SEMI E54-0306
SENSOR/ACTUATOR NETWORK STANDARD

This standard was technically reapproved by the Global Information and Control Committee and is the direct responsibility of the North American Information and Control Committee. Current edition approved by the North American Regional Standards Committee on March 14, 2004. Initially available at www.semi.org May 2004; to be published July 2004. Originally published in 1997; previously published March 2003.

NOTICE: The designation of SEMI E54 was updated during the 0306 publishing cycle to reflect the creation of SEMI E54.18.

NOTICE: The document that was previously designated as SEMI E54 (Standard for Sensor/Actuator Network Common Device Model) has been redesignated as SEMI E54.1. Because the Sensor/Actuator Network Standard is the parent document, it has been designated as SEMI E54.

1 Purpose

1.1 This specification provides the structure of SEMI's Sensor/Actuator Network (SAN) standard. It provides the definition for interoperability with respect to SEMI SAN standard-compliant Sensor/Actuator devices.

2 Scope

2.1 This standard specifies how devices interoperate on a network as part of the control system for equipment.

2.2 This specification, which is the root of the SAN standard, defines the relationships of each of the other specifications that are components of the SEMI SAN 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

3.1 *SEMI Standards*

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

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

3.2 *Other Documents*

ISO 7498[1] — Basic Reference Model for Open Systems Interconnection (OSI)

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.

4 Terminology

4.1 *Abbreviations & Acronyms*

4.1.1 *API* — Applications Programming Interface

4.1.2 *CDM — Common Device Model.* The root object application model for devices on the sensor bus.

4.1.3 *DM* — DeviceManager. An object specified in SEMI E54.1 (Standard for Sensor/Actuator Network Common Device Model).

1 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30
Website: www.iso.ch

4.1.4 *IDL* — Interface Definition Language. A programming language-independent notation for specifying interfaces.

4.1.5 *ISO* — International Organization for Standardization.

4.1.6 *NCS* — Network Communication Standard. Provides the specification for mapping the CDM and SDM to specific network technologies. These standards will be incorporated as parts of the SEMI SAN standard.

4.1.7 *OEM* — Original Equipment Manufacturer. They are usually the control system developer, but in any case they are almost always responsible for the control system.

4.1.8 *OMT* — Object Modeling Technique. A methodology developed by James Rumbaugh, et al. for using objects to model systems.

4.1.9 *OSI* — Open System Interconnection. A seven-layer model for communications developed by ISO.

4.1.10 *OSS* — Object Services Standard. SEMI E39.

4.1.11 *SAN* — Sensor/Actuator Network. It is frequently used to reference this standard.

4.1.12 *SDM* — Specific Device Model. This is an application model for a specific type of sensor, actuator, or entity.

4.2 *Definitions*

4.2.1 *application* — for software, this is a working series of computer instructions that provide end user services.

4.2.2 *applications model* — a formal description of the software elements and interactions that perform an end user task.

4.2.3 *entity* — in software engineering, this is something that is recognizable as distinct and particular from the other things that make up a software system or program.

4.2.4 *interface* — in information modeling, it is the boundary between two entities from which information will flow.

4.2.5 *user layer* — in communications, this is a set software function connected to the communication protocol's top layer, usually called the applications layer, from the user's application environment. It typically allows the user application to be protocol-independent.

5 Overview

5.1 Referring to Figure 1, the SAN standards specify a complete communications solution covering all the layers of the OSI reference model. The CDM and SDM together specify a protocol-independent set of application services, attributes, and behavior for a device that must be supported by the user layer as shown in Figure 1. An NCS completes the specification of the user layer by specifying a mapping to a network technology's interface. The NCS must specify its capabilities referenced against the ISO OSI Reference Model. An NCS may utilize an open or commercial solution. In any case, the NCS will provide references to those solution's specifications.

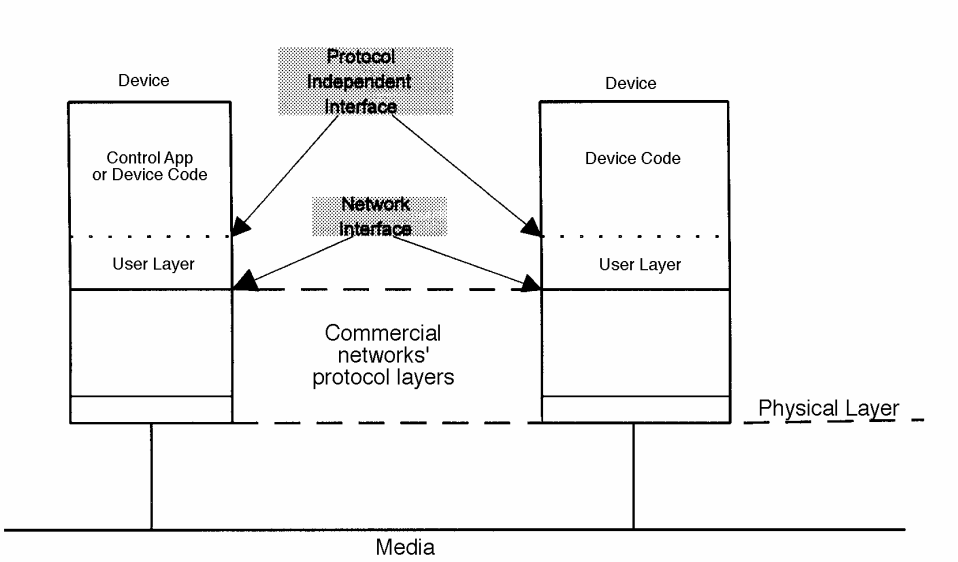


Figure 1
SEMI SAN Protocol Layers

5.2 Generally, the functionality of a software layer is embodied by the definition of the services it provides. Its services are the set of messages it generates and to which it responds, and its subsequent behavior upon receipt of any particular message. The CDM and an SDM provide a complete user layer specification in a platform and language-neutral form (dotted line in Figure 1). The goal being to provide device interoperability at the applications level. The NCS will guarantee interoperability at the network level (i.e., across a single type of network). The SAN standard does not require implementation of the top interface to the user layer, as specified by the CDM and an SDM. The CDM and an SDM provide the information needed to develop an NCS, which defines the interface requirement between the user layer and a specific network technologies communication layer (the solid line labeled as Network Interface). This means that the SAN standard allows user applications, whether from the device or the controller side implementation viewpoint, to interact directly with a network technologies interface.

5.3 *Network Technology Supplier View* of *the SAN Standard*

5.3.1 The CDM and an SDM together specify the messages, behavior, and attributes of a device type that will be visible over the network. In Figure 1, this is the area above the line that indicates the protocol-independent interface. Network suppliers will map this information to the top layer of their protocol. They do this by helping to develop and extend their NCS.

5.3.2 Most network suppliers have an application layer already defined for their protocol. In that case, the user layer as specified in a SEMI SAN NCS is quite thin. In fact, it is just a simple mapping of SDM interface to messages and data (attributes) provided by the network. Thus, an NCS (one for each supported network) specifies a user layer that sits on top of the network supplier's protocol.

5.4 *Device Supplier View of the SAN Standard*

5.4.1 Device suppliers are not required to implement designs illustrated in the specific device model (SDM) specifications. The designs are presented using an object-modeling notation as a means to clearly specify the devices interface. An SDM provides a model of a device, it is not a design for the device. Device suppliers are not expected to provide an object-oriented implementation. However, in order to ensure that a device interface responds as expected, it is necessary for the device supplier to not only provide the support for the services and attributes specified, but to ensure that the code that is implemented on the device behaves as specified by the applications model for a specific device type.

5.5 *Equipment Supplier View of the SAN Standard*

5.5.1 The OEM's have the largest task in adopting the SAN standard. They have to implement applications that use the interfaces defined for the many devices that make up the typical equipment control system. They have to be

familiar with the interfaces of a large number of the SDMs. On the other hand, they will become insulated from the inner workings of many devices, and from the code that they used to write to support those inner workings. In many cases, they may have to implement more than one Network Communication Standard.

5.5.2 During early adoption, OEM's may look to the SAN to act as a replacement for memory mapped discrete wired IO. With this type of adoption, the OEM will not lessen his burden of control code.

5.5.3 As Sensor Bus becomes more broadly adopted, OEM's may take advantage of the distributed processing capability offered by intelligent devices to reduce the complexity of their control software.

5.6 *The SAN Standard Structure*

5.6.1 A SAN employs devices from numerous sup-pliers. The SEMI SAN standard provides a framework to ensure interoperability of these devices within a network. The standard enables device suppliers and equipment control system developers to achieve that end. The SAN standard comprises five specifications.

5.6.2 The first specification is this document, which is the root document and specifies the connection between the other documents of the standard. It also specifies the construction for each of the other documents.

5.6.3 The next specification is the Common Device Model Specification. Each device on the SAN must comply with this specification.

5.6.4 The third specification defines the templates for creating ballots for specific device models and additions to the NCS.

5.6.5 The fourth specification is Specific Device Models standard. These standards contain models for various types of sensors, actuators, and entities. For instance, a model of a mass flow controller, a manometer, and a thermocouple will be specified.

5.6.6 The fifth type of specification is the Network Communication Standards. Each network technology has its own standard as an ancillary specification to the SAN standard.

5.6.7 Applications notes may accompany a number of the standards documents and are intended to guide implementors in the application of the standard.

6 Concepts

6.1 *Minimal device* — The SAN specifications allow simple and complex devices to be built compliant to the standard. This is done by specifying a minimum of required attributes, services, and behaviors in an SDM. The SDMs may also provide optional service definitions that must be followed if extended capabilities are to be built into a device. Devices that do not implement the extended capabilities must respond to extended service or access requests with an appropriate response code rather than failing to respond or issuing an exception or alarm message.

6.2 Many of the SAN NCS specifications rely on commercial technology. A significant number of network technologies are available off-the-shelf. The standards do not prescribe a choice. Once a technology described in an NCS is chosen, the NCS specifies how it is to be applied. The SAN standard is easily extensible for future technologies.

6.3 A device supplier must be able to use the standard to describe a device with little or no ambiguity for the device users. For the user, this means that he can select a device with confidence and that it can be easily integrated with his control system. The user must understand the electrical interface, the capabilities, and messages which a device supports based on the supplier's description of how this standard is supported by the device.

7 Requirements

7.1 *Organization of the SAN Specifications* — Figures 2 and 3 are intended to assist in understanding the document relationships that are specified in the following subsections. In these figures, all specifications that are standards are drawn as rounded rectangles. In Figure 3, the shadowed rectangles indicate the standards that contain requirements for implementing devices that are compliant to this standard.

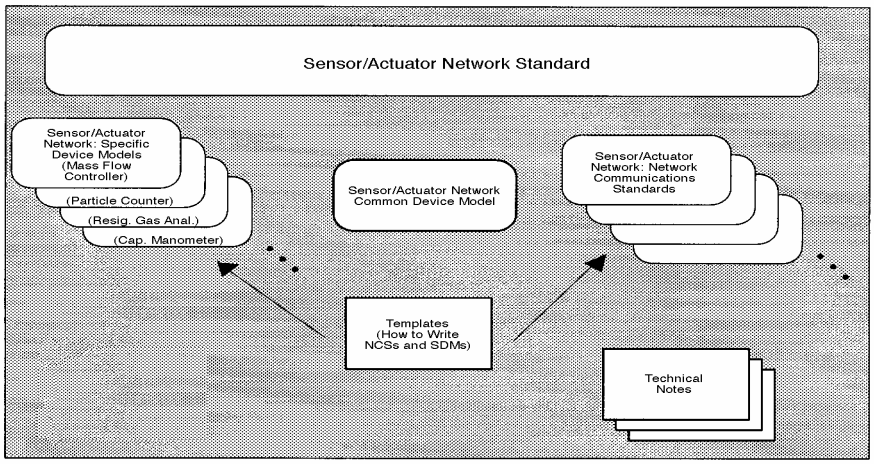


Figure 2
Organization of SAN Specification

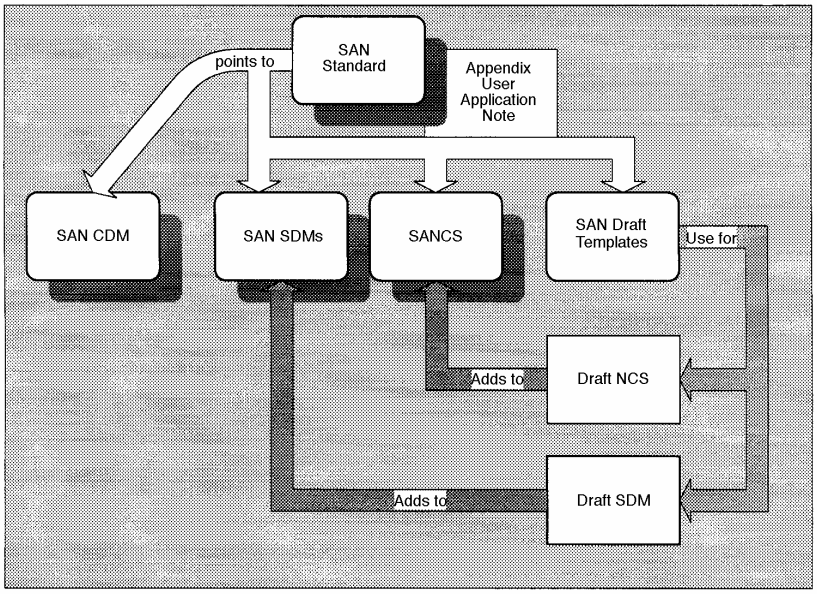


Figure 3
SAN Standard Relationships

7.1.1 The CDM specification is the first ancillary document of the SAN suite.

7.1.2 A second ancillary specification to the SAN standard provides templates that are guides to adding specific device models and network communication mappings to the SAN standard.

7.1.3 Specifications for specific device models for all types of devices are included as additional ancillary specifications of the SEMI SAN standard. Specific device models are added in the order they are developed and passed SEMI's ballot procedure.

7.1.4 Each network technology's user layer specification is included as an ancillary standard (see Figure 3). These *Network Communication Standards (NCS)* provide the mapping for each of the approved specific device models.

Therefore, each NCS is also a specification that grows over time. As new device models are developed and approved, a mapping of the model to a network technology is added to a NCS.

7.1.5 *Auxiliary Information* — Auxiliary information may be included with the standard in the form of appendices or technical notes. Generally, this information is not balloted, but is included as provided by the SEMI Standards Program regulations. Auxiliary information shall always be noted as such when it accompanies a specification or ballot.

7.2 *Interoperability*

7.2.1 *At the Applications Level* — The SDMs are used to derive the definitions for attributes, services, and behavior that are unique to each device type. A model may discuss relationships internal to a device to provide additional meaning to the services and attributes that can be manipulated over the network. Applications written to these models can operate a device independent of the network on which they communicate. The models shall assure that devices of different types can operate cooperatively at the applications level.

7.2.2 *Below the User Layer* — The various SANCS specifications describe the method by which a device communicates over a specific network such that it can meet the requirements of its type. All devices described within an NCS shall operate cooperatively on that specific network technology below the user layer.

7.2.3 The SEMI SAN standard does not address methods for internet-working below the applications layer, as described herein. An application entity may connect to more than one network. This could be typical of the application entities on the equipment controller. While the SAN standard allows that application entities could exist at any network node, it requires no mechanism for a message that begins on one network to be routed to another network.

7.3 *SEMI SAN Compliance* — To fully describe the characteristics of a network capable device, its compliance shall be specified by reference to the SAN ancillary standards. Thus, a device indicates both its type and network technology for its SDM and a NCS.

7.3.1 For a device to be compliant to the SAN standard, it shall implement all of the required attributes, behavior, and services as described in the SAN CDM.

7.3.2 *Type or Applications Model Specification* — The SDM of a device (which is its type, e.g., MFC or thermocouple) shall be denoted by SEMI E54.y, where “y” indicates the SDM ancillary standard.

7.3.3 Network Communication Specification Communication (protocol) compliance specification of a device is denoted by SEMI E54.x, where “x” is the network communications ancillary specification.

RELATED INFORMATION 1
APPLICATION NOTES

NOTICE: The material contained in these Application Notes is not an official part of SEMI E54 and is not intended to modify or supersede the official standard. Rather, these notes are auxiliary information included as reference material for implementers of the standard. The standard should be referred to in all cases. SEMI makes no warranties or representations as to the suitability of the material set forth herein for any particular application. The determination of the suitability of this material for any particular purpose is solely the responsibility of the user.

