# SEMI E54.1-0316 (Reapproved 0124)
# SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMON DEVICE MODEL

This Standard was technically approved by the Information & Control Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on December 14, 2023. Available at www.semiviews.org and www.semi.org in January 2024; originally published February 1999; previously published March 2016.

**NOTICE:** The section symbol (§) is used to reference a numbered section, paragraph, or header and all subordinate paragraphs, headers, and embedded material (i.e., EXCEPTIONs, lists) therein. The paragraph symbol (¶) is used to reference a particular numbered paragraph and embedded material therein. When duplicated (i.e., §§ and ¶¶) the symbols are used to reference more than one section or paragraph, respectively.

**NOTICE**: This Document was previously designated SEMI E54. Because this Document is part of a suite of Standards, its designation has been reassigned for ease of reference.

## 1 Purpose

1.1 This Standard defines a model comprised of device objects which are common to all devices on a semiconductor equipment Sensor/Actuator Communications Network.

## 2 Scope

2.1 This Document describes common device structure and behavior (i.e., the minimum data structure and behavior all devices must support to operate on the network). These devices may range from simple sensors and actuators through hosts, masters, or controllers.

2.2 The model specified in this Document is used in conjunction with a sensor/actuator network specific device model which describes the data structure and behavior characteristic of the specific device. Together, these two models are sufficient to completely describe a device as it appears from the network interface.

2.3 This Standard, together with a Sensor/Actuator Network Interoperability Guideline, a Sensor/Actuator Network Communication Specification, and one or more specific Device Model Specifications form a complete interoperability Specification. The Sensor/Actuator Network Document architecture is shown in Figure 1.

2.4 To comply with this Standard, a device must implement and support instances of the objects, object attributes, object services, and object behaviors identified in this Document, unless explicitly stated otherwise.

**NOTICE:** SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the Documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

## 3 Limitations

3.1 This Standard is a companion to a suite of Sensor/Actuator Network Communication Network Specifications. Therefore, using portions of this Standard that relate to network communications necessarily requires an understanding of the associated Network Specification.

3.2 As this Specification is a standard for a Common Device Model (CDM), it does not contain any information (e.g., object, attribute, services, or behavioral descriptions) that relates to a specific device or device type.

3.3 While the standards depicted in Figure 1 are sufficient to completely describe a device as it appears from the network, they do not fully describe behavior of the device which is not visible from the network. Some of the behavior detail within standard objects is intentionally left to the manufacturer to define. This allows flexibility for product differentiation and creates room for technology evolution. Manufacturer-specific objects may be defined by the manufacturer as appropriate, but are by definition outside the scope of this Standard.

3.4 This Standard is compatible, but not compliant, with SEMI E39. This means that although this Standard does not require compliance with SEMI E39, it is extensible such that implementations may be developed that are fully compliant with both Standards. Note that the concepts and terminology of this Standard are compatible with those of



Documents Document Sharing Hub

Full document details: SEMI E54.1-0316-0124R

SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and thus are not applied to the CDM.

## 4 Referenced Standards and Documents

4.1 *SEMI Standards and Safety Guidelines*

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

SEMI E39 — Specification for Object Services: Concepts, Behavior, and Services

SEMI E148 — Specification for Time Synchronization and Definition of the TS-Clock Object

4.2 *ISO® Standards*[1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection

4.3 *Other Documents*

David Harel, "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming 8, 1987

James Rumbaugh, Michael Blaha, William Premerlani, Frederic Eddy, William Lorensen, Object-Oriented Modeling and Design, Englewood Cliffs, New Jersey: Prentice-Hall, 1991

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

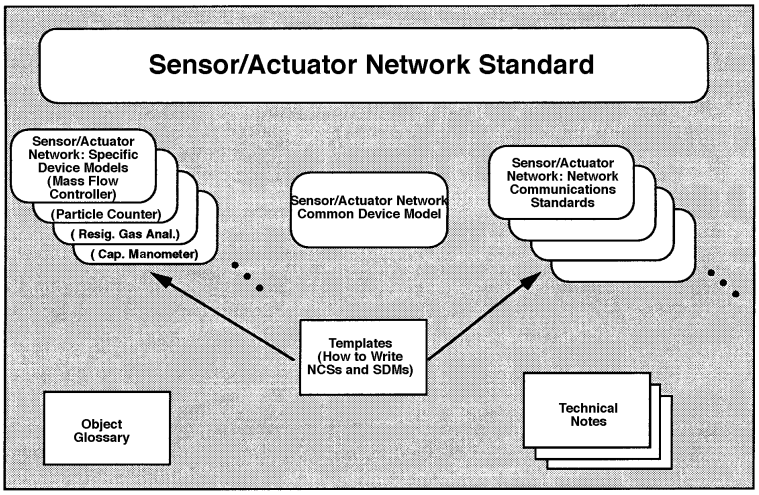



**Figure 1**
**Sensor/Actuator Network-Related Documents**

## 5 Terminology

5.1 Terminology may be reproduced here which is defined in other SEMI Standards.

5.2 *Definitions*

5.2.1 *attribute* — externally visible information concerning an object.

5.2.2 *behavior* — a specification of how an object acts. Actions result from different events the object detects, such as receiving service request, detecting internal faults, or elapsing timers.

[1] International Organization for Standardization, ISO Central Secretariat, Chemin de Blandonnet 8, CP 401-1214 Vernier, Geneva, Switzerland; Telephone: +41.22.749.01.11, Fax: +41.22.733.34.30, http://www.iso.org. ISO trademark is owned by International Organization for Standardization.



