Generic Equipment Model (GEM) Specification Manual: The GEM Specification as Viewed from the Host

SEMATECH
Technology Transfer 97093366A-XFR

Technology Transfer # **97093366A-XFR**
SEMATECH
December 31, 1997

Abstract: This document is designed to acquaint software engineers with the challenges and advantages of implementing the Generic Equipment Model (GEM) (SEMI standard E30) in semiconductor factories. Written from the standpoint of a factory host computer, it is targeted especially for Japan's semiconductor industry, which has not shown wide acceptance of GEM. It contains extensive GEM instructions, background, and scenarios. Appendix A includes the results of a survey of Japanese manufacturers and suppliers regarding their interest in, and use of, GEM.

Keywords: CIM, Generic Equipment Model, Japan, Manufacturing Systems, Specifications, Standards

Authors: Kensuke Uriga, Texas Instruments/Japan

Approvals: Margaret Pratt, Project Manager
Dan McGowan, Technical Information Transfer Team Leader

Table of Contents

List of Figures

List of Tables

Acknowledgements

SEMATECH wishes to express its gratitude to Kensuke Uriga of Texas Instruments/Japan, who authored the Japanese version of this document; and to Adam Rice, who translated it into English.

Preface

This document is a supplement to the Generic Equipment Model (SEMI Standard E30). Using general, common-sense language, it speaks to users seeking to implement GEM from a factory host computer. For proper understanding, the reader should be familiar with equipment automation (i.e, equipment networking); knowledge of E30 is desirable, but not essential. Although efforts have been made to ensure accuracy, this document may contain inadvertent contradictions with E30.

Foreword

The author of this document is Kensuke Uriga, a systems integrator in Semiconductor CIM Systems at Texas Instruments/Japan. Since 1990, Mr. Uriga has supported various equipment communications standardization efforts, including STEP/SECS and GEM.

Because Japan's semiconductor industry offers few instructional materials and little training for SECS/GEM, Mr. Uriga felt it might be useful to write a document for equipment suppliers that would provide basic knowledge and requirements for GEM implementations, since hardware engineers often lacked knowledge in production control, factory equipment control, and effective use of equipment data. Also, the author believed that if equipment suppliers had general knowledge about semiconductor manufacturing control, they would be able to develop more standardized, sophisticated online specifications for their commercial equipment.

SEMI/Japan previously published Version 1.0 of this document in August 1994 and Version 2.0 in December 1995, with the latter selling out of its 600 printed copies. (Mr. Uriga plans to write Version 3.0, which will contain major new requirements for standards, proposal and online fab requirements.) Version 2.0, presented here in English with the author's permission, can be used by GEM "beginners" to obtain basic knowledge and requirements about GEM implementations.

1 EXECUTIVE SUMMARY

This document is designed for software engineers implementing the Generic Equipment Model (GEM) (SEMI Standard E30) for semiconductor manufacturers or equipment suppliers, and is written from the perspective of a factory host computer. Targeted especially for the semiconductor industry in Japan, where GEM has been slow to gain acceptance, this report also is meant to provide the concepts behind GEM and how to implement it.

The goals of this document include the following:

- Explicate required features at the host end, specifically regarding its interaction with the client software
- Give the background behind each GEM capability, as well as reasons for their inclusion, cautions on applying them, and illustrative samples
- Examine general scenarios for GEM application and explain the major requirements for GEM automation
- Clarify GEM's features and show how GEM differs from messaging in SECS
- Examine functions not currently supported in GEM and show how GEM can be extended

There are many advantages associated with GEM compliant systems, including the following:

- GEM reduces equipment software development costs and improves functionality and reliability by standardizing the host/equipment interface.
- GEM does not define a subset of SECS scenarios. It offers a uniform, systematized specification providing flexibility for host requests and equipment functions.
- GEM is a technological foundation for future CIM in semiconductor fabs. It allows for future expansion of capabilities.

GEM is the first state toward open semiconductor-CIM systems. As these CIM systems become more standardized, CIM software product lines that offer more flexible interfaces and configurations should become available. This will allow users to purchase software off the shelf, rather than creating it themselves.

2 INTRODUCTION

Work on GEM began in the U.S. in 1988, and in 1993, SEMI Standard E30 was approved. The SEMI Japan communications committee worked closely with the SEMI USA communications committee from the start.

At the time work on this project began, there were any number of obstacles to implementation. For instance, compared to the existing SECS standard, GEM required more sophisticated data management. At the time, a lot of fabrication equipment ran on controllers meant for real-time control to handle all control functions. It would be difficult to implement all their functions strictly as GEM functions. Also, GEM required that an advanced set of functions be immediately

available on the production floor, so both in terms of functionality and installation, GEM appeared problematic.

But the advent of cheap, powerful, and reliable PCs and workstations made them attractive for use as the main controllers for fabrication equipment, which increasingly they are. From a hardware standpoint, modern computers have eliminated the constraints on GEM implementation. Since these computers ordinarily have a multitasking operating system (OS), running GEM software likewise is no longer an issue.

Periodic software updates from the U.S. expand GEM's base. With the active backing of SEMATECH, a good number of fabrication equipment makers have signed on to support GEM. Naturally, chip makers have also requested that equipment manufacturers support GEM.

However, there are very few Japanese equipment makers supporting GEM, and when they do, it is generally in equipment destined for American chip makers. Almost no cases exist where Japanese chip makers have requested that their equipment suppliers support GEM (see Appendix A, "Survey Results—SEMI Japan Survey on SECS Implementation").

Following are possible reasons behind the slow adoption of GEM in Japan:

- Implementing GEM would require the development or purchase of new software both for the host and the fabrication equipment, and it is difficult to see how the company will recoup its initial investment in GEM.
- GEM has broader functionality than its precursor, the Semiconductor Equipment Communication Standard (SECS). Implementing GEM would therefore require more sophisticated software on the host machine.
- GEM is positioned chiefly as a data-processing standard, and is perceived as having inadequate transport system interfaces and material transport automation functions.
- The current host-equipment interface complies with an existing specification, so implementing GEM offers few advantages except when bringing in new equipment. Furthermore, upgrading the existing hardware to GEM standards would be problematic, both technically and financially.
- The perception exists that unless 80% of all equipment is upgraded to GEM, there are no advantages to having GEM.
- Being a SEMATECH project, there is no central organization in Japan to back GEM.

In short, there are any number of reasons why GEM has been slow in being accepted in Japan. Moreover, GEM's features and capabilities have been poorly explained to some extent, so information about GEM needs to be made clearer in the future.

In the U.S., though, the core of the GEM host-client interface is already in place, and the general trend is towards proceeding from there. As more and more equipment manufacturers, chipmakers, and software vendors begin to support and improve GEM, more software will become available that improves the performance and sophistication of GEM at a lower price.

Computer hardware and software is essentially growing standardized around the world now. This has created intense price competition both in the hardware and software markets, and prices

could yet drop quite a bit more. We may see the same sort of thing happen to CIM systems for semiconductor fabs as well. Currently, most chipmakers are using custom software only usable in their own operations, but this has limited reusability, and software development costs are expected to tumble.

While the spread of GEM should result in cheaper client-side, GEM-compatible software, noncompatible software will probably stay at the same price, or even grow more expensive. This will give those equipment makers and chipmakers who are using GEM an increasing cost advantage over those who are not.

Furthermore, GEM growing more prevalent will result in more software vendors developing and distributing products that are compatible with GEM clients. Having a greater number of vendors selling GEM software will drive software prices down and help to lower the price of GEM implementation.

While the basic goal of GEM is to provide a standard communications interface for equipment, in the future this may be extended to components outside those in computer integrated manufacturing (CIM) systems for semiconductor fabrication, such as process control systems, technical information control systems, and other major components. This will also allow them to interface with the CIM system, ultimately enabling an open, distributed CIM system.

SEMATECH currently is working on a standards proposal for a semiconductor fab CIM system based on object technology. It remains to be seen whether or not these standards will form a unified whole with GEM. GEM is not simply standardize individual pieces of equipment, it is a major milestone on the road to creating a standardized, open CIM system for chip fabs.

3 PURPOSE OF THIS DOCUMENT

SEMI approved E30 in 1993. E30-1994 (E30 hereafter) is meant to be a reference model for implementing a host-client interface protocol for users of E4[1] and E5[2].

This document is meant for software engineers implementing GEM for semiconductor manufacturers or equipment suppliers, and should help give a better idea of the concepts behind GEM and how to implement it.

The goals of this document are as follows:

- Explicate required features at the host end, specifically regarding its interaction with the client software
- Give the background behind each GEM capability[3], as well as their reasons for inclusion, cautions on applying them, and illustrative samples
- Examine general scenarios[4] for GEM application and explain the major requirements for GEM automation
- Clarify GEM's features and show how GEM differs from messaging in SECS

[1] E4-91: semiconductor fabrication equipment standard 1: message transfer (SECS-I)

[2] E5-94: semiconductor fabrication equipment standard 1: message content (SECS-II)

[3] GEM capabilities indicate operations that are undertaken by fabrication equipment. For further details, see E30, "2. Definitions."

[4] Scenarios showing the procedures for executing GEM capabilities are in the SECS-II message listing

- Examine functions not currently supported in GEM and show how GEM can be extended

To avoid overlap, anything that is explained in E30 will be omitted here. For this reason, the reader should consult E30 for clarification or details.

The client software features described herein are modeled on GEM. While GEM's features can be extended, this document omits any description of capabilities that seem to clearly contradict GEM. Although this document may extend beyond the scope of GEM in places, it should generally adhere to the GEM standard. Note that explanations of GEM extensions and concepts not clearly defined in GEM are the author's opinions.

4 WHERE GEM FITS IN

This section shows how GEM relates to other communications protocols, and explains the purpose of each protocol.

4.1 Scope of SECS-I, SECS-II, HSMS and GEM Implementations

The following list shows the differences between features and objectives in the various communications protocols: SECS-I (SEMI E4); SECS-II (SEMI E5); HSMS (High Speed SECS Message Service (HSMS) (SEMI E37-95); and GEM.

SECS-1
This defines an transmission interface for passing messages between the fabrication equipment and host. It specifies serial point-to-point communications, and covers the required connectors, signal levels, data rates, and logical protocols for passing messages back and forth (see E4-91).

SECS-II
Defines in detail the format for messages passing between the host and clients. Uses the message transfer protocol defined in SECS-I, but defines the form for messages passing between the host and client equipment (see E5-94).

HSMS
This defines a transmission interface on the same level as SECS-I. This substitutes TCP/IP in place of RS232C ports at 9600 bps. This allows transmission speeds in excess of 10 Mbps over Ethernet.

GEM
Defines a method for machine operations over a communications link. Defines specific functions for client machines, and specifies SECS message scenarios that will cause these operations to execute. Although this does define specific operations for the client machines, it does not similarly define operations on the host machine (see E30).

4.2 Interoperation of GEM, SECS-I, SECS-II and HSMS

The postal metaphor: SECS-I and HSMS as a piece of mail. Figure 1 shows how SECS works, using a postal metaphor. Assuming the address is correct and the necessary postage has been affixed, then it should reach its intended recipient once it is put in a mailbox. The contents of the letter do not matter: as long as the procedure has been followed correctly, there should be no obstacles to the letter getting through.

The address corresponds approximately to SECS-I or HSMS. In reality, the mechanism for delivering the letter is equivalent to the physical network system and the transaction protocol, which are really what SECS-I and HSMS cover.

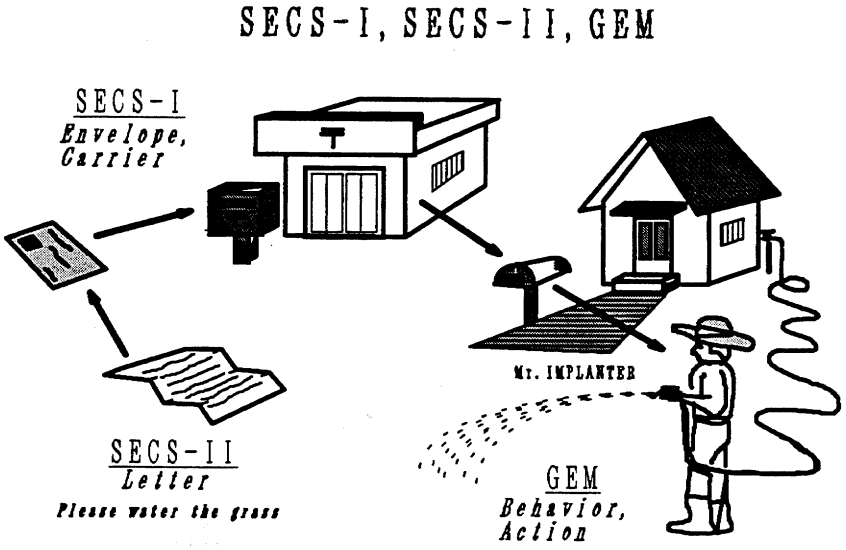

Figure 1 Postal System as a SECS Metaphor

SECS-II relates to the text, the message to be conveyed to the recipient. Conversely, SECS-II relates to the actual contents of the letter or postcard. SECS-II specifies the format of the text and the minimal unit of meaning. In the same way that unreadable letters in the text would prevent you from understanding the contents of a letter, SECS-II gives a format that must be followed in order for the information to pass back and forth correctly.

With regular mail, two different people might interpret the same text in different ways. For example, a recipient might be prompted to take some sort of action, depending on the contents of the letter, but the exact action taken might differ from between recipients. Likewise, SECS-II does not specify the exact action that a recipient (or in this case, a client machine) should take.

GEM indicates the reader's action or reaction. GEM is concerned with the traffic in specialized messages (SECS-II transactions) and the particular actions taken upon their receipt. In the GEM universe, anybody reading a given letter should always take exactly the same action in response. This assures that the sender's intent will be properly conveyed to different recipients.

To carry the postal analogy a bit further, a GEM message—which might be likened to a Christmas card, an invitation, or a request for a favor—also carries with it a partial explanation of the intended action for the recipient. Neither the order in which all these letters are exchanged nor the interrelationships among the actions they are meant to initiate are specified. These depend

on the people at both ends and the circumstances under which the letters were exchanged. In GEM terms, these depend on the plants and the message scenarios.

One purpose of these GEM guidelines is to clarify the way one should handle those things not specified in a GEM message, such as message precedence and the interrelationships among recipients (equipment).

HSMS is for high-speed packets. In 1995, SEMI issued the HSMS Standard as a substitute for high-speed TCP/IP communications for the RS-232C serial communications in SECS-I. If one thinks of SECS-I as regular mail, then HSMS would be an express service. Since HSMS can handle larger messages than SECS-I, it might be more appropriate to compare it to a courier service.

The ability to transport larger messages was not the only change TCP/IP brought; it has had a considerable effect on the operation of production equipment. These effects will be discussed later.

Message flow between equipment and host. Figure 2 shows the flow of message traffic between host and equipment, and the layered structure of SECS. Though the actual software is not limited to this sort of implementation, this example is simplified for the purpose of discussion. Following is an explanation of the order and layer in which messages are handled.

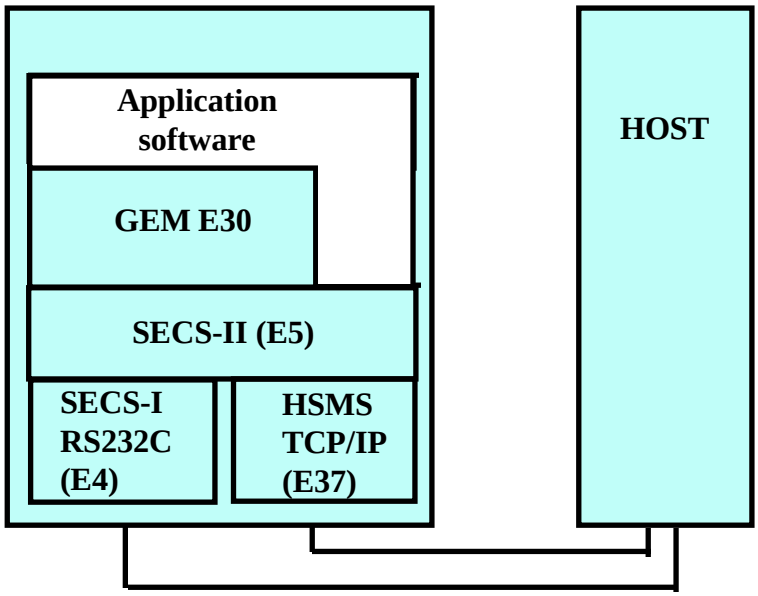

Figure 2 Flow of Message Traffic Between Host and Equipment

1. The host sends out the “establish communications request” (S1, F13) to the equipment.
2. This message is received at the SECS-I or HSMS layer and passed up to the SECS-II layer. If it is a multi-block messages, it may undergo block-deblock conversion here.
3. At the SECS-II layer, the equipment analyzes the format of the S1, F13 message, and if there are no problems with the message, it sends a message to establish communications up to the GEM layer. If there is a problem with the format, the SECS-II layer generates an appropriate error message and sends that down to the SECS-I/HSMS layer.

4. Having received the request to establish communications, the GEM layer executes a state transition in accordance with the E30 communications model, and if everything is OK, sends a message corresponding to “communications request acknowledge” (S1, F14) down to the SECS-II layer. The GEM layer may also notify the application layer that it has switched into a communication state.
5. The SECS-II layer generates the S1, F14 message and passes it down to the SECS-I or HSMS layer.
6. The SECS-I or HSMS layer transmits the message to the host.

Assembling GEM scenarios is the job of the equipment’s application software. GEM does not yet specify the way in which individual capabilities should be bundled together into “online operations” scenarios for every piece of equipment. This is a point to be investigated by GEM users. In the example of Figure 2, the application layer supports actions for all the various scenarios on that piece of equipment. Development of the application layer is the responsibility of the equipment supplier.

5 BRINGING GEM ONLINE

This chapter will explain the objectives of bringing GEM online and the results of doing so, and will discuss GEM features and conditions for GEM compliance.

5.1 Goals and effects

Compared to previous efforts, GEM specifies a more sophisticated standard for the host/equipment interface. Following are some of the intended effects of GEM:

1. Reduce costs of developing client software
2. Improve reliability of client software
3. Improve the feature set of client software
4. Reduce the time to develop and implement plant-wide online systems
5. Improve the feature set of CIM systems

GEM’s external environment allows reuse of client-side SECS software. It was important that GEM had a software development environment that gave equipment suppliers portable client software, for greater efficiency and a unified architecture. Implementing these internal and external environments helps achieve goal 1 of reduced development costs.

If the client software is portable, every piece of equipment can use the same kind of software. This vastly reduces the total number of updates that will be needed, and compared to the pre-GEM situation, dramatically improves reliability, thus achieving goal 2.

Since the client software is portable, developers need not reinvent the wheel each time they want to implement a given feature for a different machine; their time can be dedicated to improving a single client application. This contributes to realization of goals 1, 2, and 3. This also prompts equipment suppliers to place due emphasis on software development, which will further help their development programs by causing them to accurately forecast personnel and financial allocations for development.

GEM helps realize goal 4, of shortening the time to develop and implement an online system for fabrication equipment, because it vastly reduces the time to develop online specifications. Note that even with GEM, there remains a great variety of operations for each piece of equipment, so GEM may not provide simple systems configuration for every piece of equipment. The host end also requires a flexible software architecture.

In the initial phases of GEM implementation, the user still must develop new software, but at some point in the implementation process, the effects of standardization will be seen. Until that happens, however, ongoing investment will be needed in new client software, bringing standardized host software online, and training and educating employees.

GEM brings the fifth goal of improving the CIM feature set through the cumulative effects of the previous four. Raising the capabilities of CIM systems is the final goal of GEM.

Bringing GEM on line provides considerable savings by shortening the work of developing a client/host interface in automation projects. GEM provides nearly all functions needed to bring a plant online. Thus, in bringing fab equipment on line, you can provide the necessary level of customization either through the GEM detail field or by developing functions that are not supported in GEM. So compared with earlier, non-GEM-compliant equipment, the time to develop a specification is reduced dramatically. And there is no need to develop a general specification for the host at all; only special aspects of the specifications need to be created, which further reduces labor.

GEM improves the equipment/host interface. GEM offers a more thorough set of communications functions than plain SECS. It does not require that detailed specifications be delivered to equipment suppliers. Functions can be added to the base provided by GEM or deleted from it to provide just the needed ones.

The advantage of a global CIM system. If you are establishing a plant overseas, it makes sense to go with an international standard, as this will shorten development times, reduce expenses, and make it easier to set up and maintain equipment.

Failure to standardize results in duplication of effort in development for hosts and equipment. Any number of semiconductor manufacturers have already developed proprietary host-client interface specifications. In some cases, a single manufacturer may have different interfaces at different fabs, even though the same types of equipment are in use.

Equipment vendors must revise and deliver client software that conforms to the interface dictated by the semiconductor manufacturer. Taking the semiconductor industry as a whole, this could ultimately result in a total number of different versions of communications interfaces equal to the product of the number of equipment types multiplied by the number of semiconductor manufacturers (or perhaps even fabs).

This way of doing things looks at individual fabs as the unit within which standardization takes place, and within which the developmental costs of host and equipment software are to be contained. But failure to standardize across an entire company, or indeed, across an industry, creates considerable inefficiencies in equipment development, and the semiconductor manufacturers, being the purchasers of this equipment, bear the cost. Furthermore, the client-end interface must be able to accommodate revisions to the host software. This militates for a standard in communications interface software. Added software development costs for host software have similar implications.

The key to GEM's success is in having a greater number of semiconductor manufacturers adopt the GEM standard. The added costs created by the duplication of effort in software development will be negated once more semiconductor and equipment manufacturers start using GEM. These savings can be re-invested in developing products with greater added value.

5.2 GEM and Factory Integration

Factory integration and the client interface. The goal of factory integration is to get all the plant's various subsystems working as a single system for the purposes of automation. GEM can be considered as filling the equipment/host interface for integrating the host in an integration program.

Take a look at Figure 3. In this example, there are three plant subsystems: host, equipment, and automated material-transport system. In a factory that has not been fully integrated, the host, equipment, and automated material-transport system (which can be considered an interprocess transport system) operate separately, and production only takes place through an operator's mediation. Figure 3 shows arrows pointing from each of the subsystems towards the operator, to indicate that operational requests go to the operator. The operator's interface with each of the subsystems is through some sort of a terminal, such as a CRT; the operator uses this to move production along. Having production run by an operator puts a lot of flexibility into the system, and adequately covers shortfalls in the functionality of the subsystems.

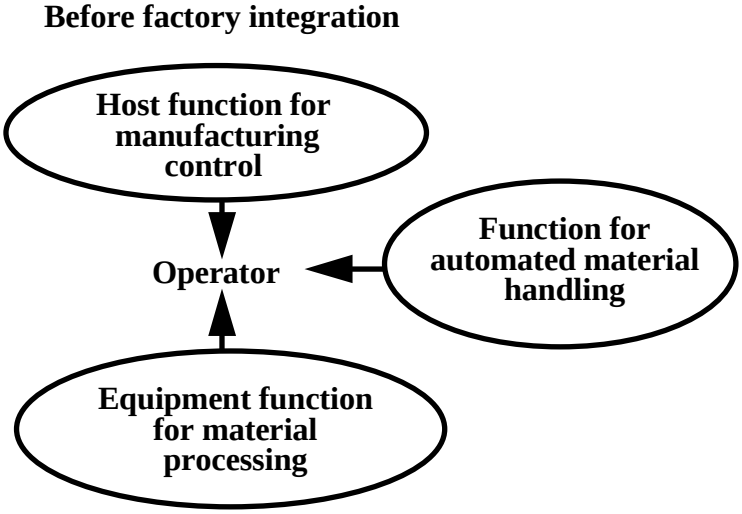

Figure 3 Operator-Mediated Integration

Figure 4 shows a completely integrated factory, in which host, production equipment, and the automated transport system (which can be thought of as an intraprocess transport system here) are directly connected to one another. Rather than sending operational requests to an operator, each of the subsystems generates an "interface function request," which is the basis for coordinating activities. Compared with an operator-mediated arrangement, this directly connected arrangement has almost no flexibility—if there is even a minor crossup, the entire system might come to a halt.

The interface function request can technically accommodate just about any subsystem. If the others can be made to accept the greatest number of functions possible, then one's own load will

generally be lighter. Although this causes disputation among the staff, ultimately it leads to more rational decision-making.

For instance, the information required for online operation of a piece of equipment that is acquired through the user interface (e.g., lot information, recipe information, and other pieces of information requiring operator confirmation), which would normally appear on the machine's screen, will often be handled as a part of the host's production management functions. Whether a machine's material process has ended, whether the input port has been cleared, and the like are often handled as material request messages that the equipment sends to the host. Similarly, the man-machine interface, malfunction processing, and alarm-related interface function requests are all specific to the operation of a plant, and thus often differ from plant to plant.

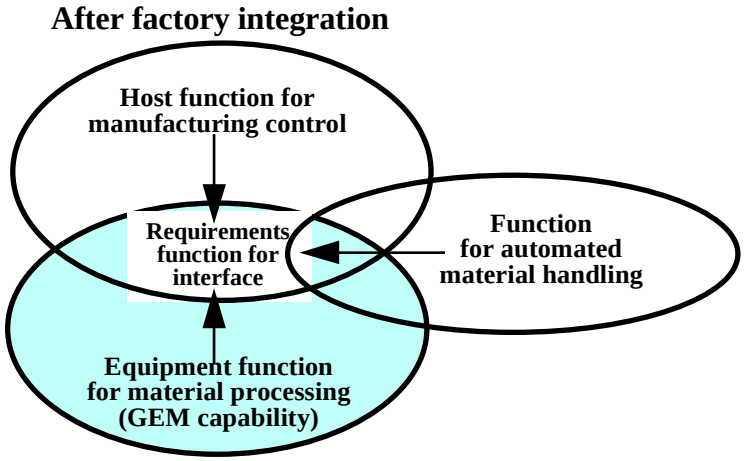

Figure 4 Total Factory Integration

An equipment interface specification consists of equipment-based process functions, host-based operation functions, and material-transport functions. As previously stated, a system for automating the production of semiconductors will comprise host, equipment, and transport subsystems. Each of these subsystems has functions particular to their role, and will have certain interface functions to enable automation. The exact interface functions will depend on which subsystem is connecting to which other subsystem, so the request specifications will vary accordingly. For example, when the host's production management facility sends a request to a specific piece of production equipment, the interface specification the host uses will naturally differ depending on the specific process functions supported by the equipment.

For function requests from the host to be compatible with the production equipment interface, the software that controls the production equipment's interface with the host must be customized. Conversely, if one focuses on functions particular to a piece of wafer-processing equipment, both the host and transport system must have their equipment interfaces customized in order to support all the production-management features in the host and transport automation functions in the transport system.

Existing host-equipment communications interface specifications are customized for individual semiconductor manufacturers. In general, when integrating equipment, the semiconductor manufacturer tries to shape it to fit either its production-management system's procedures or its computer system's functions. The equipment vendor will be expected to revise

its host-interface specification accordingly. These revisions essentially do not address functions relating to the actual processing of the wafer; rather, they generally cover how the host controls and interfaces with the equipment. Since the equipment manufacturer has little experience and knowledge of these production management procedures, they follow the semiconductor manufacturer's requested specifications for revised software. Naturally, the revisions are unique, and apart from host software, have almost no re-usability.

In extreme cases, the host computer has some sort of architectural or functional requirement that forces the equipment manufacturer to support functions unrelated to their product. The point of this is to make development of the equipment interface specification simpler at the host end, and put more of a functional load on the client equipment.

GEM is not concerned with functions that do not relate to equipment processes, such as production management. It focuses solely on generalized automation and processing functions in equipment.

Doesn't GEM have exact specifications for transport systems, operator interface, and production management? If GEM is already considered difficult enough, considering the complexity of its functions, then perhaps that is an excuse for saying so little about its production management, operator interface, and automated transport capabilities.

Host requests relating to production management and automated transport system requests will differ from fab to fab. Client equipment probably should deal with these requests to some extent, but the possible range is so broad that it currently would be difficult to develop a standard request protocol. As such, requests of this type are considered very little in GEM. Naturally, if it would be possible to support a standard equipment request protocol shared by all equipment, it would be a boon to production management, but it might not affect the equipment supported.

If, in the future, as ways are studied to standardize host and automated-transport interactions along a GEM-like model, it probably will be necessary to define a more exact request interface for dealing with equipment.

5.3 GEM Compliance

GEM compliance depends on satisfying the following conditions under E30, to support the capabilities specified under GEM:

1. GEM's basic requirements have been satisfied.
2. All relevant definitions, explanations, and necessary conditions as defined under E30 have been implemented.
3. Equipment capabilities have GEM-defined operations, and do not apply capabilities contrary to those actions.

