks, it must be preceded by S6,F5-6 to gain permission to send a multiple block message.

A3.3.6 The last data control transaction is S6,F7-8, which is initiated by the host and causes a specified DATAID to be sent to the host. The implementation of this function is highly equipment-dependent. In essence, it is equivalent to the host causing an event that triggers the sending of the data. Since the equipment may be generating the data, the actual data sent depend upon the equipment implementation. The equipment can respond with a zero length item if no data can be sent.

A3.4 *Event Reporting*

A3.4.1 The third major type of data reporting is similar to that described in A3.3 above, with the following enhancements:

a. Contents of data reports are not limited to DVVALs, but may include SVs or even ECVs.

b. Contents of data reports are user programmable.

A3.4.2 It is assumed that the equipment vendor supplies a list of all "events" identified within a particular piece of equipment. A Collection Event Identifier (CEID) must be specified for each of these events. It is further assumed that the vendor supplies a list of all available variables within the machine. This includes Status Variables (SVs) and their identifiers (SVIDs), Equipment constants (ECVs) and their identifiers (ECIDs), and Data Values (DVVALs) and their identifiers(DVNAMEs). Each of the identifiers must be unique. The term VID (Variable Identifier) encompasses all SVIDs, ECIDs, and DVNAMEs. Likewise, the term V (Variable Data) encompasses all SVs, ECVs, and DVVALs.

A3.4.3 Note that a Variable (V) may be a list (format code 0). This provides for referencing a group of related data values with one identifier. Consider the following:

```
VID1=1 zone 1 temperature ID  Format 32
V1   zone 1 temperature value Format 52
.
.
VIDn=n zone n temperature ID  Format 32
Vn   zone n temperature value Format 52
VIDx=x all temperatures ID    Format 32
Vx  L,n                       Format 0
      1. L,2
           1. <VID1>
           2. <V1>
      .
      .
      n. L,2
           1. <VIDn>
           2. <Vn>
```

Any V in a list may also be a list (for nesting).

A3.4.4 In a typical initialization sequence, the host would define all the desired programmable data reports with S2,F33/S2,F34 (Define Report) transactions. Then S2,F35/S2,F36 (Link Report/Event) transactions would be used to define which reports are to be made by the equipment upon specific events (CEIDs). An individual report may be linked to more than one event. At this point the host may request reports with the S6,F15/S6,F16 (Report Request) transactions to obtain initial report data and/or to verify reports as defined and linked. Finally, the desired reports would be enabled by the host with S2,F37/S2,F38 (Enable/Disable Event Report) transactions.

A3.4.5 There are two methods for the equipment to send event reports to the host. S6,F13 includes the Variable Identifier (VID) with each Variable Data item (V). S6,F11 is a shorter form, without the identifiers; some users prefer this form to reduce message size.

A3.4.6 When any message is long enough to require multiple blocks, it must be preceded by an inquire/grant transaction.The DATAID parameter is used only to link the inquire/grant transaction with a multiblock message. This linkage is to alleviate problems in the case of interleaved messages. A unique value for DATAID must be used for each Inquire/Grant/Send/Acknowledge conversation (similar to the use of SYSBYTES in SECS-I). The DATAID parameter should not be used for any other purpose.

A4 Process Programs

A4.1 *Introduction*

A4.1.1 Two forms of process programs are supported by SECS-II: unformatted and formatted. The contents of an unformatted process program conform to no set standard. The format of the program is defined by the vendor of the equipment and probably bears no similarity to the format used by other vendors for their equipment. Because special programming would be required at the host to understand the equipment's unique data format, the process program is most likely generated at the machine and the host is only used as a data repository, saving the foreign data for later retransmission to the equipment. S7,F3 and S7,F6 are the SECS-II messages used to move unformatted process programs between host and equipment.

A4.1.2 Unformatted process programs were the original accepted means for moving processing instructions between host and equipment under SECS-II. However, the inability of a host to generate process programs for its subordinate machines was quickly recognized as a severe problem. As a result, the formatted process program and its associated transactions were added. Five transactions are provided

under SECS-II for handling formatted process programs: S7,F23-24, S7,F25-26 allow movement of process programs between host and equipment; S7,F21-22 originates at a machine and provides a host with the information it needs to generate a process program for that machine; S7,F27-28 allows the equipment to tell the host whether or not the contents of the formatted process program received from the host are valid; and S7,F31-32 provides the host with the ability to ask the equipment to check the validity of a process program without actually downloading the program into the machine for production use.

