

SEMI E40-1107

STANDARD FOR PROCESSING MANAGEMENT

This standard was technically approved by the global Information & Control Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on April 25, 2007. It was available at www.semi.org in October 2007 and on CD-ROM in November 2007. Originally published in 1995; previously published March 2007.

1 Purpose

1.1 Automated management and command of material processing in equipment is a crucial aspect enabling factory automation. This standard addresses the communications needs within the semiconductor manufacturing environment with respect to the processing of material in equipment.

1.2 This standard specifies the application of the appropriate processing to specified material received at the processing agent. It describes the concepts of material processing, the behavior of the equipment in relation to processing, and the messaging services which are needed to accomplish the task.

1.3 The communications services defined here enable standards-based interoperability of independent systems. They allow application software to be developed that can assume the existence of these services and allow software products to be developed which offer them.

1.4 Implementation of automated processing management will help eliminate mis-processing of material. The adoption of the standards described will greatly reduce the effort required to integrate compliant equipment components and reduce time to set up for processing. Compliance requires a minimal but specific set of standard services.

2 Scope

2.1 The scope of this standard is automated material processing based on discrete processing jobs. It provides the functionality required for process management for modules within a cluster tool. It may be applied to sub-systems of other multi-resource equipment, as well as to host control of many types of equipment.

2.2 This standard supports individual management of jobs for identical processing of material within a group and concurrent processing of independent groups. Where material contains other material (such as carriers containing wafers), processing may be specified in terms of either material type.

2.3 A simple tuning mechanism is provided for limited feedforward and feedback control between process steps. A method is defined for taking advantage of recipe variable parameters. This is not expected to satisfy all closed loop control requirements. Other mechanisms are anticipated with greater flexibility for late tuning and handling complex data.

2.4 This standard does not provide services for receiving material for processing, or disposing of it after processing is complete. Automation of material transfer is assumed to be provided through other services, such as those defined in applicable SEMI standards.

2.5 This standard presents a solution from the concepts and behavior down to the messaging services. It does not define the messaging protocol.

2.6 A messaging service includes the identification that a message shall be exchanged and a definition of the data which is contained in that message. It does not include information on the structure of the message, how the data is represented within the message, or how the message is exchanged. This additional information is contained within the message protocol.

2.7 The defined services may be applied to multiple protocols. Information on the mapping of processing management services to special protocols (e.g., SECS-II) are added as adjunct standards.

2.8 The services assume a communications environment in which a reliable connection has been established between the user of the services and the provider of the services. Establishing, maintaining, releasing a connection, and handling communication failures are beyond the scope of this standard.

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

3 Referenced Standards and Documents

3.1 SEMI Standards

SEMI E5 — SEMI Equipment Communications Standard 2 Message Content (SECS-II)

SEMI E30 — Generic Model for Communications and Control of Manufacturing Equipment (GEM)

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services

SEMI E53 — Event Reporting

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

4 Terminology

4.1 The following definitions are arranged in alphabetical order. Some definitions use terms defined elsewhere within this section. No references beyond this section should be necessary for a basic understanding of these terms.

4.2 Definitions

4.2.1 *agent* — an intelligent system within a factory that provides one or more service resources and uses the services of other agents. A generalization of host, equipment, cell, cluster, cluster module, station controller, and work station. Agents are associated with a physical system or a collection of physical systems, including computer platforms.

4.2.2 *form* — type of data representing information contained in an object attribute or service message parameter.

4.2.3 *fundamental requirements* — the requirements for information and behavior that must be satisfied for compliance to a standard. Fundamental requirements apply to specific areas of application, objects, or services.

4.2.4 *post-conditioning* — activities performed by the processing resource after departure of the material being processed but related to the processing of that material (e.g., cleanup).

4.2.5 *pre-conditioning* — activities performed by the processing resource before arrival of the material being processed but related to the processing of that material.

4.2.6 *processing agent* — an intelligent system within a factory which is independently capable of providing manufacturing value added to material.

4.2.7 *processing resource* — an entity within a processing agent which provides the manufacturing value added to material.

4.2.8 *process job* — a material processing job for a processing resource specifying and tracking the processing to be applied to the material.

4.2.9 *recipe* — the pre-planned and reusable portion of the set of instructions, settings, and parameters under control of a processing resource that determines the processing environment seen by the material. Recipes may be subject to change between runs or processing cycles.