R1-1 Implementing a SAN Standard-Compliant Device

R1-1.1 This Related Information section describes how the SAN standard is used by a device supplier who wishes to offer SEMI SAN-compliant devices to the market.

R1-1.2 While the SAN standard may appear overwhelming on first viewing, it is actually built from independent components. It is not necessary to understand or 'know' the entire standard. It can be dealt with in small pieces. For device suppliers, only the pieces that deal with their particular device type need to be considered. The specifications to implement a specific device are typically less than thirty pages of text. The standard is structured such that it is extensible while providing the ability to specify a device by reference to the standard.

R1-1.3 *Synopsis of Device Operation* — All devices on a SAN have a similar operational profile. That profile is specified in SEMI E54.1 (Standard for Sensor/Actuator Network Common Device Model) and in the Common Device Model support section of each NCS.

R1-1.3.1 Table R1-1 explains where to find specifications for various device operations.

Table R1-1

Operation	*Spec.*
Power applied	CDM
Enable SAC	CDM
Enable DM	CDM
Enable Communications	NCS
Establish Communication	NCS
Send Events	CDM
Receive messages/answer	CDM
Send messages/accept	CDM

R1-1.4 *Device Type* — Each type of device is specified in the SDM specifications. One SDM is devoted to each device type. An SDM is always a specialization of the CDM. The SDM provides services, behavior, and attributes that are in addition to those specified in the CDM. For example, the SDM of a manometer might specify attributes for current pressure and alarm bands. It might have services needed for performing calibration.

R1-1.5 *Network Technology* — How a network technology is used by the SAN standard is specified in an NCS. The NCS for a technology provides a mapping of the CDM and SDM of a device type to a specific network protocol.

R1-1.5.1 It is only necessary for a device to support an NCS and the section for its type within that NCS. The NCS provides all that is needed to ensure that a device can communicate and interoperate across a given network technology. The point is that from the other side of the wire, the network-independent interface is not visible. That interface is implied by the behavior of the network interface.

R1-1.6 *Network Independence* — Device suppliers have the option of supplying an interface at the protocol-independent applications layer, as indicated in Figure 1. While this requires more implementation work for the device supplier, it speeds the integration of sensors and actuators for the OEM.

R1-1.6.1 In this case, it is still necessary for the device to support the NCS to ensure that their device will interoperate with other devices on the network.

R1-1.6.2 To supply this option, a device supplier would provide a device driver-like interface (software) to the OEM.

R1-1.6.3 Devices that supply a protocol-independent API that is compliant to the SEMI SAN standard provide advantages to both device and control system suppliers. The device supplier can assure that his device receives syntactically correct messages at his network interface layer. Control applications written to the protocol-independent interfaces have the advantage that they would not have to be rewritten in order to change to another network. To achieve this, device suppliers would have to ship software, in the form of 'device drivers', that would be installed on the control supplier's platform.

NOTE 1: Standards for these 'device drivers' may be developed at a later time as they do not currently exist. However, most operating systems used by control systems suppliers support installable device drivers.

R1-1.7 *Recommendations on Optional and Extended Device Capabilities* — It is expected that device suppliers would not compete based on their network interface. Devices traditionally compete based on cost, service, stability, reliability, accuracy and repeatability, and relevant dynamic attributes. From a network perspective, all devices have these common characteristics: periodic reporting, polled reporting, asynchronous event reporting, calibration, health, and diagnostic reporting. Access and control over the common characteristics should be implemented based on the SAN standard. Access and control of the competitive characteristics of devices may be implemented with supplier-specific interface extensions. However, the device supplier is strongly cautioned in the use of these extensions. Heavy use of extensions could be a serious competitive disadvantage unless the supplier is the clear winner in all the competitive characteristics. A locked door could lock one in, it could also lock one out. Device suppliers will usually find that their extensions can be implemented using the optional features of the CDM and SDM.

NOTE 2: Network interfaces could be a competitive factor during the early technology adoption phase.

R1-1.8 *The Controller Side Usage of the SAN Standard* — The control system supplier (in many cases this is the OEM) evolves a different view of SEMI's SAN standard. They are not as concerned with the network technology as with utilizing the applications capabilities as specified by the CDM and SDMs. It is strongly recommended that control applications not be network aware. This may produce some operating overhead, but for most applications, this is a good trade-off to gain flexibility provided by a protocol-independent interface. Many network devices may be supplied without any user layer (device driver) application side software. For these devices, the control system supplier would have to do some system's programming to provide a device driver for the application interface. Over time, it is likely that most devices will be delivered with a device driver. (Similar to the way printers are sold with Windows (TM) device drivers. This is what allows Windows applications to use a printer of the end users choosing.)

R1-1.8.1 The standards as of 1996 do not provide all the specifications needed to uniquely define how device drivers would be installed in the controller side of the API. This may become a future area of standardization, if OEMs decide to invest in the standard's development time.

R1-1.9 *Device Intelligence* — This varies considerably between device types and between devices of the same type. SEMI SAN specifications attempt to allow a range of intelligence to devices without forcing them to be 'highly' intelligent. To this end, the applications models specify a required minimum set of capabilities that must be supported for a device. Optional capabilities generally require more intelligent device operation. Devices that are not capable of supporting these extended capabilities provide a "not implemented" indication if access to those capabilities is requested.

R1-1.9.1 The following subsections describe device capabilities in order of increasing intelligence. A device's I/O direction is defined relative to the control application entity. Therefore, a sensor is an input, even though from the sensor's point of view it supplies output values.

R1-1.9.2 Polled input is the simplest sensor device requirement. All sensing devices should meet this requirement. In this mode of operation, a device only reports its value(s) when requested. This mode potentially provides the biggest burden on network bandwidth. However, it is the capability level that is most compatible with current control system architectures.

R1-1.9.3 Strobed output is the simplest actuator device requirement. All actuating devices should meet this requirement. This device only changes to a new state at the time of receipt of a specific message to do so.

R1-1.9.4 Periodic input is the capability to program the input device so that it sends its value(s) on a scheduled or periodic basis. Network bandwidth burden is reduced because it is not required to send a request each time an update for the input value is needed.

R1-1.9.4.1 This is also very compatible with current control architectures.

R1-1.9.5 Asynchronous input provides a capability for a sensing device to supply its value(s) when an event is detected. These events are usually programmed into the device. Examples of programmable events include threshold crossing of a value, value rate of change, measurement complete.

R1-1.9.5.1 Devices with this capability can significantly reduce network bandwidth burden, because they only need the network when some event has occurred. Equipment control architectures developed before 1990 may not be able to utilize this type of capability.

R1-1.9.6 Programmed output provides the capability for actuating devices to follow a programmable sequence of operations. For instance, an analog output could be given a ramp profile to follow as it moves between set points.

R1-1.9.6.1 This is another capability which can greatly reduce network bandwidth requirements. But, it is also a capability that 1990 vintage and earlier control systems may not be able to utilize easily.

R1-1.9.7 Devices with distributed processing capabilities are able to support features such as monitoring their health, providing diagnostics, and programmable preprocessing of data.

R1-1.9.7.1 These capabilities can have an impact on network burden, but more importantly may improve equipment reliability and maintainability. Again, this is a capability that is not readily utilized by control architectures developed prior to 1990.

R1-1.9.8 Devices with distributed control capabilities are capable of being programmed to create peer relationships with other devices on the network. Within these relationships, it is possible for one or more of the peers to control or regulate a subsystem within the equipment. For instance, a butterfly valve and manometer with these capabilities could be programmed to hold a particular pressure.

RELATED INFORMATION 2
INDEX FOR SENSOR/ACTUATOR STANDARDS

NOTICE: This related information is not an official part of SEMI E54 and is not intended to modify or supercede the official standard. After approval of this document, publication of this Related Information was authorized by the committee chairs solely to aid in identifying the relationships between the current Sensor Bus standards and documents in process. Determination of the suitability of the material is solely the responsibility of the user.

R2-1 Scope

R2-1.1 The table below is based on Figure 1 in SEMI E54.1, and is intended to show the relationships between standards that are currently published, as well as describe documents in process in SEMI Standards that, if approved, will be added to SEMI E54.

R2-1.2 There are five categories:

Main Document

Templates

Specific Device Model

Common Device Model

Network Communication Standards

R2-1.3 Standard titles with a designation number are published documents, and descriptors without designation numbers are documents in process.

Table R2-1 Sensor/Actuator Network (SAN) Standards

<table>
<tr><td colspan="3">SEMI E54, Sensor/Actuator Network (SAN) Standard</td></tr>
<tr><td colspan="3">Templates (How to Write Network Communications Standards (NCS) and Specific Device Models (SDM))</td></tr>
<tr><td>Specific Device Models</td><td>Common Device Models</td><td>Network Communications Standards</td></tr>
<tr><td rowspan="3">Mass Flow Controller SDM</td><td rowspan="3">SEMI E54.1, Standard for Sensor/Actuator Network Common Device Model</td><td>SEMI E54.4, Standard for SAN Communications for DeviceNet</td></tr>
<tr><td>SEMI E54.5, Standard for SAN Communications for the Smart Distributed System (SDS)</td></tr>
<tr><td>SEMI E54.6, Standard for SAN Communications for LONWorks</td></tr>
</table>

SEMI E54.1-1000
STANDARD FOR SENSOR/ACTUATOR NETWORK COMMON DEVICE MODEL

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on July 14, 2000. Initially available at www.semi.org August 2000; to be published October 2000. Originally published in 1996; previously published February 1999.

NOTE: This document was previously designated SEMI E54. Because this document is part of a suite of documents, its designation has been reassigned for ease of reference.

1 Purpose

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.

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 document is a standard for a *common* device model, 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 SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and thus are not applied to the Common Device Model.

4 Referenced Documents

4.1 *SEMI Documents*

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

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

4.2 *Other Documents*

ISO 7498[1] — Basic Reference Model for Open Systems Interconnection

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

1 International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211, Geneva 20, Switzerland. Available in the US from American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY10036.

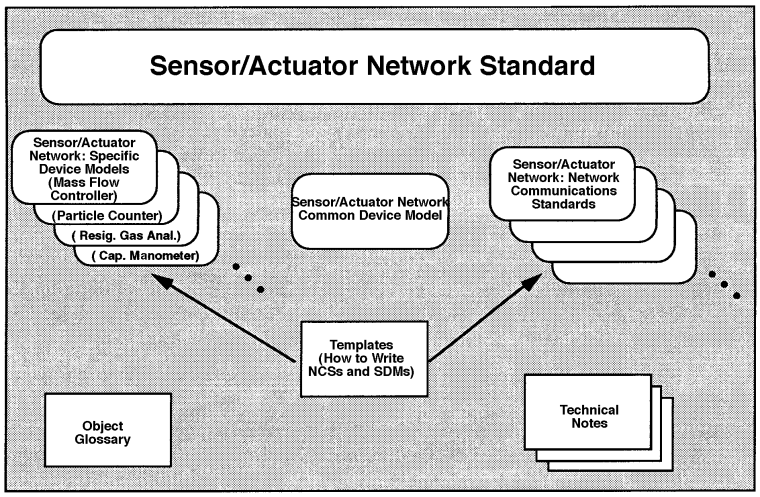


Figure 1
Sensor/Actuator Network-Related Documents

5 Terminology

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

5.1 *Device Component Terminology*

5.1.1 *attribute* — Externally visible information concerning an object.

5.1.2 *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.

5.1.3 *class* — A specific type or classification of objects.

5.1.4 *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.

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

5.1.6 *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.

5.1.7 *instance* — A specific and real occurrence of an object.

5.1.8 *manufacturer* — In the context of this document, this refers to the manufacturer of the device.

5.1.9 *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.

5.1.10 *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.

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

5.2 *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:

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

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

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

5.2.4 *character string* — A text string.

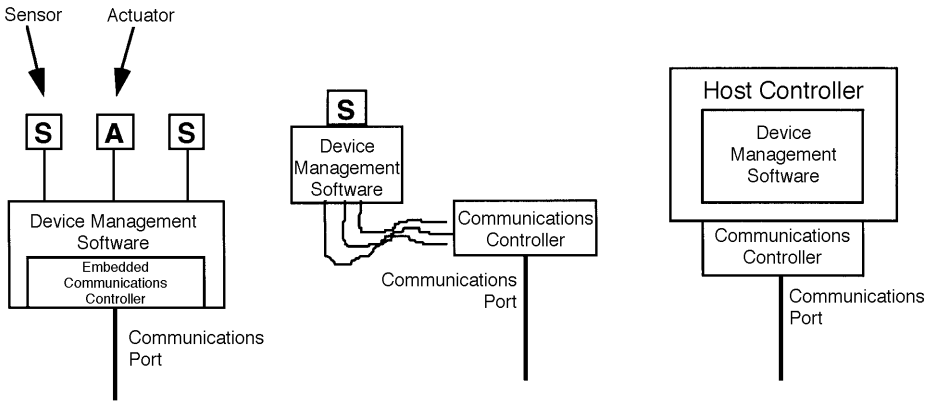


Figure 2
Examples of Devices

5.2.5 *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

5.2.6 *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.

5.2.7 *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 to 12)
3	Day	Unsigned Integer (range of 1 to 31)

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

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

5.2.10 *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.

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

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

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

5.2.14 *nibble* — A string of four adjacent binary bits.

5.2.15 *null character* — A byte with a value of zero.

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

5.2.17 *Short Integer (SINT)* — An integer, one byte long, in the range -128 to 127.

5.2.18 *Signed Integer (INT)* — An integer, 2 bytes long, in the range -32768 to 32767.

5.2.19 *text string* — A string of one byte characters.

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

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

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

5.2.23 *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.

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	(see below)

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.

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

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.

6.4.2.1 *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.

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).

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 another 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

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.

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.

7.1 *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 interrogation of an object instance via the sensor/actuator network (see 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.

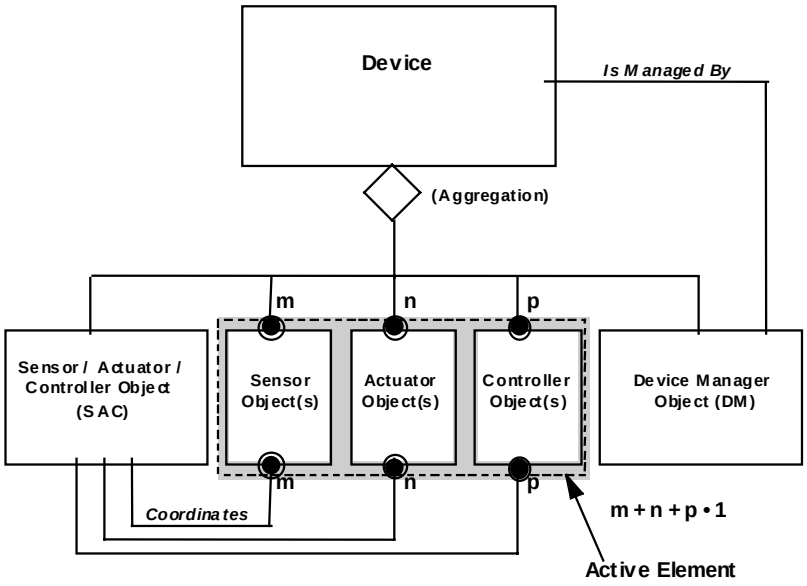


Figure 3
High Level Object View of a Device

Table 2 Device Objects and Identifiers

Object Name	*Object Identifier (tag)*	*Support Required in a Device*	*Comment*
Sensor/Actuator/Controller	SacI0*	Yes	Only one instance per device allowed.
Sensor	SenIn**	No	Zero or more instances per device allowed.***
Actuator	ActIn**	No	Zero or more instances per device allowed.***
Controller	CntIn**	No	Zero or more instances per device allowed.***
Device Manager	DmI0*	Yes	Only one instance per device allowed.

* Only one object instance per device; identifier uses "I0" to specify "instance zero."

** "In" is used to indicate the instance number of the object; "n" is a non-negative integer.

*** 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.

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.

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.

7.2 *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.

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 Section 7.2.4.