Conversely, GEM is not concerned with capabilities implemented in GEM *when they are being used outside of a GEM context*, e.g., when SECS-II messages are used outside GEM.

GEM contains *basic requirements* and *additional capabilities*. If the equipment implements the capabilities in E30, then that equipment is considered to be GEM-compliant. To provide a yardstick for levels of GEM compliance, the terms *basic prerequisite* and *additional capability* are used here.

All basic requirements must be satisfied for GEM compliance. In order to ease GEM installation, the authors have narrowed down GEM's basic requirements to a minimal set of capabilities. These are capabilities that can be implemented on any piece of production equipment. Nevertheless, if even one basic prerequisite is not present, the machine is not GEM compliant.

GEM's basic requirements *are not sufficient* for achieving automation. Simply fulfilling GEM's basic requirements will not be enough to achieve an automated system. The basic requirements do not cover remote control or process program management. There will be a number of situations where the required level of automation cannot be achieved using the basic requirements.

Additional capabilities may be chosen à la carte. Given the different types of equipment and approaches to automation found in different plants, the capabilities that GEM will be expected to support will also differ. Equipment manufacturers may select from among GEM's additional capability set as fits their strategy; semiconductor manufacturers may also request them as needed.

Failure to implement the complete E30 specification as-is will result in GEM non-compliance. Supporting the alarm report under the alarm capabilities set, but failing to support the alarm enable/disable message could be said to be providing alarm capability support but would not be GEM compliant.

The relationship between GEM capabilities and GEM compliance. See Figure 5 to understand the relationship between GEM capabilities and GEM compliance.

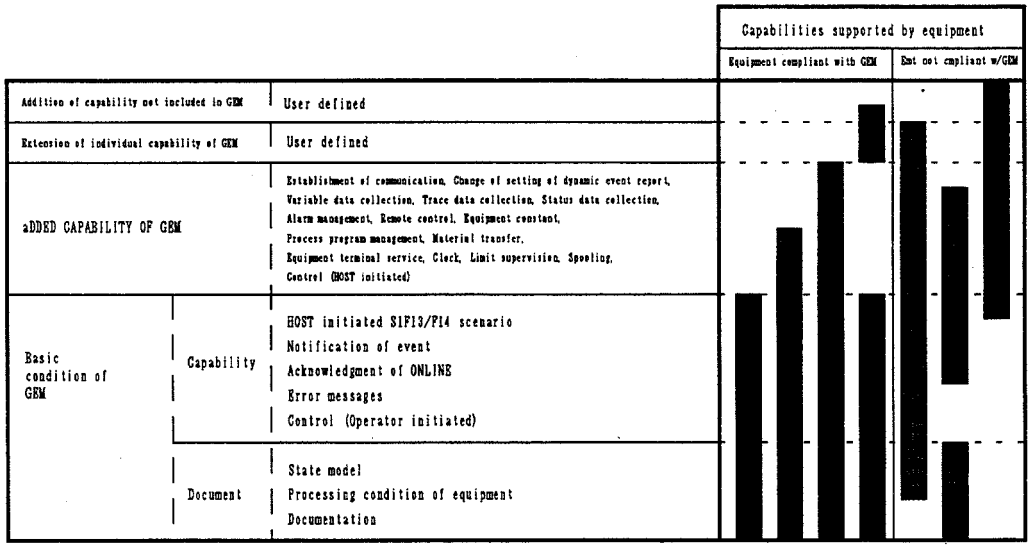

Figure 5 GEM Capabilities and GEM Compliance

Complete GEM compliance does not restrict you from supporting other necessary functions. GEM was designed to be generalized, supporting capabilities common across different types of production equipment, but each type of equipment will have machine-specific capabilities not specified under GEM. As previously mentioned, these capabilities can be treated as either *capability additions* of capabilities not present in GEM or as *capability extensions* to

capabilities that are included in GEM. It also may be necessary to add capabilities outside GEM at the host end; this can be handled in a similar manner.

What is the difference between a *capability extension* and a *capability addition*? A *capability addition* is defined here as any new capability not present in GEM. This might be the capability of the processing equipment to recognize a material ID code, which would involve a new type of transaction between the equipment and host.

GEM extensions are ways of taking advantage of existing GEM capabilities in new ways. Consider this example: when equipment is switched online, it sends an F1, S1 message to the host. Ordinarily, if there is not a response from the host within a set period of time, the equipment goes into offline. However, one could extend the equipment's normal behavior by having it retransmit the S1, F1 message periodically until it receives the S1, F2 response. This would be a capability extension.

GEM was designed from the ground up to allow for extensions to GEM software. The list below shows a taxonomy of GEM capabilities. These capabilities are all independent, and are concerned with matters such as establishing communications or control.

1. A root part of the capabilities that cannot be altered or extended (except for revisions to E30).
2. User-defined parameter data, which can be used to change processing conditions.
3. A part for the functionality of existing capabilities to be extended through extensions.
4. A part for new capabilities, supporting functions not already a part of GEM, through additions. E30 itself may also be revised or expanded.

E30 only covers parts 1 and 2. Parts 3 and 4 were included to provide a mechanism for adding new features during software development. Figure 6 shows the relationship between these parts.

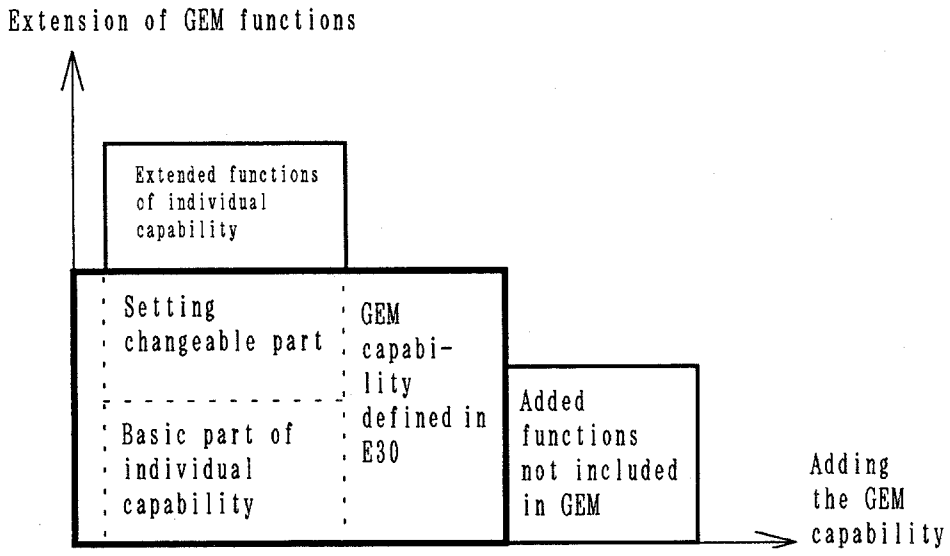

Figure 6 Expanding GEM Software

5.4 Installing GEM (Notes)

There are aspects of both hardware and software configuration to take into account when installing GEM on your equipment. This section explains some of those considerations.

Equipment systems configuration. The functions of the processing equipment, as viewed from the host, come in three broad types: Host Interface, Equipment Control, and User Interface. The relationship between these three groups of functions is shown in Figure 7.

When establishing communications in GEM, there is the host interface, the user interface (which covers the setting of system constants and terminal support) and equipment control (which covers all other GEM capabilities). The role of process control is actually filled by the equipment control functions. Equipment control can also be taken to cover service functions for interface with other processing equipment, such as data logging.

The equipment itself may comprise hardware and software with greater complexity, but that will not be handled here, as it involves the installation of device-specific controllers.

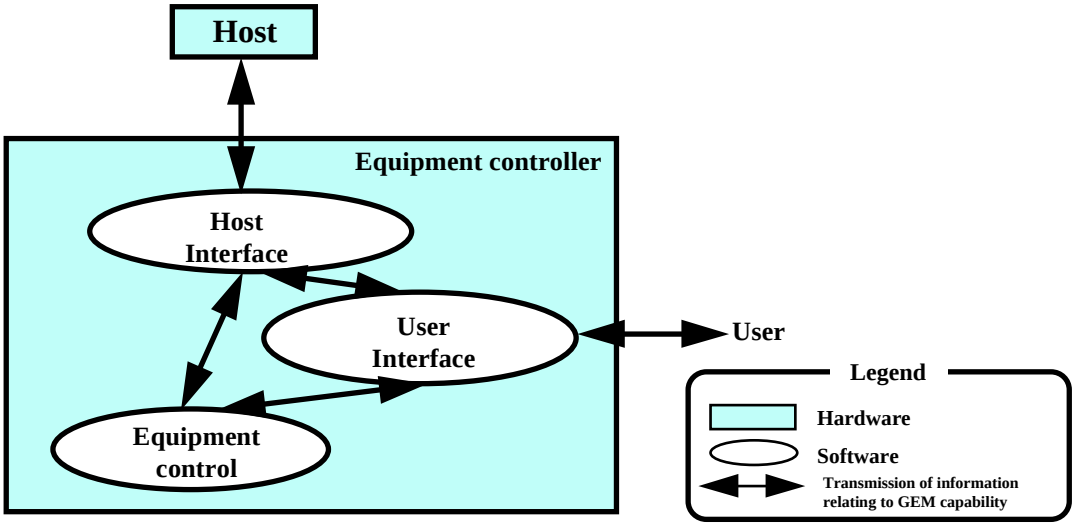

Figure 7 Structure of Equipment Controller Software

GEM installation is compatible with a multitasking OS. GEM does not specify any operation-precedence or interrelationship between its individual capabilities; all its features can be used independently of one another. Most of GEM's capabilities can be executed simultaneously and in parallel. As such, GEM is compatible with multitasking and distributed computing environments. This is an important advantage in helping you make the most of your investment in expensive processing equipment.

For instance, this allows you to edit a process program while processing is underway, and have the next recipe downloaded in advance. In order to take advantage of the independent operation of the host interface, user interface, and equipment control defined in GEM, a multitasking or distributed environment, as depicted in Figure 8, is required.

A multitasking environment will also be beneficial in order to support such maintenance and service functions as may be needed, but which are not defined in GEM. It would be difficult to implement GEM in an environment where there is one CPU for every task.

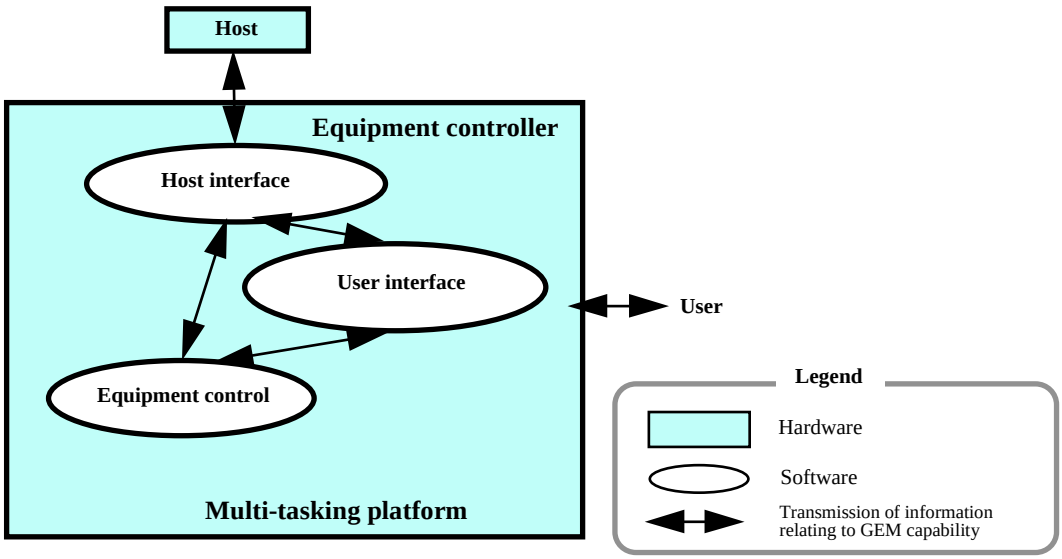

Figure 8 GEM and Multitasking

Operations that can be conducted on the control panel also can be conducted from the host. GEM requires that any operation that can be conducted on the control panel also be available for control from the host.

Figure 9 shows a schematic of how this would actually work. The heavy arrows show the equipment control interacting with the host interface and user interface, both of which use the same format for control data. This design requires that both the user interface and the host interface have access to the exact same capabilities.

Henceforth, this document will not refer to the control panel at all, only to the user interface.

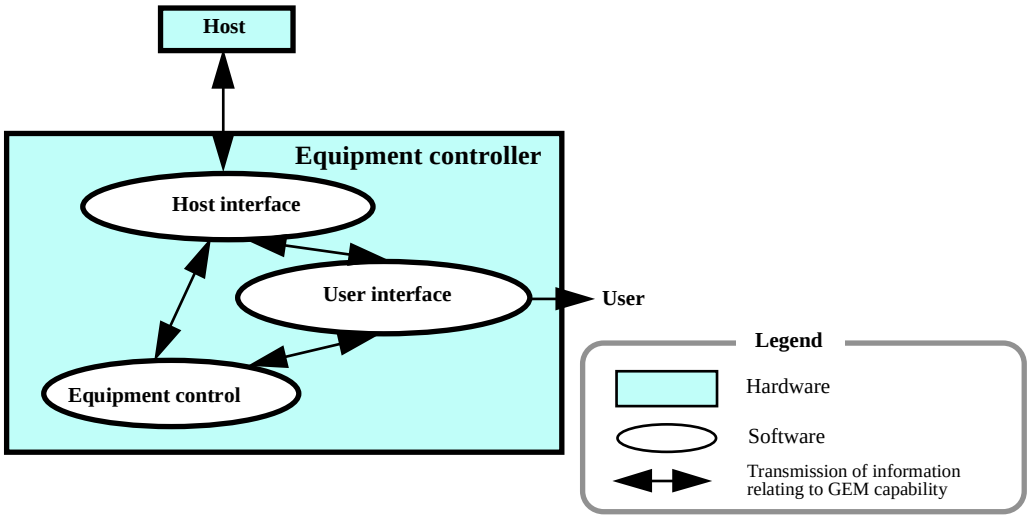

Figure 9 Host Interface and User Interface

The user interface is mediated through software. Given that operations originating from the user interface and from host interface are treated in the same manner, the control panel (directly connected to the equipment control, used for input/output), consisting of physical switches, is

dispensed with in favor of a software-based system. This makes it possible to create user-interface software that runs on a personal computer or workstation. Doing so would allow for processing equipment with a simple control panel consisting only of a power switch, emergency shutdown button, and appropriate lamps and dials to confirm operational status. Another advantage is the ease of developing and maintaining the software-based interface.

Switching between host interface and user interface. Figure 10 shows what happens in a switch between the host and user interfaces. The processing state model and control state model assume that this takes place within the equipment control module. Depending on the circumstances, if the designated linkage (i.e., either equipment control<->host interface or equipment control<->user interface) is switched—such as when there is a communications error during online operations—then it will be easy to order a switch from the user interface.

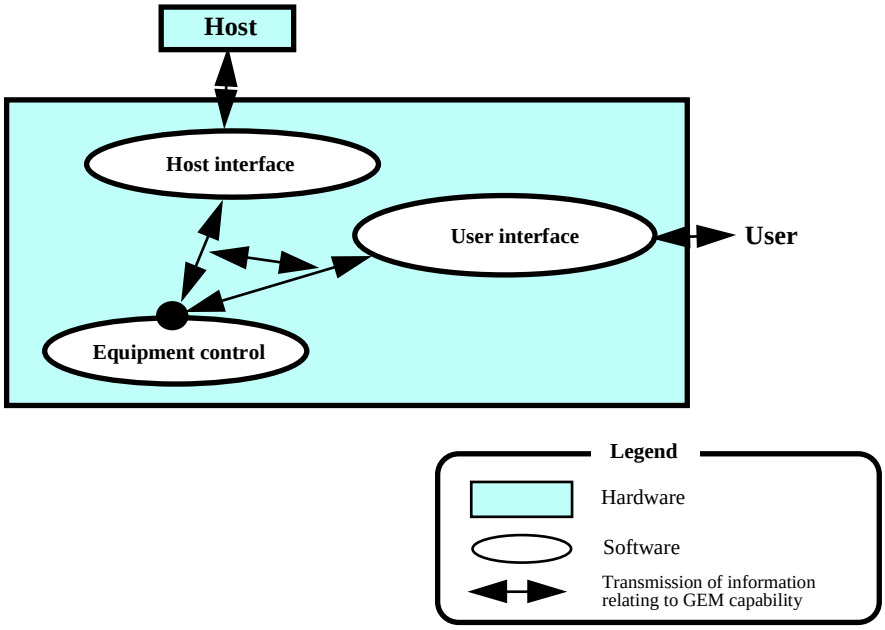

Figure 10 Equipment Control: the State Model

Distributed, multiple CPU environments. Future processing equipment configurations are likely to be multiple-CPU distributed systems, using PCs or workstations to run equipment management, treating data management and user interface as important, and using embedded controllers to run process-control functions, which need to operate in real time. Especially considering the near-term transition from SECS-I to HSMS, it will be easy to link regular PCs and workstations over a TCP/IP network to use as main controllers (see Figure 11).

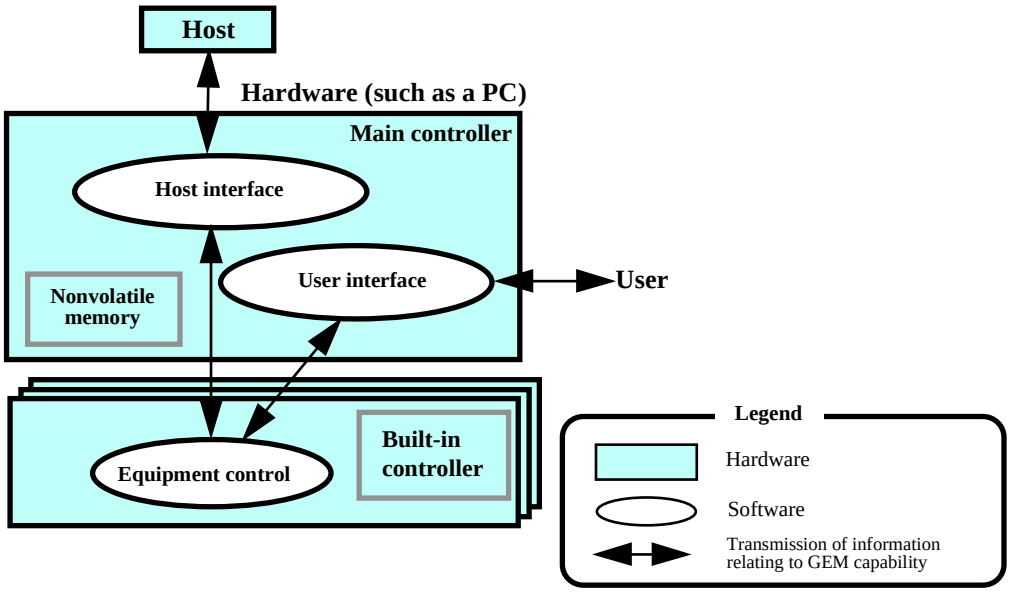

Figure 11 Equipment Control in a Distributed, Multiprocessor Environment

Problems with current software development method; Guidelines for general equipment software design. Despite the fact that GEM promotes a standard equipment interface, there still could be unanticipated needs for new functions at the host end, or there may be a need for new functions in equipment processes. When this happens, they will simply need to be added in. As such, the equipment software should be structured so as to make additions and modifications easier. The equipment software in use so far generally does not offer an easy way to deal with needed revisions or modifications. This goes for GEM software, too—unless it offers some flexibility, then it will be difficult to implement previously unforeseen needs for the host.

The timing of SECS message transmissions and the data in those messages might be within the boundaries defined by GEM, but there could still be differences between types of machines or individual users. Further, even supporting all the features currently in GEM within a single piece of equipment software may not satisfy the needs of all users. Therefore, your software architecture should be sufficiently flexible to meet the differing demands of individual users.

5.5 Implementing GEM

Will GEM be harder to deploy in a plant that is already heavily automated and standardized? Apart from the aspects of GEM that deal with automating material transport, GEM can support more functions than other communications interfaces commonly in use. It is possible that if you have extensively automated and standardized your plant around proprietary technology that it will be harder to make the switch to GEM. Each plant that converts to GEM will require different efforts to revise software and operating procedures for compatibility. Conversely, plants that have done little to standardize and automate their operations may be able to bring GEM online more easily, as they have more flexibility.

The semiconductor industry has had stunning growth throughout Asia. Most Asian countries have less resistance to SEMI standards than Japan, and are implementing them more assiduously

than Japan. Indeed, the countries that have the least experience with automation would seem to have the most to gain by adopting SEMI standards for communication and automation.

Equipment software price-setting strategy. GEM's spread is helped by pricing. This list shows how the price-setting strategy for GEM software aids GEM's popularization.

1. The initial development costs for installing GEM have been borne by equipment manufacturers, so individual semiconductor manufacturers do not need to absorb them.
2. The price of GEM-compliant equipment software is low.
3. The price of noncompliant software is higher (perhaps several times greater) than that of GEM-compliant software.
4. The delivery times for noncompliant software are long. Since noncompliant software is based on a unique specification, lead times can lengthen unpredictably.
5. GEM's streamlined specification is amenable to expansion, which has beneficial cost implications.

Naturally, this strategy is not intended to prevent software from improving. The goal is to reduce software revisions that cater to the needs of only one customer.

Also, equipment manufacturers will still have good reason to keep some proprietary communications standards. Semiconductor manufacturers do not begin development of a new system from a blank slate—it would be best to adopt a standard that fits well with the company's own specifications, which was the basis for this strategy.

6 PRODUCTION MANAGEMENT SYSTEMS

Understanding how GEM can be used as a part of a host-based production management system is key to understanding the GEM specification in general. This section will give application examples of bringing equipment online, and how GEM relates to the process. Note that by "host-based production management system," does not necessarily mean standard, popular systems. The model explained herein differs from the systems in current use in ways that probably cannot be reconciled.

6.1 An Application Example of Bringing Equipment Online

The GEM scenario has been adapted for the purposes of implementing SECS/GEM on the host. Below are some host-end management features that affect the GEM scenario.

The management features explained herein have intended relationships with SECS/GEM capabilities. That is, this report will discuss only the functions achieved by bringing your equipment online, and there will be some differences in the general management features required in semiconductor fabrication. Also, the functions described later can probably be implemented even without SECS/GEM. Nevertheless, using SECS/GEM along with host-based automation software should offer a more efficient environment. This document will be examining GEM-based automation with attention to its production-management functions.

Please note that although the hardware and software set-up you use for automating your facilities is an important topic, it will not be discussed in this document, since the range of possible set-ups is great—and more importantly, it does not directly relate to GEM.

Tracking and controlling work-in-process; recipe downloading. Work-in-process (WIP) materials could be considered any semi-finished product that is in process. Managing the progress of materials in each process (by wafer, lot or batch) is process management. The major objectives of process management are to ensure that each material unit moves through the processes correctly and that the product ships in the time required. Each of these processes may correspond to a single piece of production equipment, or when inspection equipment is involved, may involve several pieces of production equipment.

The host must execute the following tasks in a process management system: it must *track progress*, meaning that it has timely and correct information on every unit of materials and which process it is in; and it must *control production*, meaning that it orders each unit of materials to undergo specific processes with specific conditions.

Figure 12 is a diagram of a process management system. The host keeps track of the processing information for each product type in a master data file. The materials move through the system in accordance with this processing information. Materials can either by in-process, or they can be "load," "beginning processing," "processing complete," "unload," or a number of other states, all of which are reported to the host. One variable in all these reports will be the material ID, which the host uses to keep track of which material unit is at which stage of processing. The host will also download processing conditions and monitoring conditions to the equipment appropriate to the material entering processing at that equipment.

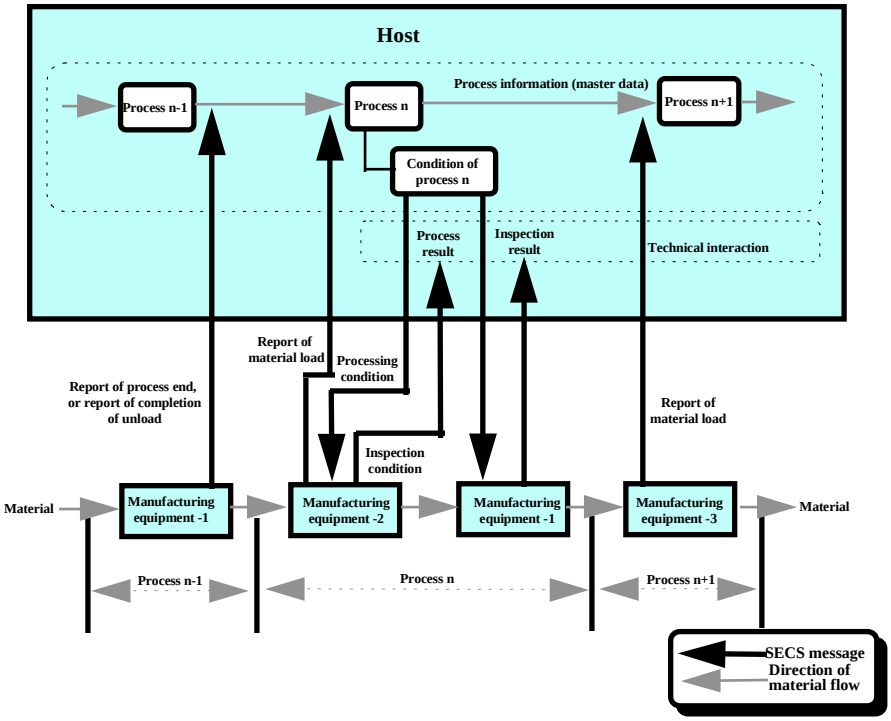

Figure 12 Process Management

There are two types of "process complete" reports: those from the production equipment and from the inspection equipment. The production equipment's report includes a field indicating "normal completion" or "completion with error," the number of wafers completed normally, and the processing time. The inspection equipment's report includes information on the wafer's

electrical characteristics and outward appearance. The host uses these two types of reports to determine whether the material should progress to the next process or whether it should be sent back for rework. Depending on the process, information about the equipment itself may be exchanged, to aid in managing reticle information. Processing results and inspection results are handled separately, and logged as technical information.

Bringing your equipment online automates the flow of data between the production equipment and the host's production management system. This can be used to prevent operator errors in recipe downloading and reticle management. It also helps keep you accurately apprised of process status, and simplifies process data collection. The GEM capabilities that pertain to this are event reporting and process program management.

Process control/Process monitoring. Process control and process monitoring involve managing the settings for processing parameters in each process and at each piece of equipment, classified by product. Further, it confirms whether the materials were properly processed according to those parameters, and decides whether the processing worked, allowing revisions to the processing conditions and feedback. The host is responsible for downloading all processing parameters, which will differ for each product and material, to the production equipment ("recipe" corresponds to process program in GEM/SECS-II). The host also collects processing results and inspection results for all materials from the production and measurement equipment.

Figure 13 is a processing flowchart for process control. As previously mentioned, the host keeps track of a master data file for all the processes involved in making every product. The master data file will include target values with "management limits" (these target values are used both as goals to be achieved in processing and in evaluating inspection results). The master data file will also include reference parameters (recipe name), and various work orders for the operator. The host stores physical parameters (key recipe parameters) and process models for each procedure and piece of equipment. Since different products may use the same process technology, sometimes the same processing parameters can be re-used. The host consults the master data file to select or generate the appropriate parameters, and download these parameters to the production equipment in a process program.

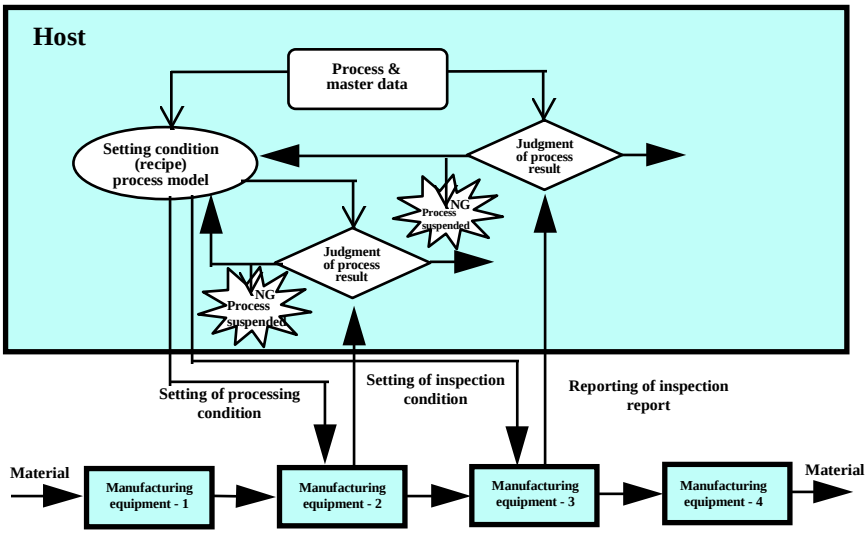

Figure 13 Process Control and Process Supervision

It is important to collect process results data for each wafer that reflects the equipment and the processes involved in its production. Wafer process results data comes in two varieties: process results are reported by the production equipment immediately after a process has been completed, and trace data is reported at regular intervals. The host compares these reports to its process model, and determines whether or not the reported data is within the management limits for the process. When the process results and the inspection results (collected from the inspection equipment) both fall outside the management limits, material processing is suspended, and corrective actions are taken (the recipe is revised, the equipment receives maintenance, etc.). Using the limit-monitoring capability in place of trace data may make it possible to determine whether management limits are being exceeded at the equipment end.