4.2.10 *recipe executor* — a component of a module that stores and executes recipes.

4.2.11 *recipe namespace* — a logical management domain with the responsibility for the storage and management of recipes. It ensures the uniqueness of recipe identifiers and provides services pertaining to recipes stored within that domain.

4.2.12 *service* — the set of messages and definition of the behavior of a service-provider that enables remote access to a particular functionality.

4.2.13 *service-provider* — the software control entity that is the provider of a particular functionality which may be accessible remotely.

4.2.14 *service-user* — the software control entity that is the user of any of the related services.

4.2.15 *supervisor* — an entity or entities having supervisory control responsibilities for one or more processing resource. It is the service-user of the processing management services.

4.2.16 *tuning* — specification of information which supplements the pre-defined recipe used to achieve the particular process goals.

4.3 Data Type

4.3.1 *boolean* — may take on one of two possible values, equating to TRUE or FALSE.

4.3.2 *enumerated* — may take on one of a limited set of possible values. These values may be given logical names, but they may be represented by any single-item data type.

4.3.3 *form* — type of data: positive integer, unsigned integer, integer, enumerated, boolean, text, formatted text, structure, list, ordered list.

4.3.4 *formatted text* — a text string with an imposed format. This could be by position, by use of special characters, or both.

4.3.5 *integer* — may take on the value of any negative or unsigned integer. Messaging protocol may impose a limit on the range of possible values.

4.3.6 *list* — a set of one or more items that are all of the same form (one of the above forms).

4.3.7 *ordered list* — a list for which the order in which items appear is significant.

4.3.8 *positive integer* — may take the value of any positive whole number. Messaging protocol may impose a limit on the range of possible values.

4.3.9 *structure* — a complex structure consisting of a specific set of items, of possibly mixed data types, in a specified arrangement.

4.3.10 *text* — a text string. Messaging protocol may impose restrictions, such as length or ASCII representation.

4.3.11 *unsigned integer* — may take the value of any positive integer or zero. Messaging protocol may impose a limit on the range of possible values.

5 Conventions

5.1 *Harel State Model* — This document uses the Harel State Chart notation to describe the dynamic behavior of the objects defined. An overview of this notation is presented in an appendix of SEMI E30. The formal definition of this notation is presented in *Science of Computer Programming 8*, “Statecharts: A Visual Formalism for Complex Systems,” by D. Harel, 1987.

5.1.1 The Harel notation does not include the concept of “creation” and “deletion” of state models to represent transient entities. The “job” described in this document is such an entity, where each new job created uses a copy of the same state model. In this document, an oval is used to denote the creation of an entity and also the deletion of that entity.

5.1.2 Transition tables are provided in conjunction with the state diagrams to describe explicitly the nature of each state transition. A transition contains columns for Transition #, Current State, Trigger, New State, Action(s). The “trigger” (column 3) for the transition occurs while in the “current” state. The “actions” (column 5) include a combination of (1) actions taken upon exit of the current state, (2) actions taken upon entry of the new state, and (3) actions taken which are most closely associated with the transition. No differentiation is made.

5.1.3 The state models included in this standard are a requirement for Processing Management compliance. A state model consists of a state model diagram, state definitions, and a state transition table. When using SEMI E30, SEMI E53, or similar style collection events, all state transitions in this standard, unless otherwise specified, shall correspond to collection events.

5.1.4 A state model represents the host’s view of the equipment, and does not necessarily describe the internal equipment operation. When using collection events, all Processing Management state model transitions shall be mapped sequentially into the appropriate internal equipment collection events that satisfy the requirements of those transitions. In certain implementations, the equipment may enter a state and have already satisfied all of the

conditions required by the Processing Management state models for transition to another state. In the case, the equipment makes the required transition without any additional actions in this situation.

5.2 *Object Attribute Representation* — The object information models for standardized objects will be supported by an attribute definition table with the following column headings:

<i>Attribute Name</i>	<i>Definition</i>	<i>Access</i>	<i>Requirement</i>	<i>Form</i>
The formal text name of the attribute.	Description of the information contained.	RO or RW	Y or N	(see below)

5.2.1 The Access column uses RO (Read Only) or RW (Read and Write) to indicate the access that service-users have to the attribute.

5.2.2 A ‘Y’ or ‘N’ in the requirement (Rqmt) column indicates if this attribute must be supported in order to meet fundamental compliance for the service.