Documents
Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

5.2.3 *class* — a specific type or classification of objects.

5.2.4 *device* — a tangible thing consisting of: (1) at least one sensor and/or actuator and/or controller, (2) a communications controller which supports a single point of access to a network as specified in this Document, and (3) interconnection and management hardware and software that provides for the consolidation of (1) and (2) into a system that has the capability to comply with the specification detailed in this Document. Examples of devices are given in Figure 2.

5.2.5 *device model* — an abstraction of a device for the purpose of understanding it before building it or using it.

5.2.6 *embedded object* — an embedded object is similar in functionality or purpose to the object in which it is embedded, or supports the functionality of the object in which it is embedded. The embedding construct is utilized solely for purposes of documentation structure and understanding. As such, it does not imply any direct relationship, inheritance, similarity in structure, or connectivity in addressing scheme between the embedded object and the object in which it is embedded.

5.2.7 *instance* — a specific and real occurrence of an object.

5.2.8 *manufacturer* — in the context of this Document, this refers to the manufacturer of the device.

5.2.9 *object* — an entity with a specific set of data and behaviors. Objects may be physical or conceptual. An object may be described in terms of its attributes, services it provides, and behavior it exhibits.

5.2.10 *service* — a function offered or supported by an object. A service consists of a sequence of service primitives, each described by a list of parameters. A service excludes definition of message structure and protocol.

5.2.11 *state diagram* — a means of representing state transitions, where the boxes represent states and the arrows represent transitions between states.

5.3 *Data Type Terminology* — unless otherwise noted, data type terminology defined in SEMI E39 will be used in this Document. The following terminology will also be used:

5.3.1 *Boolean (BOOL)* — a binary bit representing 0 and 1 corresponding to FALSE and TRUE or DISABLE and ENABLE respectively.

5.3.2 *byte* — a string of eight adjacent bits, interpreted as a unit and often representing a character.

5.3.3 *character* — a text symbol, letter, digit, or mark used to represent, control, or organize information that is one byte in length.

5.3.4 *character string* — a text string.

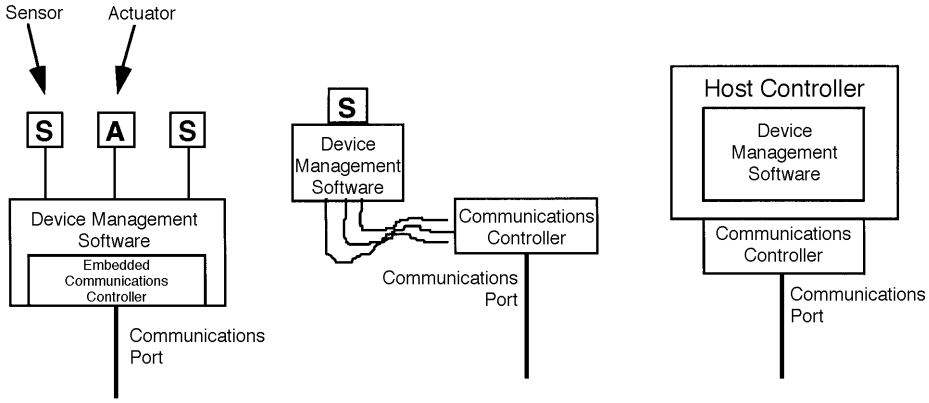



**Figure 2**
**Examples of Devices**



Documents Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

5.3.5 *data type* — an unsigned short integer formatted as an enumerated byte to specify attribute data format. The intended use of this attribute type is in cases where an attribute, or set of attributes, may be defined, allowing for more than one level of support (e.g., INT or REAL). The following values are defined:

- 0 = INT
- 1 = REAL
- 2 = USINT
- 3 = SINT
- 4 = DINT
- 5 = LINT
- 6 = UINT
- 7 = UDINT
- 8 = ULINT
- 9 = LREAL
- 10–99 = Reserved for CDM
- 100–199 = Reserved for SDMs
- 200–255 = Manufacturer-Specified

5.3.6 *data units* — an unsigned integer enumerated to specify attribute data units. The intended use of this attribute type is in cases where an attribute, or set of attributes, may be defined, allowing for more than one unit's context. The values are defined in Appendix 1 of this Document.

5.3.7 *date* — a data structure of four bytes used to represent a calendar date. Table 1 defines the format of the date data type.

**Table 1 Date Format**

| *Data #* | *Description* | *Range* |
|---|---|---|
| 0–1 | Year | Unsigned Integer <65,536 |
| 2 | Month | Unsigned Integer (range of 1–12) |
| 3 | Day | Unsigned Integer (range of 1–31) |

5.3.8 *Double Integer (DINT)* — an integer, four bytes long, in the range $-2^{31}$ to $2^{31}-1$.

5.3.9 *enumerated byte* — a byte with assigned meaning to the values 0 to 255. May take on one of a limited set of possible values.

5.3.10 *full scale range* — the defined 100% value of an attribute in its assigned units. This value is not necessarily the maximum value for the attribute. As an example, the indicated flow attribute value may attain 120% of the full scale range.

5.3.11 *Last Valid Value (LVV)* — the most recent value successfully assigned to an attribute.

5.3.12 *Long Integer (LINT)* — an integer, eight bytes long, in the range $-2^{63}$ to $2^{63}-1$.