There are a number of advantages to process control. Automating typical review and revision procedures helps to improve overall process performance. Some of the GEM capabilities involved at this level are event reporting, trace-data collection, and process-program management.

Process analysis. The goals of process analysis are to 1) Ascertain production volume (capacity) and throughput; 2) Predict which processes will cause bottlenecks; 3) Predict cycle times. This information allows you to plan work-in-process insertions and set priorities for processes.

Semiconductor production involves hundreds of consecutive processes. The wafer must pass through similar processes, each with minor variations in production parameters, in a specific order. Several identical pieces of equipment can be grouped together, job-shop style, so that processes are shared out among them. The production flow may have multiple configurations, depending on the product.

Ordinarily, the host will store management information on work-in-process in a database. This work-in-process database would keep track of which materials were in which processes, and their status. Figure 14 shows how this fits into a process analysis system. In reality, of course, the varieties of equipment and processes would be more complex.

Bottlenecks occur at equipment groups with inadequate throughput. The reverse can also happen, where one equipment group has more capacity than is needed, which results in lower equipment utilization rates.

Having equipment online provides the means to track, in real time, the time it takes materials to pass through each process, using the *material load*, *begin processing*, *processing complete*, and *material unload* event reports. If these reports are used to automatically update the WIP database, existing bottlenecks can be spotted quickly, and material status generally can be tracked more readily.

For example, if there are 50 new lots per day, and there are 400 processes, then 20,000 processes per day are needed. If only the start and end of each process are tracked, then 40,000 transactions per day are recorded. Having equipment online allows this vast quantity of information to be collected accurately and in real-time, without operator mediation.

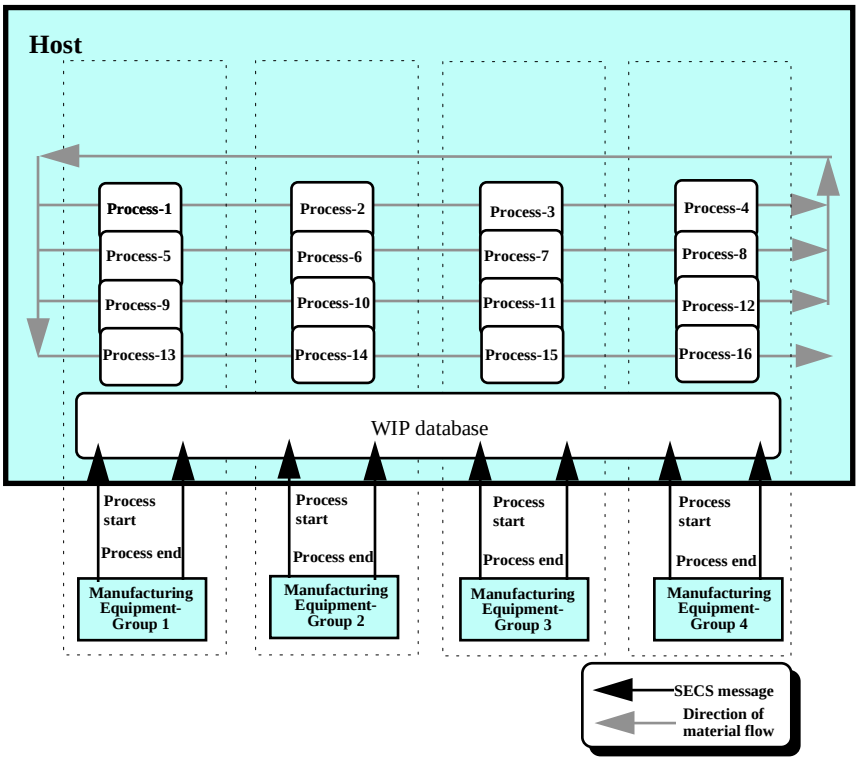

Figure 14 Process Analysis

As previously discussed, semiconductor fabrication involves passing materials through the same equipment group any number of times. This makes it extremely difficult to predict how WIP status might have changed at a given point in the future. With a WIP database that is being updated live with information from the production equipment, it becomes possible to use process simulations and online scheduling make predictions about which processes will be bottlenecks, and based on these predictions, issue dispatches. The GEM capability at the core of all this is even reporting.

Process diagnostics. Process diagnostics uses inspection data to determine whether each procedure is processing wafers at the needed level of accuracy, and whether there are problems that are pushing down yields. The object of process diagnostics is the shape of wafer patternings, chemical analysis data, etc.

Process diagnostics is often based on process capability, which uses numerically expressed data. Process capability is an overall measure of a the accuracy of a process—how close that process came to meeting its management limits. Process capabilities usually need to be quite high in order to achieve good yields, so process capability can be thought of as a yield indicator. Process capability is discovered by statistic processing of a large quantity of inspection result data on each process. The inspection result data is uploaded by the inspection equipment to the host over SECS lines, to be automatically integrated into a statistics management system, which makes it possible to track process capability in real time. Refer to Figure 15.

Modern measuring equipment can capture visual data, detect particles at certain positions on the wafer surface, and also detect chemicals. Up until fairly recently, floppy disks could be used for moving and managing data, but this is another area in which SECS-I or HSMS are beneficial, by

automating and unifying data handling. As a wafer moves through all the processes involved in its production, a huge amount of inspection data is generated along the way. For a system to be able to handle this requires that both the production equipment and inspection equipment are online, to facilitate the automated handling of measurement data. Event reporting is the key GEM capability in process diagnostics.

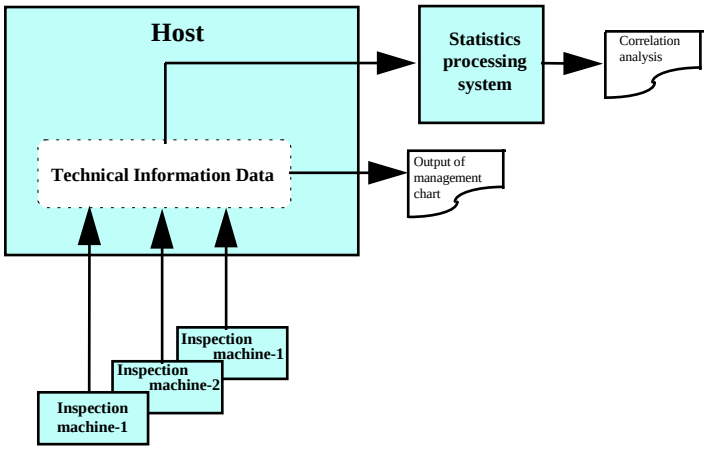

Figure 15 Process Diagnostics

Equipment diagnostics & maintenance. Even a minor change in environmental conditions can have serious effects on process capability in the manufacture of semiconductors, and of all the environmental factors, equipment condition is the one that has the greatest effect. Thus, it is very important to be able to monitor equipment condition in real time, collect trouble data, run tests for regular maintenance, and keep track of the number of wafers processed, to facilitate component swapouts and cleaning.

Continuous monitoring of the frequency with which a piece of processing equipment reports major alarms can help diagnose and predict equipment problems. SECS lines allow for the automatic collection, logging, and analysis in this kind of situation, where there is a lot of information (such as alarms) being generated in a short period of time. SECS lines also permit automated logging and range-checking for equipment operating parameters that need to be monitored in real time. Continuously tracking a large number of operational variables gives a deeper sense of equipment status and enables accurate equipment diagnostics. Important equipment parameters need storage and statistical diagnostics, using SPC charts or other techniques. Furthermore, automating the collection of equipment data and of decision algorithms can dramatically reduce the overall work involved in equipment diagnostics. Refer to Figure 16.

An online system for processing equipment, necessary for continuous, statistical monitoring of multiple status variables, can be adapted to automate diagnostics to beneficial effect. Furthermore, the equipment data that is now output via a printer interface can be sent over SECS lines to the host to be managed as part of a unified system. The GEM capabilities that are relevant to this task are event reporting, variable data collection, and alarm management.

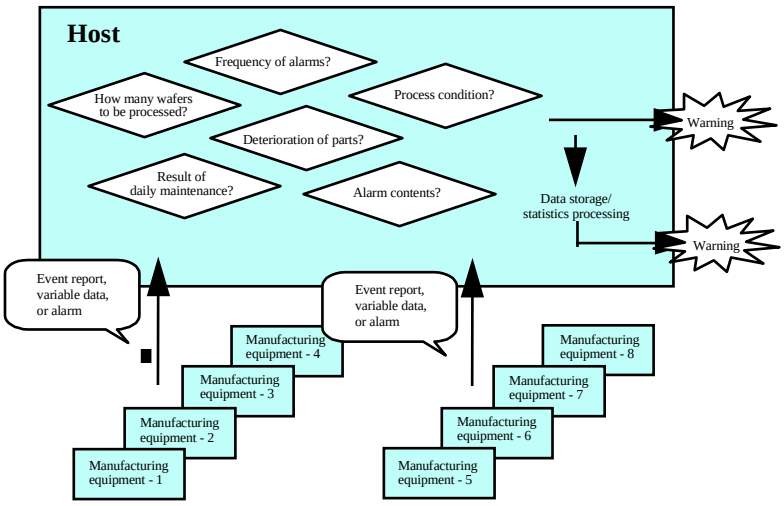

Figure 16 Equipment Diagnostics

Utilization monitoring. Utilization monitoring means knowing whether a piece of equipment is operational or not, with the intention of increasing equipment availability. Operating time management is an issue that E30 does not address, so few of GEM's capabilities are directly applicable to this. Presently, most of the data relating to a machine's operational state is collected by the host-based management system, apart from the equipment.

Figure 17 is a diagram of an operating time management system, based on the SEMI E10 specification for equipment operating states. The host compiles statistical data on equipment state transitions, from either event reports from equipment that uses this operating state model, or direct operator input.

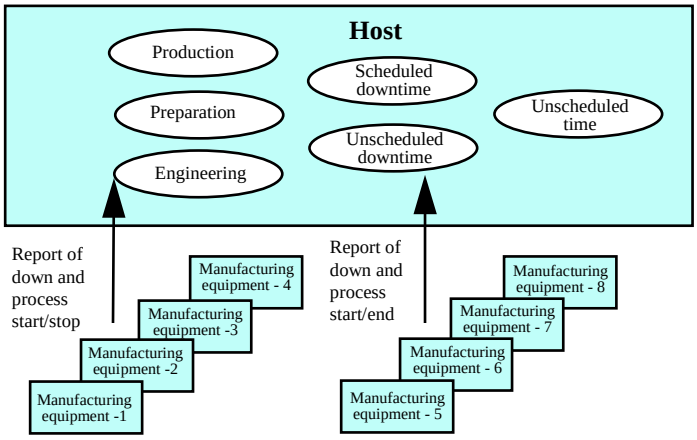

Figure 17 Host-Based Equipment Operating-Time Management

Operating-time management requires these functions from the system (the host and equipment):

- Substates to the equipment state, such as Processing and Ready, must have an input method to cause state transitions relating to the equipment state of equipment environmental variables. There must also be a way to display this information for confirmation.

- Comment input and comment selection for transitions from operating substates to nonoperating substates, and a method for entering all reasons for nonoperational status.
- Total time in all substates, plus an output function for generating daily, weekly, and monthly reports.

Most of these functions are already supported at the host end of the system.

If most processing equipment were made available with the necessary standardized input methods for operating-time management, both the resources and the effort required for operating time would drop, and management would be more precise. For example, equipment could automatically detect commands from the host, operator inputs, or internally-generated commands, make state transitions, and report these to the host. The host would calculate operating times for groups of equipment, process the data, and produce reports. Figure 18 shows how this would work.

A problem with this arrangement is that it doesn't work if equipment is not on line. If most of the equipment in a fab is not part of the system, the beneficial effects will be diminished. If there are differences in the way users have defined operating time, or in input procedures, then custom software may be needed at the equipment end.

Detailed information from equipment subsystems is needed to get an accurate picture of equipment throughput. This may, for instance, require sensor data from every unit. This information can be reported by the equipment to the host using event reporting.

Most production machinery comprises multiple subsystems. The SECS-I data-transfer protocol is not up to the task of collecting information so detailed that it that includes the operational state of these subsystems. HSMS, which has more bandwidth, is both necessary and sufficient for conveying this sort of operational data to the host. Data-collection capabilities should be redesigned with HSMS in mind.

Event reporting is the GEM capability at the core of this feature.

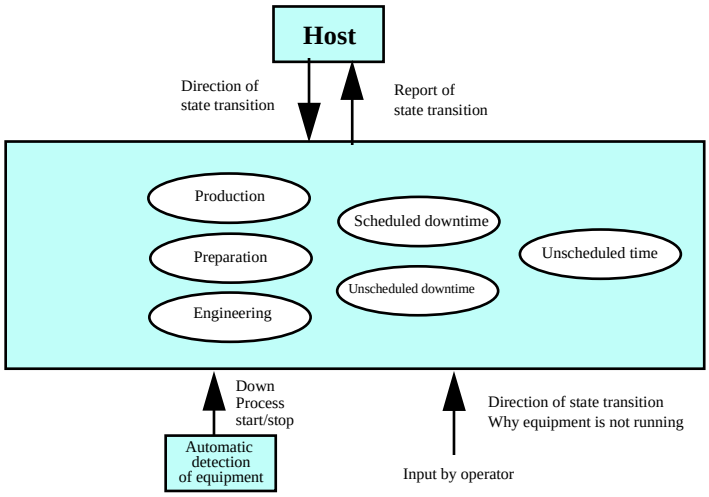

Figure 18 Equipment-Based Operating-Time Management

Automated material transport and remote control. "Remote control" means operations executed by equipment at the command of the host. This is a necessary for running an automated material transport system of robots, automated guided vehicles (AGVs), etc. Figure 19 shows a schematic of how these systems work.

One condition for a remote-control system is that the host be running remote-control software. Ordinarily, this software is developed as part of a broader automation project. The software will gather material-transport reports and process-complete reports from production and inspection equipment, and collect transport-complete reports from AGVs and the like via the automation system.

Whereas the production and inspection equipment accept process conditions and begin-processing commands (using the remote control capability), the automated transport system accepts transport-material commands. It also needs to accept recovery messages in trouble situations. Since the configuration of automated transport systems will vary from one fab to the next, this is not yet an issue amenable to standardization.

Prompt pickup and delivery at the processing equipment is a key issue in automated transport, which requires accurate forecasts of processing times. Other important points are the ability to detect and recover from trouble situations; communications between the processing equipment and transport system; and finally an interface between the two. The communications interface has been defined in the E32-1994 standard, apart from GEM.

The GEM capabilities relevant to automated transport are material transfer, process program management, remote control, event reporting, and alarm management.

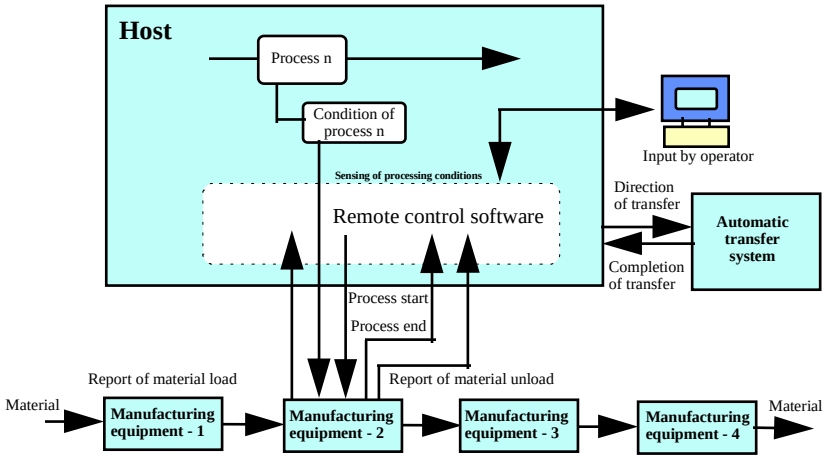

Figure 19 Remote Control

Realtime monitoring of equipment status Continuously updating the host's database with the information coming in via SECS connections allows you to track equipment operations information in real time from any host terminal. For instance, a process engineer could check recipe data and current settings from his desk, or an equipment engineer could find out what the operating statistics are for a machine, on-demand and right away. Production managers can check the operating parameters at any piece of equipment, whether any equipment is down, and get a general overview of current operating status in real time. Figure illustrates this capability.

GEM capabilities that come into play here are process program management, event reporting, status information collection, and alarm management.

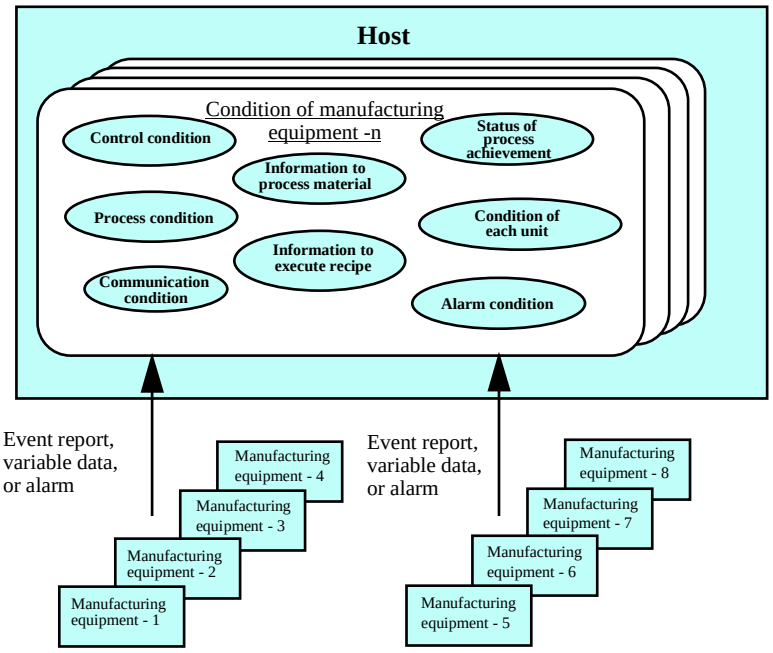

Figure 20 Realtime Monitoring of Equipment Status

7 EQUIPMENT OPERATION

7.1 Equipment Operation and GEM Capabilities

Specific GEM capabilities relating to equipment operation are as follows:

1. Equipment start
2. Establish communications
3. Set control (set online)
4. Initialize equipment settings
5. Execute materials processing (includes multiple simultaneous processes)

Figure 21 shows the relationships between specific GEM operations and equipment operations. The vertical line on the left side of the figure represents the procedures relating to start-up and operations. The left side of the figure shows a list of specific GEM capabilities. In most cases, these all can be executed in parallel when the equipment is in operation. After equipment startup, communications are established, and error-message capability is enabled. Shortly after control status variables have been set, other GEM capabilities are activated. The capabilities activated after the control status are usually event settings, alarm on/off setting, time settings, equipment piece count setting, and spool parameter settings.

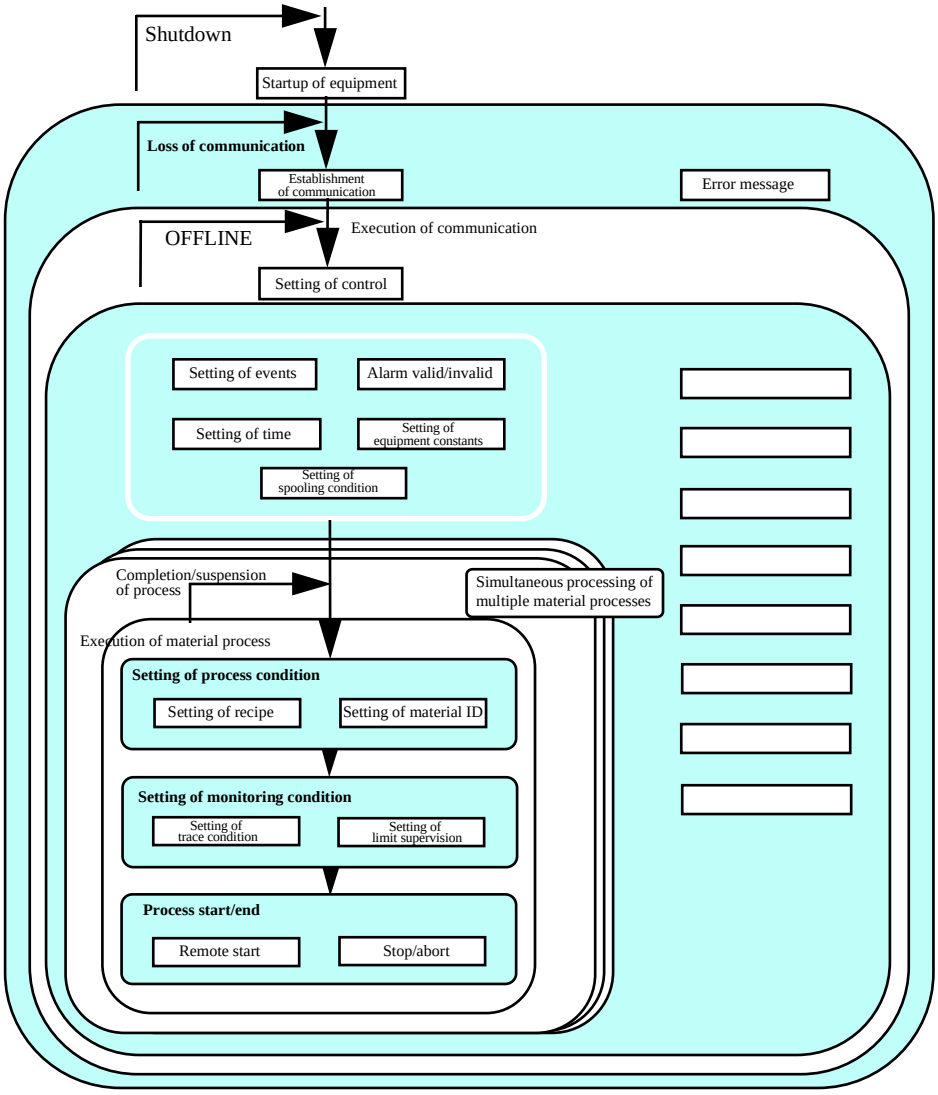

Figure 21 Equipment Operations and GEM Capabilities

Equipment operations and GEM scenarios. Table 1 shows in detail how specific actions at processing equipment relate to GEM scenarios. This example does not define the events or the order of events in an action.

Table 1 Relations of Specific Actions to GEM Scenarios

Action	Scenario	Comment
Equipment start-up	none	Power supply turned on
Establish communications	Host and equipment exchange S1, F13/14 messages	If successful, communications state goes to “Communicating”
Request online	Either S1, F1/F2 sent by the equipment or S1, F17/18 sent by the host.	If successful, online status goes to “Online.”
Define events	Host sends S2, F33/34, F35/35, or F37/38.	Defines event reporting. Need not be repeated after each restart once set.
Set time	Either S2, F17/F8 sent by the equipment or S2, F31/32 sent by the host.	
Alarm on/off	Host sends S5, F3/4	Need not be repeated after each restart, once set
Set equipment constants	Host sends S2, F15/16	
Set spool parameters	Host sends S2, F43/44; F15/16	Need not be repeated after each restart, once set
Set recipe	S7, F*xx* sent between host and equipment	Exact operation will vary depending on whether the recipe itself is normally downloaded.
Set material ID	Host sends S2, F41/42	
Set trace parameters	Host sends S2, F23/24	Trace begins when set
Set limit monitor on	Host sends S2, F45/46	Need not be repeated after each restart, once set
Remote start/stop/abort	Host orders S2, F41/42	

7.2 Equipment Operation Procedures

Operations procedures should be understood in terms of the relationship between GEM capabilities and equipment startup and actions. Operational procedures generally follows the flowchart shown below. This shows the GEM considerations and necessary parameters for each step.

Equipment start-up
[Power on, reset, initialize]
↓
Synchronization
[Establish communications, request online]
↓
Preprocessing
[Load material, set process program, tune processing parameters]
↓
Process wafer
[“Process start” command, collect process data]
↓
Postprocessing
[Unload material, format collected process data]
↓
Shutdown

Equipment startup. This involves switching on the primary power supply for a piece of equipment and following the procedure for activating each of its subsystems. Since this procedure will be specific to the particular hardware/software configuration, it is not specified in GEM. In configurations where the actual equipment can be activated separately from the host-interface controller, communications between equipment and host may only be available once the host-interface component has been activated, although if the processing equipment itself has not started up, then processing cannot begin, of course.

Equipment startup covers everything involved in bringing the equipment to standby status, including pump-up, gas supply/exhaust setup, etc.

The equipment may be in manual mode immediately after startup. In manual mode (standalone mode), all actions are executed by the operator manually. Communications status will be “not ready.”

The communications protocol and all parameters defined in GEM become available just after startup. Equipment parameters, which are all stored in nonvolatile memory, are available just after startup. This allows full synchronization with the host at startup time. If these parameters can be wiped by resetting the machine, then the machine is not GEM-compliant.

Table 2 shows variable parameters that GEM expects to have stored in non-volatile memory. Constants of course should be stored in non-volatile memory; constants that need to be available at startup are also shown for reference purposes.

Table 2 Equipment Parameters to be Stored in Nonvolatile Memory

GEM capability	Parameter name	Data description	Comment
Establish communications	Establish communications timeout (constant)	Establish communications timeout	Time in seconds within which S1, F13 must be sent during communications set-up
Control	Control state during equipment startup (constant)	Control State (variable)	Online/offline
	Online substate during equipment startup (constant)	Control State (variable)	Test equipment online/offline/host offline
Event reporting	The S2, F37 “event-enable” parameter	Events Enabled reflected in enabled event CEID	Determines whether events are transmitted
	The S2, F33 “define event” parameter	RPTID, VID	Assigns an ID number to a report, as well as a VID for report type number
	The S2, F35 “link event report” parameter: event ID, report ID	CEID, VID	Report ID linking to a collection event
Alarm management	Alarms on/off	Alarms Enabled reflected in enabled alarm ALID	Determines whether alarms are transmitted
Limit monitor on	All data defined by S2, F45, “define variable limit attribute”	VID, LIMITID, UPPERDB, LOWERDB	Defines limit-supervision variables and high-low limits
Spooling	All data defined by S2, F43, “define spooling streams and functions”	STRID, FCNID	Defines spooling messages
	All constants and variables required for spooling	MaxSpoolTransmit (constant) OverWriteSpool, SpoolCountActual, SpoolCountTotal, SpoolFullTime, SpoolStartTime	MaxSpoolTransmit sets S2, F15

Synchronization. This involves establishing communications between host and equipment, and setting the control state.

In GEM, the communications state and control state are distinct. In previous host-equipment communications schemes, the difference between the communication state and the control state (in which the host could run the equipment remotely) was not clear. In GEM, however, the connection and initial relationship between equipment and host are defined in three distinct levels:

1. *Physical connection between equipment and host:* This is the physical link and protocol connecting the host and equipment. GEM is not concerned with this level. Once all the settings at this level have been taken care of, host and equipment are electronically linked, assuming both are turned on. Errors (baud rate, parity bit, etc.) relating to the physical connection and communication protocol are handled at this level.
2. *Communications establishment between host and equipment:* So that each side can inform the other that it is ready to communicate, each sends out and accepts a specific message. The equipment informs the host that it is ready to communicate and that communications previous to this have been cut off. The host sends back confirmation to the equipment that communications have been established.
3. *Set control state:* The control state must be set, either via the user interface or from the host, prior to running remote control. This determines the extent of the host's abilities to run the equipment by remote control. The control state as set is transmitted to the host over the communications lines.

Each layer must be started in order, as each depends on the previous one. Once all three have been started, the host can begin online control of the equipment. The first level here is the province of SECS-I, and the second and third are covered by SECS-II and GEM.

The first part of synchronization is establishing communications, followed by setting the control state. Establishing communications between host and equipment, and setting the control state are distinct tasks that are executed separately. Note that if the equipment does not notify the host of its current control state, the host will not be able to properly begin remote-control operations. That is why the synchronization procedure first has communications-establishment transaction and then a control-state setting transaction.