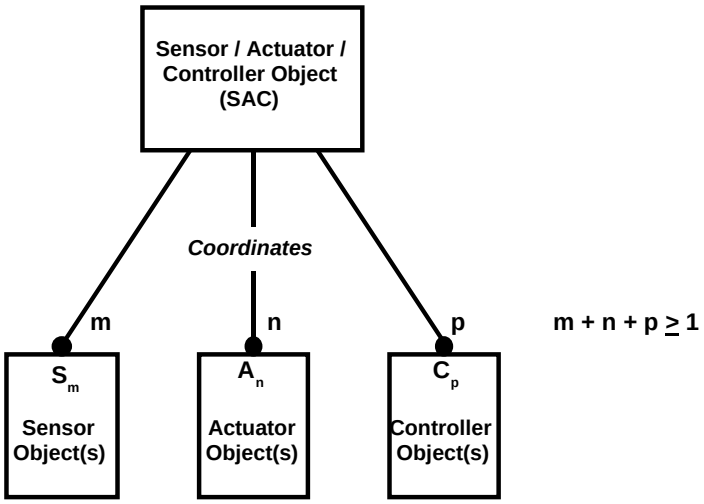


Figure 4
Detailed View of the Sensor/Actuator/Controller Object

Table 3 Objects Embedded in SAC Object and Identifiers

Object Name	*Object Identifier (tag)*	*Support Required in a Device*	*Comment*
Assembly	AsmIn*	No	Zero or more instances per device allowed.**
Local Link	LnkIn*	No	Zero or more instances per device allowed.**

* "In" is used to indicate the instance number of the object; "n" is a non-negative integer.

** 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.

7.2.1 *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 Section 7.2.3 and in an appropriate sensor/actuator network specific device model specification.

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*	Boolean
Run Hours	SacA5	R	No	unsigned integer
Reserved	SacA7-SacA64	—	—	Reserved for future expansion.

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

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

7.2.1.2 *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 Section 5.2.

7.2.1.3 *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.

7.2.1.4 *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 — see Section 7.3.1.14).

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

0 = Disable

1 = Enable

7.2.1.5 *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.

7.2.1.6 *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*	
Next Calibration Date	LVV	0,0,0	
Expiration Timer	LVV	0	
Expiration Warning Enable	LVV	0	
Run Hours	LVV	0	

* Corresponding to year, month, and day; see Section 5.2.

7.2.2 *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 common device model 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.

7.2.2.1 *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.

7.2.2.2 *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.

7.2.2.3 *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.

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 (see Section 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.

7.2.2.4 *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	*Request/ Indication*	*Response/ Confirmation*	*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.

7.2.2.5 *Set Attribut*e — 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	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Attribute ID	M	C	Network-Specific	Attribute Identifier of the attribute whose value is being modified.
Attribute Value	M	C	Context-Specific	Value of the attribute to modify.

7.2.2.6 *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.

7.2.2.7 *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	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Restore Conditions	M	C	byte, enumerated	0 = Restore specific attribute of this object 1 = Restore this object 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.

7.2.2.8 *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*

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

*The utilization of a service response message and its format is outside the scope of this document.

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

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

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

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

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.

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.

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

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.

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 non-volatile memory and reflect, in large part, the values set in a previous NORMAL OPERATING state.

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 (see Section 7.3).

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.

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 "ExpirationTimer" 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.

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.

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.

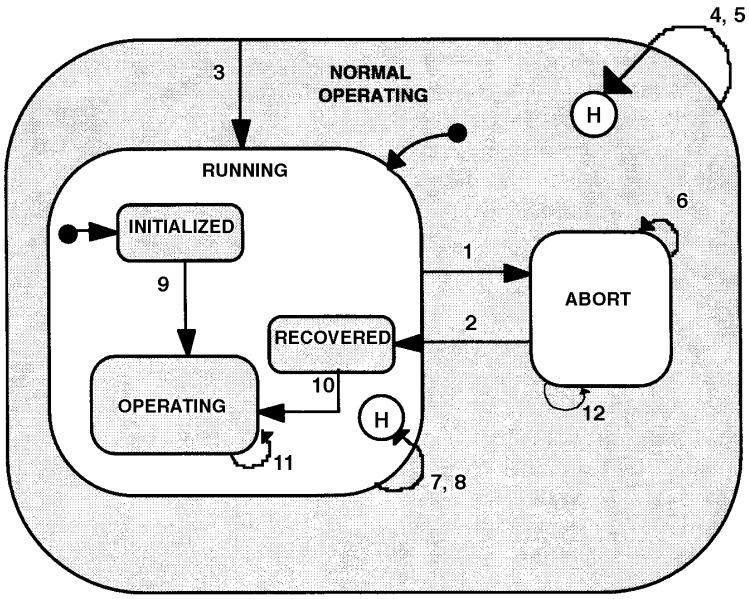


Figure 5
SAC Object Instance Behavior*

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

* Transitions 4 through 8, 11, and 12 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

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 sub-state to NORMAL OPERATING. Object instance is not in its ABORT state.
INITIALIZED	This is the entry sub-state 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 sub-states within RUNNING except INITIALIZED and RECOVERED. The further delineation of this sub-state and transition into this sub-state 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*, **

#	*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 sub-states of NORMAL OPERATING. Move to RUNNING/INITIALIZED.
4	NORMAL OPERATING	Invalid service request.	NORMAL OPERATING	Error response.	Valid for all substates of NORMAL OPERATING.

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.

* Transitions 4 through 8, 11, and 12 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

** Note: Only generic behavior is indicated in this table.

7.2.4 *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.

7.2.4.1 *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.

7.2.4.1.1 *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

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

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

7.2.4.1.2.1 *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.

Table 14 Assembly Object Get Attribute Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*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.

7.2.4.1.2.2 *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	*Request/ Indication*	*Response/ Confirmation*	*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.

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 (see Section 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.

7.2.4.1.3 *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.

7.2.4.2 *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.

7.2.4.2.1 *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

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

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

7.2.4.2.1.3 *Source Object Attribute* — The attribute ID of the source attribute to be linked.

7.2.4.2.1.4 *Destination Object Class* — Where applicable, the class of the object instance to which the destination attribute is to be linked.

7.2.4.2.1.5 *Destination Object Instance* — The instance number of the object to which the destination attribute is to be linked.

7.2.4.2.1.6 *Destination Object Attribute* — The attribute ID of the destination attribute to be linked.

7.2.4.2.1.7 *Commit* — An attribute that indicates whether the Link object will perform the link function (the link function is described in Section 7.2.4.2.3). When this attribute is FALSE, the object cannot perform the link function.

7.2.4.2.2 *Local Link Object Services* — There are no common Local Link object services specified. Services and service parameters may be provided by a Local Link object instance that are device-specific and/or implementation-specific. These services may be Notification or Request type services (see Section 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.

7.2.4.2.3 *Local Link Object Behavior* — Local Link object instance behavior common to all instances is as follows. When instantiated, this object performs the link function. That is, it first uses the GetAttribute service capability of the source object instance identified by the Source Object Instance and (possibly) Source Object Class attributes to retrieve the attribute value of the attribute identified by the Source Object Attribute. If the Commit attribute value is TRUE, the object then uses the SetAttribute service capability of the destination object instance identified by the Destination Object Instance and (possibly) Destination Object Class attributes to set the attribute value of the attribute identified by Destination Object Attribute to the (retrieved) value of Source Object Attribute. The object shall also perform the link function any time the Commit attribute value transitions to TRUE. The object shall not write to the destination object instance whenever the value of the Commit attribute is set to FALSE. Issues such as frequency of this link update behavior (while the Commit attribute is set to TRUE, or when the Commit attribute does not exist), type matching of Source and Destination attributes, etc., are beyond the scope of this document.

7.3 *Device Manager (DM) Object* — The DM object is the device component responsible for managing and consolidating the device operation. The DM object houses information about the device (e.g., serial number, device type) so that the device may be uniquely identified remotely via a sensor/actuator network. The DM object is also responsible for providing details of the device status. As such, the responsibility of the DM object includes managing the reporting of exceptions to normal processing as alarms or warnings. Note that it is outside the scope of this document to specify the detailed conditions which initiate alarms and warnings. Exception handling that is common to all devices on the network is described here. This behavior and structure may be extended to handle exceptions which are specific to certain devices or are proprietary to individual manufacturers. Extensions for specific devices would be described in a sensor/actuator network specific device model for those devices.

A device shall contain exactly one instantiation of a DM object.

7.3.1 *DM Object Attributes* — All required and "common optional" DM object attributes are listed in Table 17. 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 "write" access (via the network) to the attribute, (i.e., an indication of whether the attribute is network-settable); and (3) an attribute type and maximum length. (Note that in many cases it may be desirable to limit the length of these attributes.) Required attributes must be supported in all DM 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 Section 7.3.3 and in an appropriate sensor/actuator network specific device model specification. DM attributes are identified with an attribute identifier of the form "DmAn" where "n" is a positive decimal number. DM attribute identifiers DmA1 through DmA14 are used in this common device model standard. DM attribute identifier numbers DmA15 through DmA64 are reserved.

Table 17 Device Manager Instance Attributes*

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	DmA1	RO	Y	Text, 8 characters maximum
Standard Revision Level	DmA2	RO	Y	Text, 10 characters maximum
Device Manufacturer Identifier	DmA3	RO	Y	Text, 20 characters maximum
Manufacturer Model Number	DmA4	RO	Y	Text, 20 characters maximum
Software or Firmware Revision Level	DmA5	RO	Y	Text, 8 characters maximum
Hardware Revision Level	DmA6	RO	Y	Text, 8 characters maximum
Serial Number (optional)	DmA7	RO	N	Text, 30 characters maximum
Device Configuration (optional)	DmA8	RO	N	Text, 50 characters maximum
Device Status	DmA9	RO	Y	unsigned integer, enumerated, see text
Reporting Mode	DmA10	RW	Y	Formatted byte, see text
Exception Status Report Interval (optional)	DmA11	RW	N	unsigned integer, < 65,536

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Exception Status	DmA12	RO	Y	Formatted byte, see text
Exception Detail Alarm (optional)	DmA13	—	N	Structure of
Common Exception Detail	—	—	—	Structure of
Size	—	RO	—	unsigned integer, < 256
Detail	—	RO	—	Ordered list of bytes, see text
Device Exception Detail	—	—	—	Structure of
Size	—	RO	—	unsigned integer, < 256
Detail	—	RO	—	Ordered list of bytes, see text
Manufacturer Exception Detail	—	—	—	Structure of
Size	—	RO	—	unsigned integer, < 256
Detail	—	RO	—	Ordered list of bytes, see text
Exception Detail Warning (Optional)	DmA14	***	N	Structure of***
Visual Indicator	DmA15	RW	N	Byte, enumerated
Alarm Enable	DmA16	RW	N	Boolean
Warning Enable	DmA17	RW	N	Boolean
Exception Detail Type	DmA18	RW	N	Byte, enumerated
Exception Detail Alarm Queue	DmA19	R	N	Ordered list of type Exception Detail Alarm.
Exception Detail Warning Queue	DmA20	R	N	Ordered list of type Exception Detail Warning.
Date and Time	DmA21	RW	N	See Section 7.3.1.23
Date and Time Type	DmA22	RW	N	Enumerated USINT
Reserved for CDM	< DmA32	—	—	Reserved for future CDM DM attribute definitions.

* A “Definition” column is not included. For attribute definitions, see text.

** A value of 00 indicates that this timer is disabled.

*** Same as Exception Detail Alarm.

7.3.1.1 *Device Type* — An attribute which identifies the type of the device on the sensor/actuator network. It is formatted as a text string with a maximum length of 8 characters. The exact content may be defined in an appropriate sensor/actuator network specific device model specification. For example, “MFC” = mass flow controller, “CDG” = capacitance diaphragm gauge.

7.3.1.2 *Standard Revision Level* — An attribute which identifies the most recent version of the sensor/actuator network specific device model to which the device adheres. It is formatted as a text string with a maximum length of 9 characters as “ENNNNNNYY,” where “E” is a constant defined by SEMI, “NNNNNN” is the number of the standard, and “YY” is the last two digits of the year in which the standard/revision was published. If there is no relevant standard or the device is not compliant to the standard, then “NNNNNN” is zero length and “YY” is set to “00” (thus, the attribute is set to “E00”).

7.3.1.3 *Device Manufacturer Identifier* — An attribute which designates the manufacturer of the device. Each manufacturer is responsible for defining this variable for their products in a way which will be unique to that manufacturer and which must be the same for all devices they provide. Formatted as a text string with a maximum length of 20 characters, it will usually be the company name, as in “Acme Instrument Co.”

7.3.1.4 *Manufacturer Model Number* — An attribute which designates the basic model identification number, defined by the manufacturer. It is formatted as a text string with a maximum length of 20 characters.

7.3.1.5 *Device Software or Firmware Revision Level* — An attribute which designates the version of microprocessor code which is contained in the device, defined by the manufacturer. It is formatted as a text string with a maximum length of 5 characters.

7.3.1.6 *Hardware Revision Level* — An attribute which designates the version of the device, excluding the microprocessor code, defined by the manufacturer. It is formatted as a text string with a maximum length of 5 characters.

7.3.1.7 *Serial Number (optional)* — An attribute which designates the number defined and assigned by the manufacturer that uniquely identifies each individual device produced. It is formatted as a text string with a maximum length of 30 characters.

7.3.1.8 *Device Configuration (optional)* — An attribute which designates a manufacturer-defined device configuration (beyond model number). It is formatted as a text string with a maximum length of 50 characters.

7.3.1.9 *Device Status* — An attribute which designates the state of the DM object. The attribute (unsigned integer) values and corresponding object states specified by this attribute are enumerated in Table 18.

Table 18 Device Status Attribute Values

Attribute Value	*DM Object State*
0	Unknown
1	INITIALIZED/SELF TESTING
2	IDLE
3	SELF TEST EXCEPTION
4	EXECUTING
5	ABORT FROM IDLE OR EXECUTING
6	ABORT FROM INITIALIZED/SELF TESTING OR SELF TEST EXCEPTION
7 – 255	Reserved

This attribute is set and updated internally by the DM object as part of the object behavior associated with transition between the various states (see Section 7.3.3 and Figure 6).

7.3.1.10 *Reporting Mode* — A network-settable attribute defined as a single byte containing two nibbles: A low nibble (bits 0 – 3) and a high nibble (bits 4 – 7). The high nibble contains the value for alarm conditions. The low nibble contains the value for warning conditions. The value assigned to each nibble indicates how the alarm/warning status variable is to be reported. The possible attribute values and corresponding reporting modes specified by this attribute are enumerated in Table 19.

Table 19 Reporting Mode Attribute Values for Alarms and Warnings

Reporting Mode	*Attribute High Nibble Value (Alarm)*	*Attribute Low Nibble Value (Warning)*
Request	0	0
RequestLatched (optional)	1	1
EventTriggeredOn (optional)	2	2
EventTriggeredOn/Off (optional)	3	3
TimeTriggered (optional)	4	4
EventOnOrTimeTriggered (optional)	5	5
EventOn/OffOrTimeTriggered (optional)	6	6

The behavior of the DM object associated with this attribute and each of these reporting modes is described in Section 7.3.3.

7.3.1.11 *Exception Status Report Interval (optional)* — A network-settable attribute which specifies the time interval for periodic exception status reporting (when a time-triggered reporting mode is indicated, see definition of Reporting Mode attribute and Section 7.3.3). It is formatted as an unsigned integer whose least significant bit corresponds to a value of 1/100 second. The requirement to store this attribute in non-volatile memory (so that the value upon "power-up" is the last value active) is device-specific. Note that a value of 0.00 indicates that this timer, and thus time-triggered exception reporting, is not enabled (see also Section 7.3.3).

7.3.1.12 *Exception Status* — A single-byte attribute whose value indicates the status of the alarms and warnings for the device (see Table 19). This indication may be provided in one of two methods: Basic or Expanded. In the Basic method, bit seven of the Exception Status attribute is set to zero; all exceptions are reported exclusively through

communication of this Exception Status attribute. The format of bits zero through six in this mode is device-specific; the format may be further specified in an appropriate sensor/actuator network specific device model specification; if it is not specified, then the format of bits zero through six is equivalent to that specified in the expanded mode. In the Expanded method, bit seven of Exception Status attribute is set to one; exceptions are reported through the communication of this Exception Status attribute, formatted as specified in Table 20, and exception detail attributes as indicated by the Exception Status attribute.

Table 20 Exception Status Attribute Value*

	Exception Status Bit Map, Bit 7 set to 0	*Exception Status Bit Map, Bit 7 set to 1*
bit	*Function*	*Function*
0	Device-Specific	ALARM/device-common**
1		ALARM/device-specific
2		ALARM/manufacturer-specific
3	See sensor/actuator network specific device model for further specification	reserved
4		WARNING/device-common**
5		WARNING/device-specific
6		WARNING/manufacturer-specific
7	0 == Basic Method	1 == Expanded Method

* Behavior associated with the reporting of exception status attribute values is described in Section 7.3.3.