5.3.13 *Long Real (LREAL)* — a double floating point number, 8 bytes long, as defined by IEEE 754.

5.3.14 *nibble* — a string of four adjacent binary bits.

5.3.15 *null character* — a byte with a value of zero.

5.3.16 *Real (REAL)* — a floating point number, 4 bytes long, as defined by IEEE 754.



Documents
Document Sharing Hub

Full document details: SEMI E54.1-0316-0124R

5.3.17 *Short Integer (SINT)* — an integer, one byte long, in the range −128 to 127.

5.3.18 *Signed Integer (INT)* — an integer, 2 bytes long, in the range −32768 to 32767.

5.3.19 *text string* — a string of one byte characters.

5.3.20 *Unsigned Double Integer (UDINT)* — an unsigned integer, four bytes long, in the range 0 to $2^{32}-1$.

5.3.21 *Unsigned Integer (UINT)* — an integer, 2 bytes long, in the range 0 to 65535.

5.3.22 *Unsigned Long Integer (ULINT)* — an unsigned integer, eight bytes long, in the range 0 to $2^{64}-1$.

5.3.23 *Unsigned Short Integer (USINT)* — an integer, 1 byte long, in the range 0 to 255.

## 6 Conventions

6.1 *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.

6.1.1 Transition tables (referred to in this Document as 'state transition matrices') are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. Each transition table 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.

6.2 *OMT Object Information Model* — The object models are presented using the Object Modeling Technique developed by Rumbaugh, James, *et al*, in "Object-Oriented Modeling and Design," Prentice Hall, Englewood Cliffs, NJ, ©1991. Overviews of this notation are provided in an appendix of SEMI E39.

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

| *Attribute* | *Definition* | *Access* | *Rqmt* | *Form* |
|---|---|---|---|---|
| The formal text name of the attribute. | Description of the information contained. | RO or RW | Y or N | (refer to below) |

6.3.1 The access column uses RO (Read Only) or RW (Read and Write) to indicate the access that users of Object Services have to the attribute. Note that, in this Document, the access column is used only to indicate user access via the network; no indication is made as to local access as this level of specification is considered to be implementation-specific. Thus, in the context of this Document, a RW attribute is a network-settable attribute (i.e., an attribute whose value can be altered from the network), while a RO attribute is a non-network-settable attribute. 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. The Form column is used to indicate the format of the attribute.

6.4 *Service Message Representation*

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

| *Service* | *Type* | *Description* |
|---|---|---|
| Message name | N or R | The intent of the service. |



Documents Document Sharing Hub

Full document details: SEMI E54.1-0316-0124R

6.4.1.1 Type can be either Notification (N) or Request (R). Notification messages are initiated by the service provider. No response is expected. Request messages are initiated by the service user. Request messages ask for data or for an operation to be performed. Request messages expect a specific response (no presumption on the message content).

6.4.2 *Service Parameter Dictionary* — Each parameter should relate to either attributes from the object model or events of the dynamic model (Harel State Chart). All parameters to the services are listed in a single table or dictionary. The column headings for this parameter dictionary are as follows:

| *Parameter* | *Form* | *Description* |
|---|---|---|
| Parameter X | Data type | A parameter |

6.4.2.1 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 the contents of the parameter. The form column is used to indicate the type of data contained in the parameter and the possible values it may take on. The description column describes the meaning of the parameter and interrelationships with other parameters.

6.4.2.2 *Service Message Definition* — There is a table showing the parameter detail for each service in which parameters are explicitly specified. The column headings for the service detail are as follows:

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Description* |
|---|---|---|---|
| Parameter X | (see below) | (see below) | A description of the service. |

6.4.2.2.1 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' (Req). The receiver terms the message the 'Indication' (Ind). The receiver may then send a 'Response' (Rsp), which the original sender terms the 'Confirmation' (Cnf).

6.4.2.2.2 The request (Req/Ind) and response (Rsp/Cnf) entries can take on the following values:

| | |
|---|---|
| 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. |
| = | The entries M and C in the response can be modified with (=) to indicate that the value in the response must match the request. |

## 7 Device High Level Structure

7.1 The high level object view of a device aggregation is shown in Figure 3. A device is depicted as consisting of a sensor/actuator/controller (SAC) object, a device manager (DM) object, and at least one active element object (i.e., sensor or actuator or controller object). Each of these objects by definition has attributes, services, and behavior. Note that the 'Device' object is depicted in Figure 3 only for purposes of illustrating a high level view of the device and its component objects; in the context of this Document, the 'Device' object is not addressable, does not have addressable attributes or accessible services, and has no behavior defined.

7.1.1 This Document defines in detail only the DM object. The SAC, Sensor, Actuator, and Controller objects are defined here only in terms of characteristics common to all devices. A complete definition of a SAC object includes an appropriate sensor/actuator network specific device model. This companion model could also complete the definition of sensor, actuator, and controller objects as appropriate to specify a device model.

7.2 *General Requirements* — Objects are defined in terms of their object name and instance identifier. Identifiers for all objects described in this Document are summarized in Table 2. Note that these identifiers may be used for remote



Documents
Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

interrogation of an object instance via the sensor/actuator network (refer to appropriate Sensor/Actuator Communications Specification). Also note that, in Table 2, many of the objects specified in this Document have exactly one instantiation per device.