Host		Equipment	
		< S1, F13	Establish communications request
Establish communications acknowledge	S1, F14 >		
		< S1, F1	Switch online request
Online acknowledge	S1, F2 >		
		< S6, F11	Notify online
Online acknowledge	S6, F12 >		

Preprocessing, wafer processing, postprocessing

***Preprocessing*:** Ordinarily, equipment preprocessing involves material loading, carrier mapping, recipe downloading, process-condition tuning, etc. When multiple lots are being processed in parallel, preprocessing, wafer processing, and postprocessing may be going on in parallel as well. Chamber precleaning and preheating may also be involved in the preprocessing for each cassette. These preprocessing execution states can be reported to the host via event reporting.

***Wafer processing*:** Wafer processing begins after either the Start command comes from the user interface or the Remote Start command comes from the host. During processing, process data

(event reporting, trace data, limit monitoring, etc.) is relayed to the host. Process start, end, and interrupt events must always be reported to the host.

Postprocessing: This involves material unloading and postcleaning. It is best to issue event reports with accurate start and stop times for all of these processes, as these report make it possible to monitor equipment status accurately and to perform statistical analysis of operating-time figures.

Shutdown. Shutdown involves powering-down the processing equipment and shutting down all of its functions. In processing equipment with built-in controller computers, this involves shutting down that computer as well. It is best for the equipment to transmit a shutdown message to the host at time of shutdown.

7.3 Operational Design

Each plant's online operational style is going to be different, but they can generally be divided into two categories:

1. ***Equipment-driven (online with local control):*** Here, the operator executes all actions through the equipment's user interface. The operator is also directly responsible for all material loading.
2. ***Host-driven (online with remote control):*** Here, the host remotely orders all actions. The operator may be directly involved in material loading, or it may completely roboticized.

Almost all fabs operate on one of these two plans, or some combination of the two.

Table 3 shows typical messages sequences of startup, begin processing, and processing complete, for both of these styles. The figure shows messaging in abbreviated form only, and does not contain entire messaging scenarios.

It helps to think of operations as being host-driven or equipment-driven, as this clarifies the division of labor between the two—this division tends to be vague in most existing SECS scenarios. There tends to be a lack of coherence in operations and the design of message sequences when an operation is sometimes conducted remotely and sometimes locally.

Thinking in terms of host-driven and equipment-driven operations allows organization of message sequences into coherent scenarios, and it should be easier to make subsequent revisions to message sequences. In the host-driven scenario in Table 3, the operator is responsible only for equipment startup and material loading/unloading; everything else is run from a host terminal. In the equipment-driven scenario, the operator executes all actions through the user interface.

Even in a host-driven setup, the equipment needs to be started up locally. That is why the establish communications request and switch online request are usually both initiated by the equipment.

Table 3 Host-Driven and Equipment-Driven Scenarios

Host-driven scenario			Equipment-driven scenario		
Host		*Equipment*	*Host*		*Equipment*
Est comm req	S1, F13 >			< S1, F13	Est comm req
	< S1, F14			S1, F14 >	
Online req	S1, F17 >			< S1, F1	Online req
	< S1, F18			S1, F2 >	
	< S6, F11	Online rmt rpt		< S6, F11	Online loc rpt
	S6, F12 >			S6, F12 >	
Set time	S2, F31 >			< S2, F17	Set time
	< S2, F32			S2, F18 >	
Set alarm on	S5, F3 >		Set alarm on	S5, F3 >	
	< S5, F4			< S5, F4	
	< S6, F11	Material set-up		< S6, F11	Material set-up
	S6, F12 >			S6, F12 >	
Download recipe	S7, F3 >			< S7, F5	Req recipe DL
	< S7, F4			S7, F6 >	
Req proc start	S2, F41 >			< S6, F11	Rpt proc start
	< S2, F42			S6, F12 >	
	< S6, F11	Rpt proc complete		< S6, F11	Rpt proc complete
	S6, F12 >			S6, F12 >	

Remote control, and simplifying/standardizing equipment actions. Different kinds of equipment will have different procedures for operations conducted via their user interfaces, but these need to be standardized and simplified to facilitate remote operations by the host. Doing so has a number of other benefits: it makes personnel transfers and new equipment investments smoother, and reduces the time required for and likelihood of error in programming.

Figure 22 shows the difference between local online control and remote online control. At processing equipment #1 (online with local control), the operator requests processing parameters from the host, via the user interface, and then gives the order to begin processing. The material ID is entered via the user interface also. At processing equipment #2 (online, with remote control), all actions taking place in the equipment are being ordered from a host terminal. Each arrangement has its pros and cons.

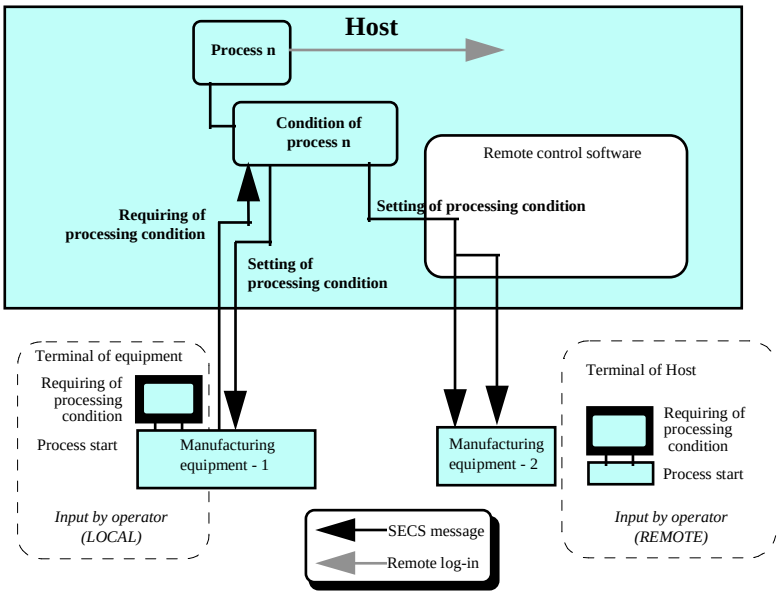

Figure 22 Online with Remote Control and Local Control

Remote control is preferable for those aspects of operations that have a standardized, simplified, and interactive operator interface. The host terminal can provide its operator with a range of services needed for production management. For example, this arrangement provides instant access to the host's database, which contains data useful for making decisions. Further, if the terminal can read the operator's ID, it can be configured to show only the information appropriate to that user. Remote control together with other GEM capabilities should allow almost all equipment operations to be handled from the host. If the equipment's software was written with GEM in mind, then host terminals should provide operators with sufficient remote control over equipment capabilities. This allows an operator at one terminal to access the host at the same time that he initiates an operation at the processing equipment. Furthermore, by using HSMS (TCP/IP), equipment-based terminals may be able to remotely log in to the host—this would allow an equipment terminal to take on the function of a host terminal.

The main obstacle to remote operations is those plants that don't have completely automated transport systems—in these cases, there need to be host terminals near every piece of processing equipment.

The other arrangement, with equipment being online but operated locally, allows operators to work at the equipment terminal, which increases performance. Nevertheless, if you upgrade equipment software to match plant operations, you will find that the initial development costs rise, that operator panels at each piece of equipment grows less and less alike (and harder to keep consistent), and that there are limits on the ability to transmit interactive data by SECS-1.

Equipment that can be operated using fundamentally simple actions probably does not need a host terminal present. If there is just a barcode reader for transmitting material IDs and operator IDs to the host, the host may be able to handle all subsequent operations automatically. This would fit in with a remote-control arrangement.

Notes on automated transport systems and patterns of equipment. It is helpful to organize method of equipment operations into a few general types, which range from standalone operations to fully automated operations:

1. All actions executed manually
2. Equipment is controlled through the host, except for material transport, which the operator handles manually. This approach also works with local online operations, where the operator is executing actions through the equipment's user interface.
3. Equipment is controlled through the host, and material transport is completely automated through the use of robotics, etc.

These three schemes are summarized in Table 4. One can observe certain patterns in how operating modes and operation types line up: the manual operating mode corresponds to the standalone operation type, the automatic mode corresponds to the online operation type.

In types 2 and 3, having the automated transport off, communications state off, and control state offline would indicate either the startup operation or regular maintenance, but in any case, something other than regular operations.

Table 4 Patterns of Equipment Operations

	Type 1	Type 2	Type 3
Material transport	no	no	yes
Automated transport modes	none	none	on/off
Operations modes	manual	automatic	automatic
Communications states	off	on/off	on/off
Control states	none	remote/local	remote/local

Try to stay flexible, so that the order of execution for GEM capabilities fits with operating scenarios. GEM places no restrictions on the order in which its various capabilities must be executed. In fact, almost all of them should be executable in parallel or in any order without causing problems. Order of execution will be customized to fit the function of each piece of equipment and the plant's operating scenarios. You should set things up so that you can select different orders of execution on the equipment side. Although the host must be able to supply standard GEM scenarios, the equipment must be amenable to revisions.

Examples of different process-start scenarios. Table 5 shows different process-start scenarios. In order to simplify the examples, some event reporting has been eliminated. In the "host-request" scenario, the host is aware of what kind of material is being transported, and has already downloaded the appropriate processing parameters to the equipment. If the equipment winds up not being set up for the expected materials, there will be some sort of error processing.

In the "equipment-dependent" scenario, the equipment can be set up for any kind of material. After the material has been loaded in the equipment, the host must be updated on the material ID and recipe ID by the equipment, after which it will acknowledge this information and download the appropriate processing conditions.

One version of equipment software should be able to handle both of these arrangements.

Table 5 Different Process-Start Scenarios

Host-request scenario			**Equipment-dependent scenario**		
Host		*Equipment*	*Host*		*Equipment*
Download recipe	S7, F3 >			< S6, F11	Material set-up
	< S7, F4			S6, F12 >	
	< S6, F11	Material set-up		< S7. F5	Req recipe DL
	S6, F12 >			S7, F6 >	
Req proc start	S2, F41 >			< S6, F11	Rpt proc start
	< S2, F42			S6, F12 >	
	< S6, F11	Rpt proc complete		< S6, F11	Rpt proc complete
	S6, F12 >			S6, F12 >	

8 INDIVIDUAL GEM CAPABILITIES

Following are the basic prerequisites and additional capabilities in GEM. The next section will explain these capabilities in the order below.

8.1 GEM Basic Prerequisites, Additional Capabilities, and Messages

8.1.1 Basic Prerequisites

- State model
- Equipment process model
- Online acknowledgment
- Error messages
- Documentation

8.1.2 Basic Prerequisites/Additional Capabilities

- Host-initiated S1, F13/F14 scenarios
- Communications acknowledgment
- Control (operator-initiated)
- Control (host-initiated)
- Event reporting
- Dynamic event report settings

8.1.3 Additional Capabilities

- Variable data collection
- Trace data collection
- State data collection
- Alarm management
- Remote control
- Equipment settings

- Process program management
- Material transport
- Equipment terminal services
- Clock
- Remote monitoring[5]
- Spooling

8.1.4 GEM Capabilities and Messages

Newly-defined capabilities and messages for GEM: This is not anything you can find in the existing SECS specifications, or any standard specification, so strictly speaking, GEM cannot be compared with SECS. Below are new or additional capabilities that have been defined for GEM.

- State model and equipment process model
- Host-initiated S1,F13/F14 scenarios
- Event notification and dynamic event report settings
- Control (operator- or host-initiated)
- Remote control
- Equipment terminal services (operator acknowledgment)
- Remote monitoring
- Spooling

Table 6 shows a breakdown of GEM-noncompliant messages (that is, messages newly defined for GEM or duplicated in GEM) and compliant ones. This shows the original messages on the left and GEM message substitutes on the right.

Table 6 SECS Messages That Do Not Comply with GEM

GEM capability	Noncompliant message	GEM substitute message
Event reporting	S6, F3/F4 S6, F9/F10	S6, F11/F12
Variable data collection	S6, F7/F8	S6, F15/F16 S6, F19/F20
Remote control	S6, F27/F28 S6, F21/F22	S6, F41/F42

8.2 Explanation of Basic Prerequisites

8.2.1 State Model

The state model is the way to visibly express a device's internal status or actions in GEM. The Harel statechart is used as the state model notation. Please refer to Section 3 of E30, **State Models** and **Applications**.

[5] The description of remote monitoring in this document is abbreviated.

When implementing the capabilities described in the GEM state model, all functions must be implemented as defined in the state model. Actually bringing the state model online should give you a better understanding of the correct procedures involved in transmitting messages, control sequences, and equipment operations from the host.

The state model is a basic prerequisite: GEM uses the state model to describe the dynamic aspects of equipment capabilities. The state models needed to implement GEM's basic prerequisites are the processing state model, the control state model, and the communications state model.

State model expansions and GEM compliance: The state model may be expanded to support expanded GEM capabilities. As long as the original model is preserved and not interfered with, these expansions will still be GEM-compliant. GEM does not define a maintenance mode, a service mode, or continuous/stepped processing sequences. The operational states needed for the actual performance of these are to be supplementally defined for each piece of equipment. This document has been expanded with this principle in mind.

8.2.2 Equipment Processing Model

The old host-equipment interface spec is inadequate for equipment operating states. The equipment processing statechart must have a timetable of its internal processes so that the host can be kept current. It is difficult to achieve this—notifying the host of the equipment's specific actions—using only existing host-equipment interface specifications (SECS). Interface software can be developed that reports on the equipment's order of operations, but this requires an engineer at the host and another at the equipment to go through repeated setups. Though technically speaking the communications interface can support it, it requires considerable work, increasing development time and lead time. Clearly, there is a lot of room for improvement.

The processing state model alone is not enough, but it is very helpful for the host-software developer, in that it gives a better understanding of the equipment's internal workings, and it gives a useful framework for deciding how to design the equipment interface.

The host's interest is in changes of material state and changes of equipment state. The host is mainly interested in two important processes in the equipment. One is the "change in material state," that is, what happens as material is loaded in the equipment, subjected to certain processes, and then unloaded. The other is the "change in equipment state." In this, it is monitoring changes in the equipment's state, according to material ID, as the equipment simultaneously processes several units of material. In some cases, this may also require monitoring of state changes in supplementary materials, as per a statechart. Tracking state changes in the equipment may permit a single statechart for the entire apparatus, or it may require separate statecharts for each subsystem.

Goals for the equipment processing statechart. When writing host-end software, developers should have possess certain knowledge sets, including the following:

- A comprehensive understanding of all equipment process actions
- An understanding of internal material-processing steps
- An understanding of state changes in the primary components of the equipment

- Knowledge of the transmission timing for process program downloads and remote control commands.
- Knowledge of the timing when major events are generated, such as process-start and process-complete

Developers all should be prepared to conduct:

- An examination of recovery operations for errors
- An examination of event-collection timing for operating time statistics

Referring to the statechart, if one collects the event reports generated during the transition periods between states, then the host can track the equipment's state in real time and transmit the proper action directives to it.

The statechart is not just documentation for its own sake. The equipment must run according to the statechart.

Processing state models differ for each type of equipment. GEM does not define the substance of statechart: Figure 23 shows an equipment processing statechart from E30. Since the actual configuration of the equipment may be exceedingly complex, it can be difficult to represent the processing states completely just using the statechart. While GEM does require the equipment supplier to document a processing statechart, it does not tell you what it should look like.

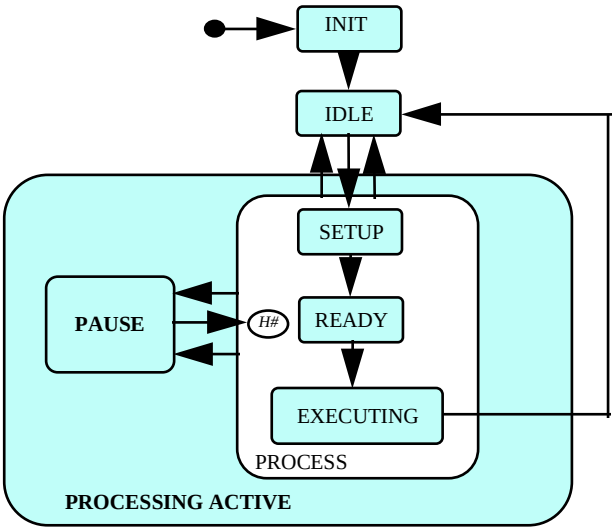

Figure 23 E30 Equipment Processing Statechart

Additions to the state processing model: A processing statechart probably needs to include a statechart for input/output ports (including wafer/casette information) and process setup state transitions. There are a number of processing state models, material state models, and port state models, which are desirable in that they allow the host to monitor the operating state of the equipment, what material is in transport or is being loaded, etc. And there need to be models that describe the operational status of the equipment.

The scope of processing state models: The states and state transitions expressed in processing state models will differ from one machine to another. However, all machines must support at least the following states:

- Wait for operator command or host command, transition to next state upon receipt of command
- Currently executing a process internally, will generate event upon completion and transition to next state

Figure 24 shows a model of this. The numbers in the figure represent the order of state transitions.

1. Receipt of command from either operator or host to begin processing. If received from operator, a “processing start” event report is sent to the host.
2. Processing-start conditions are satisfied, so machine begins material processing. Event report sent to host.
3. Processing of one basic processing unit (a wafer, casette, etc.). is completed. If necessary, process results and process completion are sent as an event report.
4. Processing of remaining material begins.
5. Processing of all material is completed. A processing-complete report is sent to host. An event report with aggregate processing results is generated.
6. Receives “Pause” command from host or operator. Transition to Pause state, send event report.
7. Receives “Resume” from host or operator. Return to state preceding Pause command, send event report. If the previous state was Processing, the decision of whether or not to return to the Processing state will differ from machine to machine. For this example, we assume it can return.
8. Receives “Abort” from host or operator. Terminates processing, transitions to Idle state, and sends event report.

Thus, the processing state model expresses the relationships of states or state transitions to the host or operator.

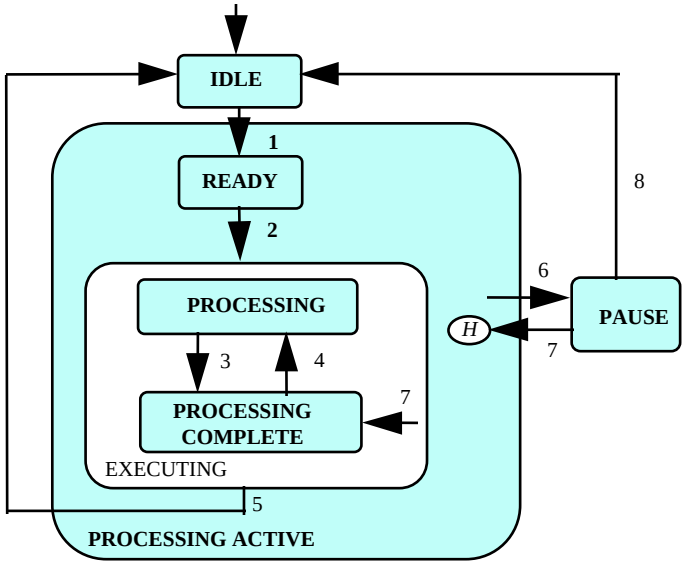

Figure 24 Example of a Processing State Model

Internal transitions that do not concern the host are concealed. State model requirements mostly consist of states and state transitions that the host needs to know about in order to manage the equipment. Internal state transitions that do not concern the host may be omitted. A complicated processing statechart would only serve to confuse things at the host end.

Consider Figure 24 again. Suppose the host is interested in knowing whether the equipment has completed processing of the last batch, and that it is not interested in tracking the processing of individual wafers or cassettes. If this is the case, then a state model as shown in Figure 25 would be sufficient. Here, only transition 5 would generate an even report, once all materials had been processed.

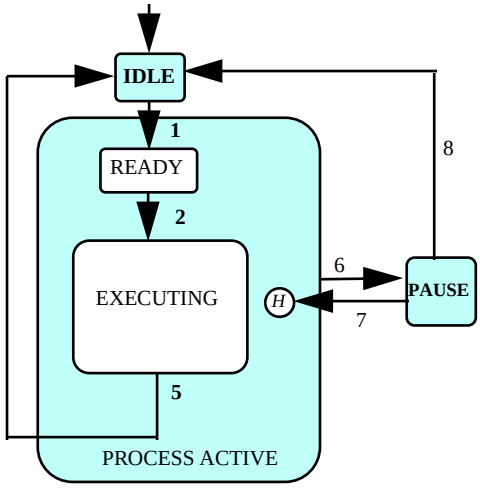

Figure 25 Simplified Example of a Processing State Model

How detailed should be the processing state model provided by suppliers? Make a list of the functions provided by the equipment. Of these, the ones that the host would need to be able to

control will be considered pertinent to GEM. The processing state model should cover all states related to the host interface that the equipment comprehends. Non-essential states may be removed from the documentation (that is, concealed). Alternatively, it is advisable to give the host some way to ignore them.

Figure 26 shows an example of this principle in action. The boxes represent a device, a subsystem, or a state. This roughly comprises the material transport input port, material transport unit, and processing chamber. A more detailed processing statechart would include all detailed state transitions for the various subsystems. Depending on the type of equipment, the material transport unit and a single or multiple processing chambers may be combined into a single processing statechart, the material transport input port statechart may be omitted, and the entire piece of equipment boiled down to a single statechart. In any case, the criterion for inclusion in a statechart is whether an item can be monitored or sufficiently controlled by the host.

The heavy arrows in the diagram show the material transport path. The host is also concerned with the various subsystems that the material passes through and the timing of these transfers (represented by arrows on dotted lines). These material transitions and the processes and state transition models they entail are also useful to include.

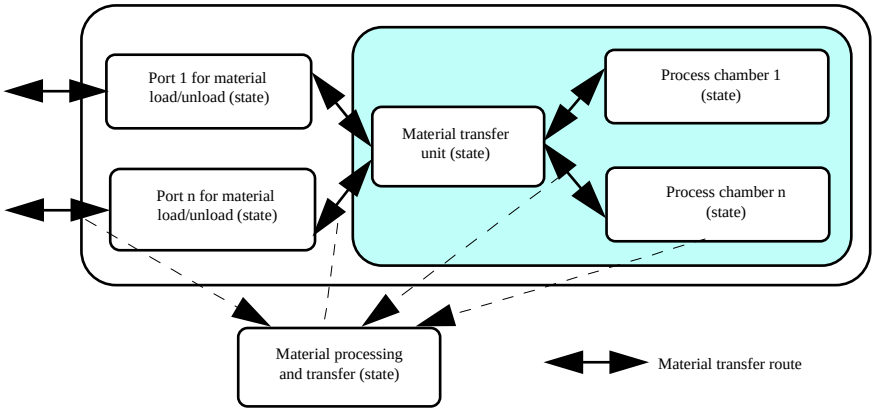

Figure 26 Depth of Equipment Processing Statechart Model

Are there standards for processing state models? There are two approaches to the processing state model: one is that regardless of equipment, the processing state model must comply with E30; the other is that different state models for each piece of equipment are acceptable.

Viewed from the host's perspective, the processing state model holds that all pieces of equipment can be treated the same way. From the equipment's perspective, the processing state model holds that depending on the type of equipment, there is a certain amount of flexibility available.

To satisfy both these approaches, it should be possible to use a multilevel approach to the processing state model, so that at the topmost level shows the host all of the equipment's universal processing state models. At the intermediate level, batch and single-wafer process types would appear, showing the different process types for the many different types of equipment. At the lowest level, every processing state that is comprehended by a certain piece of equipment appears in the processing state model. This would go from the very abstract to the very specific, and if implemented, would allow the host and equipment to interact using the state model most appropriate to the task at hand.

In this setup, the top level would be useful for the host collecting production management data, such as material processing states, processing parameters, loading capacity, and information relating to equipment utilization rates. The bottom level might reflect the equipment's specific processing functions and hardware configuration in the state model.

Equipment processing state display. Although GEM does not define this, displaying the current equipment processing state is going to be necessary. While the example shown here is drawn from a character-based display, a graphical display would convey the information in a more understandable form.

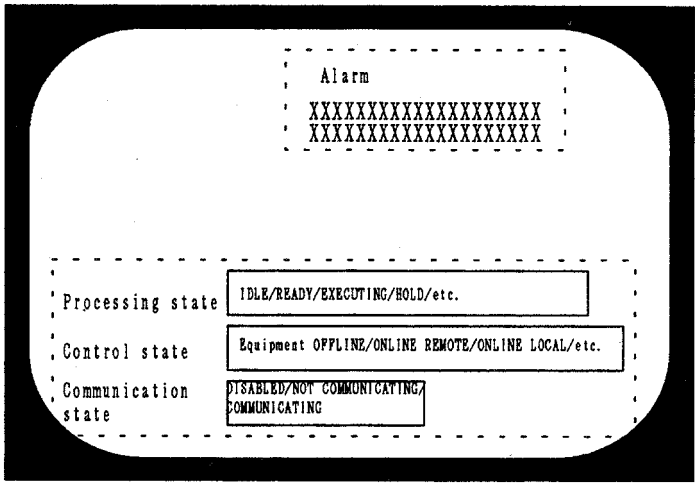

Figure 27 Character-based Equipment Processing State Display

8.2.3 Online Confirmation

Online confirmation and control capabilities are interrelated. This sections examines these capabilities and how they are similar or different.

Issuing S1, F1 mean different things to the host and the equipment. When the host sends S1, F1 to confirm online, the equipment replies with S1, F2 and that is the end of it. If the equipment generates the S1, F1 message, its significance is different from the same request when generated by the host. While the host's message is simply "confirm online," the equipment's version is "request confirm online." The S1, F1 message as generated by the equipment is treated as a control capability.

The host can transmit S1, F1 at any time. The equipment can only issue the S1, F1 message when it is set to the control state. The host, however, can generate this message anytime it needs to confirm online status.

Heartbeat monitoring. With this online confirmation function, the host can periodically send the S1, F1 message to the equipment to continuously monitor the equipment's online status. This is called "heartbeat monitoring." This is especially useful in plants where the material transport is completely automated or operators are otherwise not present at the equipment, as it allows real time detection of irregularities with the equipment or the communications system.

8.2.4 Error Messages

Error message transmissions have priority over all equipment states. Assuming communications are operative, the equipment will inform the host of detected messages or communications irregularities using the error message suite (S9, Fx). If there is an interruption in communications for example, regardless of online status or spooled action status, the equipment will notify the host when it detects an irregularity in the host's messages by transmitting an error message. The error message capability has priority over other GEM capabilities. (See Figure 21, Equipment Operations and GEM Capabilities.")

If the equipment receives messages it does not support from the host, should it respond to the host with the Unrecognized error message? Messages will ordinarily be defined in the standard, but if the equipment should encounter a message not internally defined, it should reply to the host with the "unrecognized stream" message (S9, F3) or the "unrecognized function" message (S9, F5). Note that as in the following example, when the procedure is defined in GEM, the equipment will follow it, even if it doesn't support it.

Host		Equipment	
Transmit information	S10, F5 >		
		< S10, F7	Multi-block not allowed

8.2.5 Documentation

What are the basic parameters for GEM documentation? Most existing host communications specifications have inadequately specific descriptions, so it is difficult to properly understand equipment actions. Following are some guidelines for writing GEM documents. Please refer to SEMI document E30 as well.

- Equipment actions should be noted using the state model;
- Explain SECS-II messages;
- Note GEM compliance;
- Note all collection events for each piece of equipment, plus all conditions under which events are generated;
- Specify all limit-monitoring variables;
- Note and explain all state variables (SV), equipment constant values (ECV), data values (DVVAL), and variable IDs (VID);
- Explain all alarms, alarm IDs alarm texts, and alarm codes;
- Internal timing references should be in units of 0.01 second.

Items that should be defined in the host communications specification. Existing host-communications interface specifications do not follow a common format, so every equipment supplier must deal with a hodgepodge of styles. Following are some guidelines for writing communications specifications.