A4.2 *Normal Sequence of Operations*

A4.2.1 Formatted process programs may be generated at a host or machine. The actions taken to generate one in a machine are left to the equipment manufacturer. If the process program is created at a host, a sequence of operations is assumed.

1. Once the host's process program generator has been invoked and has been told for which machine a process program is to be created, the host editor must obtain a copy of the process capabilities data for that machine. The information may already be available on the host or it may be obtained directly from the machine. In either case, the information originates at the equipment and is obtained using S7,F21-22. (See Section A4.4 for additional information.)

2. With the machine's process capabilities in its possession, the process program editor may proceed with creating the desired process program. At the conclusion of the editing session, the new machine process program will either be saved at the host or sent directly to the machine for storage and/or use.At this point, the process program is known to satisfy a number of constraints, but it is not necessarily completely acceptable to the machine due to interrelationships of the process program data which are too complex to be described in the machine process capabilities data. The host at any time may verify that a process program is truly valid by sending the process program to the machine and asking it to check the process program and tell the host whether or not the process program is, in fact, correct. If not correct, the equipment is expected to provide information on what data in the process program is unacceptable. This action is accomplished through S7,F31-32. This transaction is equivalent to S7,F23-24, with one important exception, the machine is not to do anything with the process program received under S7,F31 except acknowledge that it got the message (S7,F32) and, as soon as it is able, respond with S7,F27, which provides the host with information on the validity of the process program. In this way, a new version of a process program already held by a machine may be checked for validity without affecting the operation of the machine (i.e., a newer version of a particular process program may be checked while an older version is simultaneously being used by the equipment for material processing).

3. At some point, a host resident process program will be required by the equipment for material processing. Transfer of a program may be accomplished in either of two ways. First, the host may initiate transfer by transmitting S7,F23. In this case, immediately upon reception of the message, the equipment is required to respond with S7,F24, which tells the host that the process program arrived and whether or not the process program is accepted for further processing by the equipment. The second means is for the equipment to initiate the transfer by asking for a process program using S7,F25. In this case, the host will send the process program to the equipment or tell the equipment it is unable to satisfy the request. S7,F23 may also be used by a piece of equipment to transfer a process program to its host for archiving. In this case, the host will respond with S7,F24 and an appropriate completion code. Likewise, a host may request a process program transfer from its machine using S7,F25. The machine will respond with S7,F26, which will contain the process program or an error indication.

4. Following reception by the equipment of the process program, it is the machine's responsibility to check the contents of the process program for validity and respond to the host with a S7,F27 message formatted with the appropriate information about the just received process program. To complete the process program exchange transaction, the host will acknowledge the S7,F27 message with S7,F28. What is done with the process program once accepted and checked for validity is dependent on the state of the process equipment.

A4.3 *Equipment Process Capabilities Data*

A4.3.1 The underlying assumption of SECS-II formatted process programs is that processing instructions for equipment can be expressed as sequences of commands with parameters. Commands are integer codes which tell the machine what to do. The parameters of each command are numeric (integer or floating point) values, Boolean values, or text strings which specify how to carry out the particular command. This provides a very flexible structure for building process programs but does not provide the specific information (code values, types and number of parameters, legal parameter values, etc.) required by a host system to generate a process program for a particular piece of equipment. Under SECS-II, this information is provided to a host via the machine's Equipment Process Capabilities Data or PCD.

A4.3.2 A PCD provides three levels of information global data pertaining to the entire process program; definition of each possible command understood by the machine; and definition of each command parameter. Global process program definition data consist of MDLN, SOFTREV,CMDMAX, BYTMAX, and the list of command descriptors.

A4.3.3 MDLN and SOFTREV provide the same data to the host as the equipment's response to the S1,F1 host interrogative, "Are you there?" They are included in the PCD to provide a means of distinguishing between PCDs for different machines and revisions of PCDs for the same piece of equipment. Also, when a process program is generated, the MDLN/SOFTREV values of the PCD are provided in the process program to allow the machine an unambiguous method of determining if the process program was generated from a PCD it understands.

A4.3.4 BYTMAX and CMDMAX are two integer values which allow the equipment to limit the size of the process program which will be generated. BYTMAX specifies the maximum number of bytes a process program may occupy. CMDMAX specifies the maximum number of commands which may appear in the process program. Either value may be zero, which indicates that no maximum limit is being imposed by the equipment.