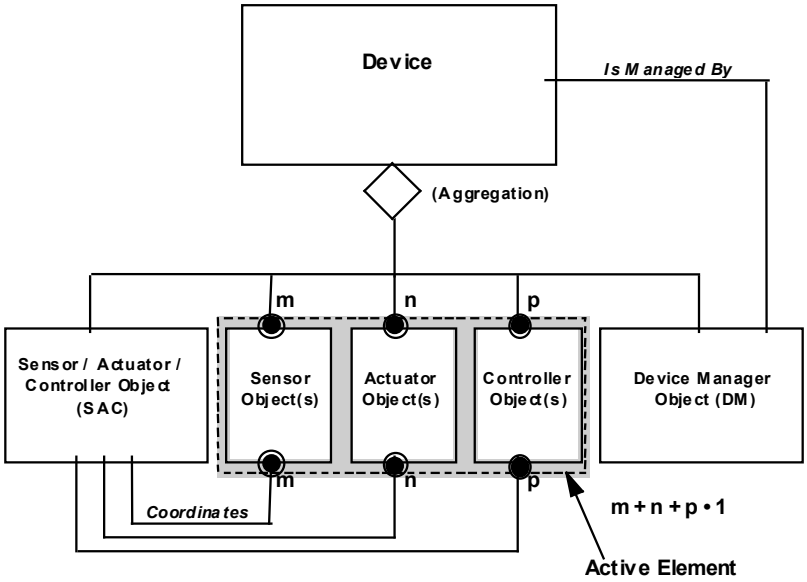



**Figure 3**
**High-Level Object View of a Device**

**Table 2 Device Objects and Identifiers**

| *Object Name* | *Object Identifier (Tag)* | *Support Required in a Device* | *Comment* |
|---|---|---|---|
| Sensor/Actuator/Controller | SacI0[#1] | Yes | Only one instance per device allowed. |
| Sensor | SenIn[#2] | No | Zero or more instances per device allowed.[#3] |
| Actuator | ActIn[#2] | No | Zero or more instances per device allowed.[#3] |
| Controller | CntIn[#2] | No | Zero or more instances per device allowed.[#3] |
| Device Manager | DmI0[#1] | Yes | Only one instance per device allowed. |

#1 Only one object instance per device; identifier uses 'I0' to specify 'instance zero.'

#2 'In' is used to indicate the instance number of the object; 'n' is a non-negative integer.

#3 The specification of the number of sensor, actuator, and/or controller object instances allowed per device may be further constrained by the appropriate Sensor/Actuator Network Specification.

7.2.1 The objects described in this Document collectively define a device's capabilities, including how it has been configured for network interoperability. The information in the attributes of these object instances must be accessible over the network and stored at the device.

7.2.2 Character strings described in this Document have a prescribed maximum length. If the contents of the string are shorter than the prescribed length, the string must be terminated with the null character: a byte whose value is 00h. Note that this Specification of null termination does not indicate that a sensor/actuator network protocol implementation communicating a character string over a network must send the null termination; the presentation of character string data over a network is sensor/actuator network communication protocol-specific.



Documents Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

7.3 *Sensor/Actuator/Controller (SAC) Object* — The SAC object is the device component responsible for coordinating the interaction of the device with the sensory/actuation/control environment. A view of the SAC object is shown in Figure 4. The SAC object coordinates operation of one or more sensor, actuator, and/or control object instances that collectively form the sensory/actuation/control portion of the device, so as to enable desired device behavior. For example, it could coordinate the operation of the device sensor, actuator, and/or control elements to enable device level data reporting or actuation, alarming detection and servicing, status reporting, device self-testing, device shutdown, etc. The number of sensor, actuator, and/or controller object instances allowed per device may be specified by the appropriate sensor/actuator network specific device model. An operating device shall contain exactly one instance of a SAC object.

7.3.1 The SAC object has embedded in it other objects that address specific tasks associated with coordinating the interaction of the device with the sensor/actuation/control environment. These objects are listed in Table 3. Note that although only one instance of the SAC object is allowed per device, a device can have multiple instances of embedded objects. These objects are described in § 7.3.5.

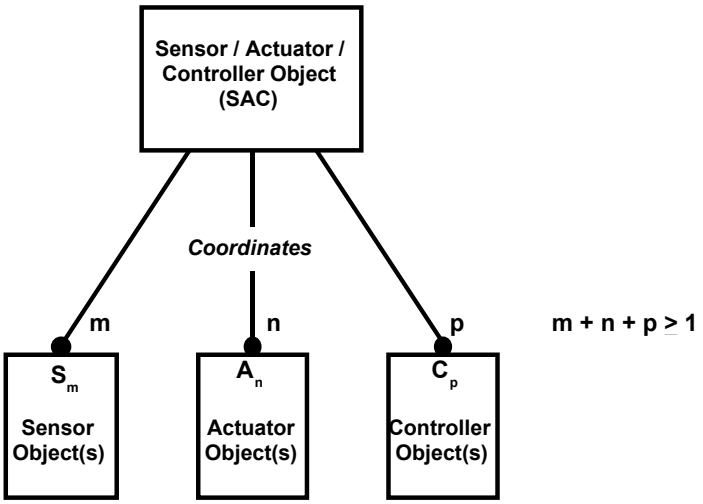



**Figure 4**
**Detailed View of the Sensor/Actuator/Controller Object**

**Table 3 Objects Embedded in SAC Object and Identifiers**

| *Object Name* | *Object Identifier (tag)* | *Support Required in a Device* | *Comment* |
|---|---|---|---|
| Assembly | AsmIn[#1] | No | Zero or more instances per device allowed.[#2] |
| Local Link | LnkIn[#1] | No | Zero or more instances per device allowed.[#2] |

#1 'In' is used to indicate the instance number of the object; 'n' is a non-negative integer.