5.2.3 The Form column is used to indicate the format of the attribute (see ¶ 4.1 for definitions).

5.3 *Service Message Representation*

5.3.1 *Service Resource Definition* — A service resource definition table defines the specific set of messages for a given service group, as shown in the following table:

<i>Message Service Name</i>	<i>Type</i>	<i>Description</i>
Message Name	N or R	The intent of the service.

5.3.1.1 Type can be either N = Notification or R = Request.

5.3.1.2 Notification type messages are initiated by the service provider, and the provider does not expect to get a response from the consumer/subscriber.

5.3.1.3 Request messages are initiated by a service consumer or subscriber. Request messages ask for data or an activity from the provider. Request messages expect a specific response message (no presumption on the message content).

5.3.2 *Service Parameter Dictionary* — A service parameter dictionary table defines the parameters used in a service, as shown in the following table:

<i>Parameter</i>	<i>Form</i>	<i>Description</i>
Parameter X	Data Type	A parameter called X is B in A.

5.3.2.1 A row is provided in the table for each parameter of the service. The first column contains the name of the parameter. This is followed by columns describing the form and contents of the corresponding primitive.

5.3.2.2 The Form column is used to indicate the type of data contained in a parameter (see ¶ 4.2 for definitions).

5.3.2.3 The Description column in the Service Parameter Dictionary table describes the meaning of the parameter, the values it can take on, and any inter-relationships with other parameters.

5.3.2.4 To prevent the definition of numerous parameters named “XxxList,” this document adopts the convention of referring to the list as “(List of) Xxx.” In this case, the definition of the variable Xxx will be given, not of the list. The term “list” indicates a collection (or set) of zero or more items of the same data type. Where a list is used in both the request and the response, the list order in the request is retained in the response. A list must contain at least one element unless zero elements are specifically allowed.

5.3.3 *Service Message Definition* — A service message definition table defines the parameters used in a service, as shown in the following table:

<i>Parameter</i>	<i>Req/Ind</i>	<i>Rsp/Cnf</i>	<i>Description</i>
Parameter X	(see below)	(see below)	A description of the service.

5.3.3.1 The columns labeled Req/Ind and Rsp/Cnf link the parameters to the direction of the message. The message sent by the initiator is called the “Request.” The receiver terms this message the “Indication” or the request. The receiver may then send a “Response,” which the original sender terms the “Confirmation.”

5.3.3.2 The following codes appear in the Req/Ind and Rsp/Cnf columns and are used in the definition of the parameters (e.g., how each parameter is used in each direction):

M	Mandatory Parameter — Must be given a valid value.
C	Conditional Parameter — May be defined in some circumstances and undefined in others. Whether a value is given may be completely optional or may depend on the value of the other parameter.
U	User-Defined Parameter
-	The parameter is not used.
=	(for response only) Indicates that the value of this parameter in the response must match that in the primary (if defined).

6 Overview

6.1 Processing management is concerned with the processing of material by a processing resource. Its principle function is to ensure that material delivered to the processing resource is processed with the correct recipe. It defines the services needed by a supervisor (service-user) to initiate and track processing of a particular material. It also defines commands which affect the processing operation.

6.1.1 The processing resource is the entity which adds manufacturing value to the material. It may take several forms, including the processing element of a cluster tool process module or the entity managing processing for a complete stand-alone equipment. The processing agent is considered to be the provider of the processing services.

6.1.2 Process management allows for pre-conditioning before material arrival and post-conditioning after material departure. A simple tuning mechanism provides support for limited feedforward and feedback control. The tuning, applied at process initiation, sets recipe variable parameters.

6.1.3 The services are fully defined in terms of the functionality provided by the processing agent (service-provider) and, as such, do not dictate the architecture of the supervisor (service-user).

6.1.4 This standard describes the concepts and processing model on which the communications are based, followed by the detailed behavioral model used. It then describes the standard object attributes and message services in detail.

6.2 *Compliance* — Compliance to this standard includes adherence to all stated requirements in this document where implemented. Standard services are to be used where related functionality is required. This includes defined message services and state models.

6.2.1 Some capabilities are not required to be supported for compliance, such as queuing, multiple concurrent jobs, material groups, manual start, pause/resume, and tuning. Required capabilities are indicated throughout the document and are also listed in the Fundamental Requirements section.