A4.3.5 The PCD command list identifies (in no particular order) each of the unique operations its associated machine is capable of performing. These operations may correspond to processing operations of the equipment (bake, spin), initialization of equipment components (set beamline controls), definition of data values referenced by later commands (define bond coordinates or inspection points), or even "pseudo-operations," which allow conditional execution of the process program (go to X; if temperature out of range, then go to y; repeat ramping until speed 200; etc.).

A4.3.6 Each command in the PCDlist has a number of data values associated with it which provide the host with the command's personality. These are CCODE, CNAME, RQCMD, BLKDEF, BCDS, IBCDS, NBCDS, ACDS, IACDS, NACDS, and the commands parameter list.

A4.3.7 CCODE defines the unique numeric code which the equipment recognizes as representing the command being defined. CNAME is a text string which hopefully describes the function of the command. The string must be unique for each command since humans generating process programs at the host will use them, and the host process program generator will translate the CNAME to the corresponding CCODE.

A4.3.8 RQCMD. This Boolean value allows equipment to specify whether or not a command must appear at least once within their process program. If true, the command must be used. If RQCMD is set false, the command may or may not be used in the process program at the discretion of the person creating the process program.

A4.3.9 In addition to the information the PCD provides on allowed data content within a process program, it also can provide information to the host on possible interdependencies between the commands. Specifically, through the PCD the host can know such things as: command code A must appear before command code B; command code A must come after command code D; command code A must immediately precede command code X; command code A must not come before command code E; command code A must not come after command code F; and/or command code A must immediately come after command code T. Each of the PCD entries, BCDS (before codes), ACDS (after codes), IBCDS (immediately before codes), IACDS (immediately after codes), NBCDS (not before codes), and NACDS (not after codes), is a SECS item which may contain one or more command codes. Each particular item defines the relation to be satisfied. The elements of the item identify the command codes which are to satisfy the relation with the command being defined. A zero length item indicates that no restrictions apply for that type of checking. For example, if the values of the various fields take on the values shown in Figure A4-1, the host process program editor will assure that the TEST command (code 10) will occur before commands with codes 5, 6, and 8; that it will come after commands with codes 100 and 2; that TEST will not appear after the command with code 20; and that each occurrence of command code 3 will have a TEST command immediately before it, subject to the block checking limitations described elsewhere.

CNAME = TEST

CCODE = 10

BCDS = 5,6,8

IBCDS = 3

NBCDS = none

ACDS = 100.2

IACDS = none

NACDS = 20

Figure A4-1

A4.3.10 Associated with before/after checking is the concept of a block which allows setting of limits on

before/after checking. A block consists of a start block command, a block terminator command and possibly body commands, commands which are included between the start and terminator commands. There are no specific command codes for start, or terminator commands in SECS-II formatted process programs. Instead, being a start block, terminator block, or body command is merely an attribute of each command defined in the PCD. The field BLKDEF defines this attribute for each command. A positive one indicates the command starts a new block. Zero indicates that the command is a body command and neither starts nor terminates a block. A value of negative one indicates that the command is a terminator command.

A4.3.11 Before/after checking for a particular command is performed only with other commands within the same block. To be within the same block, a command must have the same nesting level as the command of interest or the command must be a contained block.

The example data in Figure A4-2 shows six grouping of commands for before/after checking: (A,B',N), (B,C,D',G',M), (D,E,F), (G,H',L), (H,I',K), (I,J). A letter followed by an apostrophe (`) indicates a block which has been collapsed to a command and has the before/after attributes of its start block command. Note that body and terminator commands occur in only one grouping, while block start commands occur in two. Also, note that the outermost block is assumed to begin with the first command of the process program and to end with the last command.

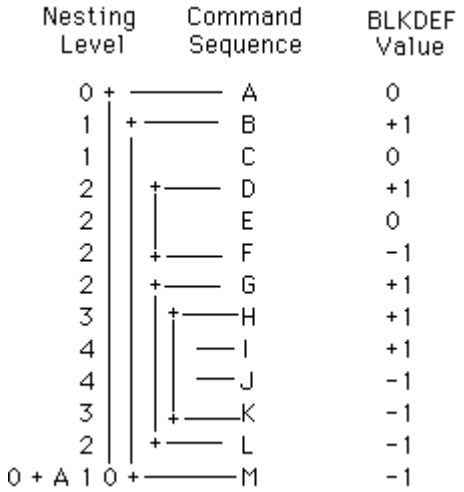


Figure A4-2

A4.3.12 Each command's parameter list defines the parameters required by the equipment to carry out each particular command. The order in which each parameter descriptor appears in the PCD parameter list also defines the order in which parameters will appear in a process program command parameter list. Each parameter is one of three possible types: numeric, text, or Boolean.

