The GEM Standard and Examples

The GEM Standard

- The GEM Standard Document

The Generic Equipment Model (GEM) Standard is published by Semiconductor Equipment and Materials International, Inc. (SEMI), and is identified as SEMI Standard E30.

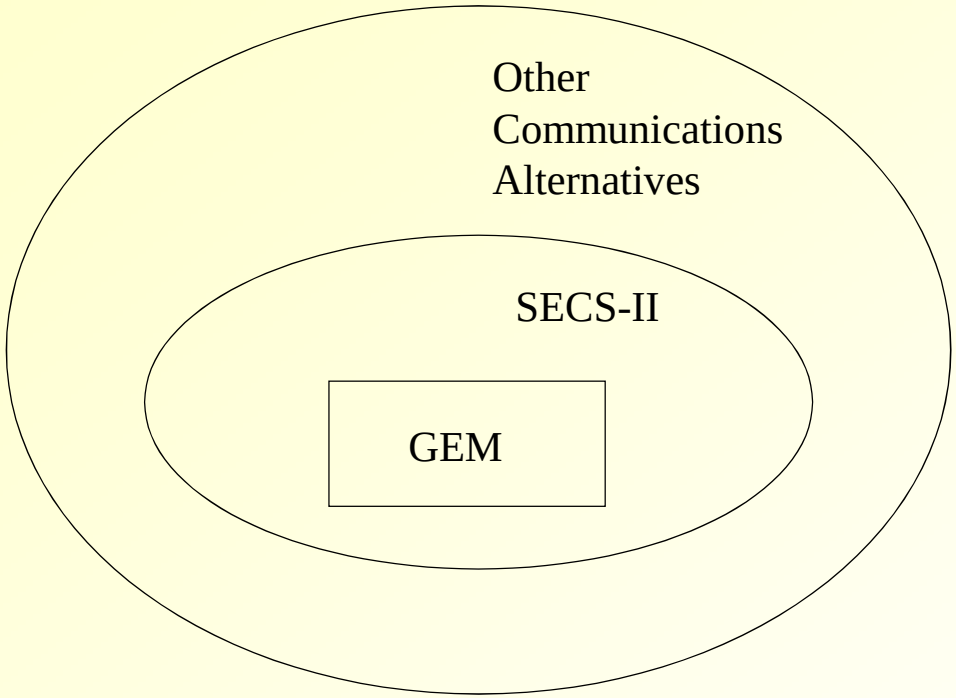
The GEM Standard (continued)

❖ The Driving Forces

By 1987, a large number of semiconductor companies and equipment supplier companies had implemented the SECS-1 and SECS-II standards for factory automation. These standards were important steps towards simplifying factory automation, but it had become clear that additional standards work was needed to solve the following problems:

1. Divergence Among Equipment Implementations of SECS-II. The SECS-II standard specified many capabilities, but almost all were optional. Various equipment types chose to implement different capabilities, and not to implement others.
2. Divergence of User Requirements. Different purchasers of equipment were demanding different SECS-II implementations form the same equipment supplier. Accordingly, the equipment supplier often needed to implement several different versions of SECS communications on the same equipment.
3. Too-Small Minimum Set. The minimum requirements imposed by SECS-II were too limited to provide the equipment capabilities needed for effective factory automation.

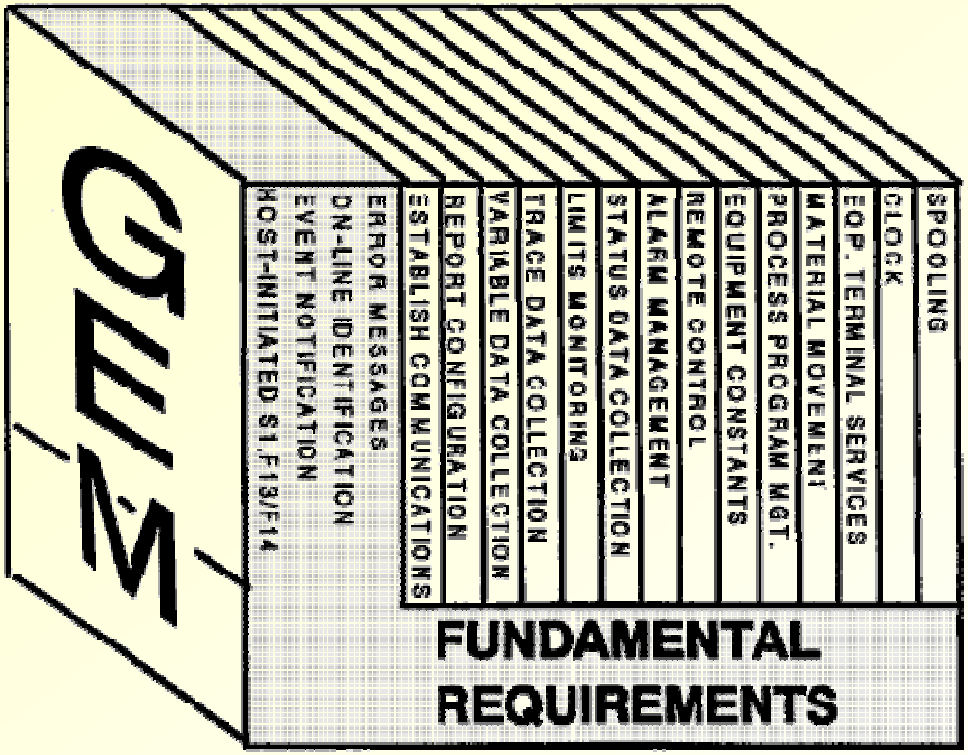
GEM



- GEM defines Equipment behavior as viewed through a SECS-II communications lines.
- GEM is both a specification and a guide. It sets minimum capabilities for the SECS communication on the Equipment and it defines that this communication is tied closely to Equipment activities through a state diagram.

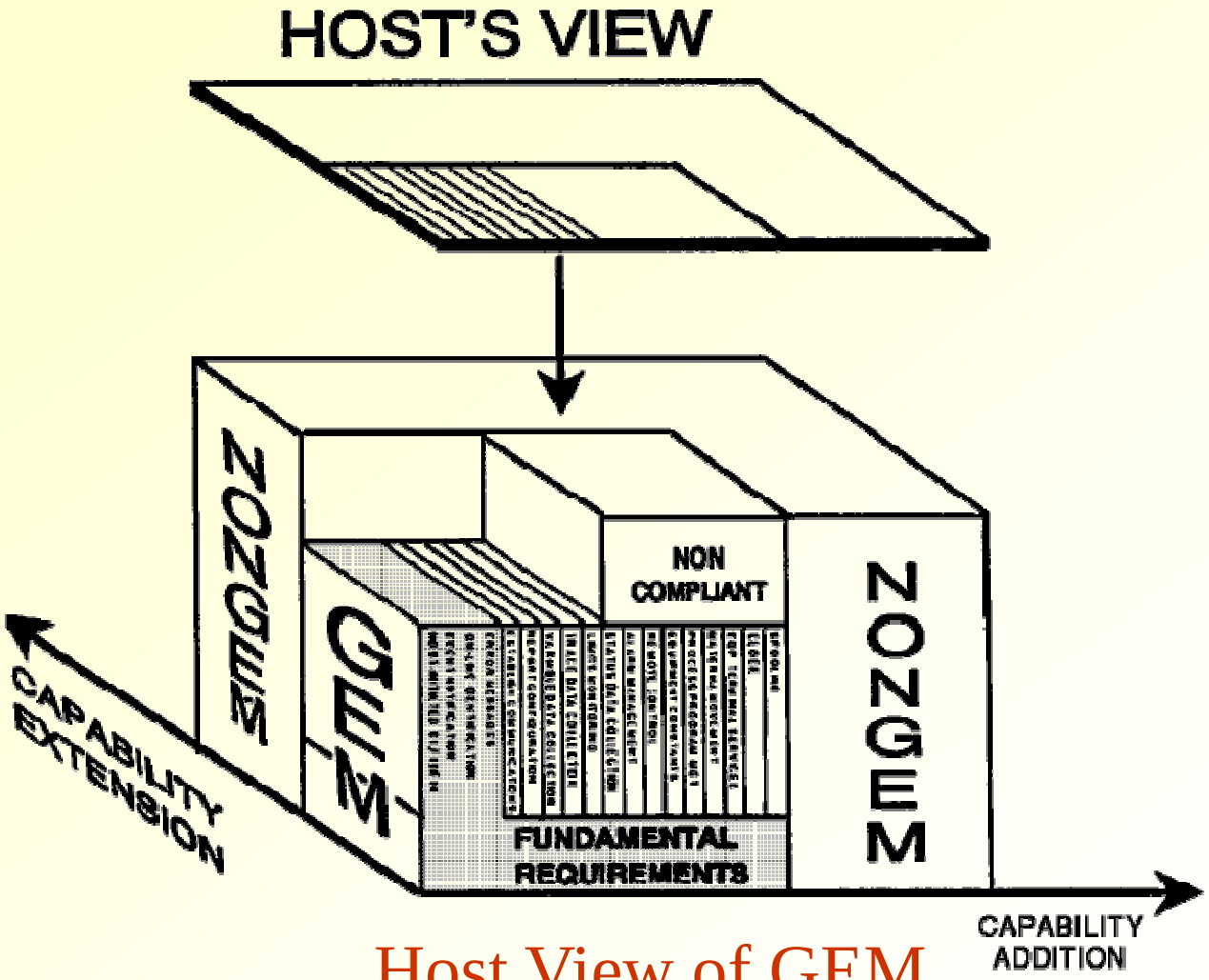
GEM

GEM Requirements and Capabilities



- Vertical text represents capabilities
- Some capabilities are also fundamental requirements.

GEM



Host View of GEM

GEM

The GEM standard contains two types of specifications:

- fundamental GEM requirements and
- requirements pertaining to additional GEM capabilities.

The fundamental GEM requirements form the foundation of the GEM standard. The additional GEM capabilities provide functionality required for some types of factory automation or functionality applicable to specific types of equipment.

GEM

Fundamental GEM Requirements

- State Models
- Equipment Processing States
- Host-Initiated S1F13/F14 Scenario
- Event Notification
- On-line Identification
- Error Messages
- Control (Operator Initiated)
- Documentation

GEM

GEM Capabilities

- Establish Communications
- Event Notification
- Dynamic Event Report Configuration
- Variable Data Collection
- Trace Data 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
- Spooling
- Control (Operator-Initiated)
- Control (Host-Initiated)

GEM

Equipment is **GEM Compliant for a specific GEM capability** if and only if the following three criteria are met:

- The fundamental GEM requirements are satisfied.
- The capability is implemented to conform with all applicable definitions, descriptions, and requirements defined for the capability in this standard.
- The Equipment does not exhibit behavior related to this capability that conflicts with the GEM behavior defined for the capability.

GEM

Equipment is **Full GEM Capable** if and only if it meets the following two criteria:

- The Equipment supplies all the GEM capabilities.
- Every implemented GEM capability is GEM Compliant.

GEM

GEM Compliance Statement

The SECS-II interface documentation provided by an equipment supplier shall address GEM compliance. This documentation shall include a GEM Compliance Statement that accurately indicates for each capability whether it has been implemented and whether it has been implemented in a GEM compliant manner.