6.2.2 A processing agent shall provide the fundamental requirements, plus the set of optional services, appropriate to achieve effective processing management for the particular hardware architecture and automated processing requirements.

7 Concepts

7.1 *Material Processing Model* — Processing management ensures that the appropriate processing is applied to a particular material by a processing resource through the definition of a process job. The process job provides a widely applicable supervisory control capability for automated processing of material in equipment, irrespective of the particular process being used.

7.1.1 This standard assumes that, given the material and the recipe specification, the processing resource is capable of independently achieving the required processing objectives.

7.1.2 Processing management does not provide services for material movement, but the service-provider does need to coordinate its activities with regard to the receiving and sending of material, thereby maintaining system integrity.

7.2 *Process Job* — The process job is a dynamic object specified by the process supervisor (service-user) to effect material processing by the processing resource. The high-level job contains all the information required by the processing resource to achieve processing of the material, once it arrives, without further intervention by the supervisor.

7.2.1 The process job encompasses up to four sequential phases:

- processing resource pre-conditioning before material arrival,
- material and processing resource preparation for processing,
- material processing, and
- processing resource post-conditioning after material departure.

7.2.2 The material processing phase is the only phase in which the material is altered and is the only required phase.

7.2.3 This standard specifies services for the creation, control (pausing, aborting, etc.) and tracking of the process job. It does not define the low-level control of processing since this is application-dependent. The processing resource performing a process job is responsible for doing whatever is appropriate to achieve its processing objectives, as specified by the recipe and tuning parameters.

7.2.4 The material specified in a process job may be the actual single material elements to be processed or a container, such as in the case of a cassette of wafers.

7.2.5 The process job lifecycle may extend beyond the active processing of the material. It may exist from before material arrival, through setup and processing, and until after material departure. This allows for material processing-related pre-conditioning of the processing resource before the material is received and for processing resource post-conditioning (e.g., cleanup) after material is sent. Pre-conditioning and post-conditioning support is not a fundamental requirement.

7.2.6 The processing resource may provide process job queuing in order to offer flexibility in systems where work is pre-scheduled or the order of material arrival is unknown. Queuing is the acceptance of multiple process jobs in advance of performing the processing activities. Queuing is generally needed to support more complex systems requiring concurrent and consecutive jobs (see below). The jobs are listed in the queue in the order created. Execution order may be significant, such as consecutive jobs on the same material. Queuing is not a fundamental requirement.

7.3 *Relation to Material Movement* — Processing Management does not provide services for receiving material into the processing agent domain for processing or sending the material away after processing is complete.

7.3.1 Processing depends on the presence of the material, and material departure depends on process completion. There is also an interdependency requiring synchronization with material movement if processing resource pre-conditioning and post-conditioning are applied. The equipment is responsible for maintaining integrity between material transfer and processing.

7.3.2 Material movement management is outside the scope of this standard but may be achieved using applicable SEMI standards.

7.4 *Processing Description* — The description of the processing to be applied by the Process Job is crucial. The description may be supplied in the form of a Process Recipe (see SEMI E42) or a Process Program (see SEMI E30). This specification will define messages referencing only Process Recipes. Where there are special considerations for using Process Programs instead of a Process Recipe, they will be noted.

7.4.1 The process job includes a processing description (a recipe or process program) identifier that shall be unique within the domain of the processing agent. The type and content of the processing description must be appropriate for the processing resource and the type of material.

7.4.2 Creation and management of recipes and process programs is outside the scope of this standard.

7.5 *Process Tuning* — Feedforward and feedback control between process steps is becoming increasingly important for process tuning in stabilizing processes, such as those which lack in-situ metrology, and demand increasing yields (or performance). The process tuning requirements differ considerably with application, and there is little consensus on any particular method. It is not the intention of this standard to provide comprehensive support for process tuning, but rather to provide a simple mechanism which may be extended. Support for recipe tuning is not a fundamental requirement.

7.5.1 Processing management provides a mechanism for specification of the type of recipe method to be applied. Two methods are defined in the standard: RecipeID only, and RecipeID and Variables. User-defined methods may also be used, requiring all communicating entities to have a common understanding of the particular definition and application requirements.

7.5.2 The RecipeID only method accepts the identifier of the recipe to be applied but no additional tuning parameters. The application recipe body is not precluded from defining any application tuning, but there is no standardized support.