#2 The specification of the number of Assembly or Local Link object instances allowed per device may be further constrained by the appropriate Sensor/Actuator Network Specification.

7.3.2 *SAC Object Attributes* — All required and 'common optional' SAC object attributes are listed in Table 4. Note that many of the attributes are in text 'human-readable' forms. Information provided in this table includes, for each attribute: (1) an attribute identifier tag; (2) an indication of user access (via the network) to the attribute, (i.e., an indication of whether the attribute is network-settable); and (3) an attribute type. Required attributes must be supported in all SAC object instantiations. Optional attributes may or may not be supported; a further specification of the requirement of support for these optional attributes can be found in § 7.3.4 and in an appropriate sensor/actuator network specific device model specification.



Documents
Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

**Table 4 SAC Object Attributes**

| *Attribute Name* | *Attribute Identifier* | *Access Network* | *Required* | *Form* |
|---|---|---|---|---|
| Last Calibration Date | SacA1 | RW | No | date |
| Next Calibration Date | SacA2 | RW | No | date |
| Expiration Timer | SacA3 | RW | No | signed integer |
| Expiration Warning Enable | SacA4 | RW | No[#1] | Boolean |
| Run Hours | SacA5 | R | No | unsigned integer |
| Last Calibration Timestamp | SacA6 | RW | No | Date and Time |
| Next Calibration Timestamp | SacA7 | RW | No | Date and Time |
| Data Timestamp | SacA8 | RW | No | Date and Time |
| Reserved | SacA9–SacA64 | - | - | Reserved for future expansion. |

#1 The Expiration Warning Enable attribute is required if the Expiration Timer attribute is supported.

7.3.2.1 *Last Calibration Date (Optional)* — An attribute which identifies the date the device was last calibrated. The attribute is formatted as a date defined in § 5.2.

7.3.2.2 *Next Calibration Date (Optional)* — An attribute which identifies the date the device is scheduled for the next calibration. The attribute is formatted as a date defined in § 5.2.

7.3.2.3 *Expiration Timer (Optional)* — An attribute which identifies the number of run hours remaining until the next recommended calibration. The attribute is an unsigned integer with a resolution of 1 hour.

7.3.2.4 *Expiration Warning Enable (Optional)* — An attribute which specifies whether the calibration expiration timer will set a specific warning status bit in the exception status attribute of the DM object instance (specifically the 'calibration expiration' exception bit—refer to ¶ 7.4.1.15).

7.3.2.4.1 The expiration warning enable attribute is Boolean and can take on one of the following values:

- 0 = Disable
- 1 = Enable

7.3.2.5 *Run Hours (Optional)* — An attribute which identifies the number of hours that the device has been powered ON. The attribute is an unsigned integer with a resolution of 1 hour.

7.3.2.6 *Last Calibration Timestamp* (Optional) — An attribute which identifies the date and time the device was last calibrated. The attribute is formatted as a Date and Time type defined in § 7.4.1.23.

7.3.2.7 *Next Calibration Timestamp* (Optional) — An attribute which identifies the date and time the device is scheduled for the next calibration. The attribute is formatted as a Date and Time type defined in § 7.4.1.23.

7.3.2.8 *Data Timestamp* (Optional) — An attribute which identifies the time at which the data value was generated. The attribute is formatted as a Date and Time type defined in § 7.4.1.23.

7.3.2.9 *Initial and Default Values* — The initial and default values for relevant SAC attributes are given in Table 5:

**Table 5 SAC Attribute Initial and Default Values**

| *Attribute* | *Initial Value* | *Default Value* | *Comment* |
|---|---|---|---|
| Last Calibration Date | LVV | 0,0,0[#1] | |
| Next Calibration Date | LVV | 0,0,0 | |
| Expiration Timer | LVV | 0 | |
| Expiration Warning Enable | LVV | 0 | |
| Run Hours | LVV | 0 | |

#1 Corresponding to year, month, and day; refer to § 5.2.



Documents Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

7.3.3 *SAC Object Services* — All SAC object instances must provide the services listed in Table 6 (the SAC object is the 'service provider' of these services). SAC services are identified with a service identifier of the form 'SacSn', where 'n' is a positive decimal number. SAC service identifiers SacS1 through SacS3 are used by this CDM Standard. SAC service identifiers SacS4 through SacS32 are reserved.

**Table 6 SAC Services**

| *Service* | *Service Identifier* | *Required* | *Type* | *Description* |
|---|---|---|---|---|
| Reset | SacS1 | Yes | R | Used to place object in INITIALIZING state. |
| Abort | SacS2 | Yes | R | Used to place object in ABORT state. |
| Recover | SacS3 | Yes | R | Used to move object from ABORT state to RECOVERING state. |
| Get Attribute | SacS4 | Yes | R | Used to read object attribute. |
| Set Attribute | SacS5 | Yes | R | Used to modify object attribute. |
| Operate | SacS6 | Yes | R | Used to move object to the OPERATING state from INITIALIZING or RECOVERING state. |
| Restore Default | SacS7 | No | R | Used to restore object attributes to their default values. |
| Publish Attribute | SacS8 | No | N | Used to proactively report an object attribute value. |
| | <SacS32 | - | - | Reserved for future extensions of the CDM. |

7.3.3.1 *Reset (Service Identifier: SacS1)* — Used to place the SAC object in its INITIALIZED state. The description of this state is device-specific. There are no parameters specified for this service.