GEM

GEM Compliance Statement (continued)

GEM COMPLIANCE STATEMENT		
FUNDAMENTAL GEM REQUIREMENTS	**IMPLEMENTED**	**GEM-COMPLIANT**
State Models	❒ Yes ❒ No	
Equipment Processing States	❒ Yes ❒ No	
Host-Initiated S1= F13/F14 Scenario	❒ Yes ❒ No	❒ Yes*
Event Notification	❒ Yes ❒ No	
On-Line Identification	❒ Yes ❒ No	❒ No
Error Messages	❒ Yes ❒ No	
Documentation	❒ Yes ❒ No	
Control (Operator Initiated)	❒ Yes ❒ No	

GEM Compliance Statement (continued)

ADDITIONAL CAPABILITIES	IMPLEMENTED	GEM-COMPLIANT**
Establish Communications	❒ Yes ❒ No	❒ Yes ❒ No
Dynamic Event Report Configuration	❒ Yes ❒ No	❒ Yes ❒ No
Variable Data Collection	❒ Yes ❒ No	❒ Yes ❒ No
Trace Data Collection	❒ Yes ❒ No	❒ Yes ❒ No
Status Data Collection	❒ Yes ❒ No	❒ Yes ❒ No
Alarm Management	❒ Yes ❒ No	❒ Yes ❒ No
Remote Control	❒ Yes ❒ No	❒ Yes ❒ No
Equipment Constants	❒ Yes ❒ No	❒ Yes ❒ No
Process Program Management	❒ Yes ❒ No	❒ Yes ❒ No
Material Movement	❒ Yes ❒ No	❒ Yes ❒ No
Equipment Terminal Services	❒ Yes ❒ No	❒ Yes ❒ No
Clock	❒ Yes ❒ No	❒ Yes ❒ No
Limits Monitoring	❒ Yes ❒ No	❒ Yes ❒ No
Spooling	❒ Yes ❒ No	❒ Yes ❒ No
Control (Host-Initiated)	❒ Yes ❒ No	❒ Yes ❒ No

* Do not mark **YES** unless all fundamental **GEM** requirements are implemented and **GEM**-compliant.

** Additional capabilities may not be marked **GEM** compliant unless the fundamental **GEM** requirements are **GEM**-compliant.

Fundamental GEM Requirements

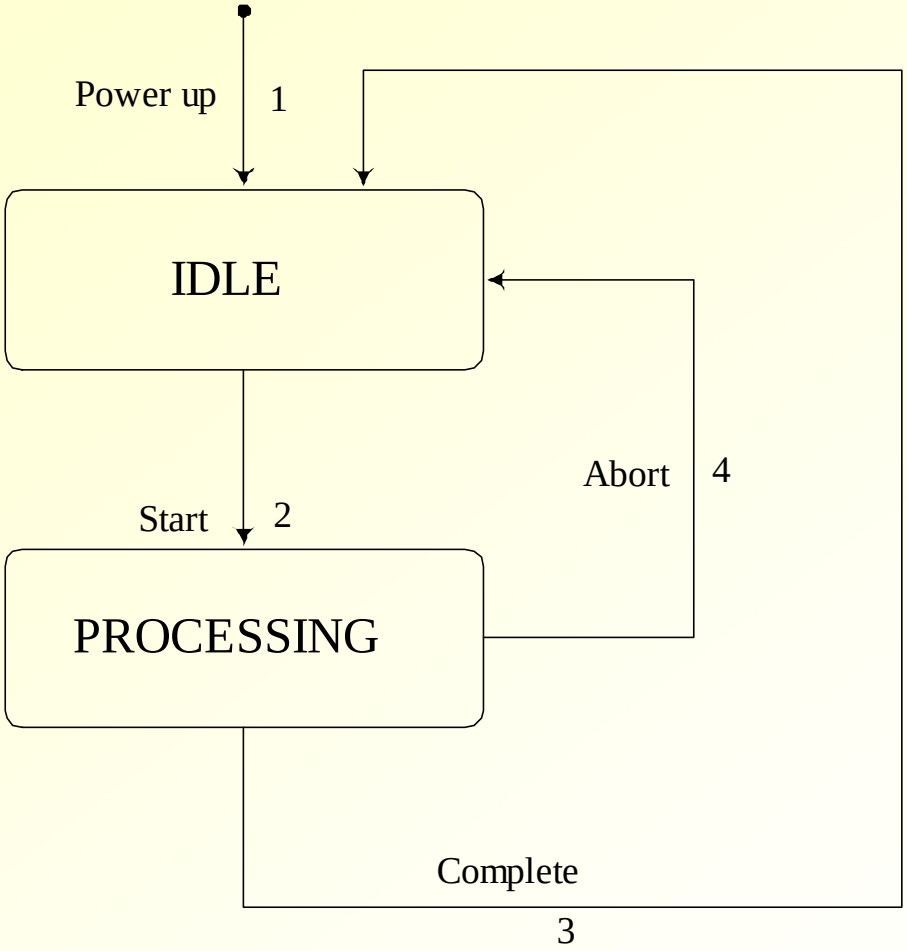
State Models

The Fundamental GEM Requirement is for use a notation called “State Models” or “Finite State Machines”, which provide a clear and unambiguous description of the behavior of equipment.The recommended notation is that invented by David Harel. A Complete description can be fount in:

> Harel, David. “Statecharts: A Visual Formalism for Complex Systems,” Science of Computer Programming 8 (1987), pages 231-274.

Fundamental GEM Requirements

Equipment Processing States



Fundamental GEM Requirements

Equipment Processing States (continued)

This Fundamental GEM Requirement is that Equipment must document its behavior in processing or inspecting material using a “Processing Finite State Machine” notation. The roundcornered rectangles each represent a “State” (in this case “Process Sate”). The arrows represent “Transitions” between states. In the example, the Process Sates are:

IDLE – The Equipment is idle.

PROCESSING – The Equipment is processing material.

Fundamental GEM Requirements

Equipment Processing States (continued)

The Transitions are:

1. **Power Up** – At power-up, the Equipment initializes its Process State to IDLE.
2. **START** – When Process State is IDLE, the Equipment will accept a START command. The Equipment begins processing a lot of material, and Process State transits to PROCESSING.
3. **COMPLETE** – When Process State is PROCESSING, and the Equipment detects that all material in the lot has been processed, the Equipment ceases processing, and Process State transits to IDLE.
4. **ABORT** – When Process State is PROCESSING, the Equipment will accept an ABORT command. The Equipment abruptly ceases processing material, and Process State transits to IDLE.

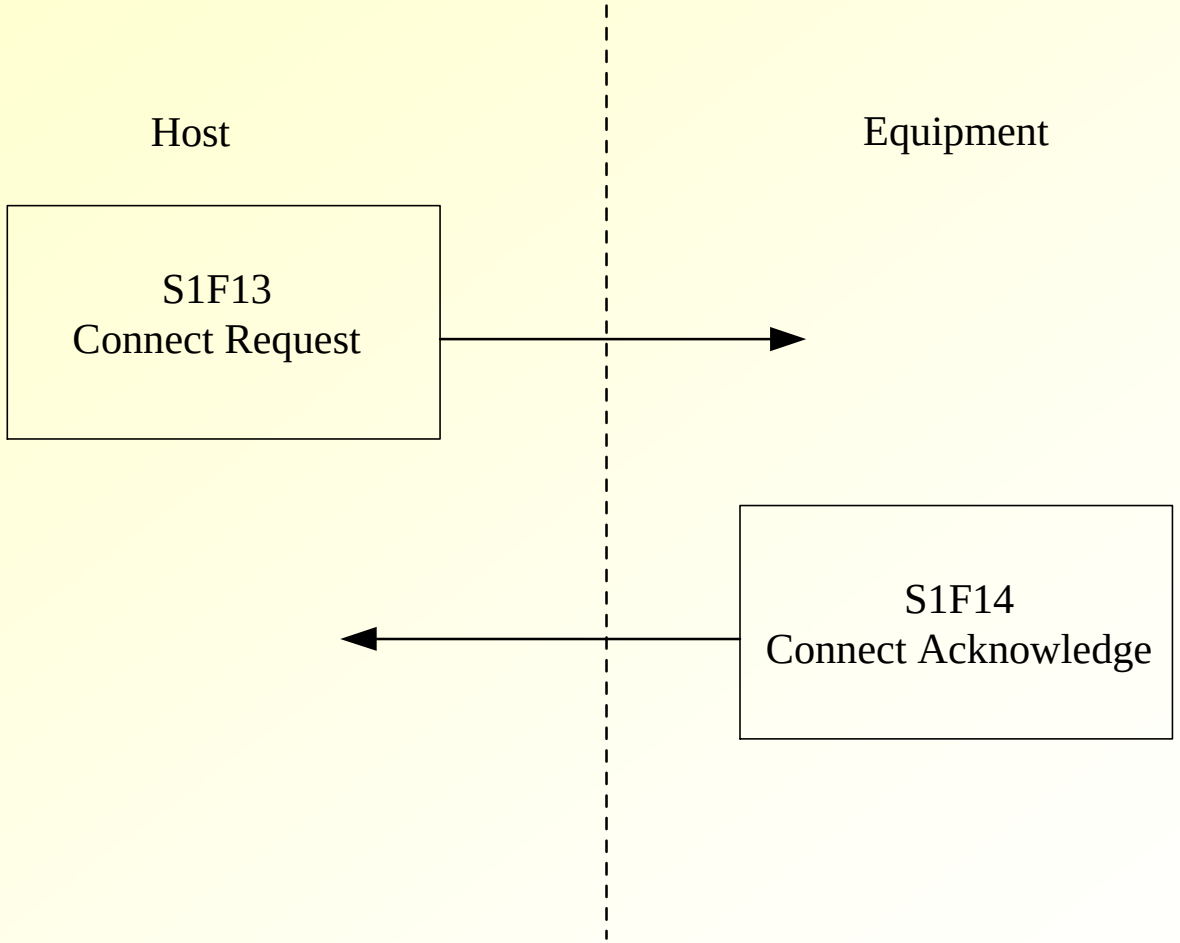
Fundamental GEM Requirements

Equipment Processing States (continued)

The Equipment should define a specific Collection Event ID (CEID) for each Transition (arrow), and should signal that CEID when the transition occurs. The power-up “transition” need not be signaled by an event. Assuming appropriate Report definitions, linking, and enabling, the Equipment will send an Event Report to the Host, so the Host will know that the Transition has occurred.

Fundamental GEM Requirements

Host - Initiated S1F13/F14 Scenario



Fundamental GEM Requirements

Host - Initiated S1F13/F14 Scenario (continued)

This Fundamental GEM Requirement is for Equipment to accept a Connect Request from the Host. The Host sends S1F13 (Connect Request):

S1F13 W *** H ⟶ E**

<L> .

The Equipment sends the S1F14 Connect Acknowledge Reply, with zero COMMACK indicating acceptance of the Connect, and the Equipment identifies itself to the Host by sending its Model Number and Software Revision.