** The alarm or warning is not specific to the device type or device type manufacturer.

7.3.1.13 *Exception Detail Alarm and Exception Detail Warning (optional)* — Attributes that relate the detailed status of the alarms or warnings associated with the device. Each attribute is a structure containing three members; these three members respectively relate the detailed status of exceptions that are device-common (i.e., not device-specific), device-specific but not manufacturer-specific, and manufacturer-specific. Each of the three structure members is defined as a structure containing an ordered list (i.e., array) of bytes of length SIZE, and an unsigned integer whose value is SIZE. Each of the bytes in each array has a specific mapping. This mapping is formatted as 8 bits representing 8 independent conditions, whereas a value of 1 indicates that the condition is set (or present), and a value of 0 indicates that the condition is cleared (or not present). Note that if a device does not support an exception detail, the corresponding bit is never set. The bitmaps for alarms and warnings in the corresponding attributes are structured in parallel so that a condition may have either alarm or warning set, depending on severity. If a condition inherently cannot be both alarm and warning, then the parallel bit position corresponding to the other state will remain “0.”

The existence of an exception detail variable structure is dependent on the value of the Exception Status Attribute; the existence of an exception detail variable structure is only required if bit seven of the Exception Status attribute is set to 1 (indicating Expanded mode reporting) and the bit (among bits zero through six) of the Exception Status attribute corresponding to the particular exception type is also set to 1 (see Table 20).

7.3.1.14 *Common Exception Detail (optional)* — This structure relates exception conditions (i.e., alarms or warnings) which are common to all devices on the network. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element Size. For each byte in the Detail field, all bits which are not identified are reserved for future standardization. The first byte in this attribute is CommonExceptionDetail[0]. Additional exception details, if provided, are named CommonExceptionDetail[1], CommonException Detail[SIZE]. The specific exception associated with each of the bitmaps is given in Table 21. The criteria details for each exception condition are outside the scope of this document.

Table 21 Common Exception Detail Attribute Value

	*Common Exception Detail**
bit	*Detail[0]*
0	internal diagnostic exception
1	microprocessor exception
2	EPROM exception
3	EEPROM exception
4	RAM exception
5	communications exception
6	internal real-time exception
7	calibration expiration

	*Common Exception Detail**
bit	*Detail[1]*
0	power supply overcurrent
1	reserved power supply
2	power supply output voltage
3	power supply input voltage
4	routine maintenance due
5	notify manufacturer
6	reset exception
7	reserved

* Note that if a device does not support an exception detail, the corresponding bit is never set.

7.3.1.15 *Device Exception Detail (optional)* — This structure, similar in form to Common Exception Detail, relates exception conditions which are specific to individual devices on the network and are defined in their respective device model documents. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element size. For a detailed description of this attribute, consult the appropriate specific device model documentation.

7.3.1.16 *Manufacturer Exception Detail (optional)* — This structure, similar in form to Common Exception Detail, relates exception conditions which are specific to the manufacturers of individual devices on the network and are defined by them in their product documentation. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element size. For a detailed description of this attribute, consult the appropriate specific device manufacturer documentation.

Additional attributes may be supported by a DM object that are device-specific. A complete definition of any additional attributes may be found in an appropriate sensor/actuator network specific device model specification.

7.3.1.17 *VisualIndicator* (Optional) — An attribute which defines the behavior of a visual indicator allowing the device to be visually located. This attribute is an enumerated byte that can take on one of the following values:

0 = Visual Indicator OFF

1 = Visual Indicator ON

2–63 = Reserved

64–255 = Manufacturer-Specified

For values visual indicator OFF and visual indicator ON, the visual indicator is defined to be a Light Emitting Diode (LED). When the value is visual indicator ON, the device is expected to flash the LED at a frequency of approximately 0.5 to 1.5 Hz at a duty cycle of approximately 30 to 70 percent. The location and labeling of the LED are manufacturer-specified.

7.3.1.18 *AlarmEnable* (Optional) — An attribute which defines whether or not the object will report alarms (as indicated by the ReportingMode, ExceptionStatus, ExceptionDetailType, and ExceptionDetailAlarm attributes). When this attribute is set to TRUE, then alarming is enabled. When this attribute is set to FALSE, all DM exception attributes are set and maintained at values that indicate a no alarm state. Thus bits 0 through 2 of the ExceptionStatus attribute are set and maintained at zero, and any ExceptionDetailAlarm attribute bytes are set and maintained at a value of zero.

7.3.1.19 *WarningEnable* (Optional) — An attribute which defines whether or not the object will report warnings (as indicated by the ReportingMode, ExceptionStatus, ExceptionDetailType, and ExceptionDetailWarning attributes). When this attribute is set to TRUE, then warning exception reporting is enabled. If this attribute is set to FALSE, all DM exception attributes are set and maintained at values that indicate a no warning state. Thus, bits 4 through 6 of the ExceptionStatus attribute are set and maintained at zero, and any ExceptionDetailWarning attribute bytes are set and maintained at a value of zero.

7.3.1.20 *ExceptionDetailType* (Optional) — An attribute consisting of two enumerated nibbles that designate the method of alarm and warning detail reporting respectively that is supported. The possible attribute values and corresponding reporting mode support indications are enumerated in Table 22. Note that if this attribute is not supported, the reporting mode is either zero (basic) or one (expanded) as indicated in bit seven of the ExceptionStatus attribute.

Table 22 ExceptionDetailType Attribute Values for Alarm and Warning Reporting Methods

Reporting Method	*Attribute High Nibble Value (Alarm)*	*Attribute Low Nibble Value (Warning)*
Basic	0	0
Expanded	1	1
Queued	2	2
Reserved for Future Extension of CDM	3 – 15	3 – 15

7.3.1.21 *ExceptionDetailAlarmQueue* (Optional) — An attribute consisting of an ordered list of alarm events; the data structure of the individual alarm events (i.e., queue elements) is determined by the values of the ExceptionStatus attribute; if bit 7 of the ExceptionStatus attribute indicates a Basic Mode reporting data format then each queue element is of the same data type as ExceptionStatus; if bit 7 of the ExceptionStatus attribute indicates an Expanded Mode reporting data format then each queue element is of the same data type as ExceptionDetail (Alarm/Warning). The behavior of this queue in storing and reporting alarms is described in Section 7.3.3.

7.3.1.22 *ExceptionDetailWarningQueue* (Optional) — An attribute consisting of an ordered list of warning events; the data structure is determined as described in Section 7.3.1.21. The behavior of this queue in storing and reporting warnings is described in Section 7.3.3.

7.3.1.23 *Date and Time* — A structure that relates the current time as perceived by the device. The form of the structure depends on the value of the *Data and Time Type* attribute as follows:

Date and Time Type	*Date and Time Structure*	*Semantics*
0 [default]	Struct of: UDINT UINT	{*Standard Time and Date*}(6 Bytes) Number of milliseconds since midnight Number of days since 01 January, 1972
1	Struct of: UDINT UINT INT	{*Standard Time and Date with Offset*}(8 Bytes) Number of milliseconds since midnight Number of days since 01 January, 1972 Number of minutes displacement from the Greenwich Meridian
2	Struct of: LINT	{*Universal Time Coordinated* (UTC)}(8 Bytes) Number of 100 nanoseconds since 15 October, 1582, 00:00:00
3	Struct of: LINT INT	{*UTC with Time Displacement Factor*}(10 Bytes) Number of 100 nanoseconds since 15 October, 1582, 00:00:00 Number of minutes displacement from the Greenwich Meridian
4 - 63	Reserved	
64 - 127	Specific Device Model definitions as required	
128 - 255	Manufacturer specified as required	

7.3.1.23.1 If the *Date and Time Type* attribute is not supported, the default value of zero (0) is assumed.

7.3.1.23.2 The *Standard Time and Date* (Type 0) uses 32 bits to represent the number of milliseconds since midnight followed by 16 bits to represent the number of days since 01 January, 1972. The reference for this format is either Greenwich Mean Time or Local Time as defined by the user. This format has a range of 01 January, 1972 to 06 June, 2151.

7.3.1.23.3 The *Standard Time and Date with Offset* (Type 1) uses the above format followed by 16 bits to represent the number of minutes displacement from the Greenwich Meridian. Meridians to the East are positive, while those to the West are negative.

7.3.1.23.4 The *Universal Time Coordinated* (UTC) (Type 2) is based on the format used by the WWV radio station of NIST. It uses 64 bits to represent the number of 100 nanoseconds since 15 October, 1582, 00:00:00. The reference for this format is always Greenwich Mean Time. This format has an approximate range of BC 27,000 to AD 30,000.

7.3.1.23.5 The *UTC with Time Displacement Factor* (TDF) (Type 3) uses the above format followed by 16 bits to represent the number of minutes displacement from the Greenwich Meridian. Meridians to the East are positive, while those to the West are negative.

7.3.2 *DM Object Services* — All DM object instances must provide the services listed in Table 23. (The DM object is the "service provider" of these services.) Note that all of these services are Request services, with the exception of the PublishAttribute service (which is a notification service). DM services are identified with a service identifier of the form "DmSn", where "n" is a positive decimal number. Dm service identifiers DmS1 through DmS8 are used by this common device model standard. DM service identifiers DmS9 through DmS32 are reserved. Service parameters that are specified in this standard are summarized in Table 24.

Table 23 DM Object Service Resource Definition

Service	*Service Identifier*	*Required*	*Type*	*Description*
Reset	DmS1	Y	R	Used to place object in INITIALIZED state.
Abort	DmS2	Y	R	Used to place object in ABORT state.
Recover	DmS3	Y	R	Used to move object from ABORT state to RECOVERED state.
GetAttribute	DmS4	Y	R	Used to read object attribute.
SetAttribute	DmS5	Y	R	Used to modify object attribute.
Execute	DmS6	Y	R	Used to move object from IDLE state to EXECUTING state.
PerformDiagnostics	DmS7	Y	R	Used to instruct object to perform diagnostic test.
PublishAttribute	DmS8	N	N	Used to proactively report exception status attribute.
Lock	DmS9	N	R	Used to restrict access to READ only attributes.
Unlock	DmS10	N	R	Used to make READ only attributes modifiable.
Get Exception Queue	DmS11	N	R	Used to retrieve all, or a specified number of, exception events from the front of the ExceptionDetailAlarm/WarningQueue.
Clear Exception Queue	DmS12	N	R	Used to clear all exception events from the ExceptionDetail Alarm or WarningQueue.
Reserved by CDM	DmS13 – DmS32	—	—	Reserved for CDM expansion.

Table 24 DM Object Service Parameter Dictionary

Parameter	*Form*	*Description*
AttributeID	Enumerated	DM object attribute identifier (see Table 17)
AttributeValue	Context-specific (see Table 17)	Value of DM object attribute.
TestID	Non-negative integer, < 256	Type and possibly detail of diagnostic test to be performed.

A detailed description of these services follows.

7.3.2.1 *Reset (Service Identifier: DmS1)* — Used to place the DM object in its INITIALIZED state. There are no parameters specified for this service.

7.3.2.2 *Abort (Service Identifier: DmS2)* — Used to place the DM object in its ABORT state. There are no parameters specified for this service.

7.3.2.3 *Recover (Service Identifier: DmS3)* — Used to move the DM object from its ABORT state to a recovered state (which is the IDLE state for this object). There are no parameters specified for this service.

7.3.2.4 *GetAttribute (Service Identifier: DmS4)* — Used to read a DM object attribute.

GetAttribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttributeID	M	C**, =	Attribute Identifier of attribute whose value is being requested. Inclusion of attribute in Rsp/Conf. is optional.
AttributeValue	—	M	Value of attribute requested.

* See Section 6.4.3 for a further description of terminology used in this table.

** The determination of whether Attribute ID is supplied in the service response/confirm is outside the scope of this document.

7.3.2.5 *SetAttribute (Service Identifier: DmS5)* — Used to modify a DM object attribute.

Set_Attribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttributeID	M	C, =	Attribute Identifier of attribute whose value is to be modified. Inclusion of Attribute in Rsp/Conf. is optional.
AttributeValue	M	C	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.

* See Section 6.4.3 for a further description of terminology used in this table.

7.3.2.6 *Execute (Service Identifier: DmS6)* — Used to move the DM object from its IDLE state to its EXECUTING state. Note that this service may be internally generated by the DM object. There are no parameters specified for this service.

7.3.2.7 *PerformDiagnostics (Service Identifier: DmS7)* — Used to instruct the DM object to perform a diagnostic test. A diagnostic test is either of type "common" or "device-dependent." "Common" diagnostic tests could include: RAM, EPROM, non-volatile memory, and communications. The structure of "common" type diagnostic tests is implementation-specific. All detail of "device-dependent" diagnostics is outside the scope of this document. The following table describes parameters specified for this service.

Perform_Diagnostics Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
TestID	M	C, =	Indicates the type and possibly detail of the test to be performed.

* See Section 6.4.3 for a further description of terminology used in this table.

Additional services and/or service parameters that are device-specific and/or implementation-specific may be provided by a DM object. These services may be Notification or Request type services (see Section 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.

7.3.2.8 *PublishAttribute (Optional — Service Identifier: DmS8)* — Used to proactively communicate a DM object attribute. The following table describes parameters specified for this service.

PublishAttribute Service Message Definition Table*

Parameter	*Ind*	*Description*
AttributeID	M	Attribute Identifier of attribute whose value is being reported.
AttributeValue	M	Value of attribute being reported.

* The utilization of a service response message and its format is outside the scope of this document.

7.3.2.9 *Lock* (Optional) — This service is used to restrict network access to all READ only attributes of all objects of this device. This service guarantees that the READ only attributes of this device are not modified over the network. There are no parameters specified for this service.

7.3.2.10 *Unlock* (Optional) — This service is used to make READ only attributes of objects of this device modifiable over the network. Not only is support of this object optional, but the degree to which it is supported is also specified by the manufacturer (i.e., only some attributes may become "unlocked"). Table 25 describes the parameters specified for this service.

Table 25 DM Object Unlock Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Password Key	C	C	Manufacturer-Specific	Password may be specified to restrict access or to provide a manufacturer-specified security scheme.

7.3.2.11 *GetExceptionQueue* (Optional) — This service is used to read all or portions of the AlarmQueue and WarningQueue elements. Table 26 describes the parameters specified for this service.

Table 26 DM Object GetExceptionQueue Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
ExceptionType	M	C**, =	Byte, enumerated0 = Alarm1 = Warning	Used to indicate which exception queue is being queried.
ElementNumb	M	M	unsigned integer < 65,536	Used to indicate the number of Exception queue events to be returned with the service response. A zero value implies that all elements in the queue should be returned.
ExceptionList	—	M	List of elements; element data type. Dependent on the Reporting Mode (Basic or Expanded — See Sections 7.3.1.12, 7.3.1.19, 7.3.1.20, and 7.3.3).	An array of exception queue events of size ElemNumb returned with the response.
Clear	M	C	Boolean	Used to indicate whether or not the exceptions are to be removed from the exception queue upon reporting.

7.3.2.12 *ClearExceptionQueue* (Optional) — This service is used to clear all elements out of either the AlarmQueue or WarningQueue. Table 27 describes the parameters specified for this service. If the ExceptionType attribute is set to 0, 1, or 2 with the service request, the Alarm, Warning, or both queues respectively are cleared completely of their members.

Table 27 DM Object Clear ExceptionQueue Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
ExceptionType	M	C**, =	Byte, enumerated 0 = Alarm 1 = Warning 2 = Both	Used to indicate which exception queue(s) is/are to be cleared.

7.3.3 *DM Object Behavior* — The DM object behavior is illustrated in Figure 6 and in Tables 28 and 29.