1. Revision history
2. Contents

3. Applications. Specify objectives of the document and scope of applications.
4. Necessary items pertaining to SECS-I/HSMS. If you are compliant with SEMI standards, you can omit the contents of "SEMI E4-1993." etc., and merely mention them. This is the place to mention any important special items that need explaining, such as message interleaving, multiplexed message checking, revisions to the message header transmitted by equipment, etc.
5. Notation of GEM compliance. Using the E30 GEM Compliance Sheet (E30 Table 8.3), describe the level of GEM compliance. Also, explain anything else relevant to GEM compliance here.
6. SECS-II message list
7. Variable parameters to be set. If there are any variable parameters relating to equipment-side communications, explain the setting procedure and how they relate to SECS-I, SECS-II and GEM here.
8. System overview, system configuration. Use a diagram to explain the equipment's system configuration, including the controller, host interface, and automated transport system interface. Also diagram the physical configuration of the unit and the input/output ports.
9. Equipment operations. Explain the equipment's operation in easily understood terms. This should cover lot makeup, unit of processing, lot insertion and extraction procedures, wafer processing pathways, preprocessing, dummy wafer handling, process program and recipe formats, variable-setting procedures, and finally error processing. Explain the equipment's operational modes using process statecharts. Explain in detail statecharts, transition tables, event reporting, and variables involving settings. Process chamber states, I/O port states, material states, and the like must all be included as well. Explain in detail how these states correspond to the actual circumstances of the equipment or materials, in a way that makes it easy for the host's operator to visualize what is actually happening in the equipment. For GEM-compliant capabilities, it is OK to simply refer to "SEMI E30-1994 Compliance with the Establish Communications capability" and omit detailed descriptions of it.
10. Communications scenarios. Explain the communications scenarios some representative operations. If it is easier to understand an explanation pitched in terms of equipment operations, that is acceptable too. Although specification writers often just cover normal operations and omit any discussion of problem situations, be sure to cover errors and error processing on the host.
11. Explanation of on-screen operations
12. Details of SECS-II message format
13. Overview of variables, collection events, and alarms

8.3 Explanation of Basic Requirements and Additional Capabilities

8.3.1 Host-initiated S1,F13/F14 Scenarios

In GEM, the Establish Communications capability is used to establish host-equipment communications. This can be used for one to inform the other that there has been a period of interrupted communications. In order to establish communications, use the Establish Communications Request message.

There is a procedure for establishing communications when using basic requirements and when using additional capabilities. Using the basic requirements, the host can establish communications by sending the S1, F13 message. Using the additional capabilities, either the host or the equipment can establish communications by issuing this message. Following is an explanation of the various functions involved.

The host-initiated S1, F13/F14 scenario qualifies as a basic requirement. As previously stated, either the host or the equipment can issue the S1, F13 message, and upon receipt of this, the other will respond with S1, F14. This establishes a line of communications. In order to keep the procedure for setting up communications simple in GEM's basic requirements, only the host is required to be able to issue the S1, F13 message; the equipment need not be able to do so.

The communications model using the basic requirements. The basic requirements only support communications establishment by the host. (See E30 4.1.5.1.) Figure 28 shows the communications state model. The diagram's transition numbers correspond to the E30 communications statechart. However, since the S1, F13 message is never sent to the host, handling of this message is omitted.

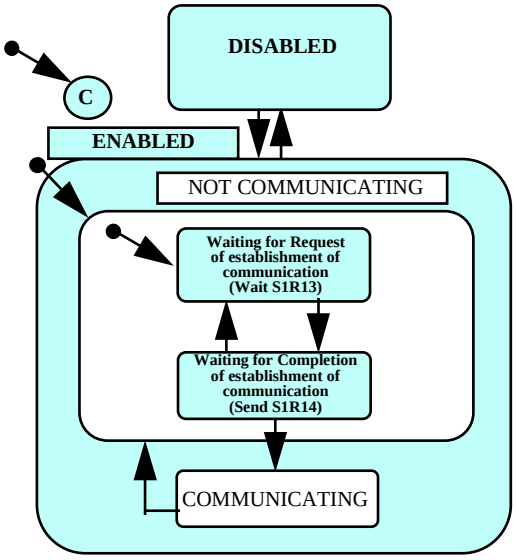

Figure 28 Communications State Model for Host-initiated S1, F13/F14 Scenarios

Under the basic requirements, the host issues the S1, F13 Establish Communications Request. The equipment responds with the S1, F14 Communications Online message. In Figure 28, if the equipment's user interface is used to set the communications state to "enabled," then the equipment first will go into a "waiting for communications establishment request" state

(WAIT S1, F13). Once the host has issued S1, F13, the equipment immediately responds with S1, F14.

Note that when the equipment's communications state is "offline," all messages (including the host's S1, F13) will be ignored. Also, when the equipment is in the "communications enabled" state, any message from the host other than S1, F13 will be ignored.

ENQ and ACK handshaking available even if state is Disabled. Communications establishment uses the higher functions of the block transfer protocol. Accordingly, an electronic connection between host and equipment is assumed to be enabled at all times. In the Communications Disabled state, the equipment will receive the message, but will ignore it, and will not send any reply to the host.

Under the basic requirements, there is no procedure by which equipment notifies the host that it has started up. Under the basic requirements, the equipment does not send the S1, F13 message to the host. As such, even if the equipment controller has started up, there is no way of conveying that information to the host. Likewise, until the host has sent the Establish Communications message to the equipment, there is no way for online operations to begin.

In order to deal with this, the operator must use a terminal to inform the host that the equipment has started up. In terms of automation, this would be considered extra work for the operator, and therefore undesirable, so in actual implementation where automation was a priority, the basic requirements alone would not provide a satisfactory way to establish communications.

The basic requirements do not implement spool capabilities. Under the basic requirements model, the equipment does not issue the S1, F13 message, so despite knowing that the equipment will ignore all communications, there is no way to convey that to the host. As such, even if the equipment does support spooling, there would be no way for the host to initiate a spooling transfer or trigger the dropping of spooled messages. In short, if the communications model of Figure 28 is in effect, spooling is impossible. In order to take advantage of spooling, the Establish Communications capability must be able to use additional capabilities at the client end.

8.3.2 Establishing Communications

Under additional capabilities, either the host or the equipment can issue the S1, F13 message independently. Figure 29 shows the E30 communications statechart. If communications are not in effect, then both the "host-initiated connection" or "equipment-initiated connection" are simultaneously available. In this model, both the host and equipment can set up communications in parallel, simultaneously. Please refer to E30 for explanations of the transition numbers in the diagram.

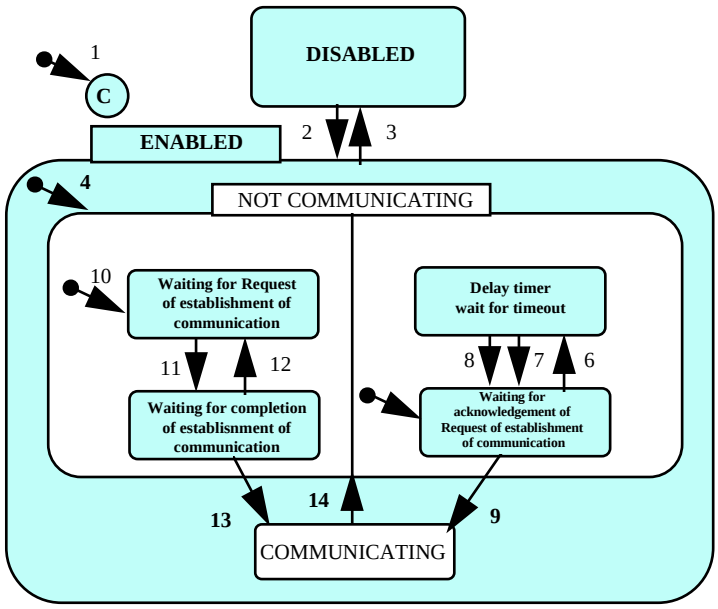

Figure 29 E30 Communications Statechart

Equipment that supports additional capabilities for communications establishment and also supports message interleaving can send out the S1, F13 message and accept the S1, F13 message from the host immediately thereafter. In cases like this, where both sides receive the S1, F13 message more or less simultaneously, both will respond with the S1, F14 message and then communications will be established. If the S1, F14 message is received before the equipment has a chance to send out its own S1, F14 message, communications will be established at that time (the communications state is Communicating).

	Host	Equipment	
Establish communications request	S1, F13 >	< S1 F13	Establish communications request
Establish communications acknowledge	S1, F14 >	< S1, F14	Establish communications acknowledge

Communications establishment is completed once the equipment sends the host S1, F13. When the equipment sends the host S1, F13, the host responds with S1, F14, and communications are established. However, GEM in no way defines the capabilities of the host, so unless it is necessary, the host can drop its end of the S1, F13/F14 transaction in this case. A simple equipment-initiated communications establishment scenario follows. Technically, this is complete once the equipment sends the S1, F13 message.

	Host	Equipment	
		< S1 F13	Establish communications request
Establish communications acknowledge	S1, F14 >		
Communications established			

When the equipment sends S1, F13 to the host, this should be interpreted as a recovery from Not Communicating state. The equipment will never send the S1, F13 message to the host except when it is in the Not Communicating state and it is trying to establish communications. As such, when the host receives the S1, F13 message, it should interpret that to mean that a Not Communicating state exists. Having recognized a Not Communicating state, the proper response is entirely the host's responsibility. GEM specifies that, if necessary, the host should check if it can begin transmitting spooled messages to the equipment. This will probably require that the host confirm the equipment's state information.

The host can check the communications state using S1, F13/F14. The host can issue the S1, F13 message any time it needs to check the communications state. When the equipment receives this message, it will reply with S1, F14 if it is already in the Communicating state. GEM does not specify that the communications-enabled equipment undertake any process in response to S1, F13 other than replying with S1, F14. For example, if the equipment received S1, F13 it could assume that the communications state was Not Communicating , but there is no specification that the equipment should begin spooling.

GEM's capabilities do not include the S2, F25/F26 loopback diagnostic request, so the S1, F13/F14 transaction's usefulness is not limited to establishing communications, it is also useful for confirming communications.

On the other hand, the equipment can only use S1, F13 to establish communications with the host.

Communications state settings display. The equipment's screen should always display the current communications state. In this example, the lower portion of the screen shows Disabled or Enabled, and if enabled, Communicating or Not Communicating. There also needs to be a switch to toggle between Enabled and Disabled.

Whether the Establish Communications operation is successful or more importantly if it fails, there needs to be a function providing feedback to the operator. In this document, an Alarm window is used for this purpose.

When the operator starts up the equipment, he or she can use this screen to set communications to Enabled. After the equipment goes through the Establish Communications procedure, the screen will be updated with the new communications state.

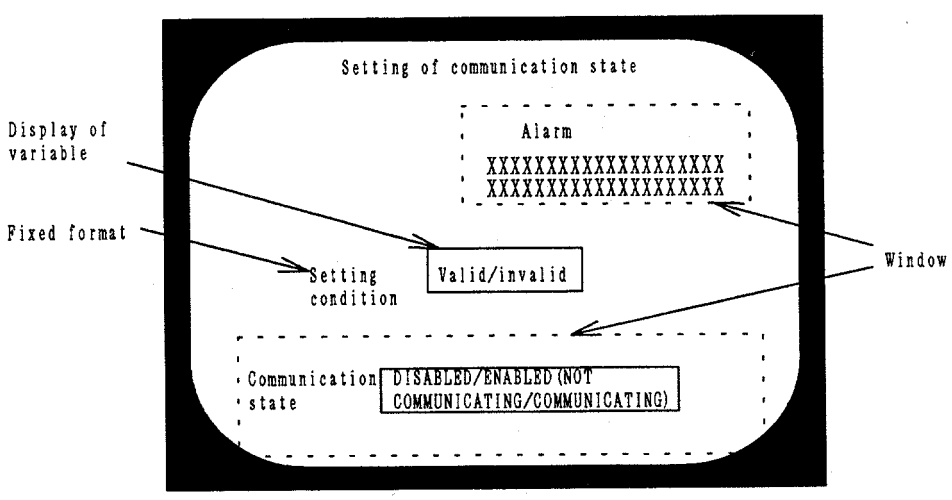

Figure 30 Communications State Settings Display

Continuous communications-establish operations at the equipment. When the communications state is set to Enabled, the procedure to establish communications begins. This will continue until communications have been established at the equipment and the state is Communicating or until the state is Disabled.

Switching communications between Enabled/Disabled does not involve an event report. Regardless of the message, if communications are not enabled, the host will not be able to transmit it. When the operator sets communications to Enabled, an event report cannot be sent to the host, as communications have not been established. Also, if the state is Communicating and is switched to Disabled, the new state is entered immediately, and again, an event report is not and cannot be sent.

Be careful about Enabled/Disabled switching during equipment operations. GEM allows switches between Enabled and Disabled at any time. Even during online operations, all communications can be halted by switching to Communications Disabled. Note that if this happens accidentally during online operations, all messages subsequently sent from the equipment to the host will be lost. Even if spooling is supported, spooling will not be in effect when the communications state is Disabled. Take care to design the operator's screen so that it is difficult to accidentally or mistakenly switch communications to Disabled. In the sample screen below, the communications state setting is accompanied by processing state and control state readouts, to remind the operator what is going on. Changing communications states while a material process is underway could bring up a window requiring confirmation to double-check the operator.

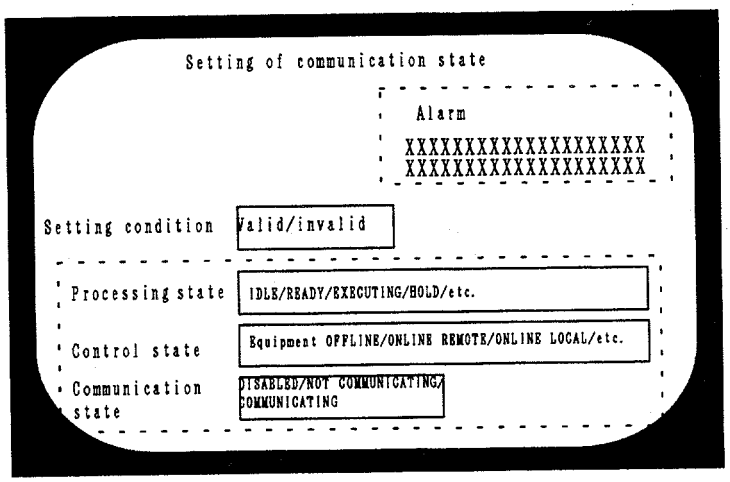

Figure 31 Communications State Setting with Process State and Control State Readouts

Setting up the default settings and timeouts for the communications state. The default communications state can be switched between Enabled and Disabled. If the default is set to Enabled, the equipment will send an S1, F13 message to the host at the same time that its control systems come online.

Because the procedure to establish communications is meant run continuously, there is an Establish Communications timeout, the duration of which can also be set. If an equipment-initiated attempt to establish communications has failed, S1, F13 will be transmitted again after the period defined by this timeout. These values should be entered on the equipment's screen, a sample of which is shown here.

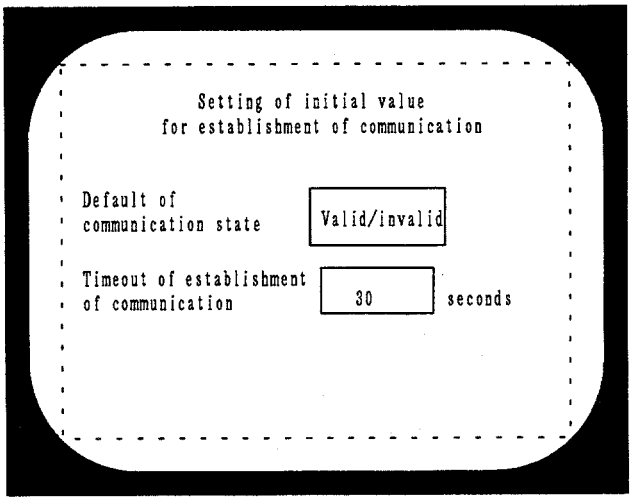

Figure 32 Example of Establish Communications Timeout

Only make the transition to the Not Communicating state when there has been a communications failure. The only time that the communications state should go to Not Communicating is when there has been a Communications Failure, as defined in SECS-I as the

inability to send a message for the RTY limit. This should not be used in the same way as the Communications Fault, which is used for transaction timeouts.

A timeout in the S1, F13/F14 reply before communications have been established results in a Communications Failure. If the equipment detects a reply timeout when attempting to establish communications using the S1, F13/F14 transaction, it enters S1, F13 retransmission processing. In short, it considers this a Communications Failure.

A timeout in the S1, F13/F14 reply after communications have been established results in a Communications Fault. If communications have already been established and the communications state is Communicating, then a timeout waiting for an S1, F14 reply to an S1, F13 message will result in a Communications Fault rather than a Communications Failure. A fault results in sending an S9, F9 Transaction Timeout message, rather than entering S1, F13 retransmission processing. Following is a sample scenario.

Host		**Equipment**	
		< S1 F13	Establish communications request
Establish communications request	S1, F13 >		
		< S1, F14	Communications established
			S1, F14 timeout detected
		< S9, F9	Transaction timeout

When the communications state is Not Communicating, no processes are available except for sending S1, F13 or replying with S1, F14. The equipment ignores all messages from the host when it is Not Communicating except for S1, F13. Consequently, if it receives a message from the host other than this, it need not respond with an Abort.

Even if the state is Not Communicating, if the equipment detects a format error in a message from the host, it will execute error processing. If the equipment detects a format error in a message received from the host, it immediately notifies the host with a system error message. Figure 33 is a flowchart showing the generalized message-receipt processing sequence.

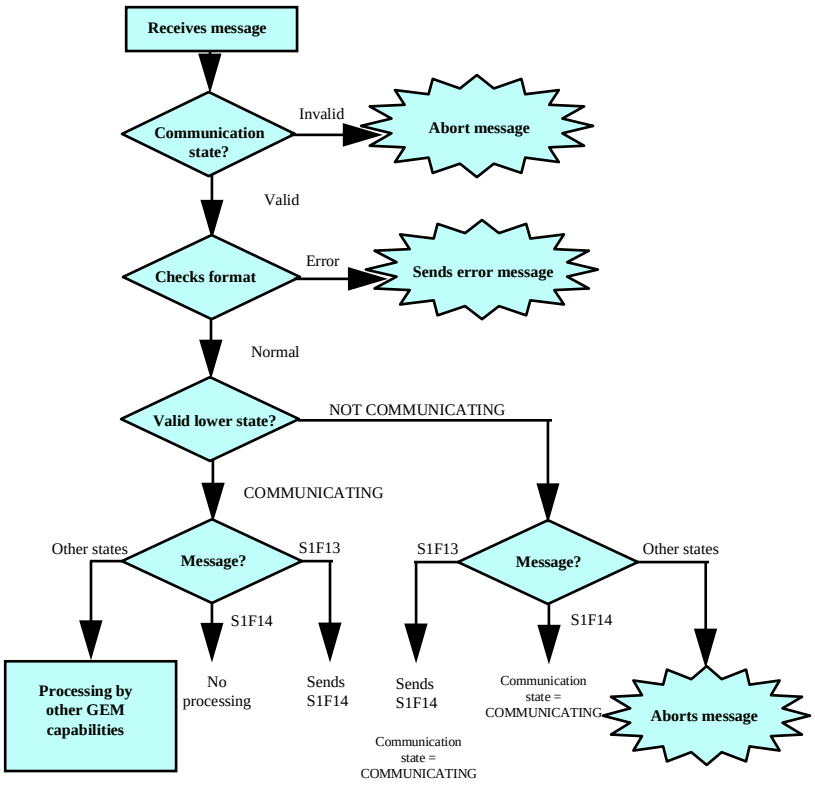

Figure 33 Generalized Message-receipt Processing

The equipment will ignore messages saved from before communications went down. If the equipment detects a communications error, it switches states to Not Communicating and sends S1, F13 to the host. Assuming this message arrives at the host intact, the host will reply with S1, F14. If the host already had another message in reply to something before communications went down, the host may or may not send that other message, but the equipment will ignore it if sent.

What happens when the equipment receives one message correctly but there is an error in the second so it cannot transmit to the host? The equipment will execute the processes indicated by the first message. When the second message turns out to be impossible to transmit, the equipment goes through the procedure of establishing communications.

8.3.3 Control (Operator-driven)

Control capabilities can be driven from either the host or the equipment, so there must be a way to make clear what the equipment is doing. Since control capabilities can be driven from either the equipment or the host, the division of responsibility must be made clear. After the equipment has started up and communications have been established, the control state is set to reflect control by the operator or the host.

Under the basic capabilities, the control state is set through the user interface. There are basically three possible control states: Online Remote, Online Local, and Offline. Under GEM's basic capabilities, the control state is selected by the operator through the equipment's user interface (see E30 Application Note A6, "Examples of remote control applications.") The operator functions covered by the basic functions are as follows:

- Operator-driven switch to online (basic requirement)
- Operator-driven switch to offline (basic requirement)
- Operator-driven switch between local and remote (basic requirement)

The control state model in the basic requirements. Figure 34 shows a control state model that satisfies the basic requirements. Transition numbers correspond to the state transition numbers in E30, Section 3.3, The Control State Model.

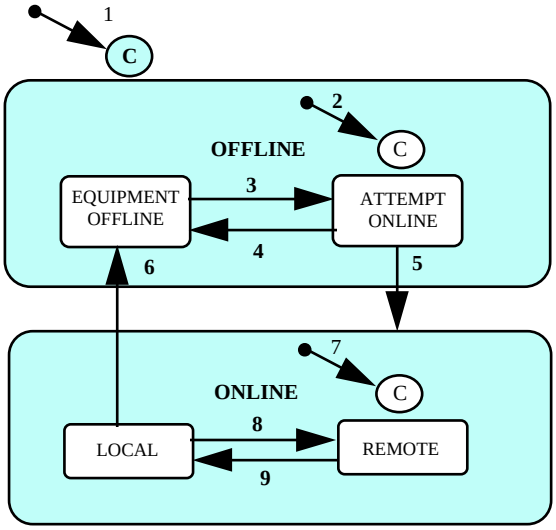

Figure 34 Control State Model That Satisfies Basic Requirements

8.3.4 Control (Operator-driven or Host-driven)

This section describes control state selection and operation from the host's perspective. The control substates will vary from one fab to another as appropriate to its equipment operations; what follows is an explanation of the control substates, using operational examples. Figure 35 shows the control statechart as per E30 for reference purposes. Transition numbers correspond to the state transition numbers in E30, Section 3.3, The Control State Model.

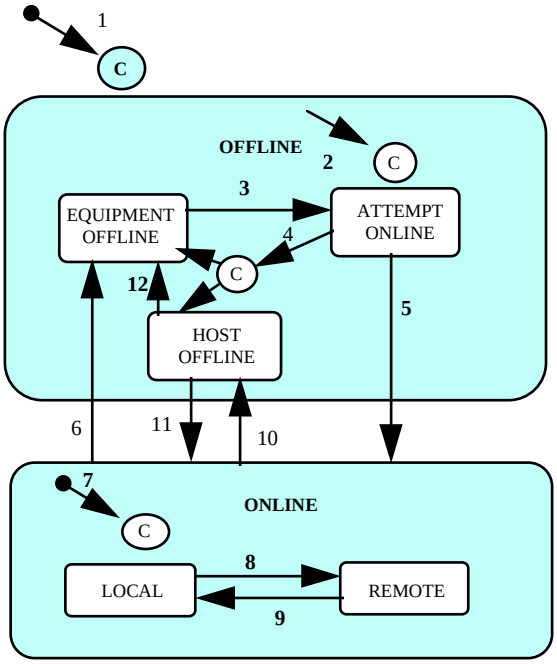

Figure 35 E30 Control State Model

Online Remote state: The host can direct almost all equipment operations remotely. This provides the functions needed if an automated transport system is in place. Even if there is no automated transport system, this is useful when you wish to control equipment operations through a host terminal.

Online Local state: The operator can execute almost all equipment operations through the user interface. This is useful if the duties of the fab's host are concentrated on production monitoring and data collection. It is also useful for specialized operations, where the parameters do not reside on the host, or when you want to monitor a trial run. Under online local control, the operator enters the process parameters into the equipment through the user interface and directly orders the equipment to begin processing. The host is scarcely involved in the processing, except for being able to track it.

Online local control is also applicable to measuring equipment. Almost all normal measuring work directly involves the operator. A measuring process program can be downloaded. This collects the operations involved in the measurement terminal, which in many cases obviates the need for a remote start order from the host. So it is possible to run measurement equipment under online local control.

Note that when making use of host functions, for example, plotting the result of each measurement on a totalizer chart, it may be advantageous to use online remote control: when there are errors that require the operator be notified, these can be handled through a host terminal.

Equipment Offline state: Use the offline state for equipment operations or maintenance that involve absolutely no communications with the host. Placing the equipment in the offline state is one way to cut down dramatically on host-equipment message traffic.

Automated operations should be avoided at all costs in the offline state.

It is also possible to perform equipment maintenance while online, most likely in the online local state. This allows the host to constantly monitor the equipment, tracking operating information and the like. If in the online local state, the host will not issue any operation commands to the equipment, so there is little risk of receiving a mistaken command from the host. Once a piece of equipment has been placed in service on a production line and is visible to the host, it is best to keep it online.

Host Offline state: For one reason or another, the host may temporarily make a transition to a state that does not permit it to requests to switch online from the equipment. In this state, the equipment will wait until it receives a request to switch online from the host, or is switched online by the operator through the user interface.

The control state is set through the equipment's user interface. Ordinarily, the operator switches the equipment's control state through the user interface (S1, F1/F2). Regardless of the state the equipment is already in, the state selected through the user interface takes priority.

Switching to an online state requires the host's permission; switching offline does not. In order for the equipment to switch to an online state, it requires permission from the host (S1, F2), but it can switch offline on its own, without the host's permission.

Note that the default control state to be entered at equipment startup can be set to online local or online remote. If this is the case, the equipment will go into the online state without the host's permission. With this procedure, even if the equipment does not notify the host by sending an online event report because of a communications error, because the equipment is in an online state, there may be a conflict.

If the host is offline, the equipment will wait for the host to request a switch to online. Although the control state is (switch online) set through the equipment's user interface, the host may not be able to respond due to unforeseen circumstances. If the equipment is set for host-driven control, and this problem makes it difficult to set the equipment to operator-driven control, then wait for the host to recover. Once things on the host end are back in order, the host will send a request to switch online (S1, F17/F18). At this time, if the equipment was in the Host Offline state, it will switch online. Following is a sample message sequence.

	Host	**Equipment**	
		< S1 F1	Request online
Online request denied	S1, F10 >		transition to Host Offline state
			(no event report issued)
Online operations enabled state			
Request online	S1, F17 >		
		< S1, F18	Acknowledge switch online
		< S6, F11	Online event report
Event confirmed	S6, F12 >		

This allows the host to be started up after the equipment without requiring that an operator go around and manually set each piece of equipment to the Online state. The Host Offline state is useful if the host drops out of online operations for some reason, or if there is a need to take the

equipment temporarily offline. In these instances, the host sends a request to switch offline (S1, F15/F16) to the equipment. Note that the Host Offline state does not really mean that the host really is offline, but that the equipment should treat the host as if it were offline.

Another possible operating procedure might involve the equipment trying to establish communications immediately after startup, but with the host going into an offline state after this. The equipment would then wait for the host to request a switch online. If you have set up the messaging sequence so that once communications have been established, the host will always send S1, F17, then the operator will not need to manually switch online.

Can the equipment run in the Host Offline state? GEM does not specify this. However, once operations have commenced in a certain operating state, the equipment may deny the host's request to switch online. In the Host Offline state, it may not be possible for the host to know what the equipment's operating state is, so if you permit the equipment to begin operations in the Host Offline state, it is possible that using only the SECS functions still available will be insufficient for ascertaining the timing of a switch to an online state.

Differences between Online Local and Online Remote:

Online Remote: All command operations originate with the host. All aspects of the equipment's operation can be controlled remotely without direct operator intervention, but the operator can change parameters and other settings through the equipment's user interface when needed. This state is best used when conveyances are automated. The automated transport system should fit in with the equipment and process materials. Online local control may also be a possibility for entering commands directly in the equipment, without using the SECS interface.

Online Local: The operator enters parameters, start commands, and all other operations directly through the user interface. The host's duties are to monitor actions, but not to control the equipment. Starting operations remotely would not be possible, although downloading parameters, etc., would be possible.

Automated processing can run under either the Online Local or Online Remote state. Normal equipment operations can be executed in either of these states. These correspond to the two types of operational modes in the explanation of plant automation concepts, where Operational mode 1 would be Online Local and Operational mode 2 would be Online Remote. It may also be possible to combine these two modes.

Equipment operation in an offline state should be avoided as much as possible, as the host cannot ensure the accuracy of the parameter settings in the equipment or monitor the equipment's operations.

The difference between Online Local and Online Remote is in the command to begin operations. The basic difference between these two states is whether physical operations in the equipment, such as begin processing or stop processing, can be ordered remotely. Remote commands will only be accepted in the Online Remote state, not in Online Local. Apart from remote commands, all SECS messages should in principle be handled the same way regardless of state. Following are typical scenarios.

Online Remote

Host		Equipment	
Send command	S2, F41 >		
		< S2, F42	Acknowledge command
		< S6, F11	Report process start
Start acknowledged	S6, F12 >		

Online local

Host		Equipment	
		< S6, F11	Report process start input by operator-initiated command
Input acknowledged	S6, F12 >		
		< S6, F11	Report process start
Start acknowledged	S6, F12 >		

There can be restrictions placed on what operations can be ordered through the equipment's user interface when in Online Local. It may also happen that the order of operations changes due to changes in operations years after operations have commenced, so these restrictions can be a handy way of dealing with variable settings. Restricting the input of settings could be on a command-by-command basis, or simply lock out the entire screen.