S1F14 *** H ⟵ E**

<L [2]

<B [1] 00> *** COMMACK – Connect Acknowledge**

<L [2]

<A MDLN> *** MDLN = Equipment Model Number**

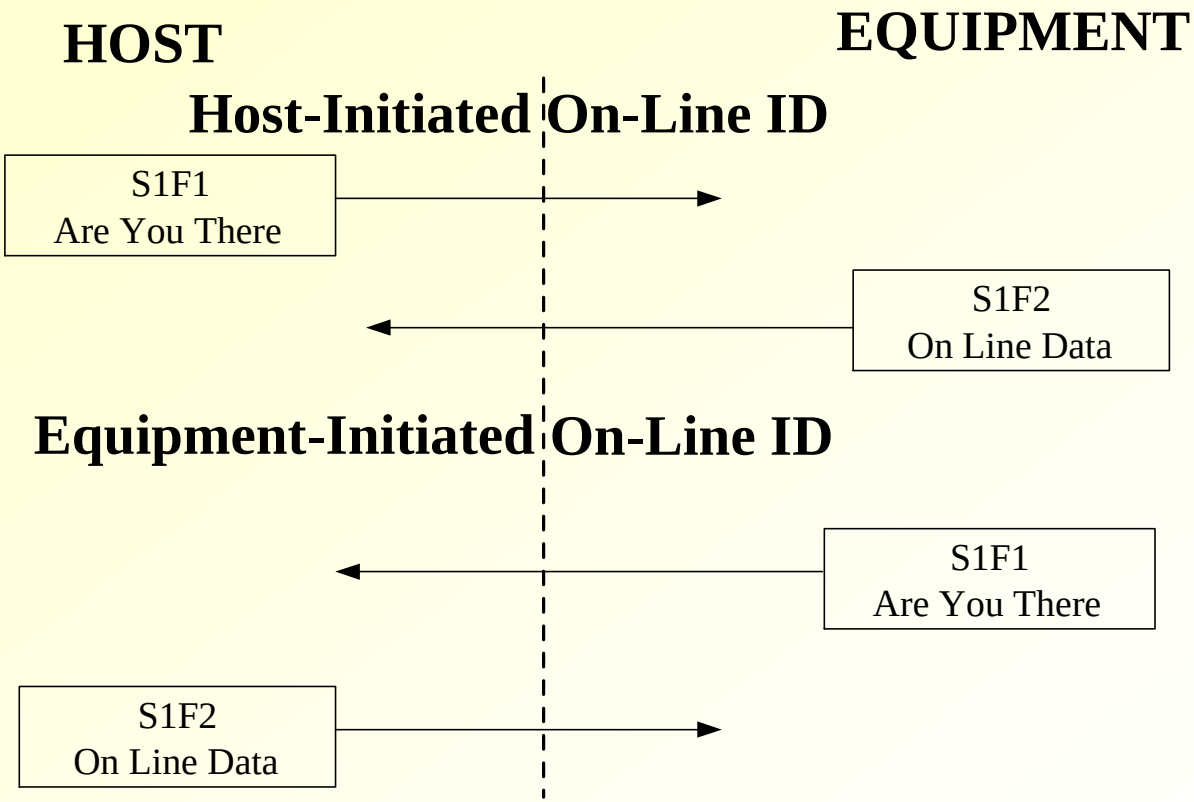
<A SOFTREV> *** SOFTREV – Equipment Software Revision**

>

> .

Fundamental GEM Requirements

On-Line Identification



Fundamental GEM Requirements

On-Line Identification (continued)

The most basic SECS-II transaction is On-Line Identification, sometimes called "Are You There".

❖ Host-to-Equipment

- The host sends S1F1 (Are You There). The equipment responds with S1F1 (On line Data). In S1F2, the data contains the equipment model type and software revision code. The idea here is for the host to be able to verify the equipment type and revision level by examining the contents of the S1F2 message. The host needs to know the Device ID for the equipment to accept the message.
- The Host could use S1F1/S1F2 as a very simple status check of the SECS link after an extended period of no message form the equipment.
- All equipment which supports SECS is required to accept S1F1 and respond with S1F2.

Fundamental GEM Requirements

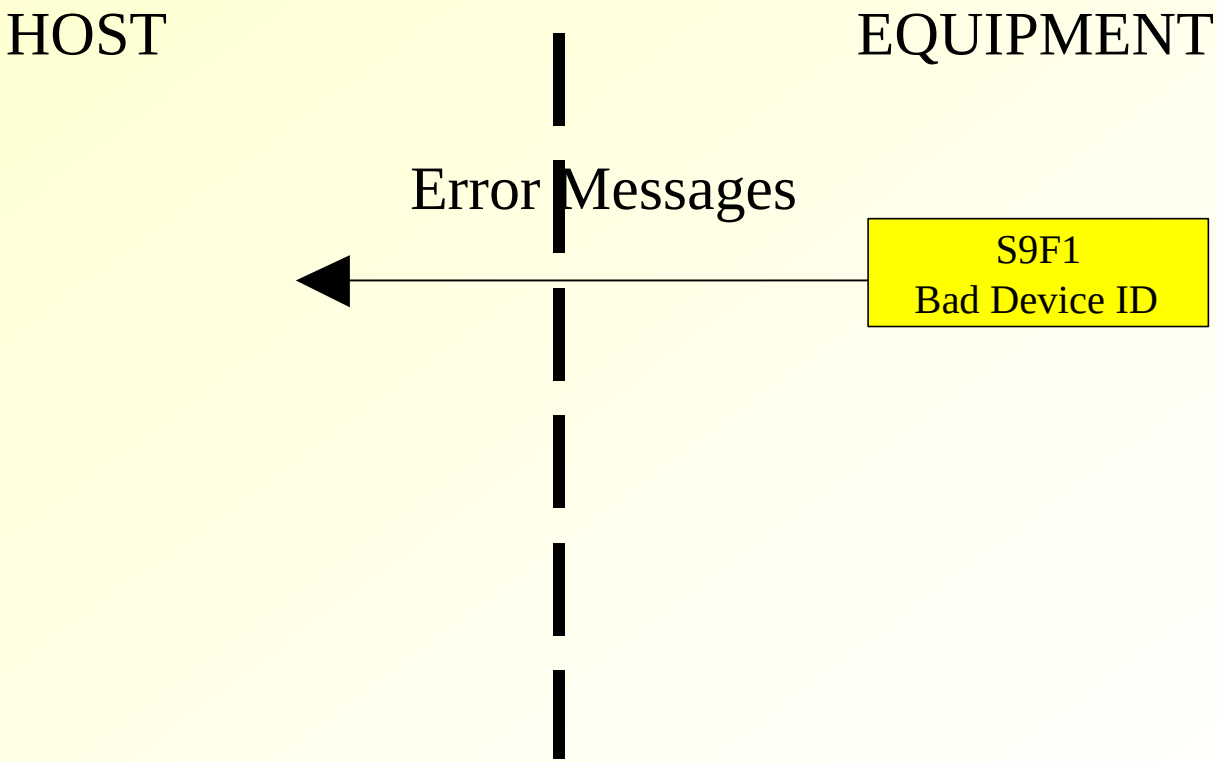
On-Line Identification (continued)

❖ Equipment-to-Host

- Optionally, the equipment may provide for the conversation to go in the other direction. When the equipment sends S1F1 the host responds with S1F2 containing the header only. The equipment cannot determine what type of host is attached to the SECS line.
- The Equipment could use S1F1/S1F2 as a very simple status check of the SECS link after an extended period of no messages from the equipment.

Fundamental GEM Requirements

Error Messages



Fundamental GEM Requirements

Error Messages (continued)

This Fundamental GEM Requirement specifies how the Equipment should report certain SECS protocol errors which occur in messages from the Host. The Equipment uses Stream 9 messages, which are always Equipment to Host. The Equipment can use them to complain to the Host, but the Host can never complain to the Equipment.

Fundamental GEM Requirements

Error Messages (continued)

Message Header Errors

Some of the messages in Stream 9 are used for the equipment to complain about format errors in messages sent from the host. A error message format is:

S9F1 * H←E

< B [10] 8001 0701 8001 00000000 > .

Suppose the host sends a message to the equipment, but the Device ID in the message header is not the Device ID of this equipment. What should the equipment do? The equipment sends S9F1 (Unrecognized Device ID). No host response is required. In the data portion of S9F1, the equipment places the 10-byte header (MHEAD) of the original message, that is the one with the invalid Device ID. By receiving S9F1, the host knows it has made a mistake.

Fundamental GEM Requirements

Error Messages (continued)

Stream 9 includes several messages of this type, each indicating a particular type of error:

S9F1, Unrecognized Device ID

The Device ID in the header does not match the Device of this equipment. The text (MHEAD) of this message shows the header of the host-to-equipment block containing the bad Device ID. Of course, the header of S9F1 contains the correct Device ID, as presently configured in the Equipment.

S9F3, Unrecognized Stream Type

The equipment does not accept messages for this Stream. The text (MHEAD) of this message shows the header of the host-to-equipment block containing the bad Stream.

Fundamental GEM Requirements

Error Messages (continued)

S95, Unrecognized Function Type

This might mean that the equipment does not support this Stream and Function. However, this same error might mean that the host sent a valid Stream/Function, but in a wrong point within a conversation. Examples might be:

- The host send a block of multi-block message out of order.
- The host sent a secondary message (response), but the equipment had not sent the primary message (query).
- The equipment is busy with previous host messages, and cannot accept this additional message at this time.

The text (MHEAD) of this message shows the header of the host-to-equipment block containing the bad Function.

Fundamental GEM Requirements

Error Messages (continued)

S9F7, Illegal Data

The data portion of a message was malformed. The text (MHEAD) of this message shows the header of the host-to-equipment block containing the bad data. Causes might be:

- A Data Item header was malformed.
- The host used a Data Item format (e.g. floating point), which the equipment does not support.
- A required Data Item was missing.
- An unexpected Data Item was present.

S9F11, Data Too Long

For a message with variable-length data, the host sent more data than the equipment could handle. The text (MHEAD) of this message shows the header of the host-to-equipment block containing the bad data.

Fundamental GEM Requirements

Error Messages (continued)

Transaction Timeout (S9F9)

The Equipment sends S9F9 to indicate that a Transaction Timeout (T3 or T4) error has occurred. Suppose the Equipment sends S1F1, but the Host fails to respond with S1F2. Eventually, the Equipment gets the T3 timeout. The Equipment sends the message S9F9, whose data contains the header of the reply (S1F2) that the Equipment was expecting, but never received.

The text (SHEAD) of this message shows the expected header. That is, it shows the header of the host-to-equipment message which the Equipment expected to receive next, but did not receive.

Fundamental GEM Requirements

Error Messages (continued)

Conversation Timeout (S9F13)

- A “Conversation”, as the term is used here, refers to a related exchange of Primary Messages.

- As an example, consider the conversation in which the host sends an Unformatted Process Program to the equipment. The host sends S7F1 and the equipment responds S7F2. At this point, the equipment expects the host to send S7F3, and the equipment starts a “Conversation Timer” to insure that the S7F3 is sent within a reasonable time.

Fundamental GEM Requirements

Error Messages (continued)

Conversation Timeout (S9F13) (continued)

- The conversation timeout is somewhat unlike T1, T2, T3,and T4 timeouts. First, the conversation timeout is application dependent, and is usually considered to be implemented in “application software”, at a higher level than SECS-II. Each conversation has its own separate conversation timeout. There is no “standard” duration for a conversation timeout, since some conversations may be short, while others may require mechanical motion or even operator interaction, and could take a long time.
- If a Conversation Timer times out, the equipment sends S9F13(Conversation Timeout). No host response is required.

Fundamental GEM Requirements

Message Header Errors (continued)

Conversation Timeout(S9F13) (continued)

- Unlike the other Stream 9 messages, the message data in S9F13 does not contain a header. Instead, it shows the expected Stream and Function, and a further data identifier (EDID) which depends on the specific conversation. That is, the S9F13 data shows information identifying the message which the Equipment expected to receive, but did not receive. The following is an example:

```
S9F13                                   * H←E
    <L   [2]
        <A 'S07F03'>                    * MEXP
        <A 'PROG007'>                   * EDID
    > .
```

Fundamental GEM Requirements

Documentation

This Fundamental GEM Requirement is that the Equipment Supplier must provide a detailed specification document for this SECS/GEM interface on his equipment.

GEM Capabilities

GEM Capabilities in three broad categories:

- **Data Collection:** Variable Data Collection, Status Data Collection, Event Notification, Dynamic Event Report Configuration, Trace Data Collection, Alarm Management, Clock, Limits Monitoring.
- **Equipment Control:** Remote Control, Equipment Constants, Process Program Management, Control State (On-Line/Off-Line), Equipment Terminal Services.
- **Protocol:** Establish Communications, Spooling.

GEM Capabilities

❖ Alarm Management

This GEM Capability provides for the orderly reporting of Alarm conditions from the Equipment to the Host.