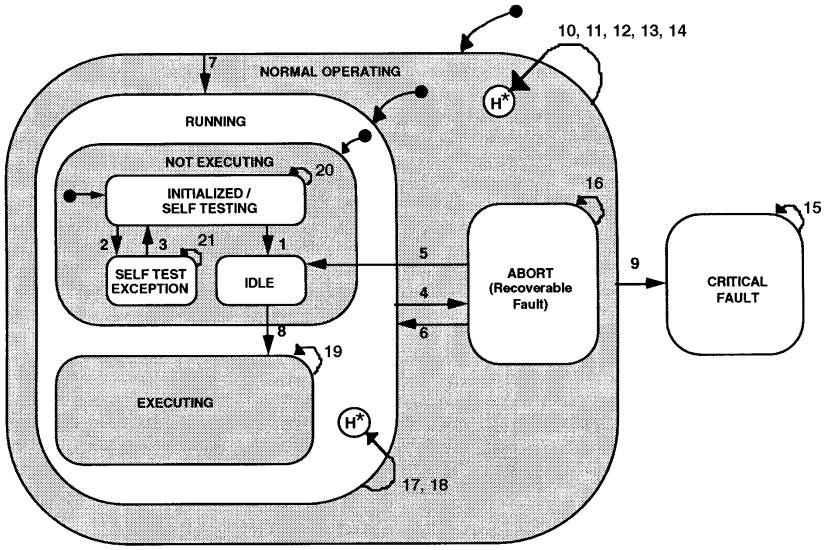


Figure 6
DM Object Instance Behavior*

*Transitions 10 through 21 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

Table 28 DM Instance Behavior State Description

State	*Description*
NORMAL OPERATING	Object instance exists. Object services can be processed.
RUNNING	This is the entry sub-state to NORMAL OPERATING. Object instance is not in an aborted state. Object is capable of reporting attribute values, including exceptions.
NOT EXECUTING	Device is not executing (e.g., it is not performing its device-specific function). A detailed description of this state is outside the scope of this document.
	This state may be further specified in an appropriate sensor/actuator network specific device model specification.
EXECUTING	Device is executing (e.g., it is performing its device-specific function). A detailed description of this state is outside the scope of this document.
	This state may be further specified in an appropriate sensor/actuator network specific device model specification.
INITIALIZED/SELF TESTING	This is the entry sub-state to NORMAL OPERATING and RUNNING.
	Object instance exists and has been initialized; all attributes have appropriate initial values(as indicated herein and in an appropriate sensor/actuator network specific device model specification).
	Device is performing device-specific and device type-specific tests to determine if it is qualified to be running.
	Note that as this is a sub-state of RUNNING, object is not initialized unless it is capable of reporting attribute values, including exceptions.
SELF TEST EXCEPTION	Object has detected an exception condition during self testing. The details of the exception are stored in the appropriate attribute values of the DM object.
IDLE	Object and device have been initialized and have successfully completed self testing.
ABORT	Object instance is in an aborted state; a detailed description of this state is outside the scope of this document. The object will not initiate communication over the network while in this state.
CRITICAL FAULT	The object (and device) are in a fault state from which there is no recovery. Object services cannot be processed. The conditions required for exit from a critical fault are outside the scope of this document.

Table 29 DM Instance Behavior State Transition Matrix*

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
1	INITIALIZED/ SELF TESTING	DM Object determines internally that the device is initialized and self test passed.	IDLE	Set Device Status attribute to appropriate value.	Device is in an idle state.
2	INITIALIZED/ SELF TESTING	One or more exceptions reported as a result of device testing.	SELF TEST EXCEPTION	Report exceptions through appropriate attribute settings in DM object. Set Device Status attribute to appropriate value.	Conditions resulting in the reporting of a self test exception are device-specific.
3	SELF TEST EXCEPTION	Exception condition cleared.	INITIALIZED/ SELF TESTING	Set Device Status attribute to appropriate value. Initiate self testing procedure from beginning.	Enter RUNNING/NOT EXECUTING/INITIALIZED/SELF TESTING.
4	RUNNING	Abort request.	ABORT	Abort response. Set Device Status attribute to appropriate value.	Abort request may be generated internally (e.g., as a result of Self Test catastrophic failure).
5	ABORT	Recover request; Device Status attribute value == ABORT FROM IDLE OREXECUTING.	IDLE	Recover response. Set Device Status attribute to appropriate value.	Enter IDLE state.
6	ABORT	Recover request; Device Status attribute value == ABORT FROM INITIALIZED/ SELF TESTING OR SELF TEST EXCEPTION.	INITIALIZED/SELF TESTING	Recover response. Initiate self testing procedure from beginning. Set Device Status attribute to appropriate value.	Enter RUNNING/NOT EXECUTING/ INITIALIZED/SELF TESTING.
7	NORMAL OPERATING	Reset request.	RUNNING	Reset response. DM Object is initialized. Set Device Status attribute to appropriate value.	Valid for all sub-states of NORMAL OPERATING. Move to RUNNING/NOT EXECUTING/ INITIALIZED/SELF TESTING.
8	IDLE	Execute service request (this service request may be internally generated).	EXECUTING	Execute response. Set Device Status attribute to appropriate value.	Device is now executing.
9	NORMAL OPERATING	Critical Fault detected.	CRITICAL FAULT	The object (and device) is in a fault state from which there is no recovery. Object services generally cannot be processed.	The mechanism for exit from a critical fault is device-specific.
10	NORMAL OPERATING	GetAttribute or Set Attribute request.	NORMAL OPERATING	Get or Set Attribute appropriate response.	Valid for all sub-states of NORMAL OPERATING.
11	NORMAL OPERATING	PerformDiagnostics request.	NORMAL OPERATING	PerformDiagnostics response. Diagnostic is performed if possible. Exceptions are set within DM object as necessary.	Conditions which are deemed exceptions are device-specific.
12	NORMAL OPERATING	Lock or Unlock request.	NORMAL OPERATING	Adjust write access to READ only attributes of all device objects as appropriate. Lock or Unlock service response.	The mechanism for exit from a critical fault is device-specific.
13	NORMAL OPERATING	GetExceptionQueue or Clear Exception Queue request.	NORMAL OPERATING	GetExceptionQueue response with appropriate exception queue data. Clear appropriate	Valid for all substates of NORMAL OPERATING.

				exception queue and send service response.	
14	NORMAL OPERATING	Invalid service request.	NORMAL OPERATING	Error response to appropriate request.	Valid for all sub-states of NORMAL OPERATING.
15	CRITICAL FAULT	Any service request.	CRITICAL FAULT	none	Object services generally cannot be processed.
16	ABORT	Abort or Execute request.	ABORT	Error response.	Not valid requests in this state.
17	RUNNING	Recover request.	RUNNING	Error response.	Recover request only valid while in ABORT state.
18	RUNNING	Internally generated PublishAttribute Notification Service.	RUNNING	Attribute Identified by Attribute ID in PublishAttribute Service is Published.	Notification only occurs if specific conditions are met (see Section 7.3.3).
19	EXECUTING	Execute Request.	EXECUTING	Error response.	Not a valid request in this state.
20	INITIALIZED/ SELF TESTING	Execute service request.	INITIALIZED/ SELF TESTING	Error response.	Not a valid request in this state.
21	SELF TEST EXCEPTION	Execute service request.	SELF TEST EXCEPTION	Error response.	Not a valid request in this state.

* Transitions 10 through 21 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

The following also applies to DM behavior:

A DM instance service may be requested internally by the DM object.

If a SetAttribute request is received over the network for an attribute that is not network-settable, the attribute value is not changed and an error service response is generated.

When processing an Abort, Recover, or Reset service request, the DM object issues a corresponding Abort, Recover, or Reset service request to the SAC object.

While in the Abort state, the DM object will not initiate communications over the network (e.g., it will not initiate communication of exceptions over the network). It may, however, communicate over the network in response to service requests.

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

The following identifies additional behavior associated with specific values of DM object attributes:

The Device Status attribute value indicates the state of the DM object (see Table 18). It is updated on appropriate state transitions within the DM object (see Table 29). Attribute values 1 through 6 represent valid states. A value of zero indicates that the DM state is unknown; conditions under which a zero value may occur are outside the scope of this document.

The Reporting Mode attribute determines five different conditions under which alarms and warnings are reported through communication of the Exception Status attribute (see Table 19). Both warnings and alarms have separate values within Reporting Mode. This allows alarms to be reported more aggressively than warnings because alarms are presumably more time-critical. The behavior associated with the five Reporting Mode conditions which are used for both alarms and warnings is described in the following paragraphs. Optionality indicates that the mode may or may not be supported; this optionality may be further specified in a sensor/actuator network specific device model and/or device manufacturer specification.

_ *Request* — When a bit in the status attribute transitions from cleared to set, the device will not report until requested. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared. In this mode, a device may go in and out of an exception condition without ever reporting the condition because there has been no request. The device must always respond to a request for exception conditions, regardless of Reporting Mode state.

_ Optional *RequestLatched* — When a bit in the status attribute transitions from cleared to set, the device will not report until requested. When the exception condition no longer exists, the exception status bit and the specific bit in the exception detail attribute are not cleared until they have been requested and reported. This mode guarantees that a brief occurrence of an exception condition will always be reported, barring an unrecoverable communications error. Within each exception variable, active bits will be cleared as they are reported.

_ Optional *EventTriggeredOn* — When a bit in the status attribute transitions from cleared to set, the device will automatically report the exception status attribute. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail variable are cleared.

_ Optional *EventTriggeredOn/Off* — When a bit in the status attribute transitions from cleared to set, the device will automatically report the exception status attribute. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared, and the device will again automatically report the exception status attribute.

_ Optional *TimeTriggered* — If this mode is supported, the Exception Status Timer attribute value indicates the time interval for periodic status reporting. When the indicated time period expires, the device will automatically report the exception status attribute. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared.

_ Optional *EventOnOrTimeTriggered* — This is a logical "or" of EventTriggeredOn mode and TimeTriggered mode. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared.

_ Optional *EventOn/OffOrTimeTriggered* — This is a logical "or" of EventTriggeredOn/Off mode and Time-Triggered mode. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared.

The Reporting Mode attribute value defaults to zero upon object initialization (i.e., request mode), unless otherwise specified in an appropriate sensor/actuator network specific device model specification.

The Reporting Mode attribute is defined as network-settable; however, it is only settable to (valid) values corresponding to supported reporting modes (see Table 19, appropriate sensor/actuator network specific device model specification, and device manufacturer specification). Note that at least one reporting mode, Request, must be supported. If an attempt is made, via a Set_Attribute service request, to set the Reporting Mode attribute to an invalid value, an error service response is returned.

All exceptions are communicated through reporting the values of the Exception Status and, conditionally, the Exception Detail Alarm and Exception Detail Warning attributes. An Exception Detail attribute may only be reported if both bit 7 and the corresponding exception bit of the Exception Status variable are set to 1.

If the Reporting Mode attribute indicates a reporting mode of TimeTriggered, EventOnOrTimeTriggered, or EventOn/OffOrTimeTriggered, the time interval for reporting is the value of the Exception Status Report Interval attribute. Time zero is defined as the time at which the Reporting Mode attribute is last modified utilizing the SetAttribute service. The first report is generated at time zero (not after the first time interval). If this value of the Exception Status Report Interval attribute is zero, then time triggered exception reporting is disabled.

If the Reporting Mode attribute indicates a reporting mode of EventOnOrTimeTriggered, or EventOn/OffOrTimeTriggered, the reporting associated with the occurrence of the appropriate event and the reporting associated with the report interval are independent (e.g., the occurrence of reporting due to an event does not impact the timing associated with reporting according to a specified time interval).

The Exception Status Report Interval attribute value defaults to 0.0 seconds upon object initialization, unless otherwise specified in an appropriate sensor/actuator network specific device model specification.

Additional behavior may be provided by a DM object that is device-specific. A complete definition of this additional behavior may be found in an appropriate sensor/actuator network specific device model specification.

The PublishAttribute service would be used for automatic reporting of the DM object exception status attribute when (1) an exception exists, (2) an automatic reporting mode is indicated by the Reporting Mode attribute (see Section 7.3.1), and (3) the indicated reporting mode is supported by the device (in its current state).

Behavior associated with the AlarmEnable and WarningEnable attributes is as follows. If the alarm/warning attribute is TRUE or if the attribute is not supported, alarm/warning reporting behavior as described elsewhere in this section is not impacted by this attribute. If the alarm/warning attribute is set to FALSE, all alarm/warning DM exception attributes are set and maintained at values that indicate a no alarm state. Thus, bits 0 through 2 of the ExceptionStatus attribute (alarms) or bits 4 through 6 (warnings) are set and maintained at zero, and any ExceptionDetail Alarm/Warning attribute bytes are set and maintained at a value of zero. When the Alarm/WarningEnable is toggled to TRUE, the device immediately determines its alarm/warning state, updates the appropriate DM attributes as necessary, and reports any alarms/warnings according to the mode defined by the ReportingMode attribute.

Behavior associated with the (Alarm and Warning) Exception Queues (see Sections 7.3.1.21 and 7.3.1.22), if supported, is as follows: Upon device initialization (start-up) or reset, both exception queues are cleared. The data type of queue elements is determined by the value of bit 7 of the status attribute (as explained in Section 7.3.1.21). As alarm events are issued to the DM object, the appropriate exception parameters are set (ExceptionStatus and ExceptionDetailAlarm/Warning) and the events are also logged onto the appropriate Alarm or Warning Exception Queue on a first-in, first-out basis. Any alarm or warning clear events are also logged onto these queues in the same manner; these events are logged in the same data format as the exception, but with the data value indicating that the exception no longer exists. The maximum number of elements that are retained by the queue is application-specific. Once the maximum queue length is reached, new exception events depose the oldest exception event from the queue in a first-in, first-out fashion (similar to a shift register). The GetExceptionQueue service may be used to retrieve any number or all exception events from either (alarm or warning exception) queue. If the value of the Clear attribute is TRUE, retrieved events are deposed from the queue; otherwise, they are left in the queue.

Behavior associated with the optional "Date and Time" attribute is as follows. This attribute indicates the current date and time as perceived by the device whenever it is reported. The resolution of the "Time of Day" reported shall be milliseconds unless the "Date and Time Type" attribute is supported and it indicates a resolution other than milliseconds.

7.4 *Sensor, Actuator, and Controller Objects* — The Sensor (S), Actuator (A), and Controller (C) objects are the model components that model the high level sensory, actuation, and/or control element capabilities of the device as viewed from the network. Although the structure and behavior of these elements can vary widely from device to device, there are common aspects of structure and behavior that can be identified. Specifically, there are structure and behavior characteristics common to all active elements (i.e., all S, A, and C objects), common to all elements of an element type (i.e., all S objects, A objects, or C objects), or common to all elements of an element sub-type (e.g., all analog input sensor objects). This commonality of structure and behavior among these object types is depicted in the class generalization hierarchy of Figure 7. With this generalization hierarchy in place, objects inherit structure and behavior from their parent object classes, thus reducing the level of redundancy in description. For example, object instances of the Sensor-AI class inherit (i.e., also have) the attributes, services, and behavior of the Sensor and Active Element parent classes.

Note that the only classes from which accessible object instances can be created in this hierarchy exist at the lowest level (i.e., Sensor-AI, Sensor-BI, Actuator-AO, Actuator-BO, and Controller objects). Also note that the object identifiers for these object instances are designated as indicated in Table 1 (e.g., a Sensor-AI object instance has an Object Identifier of SenI'n', where 'n' is the instance number of the sensor object for the device).

In the following sub-sections, the attributes, services, and behavior at each level of the active element class generalization hierarchy are identified and defined. Note that the specific device model would complete the definition of each sensor, actuator, and controller element of a specific device type through definition of additional structure and behavior for the S, A, and C object instances that collectively realize the sensory/ actuation/control capability portion of the device model.

