EDGE INTEGRATION

A Guide
to
Understanding
GEM - SECS - HSMS

March 24, 2004

Version 1.1

Table of contents

Introduction

The intent of this guide is to help simplify some of the principles around interfacing with semiconductor equipment and their communications standards. This is not intended to be a substitute for the Standards themselves since this guide does not cover all details o f the Standards . It only serves as general overview to aid in the support of applications that use or deal with such interfaces.

The information presented in this guide is believed to be correct and accurate. Please send all comments and questions to john@edgeintegration.com

The information presented within this guide may be used to your benefit, and may be reproduced and distributed freely providing the copyright notice remains intact. No further permissions are required.

Overview

This guide discusses some of the SEMI Standards that cover the interfacing to semiconductor equipment. The reader should have as a minimum, a technical background and some knowledge about semiconductor fabs and computer science.

SECS

This is a discussion of SECS (Semiconductor Equipment Communication Standard). SECS is part of the SEMI Standards, E4 and E5, Equipment Automation /Software Volume. It was developed in 1980 by Hewlett Packard. SECS is a point-to-point protocol via RS-232.

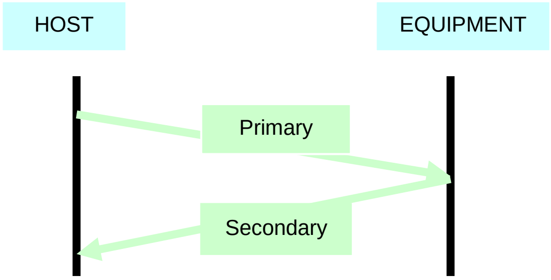
SECS is a layered protocol consisting of 3 levels: Message Protocol, Block Transfer Protocol, and the Physical Link.

SECS-II	Message Protocol
SECS-I	Block Transfer Protocol
RS-232	Physical Link

SECS-II Message Protocol

SECS-II	Message Protocol
SECS-I	Block Transfer Protocol
RS-232	Physical Link

The Message Protocol is used to send SECS-II messages between the Host and Equipment. Each SECS-II message contains a primary message and an optional secondary reply message. This is also referred to as a transaction.

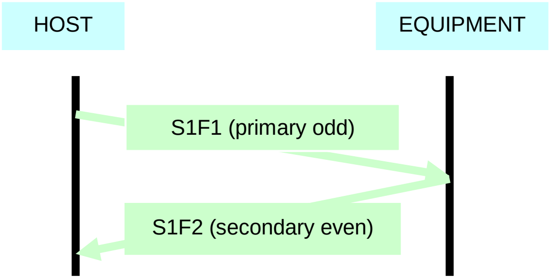


Primary – Is a SECS-II message initially sent by either the Host or Equipment.

Secondary – Is an optional SECS-II message send in response to a primary message.

Streams and Functions

SECS-II messages are referred to as Streams and Functions. Each message has a Stream value (Sx) and a Function value (Fy). In the case of a Stream 1 Function 1, it is written as S1F1, and spoken as “S1F1”. Streams are categories of messages while Functions are specific messages within the category. The function value is always an odd number in a primary message, and one greater, or even, in the associated secondary reply.



Stream	Catagory
1	Equipment Status
2	Equipment Control
3	Material Status
4	Material Control
5	Alarm Handling
6	Data Collection
7	Recipe Management
8	Control Program Transfer
9	System Errors
10	Terminal Services
11	(Not Used)
12	Wafer Mapping
13	Unformatted Data
	Set Transfers

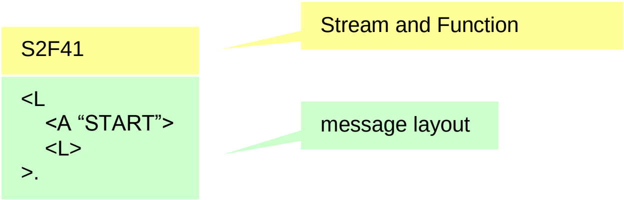
Some Streams and Functions are reserved while others are user defined.

Stream	Function	Availability
0	0 to 255	Reserved
1 to 63	0 to 63	Reserved
64 to 127	0	Reserved
1 to 63	64 to 255	User defined
64 to 127	1 to 255	User defined

All function 0's have a special meaning. They are sent as a reply to an aborted primary message.

Message Layout

A SECS-II message contains a structure or layout. The layout defines the all data items for the SECS-II message. The layout is that part of the message which follows the Stream and Function notation.



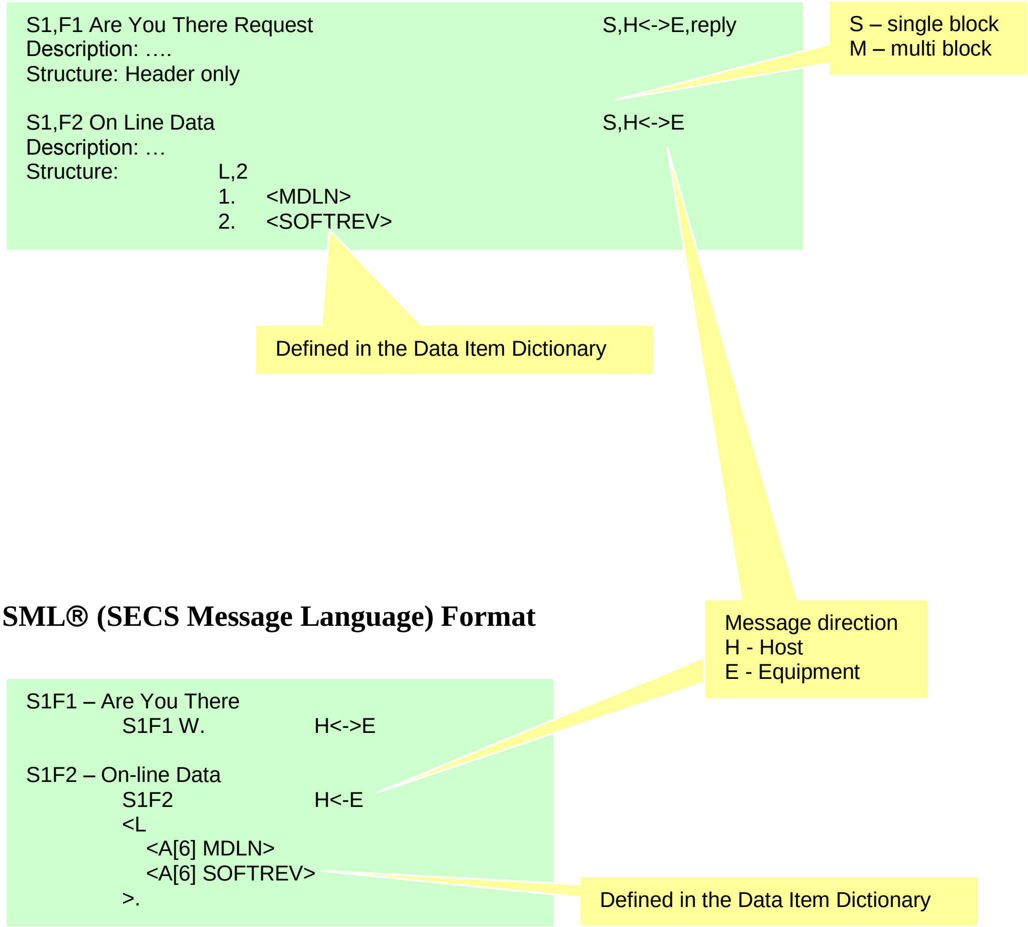
```
S2F41
<L
   <A "START">
   <L>
>.
```