▪ Defining Alarms

For a given piece of equipment, the Equipment designer defines the set of possible Alarms which can occur. Each Alarm type has:

ALID Alarm ID. This is a unique name for the alarm. ALID must be Unsigned (Ux) format, so alarms are named “1”, “2”, and so on.

GEM Capabilities

Defining Alarms (continued)

ALCD Alarm Code. This is a one-byte binary item, and it is split into two parts:

Alarm State – The high-order bit of ALCD is “1” if the Alarm is currently ON (Unsafe), or “0” if the Alarm is currently OFF (Safe).

Severity – The low-order 7 bits are not meaningful for GEM. However, for compatibility with non-GEM hosts using older versions of SECS-II, it is useful if these bits contain a code indicating the severity of the alarm – personal safety, process error, and so on.

GEM Capabilities

Defining Alarms (continued)

ALTX Alarm Text. This is a text message describing the alarm. An example might be “TEMPERATURE TOO HIGH”. The maximum length is 40 bytes.

ALED Alarm Enable/Disable State. Each alarm (ALID) may be ENABLED or DISABLED for purposes of the Equipment reporting this particular alarm type to the Host.

CEID for Going ON

A Collection Event ID which the Equipment will signal when this Alarm State transits to ON.

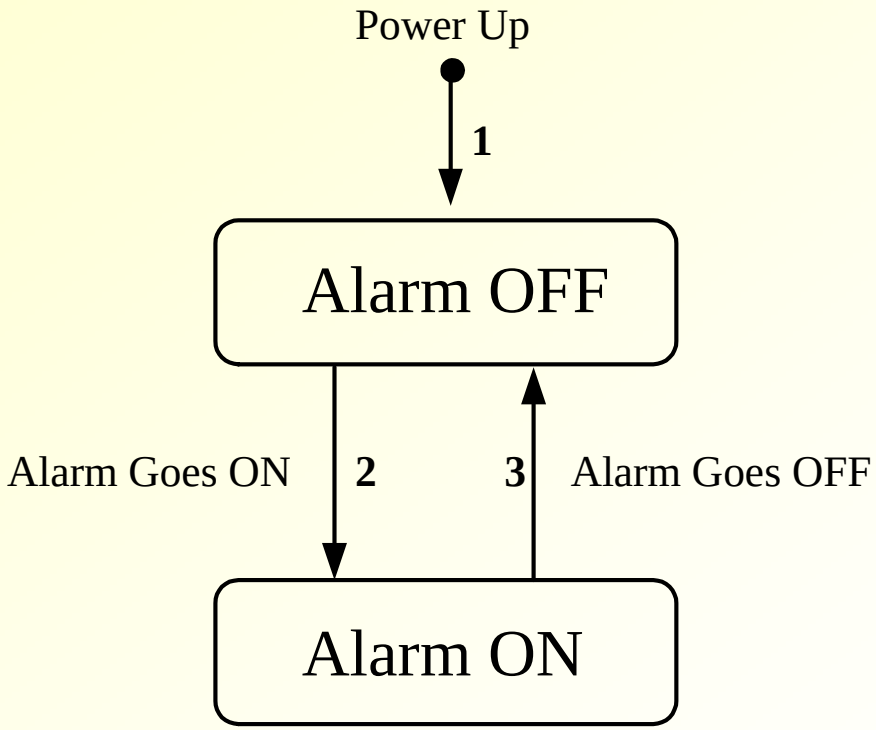
CEID for Going OFF

A Collection Event ID which the Equipment will signal when this Alarm State transits to OFF.

GEM Capabilities

Alarm Management (continued)

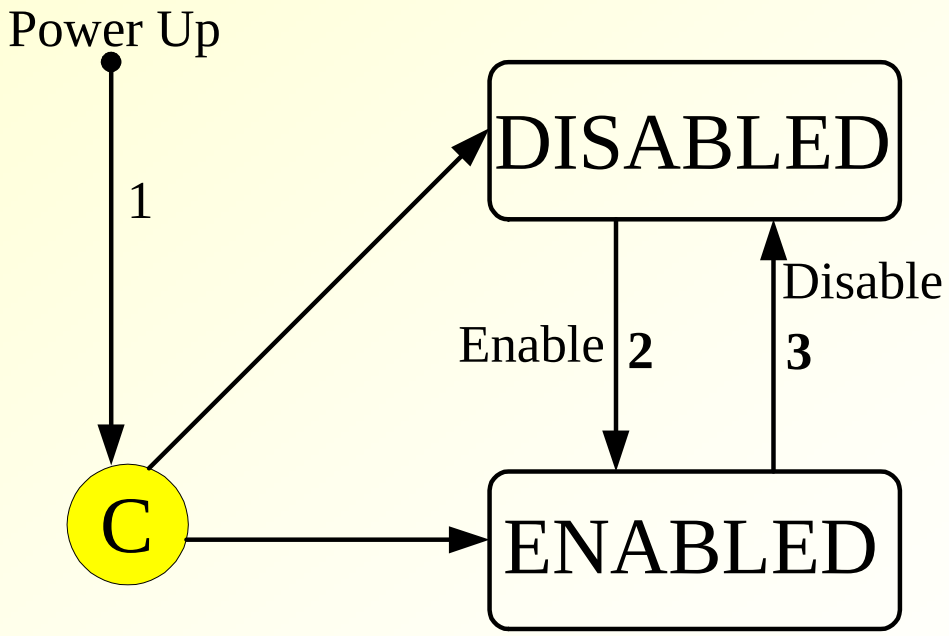
Alarm State Machine



GEM Capabilities

Alarm Management (continued)

Alarm Enable/Disable State Machine



GEM Capabilities

Alarm Management (continued)

Alarm State – Finite State Machine

Each Alarm (ALID) has an Alarm State finite state machine. The Alarm States are:

OFF – The Alarm is in the OFF (Safe) state.

ON – The Alarm is in the ON (Unsafe) state.

The sate transition are:

1. At power-up, an Alarm is normally in the OFF state, unless a condition existing at power-up causes the Alarm to be in the ON state. It may be good practice for Equipment to initialize Alarm State to OFF, and send an initial Alarm Going ON Report to the Host for each alarm condition existing at power-up.
2. Alarm goes ON. When Alarm State is OFF, the Equipment may recognize that an alarm condition occurs, and Alarm State transits to ON.
3. Alarm goes OFF. When Alarm State is ON, the Equipment may recognize that an alarm condition disappears, and Alarm State transits to OFF.

GEM Capabilities

Alarm Management (continued)

Alarm Enable/Disable-Finite State Machine

Each Alarm (ALID) has an Alarm Enable/Disable finite state machine. The Alarm Enabled/Disabled states are:

ENABLED – The Equipment will send S5F1 to the Host when any Alarm State transition occurs for this ALID.

DISABLED – The Equipment will not send S5F1 to the Host when an Alarm State transition occurs for this ALID.

GEM Capabilities

Alarm Management (continued)

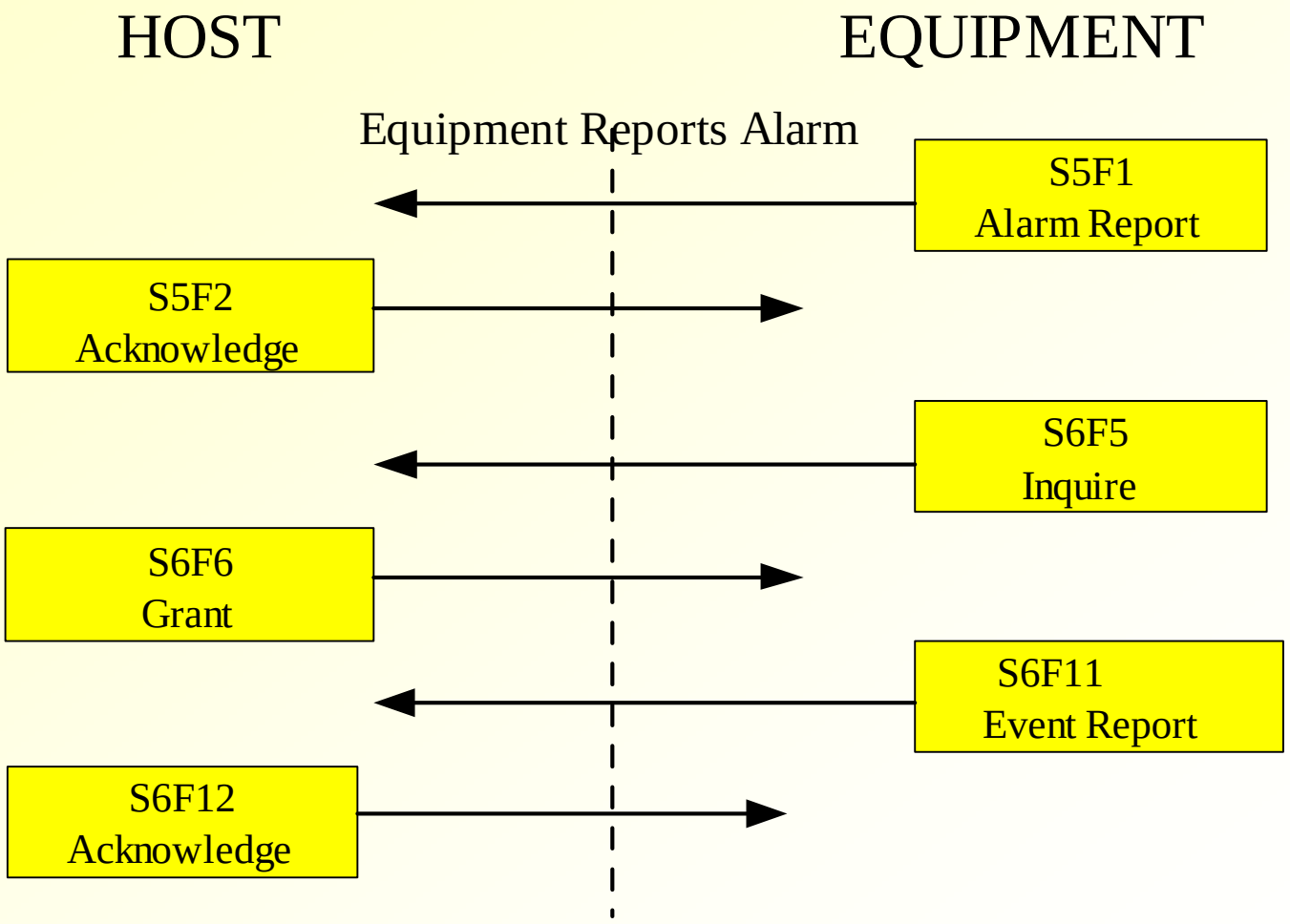
Alarm Enable/Disable-Finite State Machine (continued)

The state transitions are:

1. At power-up. Equipment should restore each Alarm (ALID) to the Alarm Enabled/Disable state which existed when the Equipment was powered off.
2. Enable Alarm. When Alarm Enable/Disable State is DISABLE, the Host can send S5F3 to enable the Alarm, and Alarm Enable/Disable State transits to ENABLED.
3. Disable Alarm. When Alarm Enable/Disable State is ENABLED, the Host can send S5F3 to disable the Alarm, and Alarm Enable/Disable State transits to DISABLED.

GEM Capabilities

Alarm Management (continued)



GEM Capabilities

Alarm Management (continued)

Equipment Reports Alarm

When an Alarm State transition occurs at the Equipment, the Equipment reports it to the Host in two ways – by sending S5F1 (Alarm Report Send) and also by sending and Event Report (normally S6F11).

The Equipment sends S5F1 (Alarm Report Send).