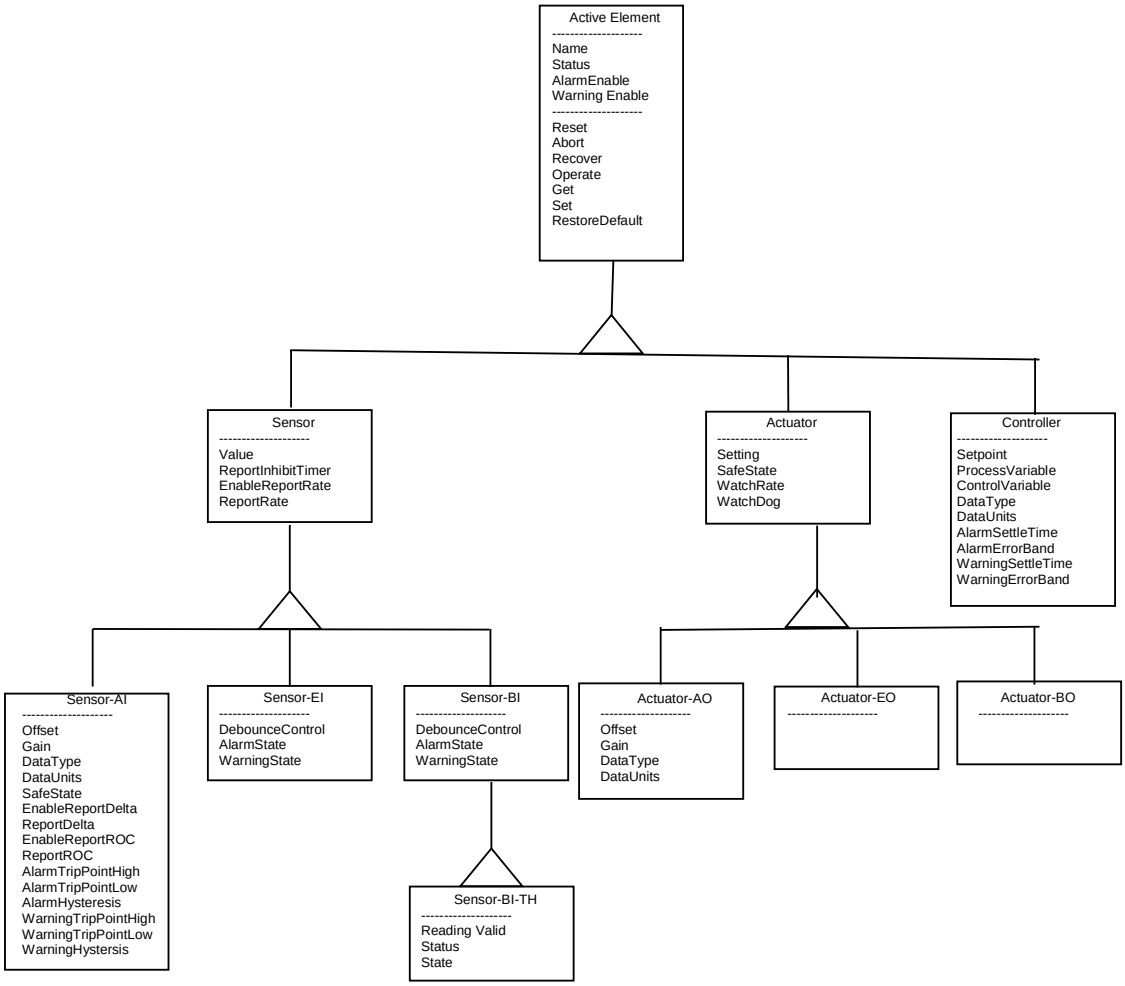


Figure 7
Active Element Class Generalization Hierarchy

7.4.1 *Active Element (AE) Class* — The AE class at the top of the active element generalization hierarchy contains structure and behavior common to all active element instances in a device.

7.4.1.1 *Active Element Class Attributes* — All required and "common optional" AE class attributes are listed in Table 30 and described below. For an explanation of the table format, see Section 7.3.1.

Table 30 AE Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Name	nA1**	RO	N	Text, 16 characters maximum
Status	nA2	RO	Y	Context-specific
AlarmEnable	nA3	RW	N	Boolean
WarningEnable	nA4***	RW	N	Boolean

* A "Definition" column is not included. For attribute definitions, see text.

** Where 'n' is the type of the object instance to which the particular attribute is associated (e.g., Sai, Sbi, Aao, Ado, or C), see Sections 7.4.4 through 7.4.8.

*** ID's nA5 through nA15 are reserved for future AE class attribute definition.

7.4.1.1.1 *Name* — A label used to identify the active element associated with the object instance.

7.4.1.1.2 *Status* — An indication of the state of the active element associated with the object instance. The form and interpretation of the value of this attribute is context-specific.

7.4.1.1.3 *AlarmEnable* — An attribute which specifies whether alarm exception conditions of the active element associated with the object instance are to be reported. Reporting is enabled if this variable is set to TRUE. The method of reporting is beyond the scope of this document.

7.4.1.1.4 *WarningEnable* — An attribute which specifies whether warning exception conditions of the active element associated with the object instance are to be reported. Reporting is enabled if this variable is set to TRUE. The method of reporting is beyond the scope of this document.

7.4.1.2 *Active Element Class Services* — All Active Element common services are listed in Table 31 (the specific AE object is the "service provider" of these services). Note that all of these services are Request services. Service parameters that are specified in this standard are summarized in Table 32.

Table 31 AE Class Service Resource Definition

Service	*Service ID*	*Type*	*Description*
Reset	nS1*	R	Used to place object in INITIALIZED state.
Abort	nS2	R	Used to place object in ABORT state.
Recover	nS3	R	Used to move object from ABORT state to RECOVERED state.
Operate	nS4	R	Used to move object to the operating state from INITIALIZING, or RECOVERING state.
GetAttribute	nS5	R	Used to read object attribute.
SetAttribute	nS6	R	Used to modify object attribute.
RestoreDefault	nS7**	R	Used to restore object attributes to their default values.

* Where 'n' is the type of the object instance to which the particular service is associated (e.g., Sai, Sbi, Aao, Ado, or C), see Sections 7.4.4 through 7.4.8.

** ID's nS8 through nS15 are reserved for future AE class service definition.

Table 32 AE Class Service Parameter Dictionary

Parameter	*Form*	*Description*
AttributeID	Positive Integer	Object instance Attribute Identifier.
AttributeValue	Context-specific	Value of object instance attribute.
RestoreConditions	Byte enumerated	Level of restore requested.

A detailed description of these services follows:

7.4.1.2.1 *Reset* — Used to place the object instance in its INITIALIZED state. There are no parameters specified for this service.

7.4.1.2.2 *Abort* — Used to place the object instance in its ABORT state. There are no parameters specified for this service.

7.4.1.2.3 *Recover* — Used to move the object instance from its ABORT state to a RECOVERED state. There are no parameters specified for this service.

7.4.1.2.4 *Operate* — Used to move the object instance to the OPERATING state from the INITIALIZING or RECOVERING state. There are no parameters specified for this service.

7.4.1.2.5 *GetAttribute* — Used to read a value of an object instance attribute over the network. The following table describes the parameters specified for this service.

GetAttribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Form*	*Description*
AttributeID	M	C**, =	Network-Specific	Attribute Identifier of attribute whose value is being requested. Inclusion of attribute in Rsp/Conf. is optional.
AttributeValue	-	M	Context-Specific	Value of attribute requested.

* See Section 6.4.3 for further description of terminology used in this table.

** The determination of whether Attribute ID is supplied in the service response/confirm is outside the scope of this document.

7.4.1.2.6 *SetAttribute* — Used to modify the value of an object instance attribute over the network which has been identified as modifiable. The following table describes the parameters specified for this service.

SetAttribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Form*	*Description*
AttributeID	M	C, =	Network-Specific	Attribute Identifier of attribute whose value is to be modified. Inclusion of Attribute in Rsp/Conf. is optional.
AttributeValue	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.

* See Section 6.4.3 for further description of terminology used in this table.

7.4.1.2.7 *RestoreDefault* — Used to restore attributes of this object instance to their default values. The following table describes the parameters specified for this service.

RestoreDefault Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Form*	*Description*
Restore Conditions	M	C	byte, enumerated	0 = Restore specific attribute of this object 1 = Restore all attributes of this object 2 – 63 = Reserved 64 – 255 = Application-Defined
AttributeID	C	C	Network-Specific	Attribute ID of the attribute whose value is being restored.

* See Section 6.4.3 for further description of terminology used in this table.

7.4.1.3 *Active Element Class Behavior* — The AE class behavior is illustrated in Figure 8. Associated states and state transitions are described in Tables 33 and 34. The following also applies to AE class behavior:

An AE class service may be requested internally by the AE object instance.

Note that the behavior associated with the Reset, Abort, Recover, Operate, and Restore Default services differs from the SAC object instance behavior in that no service requests are necessarily issued to any other object instances.

Upon instantiation or upon receipt of a reset request, an AE object instance responds by entering or staying in its INITIALIZED state.

The object INITIALIZING application process is described as follows: When a request to initialize is received (e.g., through a Reset service request), all object instance attributes are restored to their initial value; the interpretation of these values and how they are stored/retrieved is beyond the scope of this document. The object performs any manufacturer-specified self tests and diagnostics to determine if it is qualified to enter the OPERATING state. Upon completion of these tests, the appropriate Pass or Fail service response is reported to the requesting object instance. The content, format, and protocol of this service response are specified by the network for requests made over the network.

Upon receipt of an abort request, an AE object instance responds by entering or staying in its ABORT state.

Upon receipt of a recover request, an AE object instance 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.

The object RECOVERING application process is described as follows: Upon receipt of a Recover service request, the object performs any manufacturer-specified self tests and diagnostics to determine if it is qualified to enter the OPERATING state.

Upon completion, the appropriate Pass or Fail service response is reported to the requesting object instance. The content, format, and protocol of this service response are specified by the network for requests made over the network.

The AlarmEnable attribute determines whether or not alarms are reported through the ExceptionStatus and ExceptionDetail attributes of the DM object (see Section 7.3). If the AlarmEnable attribute is set to the enabled state, then any alarm is reported to the DM object instance upon its occurrence. The WarningEnable attribute works analogously with respect to reporting a warning.

Additional behavior may be exhibited by an AE object instance that is defined elsewhere in its class generalization hierarchy, or is device-specific. A complete definition of AE object type instance behavior includes the appropriate sensor/actuator network specific device model specification.

Behavior associated with receipt of an invalid service request identified in this document applies to all services required for this class. Unless otherwise specified (e.g., in an appropriate sensor/actuator network specific device model specification), behavior associated with the receipt of an invalid service request for additional services defined (elsewhere) for this class shall be the same as that for required services.

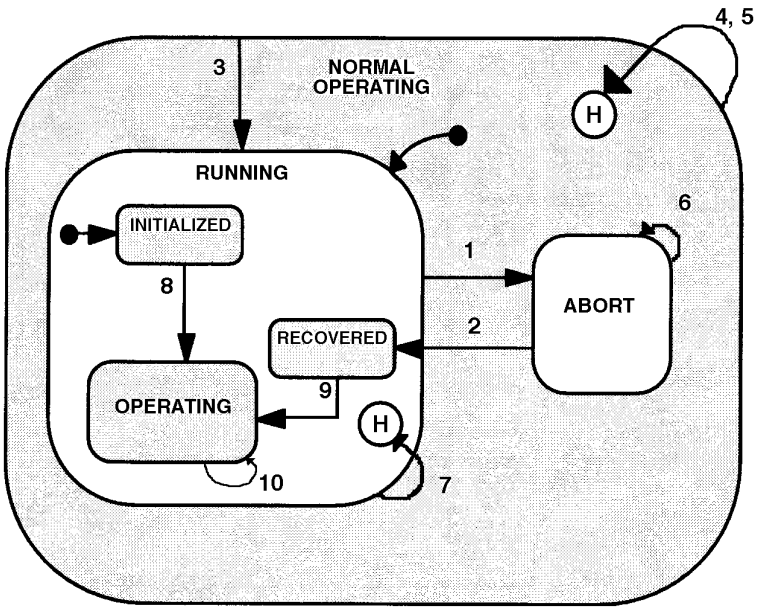


Figure 8
AE Class Behavior

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

Table 33 AE Class Behavior State Description

State	*Description*
NORMAL OPERATING	Object instance exists. Object services can be processed.
RUNNING	This is the entry sub-state to NORMAL OPERATING. Object instance is not in its ABORT state.
INITIALIZED	This is the entry sub-state 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 sub-states within RUNNING except INITIALIZED and RECOVERED. The further delineation of this sub-state and transition into this sub-state is outside the scope of this document.
ABORT	Object instance is in an aborted state and is not running any application process; a detailed description of this state is outside the scope of this document.

Table 34 AE Class Behavior State Transition Matrix*

#	*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 sub-states of NORMAL OPERATING. Move to RUNNING/ INITIALIZED.
4	NORMAL OPERATING	GetAttribute, SetAttribute, RestoreDefault.	NORMAL OPERATING	Get Attribute, Set Attribute, Restore Defaults. Appropriate Response.	Valid for all sub-states of NORMAL OPERATING.
5	NORMAL OPERATING	Invalid service request.	NORMAL OPERATING	Error response.	Valid for all substates 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	INITIALIZING	Operate request.	OPERATING	Device-specific.	Behavior associated with this transition may be further specified in an appropriate sensor/ actuator network specific device model specification.
9	RECOVERED	Operate request.	OPERATING	Device-specific.	Behavior associated with this transition may be further specified in an appropriate sensor/actuator network specific device model.
10	OPERATING	Operate request.	OPERATING	Operate response.	Object remains in the OPERATING state. Additional behavior associated with this transition may befurther specified in an appropriate sensor/actuator network specific device model.

* NOTE: Only generic behavior is indicated in this table.

7.4.2 *Sensor (S) Class* — The S class at the second level of the active element generalization hierarchy contains structure and behavior common to all sensing element instances in a device.

7.4.2.1 *Sensor Class Attributes* — All required and "common optional" S class attributes are listed in Table 35 and described below. For an explanation of the table format, see Section 7.3.1. Note that the S class also inherits all AE class attributes.

Table 35 S Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Value	nA16**	RO	Y	Context-Specific
ReportInhibitTimer	nA17	RW	N	Context-Specific
EnableReportRate	nA18	RW	N	Boolean
ReportRate	nA19***	RW	N	Context-Specific

* A "Definition" column is not included. For attribute definitions, see text.

** Where 'n' is the type of the object instance to which the particular attribute is associated (e.g., Sai or Sbi), see Sections 7.4.4 through 7.4.8.

*** ID's nA20 through nA63 are reserved for future S class attribute definition.

7.4.2.1.1 *Value* — The value of the sensor (associated with the object instance) that is to be conveyed. The form and content of this attribute is context-specific.

7.4.2.1.2 *ReportInhibitTimer* — An attribute which is used to identify the absolute minimal time interval (i.e., maximum rate) for reporting the attribute "Value"; expressed in seconds. A value of zero indicates that there is no minimal time interval.

7.4.2.1.3 *EnableReportRate* — An attribute which enables the reporting of the attribute "Value" based on the "ReportingRate" attribute value; a value of TRUE signifies that the reporting at the "ReportingRate" is enabled.

7.4.2.1.4 *ReportRate* — An attribute which defines the time interval at which the attribute "Value" will be reported; expressed in seconds.

7.4.2.2 *Sensor Class Services* — No S class common services are defined. Note that the S class inherits all AE class services.

7.4.2.3 *Sensor Class Behavior* — The following applies to S class behavior:

Some form of measurement or other sensory activity is made by the active sensory element. This measurement is converted into the appropriate data and becomes the value of the "Value" attribute. The method of conversion is beyond the scope of this document.

No other S class common behavior is defined. Note that the S class inherits all AE class-defined behavior.

7.4.3 *Actuator (A) Class* — The A class at the second level of the active element generalization hierarchy contains structure and behavior common to all actuator element instances in a device.

7.4.3.1 *Actuator Class Attributes* — All required and "common optional" A class attributes are listed in Table 36 and described below. For an explanation of the table format, see Section 7.3.1. Note that the A class also inherits all AE class attributes.

Table 36 A Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Setting	nA16**	RW	Y	Context-Specific
SafeState	nA17	RW	N	Context-Specific
WatchRate	nA18	RW	N	Unsigned Integer < 65,536
WatchDog	nA19***	RW	N	Boolean

* A "Definition" column is not included. For attribute definitions, see text.

** Where 'n' is the type of the object instance to which the particular attribute is associated (e.g., Aao or Ado), see Sections 7.4.4 through 7.4.8.

*** ID's nA20 through nA63 are reserved for future A class attribute definition.

7.4.3.1.1 *Setting* — Specifies the value that is to be actuated by the actuator associated with the object instance. The form and content of this attribute is context-specific.

7.4.3.1.2 *SafeState* — Specifies the state in which the actuator will be placed for ABORT and INITIALIZING states. Unless otherwise specified, this attribute is a Boolean type corresponding to Zero (or "no output") or One (or "full output").

7.4.3.1.3 *WatchRate* — Specifies the rate at which the "Setting" attribute is to be refreshed (i.e., set externally). If the "Setting" attribute is not refreshed at this rate, it reverts to its default value; expressed in (integer/100) hertz.

7.4.3.1.4 *WatchDog* — Enables or disables the capability of setting the "Setting" attribute to its default value due to a refresh rate lower than specified by the "WatchRate" attribute value; a value of TRUE signifies that the WatchRate functionality is enabled.

7.4.3.2 *Actuator Class Services* — No A class common services are defined. Note that the A class inherits all AE class services.

7.4.3.3 *Actuator Class Behavior* — The following applies to A class behavior:

Some form of actuation activity is made by the active actuation element. The value of the "Setting" attribute is converted into the appropriate signal to cause the desired actuation.

A class-defined behavior associated with the "WatchRate" and "WatchDog" attributes is defined in Section 7.4.3.1 (above). No other A class common behavior is defined. Note that the A class inherits all AE class-defined behavior.

7.4.4 *Controller (C) Class* — The C class at the second level of the active element generalization hierarchy contains structure and behavior common to all controller element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device type (see Figure 7), object instances can be created as instances of this class type.

7.4.4.1 *Controller Class Attributes* — All required and "common optional" C class attributes are listed in Table 37 and described below. For an explanation of the table format, see Section 7.3.1. Note that the C class also inherits all AE class attributes.