7.3.3.2 *Abort (Service Identifier: SacS2)* — Used to place the SAC object in its ABORT state. The description of this state is device-specific. There are no parameters specified for this service.

7.3.3.3 *Recover (Service Identifier: SacS3)* — Used to move the SAC object from an ABORT state to its RECOVERED state. The description of this state is device-specific. There are no parameters specified for this service.

7.3.3.3.1 Additional services and/or service parameters may be provided by a SAC object that are device-specific and/or implementation-specific. These services may be Notification or Request type services (refer to § 6.4.1). A complete definition of additional services and service parameters that are device-specific may be found in an appropriate sensor/actuator network specific device model specification.

7.3.3.4 *Get Attribute* — This service is used to read the value of an object instance attribute over the network. Table 7 describes the parameters specified for this service.

**Table 7 SAC Object Get Attribute Parameter Definition**

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Form* | *Description* |
|---|---|---|---|---|
| Attribute ID | M | C | Network-Specific | Attribute Identifier of the attribute whose value is being requested. |
| Attribute Value | - | M | Context-Specific | Value of the attribute requested. |

7.3.3.5 *Set Attribute* — This service is used to modify the value of an object instance attribute over the network which has been identified as modifiable. Table 8 describes the parameters specified for this service.

**Table 8 SAC Object Set Attribute Parameter Definition**

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Form* | *Description* |
|---|---|---|---|---|
| Attribute ID | M | C | Network-Specific | Attribute Identifier of the attribute whose value is being modified. |



Documents
Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Form* | *Description* |
|---|---|---|---|---|
| Attribute Value | M | C | Context-Specific | Value of the attribute to modify. |

7.3.3.6 *Operate* — This service is used to move the object instance to the OPERATING state from the INITIALIZING or RECOVERING state. There are no parameters specified for this service. Operate requests are issued to all S, A, and C objects.

7.3.3.7 *Restore Default (Optional)* — This service is used to restore attributes of this object instance to their default values. Table 9 describes the parameters specified for this service. For Restore Default Conditions = 2, a Restore Default Condition = 1 request is issued to all S, A, and C objects.

**Table 9 SAC Object Restore Default Parameter Definition**

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Form* | *Description* |
|---|---|---|---|---|
| Restore Conditions | M | C | byte, enumerated | 0 = Restore specific attribute of this object<br>1 = Restore this object<br>2 = Restore all objects of this device |
| Attribute ID | C | C | Network-Specific | Attribute ID of the attribute whose value is being restored. This parameter is only used for Restore Conditions = 0. |

7.3.3.8 *Publish Attribute (Optional)* — Used to proactively communicate a SAC object attribute. Table 10 describes parameters specified for this service.

**Table 10 SAC Publish Attribute Service Message Definition Table#1**

| *Parameter* | *Req/Ind* | *Rsp/Conf#1* | *Description* |
|---|---|---|---|
| Attribute ID | M | - | Attribute Identifier of the attribute whose value is being reported. |
| Attribute Value | M | - | Value of attribute being reported. |

#1 The utilization of a service response message and its format is outside the scope of this Document.

7.3.4 *SAC Instance Behavior* — SAC object instance behavior that is common to all devices is illustrated in Table 5. Associated states and state transitions are described in Tables 11 and 12. The following also applies to SAC object behavior:

7.3.4.1 A SAC object service may be requested internally by the SAC object.

7.3.4.2 Upon SAC object instantiation or upon receipt of a reset request, the SAC responds by entering or staying in its INITIALIZED state.

7.3.4.3 Upon receipt of an abort request, the SAC object responds by entering or staying in its ABORT state.

7.3.4.4 Upon receipt of a recover request, the SAC object responds by transitioning to its RECOVERED state if and only if it is currently in its ABORT state. Otherwise, it responds with an error indication.

7.3.4.5 Additional behavior may be exhibited by a SAC object that is device-specific. A complete definition of this additional SAC object behavior may be found in an appropriate Sensor/Actuator Network Specific Device Model Specification.

7.3.4.6 All SAC services have the same behavior associated with invalid service requests unless otherwise indicated. Common service error responses are listed in Appendix 2.

7.3.4.7 If a Set Attribute request is received over the network for an attribute that is not network-settable, or if the value in the SetAttribute request is beyond the specified range allowed for the value of the specified attribute, the attribute value is not changed and an error service response is generated.



Documents Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

7.3.4.8 When the SAC object instance is INITIALIZED, all attributes are set to their appropriate initial values (as indicated in their associated object instance attribute descriptions). Initial values could correspond to those retrieved from the device's nonvolatile memory and reflect, in large part, the values set in a previous NORMAL OPERATING state.

7.3.4.9 Additionally, whenever the SAC object instance enters the INITIALIZED state, it must issue a Reset service request to all Sensor, Actuator, and Controller object instances. Each object instance must then respond with a Pass or Fail response indication upon completion of its respective INITIALIZING application process. The SAC object instance must coordinate these responses in order to determine whether the device is qualified to enter the OPERATING state. If qualified, an Execute service request is issued to the DM object instance. If not qualified, an exception condition is reported to the DM object instance (refer to § 7.4).