S5F1 [w] * H⟵ E

<L [3]

<B [1] ALCD> * Alarm ON/OFF and Severity

<Ux ALID> * Alarm ID

<A ALTX> * Alarm Text

>.

If equipment sets the W-Bit to “1” in the header of the S5F1 message, then the host must respond with S5F2 (Alarm Report Acknowledge). Otherwise, no host response is required. It’s probably simpler for the equipment not to require the S5F2 response, since it’s not clear exactly what the equipment should do if the host refuses to accept an alarm.

GEM Capabilities

Alarm Management (continued)

Equipment Reports Alarm (continued)

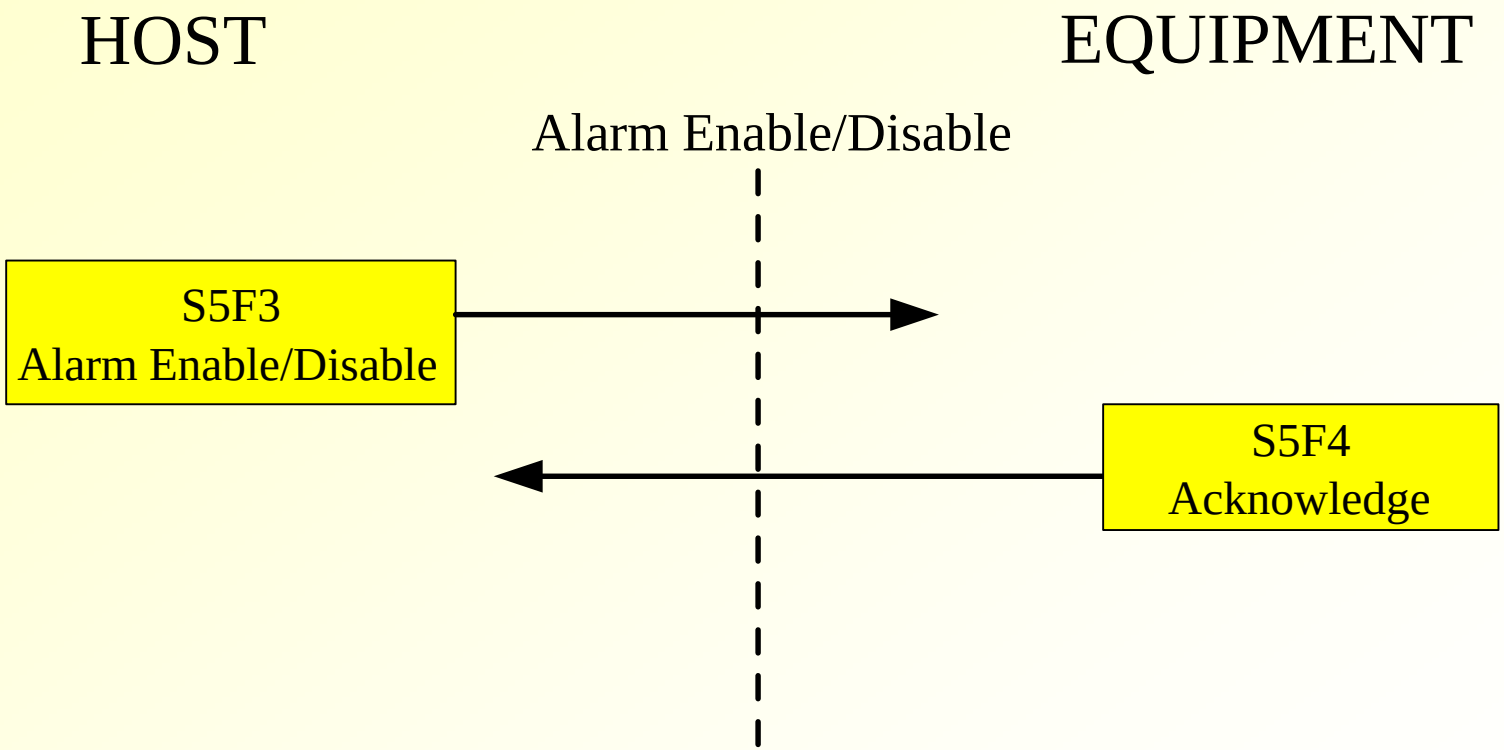
S5F2 * H⟶ E

<B [1] 00>. * ACKC5

- The Equipment sends the Alarm Event (S6F1) as described under Event Reporting. The CEID used is the appropriate one defined for the Alarm State transition which has occurred. As for any other Event Report, the Host may define and link a Report to this CEID, and may enable or disable the CEID for reporting.
- The Equipment must report both Alarm Goes ON and Alarm Goes OFF transitions. The host can then determine how long the alarm condition existed. For Alarm Goes ON, the S5F1 will have ALCD high bit = 1, and S6F11 will have the CEID for Going-ON. For Alarm Goes OFF, the S5F1 will have ALCD high bit = 0, and the S6F11 will have the CEID for Going-OFF.

GEM Capabilities

Alarm Management (continued)



GEM Capabilities

Alarm Management (continued)

Enable/Disable Alarm

- The Host can instruct the Equipment about of the possible alarms the equipment should report. Some equipment defines hundreds of different alarms (ALID's), so the Host may find it useful to "condition" the equipment to send only those alarms in which the host is interested.
- The host sends S5F3 (Enable/Disable Alarm Send). In S5F3, the host specifies the desired ALID, and a code to enable or disable that specific alarm. With S5F3, the reply is optional.

S5F3 [W]	*H⟶E
<L [2]	
<B [1] 80>	*ALED –Enable Code
<Ux 1>	*ALID – Alarm ID
>.	

GEM Capabilities

Alarm Management (continued)

Enable/Disable Alarm (continued)

The host can send S5F3 with an alarm enable/disable code (ALED), but with a zero-length ALID. This indicates that the enable/disable applies to all alarms in the equipment.

S5F3 W * H ⟶ E

<L [2]

<B [1] 00> * ALED - Disable Code

<Ux> * ALID – All Alarms

>.

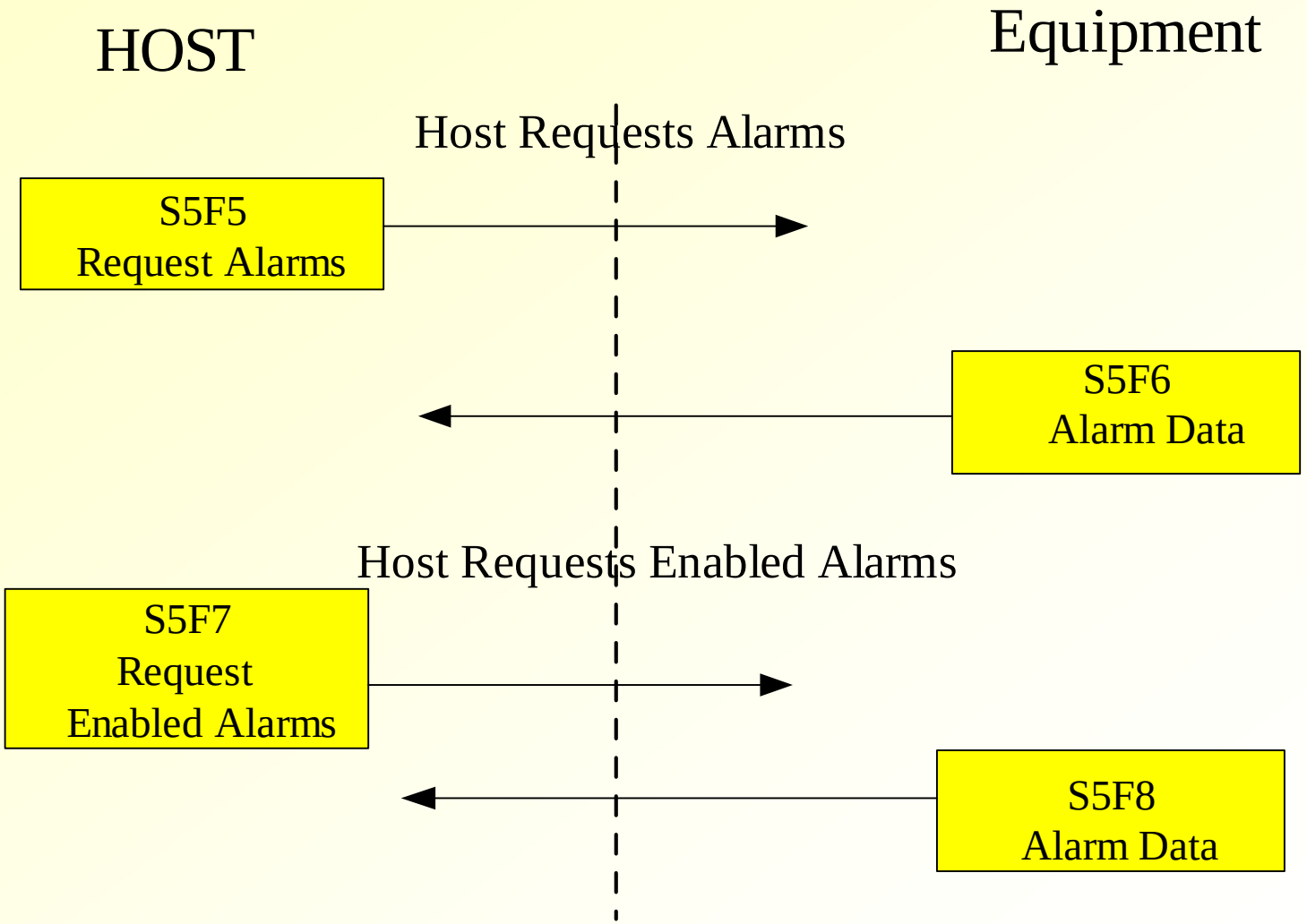
If the host sets the W-Bit to “1” in the header of S5F3, then the equipment must respond with S5F4 (Enable/Disable Alarm Acknowledge). Otherwise, no equipment response is required.

S5F4 * H ⟵ E

<B [1] ACKC5>.

GEM Capabilities

Alarm Management (continued)



GEM Capabilities

Alarm Management (continued)

Host Requests Alarms

Another way to report alarms is on host request. The host sends S5F5 (List Alarms Request), to request alarm status. In S5F5, the host specifies an array of ALID's, indicating the particular alarms of interest.

S5F5	W	* H⟶ E
	<Ux 1 2 3>.	* ALID – Desired Alarms

The request all possible alarms, the Host specifies a zero-length item in S5F5.

S5F5	W	* H⟵E
	<U2> .	* ALID – Desired Alarms

GEM Capabilities

Alarm Management (continued)

Host Requests Alarms (continued)

The equipment responds with S5F6 (List Alarm Data), which contains data for all the ALID's requested. Note that S5F6 reports data for each alarm regardless of whether its Alarm State's "ON" or "OFF". In S5F6, for each alarm, the high-order but of ALCD tells the host whether this alarm is "on" or "off". Also, specified alarms are included in S5F6 whether or not the alarms are "enabled" or disabled".

```
S5F6                                    * H⟶E
        <L
            <L [3]
                <B [1] xx>              * ALCD – On/Off and Severity
                <Ux ALID>               * ALID  – Alarm ID
                 <A ALTX>               * ALTX – Alarm Text
                >
               …
            >.
```

GEM Capabilities

Alarm Management (continued)

List Enabled Alarms

The Host may find S5F6 clumsy, since it includes disabled alarms, which probably are not of interest to the Host. S5F7 (List Enabled Alarms) requests the equipment to send S5F8, which is similar to S5F6, but contains only alarms which are currently enabled. Alarms which are disabled do not appear in S5F8. S5F7 is a simple message consisting only of the header (no text), so it always requests “all” enabled alarms, and cannot request individual alarms by ALID.