7.5.3 The RecipeID and Variables method provides a simple process tuning mechanism at process job creation to support limited run to run feedforward and feedback control. It defines the VariableTuning method, which supplies a list of variable names and values in the process job create. This sets variables defined in the recipe. Each variable name shall be one of the exposed variable definitions supported in recipe management, and its value shall fall within the range specified in the variable definition.

7.5.4 Recipe parameter names (RecipeVarName) shall be specified using the nomenclature defined for ‘Object Specification’ (ObjSpec) in SEMI E39 (Object Services Standard) within the scope of SEMI E40 (Note: see OBJSPEC in SEMI E5). This use of the ObjSpec nomenclature is required to unambiguously identify parameters within some complex recipes (e.g., cluster tool process recipes). If the specification of a recipe parameter is unambiguous, then the form of the ObjSpec may be simplified to just the parameter name. Using the ObjSpec nomenclature for SEMI E40 recipe parameter names requires an object model to describe equipment recipe structure.

7.5.5 Figure 1 shows an example of a typical set of cluster tool hierarchical recipe relationships. A processing parameter might be specified through Sequence Step AB, Process Recipe BB, Process Step CB, to Process Parameter DB. In this example, the object specifier for Process Parameter DB would be:

“Sequence:Erma>SequenceStep:AB>ProcessRecipe:
BB>ProcessStep:CB>ProcessParameter:DB>”

or

“Erma>AB>BB>CB>DB>”

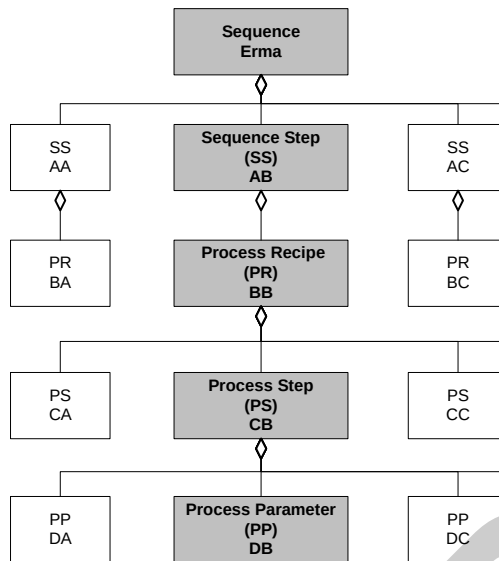


Figure 1
Cluster Tool Sequence Recipe Hierarchical Relationships

7.6 Processing Material Groups — Many equipment architectures require concurrent processing of groups of material. A single process job can control a group of material, with certain restrictions. All material in the group needs to be of the same type and processed identically. The processing of each material should be dependent on the arrival of the group. That is, the full material group shall be received by the equipment before processing begins and may not depart until all processing is complete. This ensures that the process job remains a simple logical control mechanism while maintaining robust control, good coordination with material movement, and effective material data tracking.

7.6.1 Two common examples of allowable material group processing are:

- an equipment receiving a cassette of wafers to be processed identically. All the wafers in the group are received together, in the cassette. The processing of the wafers may or may not be simultaneous but can only proceed once the cassette has arrived. All wafer processing shall complete before the cassette may be removed. Note that this may alternatively be specified as a simple process job with the cassette as the material; and
- a cluster tool batch process module processing wafers together. Wafers are received singly. Processing of the wafer group in the batch chamber begins when all the wafers specified in the process job have arrived. Wafers are sent after processing is complete.

7.7 Concurrent Process Jobs — Concurrent process jobs is the situation where multiple jobs are active (not queued) at the same time. A single process job is not always appropriate for concurrent processing of multiple material. In such cases, multiple concurrent jobs shall be supported by the processing resource. Support of concurrent process jobs is not a fundamental requirement.

7.7.1 An example of concurrent wafer processing which may not be achieved with a single process job is a carousel type cluster tool process module. The processing of a single wafer is not dependent on the arrival of the group. In this case, concurrent process jobs would be created, one for each wafer, even if they were to be processed identically. This allows effective control and tracking of the process jobs.

7.7.2 Processing management considers concurrent process jobs to be logically independent of one other. They are distinguished by unique job identifiers. Concurrent jobs may not apply to the same material because the processing resource may not have more than one active process job associated with a particular material.

7.7.3 There may be interdependencies between concurrent jobs due to resource availability and equipment hardware architecture.