Message Annotation

Providers of SECS interfaces typically provide an interface manual. This manual details the applicable SECS messages and their layouts. It may also include a Data Item Dictionary that defines the message items.

SECS-II messages can be annotated using several conventions. Two of the more common conventions are show below:

SEMI Standard



```
S1,F1 Are You There Request                          S,H<->E,reply
Description: ….
Structure: Header only

S1,F2 On Line Data                                   S,H<->E
Description: …
Structure:          L,2
                    1.    <MDLN>
                    2.    <SOFTREV>
```

S – single block
M – multi block
Defined in the Data Item Dictionary
Message direction
H - Host
E - Equipment
Defined in the Data Item Dictionary

SML® (SECS Message Language) Format

```
S1F1 – Are You There
          S1F1 W.            H<->E

S1F2 – On-line Data
          S1F2               H<-E
          <L
            <A[6] MDLN>
            <A[6] SOFTREV>
          >.
```

Data Items

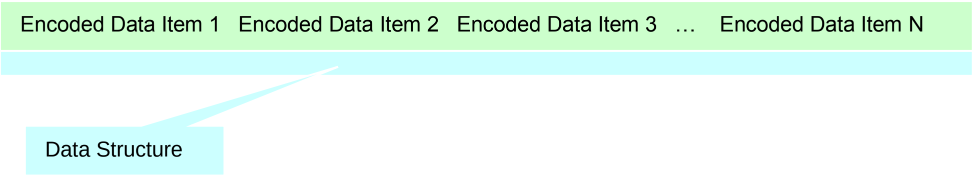
From above, SECS-II message layouts contain data items. Data items are defined by format codes. All possible data items, along with their format codes, are listed in the following table. The format code is defined by 6 bits only. This will become clearer later.

Note that the List data item is a data item used for grouping more data items. And Lists can contain Lists.

Data Item Type	Format Code
List	000000
Binary	001000
Boolean	001001
ASCII	010000
JIS-8	010001
8-byte integer (signed)	011000
1-byte integer (signed)	011001
2-byte integer (signed)	011010
4-byte integer (signed)	011100
8-byte floating point	100000
4-byte floating point	100100
8-byte integer (unsigned)	101000
2-byte integer (unsigned)	101001
2-byte integer (unsigned)	101010
4-byte integer (unsigned)	101100

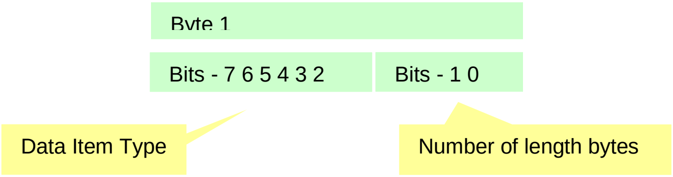
Layout Encoding

Before a SECS-II message can be sent, the layout must be encoded. Each data item in the message layout is encoded and becomes part of the Data Structure. This is repeated until all the data items in the layout are encoded. The Data Structure is subsequently sent via the SECS-I protocol.



A data item is encoded as follows:

The first byte of the Encoded Data Item contains the data item's format code and also defines how many of the following bytes define the length of the data item. The format code is contained within bits 2 to 7, and the number of length bytes is defined in bits 0 to 1. Therefore, the number of length bytes to follow ranges from 0 to 3 bytes.

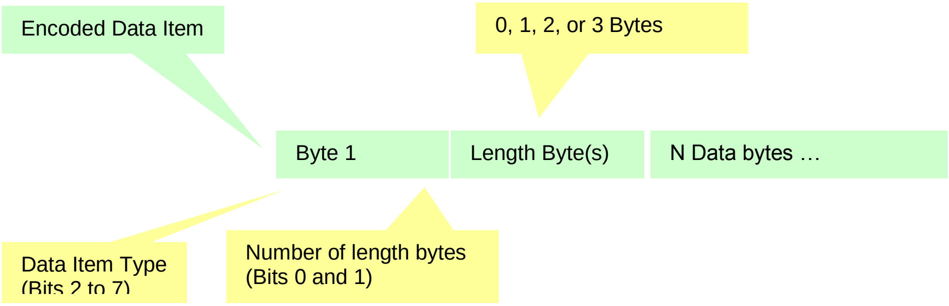


The next 0, 1, 2, or 3 bytes define the length of the data. So for example, if the size of the data is 1000 bytes, then 2 length bytes would be needed to represent 1000 (03 E8 Hex or 00000011 11100100 Binary)

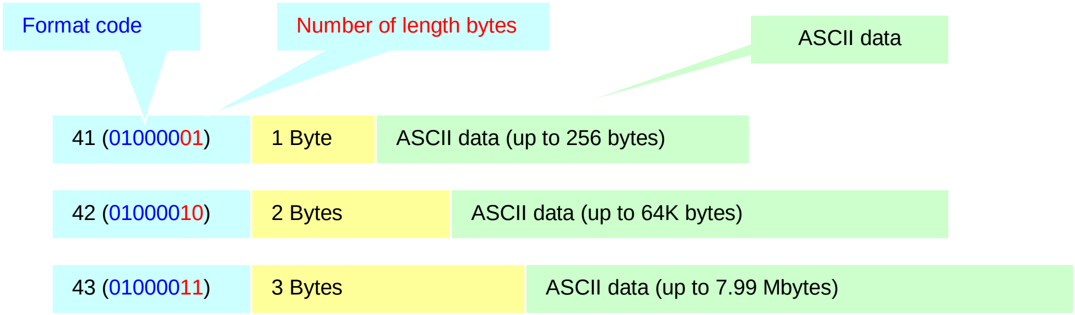
The following table shows the max number of data bytes that can be represented by the number of length bytes.