Table 37 C Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Setpoint	CA16	RW	Y	Context-Specific
ProcessVariable	CA17	RW	N	Context-Specific
ControlVariable	CA18	RW	N	Context-Specific
DataType	CA19	RW	N	Context-Specific
AlarmSettleTime	CA21	RW	N	Context-Specific
AlarmErrorBand	CA22	RW	N	Context-Specific
WarningSettleTime	CA24	RW	N	Context-Specific
WarningErrorBand	CA25	RW	N	Context-Specific

* A "Definition" column is not included. For attribute definitions, see text.

** ID's CA27 through CA63 are reserved for future C class attribute definition.

7.4.4.1.1 *Setpoint* — This attribute specifies the value to which the Process Variable will be controlled via the Control Variable. This is one of the inputs of a classic single-stage closed-loop controller. The setpoint attribute value is expressed in units identified by the "DataUnits" attribute.

7.4.4.1.2 *ProcessVariable* — This attribute value is the measurement of the process being controlled. This is one of the inputs of a classic single-stage closed-loop controller.

7.4.4.1.3 *ControlVariable* — This attribute value is the control signal being sent to the process controlling element. This is the output of a classic single-stage closed-loop controller.

7.4.4.1.4 *DataType* — An attribute which defines the format of the data associated with "Value" attribute (inherited) of the object instance.

7.4.4.1.5 *DataUnits* — An attribute which defines the current units associated with "Value" attribute (inherited) of the object instance.

7.4.4.1.6 *AlarmSettleTime* — An attribute which specifies a time used in the determination of a Controller object instance alarm exception condition. This value is expressed in units of seconds. The minimum allowable value for this attribute is specified by the manufacturer. An out-of-range error will result from an attempt to set this attribute to a value lower than that specified as the minimum allowable.

7.4.4.1.7 *AlarmErrorBand* — An attribute which specifies an amount by which the process variable may not equal the setpoint used in the determination of a Controller object instance alarm exception condition. This value is expressed in units identified by the "DataUnits" attribute and in a format specified by the "DataType" attribute.

7.4.4.1.8 *WarningSettleTime* — An attribute which specifies a time used in the determination of a Controller object instance warning exception condition. This value is expressed in units of seconds. The minimum allowable value for this attribute is specified by the manufacturer. An out-of-range error will result from an attempt to set this attribute to a value lower than that specified as the minimum allowable.

7.4.4.1.9 *WarningErrorBand* — An attribute which specifies an amount by which the process variable may not equal the setpoint used in the determination of a Controller object instance warning exception condition. This value is expressed in units identified by the "DataUnits" attribute and in a format specified by the "DataType" attribute.

7.4.4.2 *Controller Class Services* — No C class common services are defined. Note that the C class inherits all AE class services.

7.4.4.3 *Controller Class Behavior* — The following applies to C class behavior:

The controller element uses the value of the Process-Variable and Setpoint attributes and determines the error signal. It then adjusts the ControlVariable attribute value as necessary and appropriate to minimize this error signal. The method of, and conditions for, adjustment are beyond the scope of this document.

Note that the C class inherits all AE class-defined behavior.

7.4.5 *Sensor-Analog Input (SAI) Class* — The SAI class at the third level of the active element generalization hierarchy contains structure and behavior common to all analog input sensor element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.5.1 *Sensor-Analog Input Class Attributes* — All required and "common optional" SAI class attributes are listed in Table 38 and described below. For an explanation of the table format, see Section 7.3.1. Note that the SAI class also inherits all AE and S class attributes.

Table 38 SAI Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Offset	SaiA64	RW	N	Context-Specific
Gain	SaiA65	RW	N	Context-Specific
DataType	SaiA66	RW	N	Context-Specific
DataUnits	SaiA67	RW	N	Context-Specific
SafeState	SaiA68	RW	N	Enumerated Byte
EnableReportDelta	SaiA69	RW	N	Boolean
ReportDelta	SaiA70	RW	N	Context-Specific
EnableReportROC	SaiA71	RW	N	Boolean
ReportROC	SaiA72	RW	N	Context-Specific
AlarmTripPointHigh	SaiA73	RW	N	Context-Specific
AlarmTripPointLow	SaiA74	RW	N	Context-Specific
AlarmHystersis	SaiA75	RW	N	Context-Specific
WarningTripPointHigh	SaiA76	RW	N	Context-Specific
WarningTripPointLow	SaiA77	RW	N	Context-Specific
WarningHystersis	SaiA78**	RW	N	Context-Specific

* A "Definition" column is not included. For attribute definitions, see text.

** ID's SaiA78 through SaiA127 are reserved for future Sai class attribute definition.

7.4.5.1.1 *Offset* — An attribute which specifies the amount of offset correction being applied to derive the value of the "Value" attribute (inherited) of the object instance. It is expressed in terms of the units specified by the "DataUnits" attribute.

7.4.5.1.2 *Gain* — An attribute which is used to define the range of the "Value" attribute (inherited) of the object instance. The method of conveying this range with this attribute is beyond the scope of this document.

7.4.5.1.3 *DataType* — An attribute which defines the format of the data associated with "Value" attribute (inherited) of the object instance.

7.4.5.1.4 *DataUnits* — An attribute which defines the current units associated with "Value" attribute (inherited) of the object instance.

7.4.5.1.5 *SafeState* — An attribute which defines the Value (inherited attribute) to be reported by the object instance when it is in a non-OPERATING state. The enumeration of the SafeState attribute is given in Table 39.

Table 39 SafeState Attribute Values

Attribute Value	*DM Object State*
0	Zero
1	Full-Scale
2	LVV
3	Undefined
4 – 63	Reserved
64 – 255	Open

7.4.5.1.6 *EnableReportDelta* — An attribute which enables the reporting of the attribute "Value" based on the "ReportDelta" attribute value (see below); a value of TRUE signifies that reporting using the "ReportDelta" method is enabled.

7.4.5.1.7 *ReportDelta* — The value of this attribute indicates the amount by which the sensor value (indicated by the "Value" attribute (inherited)) must change before it is published. When the value of the EnableReportDelta attribute indicates that this method of reporting is "enabled", and when the value of the "Value" attribute changes by an amount in excess of the amount indicated by the ReportDelta attribute value, the object will publish the Value attribute.

7.4.5.1.8 *EnableReportROC* — The Enable ReportROC (rate of change) attribute enables the reporting of the attribute "Value" based on the "ReportROC" attribute value (see below); a value of TRUE signifies that reporting using the rate of change method is enabled.

7.4.5.1.9 *ReportROC* — The value of this attribute indicates the rate by which the sensor value (indicated by the "Value" attribute (inherited)) must change before it is published. The attribute value context is DataUnits/Time; the specific context of DataUnits is defined in the DataUnits attribute, while the unit of Time is context-specific. When the value of the EnableReportDelta attribute indicates that this method of reporting is "enabled", and when the value of the "Value" attribute changes at a rate in excess of the amount indicated by the ReportDelta attribute value, the object will publish the Value attribute.

7.4.5.1.10 *AlarmTripPointHigh* — An attribute which specifies the value above which the object instance "Value" attribute (inherited) value must be to generate an alarm status indication. The behavior associated with the generation of an alarm status indication (beyond that described in Section 7.4.1.3) is beyond the scope of this document.

7.4.5.1.11 *AlarmTripPointLow* — An attribute which specifies the value below which the object instance "Value" attribute (inherited) value must be to generate an alarm status indication. The behavior associated with the generation of an alarm status indication (beyond that described in Section 7.4.1.3) is beyond the scope of this document.

7.4.5.1.12 *AlarmHystersis* — An attribute which specifies the amount of hysteresis to be applied in the clearing of an alarm condition. For example: a Trip Point High value of 100 with a hysteresis value of 2 will result in an exception condition being set when the Value attribute exceeds 100 in the positive direction and cleared when the Value attribute exceeds 98 in the negative direction. Similarly, a Trip Point Low value of 100 with a hysteresis value of 2 will result in an exception condition being set when the Value attribute exceeds 100 in the negative direction and cleared when the Value attribute exceeds 102 in the positive direction. The behavior associated with the generation and clearing of exception conditions is beyond the scope of this document.

7.4.5.1.13 *WarningTripPointHigh* — An attribute which specifies the value above which the object instance "Value" attribute (inherited) value must be to generate a warning status indication. The behavior associated with the generation of a warning status indication is beyond the scope of this document.

7.4.5.1.14 *WarningTripPointLow* — An attribute which specifies the value below which the object instance "Value" attribute (inherited) value must be to generate a warning status indication. The behavior associated with the generation of a warning status indication is beyond the scope of this document.

7.4.5.1.15 *WarningHystersis* — An attribute which specifies the amount of hysteresis to be applied in the clearing of an alarm condition (see Section 7.4.5.1.12). The behavior associated with the generation and clearing of exception conditions is beyond the scope of this document.

7.4.5.2 *Sensor-Analog Input Class Services* — No SAI class common services are defined. Note that the SAI class inherits all AE and S class services.

7.4.5.3 *Sensor-Analog Input Class Behavior* — No SAI class common behavior is defined. Note that the SAI class inherits all AE and S class-defined behavior.

7.4.6 *Sensor-Binary Input (SBI) Class* — The SBI class at the third level of the active element generalization hierarchy contains structure and behavior common to all binary input sensor element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.6.1 *Sensor-Binary Input Class Attributes* — All required and "common optional" SBI class attributes are listed in Table 40 and described below. For an explanation of the table format, see Section 7.3.1. Note that the SBI class also inherits all AE and S class attributes.

Table 40 SBI Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
DebounceControl	SbiA64	RW	N	Context-specific
AlarmState	SbiA65	RW	N	Boolean
WarningState	SbiA66**	RW	N	Boolean

* A "Definition" column is not included. For attribute definitions, see text.

** ID's Sbi67 through SbiA127 are reserved for future Sbi class attribute definition.

7.4.6.1.1 *DebounceControl* — This attribute is used to control sensitivity to 'false' transitions. The method of utilization of this attribute for debounce control is application-specific.

7.4.6.1.2 *AlarmState* — The value of the Value attribute (i.e., TRUE or FALSE) that will cause an alarm condition.

7.4.6.1.3 *WarningState* — The value of the Value attribute (i.e., TRUE or FALSE) that will cause a warning condition.

7.4.6.2 *Sensor-Binary Input Class Services* — No SBI class common services are defined. Note that the SBI class inherits all AE and S class services.

7.4.6.3 *Sensor-Binary Input Class Behavior* — No SBI class common behavior is defined. Note that the SBI class inherits all AE and S class-defined behavior.

7.4.6.4 *Sensor-Enumerated Input Class Services* — No SEI class common services are defined. Note that the SEI class inherits all AE and S class services.

7.4.6.5 *Sensor-Enumerated Input Class Attributes* — All required and "common optional" SEI class attributes are listed in Table 41 and described below. For an explanation of the table format, see Section 7.3.1. Note that the SEI class also inherits all AE and S class attributes.

Table 41 SEI Class Attributes*

Attribute Name	*Attribute Identifier*	*Access Network*	*Rqmt*	*Form*
DebounceControl	SeiA64	RW	N	Context-Specific
AlarmState	SeiA65	RW	N	Enumerated
WarningState	SeiA66	RW	N	Enumerated

* A "Definition" column is not included. For attribute definitions, see text.

7.4.6.5.1 *DebounceControl* — This attribute is used to control sensitivity to 'false' transitions. The method of utilization of this attribute for debounce control is application-specific.

7.4.6.5.2 *AlarmState* — The value of the Value attribute that will cause an alarm condition.

7.4.6.5.3 *WarningState* — The value of the Value attribute that will cause a warning condition.

7.4.7 *Actuator-Analog Output (AAO) Class* — The AAO class at the third level of the active element generalization hierarchy contains structure and behavior common to all analog output actuator element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.7.1 *Actuator-Analog Output Class Attributes* — All required and "common optional" AAO class attributes are listed in Table 42 and described below. For an explanation of the table format, see Section 7.3.1. Note that the AAO class also inherits all AE and A class attributes.

Table 42 AAO Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Offset	AaoA64	RW	N	Context-specific
Gain	AaoA65	RW	N	Context-specific
DataType	AaoA66	RW	N	Context-specific
DataUnits	AaoA67**	RW	N	Context-specific

* A "Definition" column is not included. For attribute definitions, see text.

** ID's AaoA68 through AaoA127 are reserved for future Aao class attribute definition.

7.4.7.1.1 *Offset* — An attribute which specifies the amount of offset correction being applied to derive the value of the "Value" attribute (inherited) of the object instance. It is expressed in terms of the units specified by the "DataUnits" attribute.

7.4.7.1.2 *Gain* — An attribute which is used to define the range of the "Value" attribute (inherited) of the object instance. The method of conveying this range with this attribute is beyond the scope of this document.

7.4.7.1.3 *DataType* — An attribute which defines the format of the data associated with "Value" attribute (inherited) of the object instance.

7.4.7.1.4 *DataUnits* — An attribute which defines the current units associated with "Value" attribute (inherited) of the object instance.

7.4.7.2 *Actuator-Analog Output Class Services* — No AAO class common services are defined. Note that the AAO class inherits all AE and A class services.

7.4.7.3 *Actuator-Analog Output Class Behavior* — No AAO class common behavior is defined. Note that the AAO class inherits all AE and A class-defined behavior.

7.4.8 *Actuator-Binary Output (ABO) Class* — The ABO class at the third level of the active element generalization hierarchy contains structure and behavior common to all binary output actuator element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.8.1 *Actuator-Binary Output Class Attributes* — No ABO class common attributes are defined. Note that the ABO class inherits all AE and A class attributes.

7.4.8.2 *Actuator-Binary Output Class Services* — No ABO class common services are defined. Note that the ABO class inherits all AE and A class services.

7.4.8.3 *Actuator-Binary Output Class Behavior* — No ABO class common behavior is defined. Note that the ABO class inherits all AE and A class-defined behavior.

7.4.9 *Actuator-Enumerated Output (AEO) Class* — The AEO class at the third level of the active element generalization hierarchy contains structure and behavior common to all Enumerated output actuator element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.9.1 *Actuator-Enumerated Output Class Attributes* — No AEO class common attributes are defined. Note that the AEO class inherits all AE and A class attributes.

7.4.9.2 *Actuator-Enumerated Output Class Services* — No AEO class common services are defined. Note that the AEO class inherits all AE and A class services.

7.4.9.3 *Actuator-Enumerated Output Class Behavior* — No AEO class common behavior is defined. Note that the AEO class inherits all AE and A class-defined behavior.

7.4.10 *Sensor-Binary Input-Threshold (SBITH) Class* — The SBITH class at the fourth level of the active element generalization hierarchy contains structure and behavior common to all binary input threshold sensor element instances in a device.

7.4.10.1 *Sensor-Binary Input Threshold Class Attributes* — All required and "common optional" SBITH class attributes are listed in Table 43 and described below. For an explanation of the table format see Section 7.3.1. Note that the SBITH class also inherits all AE, S, an SBI class attributes.

Table 43 SBITH Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Reading Valid	SbithA64	R	N	Enumerated, Byte
State	SbithA65	R	N	Enumerated, Byte
Status	SbithA66	R	N	Enumerated, Byte

* A "Definition" column is not included. For attribute definitions, see text.

** ID's Sbith67 through Sbith127 are reserved for future Sbith class attribute definition.

7.4.10.1.1 *Reading Valid* — This attribute is used to specify the validity of the "*Value*" attribute of the SBITH object. The method of utilization of this attribute is application specific.

7.4.10.1.2 *State* — This attribute is used to record the current state of the SBITH object. The method of utilization of this attribute is application specific.

7.4.10.1.3 *Status* — This attribute is used to record the current active reporting status of the SBITH object. The method of utilization of this attribute is application specific.

7.4.10.2 *Sensor-Binary Input-Threshold Class Services* — No SBITH class common services are defined. Note that the SBITH class inherits all AE, S, and SBI class services.

7.4.10.3 *Sensor-Binary Input-Threshold Class Behavior* — No SBITH class common behavior is defined. Note that the SBITH class inherits all AE, S, and SBI class behavior.

APPENDIX 1
DATA UNIT ENUMERATION

NOTE: This appendix was approved as an official part of SEMI E54.1 by full letter ballot procedure.

The Data Units data type term is defined in Section 5.2. The following is a partial enumeration of that data type; further enumeration may be provided in an application-specific context.