S5F7 W. * H⟶ E

S5F8 * H⟵ E

```
<L
    <L [3]
        <B [1]  xx>          * ALCD – On/Off and Severity
        <Ux  ALID>           * ALID – Alarm ID
        <A    ALTX>          * ALTX – Alarm Text
        >
    ….
    >.
```

GEM Capabilities

Alarm Management (continued)

Variables for Alarms

GEM requires Equipment to provide the following Status Variables relating to Alarms:

AlarmsEnabled – This Status Variable shows the Alarm IDs which are currently enabled for reporting to the Host (see S5F3). The format of the Status Variable is a list of Alarm IDs, as follows:

```
<L
        <Ux   ALID>          * ALID – Alarm ID
        …
        >
```

AlarmsOn – This Status Variable shows the Alarm IDs which are currently in the Alarm ON state. The format of the Status Variable is a list of Alarm IDs, as follows:

```
<L
        <Ux ALID>            *ALID – ALarm ID
        …
        >
```

GEM Capabilities

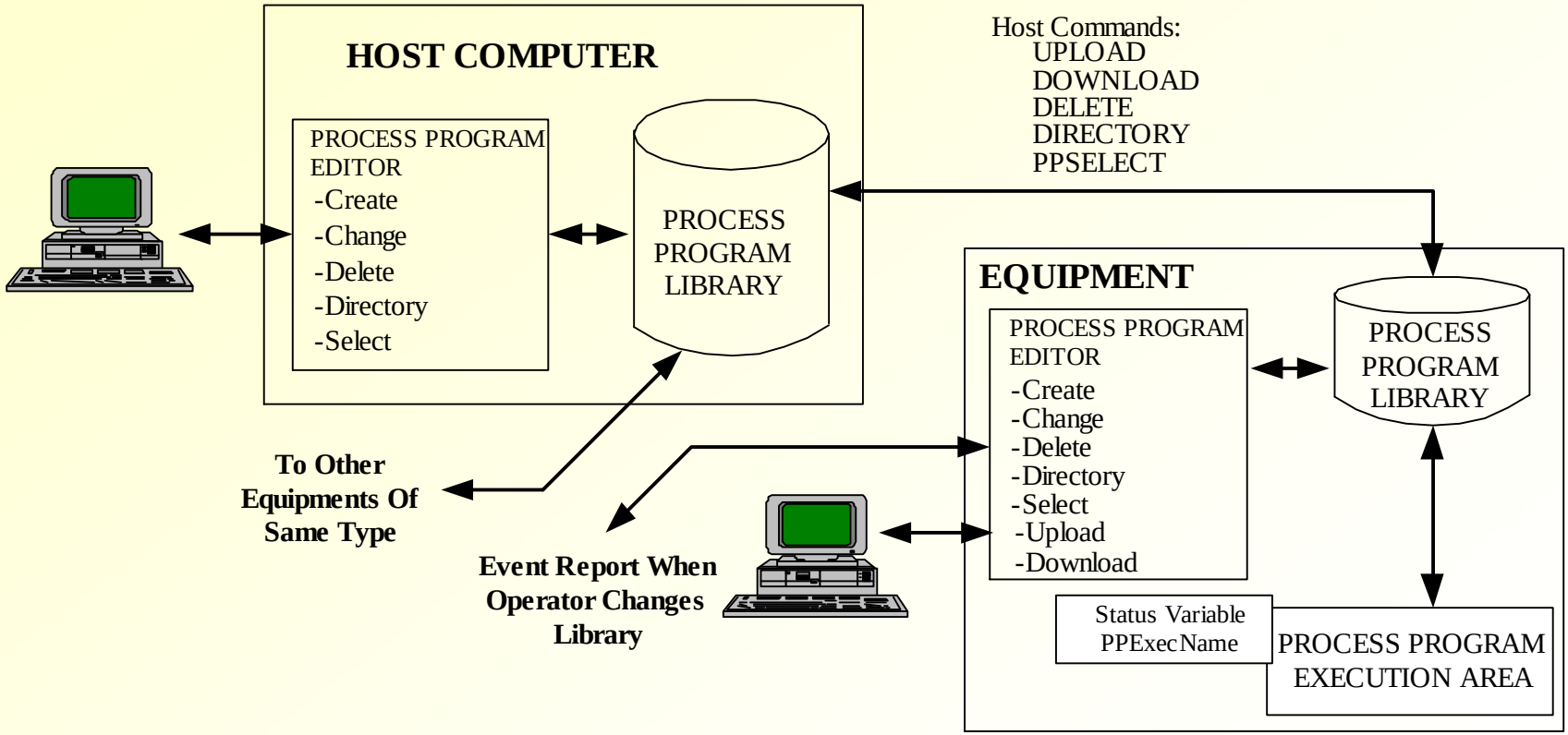
Alarm Management (continued)

Debouncing Alarms

- A common equipment problem is determining the frequency of reporting alarms to the host. Typically, equipment wants to report alarms immediately when they occur. However, imagine an alarm caused by an over-temperature condition, and then imagine a situation where the actual temperature was "wiggling" right at the alarm threshold. The equipment could generate hundreds or even thousands of "alarm on/off" messages every second. These could clog up in the equipment by overrunning the speed of the SECS link. Worse, since the equipment is the SECS-I "master", a flood of alarm messages from the equipment could block off even a disable command from the host.
- Well-designed equipment will "debounce" alarms reported to the host, so the overload conditions cannot occur. For example, equipment might limit alarm reporting to one alarm every 5 seconds.

GEM Capabilities

Process Program Management



GEM Capabilities

Process Program Management (continued)

- In the old days, the equipment operator had manually to set dozens (sometimes hundreds) of dials and switches to the desired setpoints before running material. Any operator error caused errors in processing, so repeatability was poor.
- Today, most Equipment uses Process Programs (sometimes called “recipes”) to control processing operation. A Process Program is simply a predefined set of all the setpoints and options which the operator used to set by hand.

GEM Capabilities

Process Program Management (continued)

- A process Program is an important unit of information exchange between the Host and the Equipment. In a simplified view, a factory Host computer might determine that a particular lot of wafers, of a particular type and at a particular stage of processing, should be scheduled for processing at a particular unit of Equipment. The Host would like to select the required Process Program form its central library, download the Process Program to the Equipment, and then simply stand back and let it happen. The Host would like to be sure that processing will be repeatable. That is, the same recipe will give the same results day after day.

GEM Capabilities

Process Program Management (continued)

What is a Process Program?

A Process Program should contain information to control the process. For a diffusion furnace, this would certainly include desired heater temperatures, gas flow rates, process sequences, and so forth. For an inspection station, the Process Program might specify number of points to inspect, pattern recognition information, reporting options, and so forth.

1. Specify the Entire Process. The Process Program should specify all aspects which control how the Equipment processes material.

   Host System get annoyed with equipment which requires the Host select a whole bunch of different “fragments” of Process Programs.

2. Interchangeable. The Host wants to be able to upload a given Process Program from one unit of Equipment, then download that Process Program to another unit of the same Equipment type, and get the same processing results. This means that Process Programs should not contain unit-specific “calibration” information, which might prevent interchange.
3. Useable for any Lot. This means the Process Program should not contain wafer counts, Lot ID, and other information which is properly part of the Lot.

GEM Capabilities

Process Program Management (continued)

Creating Process Programs

- Usually, the Equipment provides a method for the operator to create and modify Process Programs on the equipment itself.
- Sometimes, the Host may also provide an editor which can create and modify process programs on the Host.
- The host and the equipment can exchange Unformatted Process Programs using SECS. Commonly, we refer to the transfer to a Process Program from the equipment to the Host as an “Upload”. The reverse direction, a transfer from the host to the equipment, is referred to as a “Download”.

GEM Capabilities

Process Program Management (continued)

Defining Process Programs

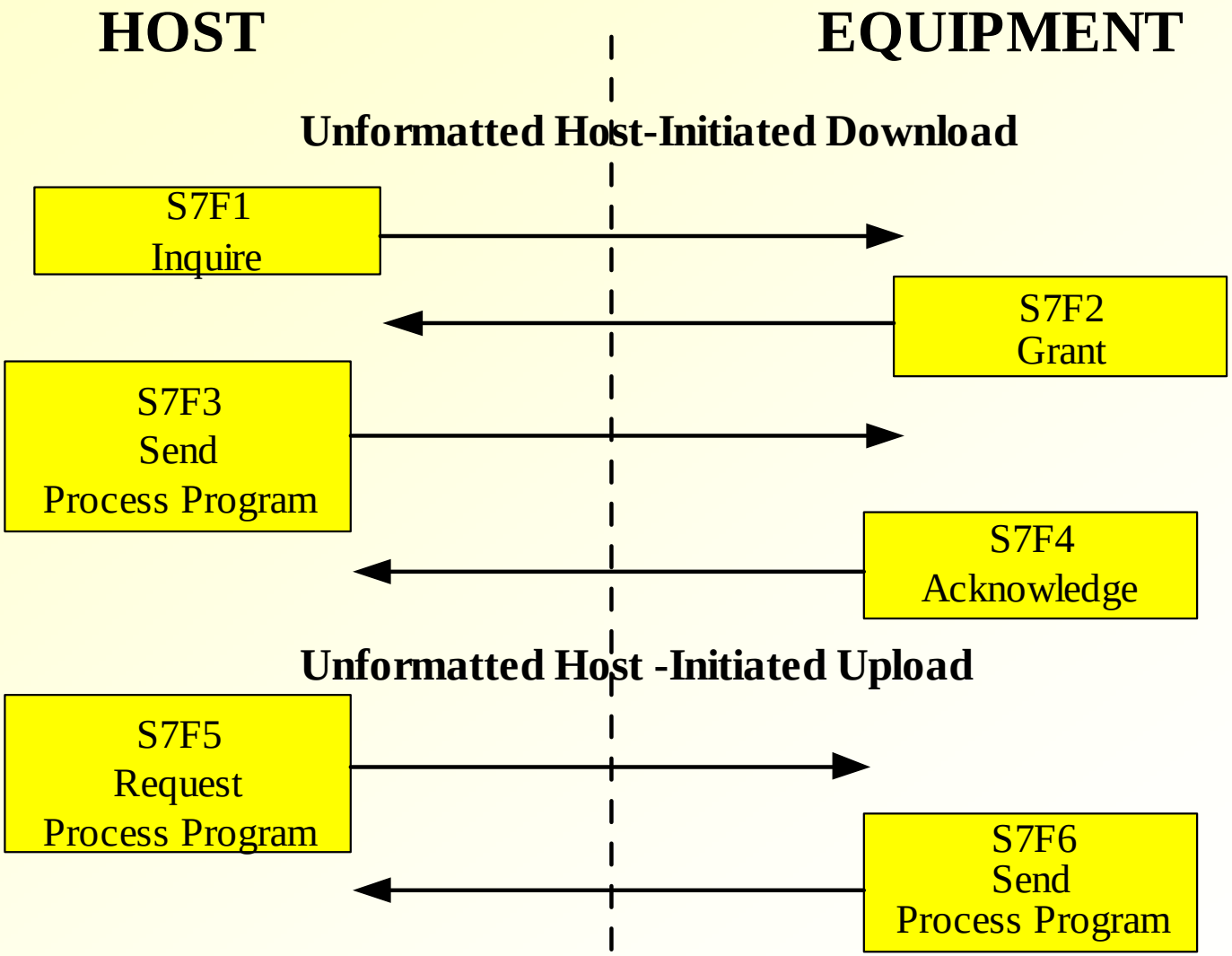
Each Unformatted Process Program has:

PPID Process Program ID. This is a unique name for this Process Program. PPID should be in ASCII format, and may be from 1 to 16 bytes long.