Number of length bytes	Max data size
0	Empty item
1	256 bytes
2	64 Kbytes
3	7.99 Mbytes

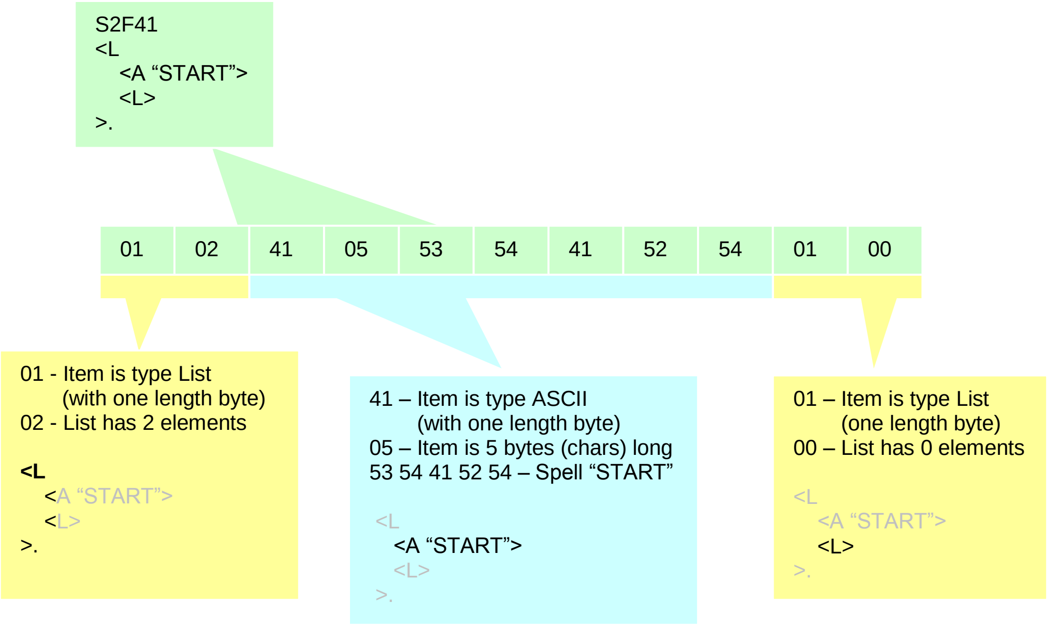
The next n bytes contain the data for the data item.



The following shows how an ASCII data item can be encoded. Depending upon how many data bytes it contains, it may require zero, one, two, or three length bytes to represent the total number of data bytes.



The following shows how an example SECS-II message layout is encoded. Note that only the message layout is encoded, the stream and function are not encoded in this example. All numeric values are in Hex, (ex: 52 = 01010010).


```
S2F41
<L
    <A “START”>
    <L>
>.
```

01	02	41	05	53	54	41	52	54	01	00

```
01 - Item is type List
       (with one length byte)
02 - List has 2 elements

<L
    <A “START”>
    <L>
>.
```

```
41 – Item is type ASCII
        (with one length byte)
05 – Item is 5 bytes (chars) long
53 54 41 52 54 – Spell “START”

<L
    <A “START”>
    <L>
>.
```

```
01 – Item is type List
        (one length byte)
00 – List has 0 elements

<L
    <A “START”>
    <L>
>.
```

Multi-Block Messages

Due to the data size limitations in the SECS-I protocol, the encoded SECS-II message (Data Structure) may not fit into one SECS-I transaction. Therefore the encoded SECS-II message is divided into smaller blocks, and sent one block at a time. This is referred to as multi-block messaging.

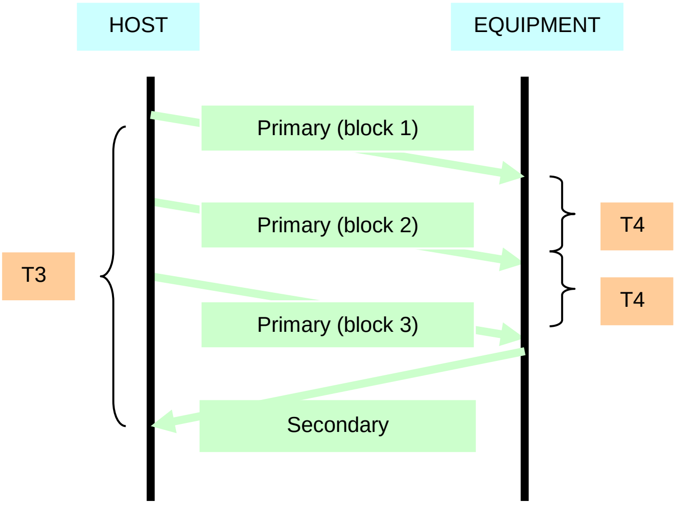
Each SECS-I message block is limited to 244 of encoded SECS-II messages bytes. The maximum number of message bocks is 32,767, so the maximum SECS-II encoded message size is 244 x 32,767 bytes long.

SECS-II Parameters - Are used to define some of the bounds while implementing the Message Protocol. The following table shows the SECS-II parameters:

SECS-II Parameters	
T3	Reply Timeout
T4	Inter-Block Timeout

T3 – This is the time period between sending a primary SECS-II message and receiving the secondary reply.

T4 – This is the time period between receiving subsequent blocks in a multi-block message.



Interleaving Messages

This is when more than one transaction is being handled at a time. A transaction is the sending of a primary message and the optional receiving a secondary reply. A transaction is “opened” when it is waiting for its reply. Interleaving is the ability to have more than one “opened” transaction. This allows for the sending of multiple primary messages without first waiting for the secondary reply. Supporting this feature is not a requirement of SECS-II.

Common SECS-II Messages

Here we list the more commonly used SECS-II messages.

The first set of messages are typically used at initialization when first connecting to the equipment.

SECS-II	Description
S1F1	Say hello to the equipment
S1F13	Establish communications with the equipment
S2F15	Set equipment constants
S2F43	Turn off spooling
S2F33	Create / Delete reports
S2F35	Link reports to events
S2F37	Enable / Disable events
S5F3	Enable / Disable alarms

These message(s) are used for recipe selection, remote start, wafer selection etc

SECS-II	Description
S2F41	Remote equipment control

These message(s) are used for data collection.

SECS-II	Description
S6F3	Older version of Event Reports
S6F9	Older version of Event Reports
S6F11	Event Reports

The S5F1 is used to handle equipment alarms.

And S9's are sent by the equipment when it detects a previous message error.

SECS-I Block Transfer Protocol

SECS-II	Message Protocol
SECS-I	Block Transfer Protocol
RS-232	Physical Link

The Block Transfer Protocol is used to establish the direction of communication and provide and environment for passing message blocks. A primary or reply message may require multiple Block Transfers (multi-block message.)