A4.3.13 Regardless of parameter type, the first four elements of any parameter descriptor list are the same. The first field, PNAME, specifies the text string which names the parameter. This data will be displayed by a host when prompting a human for the parameter data. The second field, RQPAR, specifies if the value must be specified at the time the process program is generated (true) or if specifying the data is optional (false). The third field, PDFLT, identifies the type of data to be accepted for this parameter as well as providing default values to include in the process program if the RQPAR is false and no data is input for the parameter when the process program is generated. PDFLT will have zero length if no default value is provided.

A4.3.14 The final field, PMAX, specifies the maximum length of the parameter data placed in a process program. For numeric and Boolean data, it specifies the maximum number of data entries in the SECS-II item. For a string parameter, it specifies the maximum number of characters acceptable to the machine. In either case, negative values are invalid and a value of zero indicates there is no length restriction.

A4.3.15 For numeric and Boolean parameters which are multi-valued items, usage of PDFLT becomes a bit more complex. In these cases, PDFLT may also be a vector of values. When default values are to be included in a process program, the entry of the default vector in the same ordinal position as the parameter entry requiring the default is used. If the parameter is allowed to have N entries but only M defaults are provided, the last N-M parameter entries will have no defaults.

A4.3.16 If the numeric or Boolean vector parameter is required to be entered, PDFLT will contain no default data values, but dummy values must be provided so that the length of the item specifies the minimum number of entries the equipment expects to receive for the parameter. If this minimum number of entries exceeds the maximum number of entries allowed for the parameter (PMAX), then only PMAX entries will be provided in the process program.

A4.3.17 Numeric parameters may be any of the SECS recognized floating point or integer data types. PDFLT identifies the particular type. ULIM and LLIM will be of the same data type and specify the range of legal values for the parameter (LLIM≤ x ≤ULIM).UNITS is a character string formed according to E5, Section 9, which specifies the expected units of measure of the

numeric value. RESC specifies whether the resolution of the data item to be entered is to be in terms of a fundamental increment or significant digits. In the case of the former, RESV will be of the same type as the expected parameter and will specify the base increment. In the case of the latter, RESV will be an integer and will specify the number of significant digits to accept for the parameter.

A4.3.18 In addition to the standard fields described above, string parameter descriptors have one unique field. This field provides a set of template strings. A text parameter will be assumed valid by a host process program generator if it matches one of the template strings. A match occurs if the input string is at least as long as the corresponding template and each position of the template and data strings match. A null string specified as a template will result in a match with any data string. A null data string will match only a null template string. A null template list indicates all strings are acceptable to the equipment.

A4.4 *Equipment Capabilities Descriptor Availability*

A4.4.1 Ideally, each piece of equipment should be able to respond to a host PCD request at any time. However, inasmuch as an equipment's PCD may be rather large and the equipment may have limited storage capacity, constant availability may be impossible. In these cases, some compromise will have to be made such as making it available only at machine initialization or when idle. In extremely severe cases, an equipment manufacturer may have to provide the PCD data with the rest of the machine documentation, requiring his customer to manually enter the data into his host system.

A4.4.2 In light of this difficulty, host systems should maintain copies of PCD's for each piece of equipment under its control and not expect to be able to obtain the PCD from the machine whenever it is required. Doing so, in fact, will permit more flexible process program development in the host, allowing creation of process programs even when equipment is not online and encourage the use of formatted process programs by equipment manufacturers.

A5 Suggested Baseline SECS Equipment Implementation

A5.1 *Purpose and Scope*

A5.1.1 This document provides a recommendation prepared by the Rigid Disk Subcommittee for generating a baseline implementation of the SECS (SEMI Equipment Communication Standard) standards on production process and test equipment. This document is not a tutorial to aid in understanding SECS but rather serves as an introduction to the requirements of SECS, and a brief guide to the selection of SECS messages for equipment. Actual system requirements of many implementations are beyond the scope of this document. The full standards, SEMI Equipment Communications Standard I (SECS-I), SEMI E4 and SEMI Equipment Communications Standard II (SECS-II), SEMI E5 should be consulted by all users.