Assuming that the equipment supports remote execution of all functions, the user interface could be made almost completely superfluous. The switches available to the operator could be restricted to Emergency Stop, Stop, Abort, and other emergency-related functions.

Operations in Online Local are no different from operations in Offline. Since all operations in Online Local are operator-driven, they are handled in the same manner as operations in Offline. The same operations that can executed by the operator in Offline must be executed in the same way in Online Local. There might be no visible differences between operations in the two states.

Indicate when the ranges of operations available in online and offline states are not identical. There should be a clear explanation in the specifications of operations that are available through the user interface Offline, but restricted in an online state (either local or remote). While it is reasonable to expect some procedural differences between online and offline operations, it is best to keep things as similar as possible, except for functions relating specifically to remote control.

It is best to make all normal operations available while online. Automated equipment will normally be run in an online state. Even during maintenance, or for operations run through the user interface, it is best to run all operations in an online state. Everyday operations, including recoveries from minor troubles, can cause problems if there are constant switches between online and offline states.

Switching offline can cause serious errors, either for the host or the equipment. Switch offline only when it is impossible to continue operating online.

Online Local is appropriate for use in recovery mode from equipment problems. As previously explained, when equipment problems arise, sometimes the recovery procedure requires dropping Offline, but since the procedure for switching between online and offline states is complex and time-consuming, it is best to avoid switching too frequently. If commands from the host cannot be accepted, switch the control state to Online Local. This should allow necessary recovery actions in recovery and at the same time generate event reports.

Simplifying remote/local/offline operations. The relationship between control states and equipment operations may appear complex, but was designed this way to ensure the spread of GEM. Table 7 shows a simplified relationship between equipment operations and control states. Note that this is only one example, and not universally definitive.

Table 7 Equipment operations under each control state

	Remote	Local	Offline
Download program	Host-driven	Equipment-driven	NA
Select program	Host	Operator	Operator
Revise program	Host	Operator	Operator
Start program	Host	Operator	Operator
Detect process complete/ collect process data	Host	Host	Operator
User interface operations	Not permitted without state change, for emergencies	No restrictions	No restrictions

Operator switching to Host Offline state. GEM does not restrict the operator from switching to the Host Offline state through the equipment. If needed (for testing purposes), this function can be implemented.

The default control state can be set. The set control state is stored in nonvolatile memory. At startup, the equipment will automatically begin the actions needed to transition to the state specified in the startup preferences. A screen for setting default values might work as shown in Figure 36 below.

Once a switch to online mode has been completed, the equipment checks the Remote/Local switch setting for which particular state to enter.

A default setting is used to indicate the preferred state after a failure to switch to online mode, Host Offline or Equipment Offline.

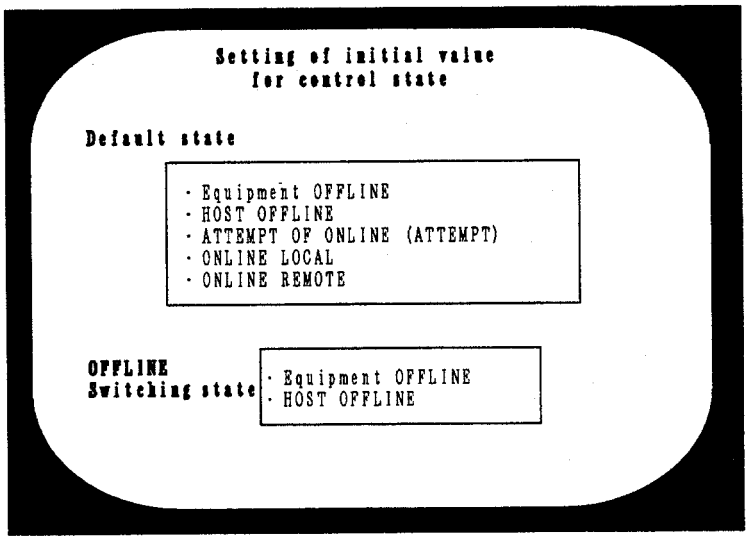

Figure 36 Sample of Setting of Default Control State

Equipment Offline: Immediate transition to Equipment Offline state, reports "Equipment Offline" (S6, F11) event to the host.

Host Offline: Immediate transition to Host Offline state, reports "Host Offline" (S6, F17) event to the host. Waits for request to set online (S1, F17) from the host or equipment online switch from the operator.

Attempt Online: Sends a switch online request (S1, F1) to the host. If permission to switch online (S1, F2) is received, the equipment goes into the "attempt online successful" state and then transitions to the preset online state, Online Local or Online Remote. If the attempt is denied by the host (S1, F0), or if a communications timeout is detected, the equipment goes into the "attempt online failed" state, and then transitions to the preset offline state, Equipment Offline or Host Offline. After the transition, the equipment sends an event report.

Online Local: Immediate transition to Online Local, reports "Online Local" to the host. If a communication error occurs and the report is not properly transmitted to the host, the equipment will still make the transition to the Online Local state, so strictly speaking, it is not necessary to inform the host of the transition to the Online Local state.

Online Remote: Immediate transition to Online Remote, reports "Online Remote" to the host. If a communication error occurs and the report is not properly transmitted to the host, the equipment will still make the transition to the Online Remote state, so strictly speaking, it is not necessary to inform the host of the transition to the Online Remote state.

Control state set-up screen. The equipment's control state can be changed through the screen. Either the local or remote online state should work properly, assuming the communications state is Communicating and Enabled. If you arrange both control state and communications state on the screen together, as in the illustration below, it will be easier to tell if the control state is in effect when there is a communications error. This display includes an alarm window to inform the operator whether an attempt to set the control state has been completed or failed.

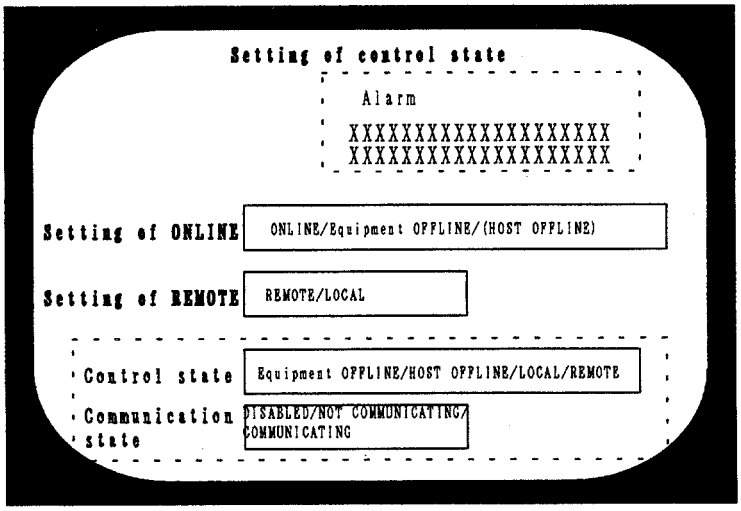

Figure 37 Sample of Control State Setup Screen

Online settings: Selecting an online state when in an offline state sends a Switch Online request (S1, F1) to the host. Upon receiving permission to switch online (S1, F2), the equipment transitions into either Online Local or Online Remote, whichever is specified in the initial control state set-up screen as the state for a successful switch online.

If the equipment does not receive permission to go online from the host (S1, F0) or detects a timeout, it transitions into either Host Offline or Equipment Offline, whichever is specified in the initial control state set-up screen as the state for a failed switch online.

If offline is the selected state, then the equipment goes directly offline and sends the host an event report.

Remote/Local settings: You can switch between Local and Remote when online, but not when offline.

Problems in setting the control state default that follows communications establishment. Setting the following defaults can create a state conflict between the host and equipment at equipment startup time.

1. Communications state default: Not communicating
   Control state default: Online (remote or local)
2. Communications state default: Communicating, but with failure to establish communications
3. Control state default: Establish Online or Online (remote or local)

The online states are meaningless unless the communications state is Communicating and communications have been established. When starting up equipment with a default control state of online, pay extra attention to the communications state.

After making a transition offline, what happens to subsequent messages involved in open transactions? GEM assigns the processing of this to the equipment. It doesn't matter whether all transactions have been completed before making the transition offline, or if it sends an Sx, F0

message after making the transition offline. Going offline requires that all transactions be closed out, but subsequent message are not replied to.

If subsequent messages are received after going offline, must the equipment process them as usual? GEM does not specify this one way or the other, but since it doesn't relate to the host, this can be decided as appropriate to the equipment.

A control state transition generates an event report. Immediately after a control state transition, the equipment sends an event report to the host with the new control state and the previous one.

Messages that can be exchanged while offline.

- Messages that can be sent: S1, F1; S1, F13; S1, F14; S9, Fx
- Messages that can be received: S1, F2; S1, F13; S1, F14; S1, F17 (only received when Host Offline). Receipt of any other message results in an abort and Sx, F0 reply to the host.

Although E30 states that the appropriate reply message should be sent in response to S9, Fx messages when offline, a particular communications software setup may make this difficult. As explained in the section on establishing communications, there should be a format check on received messages immediately after receipt. If the equipment detects an undefined device or undefined stream error, the host should be notified before passing the message along to the control capabilities for processing. Figure 38 shows a generalized message processing flowchart. This is a simplified flowchart, and other procedures may also work, depending on the software design.

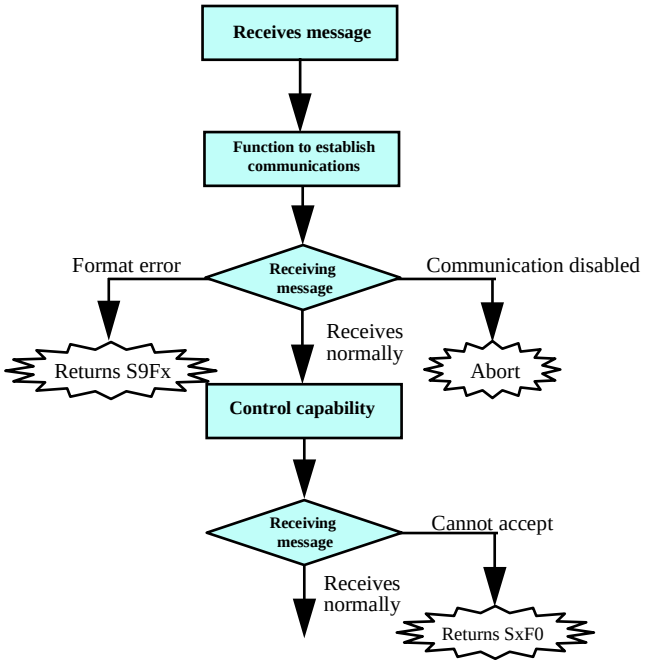

Figure 38 Received Message Processing

Switching offline during a multiblock transmission. If the equipment is switched offline while it is sending a multiblock transmission to the host, the transmission is interrupted, and an offline

event report is sent to the host. If the transmission was not complete, the message is lost. If the loss of this message will cause some sort of problem, it is all right to delay switching offline until the message is complete. GEM does not address this issue.

Control state selection. E5-95 defines two classes of control state variables, listed below. The reason for this is probably that GEM was defined afterwards, but the previously-defined format codes were changed and redefined over time. Right now only the earlier set is used.

Format: 10	**Format: 51**
1: Offline/equipment offline	0: Local
2: Offline/attempting to switch online	1: Remote
3: Offline/host offline	2-63: Reserved
4: Online/local	
5: Online/remote	
6-63: Reserved	

8.3.5 Event Reporting

Event notification is a necessary function for equipment monitoring. An “event” is any new condition appearing within the equipment. An event notification is the way the equipment informs the host of detectable events. Most actions within the equipment are independent of the host and asynchronous. Because of this asynchronous nature, it would be difficult for the host to query the equipment as to every single event, so the equipment has the responsibility for detecting important events and notifying the host of them. This allows the host to monitor the equipment’s status in real time, without querying the equipment.

If the event notification includes a variable report, all the equipment’s internal data is collected, such as equipment state, process data, and equipment operating statistics. Material process management is also made possible by monitoring event-start and event-complete events.

A noteworthy feature of the event report is flexibility in changing report format settings. The report format of an event report must be open to revision, a requirement that benefits both the equipment supplier and the host. The equipment supplier has the flexibility to respond to the differing demands of equipment and host, and the host can handle changes in the contents of event reports after new equipment is brought online. Much of the value of the event report is in the editing functions for the report format: without them, even if only event report messages (S6, F11/F12) were supported, there would be no GEM compliance.

Event reports as substitutes for SECS messages. Event reports can act as substitutes for the messages below. The major difference between them is that the contents of an event report can be changed. These sorts of reports to the host containing variable data take advantage of event reporting functions.

1. Transmit formatted variable data (S6, F19/F10)
2. Transmit unformatted variable data (S6, F3/F4)
3. Request data transmission (S6, F7/F8)

As a substitute for these messages in GEM, you can use Event Report Send (S6, F11/F12), Event Report Request (S6, F15/F16), and Request Spooled Data (S6, F19/F20).

Event notification under the basic requirements.

Following are the functions provided under the basic requirements:

- Notify host of event (S6, F11/F12).
- Send event report data in response to request from host (S6, F15/F16).
- Events set to enabled/disabled through user interface.
- Initial settings for event reports provided by supplier, changes made through user interface.

Data provided in the event reports includes:

- State variable (SV)
- Equipment constant values (ECV)
- Unformatted data values (DVVAL)

More specifically:

- Changes in the control state
- Changes in equipment state
- Changes in Material input/output port state
- Results of remotely-executed commands
- Operator command inputs
- Changes in recipe settings entered through user interface
- Alarm set/reset

GEM does not use S6, F13 (event report with comment) including variable IDs.

When the host sends the event report request (S6, F15/F16) to request data, there may be undefined elements, depending on the data variables (DVVAL). These variables only have meaningful values when an event has been generated.

Changing event report settings through the user interface. The event report can follow the equipment supplier's original settings in response to a host request, or can be freely altered through the equipment's user interface. For clarity's sake, in this document we will concentrate on sample settings using the user interface.

The screen shown in the example here would in reality be very hard to use; preferable to this would be one that updates the settings depending on the message.

8.3.5.1 Variable ID list display

Initially, the variables that control whether event reports are sent will be displayed in a list, as shown in Figure 39 below.

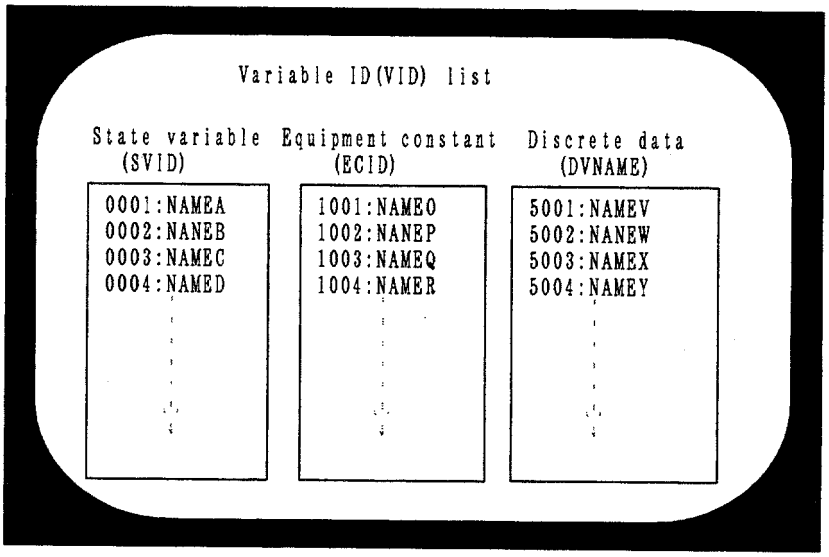

Figure 39 Sample of Variable ID List Display

This display shows three variable IDs (VID):

- SVID (state variable)
- DVNAME (unformatted data variable)
- ECID (equipment constant value)

These VIDs are all specific and cannot be overlapped. Since they depend on the equipment's capabilities, the user cannot freely change them or reset them.

The purpose of this screen is to show the user the VIDs in a way that links them to their event reports, so this screen will only show VIDs as set by the equipment supplier, and provides no function for modifying the VIDs themselves. While it is OK to simply provide the VIDs in a document, doing so will require that you ensure there is a strict one-to-one correspondence between the document and the actual software. GEM uses unencoded integers for these VIDs, so this display shows the names of all the variables as comments (such as NAMEA).

8.3.5.2 Report ID • Variable Definition

Below is a screen for recording report IDs. Since the variables are defined by the report where they appear, both the report ID and the linked variable IDs are shown here. The left side of the screen has the report ID box, where new reports are entered and recorded reports are called up. The right side has the linked VIDs. One report ID can be linked to several VIDs. Since the report ID is just an unformatted integer, the report ID box allows for comments.

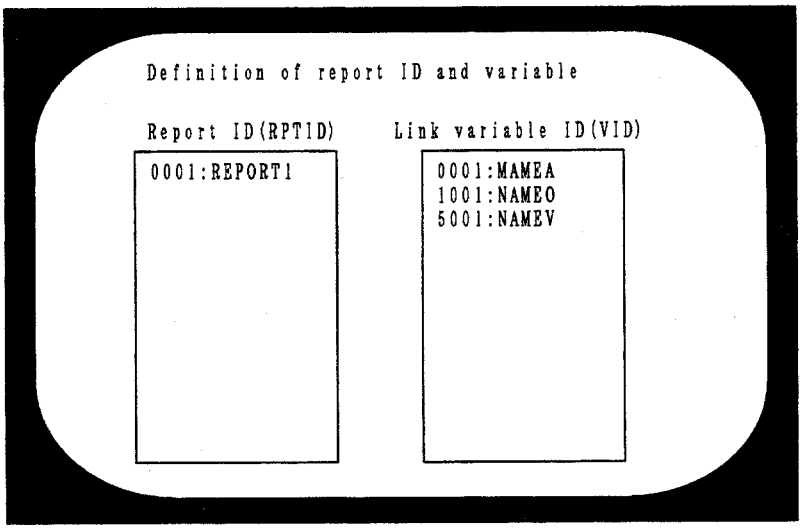

Figure 40 Sample of Screen for Recording Report IDs

8.3.5.3 Event Report Link

The following screen lets the user control whether a report should be sent to the host for each supplier-defined collection event ID, and also allows the user to set a linked report ID. This screen allows many-to-many linking relationships between the event reports and the report IDs. It also allows the send/do not send setting to be applied to multiple event IDs simultaneously. Since the event ID format is just an unformatted integer, this screen allows for comments.

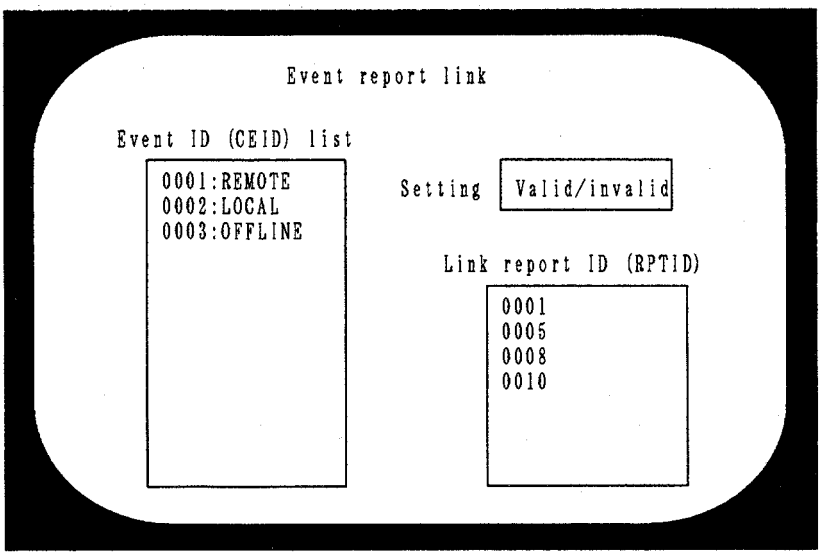

Figure 41 Sample of Report Control Screen

How to use the Data ID in the S6, F11 message. The Data ID can be set at the equipment end to allow the host to discriminate linked multiblock messages. In short, the Data IDs of a multiblock data transmit request/permission (S6, F5/F6) and the subsequent event report (S6,

F11/F12) are set to the same value. Note that this function may not be necessary with open transactions in multiblock messages using the same stream/function code.

Individual Report Requests may be supported using the same function as for Selected Equipment Status Requests or Equipment Constant Requests. Depending on the individual report request (S6, F19/F20), the host may request variables pertaining to the selected report ID from the equipment. The state variable ordinarily would be bundled with a selected equipment state request (S1, F3/F4), and the equipment constants would be bundled with the equipment constant request (S2, F13/F14). As such, these functions should be supported, depending on how the reports are defined.

Reporting multiple events in one event report. Reporting every single event as it happens could overload the communications system, so it will often be desirable to combine several events in one event report. For instance, when a certain event has been generated, rather than reporting it to the host, a notation will be added to the report contents, and only when a separate trigger goes off will the combined report be sent. This arrangement could result in one event report containing multiple instances of unformatted data variables containing the same report ID. When it is important to discriminate between different reports containing the same ID, the variable data could be time-stamped or tagged with a special sequential number. If processing results are being saved for each individual wafer, an aggregate report might be sent to the host after all wafers have finished processing.

An example of event reporting for collecting process data. The user can select the increments in which process data reports are transmitted, i.e, after every wafer, every lot, every cassette, or no report transmitted at all. It is rather laborious to define this for every event when setting up the host/equipment communications specification: using the event-reporting capabilities here makes collecting these data much easier. The equipment will have assigned collection event IDs to all these situations, and there will also be variable IDs already assigned to each of the report data. Using the variable report setting capability, the host can acquire these reports in the required format at any of these intervals. These variable IDs also come in handy when requesting average processing results for multiple processes or deviations from average.

Hidden variable data reporting. Depending on the variables combined with a collection event, variables meant to be hidden in an event report could be reported. This might happen, for example, with execution recipe variables prior to set-up, or (using the previous example), if average process results are requested for each process completed on each wafer before processing of the entire lot is complete. If variable data is reported to the host that is meant to be hidden, the equipment can set the variable data item length to zero. This may require some advance work setting up the variables to be hidden. It is impossible for the equipment to predict the event report format setting that the host will use, so pay special attention by immediately clearing hidden variables (or some other preventative measure) so that these variables do not wind up getting reported to the host.

8.3.6 Variable Event Data Collection

Advantages to dynamic event data collection. Dynamic event data collection means letting the host execute setting changes shown on the event report screen (as explained in the event notification section) over the communications lines. This has the following advantages:

- The host can set or alter report formats dynamically, as needed;

- It makes it impossible to set report formats through the user interface that the host can't deal with.
- The event report set-up screen on the equipment end is not needed.

This means that the host must set up the report format correctly; if it sends a report request specifying an incorrect format, it will get a setting-error in return. But for that matter, it is possible to enter incorrect formats through the user interface as well.

Timing setting changes in dynamic event reports. Setting changes can be made in dynamic event reports when the equipment is in an online state. Note that the event report settings are changed while the equipment is in the midst of an operation, it may cause an error, depending on how the equipment's software was written. If, for example, there is a conflict over access to a file defined in the event report, the equipment will not know when to call the newly defined file from its nonvolatile storage into main memory.

To avoid problems like this, the dynamic event report settings should be revised only after first temporarily disabling all event reports. Following is a sample message exchange illustrating this. In this example, the event report being changed can be called from main memory once it is re-enabled.

	Host	**Equipment**	
Disable collection events	S2, F37 >		
		< S2, F38	Confirm disable
Transmit report definition	S2, F33 >		
		< S32, F34	Acknowledge report definition
Transmit link event report	S2, F35 >		
		< S2, F36	Acknowledge link event report
Enable collection events	S2, F37 >		
		< S2, F38	Confirm enable

Since events have been disabled, the equipment will not generate any event reports while the report format is being reset. Regardless of the equipment's state, event reporting will be unnecessary if the collection event report format is being changed.

Design principles for event report capabilities. Event reports are meant to give the host flexibility in the type, size, frequency, etc., of the data it collects from the equipment, so the equipment should be designed from the start to meet future demands, and in any case, should allow all variables to be listed in event reports and also easily permit additions.

Also, to deal with the differing demands of different users, the equipment should be able to respond flexibly without changing the software on every piece of equipment. The same event-reporting software can be used on every piece of equipment. Figure 42 shows the event-report creation mechanism.

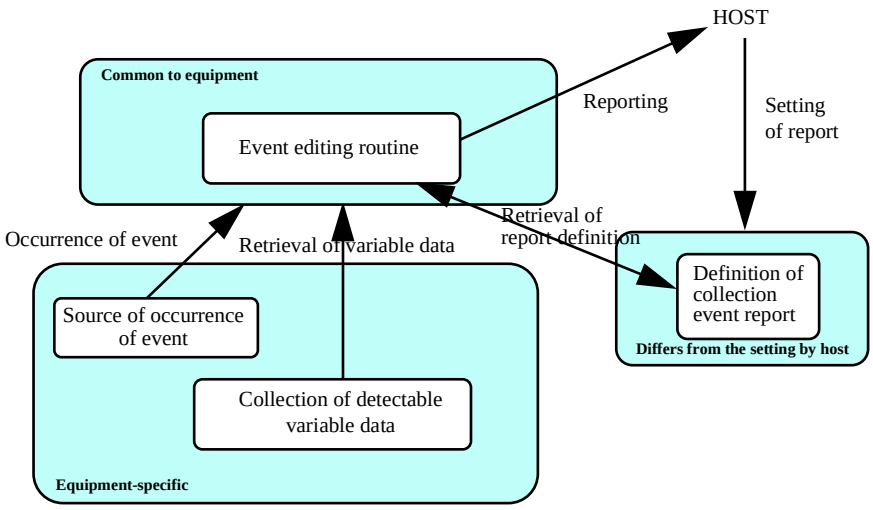

Figure 42 Event-report Creation Mechanism

8.4 Explanation of Additional Capabilities

8.4.1 Trace Data Collection

Use trace data collection to allow the host to closely monitor material processes in the equipment. Trace data collection allows the host to closely monitor material processes in the equipment in real time. Implementing a trace over a SECS-I connection limits the number of samples that can be reported to the host. To keep the message transaction load down, trace data transmissions (S6, F1) should be restricted to simple blocks. This is not strictly necessary, but reported state variables are permitted by variable list structures, so be careful that variable list structures do not exceed one block (244 bytes).

Trace data collection variables

TRID: Trace ID. The TRID is repeated in each S6, F1. This ID allows the host to establish a relationship between the trace data sent by the equipment and the Trace Initialize command (S2, F23).

DSPER: Used for the data sampling period, to define the frequency for the sample log. Format is “hhmmss” using 6 bytes of plain ASCII.

TOTSMP: Total number of samples collected. When this number of sample have been reported, the trace ends. TOTSMP should be greater than REPGSZ.

REPGSZ: Reported group size. Ordinarily, this should be 1. The RPGSZ corresponds to the number of trace data reported together. Sometimes you will wish improve transmission performance with large quantities of data; a higher number causes several traces to be grouped under one S6, F1 message.

SMPLN: The sample number—a serial number assigned to a collected sample. The first sampled data that is logged is numbered 1, with subsequent logged data sample numbers incremented by 1. If REPGSZ is greater than 1, meaning that multiple samples are being reported under one S6, F1 message, SMPLN will be equal to the number of the last sample reported in the message.

STIME: Sample time. Represents the data and time when the state variable was read from the equipment. Format is “YYMMDDhhmmss” using plain ASCII. If REPGSZ is greater than 1, meaning that multiple samples are being reported under one S6, F1 message, STIME will represent when the last sample was taken.

An example of trace data collection: In this example, there are three state variables, the sampling period is thirty seconds, and the reported sample group size is two.

```
S2, F23 format
     <L [5]
          <U2 1>    TRID #1
          <A ‘000030’>   DSPER 30 seconds
          <U2 120>  TOTSMP is 120
          <U2 2>    REPGSZ is 2
          <L   List of SVIDs
               <U2 1>    Request SVID #1
               <U2 2>    Request SVID #2
               <U2 3>    Request SVID #3
          >
     >.
```

The trace data collection process runs like this:

10:00:00 Initiate trace data by S2, F23

10:00:30 Equipment collects first sample. The equipment records the values of SVID 1, 2, and 3. Since REPGSZ is 2, the equipment does not send S6, F1

10:01:00 Equipment collects second sample. The equipment again records the values of SVID 1, 2, and 3. This is sample #2. Since REPGSZ is 2, the equipment does send S6, F1. This message is carrying six state variables, three each from samples #1 and #2. Since this is the last sample involved in this transmission, the equipment adds SMPLN and STIME, with SMPLN being 2 and STIME being *YYMMDD*100100.

10:01:30 Equipment collects third sample.

10:02:00 Equipment collects fourth sample and transmits next S6, F1 message. SMPLN is 4 and STIME is *YYMMDD*100200. Trace execution continues.