Handshake Sequence

The handshake sequence is the exchange of bytes between the Host and Equipment.

Handshake Codes – There are four handshake codes. These are used to control data flow in the block transfer protocol. The following table shows the handshake codes:

Handshake Codes	Hex Value	
ENQ	05	Ready to Send
EOT	04	Ready to Receive
ACK	06	Correct Reception
NAK	15	Incorrect Reception

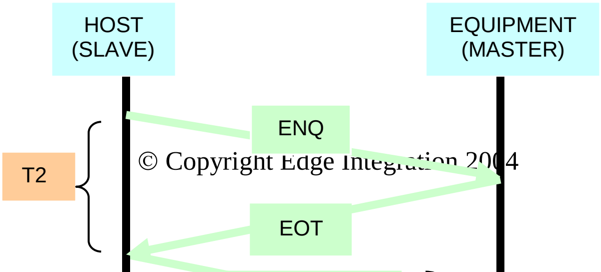
SECS-I Parameters – Are used to define some of the bounds while implementing the block transfer protocol. The following table shows the SECS-I parameters:

SECS-I Parameters	
T1	Inter-Character Timeout
T2	Protocol Timeout
RTY	Retry Limit
Master/Slave	Resolve contention

T1 – Is the time between receiving each character (after receiving the first character) in the BLOCK DATA

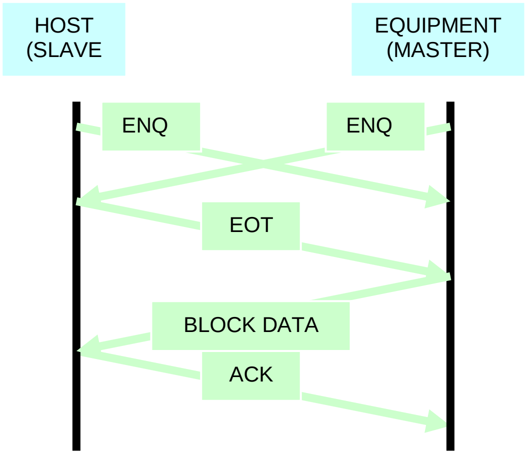
T2 – Is the time between sending the ENQ and receiving the EOT

The following diagram shows the handshake codes during the bock transfer protocol. It also shows the T1 and T2 timeouts and a possible NAK:



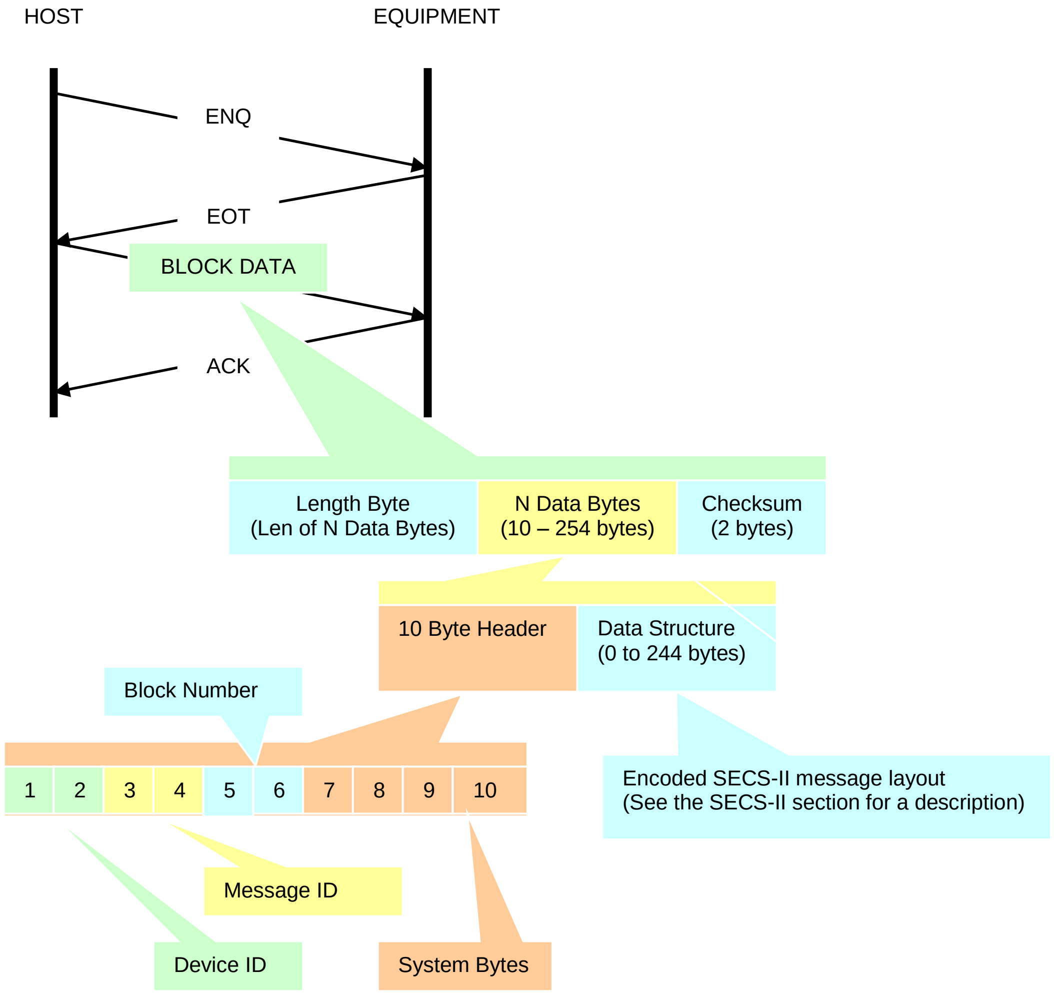
RTY – This defines the number of times to initiate the block transfer (start sending with the ENQ.) Upon the receipt of a NAK (or other protocol error), the RTY value is adjusted and the block transfer is again attempted.

Master/Slave – Is used to resolve contention. The host is slave and the equipment is the master. Contention is when both the host and the equipment attempt to communicate at the same time. The following shows how contention is resolved



Block Data

The Block Data is the data portion of either a single-block or multi-block message. It contains a Length Byte, N Data Bytes, and a two byte Checksum.



Length Byte – Is the first byte sent in the Block Data. It is the number of bytes in the N Date Bytes (10 Byte Header + Data Structure). The two byte Checksum is not included in this count The value ranges from 10 to 254 bytes.