A5.2 *Introduction*

A5.2.1 The SECS standards are an existing and developed set of communication standards currently used by the semiconductor and other industries to support automated production. The standards provide a means for communicating information and control between production equipment and a "host" computer. This transfer of information and control can be used to provide production tracking and location of WIP (Work-In-Process), scheduling of WIP and control of material transfer at the equipment. The process measurements and records can be used to provide process engineers with a database for statistical process control. SECS messages are appropriate to a wide variety of applications, including measurement, processing, and material transport equipment.

A5.3 *SECS-I Standard*

A5.3.1 SECS-I defines the lower protocol layers of a point-to-point interface between equipment and a host computer system. The standard requires a simple, well understood physical interface, RS-232. The SECS-I protocol allows the equipment control over the protocol: the equipment is the master of the link and can initiate the transfer of a message to the host. Likewise, the equipment can regulate the receipt of a message by its response to a handshake to receive the message. Thus, the interface takes place at the convenience of the equipment, and equipment with very limited computer resources can still support the standard.

A5.4 *SECS-II Standard*

A5.4.1 SECS-II defines the higher layer in the protocol, including message content, structure, and data types and their formats. SECS-II defines messages in sets with related functions called Streams. The actual content of the messages are specific to an application, but it is possible for a properly designed host software system to unpack a message and present the data in a meaningful way with no prior definition as to the content. Equipment need only implement those messages appropriate to meet its system requirements; thus, very simple equipment will require implementation of few messages.

A5.4.2 Application Note A2 contains some suggestions for message utilization, including some

information on minimum message sets. It is the intent of this report to expand in a somewhat different direction, to identify those messages which typically constitute a sufficient set given a selected equipment function.

A5.4.3 To select a message set, the requirements for the equipment must be identified. This requirement, in turn, determines a message set. Identified below are a number of types of equipment requirements and a baseline set of messages supporting those requirements. The message sets identified are baseline recommendations. Actual equipment implementations may need more messages than those specified here to satisfy all system requirements. The published standards should be consulted for all applications. Issues such as handling of multi-block messages, optional replies, and others are described in detail in the SEMI specifications and are not a topic of this baseline recommendation.

A5.4.4 The implementation for a specific equipment type begins by specifying which tasks that equipment is required to perform from the following list, and then studying the expanded descriptions for those tasks chosen in the SECS-II implementation section.

Typical Tasks for Measurement and Process:

1. Measurement or Action Reports
2. Equipment Alarm Reports
3. Remote Request for Equipment Condition or State
4. Operator Interface to the Host
5. Remote Access to Process Programs
6. Remote Commands

Typical Tasks for Material Control and Transport:

1. Material Status Information
2. Material Transport Control

Additional Tasks for Special Situations:

1. File Transfer

A5.5 *Baseline SECS Implementation Recommendations: SECS-I*

A5.5.1 The baseline SECS-I requirement for equipment is to fully implement the SECS-I protocol as defined in SEMI E4. This requirement applies to all SECS compatible equipment independent of the equipment's function. The flow chart (Figure 2 of SEMI E4) illustrates the block transfer protocol of SECS-I. The body of the SECS-I standard describes protocol timeouts and other requirements that are essential components of the specification.

A5.6 *Baseline SECS Implementation Recommendations: SECS-II*

A5.6.1 The SECS-II implementation begins with a choice or selection of messages for the equipment. In order for equipment to meet the baseline requirements for a viable SECS-II interface, the equipment must be capable of generating a certain set of messages and recognizing another set. The messages required for either set will depend on tasks of equipment and on system requirements for production and process control by the host. Equipment must accept S1,F1 and send S1,F2. Note that implementation in the reverse direction is optional. In order to ensure a viable data communication link, certain messages are required:

Messages for All Equipment — Required by SECS-II

a. Messages Generated by the Equipment

   S1,F2:On Line Data

   S9,F1: Unrecognized Device ID

   S9,F3: Unrecognized Stream Type

   S9,F5: Unrecognized Function Type

   S9,F7: Illegal Data

b. Messages Recognized by the Equipment

   S1,F1: Are You There Request

A5.6.2 In addition to those messages which are required, the following are strongly recommended. These messages are used for diagnostic purposes:

Messages for All Equipment — Strongly Recommended

A. Messages Generated by the Equipment

   1. S2,F25: String Diagnostic Request

B. Messages Recognized by the Equipment

   1. S2,F26: String Diagnostic Data

A5.6.3 Note that messages are identified by a "stream" number and a "function" number. All primary messages have an odd-numbered function, and the corresponding reply is the next consecutive even-numbered function. Thus, many messages are paired. For example, the Stream 1, Function 1 (S1,F1) message generated by either host of equipment has the reply of Stream 1, Function 2 (S1,F2).