PPBODY Process Program Body. This is the Process Program itself. It consists of a string of bytes in Binary or ASCII format. It may be quite large, sometimes thousands of bytes in length. The internal structure of PPBODY is understood by the equipment, but typically cannot be understood by the host computer.

GEM Capabilities

Unformatted Process Programs



GEM Capabilities

Unformatted Process Programs (continued)

The simpler SECS format used is “Unformatted Process Program”. This term means that the internal format of a Process Program Body is not known by the host computer. The host computer views an Unformatted Process Program as a “black box”, or in more precise terms as a particular string of binary bytes. Notice that even though the host cannot understand the actual internal structure of an Unformatted Process Program, it can nevertheless accept an “upload” of the program, store that program away, and later “download” it back to the equipment exactly as it was uploaded.

GEM Capabilities

Unformatted Process Programs (continued)

Host Initiated Download

Let's assume the host wants to download an Unformatted Process Program to the equipment, and the Process Program is short enough to fit within one SECS message block. The host sends the recipe using S7F3 (Process Program Send).

S7F3 W * H⟶E

<L [2]

<A 'PROG7'> * PPID – Process Program ID

<B PPBODY> * PPBODY – The Process Program

> .

The equipment responds with S7F4 (Process Program Acknowledge)

S7F4 * H⟵ E

<B [1] ACKC7> * ACKC7 (Positive or Negative response)

GEM Capabilities

Unformatted Process Programs (continued)

Host Initiated Download (Large)

For some equipment, process programs may be large, and S7F3 (Process Program Send) may require a multi-block message. This requires an Inquire, Grant, Send. Acknowledge type of SECS conversation.

The host sends S7F1 (Process Program Load Inquire) to ask permission to send a Process Program to the equipment. Included in this request is the length of the program, so the equipment can determine whether it has sufficient storage space to receive it:

```
S7F1   W                          *  H⟶ E
    <L [2]
        <A 'PROG7'>               * PPID (Process Program ID)
        <U4 357>                  * LENGTH (Length in byte of Process Pgm)
        >.
```

The equipment responds with S7F2 (Process Program Grant) to indicate whether it can accept the process program:

```
    S7F2                          * H ⟵E
        <B [1]   PPGNT> .         * PPGNT (Positive or Negative response)
```

GEM Capabilities

Unformatted Process Programs (continued)

Host Initiated Download (Large) (continued)

Assuming the equipment has responded positively, the host now sends the recipe using S7F3 (Process Program Send), which may be a multi-block message:

S7F3 W * H ⟶ E

<L [2]

<A 'PROG7'> * PPID (Process Program ID)

<B PPBODY> * PPBODY (The actual Process Pgm)

> .

After the host has sent all blocks of S7F3, the equipment responds with S7F4 (Process Program Acknowledge):

S7F4 * H ⟵ E

<B [1] ACKC7> * ACKC7 (Positive or Negative response)

GEM Capabilities

Unformatted Process Programs (continued)

Host Initiated Upload

The Host can Upload a Process Program from the Equipment's library. The Host sends S7F5, specifying the desired PPID:

S7F5 W	* H ⟶ E
<A PPID> .	* PPID – Process Program ID

The equipment sends the S7F6 reply message to upload the Process Program to the Host:

S7F6	* H ⟵ E
<L [2]	
<A PPID>	* PPID –Process Program ID
<PPBODY>	* PPBODY –Process Program Body
> .	

GEM Capabilities

Unformatted Process Programs (continued)

Equipment Initiated Download

In some factories, it may be desirable for the Equipment to requests a Download. The Equipment sends S7F5 (Process Program Request), specifying the desired PPID. The host sends the requested Process Program as S7F6 (Process Program Data).

S7F5 W	* H ⟶ E
<A PPID> .	* PPID –Process Program ID
S7F6	* H ⟵ E
<L [2]	
<A PPID>	* PPID – Process Program ID
<PPBODY>	* PPBODY – Process Program Body
> .	

GEM Capabilities

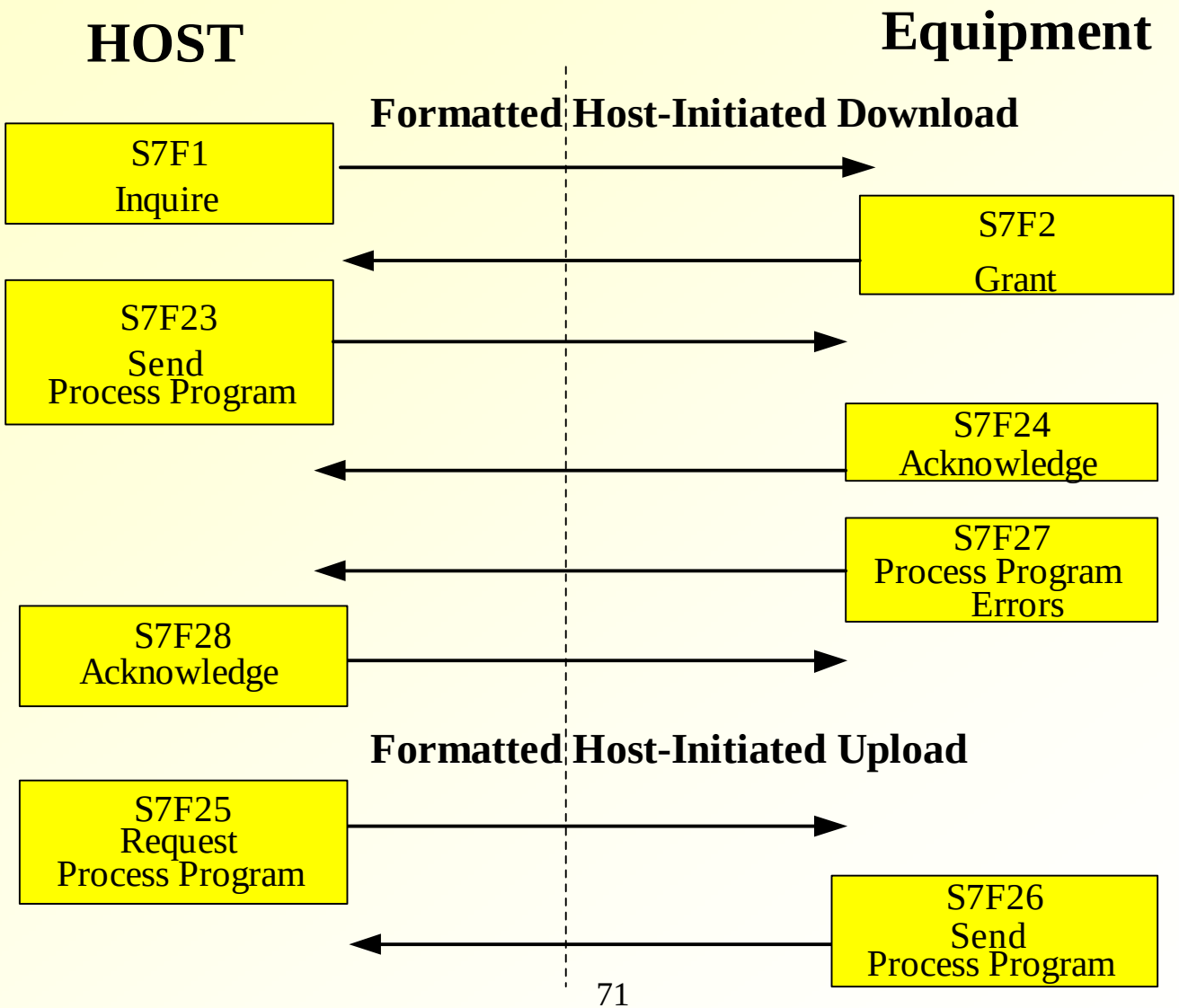
Unformatted Process Programs (continued)

Equipment Initiated Upload

The conversation (S7F1/S7F2/S7F3/S7F4) can be turned around, so that the equipment initiates uploading a Process Program to the host.

GEM Capabilities

Formatted Process Programs



GEM Capabilities

Formatted Process Programs (continued)

- Conceptually, “Unformatted” and “Formatted” Process Program are just two different methods for representing the same Process Program (“Recipe”). The difference is that the Formatted Process Program opens up the mysteries of the internal program structure to the host computer.
- In theory, this makes it possible for host computers to implement generalized “Recipe Editor” software to create, display and modify Process Programs for a wide variety of different equipment. Generalized host “editor” software of this type is of some interest to Process Engineers, who otherwise must create process programs on each piece of equipment, using different procedures for each equipment type.
- In practice, very few types of equipment support Formatted Process Programs, so their value remains theoretical.
- A formatted Process Program is typically much larger than the same program expressed as an Unformatted Process Program.

GEM Capabilities

Formatted Process Programs (continued)

Defining Formatted Process Programs

- **PPID** Process Program ID. This is the same as used for Unformatted Process programs.
- **MDLN** Model Number for the Equipment, as returned in S1F2.
- **SOFTREV** Software Revision Level for the Equipment, as returned in S1F2. The idea here is that when a Formatted Process Program is created, it is "stamped" with the current MDLN and SOFTREV of the equipment. Later, if the equipment is upgraded in a way which changes the requirements of Process Programs, the equipment's MDLN and/or SOFTREV would be updated.

GEM Capabilities

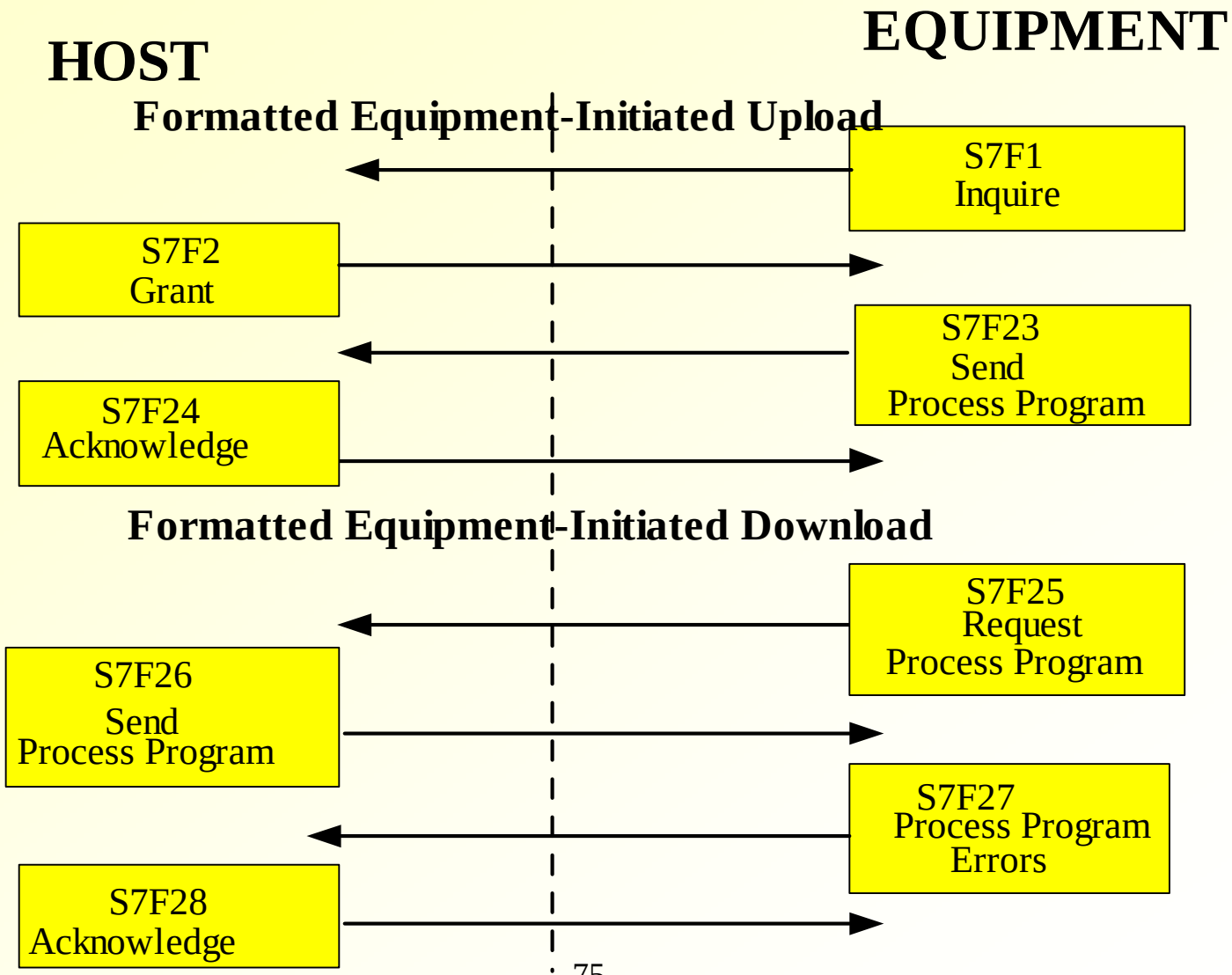
Formatted Process Programs (continued)

Defining Formatted Process Programs (continued)

- **Body** The main body of a Formatted Process Program consists of a series of “command steps”. Each command step consists of a Command Code (CCODE), followed by a list of Process Parameters (PPARM). The particular commands and parameters used, of course, vary widely among different equipment types. However the same general format applies.