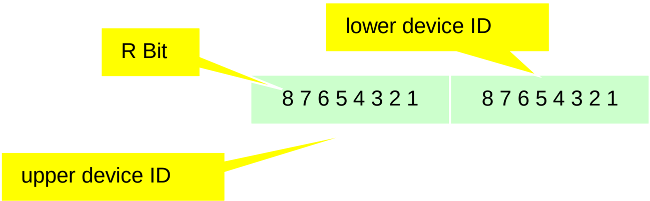
N Data Bytes – Is the data portion of the message block. It contains a 10 Byte Header, and the Message Data. The size ranges from 10 bytes to 254 bytes. The minimum size of N Data Bytes is 10 (contains only the 10 Byte Header), and the maximum is 254.

Message Data - Is the data portion of the message. It contains all or a portion (if a multi-block message) of the encoded SECS-II.

10 Byte Header

Each message contains a header. The header is used to describe the data contained within the Block Data. It contains the Device ID, Message ID, Block Number, and System Bytes. Other important indicators are also included.

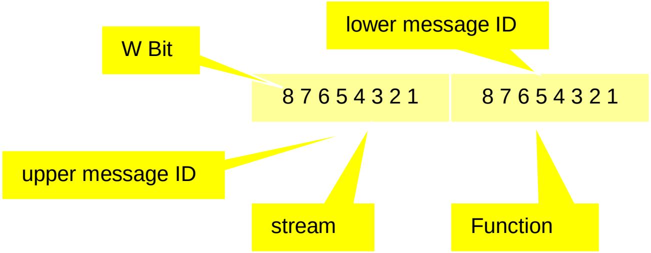
Device ID – Bytes 1 and 2 of the header are the device ID. The left byte is the upper device ID, and the right byte is the lower device ID. The left most bit of the left byte is the reverse bit (R-bit). The R-bit determines the direction of the message.



R-Bit	Direction
0	Host to Equipment
1	Equipment to Host

Message ID – Bytes 3 and 4 of the header is the message ID. The left byte is the upper message ID, and the right byte is the lower message ID. The left most bit of the left byte is the Wait bit (W-bit). The W-Bit is used to indicate that the sender of the primary message is expecting a reply.

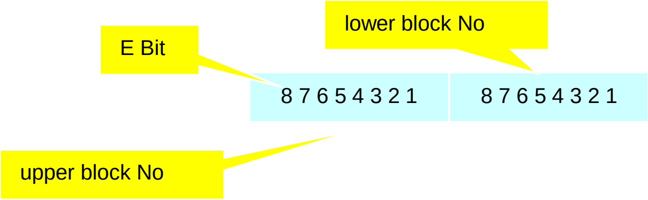
The message ID indicates the SECS-II stream and function. The upper message ID is the stream, and the lower message ID is the function.



W-Bit	Expected reply
0	No reply is expected
1	Reply is expected

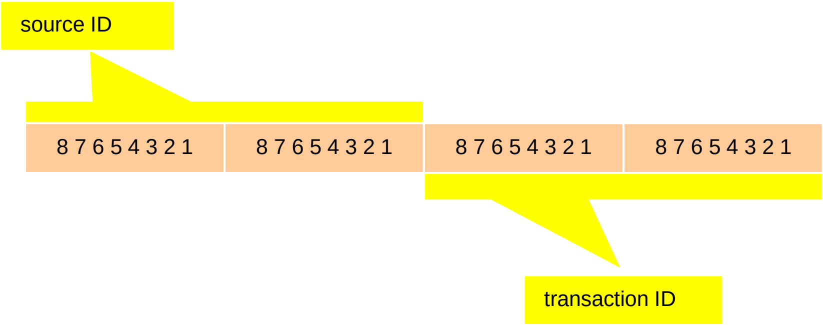
Block Number – Bytes 5 and 6 of the header is the block number. The left byte is the upper block number, and the right byte is the lower block number. The left most bit of the left byte is the end bit (E-bit). The E-bit indicates this is the last block of a message.

A multi-block message is used when the sent message requires more than one block. Each block is number and sent in order. Since the receiver of the message does not know how may blocks to expect, the E-bit indicates which block is the last, or if there are more bocks to follow. The maximum number of blocks in a multi-block message is 32,767.



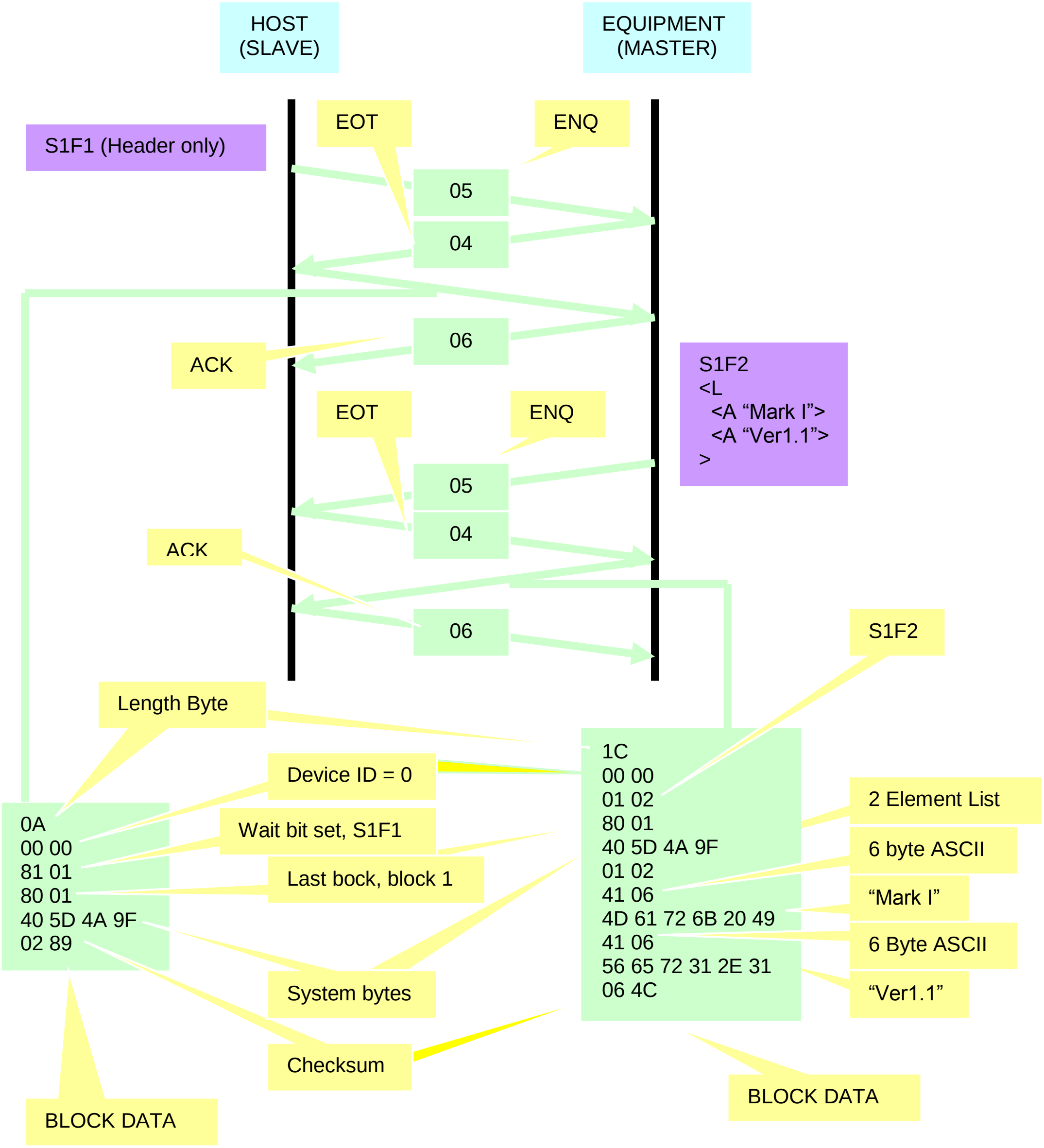
E-Bit	Last block
0	More blocks to follow
1	This is the last block