A5.6.4 *Reports of Measurements or Process Actions* — The simplest task for measurement and process equipment is to report their measurements or actions to the host. This report could also include equipment setup parameters and sample identification. The ability to accurately transfer this data from equipment to a factory

host is of prime importance in automating production. Depending on the equipment requirements, the function can be far greater.

A5.6.4.1 *Measurements or Action Reports* — Given an equipment requirement to relay data to a host computer, the equipment must handle one or both of the following messages as defined by SEMI E5.

S6,F3: Discrete Variable Send

S6,F9: Formatted Variable Send

A5.6.4.2 *Equipment Alarm Reports* — These messages are unsolicited by the host, they transmit information warning of conditions threatening personal safety, equipment safety, or out of limit equipment parameters which may cause harm to the product or indicate equipment malfunction:

S5,F1: Alarm Report Send

A5.6.4.3 *Remote Request for Equipment Condition or State* — This is the method where the host requests data from the equipment.

Messages Transmitted to the Equipment:

S6,F7: Data Transfer Request

Messages Transmitted by the Equipment in Reply:

S6,F8: Data Transfer Data

A5.6.4.4 *Operator Interface to the Host* — Many equipment systems will have an interactive operator's console; some will have computer terminals used for this purpose. SECS-II has a message type to transfer text from host to the equipment console. There are also messages through which equipment operators may transmit text from their console, directly to the host. This text may include desired information which is not accessible by the equipment computer directly.

Message Transmitted by the Equipment:

S10,F1: Terminal Request

Message Recognized by the Equipment:

S10,F3: Terminal Display, Single

A5.6.4.5 *Remote Access to Process Programs* — SECS-II defines a means for storing or retrieving equipment process programs. This function allows upload and download of such programs through the SECS interface. By this means, programs for equipment may be archived in the host computer system. Either the host or the equipment may request the transfer of a Process Program from the other. To do this:

The Requestor transmits:

S7,F5: Process Program Request

The Sender of the Process Program replies:

S7,F6: Process Program Data

In addition, either host or equipment may initiate sending the Process Program. In this case:

The sender transmits:

S7,F1: Process Program Load Inquire,

Receiver replies with:

S7,F2: Process Program Load Grant,

Sender transmits:

S7,F3: Process Program Send,

The receiver answers:

S7,F4: Process Program Acknowledge.

A5.6.4.6 *Remote Commands* — The host can initiate a command to the equipment in a manner similar to an operator pressing a button.

The host transmits:

S2,F21: Remote Command Send,

The equipment replies:

S2,F22: Remote Command Acknowledge.

Execution of the remote command may cause the equipment to transmit a message to the host at a time greater than the reply time required for the S2,F22 message. If this type of reply is desired, the S6,F3: Discrete Variable Data Send may be used to transfer data to the host.

A5.6.5 *Typical Tasks for Material Control and Transport*

A5.6.5.1 *Material Status Information* — The host may query the equipment for material-in-process information. The information is transmitted only as a answer to a host request.

The equipment recognizes:

S3,F1: Material Status Request.

The equipment transmits:

S3,F2: Material Status Data.

A5.6.5.2 *Material Transport Control* — The SECS-II protocol includes the means to affect automated transfer of material from one SECS-compatible device to another. Baseline compatibility requires the equipment to perform a simple material transfer process, an actual implementation may require means for graceful error recovery as well. This recommendation does not include messages to handle error conditions.

Receiving Material:

Equipment Recognizes S4,F1:

Ready to Send Material

Equipment Transmits:

S4,F3: Send Material,

S4,F5: Handshake

Sending Material:

Equipment Transmits:

S4,F1: Ready to Send Material

Equipment Recognizes:

S4,F3: Send Material,

S4,F5: Handshake Complete

Equipment Recognizes:

S4,F2: RTS Acknowledge

A5.7 *Conclusion* — The baseline requirements for equipment using the SECS standards includes all of SECS-I and a limited selection of messages from SECS-II. The choice of SECS-II messages, and data contained therein, is dictated by the equipment and system requirements. The benefits in using the standards are many, including support for growth in equipment function, and standardization needed for effective automation. The results include automated process monitoring and all of the associated benefits.