7.3.4.10 When the SAC object instance is recovering from an ABORT state, all attributes are set to their appropriate values (as determined by the associated object instance recovering application process defined by the manufacturer). Additionally, the SAC object instance must issue a Recover service request to all Sensor, Actuator, and Controller object instances. Each object instance must then respond with a Pass or Fail response indication upon completion of its respective recovering application process. The SAC object instance must coordinate these responses in order to determine whether the device is qualified to enter the OPERATING state. If qualified, an Execute service request is issued to the DM object instance. If not qualified, an exception condition is reported to the DM object instance.

7.3.4.11 The SAC object optionally includes an Expiration Timer attribute. The value of this attribute is decremented at a rate described by the 'ExpirationTimer' attribute format definition. Whenever the 'Expiration-Timer' attribute is less than or equal to zero, a warning exception condition exists and is reported to the DM object instance (utilizing the PublishAttribute service) provided the 'ExpirationWarningEnable' attribute is set to Enable. When the 'ExpirationWarningEnable' attribute is set to Disable, the 'ExpirationTimer' attribute continues to decrement; however, a warning exception condition will not be reported to the DM object instance.

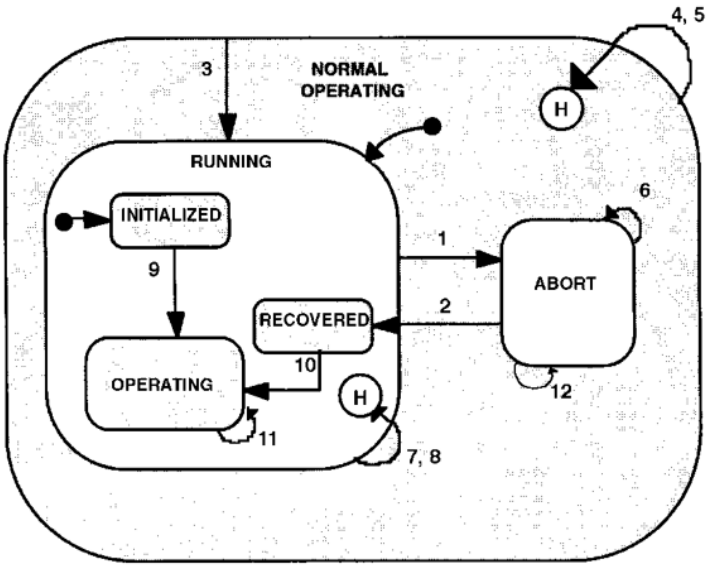
7.3.4.12 The 'ExpirationTimer' attribute will continue to be decremented to values less than zero. Therefore, a negative number indicates an elapsed time past expiration. When this attribute reaches its most negative value, it maintains this value and no longer decrements.

7.3.4.13 The 'ExpirationTimer' is also maintained by the object while in the ABORT state. In the RECOVERING state, the current status of the device is evaluated and any exception conditions that exist are reported to the DM object instance.



Documents
Document Sharing Hub

Full document details: SEMI E54.1-0316-0124R




#1 Only generic behavior is indicated in this chart. A complete state chart necessarily requires the inclusion of device-specific behavior.

#2 Transitions 4–8, 11, and 12 are recursive (i.e., the New State is the same as the Current State [including all nested substates]).

**Figure 5**
**SAC Object Instance Behavior**

**Table 11 SAC Object Instance Behavior State Description**

| *State* | *Description* |
|---|---|
| NORMAL OPERATING | Object instance exists. Object services can be processed. |
| RUNNING | This is the entry substate to NORMAL OPERATING. Object instance is not in its ABORT state. |
| INITIALIZED | This is the entry substate to NORMAL OPERATING and RUNNING. Object instance exists and has been initialized. All attributes set to appropriate initial values (as indicated in an appropriate Sensor/Actuator Network Specific Device Model Specification). |
| RECOVERED | Object instance is in an abort recovered state. |
| OPERATING | This represents the entire collection of substates within RUNNING except INITIALIZED and RECOVERED. The further delineation of this substate and transition into this substate is outside the scope of this Document. |
| ABORT | Object instance is in an aborted state; a detailed description of this state is outside the scope of this Document. |

**Table 12 SAC Object Instance Behavior State Transition Matrix[#1,#2]**

| # | *Current State* | *Trigger* | *New State* | *Action* | *Comment* |
|---|---|---|---|---|---|
| 1 | RUNNING | Abort request. | ABORT | Abort. Abort response. | ABORT state is device-specific. |
| 2 | ABORT | Recover request. | RECOVERED | Recover. Recover response. | RECOVERED state is device-specific. |
| 3 | NORMAL OPERATING | Reset request. | RUNNING | Reset. Reset response. | Valid for all substates of NORMAL OPERATING. Move to RUNNING/INITIALIZED. |
| 4 | NORMAL OPERATING | Invalid service request. | NORMAL OPERATING | Error response. | Valid for all substates of NORMAL OPERATING. |