11:00:00 Equipment collects 120th trace and transmits final S6, F1 message. Since TOTSMP is 120, the trace is now complete.

8.4.2 State Data Collection

State data collection lets you ensure that the equipment’s actual state and the host’s perception of the equipment’s state agree. Before sending an important command message, it is helpful for the host to check the equipment’s state to find out whether the command will go

through before it actually happens. You can also periodically check the equipment's state and log it or display it on a host terminal. State data collection is necessary for keeping the host updated as to the equipment's state. States that the host would be concerned with are the processing state, the control state, the material port state, and the various process parameter values.

8.4.3 Alarm Management

The alarms needed by the host are decided after delivery of the equipment. The equipment supplier will have no way of knowing what alarms will be needed before delivering the equipment to the customer. Rather, the equipment is delivered in a manner allowing it to deal with all reportable alarms.

After the user has placed the equipment in service in the plant, he selects the needed alarms. It is also possible that the alarms that the user wishes to detect will change over time, so by enabling and disabling specific alarms, the user can have the exact assortment that meets his needs. Alarm management provides the mechanism by which the user has access to the right alarms at the right times.

Events include alarms. Any event that poses a danger to people, equipment, or materials in process should be treated as an alarm: "event" has a broader meaning than "alarm," and every alarm accompanies the event that causes it. See Figure 43.

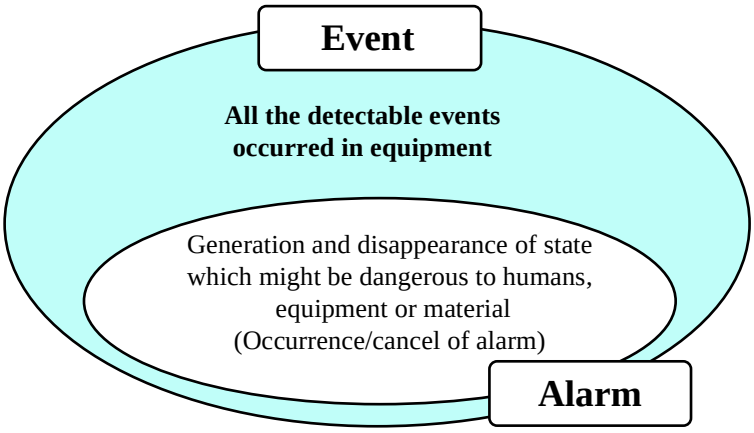

Figure 43 Relationship Between Events and Alarms

Alarms have two states, Set and Clear, both of which are always available. The transition between the two is reported as an event. An alarm notification and event report are sent to the host when there is an alarm state transition. In short, for each alarm ID (ALID), there should be two event IDs (CEID): one when the alarm is set, and another when it is cleared. Alarm messages are sent before event reports in these instances. GEM's alarm management scheme comprises the following three types of messages:

- Alarm messages (S5, F1/F2): an alarm being set or cleared
- Alarm-set events
- Alarm-cleared events

If event reports are not necessary, you can disable event reporting to correspond with the alarm set/clear events.

Linking alarms to events allows you to collect variable data needed for diagnosing the cause of the alarm in a timely manner. Linking the alarm set/clear to each event may seem complex, but if you use the event reporting capability, it makes it possible to quickly collect data that will be useful in diagnosing the cause of the alarm.

The host can enable or disable alarms. Although you probably won't want to be disabling and enabling the alarm too often, this is the sort of thing that can't only be controlled through the equipment. Managing the alarms reported by the equipment is not so much a task for the shop-floor operator as for the host. Some equipment may report a prodigious number of alarms. Ordinarily, the host decides which of these alarms to act on and which to ignore. The host uses the enable/disable alarm send message (S5, F3) to manage the alarms directly. The GEM philosophy indicates that you should give the host as much control as possible.

Distinguishing alarms from warnings. Some equipment may issue warning messages, which are less severe in nature than alarms, and do not set or clear the alarm state. Since these are not defined in GEM, they are considered extended functions. GEM does not use alarm codes (ALCD). Warnings and alarms can be distinguished by assigning different ranges of alarm IDs (ALID). The alarm ID can also be used to determine specific information about the alarm (point where generated, nature of alarm, etc.). Since the alarm text is 40 characters long, this can be used to convey an extended and clearer description of the alarm type.

Should the non-arrival of the next processing lot be an event or an alarm? In some circumstances, this may be interpreted differently by the equipment and host. Suppose that a material process at a certain piece of equipment is complete, and the equipment is ready to begin processing the next lot. The equipment will be in the "waiting for material" state. This state doesn't really have anything to do with whether the equipment is really ready to begin processing, but the fact that the material hasn't arrived might be a sign to the host that an alarm is in order. Suppose further that, regardless of whether there is an automated material transport system, any equipment stoppage due to a wait for materials is interpreted as an abnormality. This state may be interpreted as an alarm state by the host, which will send a "waiting for material alarm" to the equipment.

From the equipment's perspective, though, being in the "waiting for materials" state is simply idling. This shows how the equipment and host can interpret the state differently. GEM does not consider any situation not posing a danger to people, equipment, or material to be an alarm. However, the host is free to interpret any detected situation any way it wants. In this example and other cases, the host could interpret specific event reports as either warnings or alarms, thus containing differences of interpretations between itself and GEM.

8.4.4 Remote Control

Remote control commands can only be executed when the equipment is in the Online Remote state. As explained in the control-state section, the equipment will only accept remote control commands when in the Online Remote state.

GEM's remote-control commands may be substituted for previous remote-command send (S2, F21/F22) and request process start (S2, F27, F28). The previous Remote Command Send with structure: <RCMD> and Request Process Start with structure <LOC>, <PPID>, <MID> commands did not contain parameters sufficient for remotely ordering a process start. The Host

Command Send with structure <RCMD>, <CPNAME>, <CPVAL> provide the necessary flexibility by using these parameters:

- <RCMD>: orders actions like “start” and “stop”
- <CPNAME>: gives the parameter names of the process being started, such as “Port ID” or “Process Program ID.” Also gives the names of parameters needed in order to begin actions ordered of the equipment, such as specific setting IDs within a process program.
- <CPVAL>: Gives actual data values corresponding to <CPNAME>s such as the port ID.

The Host Command Transmit bundles together these disparate parameters to be transmitted all together.

Remote commands can be used to give permission to the equipment to begin an action. The use of remote commands is not limited to acting as the final trigger for a physical action by the equipment. After the equipment has received an Online Remote Start command from the host, the operator can order the process-start through the user interface; this would also work in the Online Remote state.

Remote command results are reported using event reports. After a command has been received and executed, the results of the execution are always reported to the host through an event report.

The handling of operator commands when in the Online Remote state can be decided by equipment settings. The operations that will remain executable when the equipment is in the Online Remote state can be set by the user. Depending on the production environment where the equipment is being used, limiting operator input may cause problems. Rather than setting or unsetting input restrictions at the user end, you can alter the equipment’s response to user requests, which will not require changes in the equipment software.

8.4.5 Process Program Management

Responding to host requests regarding process program management. While an automated system should require only minimal management by the host, it is necessary at certain times to ensure that the equipment has the correct process program and is executing it. What follows is a step-by-step explanation of the functions that make this possible. As used here, “process program” should be taken to mean the same thing as “recipe.”

The two functions needed as a bare minimum for process program management and remote control are:

- That the host know the name of the process programs resident in the equipment (S7, F19/F20)
- That the host be able to select a process program resident in the equipment (S2, F41/F42)

However, even being able to select a process program name and order the equipment to execute it is no guarantee that the host is really causing the program to be executed, so the following functions are also necessary:

- Uploading to the host what can actually be used in a process from the resident process programs (S7, F3/F4; S7, F5/F6)
- Of the process programs uploaded to the host, downloading and executing one when needed (as above)

This allows a process engineer to upload a verified program to the host, and then have the host download it to the equipment when needed and execute it.

If program creation and editing functions are available at the host end, the following functions will also be necessary:

- Downloading a formatted process program to the equipment, and then notifying the host whether this is an executable format and whether it contains values (S7, F23/F24; S7, F25/F26; S7, F27/F28)

This allows the host to confirm that the edited program is correct before execution actually begins. Also:

- Clearing process programs stored in the equipment not needed by the host (S7, F17/F18)

Clearing out unneeded programs reduces the risk of operator mistakes and makes room for new programs to be downloaded.

If the process program itself is too big, it may not be possible to upload or download it, or if each process must be downloaded separately, it may take too much time. So being able to authenticate the contents of the process program could be useful. The function allowing this is:

- A report to the host in the form of an event report showing whether the equipment's process program has been changed (S6, F11/F12).

If a process engineer or operator revises the process program through the equipment interface, that fact will be reported to the host. This keeps the host up to date on when changes have been made to the process programs. However, if the equipment is not always online, this function will not work. If the process program is revised while the equipment is offline, some sort of special process is needed for reporting the fact to the host once the equipment is switched online.

It is important to distinguish between the process program being executed and stored process programs. When selecting a program to be executed, either through the host or the user interface, that program will be transferred from the program-storage area to the program-execution area. Doing this blocks the program to be executed from being edited or transmitted to the host or another piece of equipment.

There are times when you wish to save a process program download as a stored program, and times when you wish to just use it as the program to be executed. Sometimes the equipment requests a process program download from the host that it is only going to use one time, and other times it will be using it repeatedly for the same process. In the former case, the program can simply be handled as the program to be executed. In the latter case, the program should be written to nonvolatile memory. GEM only defines the latter; the former must be handled as a GEM capability extension.

If the equipment does not have permission from the host to write the program to be executed, it can reserve a specific process program name (area) for the received process program name (area), and use this area for every process downloaded from the host.

When a new program is downloaded, the old one gets overwritten, so different process programs will be referred to under the same program name. This sort of arrangement is in keeping with GEM's specifications.

Revising the process program to be executed. After the equipment has downloaded the process program, a frequently-requested function is the modification of some parameters followed by process execution. Either the host or the operator would modify these parameters and then begin the processing. In this case, the revision results would not be saved and the revision results would not be reflected in the original program.

Process execution and program upload/download are separate functions. While process execution and program download may be linked in certain scenarios, the two may in other cases be handled separately.

There need to be fine-grained state variables for the process program creation, revision, and deletion event reports. Creating, revising, or deleting a process program through the user interface will result in a process program change report being sent to the host. This event report will include the process program name (PPChangeName) and status (PPChangeStatus [1=create; 2=revise; 3=delete]). Some of the process program's parameters may also have been changed, but so far there is no way of knowing which were changed or in what way they were changed. This is why we need more specific state variables.

Again, if the equipment is offline while the recipe is being edited, the changes will not be reported to the host. Also, if it is not online, the utility of this function is weakened, since there is no way to exercise control over forbidden recipe edits, etc.

Restricting and reporting process program operations. If it is possible for anyone to alter the process programs settings by using the equipment user interface, then a security problem exists. For this reason, it is important to restrict access with passwords to control levels of access. By sending an event report with an operator's ID, the host can track the revision history for process programs more accurately.

Process programs and processing results. Depending on the equipment, the settings in the downloaded process program and the settings actually used by the equipment may differ slightly. The host will want to know what settings were actually used, so it may be important to report both the downloaded settings and the settings actually used. This can be supported using the event report function.

The format of the process program is not defined. There are as many possible process program formats is there are pieces of equipment, so GEM does not define any format for process programs.

Editing process programs more efficiently. Process programs, or recipes, are complex things that demand a lot of attention and time. This is not the sort of work that should be done at the equipment user interface, in the clean room: it interferes with the fab's real work, and the distractions make input errors quite likely.

Furthermore, when you have several pieces of identical equipment, you might want to copy or share the process program among them. For all these reasons, it is better to do process programming on a computer dedicated to the task than at the equipment. The host should be able to emulate the editing functions for a complex recipe. Equipment suppliers could supply dedicated recipe-editing systems. Such a system should:

- Offer the same recipe editing functions as the equipment (use the same software)
- Support the same SECS-II message as the equipment. Bidirectional recipe transfer, recipe name lists, and recipe deletion functions should all be supported
- Manage specific recipes for multiple pieces of equipment (if the recipe editor is for only one piece of equipment, this is not needed)
- Distinguish recipes shared by several pieces of equipment from recipes used on single pieces of equipment, provide efficient editing tools (such as copy and pasting)
- Make it so that the host has flexible management of recipe names and versions

Figure 44 shows two possible configurations. Others would be possible: on a network, there could be a recipe server, or there could be multiple pieces of equipment and multiple terminals.

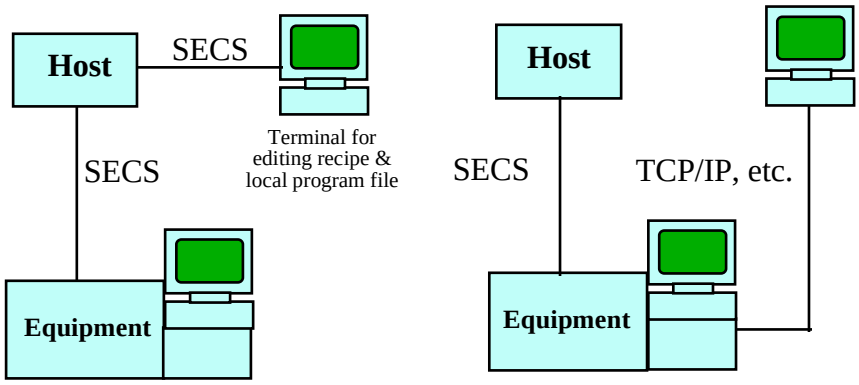

Figure 44 Recipe Editing Systems

Parallel recipe downloading and processing. For every process, the recipe must be downloaded from the host and the process-start order executed. Large recipes can take a relatively long time to download, so downloading every recipe can become impractical. As explained before, recipe download and process execution can be handled as completely independent functions, so it is possible to develop functions allowing the two to proceed simultaneously, in parallel. By beginning to download the recipe for the next process while the current one is still executing, a big enough headstart can be made so that even long recipes of tens of kilobytes will not noticeably delay the start of the next process.

8.4.6 Material Transfer

Material transfer capability requests come from the host. Managing an automated transport system requires that the host monitor the equipment's material state in real time. The state of the material input/output ports is an especially important part of material handling. Material transport issues have been standardized in E32-1994 (MMM), and are not discussed in this document.

It may be necessary to extend equipment operation modes to accommodate automated transport. In Online Remote mode, material transport can be executed automatically using robots etc., or manually by the operator. If an automated transport system has a problem that requires operator intervention, though, some way of discriminating between automated material handling and operator material handling may be needed. An "automated transport mode" could be added to the equipment's operating modes. This could be switched by either the host or the operator, distinguishing between automated transport and manual set-up. The state model for a transport mode could follow the control state model.

8.4.7 Equipment Terminal Services

Terminal services allow information about equipment operations to be transmitted effectively between the host and equipment. The equipment can send text data at its option to the equipment (S10, F3/F4), which is displayed on the equipment user interface (equipment terminal). Similarly, the operator can enter textual information using the equipment user interface and transmit this to the host at any time (S10, F1/F2).

The biggest problem in implementing this is designing the equipment's display area. The equipment needs to use this screen to convey information received by the host to the operator in a readily understood fashion. There also needs to be a way for the operator to enter information to respond to the host. If the user interface uses a windowing interface, host messages could be displayed in a window at the top of the screen, and the operator-input window could send an event report when dismissed (S6, F11/F12). It probably isn't necessary to keep a text-entry window (for the operator to send information to the host) permanently open—it could be invoked through a menu command.

Figure 45 shows a mockup of how this might look. When the operator needs to enter text in response to information sent by the host, the text-display and text-entry windows could be combined. It does not matter whether the equipment can keep multiple messages from the host on-screen; GEM does not demand it.

What to do if there is no persistent host-text display. Depending on how the equipment is set up, the terminal-services window might not always be visible. If this is the case, the equipment should provide an alarm or some other alert that goes off when a message is received from the host, so that the operator will be prompted to invoke the terminal-services display.

Under GEM, terminal service functions are one-way between the host and the operator; GEM does not support interactive communications. The host can send text for the operator to read to the equipment at any time (S10, F3). Since the equipment will only store the most recent message, any new text that comes from the host will overwrite the current message.

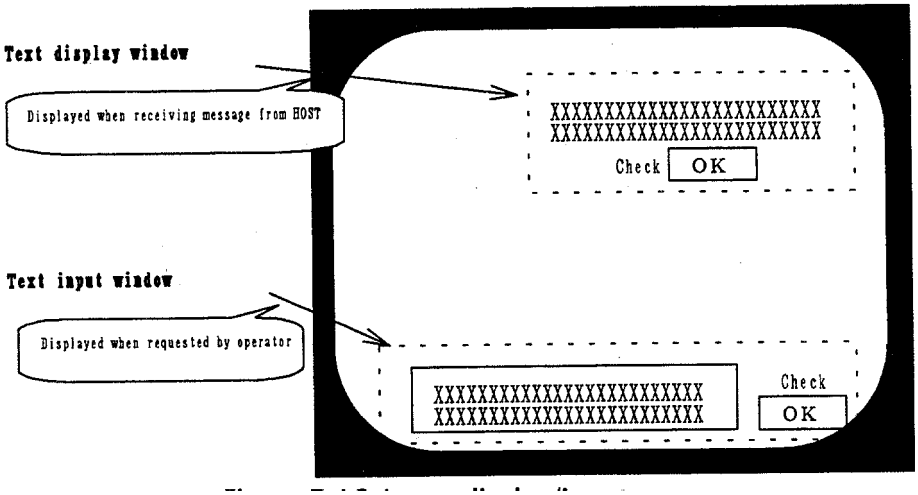

Figure 45 Text-display/input Screen

The equipment should support an operator-confirmation function. By taking the operator's confirmation of the message, the equipment generates an event report (S6, F11) and sends this to the host. The host interprets this to mean that the operator has confirmed receipt of the message. Until this confirmation is received, the host should not send any further text messages. This may complicate processes at the host end, but it prevents the host from sending one message before the operator has had a chance to read the previous one.

When the host wants to clear the text display on the equipment, it can send a message containing a text-data item of length 0 (S10, F3). On the other hand, the text entered through the user interface is sent to the host immediately after input is complete (S10, F1). In any case, data is being sent one way only; there is no bidirectional communication going on. It would be difficult to support interactive transactions under GEM.

Equipment terminal services are limited, but cannot be completely replaced by host terminals. If all host-terminal processes could be executed through the equipment's user interface, there would be no need for dedicated host terminals in the clean room at all, which might greatly improve clean-room operations. This presents certain problems though:

- The equipment's display is somewhat limited, and would not be able to show all the information the host would want it to. Unless the equipment had a full blown windowing system, all the terminal services would need to be squeezed into the rudimentary little equipment readout, which would not be practical.
- Likewise, the equipment's input capability is limited, and would not allow the operator to readily enter information as needed. Equipment terminal services support the transmission of textual data between host and equipment, and because of the equipment's limited display capabilities, it is difficult to provide user guidance or display options in a proper manner.
- SECS does not support text formatted in Japanese.[6]

[6] Editor's note: This should be available in 1998.

There could be some mechanism by which the equipment temporarily passed all of its screen input/output functions to the host, or there could be functions allowing host terminals to emulate the equipment, but until all equipment shares these features, it will remain very difficult to use the host terminal as a complete substitute for the equipment user interface.

Using TCP/IP (HSMS) allows the equipment user interface to support host terminal functions. As the transition from SECS-I to HSMS is made, equipment controllers will come with communications software like Telnet, which would allow remote login from the equipment terminal to the host. With a windowing system, the equipment terminal could be used to run both equipment operations and host operations simultaneously.

Entering material IDs and operator IDs through the user interface. The equipment user interface could be used for entering operator IDs and material IDs. This information could be keyed in (or scanned using an attached barcode reader) and transmitted to the host using the equipment terminal service functions (S10, F1), or an input device (barcode reader, etc.) could be hooked up to the serial interface and sent to the host as an event report (S6, F11). Note that since the S10, F1 message can carry any text string, the host may have no way of authenticating the contents of the data it receives.

Application examples of equipment terminal services. The equipment terminal services are convenient to use in Online Local operations. Described below is an example based on equipment in the Online Local state. Note that this is just an example, and not a GEM specification.

1. The operator has the Begin Operations display on the equipment terminal.
2. The operator enters necessary information like operator ID, material ID using a keyboard (or other input device).
3. The equipment transmits event report with this information to the host.
4. If the host sends an acknowledgment of this event report (ACKC6=0), it sends a process program download request. If the host returns some sort of error (ACK6>0), like a problem with the material ID, processing is stopped. When the host returns an error, it sends an explanation using S10, F3.
5. The equipment downloads the process program parameters from the host and displays them on the Begin Operations display.
6. The operator confirms the information in the display, and after making changes to the parameters (if necessary), orders the process start.
7. The equipment sends a Process Start event report to the host.
8. The host confirms the process start for the operator using S10, F3.
9. Process ends.
10. The host confirms the process end to the operator using S10, F3. It may also add text information ordering the next process.

Online Local start:

Host		Equipment	
		< S6, F11	Material ID entered by operator
Acknowledge material ID	S6, F12 >		Processing stopped if ACK6 > 0
If ACK6 > 0			
Report description of problem	S10, F3 >		
		< S10, F4	Acknowledge text and display
If ACK6 = 0			
		< S7, F5	Request process program
Send process program	S7, F6 >		
		< S6, F11	Report operations start order
Acknowledge operations start	S6, F12 >		
Display acknowledgment of operations start	S10, F3 >		
		< S10, F4	Acknowledge text and display
		< S6, F11	Report process end
Acknowledge process end	S6, F12 >		
Display operations end	S10, F3 >		
		< S10, F4	Acknowledge text and display

Equipment terminal services with multiple terminal on equipment. When a piece of equipment has multiple terminals, the question arises as to how the operator should confirm messages. In other words, if multiple copies of the same S10, F3 text message are displayed on multiple terminals, what procedure should you follow?

1. The operator can enter a confirmation from any one terminal, after which the message on that terminal is cleared and a confirmation event (S6, F11) is sent;
2. After the operator confirms the messages on all terminals, the message is cleared from all the screens and a confirmation event (S6, F11) is sent.

GEM does not define how this is supposed to be handled, but it would probably be best to allow the programmer to change this through variable settings.

8.4.8 Clock

The clock can be used to time-stamp all collection events. In any transaction, the equipment can request the time from the host (S2, F17/F18), or the host can set the time on the equipment (S2, F31/F32). Clock-synching should be performed before event reports that might use a time reference. In this document, the equipment requests the time from the host immediately after communications are established.

Why is it important for the host to have a clock? In any one request, all of the following elements may come into play: the host may be controlling or monitoring many pieces of equipment. Each piece of equipment may show a time different from the others, or from the host, and if so, the time-stamps on their messages will only confuse management. This is really just a

management problem, not a show-stopper, but if the host has a clock capability, the host can ensure that all pieces of equipment are in synch.

The host can also time-stamp messages it receives based on its own clock. This pretty much obviates synching problems between the host and equipment.

The clock capability cannot provide the equipment with the exact time. Since there will obviously be a time lag in communications between the host and equipment, so when the host sets the equipment's clock, there could be a difference of as much as a few seconds.

The clock capability in GEM is there to ensure the correct order of events. Another purpose for the clock capability is to ensure that the host knows the order in which events were generated by a given piece of equipment. It may also be helpful in investigating errors related to minor timing lapses.

8.4.9 Spooling

Host requests relating to spooling. As equipment operations grow more automated, they approach being online 24 hours a day, and the amount of process data that the operator must enter through a host terminal drops dramatically. This is most effective when the equipment can transfer large amounts of data to the host.

However, attempting these levels of automation increases vulnerability to problems with the communications system, which will shut down data transmissions from the equipment to the host. All process data may be lost, and may require a surprising amount of work to recover from the host terminal. Using the spooling capability makes process data collection more robust.

Spooled messages are saved. According to GEM, spooled messages should be stored in non-volatile memory, so even if power is lost, the spooled messages will be saved.

The message is the "unit" of spooling. This includes all primary messages other than S1, F1 and S1, F13, that is, any message whose loss through a communications breakdown could cause a problem for the fab's operations benefits from spooling. This would include a Process Complete event report that included process results, a process program revision request, a control state transition event, etc. Secondary messages are not spooled. Because it is involved in establishing communications, S1, F13 is not spooled, likewise S1, F1 is not, because it is involved in switching online.

Once the first switch online after establishing communications is complete, the host should always query whether spooling has been enabled. When the equipment supports spooling, this is the first thing the host should do after switching online (S6, F23/F24). This will prevent it from overwriting unused spooled messages.

How the host handles spooled messages. When the host receives a spooled message from the equipment, it comes in as plain data, subject only to the appropriate logging procedure. That is, when it receives a given message from the equipment, it handles it differently depending on whether the current operating state is normal (in which case it accepts the message as "spooling paused") or the state is error-recovery (in which case it accepts the message as "spool transmitting"). Until the spooled message is complete, the host treats it as a trigger for received messages, and will not transmit the order for the next command. If the host detects any sort of problem with the spooled message, it ignores the message and returns an error message, but other

than that, there is no special handling. If the equipment receives an error-reply, it just sends any subsequent messages it has ready, and need not have any special error-handling.

Spooling buffer size. The equipment's spooling buffer should be at least large enough for one process cycle. The reasoning behind this is that after a communications breakdown, the host will not be able to send an action-start command or process program selection to the equipment. The maximum buffer size would depend only on the hardware. In E30, the spool message is overwritten using the OverWriteSpool procedure, although conditions that overwrite the spool really have nothing to do with the spooling capability itself. Memory capacity sufficient for one process cycle should be enough.

Spooled data contains past values. Naturally spooled messages contain variable data, the values of which were set when the messages were generated by the equipment, so there may be discrepancies between the values shown in the spooled message when it is received and the values currently in effect.

Are retry messages spooled? Spooling begins when a loss of communications is detected and the communications state is set to Not Communicating, so even if message transmissions are retried, the message will not be spooled until Retry Over is detected. A new message created during a retry will normally be placed in the message queue. When Retry Over is detected, the message being retried and queued messages are all sent to the spool memory. After this, an S1, F13 retry can begin. Even if the S1, F13 message is lost, it is not spooled.

Flow control in spool transfer processing. Spool transfers are normally conducted as single transfers. The equipment will not send the next message until it receives the secondary message from the host, so the spool-transfer speed is limited only by the host's ability to receive it. When the transmitted message is only a primary message that needs no reply, the equipment will transfer the message to the host as fast as possible. When the host and equipment have different message-processing abilities, the host will try to accept spooled messages from the equipment continuously. In this case, there needs to be some kind of flow control, so the equipment can be set to transmit only up to a certain maximum number of spooled messages (MaxSpoolTransmit), which limits the number of messages the equipment will transmit continuously. The host must send however many spooled data requests (S6, F23) that are needed to complete the spooled message transmission.

Spooling multi-block messages. The process program inquiry (S7, F1) can only be used when multi-block permission has been requested, so it is not spooled. If a communications error is detected during a multi-block transfer, the multi-block message is spooled from the beginning. If a multi-block message is spooled from the middle, it cannot be sent as a spool transfer.

Switching offline during a spool transfer. Under GEM, when you switch equipment offline, all transactions that were opened by the host are completed, and this applies to spool transfers as well.

When switching to an offline state (either Equipment Offline or Host Offline), the equipment will stop spool transfers, and the spool state will transition to No Spool Output. The Equipment Offline event is then sent to the host.

Since the equipment is switched offline and transfers are not possible, spooled data is saved. However, before the host receives a Spooling Deactivated event, it will receive an Equipment

Offline event. This way, the host knows that the equipment is going offline before the spool transfer is complete.

After a subsequent switch online, the host can send a Request Spooled Data or Request Spooled Data Purge. Because of this, we must add Switch Offline to the E30 conditions involved in a state transition from Transmit Spool to No Spool Transmit, along with Communications Failure, and Max Spool Transmit (E30-95, p. 434, Table 4.11, “Spooling State Transitions”).

Switching offline during a spool purge. Because purging is an operation that can be completed internally by the equipment, switching offline during a purge still allows the purge to be successful, but a Spooling Deactivated event will not be transmitted to the host. Since the host has not received this report, it may not know whether the purge succeeded, so after the subsequent switch online, the host should request the number of actual spooled messages (SpoolCountActual) with S1, F3/F4 so that it can either confirm the purge’s success or reorder it.

The equipment acts on the Switch Offline order when it receives it. This document assumes that when the equipment receives a Switch Offline Request from either the operator or host, it executes it immediately (taking into account whatever time is needed for the equipment’s internal processes). Under GEM, switching to an offline state causes the equipment to complete all host-initiated transactions. After that, the equipment will ignore any incoming primary message from the host other than Establish Communications or Switch Online. All the foregoing information on spooling should apply here as well.