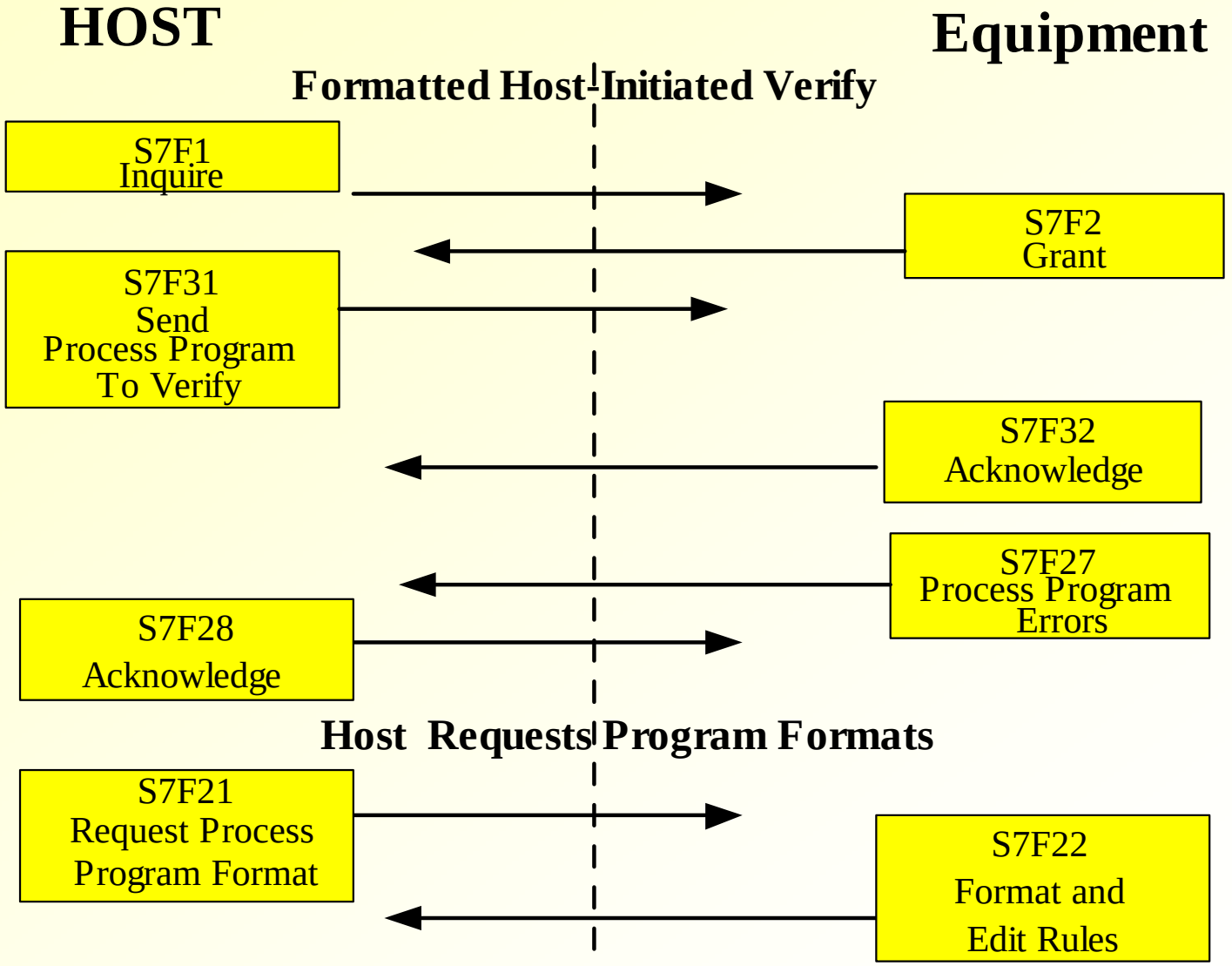
GEM Capabilities

Formatted Process Programs (continued)



GEM Capabilities

Formatted Process Programs (continued)



GEM Capabilities

Process Program Management

Host Delete

The Host can delete one or more Process Programs from the Equipment's library.

```
S7F17   W                          * H⟶E
      <L
            <A PPID>               * Process Program ID
            …
      >.
S7F18                              * H⟵E
      <B [1] 00> .                 * ACKC7
```

Host Directory

The Host can request a directory of the Process Programs which currently exist in the Equipment's library.

```
S7F19   W .                        * H⟶E

S7F20                              * H⟵E
      <L
            <A PPID>               * Process Program ID
            …
      > .
```

GEM Capabilities

Process Program Management (continued)

Variables for Process Programs

GEM requires the Equipment to provide the following Variables relating to Process Programs.

PPExecName – This Data Variable shows the PPID of the Process Program which is currently being executed by the Equipment. If the Equipment can execute more than one PPID simultaneously, this Variable may take the format of a List of PPID's.

PPChangeStatus – When the equipment Operator changes the Equipment's Process Program Library, this Data Variable shows the type of change made (add, change, delete).

PPChangeName – When the Equipment Operator changes the Equipment's Process Program Library, this Data Variable shows the PPID of the Process Program affected (added, changed, or deleted).

GEM Capabilities

Process Program Management (continued)

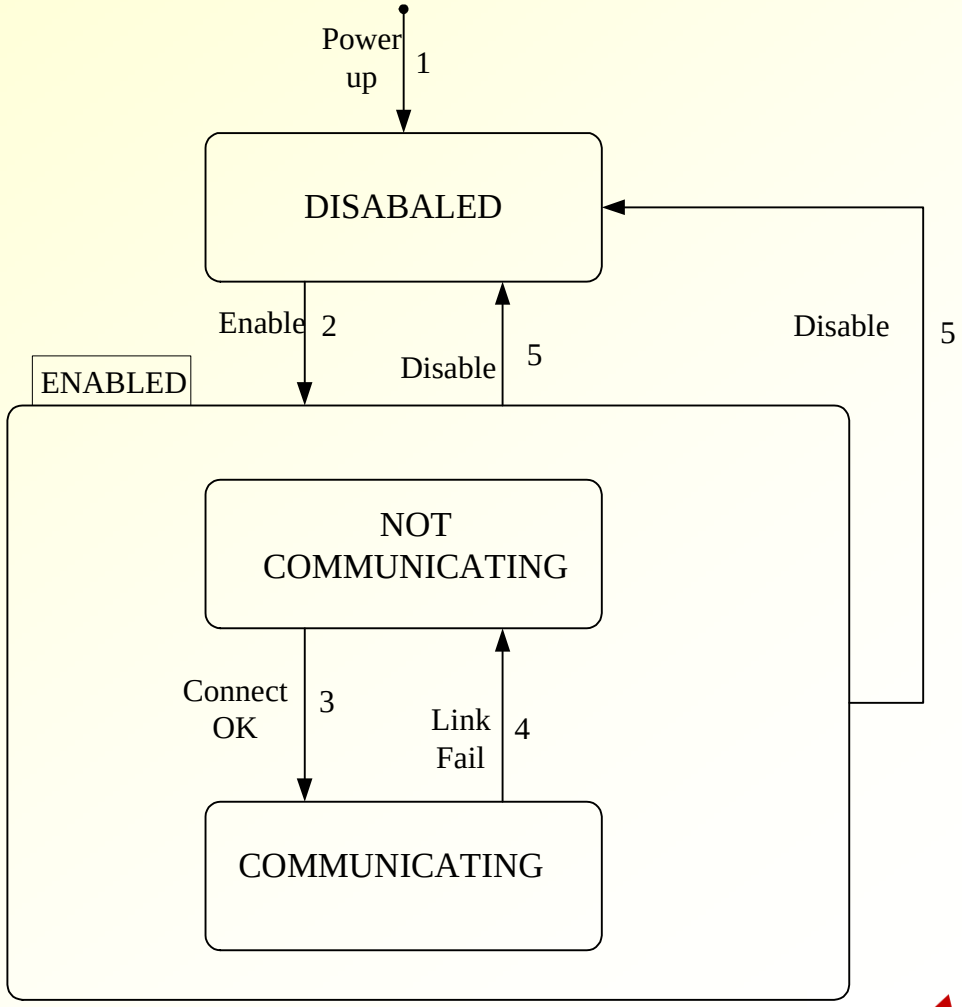
Events for Process Programs

GEM requires the Equipment to provide a Collection Event (CEID) which the Equipment signals whenever the Equipment Operator changes the Equipment's Process Program Library.The purpose is to inform the Host that the Equipment's Library has changed. At the event, the Data Variable **PPChangStatus** shows the type of change, and the Data Variable **PPChangeName** shows the Process Program ID affected.

GEM Capabilities

Establish Communication

This GEM Capability provides a standard orderly method by which Equipment will establish its SECS communication link with the Host.



GEM Capabilities

Establish Communication (continued)

Communication States:

- **DISABLED**

The Equipment sends no messages to the Host and refused to accept incoming messages from the Host. (Equivalent to disconnecting the RS-232 cable for SECS-I)

- **ENABLED**

The SECS link has been enabled.

ENABLE containing two sub-states :

NOT COMMUNICATING and **COMMUNICATING** .

GEM Capabilities

Establish Communication (continued)

❖ NOT COMMUNICATING

The equipment periodically tries to send S1F13 to the Host and receive the reply S1F14 from the Host. If it fails, the Equipment delays per the Establish Communication Delay time before trying again.

The Host may also send S1F13 to the Equipment and the Equipment must send reply S1F14.

❖ COMMUNICATING

The Equipment/Host link is up and operating.

GEM Capabilities

Establish Communication (continued)

Communications State Transition Table

#	Current State	Trigger	New State	Action/Comments
1		Power up	DISABLED	Power up the Equipment
2	DISABLED	Enable	ENABLED	Enable the link (RS-232 cable for SECS-I)
3	NOT COMMUNICATING	Connect OK	COMMUNICATING	Either an Equipment initiated or Host-initiated connect transaction succeeds. Send S1F13 and wait for S1F14.
4	COMMUNICATING	Link Fail	NOT COMMUNICATING	For SECS-I, the Equipment tries to send a block to the Host but encounters the Retry Limit error condition.
5	ENABLED	Disable	DISABLED	The Equipment operator takes action to disable the link.