System Bytes – The last four bytes of the header are the system bytes. The left two bytes are the source ID, and the right two bytes are the transaction ID. The source ID identifies the sender of the message and is used for message routing. The transaction ID identifies the message and is unique for each message sent. A primary message at its associated reply have the same (matching) system bytes.



Example of S1F1 / S1F2

The following shows the bytes transferred while a sending primary message, and receiving a secondary reply. This is not a multi-block message. The Host, send a S1F1 and the Equipment sends a S1F2 reply.



Typical SECS Protocol Parameters

Here is a summary set of the SECS-I / SECS-II Protocol Parameters along with some typical values.

Parameter	Function	Typical value
Baud Rate	Serial line speed	9600
Device ID	Identifier assigned to equipment	0
T1	Inter-Character Timeout	1 (second)
T2	ENT / EOT Timeout	10 (seconds)
T3	Reply Timeout	45 (seconds)
T4	Inter-Block Timeout	45 (seconds)
RTY	Retry Limit	3
M / S	Master / Slave	Host-Slave Equip - Master

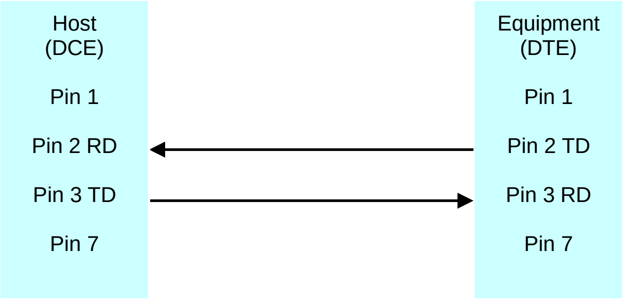
RS-232 Physical Link

SECS-II	Message Protocol
SECS-I	Block Transfer Protocol
RS-232	Physical Link

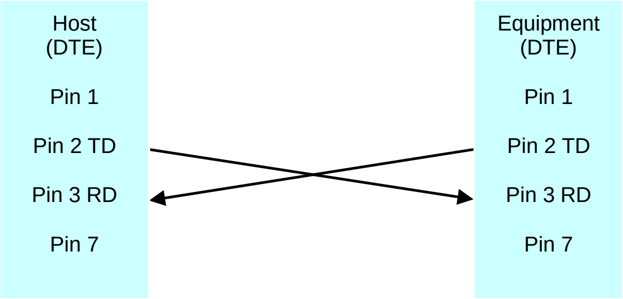
This is the lowest protocol layer. It defines the physical interface at the equipment. It is based on the RS-232-C standard. 25-pin type "D" connectors (or other) are used. The following are the pins used for communication.

Pin	RS-232-C Circuit	Description
1	AA	Shield
2	BA	Data from Equipment
3	BB	Data to Equipment
7	AB	Signal Ground
18		+12 to +15 volts (opt)
25		-12 to –15 volts (opt)

The equipment sends data on pin 2, and receives data on pin 3. It is therefore, called "Data Terminal Equipment" (DTE, or Computer). If the other end (Host side) is "Data Communication Equipment" (DCE) a straight through cable is used between the two.



However, if the other end (Host side) is also a DTE, then a null-modem connector or cable is required. This effectively swaps pins 2 and 3.

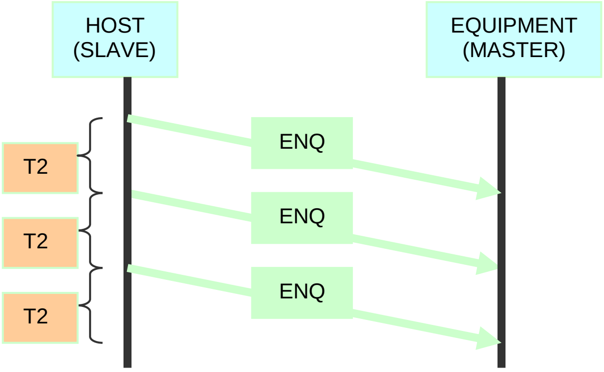


Protocol Troubleshooting

This sections covers some of the more common error scenarios encountered when working with the SECS protocol.

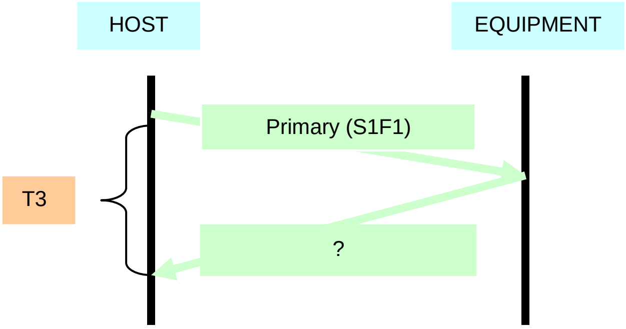
T2 Timeout

The following shows a T2 timeout scenario. The host is attempting to communicate with the equipment but the equipment is off-line and not responding. This error condition could be caused by a physical disconnect, or equipment being off-line.



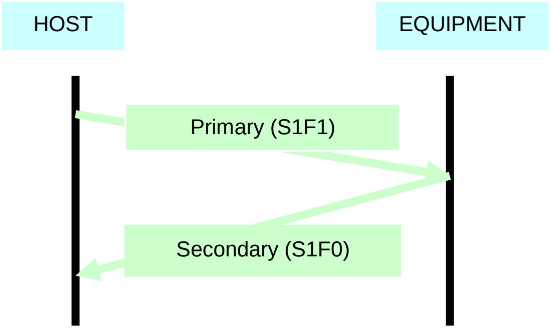
T3 Timeout

The following shows a T3 timeout scenario. The host sends a primary message to the equipment. The is done without any errors. However, the equipment fails to send the secondary reply message. The error condition shows that the host and equipment are communicating, but the equipment is not responding perhaps because communications have not yet been established.