Reserve Switch Offline. An offline reservation allows the equipment to switch itself offline after a certain event, rather than switching it offline as soon as the order comes in. This is an extended capability in GEM. Using this function makes possible the process described below.

Deny or suspend temporarily a switch offline during spool transfer. A Switch Offline Request received during a spool transfer or spool purge can be either denied or suspended temporarily, although this assumes that it will be executed once the spool operation is complete. The equipment responds to the host’s request with OFLACK (S1, F16), appending a Deny response. The equipment can suspend the switch by sending the Deny response and then sending an Equipment Offline event report immediately after that. If the equipment does not use the denial to either suspend or reserve the switch offline, the equipment or operator (as the case may be) will not be able to plan subsequent actions, so failure to do so would be a bad idea.

8.5 Compatibility Between Different Individual Capabilities

E30 does not describe in detail the compatibility between the various capabilities and the effects they have on one another. This section addresses the issue for important capabilities. The following table recapitulates some of the information in previous sections, but in a more structured form.

Table 8 Compatibility between individual capabilities (numbers show referenced section)

	Spooling	Remote Control	Event notification	Control states	Establish communications	Equipment process states
Equipment process states	**1**	**2**	**3**	**4**	**5**	
Establish communications	**5**	**6**	**7**	**8**		
Control states	**9**	**10**	**11**			
Event notification	**12**	**NA**				
Remote Control	**NA**					
Spooling						

8.5.1 Equipment Process States and Spooling

Spooled message buffer size. The message buffer should be at least large enough to store all the messages generated during one material-processing cycle.

8.5.2 Equipment Process States and Remote Control

Starting and stopping material processes. The remote-control capability must at minimum be able to order process starts and process stops.

8.5.3 Equipment Process States and Event Notification

The order in which event reports are sent follows the equipment processing state model. If an equipment state model (processing state model, etc.) is already defined, then the event reports follow the same state transition table sequence and timing. If the host is providing the state model and transition table, the sequence of messages sent and received will be based on that. The order of messages sent by the equipment must agree with the transitions mapped out in the processing state model.

A situation where the order for the equipment generating an event and the host receiving the event is unclear. Look at the Report Process Complete event and the Report Process Results Data event. The state transition sequence shows that the process results are reported first, the process complete report after. If these two messages are combined in one block, they are still reported to the host in the defined order. However, if the event report capability is being used, and the host is receiving a process results report containing a very large number of variable reports, the equipment may break up the process results data into a multiblock transmission.

If one assumes that the equipment supports interleaving, the process-complete report could be sent between parts of the process results data. When the host receives the process-complete report, it will re-initialize the current process state and execute other related processes, and if any process-results data appears after this initialization has been completed, it might be ignored.

Be careful in handling reports to avoid getting confused by ambiguous report sequences.

8.5.4 Equipment Process States and Control States

Changes in the control state may be dictated by the processing state. For example, only switch to the Equipment Online state when in the Idle process state. In this case, after operations commence in the Host Offline state, even if the equipment receives a Request Online (S1, F17) from the host, it will deny it (ONLACK=1).

Switching online/offline by the host. Following is a discussion of setting the control state, that is, switching between online and offline. Generally, the host receives an Online report from the equipment and then begins to monitor or control it. When it receives an Offline report, it stops monitoring, and clears any Equipment Operating States generated during the monitoring. When it detects a Switch Offline report from the equipment, if the host clears its Equipment State information, then the next time it receives a Switch Online report, it will revert to initial settings. If any troubles have arisen and it becomes desirable to restart equipment operations from initial settings, switching offline and then online is an easy way for the operator to accomplish this.

The host may react badly if the same material is being processed continuously throughout an online/offline switch. If it receives an Online report during equipment operations, it will record the current operating state and re-start monitoring in mid-process, which is just nuisance work that it cannot perform correctly anyhow. The equipment should always be idling when sending an Online report, as this lowers the processing burden on the host.

When switching online, the host sets an Equipment Operations State, and likewise when you switch offline, the host updates the mid-process state and then clears it. Consider switching online as re-initializing this state without intentionally clearing it.

As far as the operator in the clean room is concerned, switching between online and offline states is just starting and stopping automated operations. Figure 46 shows the host's state transitions in an online/offline switch.

Switching offline during operations still allows the equipment to continue operating. However, if the equipment is set to idle after completing an online operation and the equipment is not switched online, it will be impossible to re-open online operations afterward.

As in Equipment Online Operations Reset, it is proper to handle switches corresponding to switching offline separately. In this case, even after the operating state has been reset, the control state will remain Online. This makes the purpose of the message clearer. However, repeatedly restarting operations will require separate messages.

In any case, when there are discrepancies between the equipment's actual state and the state perceived by the host, there needs to be a simple way to get them in synch. With most equipment, you achieve this through shutting down or switching offline.

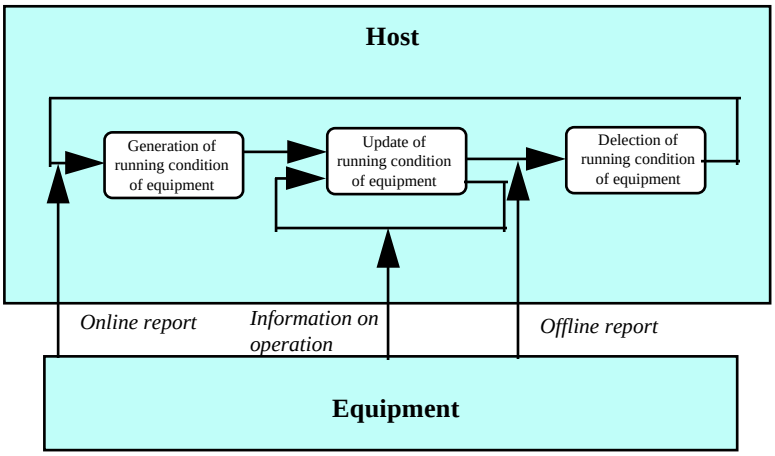

Figure 46 The Host's State Transitions During Online/Offline Switches

Switching from offline to online, or from local to remote during operations. Contrary to the previous part, there are some situations where it is necessary to switch from offline to online or from local to remote. Suppose that operations begin online, but some problem arises in the middle of material processing, requiring that a switch to either offline or to Online Local. Once the recovery is complete, you will want to switch online of back to Online Remote. Or if there is a problem with the host, equipment, or communications lines, and operations began in the online mode, return to the online mode after recovering. This is not to say that the material in the equipment must be cleared and subsequent units of material stopped, as that would harm operating time and throughput.

In these cases, the switch online, or to Online Remote, is not a new state but a return to an old one. It would also be possible for the host to order a switch online or to Online Remote, in order to manage the start or continuation of material processing.

8.5.5 Equipment Process States and Communications Establishment

The processing state is not dictated by changes in the control state or communications state. It is possible to execute processes while offline, and it should be possible to do so if the process begins online and is switched offline, or if the communications state goes to Not Communicating or Interrupted. The processing state should not be influenced by these state changes.

8.5.6 Communications Establishment and Spooling

Communications errors and re-establishment are spool-processing triggers. Spooling begins when the equipment detects communications errors or the communications state goes to Disabled. When communications are re-established, the host orders either a spool transfer or spool purge, which the equipment then executes.

8.5.7 Communications Establishment and Remote Control

Time dependencies for the host to use the remote-control capability. Remote control becomes available only after communications have been established and the control state set to Online Remote.

8.5.8 Communications Establishment and Event Reporting

Switching the communications state to Not Communicating disables events. The host cannot even be informed of this event through event reporting, as the equipment cannot send any messages to the host.

8.5.9 Communications Establishment and Control States

The transaction timer timeout and switching offline. If, in an online state, the equipment detects a transaction timer timeout, it sends a communications error message (S9, F9) to the host. Should the equipment then remain online or switch offline? There is no particular relationship between the control state capability and the error message capability, such as a transaction timer timeout (communications error: fault). As such, it is best to treat the two as unrelated. So the short answer is, stay online.

Communications failures while online and recovery processing. Even if communications have been lost while in an online state due to exceeding an RTY limit, maintain the Communicating and Online states. If the equipment succeeds in establishing communications by sending S1, F13, it should automatically to normal online operations.

If communications are set to Not Communicating, there should be a separate control state transition. E30 does not define any particular relationship between the control and communications states. In reality, of course, if communications are not available, the control state (especially the online states) is pretty much meaningless. If the communications state is set to Not Communicating, then the control state should be set offline, or to some other mode, such as Manual. Leaving the control state online after communications have become unavailable could mess up equipment operations.

8.5.10 Control State and Spooling

The current control state cannot be judged from spooled messages. Messages in a spool transfer are messages from the past. Be careful not to infer anything about the current control state from them.

8.5.11 Control State and Remote Control

Remote control order. Remote control orders are only in effect when the control state is Online Remote.

8.5.12 Control State and Event Notification

Changes in the control state are reported as events.

8.5.13 Event Notification and Spooling

The spooling mechanism itself creates event report messages. This is not explained in GEM, but the figure below gives a simplified picture of how this works. Remember that:

- The spooling mechanism and event reporting mechanism are completely independent
- Any kind of message can be sent to the spooling mechanism

- Spooling mechanism event reports are themselves spooled just like any other event report

Bearing this in mind, you should understand that event reports generated by the spooling mechanism that themselves get spooled will be the last to be added. The spooling mechanism cannot change the position of a spool-related message in the queue. Suppose that communications are lost during a spool transfer. Since the resultant Spool Transfer Failure event report cannot be sent, it will be spooled at the end of the remaining messages.

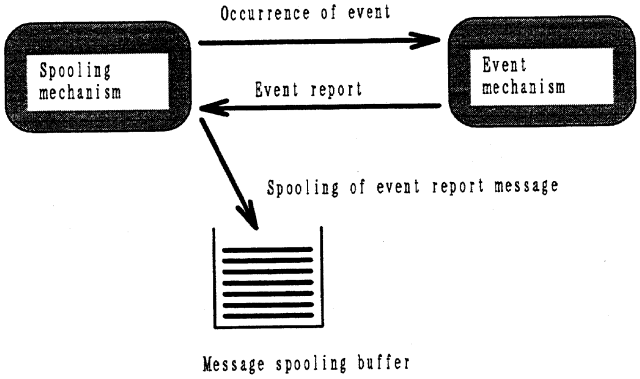

Figure 47 Event Reporting and Spooling

9 OTHER FEATURES

1. Block control (cell control)

Having several pieces of equipment grouped together to be jointly controlled is known as block control, cell control, or inline control. Block control is widely used in photolithography and diffusions processes. Figure 48 shows a block-control schematic.

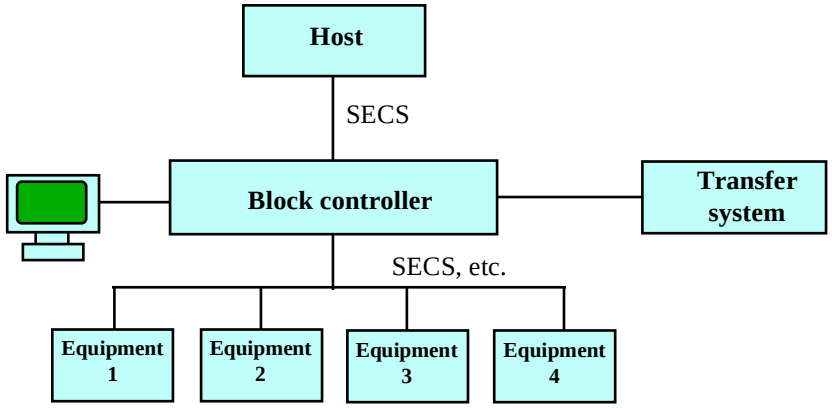

Figure 48 Block Control

9.1 Block Control Functions

1. Simplify and coordinate host interface
2. Centralized process program management
3. Centralized process result management
4. Equipment operations via block-control terminals
5. Wafer transport between pieces of equipment
6. Realtime equipment monitoring and operating time management
7. Automated transport system interface
8. Joint Pilot wafer processing and joint process control with detection equipment

Block controllers should consist of a fast CPU, video display, big hard disk, and other standard pieces of computer hardware. It may also need to be able to communicate with the equipment under its control over a TCP/IP interface. One advantage of block controllers is that they can provide high-level services that would be difficult to support for each piece of equipment separately. Block controllers are usually supplied by equipment suppliers or systems vendors.

Where the block controller fits in. The block controller functions described above provide a layer of control closer to the individual pieces of equipment than that typically supported by the host. That is to say, it offers control functions to equipment groups that are typically requested in processing or by users. This means that at some fabs, the host and block controllers will reproduce some of each other's functions. Figure 49 shows this graphically. The functions that the block controller supports can be interposed between the equipment and host, offering the user duplicated services.

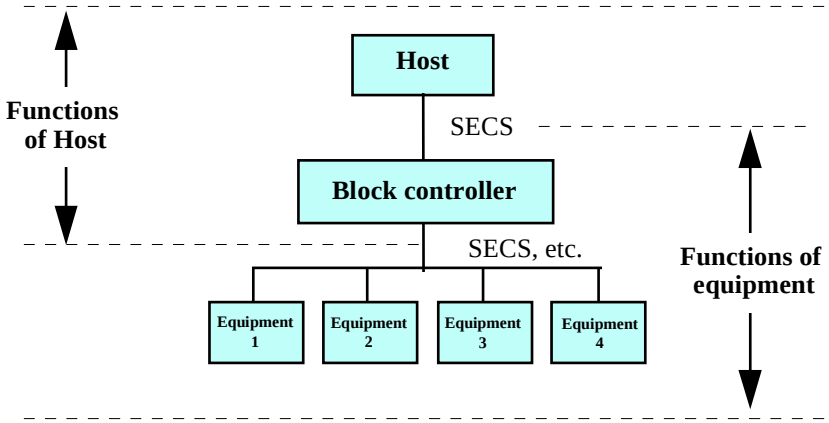

Figure 49 Where Block Controller Fits In

The block controller/host interface. GEM does not address the block-controller model. The interface the host has with the block controller is not the same is it is with the equipment; you need to pay due care to the organization of the host's management functions. Designing for grouped equipment control from the start helps avoid problems when installing a block controller, but the grouped control functions the host has often will overlap with the block controller

Since the block controller provides a layer of control closer to the equipment than the host does, it is susceptible to operating changes in the host. The block controller's control software needs to be more flexible in its host interface than does the equipment's interface software. Some points to keep in mind for the block controller:

- Provides an interface to the host for the controlled equipment group
- May need to provide an interface to the host for individual pieces of equipment
- Has grouped control functions that respond flexibly to changes in the host's operations. In particular, should be able to accommodate differing processes or differing types of equipment, as demanded by the host

A program control system using distributed processing. If the block controller was not supplied by the equipment manufacturer and each piece of equipment cannot be directly connected to the host, then it will be impossible to bring the equipment online, making matters difficult for the host.

Conversely, if the equipment manufacturer does supply a block controller, it may or may not provide functions needed by the host. These might include recipe management, operating time management, or real time process monitoring.

Figure 50 shows a distributed system model designed to meet these requirements. The host communicates with the equipment directly, and runs things by remote control. The block controller provides services to the host that relate to equipment groups, such as recipe management. Each piece of equipment communicates with both the host and block controller, treating them as two hosts. Here, it is somewhat inaccurate to use the phrase "block controller"—instead, "process data management server" would be more appropriate.

If the block controller is being used to control an equipment group directly, it is permissible to have the host communicating only with the block controller. Remote terminals on the network should be able to access either system. Note that a distributed system like this is precedented upon the availability of HSMS or some kind of high-speed data network.

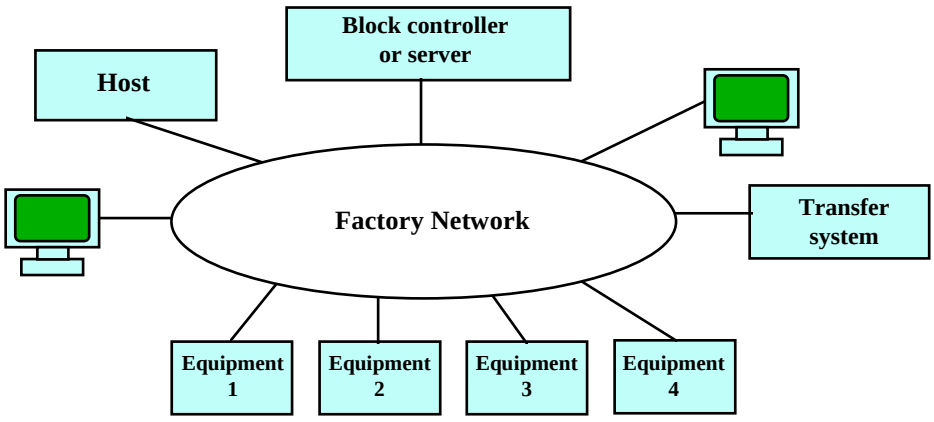

Figure 50 Block Controller in a Distributed Processing System

9.2 Material tracking

GEM does not address material tracking. If equipment is continuously processing multiple units of materials, the equipment must keep track of the materials for the host, and report processing information to the host with material IDs for each unit of materials. This is illustrated in Figure 51. When the material is loaded in the equipment, the equipment sends an event report to the host. When the host receives this, it sends an appropriate material ID. The equipment subsequently reports each change in the material's status to the host using this ID. In fact, the equipment may use its own ID internally without notifying the host. The equipment stores the material ID it receives, but need not use it except for reporting to the host. If the equipment has a way to automatically recognize the material ID, the host need not send it.

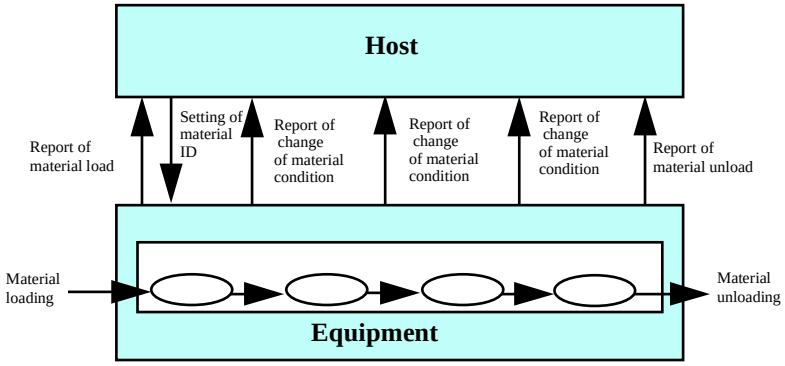

Figure 51 Material tracking in the equipment

The material ID is set using remote-control commands (S2, F41/F42). Following is a scenario showing the material ID setting process, up until recipe setting.

Host		Equipment	
		< S6, F11	Material loaded report
			(cassette location)
Acknowledge	S6, F12 >		
Set material ID	S2, F42 >		
(cassette location)			
		< S2, F42	Acknowledge material ID
		< S6, F11	Acknowledge material ID event
Acknowledge	S6, F12 >		
Set recipe	S2, F41 >		
(cassette location or material ID)			
		< S2, F42	Acknowledge recipe
		< S6, F11	Select recipe event
Acknowledge	S6, F12 >		

9.3 GEM and SEM

GEM has a great degree of freedom. GEM was not meant for equipment used in isolation, so the way GEM is applied may differ when equipment types differ. From the perspective of designing an interface for one piece of equipment, GEM offers a great deal of freedom. The original schedule called for GEM to be developed before Specific Equipment Models (SEMs), which just now are entering development.

SEM development. In attempting to develop a more specific standard than GEM, it is impossible to develop an equipment model without considering differences in configuration and operation for different equipment types. This is the area that SEM addresses. SEM addresses distinct equipment models for major classes of equipment (implanters, etchers, furnaces, steppers, etc.). Some of the issues underlying SEM development are as follows:

- Even with generally-recognized categories of equipment, design concepts and configurations will differ among manufacturers. Defining a common model requires extracting the relevant features.
- Even if the host sees various pieces of equipment as different, it should still be able to treat them identically to the greatest extent possible. Defining SEM will require finding which capabilities are special to one category of equipment, and which are common to all.
- While there is a wide variety of production equipment in any one fab, SEM should not address differences in equipment between every single process stage. Specifying the differences between every piece of equipment will only lessen SEM's impact.
- Making the functions too specific leaves them vulnerable to changes in technology. In defining the functions, due attention must be paid to their survivability.

9.4 GEM and HSMS

HSMS promotes a client-server architecture. Installing HSMS ties your equipment directly into the network. Since HSMS runs on TCP/IP, you can run various higher-level protocols (FTP and Telnet), or download files using a networking OS: you can extend the same network services connecting your computers to your processing equipment and host. This allows you certain new capabilities:

- Remote terminals on the network can access recipe management and editing functions resident on the processing equipment
- Large files (recipes, process log data) can be transferred using FTP, and files on the host can be directly accessed
- Remote diagnostics and maintenance
- Logging in from an equipment terminal to the host to access production information

There are still a lot of issues to be worked out here, including hardware/software standardization, security, etc.

10 CONCLUSIONS AND FINAL CONSIDERATIONS

This document has examined a number of aspects of GEM as they are perceived by the host. The basic points are:

1. GEM reduces equipment software development costs and improves functionality and reliability by standardizing the host/equipment interface.
2. GEM does not define a subset of SECS scenarios. It offers a uniform, systematized specification providing flexibility for host requests and equipment functions.
3. GEM is a technological foundation for future CIM in semiconductor fabs. It allows for future expansion of capabilities.
4. In order to implement GEM and HSMS effectively, the processing equipment should use common hardware and software.
5. GEM software must provide adequate flexibility to deal with revisions and expansions to equipment functions.

GEM is the first state toward open semiconductor-CIM systems. As these CIM systems become more standardized, CIM software product lines that offer more flexible interfaces and configurations should become available. This will allow users to purchase software off the shelf, rather than creating it themselves.

10.1 In Closing

This document was intended to explain GEM from the host's perspective. As GEM implementation spreads, more questions and problems are sure to arise. The SEMI standard committees in the U.S. and Japan will continue working to resolve these questions and to expand the range of GEM applications. The ongoing GEM development process is an excellent opportunity for readers to get involved.

10.2 References

1994 SEMI International Standards production equipment (software)

APPENDIX A
Survey Results – SEMI Japan Survey on SECS Implementation

Summary

Equipment makers showed great interest in GEM. There were several manufacturers implementing GEM, with almost half either using it or planning to do so. Semiconductor manufacturers had little interest in GEM. Some respondents had never heard of GEM, although there were also several planning to implement it. Although both equipment manufacturers and semiconductor manufacturers felt GEM offered the promise of acquiring software from outside suppliers, software vendors themselves did not see it as an important business opportunity.

Survey conducted May 1995

Number of respondents:

Semiconductor manufacturers:	11 (16 factories or companies)
Equipment manufacturers:	16 (18 factories or companies)
Software suppliers	4 (systems integrators)
Total:	31 (40 factories or companies)

Respondent's job description

	Semiconductor manufacturer	Equipment maker	Software supplier
Development	7	3	1
Support	5		
Software engineer	2	7	
Non-software engineer	5	1	
Technical support	1	1	1
Technology management	2	7	3
Sales			2
General management			
Other			1

General questions

1. Awareness of communications standards

	Don't know			Know a little			Know well		
	Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software
SECS-I	2			4	3		10	15	6
SECS-II	2			5	4		9	14	6
HSMS	7	1		8	13	4		4	2
GEM	**6**	**1**		**10**	**7**	**5**		**10**	**1**

2. Importance of seminars on these standards

	Semi-conductor	Equipment	Software
Not important		2	2
Might attend if offered	10	9	4
Would certainly attend	5	7	
No response	1		

3. Which seminars would you attend?

	Semi-conductor	Equipment	Software
SECS-I	2		1
SECS-II	5	1	1
HSMS	11	12	3
GEM	**13**	**8**	**4**
No response	1	1	

4. Importance and implementation of specific GEM functions (respondents here skip question 5)

	Semicond uctor	Eqpmnt	Software
Don't know GEM well	7	2	1
GEM is not important			
Haven't thought about it	6	7	3
Other	1	1	1
No response	2	11	1

5. Importance of specific GEM functions

	Don't know			Know a little			Know well		
	Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software
state model				3	5			7	1
eqpmnt process model		1		2	5		1	6	1
S1,F13/F14 scenario	2			2	4			8	1
event notification	1			2	4		1	8	1
ack online				1	4		3	8	1
error msgs				1	4		3	8	1
operator-initiated cntl				2	5		3	7	1
est. comm	2			1	4		1	8	1

	Don't know			Know a little			Know well		
	Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software
dynamic event rpt setting	2	2		2	5			5	1
variable data collection				3	6		1	6	1
trace data collection	2	2		3	6			4	1
state data collection				1	4		3	7	1
alarm mgmt				1	4		3	8	1
remote cntl	1	2		3	5			5	1
eqpmt consts	1			2	7		1	5	1
process pgm mgmt	1	2		2	7		1	3	1
eqpmt tmnl svcs	2	2		2	4			6	1
clock	1			2	3		1	9	1
remote monitoring	2	8		2	4				1
spooling	1	6		3	4			2	1
host-initiated cntl	3	4		1	4			4	1

6. Are there any functions that GEM does not support that you think you will need?

Yes			No		
Semi-conductor	Equipment	Software	Semi-conductor	Equipment	Software
2	8	1	2		

Questions for equipment manufacturers

1. What standards have you implemented in your equipment?

	No	Studying	Yes	No response
SECS-I	2	1	15	
SECS-II	2	1	15	
HSMS	13	5		
GEM	**8**	**2**	**7**	

2. Plans to implement SECS functions

	SECS-I	SECS-II	HSMS	**GEM**	No response
Plan to implement	1	1	7	**7**	5
Substituting something else			2	**3**	14

3. Level of SECS implementation

	Not implemented	Implementing other system	Partially implemented	Standard implementation	No response
SECS-I	2			7	
SECS-II	2			9	
HSMS	17	1	1	1	
GEM	**9**	**4**	**4**	**4**	

4. Importance of SEMI standards

	Not important	Desirable	Necessary	No response
SECS-I		10	10	
SECS-II		9	11	
HSMS			6	2
GEM			**14**	**1**

5. Demand for SECS software packages

Develop in-house; not needed..... 2

Would consider, depending on price and performance.....12

Would prefer to buy from outside supplier:

SECS-I.....5

SECS-II.....5

HSMS.....15

GEM.....12

No response.....1

6. SECS software development strategies

Develop as requested by customers.....4

Offer own standard, do not offer major customizations.....2

Customize based on own standard.....14

Questions for semiconductor manufacturers

1. Have you implemented a host computer system?

	No	Studying	Yes	No response
SECS-I	3	1	13	1
SECS-II	2		14	1
HSMS	16		1	
GEM	**13**	**2**	**1**	

2. Plans to implement SECS functions

	SECS-I	SECS-II	HSMS	**GEM**	No response
Plan to implement	1	1	5	**3**	8
Substituting something else	2	2	3	**2**	9

3. Implementation of SECS functions in fab's host computer

	Not implemented	Implementing other system	Partially implemented	Standard implementation	No response
SECS-I	2		2	11	1
SECS-II	2	3	2	9	1
HSMS	12	3			1
GEM	**13**	**1**	**1**		**1**

4. Plans to bring more equipment online

Current implementation is adequate.....2

Plan to expand online implementation.....12

No response.....0

5. Priority for putting equipment online in fab

Not important.....0

Case-by-case.....7

Always required.....9

6. Problems bringing equipment online

Cost of developing equipment-end software.....14

Time to get equipment-end software.....7

Quality of equipment-end software.....9

Functionality of equipment-end software.....6

Cost of developing host-end software.....8

Time to get host-end software.....3

Quality of host-end software.....4

Functionality of host-end software.....5

7. Importance of SEMI standards

	Not important	Desirable	Necessary	No response
SECS-I		2	13	1
SECS-II		2	13	1
HSMS	1	10	1	4
GEM	**1**	**9**	**3**	**3**

8. Demand for general-purpose SECS software packages

Develop in-house; not needed.....2

Would consider, depending on price and performance.....13

Would prefer to buy from outside supplier:

SECS-I.....2

SECS-II.....3

HSMS.....9

GEM.....11

No response.....2

Questions for software suppliers

1. Have you sold any products implementing SECS functions?

	No	Studying	Yes	No response
SECS-I	1		5	
SECS-II	1		5	
HSMS	6			
GEM	**4**		**1**	1

2. Plans to offer SECS software

	SECS-I	SECS-II	HSMS	**GEM**	No response
Plan to implement	1	1	5	**3**	8
Substituting something else	2	2	3	**2**	9

3. Type of sales

Custom-ordered.....2

General-purpose packaged software.....13

4. Potential foreseen for general-purpose SECS software packages

None.....1

Minor niche market.....3

Will probably expand with time.....2

No response.....1

SEMATECH Technology Transfer
2706 Montopolis Drive
Austin, TX 78741

http://www.sematech.org