Documents Document Sharing Hub
Full document details: SEMI E54.1-0316-0124R

| # | *Current State* | *Trigger* | *New State* | *Action* | *Comment* |
|---|---|---|---|---|---|
| 5 | NORMAL OPERATING | GetAttribute, SetAttribute, or RestoreDefault request. | NORMAL OPERATING | Get or Set appropriate attribute or restore defaults. | Valid for all substrates of NORMAL OPERATING. |
| 6 | ABORT | Abort request, Operate request. | ABORT | Error response. | Invalid request in this state. |
| 7 | RUNNING | Recover request. | RUNNING | Error response. | Recover service can only be invoked while in ABORT state. |
| 8 | RUNNING | PublishAttribute. | RUNNING | Attribute identified by attribute ID in PublishAttribute service is published. | Notification service. |
| 9 | INITIALIZED | Operate request or Device-specific. | OPERATING | Device-specific. | Behavior associated with this transition may be further specified in an appropriate Sensor/Actuator Network Specific Device Model Specification. |
| 10 | RECOVERED | Operate request or Device-specific. | OPERATING | Device-specific. | Behavior associated with this transition may be further specified in an appropriate Sensor/Actuator Network Specific Device Model Specification. |
| 11 | OPERATING | Operate request. | OPERATING | Error response. | Invalid request in OPERATING state. |
| 12 | ABORT | PublishAttribute. | ABORT | None. | Attributes cannot be published while in ABORT state. |

#1 Transitions 4–8, 11, and 12 are recursive (i.e., the New State is the same as the Current State [including all nested substates]).

#2 Only generic behavior is indicated in this table.

7.3.5 *Objects Embedded in SAC Object* — The object types that are embedded in the SAC object are listed in Table 3. The structure and behavior of instances of these object types are detailed in the remainder of this section.

7.3.5.1 *Assembly Object* — Assembly object instances may be used to provide a mechanism for grouping more than one attribute from one or more object instances in a device into a single data structure for communication over the network.

7.3.5.1.1 *Assembly Object Attributes* — All required and 'common optional' Assembly object attributes are listed in Table 13.

**Table 13 Assembly Object Attributes**

| *Attribute Name* | *Attribute Identifier* | *Access Network* | *Required* | *Form* |
|---|---|---|---|---|
| Data | AsmA1 | Context-Specific | Yes | Context-Specific |

7.3.5.1.1.1 *Data* — An attribute which contains the assembled data. The structure of this data is context-specific.

7.3.5.1.2 *Assembly Object Services* — The Assembly object instance provides two services: GetAttribute and SetAttribute.

7.3.5.1.2.1 *Get Attribute* — This service is used to read the value of an object instance attribute over the network. Table 14 describes the parameters specified for this service.



Documents
Document Sharing Hub

Full document details: SEMI E54.1-0316-0124R

**Table 14 Assembly Object Get Attribute Parameter Definition**

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Form* | *Description* |
|---|---|---|---|---|
| Attribute ID | M | C | Network-Specific | Attribute Identifier of the attribute whose value is being requested. |
| Attribute Value | - | M | Context-Specific | Value of the attribute requested. |

7.3.5.1.2.2 *Set Attribute* — This service is used to modify the value of an object instance attribute over the network which has been identified as modifiable. Table 15 describes the parameters specified for this service.

**Table 15 Assembly Object Set Attribute Parameter Definition**

| *Parameter* | *Req/Ind* | *Rsp/Cnf* | *Form* | *Description* |
|---|---|---|---|---|
| Attribute ID | M | C, = | Network-Specific | Attribute Identifier of attribute whose value is to be modified. Inclusion of Attribute in Rsp/Conf. is optional. |
| Attribute Value | M | C | Context-Specific | Value to which the attribute is to be modified. Inclusion of Attribute in Rsp/Conf. is optional. Conditions under which the value may differ from that supplied in the associated Req/Ind. message (e.g., negative response) are not defined in this Document. |

7.3.5.1.2.3 Additional services and/or service parameters may be provided by an Assembly object instance that are device-specific and/or implementation-specific. These services may be Notification or Request type services (refer to § 6.4.1). A complete definition of additional services and service parameters that are device-specific may be found in an appropriate Sensor/Actuator Network Specific Device Model Specification.

7.3.5.1.3 *Assembly Object Behavior* — The behavior exhibited by the Assembly object instance is in support of the GetAttribute and SetAttribute services. That is, when a GetAttribute service request is received for the attribute 'Data,' the response contains the 'Data' attribute value. Similarly, when a SetAttribute service request is received for the attribute 'Data,' the value of 'Data' is modified, if indeed the attribute is modifiable, and a successful service response is returned; otherwise, a service not successful is returned.

7.3.5.2 *Local Link Object* — Local Link object instances may be used to 'link' an attribute of one object instance to an attribute of another object instance.

7.3.5.2.1 *Local Link Object Attributes* — All required and 'common optional' Local Link object attributes are listed in Table 16.

**Table 16 Local Link Object Attributes**

| *Attribute Name* | *Attribute Identifier* | *Access Network* | *Required* | *Form* |
|---|---|---|---|---|
| Source Object Class | LnkA1 | Context-Specific | N | Context-Specific |
| Source Object Instance | LnkA2 | Context-Specific | N | Context-Specific |
| Source Object Attribute | LnkA3 | Context-Specific | Y | Context-Specific |
| Destination Object Class | LnkA4 | Context-Specific | N | Context-Specific |
| Destination Object Instance | LnkA5 | Context-Specific | N | Context-Specific |
| Destination Object Attribute | LnkA6 | Context-Specific | Y | Context-Specific |
| Commit | LnkA7 | Context-Specific | N | Boolean |

7.3.5.2.1.1 *Source Object Class* — Where applicable, the class of the object instance from which the source attribute is to be linked.

7.3.5.2.1.2 *Source Object Instance* — The instance number of the object from which the source attribute is to be linked.



Documents Document Sharing Hub

Full document details: SEMI E54.1-0316-0124R