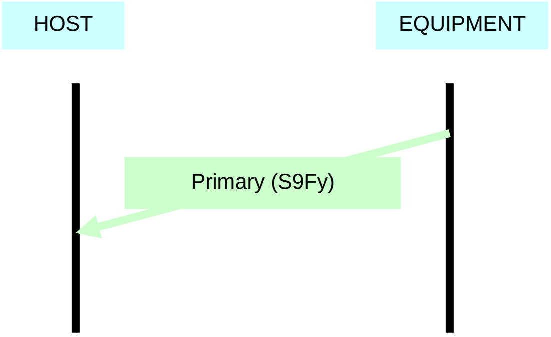
Function 0

Function 0 is reserved for all Streams to indicated a failed transaction. This secondary reply is sent by the receiver, so the sender does not have to wait for a T3 timeout. In any case, it indicates a failed transaction just like a T3 timeout.



Stream 9's

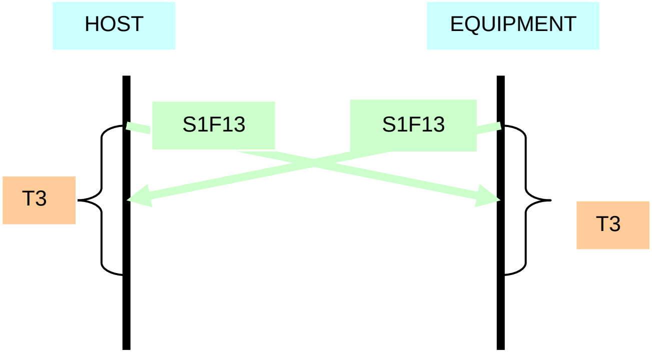
Streams 9's are sent by the equipment to host to indicating: either a message from the host cannot be handled, or the equipment detected a timeout on a previously received message.



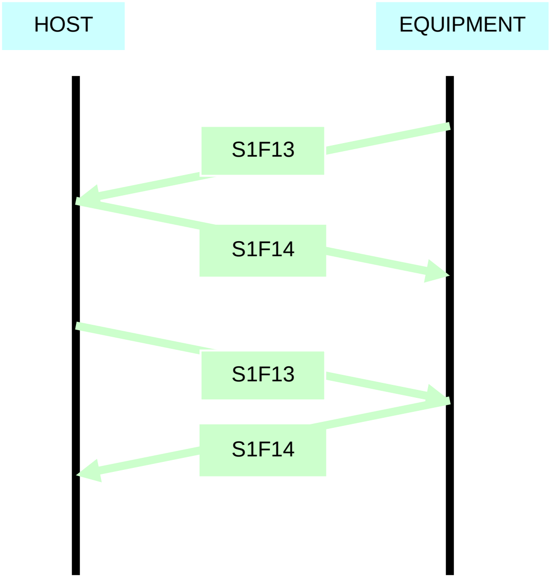
Establishing Communications

Before SECS-II messages can be sent between host and equipment, communications must first be “established”. This is done via a S1F13 (Establish Communications Request) message. This should be the first message sent following an initial startup or after a long period of not communicating.

Often times, the host and the equipment both initiate a S1F13, and unfortunately, both timeout waiting for the reply. Doing this is not recommended.



It may be best to allow the equipment to send the S1F13 first. Then the host can send a S1F13 after replying to the equipment’s S1F13. Getting the equipment to send the S1F13 may take some experimenting with the equipment. Sometimes taking the equipment in and out of remote mode, or taking the equipment off-line then on-line, causes the equipment to send a S1F13.



HSMS

This is a discussion of the HSMS (High-Speed SECS Message Services). HSMS is based on the SEMI E37 standard. It provides a means for sending SECS-II messages to the equipment at greater speeds. HSMS is SECS-II over TCP/IP.

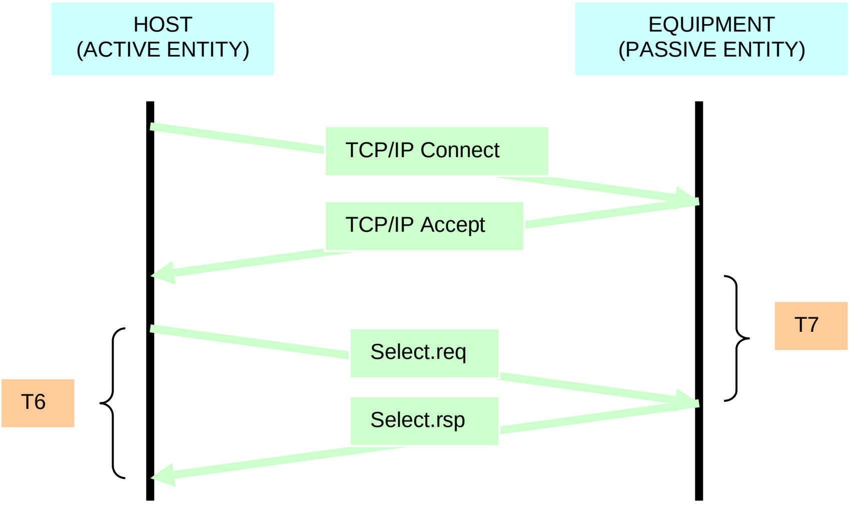
HSMS Procedures

HSMS procedures are the different types of messages passed between the host and the equipment. Here is a summary of the procedures:

Procedure	Description
Select	Establish a connection with the equipment
Deselect	Break the HSMS connection
Data	SECS messages between host and equipment
Link-test	Determine if the host-equipment link is active
Separate	Terminate HSMS connections immediately

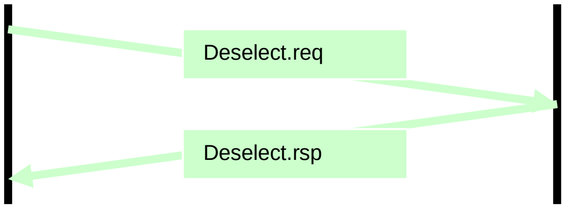
Select

The Select procedure is used to by the host to establish a connection with the equipment



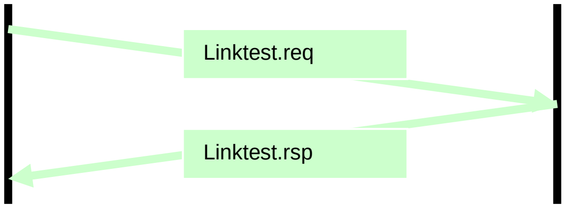
Deselect

The Deselect is used by either the host or equipment to break the connection.



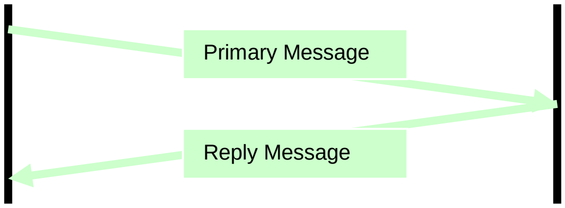
Link Test

The Link Test procedure can be sent by either the host or equipment to test if the link is active.



Data

The Data procedure is used to send and receive SECS-II messages between the host and equipment



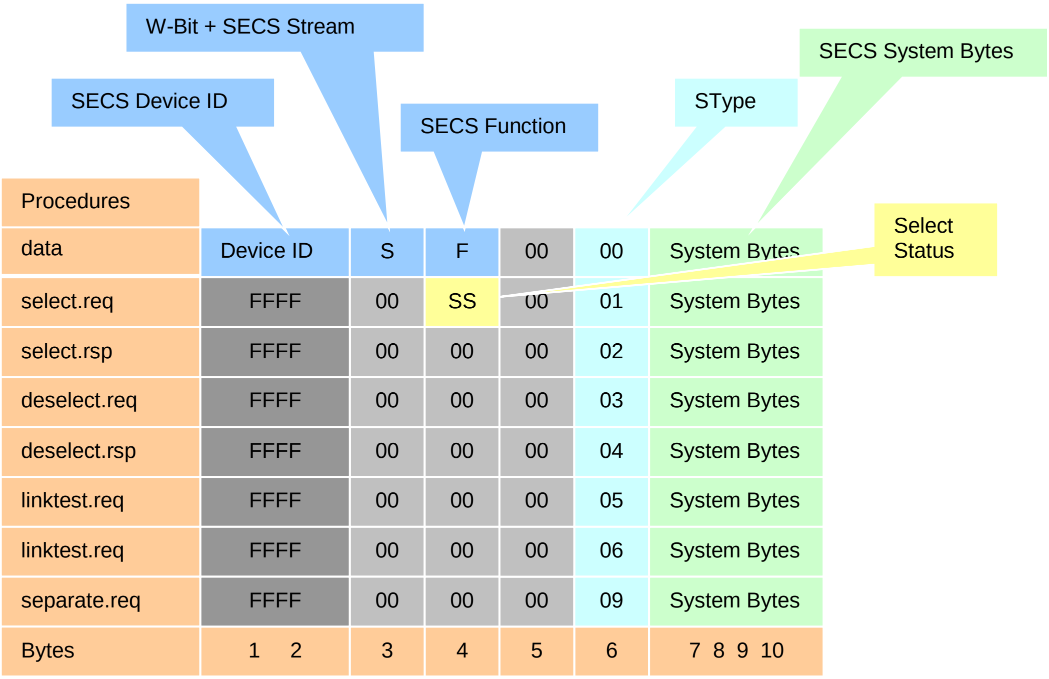
HSMS Message Format

This is the format of a HSMS message. Each message is begins with a Message Length, then a Message Header, followed by the Message Text.

Message Length (4 Bytes)	Message Header (10 Bytes)	Message Text (0 to ~ 8 Mbytes)

Message Length – Is a four byte value that defines the length of the entire HSMS message (Message Header + Message Text).

Message Header – Describes the HSMS message or procedure.



Procedures						
data	Device ID	S	F	00	00	System Bytes
select.req	FFFF	00	SS	00	01	System Bytes
select.rsp	FFFF	00	00	00	02	System Bytes
deselect.req	FFFF	00	00	00	03	System Bytes
deselect.rsp	FFFF	00	00	00	04	System Bytes
linktest.req	FFFF	00	00	00	05	System Bytes
linktest.req	FFFF	00	00	00	06	System Bytes
separate.req	FFFF	00	00	00	09	System Bytes
Bytes	1 2	3	4	5	6	7 8 9 10

Message Text – Is the data portion of the HSMS message. It is only relevant for the Data procedure. The data in the Message Text is the encoded SECS-II as described in the SECS-II Message Protocol section.

Typical HSMS Protocol Parameters

Here is a list of the HSMS Protocol Parameters and some typical values:

Parameter	Function	Typical Value
SECS Device ID	Device ID	0
Active or Passive	Client or Server	Host – Active (client) Equip – Passive (server)
Passive Entity IP Address	IP Address	192.9.200.1
Passive Entity TCP Port	Port Number	5000
T3	Reply Timeout	30 (seconds)
T5	Connection Separation Timeout	10 (seconds)
T6	Control Transaction Timeout	10 (seconds)
T7	Not Selected Timeout	10 (seconds)
T8	Network Timeout	10 (seconds)

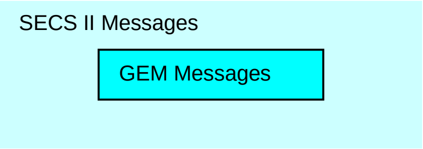
GEM

This is a discussion of the GEM (Generic Model for Communications and Control of Manufacturing Equipment) GEM is based on the SEMI E30 standard. It defines a common set of equipment behavior and communications capabilities.

What's it all about

Where SECS-II defines the messages and data items sent between the host and equipment, GEM defines which SECS-II messages should be used and when. The GEM standard does not effect nor apply to the behavior of the host. When a GEM message is sent or received by the equipment, the equipment will behave in a predictable manor.

The below shows that GEM messages are a subset of SECS-II messages



GEM Compliance

What is GEM Compliance?

The GEM standard contains two specifications:

- Fundamental GEM Requirements
- Additional GEM Capabilities

The following table show the Fundamental GEM Requirements:

Requirement
State Models
Equipment Processing States
Host-Initiated S1F13/F14 Scenario
Event Notification
On-line Notification
Error Messages
Control (Operator-Initiated)
Documentation
Variable data items (as they apply)
SECS-II data item restrictions (as they apply)
Collection Events (as they apply)

Additional GEM Capabilities

Capability
Establish Communications
Event Notification
Dynamic Event Report Configuration
Variable Data Collection
Trace Date Collection
Limits Monitoring
Status Data Collection
On-line Identification
Alarm Management
Remote Control
Equipment Constants
Process Program Management
Material Movement
Equipment Terminal Services
Error Messages
Clock

To be GEM compliant, you must:

- Meet the Fundamental GEM requirements
- Where Additional GEM Capabilities are implemented, you must conform to the Capability as GEM defines it. For example, you need not implement Remote Control to be GEM compliant, but if you do, you need to implement it the GEM way.