Table A1-1 General

Data Units	*Description*
0	Counts
1	Counts per Second
2	Counts per Millisecond
3	Counts per Microsecond
4	Counts per Minute
5	Counts per Hour
6	Counts per Day
7	Percent
8–191	Reserved
192–255	Open

Group 1 — Time and Frequency (256–511)

Table A1-2 Time and Frequency

Data Units	*Description*
256	Seconds
257	Milliseconds
258	Microseconds
259	Minutes
260	Hours
261	Days
262	Hertz
263	KiloHertz
264	MegaHertz
265	GigaHertz
266–447	Reserved
448–511	Open

Group 2 — Temperature (512–767)

Table A1-3 Temperature

Data Units	*Description*
512	Centigrade/Celsius
513	Fahrenheit
514	Kelvin
515–703	Reserved
704–767	Open

Group 3 — Pressure (768–1023)

Table A1-4 Pressure

Data Units	*Description*
768	PSI
769	Torr
770	MilliTorr
771	mm Hg (0°C)
772	inches Hg (0°C)
773	cm H_2O (25°C)
774	inches H_2O (25°C)
775	Bar
776	MilliBar
777	Pascal
778	KiloPascal
779	Atmosphere
780–959	Reserved
960–1023	Open

Group 4 — Flow (1024–1279)

Table A1-5 Flow

Data Units	*Description*
1024	SCCM
1025	SLM
1026	CFM
1027	Pa-m^3/s
1028–1215	Reserved
1216–1279	Open

Group 5 — Electrical (1280–1535)

Table A1-6 Electrical

Data Units	*Description*
1280	Volts
1281	MilliVolts
1282	MicroVolts
1283	NanoVolts
1284	PicoVolts
1285	FemtoVolts
1286	KiloVolts
1287	MegaVolts
1288	GigaVolts
1290	Amps
1291	MilliAmps

1292	MicroAmps
1293	NanoAmps
1294	PicoAmps
1295	FemtoAmps
1296	KiloAmps
1297	MegaAmps
1298	GigaAmps
1300	Ohms
1301	MilliOhms
1302	MicroOhms
1303	NanoOhms
1304	PicoOhms
1305	FemtoOhms
1306	KiloOhms
1307	MegaOhms
1308	GigaOhms
1310	Farads
1311	MilliFarads
1312	MicroFarads
1313	NanoFarads
1314	PicoFarads
1315	FemtoFarads
1320	Henries
1321	MilliHenries
1322	MicroHenries
1323	NanoHenries
1324	PicoHenries
1325	FemtoHenries
1326–1471	Reserved
1472–1535	Open

APPENDIX 2
ERROR RESPONSES

NOTE: This appendix was approved as an official part of SEMI E54.1 by full letter ballot procedure.

The following is a listing of common service error responses. The details of presentation of these responses over the network (such as response codes) are network-specific and, thus, are not part of this document. Also, note that there may be additional service error responses delineated in the appropriate SDM specification or manufacturer specification.

A2-1 Error Responses

A2-1.1 *Resource Unavailable* — Resources needed for the object to perform the requested service are unavailable.

A2-1.2 *Service Not Supported* — The requested service is not implemented or is not defined for this Object Class/Instance.

A2-1.3 *Invalid Attribute Value* — Invalid attribute data detected.

A2-1.4 *Already in Requested Mode/State* — The object is already in the mode/state being requested by the service.

A2-1.5 *Object State Conflict* — The object cannot perform the requested service in its current mode/state.

A2-1.6 *Attribute Not Settable* — A request to modify a non-modifiable attribute was received.

A2-1.7 *Device State Conflict* — The device's current mode/state prohibits the execution of the requested service.

A2-1.8 *Service Parameter Data Not Complete* — Not enough data supplied with the service request.

A2-1.9 *Attribute Not Supported* — The attribute specified in the request is not supported.

A2-1.10 *Too Much Data for Service Parameters* — The service supplied more data than was expected.

A2-1.11 *Object Does Not Exist* — The object specified does not exist in the device.

A2-1.12 *Vendor-Specific Error* — Vendor-specific error.

A2-1.13 *Invalid Service Parameter* — A parameter associated with the request was invalid.

SEMI E54.2-0698 (Reapproved 0704)
GUIDE FOR WRITING SENSOR/ACTUATOR NETWORK (SAN) STANDARD BALLOTS

This guide was technically reapproved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 14, 2004. Initially available at www.semi.org May 2004; to be published July 2004. Originally published June 1998.

1 Purpose

1.1 This guide recommends the method, form, and content for adding specific device models and associated Sensor Actuator Network Communications Standard (SANCS) extensions to the SEMI Sensor Actuator Network (SAN) standard. This guide facilitates consistency of these ballots, leading to better consistency and clarity of the resulting specifications and to easier development of SAN standard-conformant devices that achieve a high level of interoperability.

2 Scope

2.1 This guide provides written templates for ballots that will add either a specific device model (SDM), an SANCS ancillary standard, or SANCS extensions to support an SDM to the SEMI SAN standards. The guide provides directions for using the templates.

2.2 Physical devices will not be compliant with this guide; they will be compliant to standards which are spawned from the templates specified herein.

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

3.1 *SEMI Standards*

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

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

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

SEMI E54 — Sensor/Actuator Network Standard

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Standard for Sensor/Actuator Network Specific Device Model for Mass Flow Device

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

4 Terminology

4.1 *Abbreviations and Acronyms*

4.1.1 *A* — Actuator (a CDM class definition)

4.1.2 *AE* — Active Element (a CDM class definition)

4.1.3 *C* — Controller (a CDM class definition)

4.1.4 *CDM* — (SEMI) Common Device Model

4.1.5 *DM* — Device Manager (a CDM class definition)

4.1.6 *ISO-OSI* — International Organization for Standardization - Open Systems Interconnect (model)

4.1.7 *NCS* — Network Communications Standard

4.1.8 *OMT* — Object Modeling Technique

4.1.9 *S* — Sensor (A CDM object)

4.1.10 *SAC* — Sensor/Actuator/Controller (a CDM class definition)

4.1.11 *SAN* — Sensor/Actuator Network (or Sensor Bus)

4.1.12 *SANCS* — Sensor/Actuator Network Communications Standard, frequently abbreviated to NCS

4.1.13 *SDM* — (SEMI) Specific Device Model

4.2 *Definitions*

4.2.1 *connection oriented* — a situation between communicating objects wherein they are connected in a mode analogous to a two-way phone conversation.

4.2.2 *specific device model (SDM)* — a model used to specify each type of device, such as a Mass Flow Controller or Thermocouple.

5 Background

5.1 The SEMI sensor bus (SAN) communications model consists of three components: a CDM, an SDM, and an SANCS (NCS). The following subsections provide a brief description of the specifications which define these components:

5.1.1 *Network Communications Standard Specification* — An NCS specification is required for each network technology. Each one consists of two major parts:

5.1.1.1 *Part 1, Enabling Protocol and CDM Object Presentation* — This part of an SANCS specification should achieve the following goals:

- provide an ISO-OSI layered specification of the protocol, specifying the protocol requirements at each layer,
- specify the object oriented communication environment, including object representation and addressing, attributes and services addressing, etc.,
- specify how CDM objects are represented and addressed to/from the network, and
- specify how AE class and subclass objects, defined in the CDM and the SDMs, are represented and addressed to/from the network.

5.1.1.2 *Part 2, Enabling Specific Device Models* — This part provides a mapping of SDM objects for the SANCS specification. There should be a section for each SDM which identifies specific objects that must be supported to enable the SDM and how the SDM structure fits into the protocol application layer foundation.

5.1.2 *Common Device Model* — Each device's functions on a network should comply with the specifications of the Common Device Model, SEMI E54.1. Three types or classes of objects may be combined to create models of real devices. The types are Active Element (AE) and its subclasses, Sensor/Actuator Controller (SAC) and its embedded objects, and the Device Manager (DM).

NOTE 1: The CDM specifies the relationship between these objects. It specifies some of the structure and behavior for the DM and SAC objects. The subclasses of the AE object provide capabilities which are useful to most devices.

5.2 *Specific Device Model*

5.2.1 Specific Device Models are defined in the SEMI SAN SDM (SEMI E54.3). Each SDM is added as a separate new section (similar to the way a new section is added to SEMI E5 when a new stream is added).

5.2.2 The Specific Device Models (SDM's) should be a specialization of the CDM and may extend the attributes, services, and behavior of any of the Sensor, Actuator, or Controller (subclasses of the AE class) objects of which they are made. SDM's may add to the attributes, services, and behavior of the SAC object.

6 Ballot Preparation

6.1 *The Cover Letter* — All ballots are submitted with a cover letter. The cover letter sets the context for the ballot. The Ballot Cover Letter Template provides guidance to prepare the cover letter.

6.2 *Specific Device Model Ballots* — A ballot to add an ancillary specification for a new SDM should be prepared as defined in the template in the section titled Template for SDM Ballot Proposals.

6.2.1 SEMI E39 is the basis for the tables and figures called out in the template. Refer to SEMI E39 for clarifications on using and defining object models. General instructions for using the template are contained in the subsections of this section. Specific instructions are included within the templates.

6.2.2 Device application models should be presented using Rumbaugh's Object Modeling Technology notation for information models. (See SEMI E39 for details on using object notation for defining standards.)

6.2.3 The behavior of objects should be defined using Harel State charts, state definition tables, and state transition tables. A discussion of the notation may be found in Appendix A.5 of SEMI E30. An alternative table, Object Behavior State Transition Matrix, may be used in place of the more traditional (State) Transition Table. Cells in the Transition Matrix indicate the transition action which is taken for an event for any of the states in which it can be detected. This can be particularly useful for describing the variation in device response depending on a sub-state when the parent state detects an event.

6.2.4 The set of tables which define the device's interfaces are the only testable parts of the specification for the application models. The interface for each device model should be defined in a series of tables, as follows:

6.2.4.1 *Attribute Tables* — For each object defined as an element of a device, a table which defines its network visible attributes should be specified. Each attribute is named and tagged. Its network access type (read only or read/write) and storage class (volatile, non-volatile, or constant) are also specified. The table should also indicate which attributes have standardized values and which are device supplier reserved. The definition or description of an attribute can either be included in the table or else should immediately follow the table. The "Required" column indicates if the attribute must be supported for a device to be SDM-compliant. Use the letter "Y" to indicate yes and the letter "N" to indicate no. The form field is used to indicate the data type of the attribute, such as integer,

floating point, array of characters, etc. Actual format of these data types is network-dependent.

6.2.4.2 *Optional Service Parameter Table* — If this table is included in the specification of an SDM, the parameters, which are used by the device's services, are defined in it, including the data type and length.

6.2.4.3 *Service Definition Table* — Each service and its parameters are defined in a table. If the optional parameter table is not used in the SDM specification, then the form field should be included in the service definition table.

6.3 *SANCS Ballots* — A ballot to add an ancillary specification for a new SANCS should be prepared as defined by the template in the section titled Template for Part 1 of an SANCS Ancillary Standard.

6.3.1 When a new SDM is added to the SAN standard, an update to Part 2 of the ancillary SANCS specifications is needed. A ballot should be prepared as defined in the template in the section titled Template for an SANCS ballot to support an SDM.

6.3.2 The nature of the Template for an SANCS ballot to support an SDM is that it adds SDM information to only one SANCS specification at a time. Additions and changes to support each new SDM are balloted separately for each SANCS specification.

6.3.3 *Coordination with Industry User Groups* — Most of the network technologies are supported by an industry user group or association. These groups are responsible for the assignment (i.e., mapping) of identifiers to the variables or attributes, parameters, and messages/services that will be managed by their protocol. When the SEMI SAN standards require new identifiers, they should be requested from the appropriate industry group.

6.3.3.1 *Technical Notes* — SEMI will make technical notes available, for each protocol supported by the SEMI SAN standards, that list the protocol's identifiers.

6.4 *Template Conventions*

6.4.1 Instructions or text that is meant to be replaced with ballot-specific information are included within the templates in italics. Non-italicized text and tables are intended to be included in the ballot as depicted in the template.

6.4.2 Section, figure, and table numbers specified using letters are meant to indicate structure and format. Specific numbers should be generated by the ballot author. Ballots which require adding to or modifying existing SAN standards will be edited by SEMI for consistency with the existing standard where necessary.

7 Ballot Cover Letter Template

7.1 Each ballot is accompanied by a cover letter which includes the following sections:

7.1.1 *Proposal* — Provide a brief description, typically one or two sentences, of the content of the ballot proposal.

7.1.2 *Background* — In 1–3 paragraphs, give the readers the background they need to understand the content and value of the proposed standard.

7.1.3 *Impact* — Use one or two paragraphs to describe how this standard affects the SEMI community.

7.1.4 *Ballot Description* — Describe the form of the ballot proposal. Use one or two paragraphs.

7.1.4.1 Note that SDM ballots are additions to the SEMI SAN SDM Specification (SEMI E54.3). This is a parent document which already has sections for overall purpose, scope, terminology, conventions, etc. for SDMs. Each individual SDM ballot should include only device-specific purpose, scope, and terminology. Items that are used in multiple SDMs should be balloted as updates to the SEMI E54.3 parent document.

8 Template for SDM Ballot Proposals

SEMI Document Number

Title

1 Purpose

Describe the purpose of the document. This is typically to provide a network-independent application model for the specific device - NamedDevice.

2 Scope

Describe the scope of the document.

3 Referenced Documents

List documents referenced from within this specification.

4 Terminology

Define terms used within this document. This is only for terms that are used within this document and not defined elsewhere (in the SEMI E54.3 parent document or other referenced documents).

5 NamedDevice High Level Structure

5.1 *General Description* — Describe what this device does, how it is used, etc.

5.1.1 *NamedDevice Description* — Provide a description of NamedDevice, including information model(s) in figure(s) that use a Rumbaugh OMT (see SEMI E39 Appendix A1-2) diagram.

NOTE 2: The device may actually consist of multiple devices (e.g., Mass Flow Device includes a Controller device and a Meter device), in which case multiple sections should be created - one per device.

5.1.x *General Requirements*

5.1.x.1 *Device Objects* — All objects are defined in terms of their object name and instance identifier. Identifiers for all objects described in this document are summarized in Table m.

Table m NamedDevice Device Objects

Referenced Document Section	*Object Name*	*Object Identifier*	*Minimum Instances*	*Maximum Instances*

All parameters used by the device's services can be summarized using the table (Table n). This is optional, as the parameters are further defined in association with specific objects (see 5.x.2).

Table n Parameter Definition Table

Parameter Name	*Data Type*	*Tag*	*Description*

For each Object in Table m, a section (5.x) is created as below, beginning with 5.2.

5.x *Name1 Object*

5.x.1 *Name1 Object Attributes* — Provide the object attribute information using the table (Table o) below.

Table o Object Name1 Attributes

Attribute Name	*Definition**	*Attribute Identifier*	*Network Access*	*Form*	*Storage Class*	*Required*

* The definition column can be omitted if the definitions immediately follow the table. The definitions can be provided as subsections.

Where appropriate, provide initial and default values for object attributes. This can be done as a table.

5.x.2 *Name1 Object Services* — Describe the services provided by this object using the following table (Table p) to list the services:

Table p Object Name1 Services

Service	*Service Identifier*	*Type*	*Description*

Provide a detailed description of each service, including a definition of the parameters for the service. The parameters can be defined using the following table (Table q) format:

Table q Service1 Service Parameter Definitions

Parameter	*Request/Indication*	*Response/Confirmation*	*Data Type**	*Description*

* This column can be removed if there is an overall parameter definition table (Table n) included in the specification.

5.x.3 *Name1 Object Behavior* — Describe the object states and sub-states as needed with a Harel State Diagram(s):

Figure

Harel State Diagram

Provide a state definition table as given in Table r. More than one state table could be added if a device is sufficiently complex. Put in any text needed to explain object states.

Table r Name1 Object Behavior State Descriptions

State Name	*Description*

Describe the state transitions using a table as given in Tables s or t. More than one table could be needed.

Table s Name1 Object Behavior State Transition Table

Event or Transition #	*Current State*	*Trigger*	*New State*	*Action(s)*	*Comments*

Table t Name1 Object Behavior State Transition Matrix

Event Number	*State 1*	*State 2*	*State 3*	
			Sub A	*Sub B*
1				
2				
3				
4				

9 Template for Part 1 of an SANCS Ancillary Standard

9.1 The body of the ballot should then contain the following sections:

1 Purpose

This section contains wording describing the purpose with customization for the specific protocol. A “Background and Motivation” subsection could be included to generally describe the protocol.

2 Scope

This section contains wording describing the scope with customization for the specific protocol.

3 Limitations

This section contains wording describing the limitations customized for the specific protocol. Any limitations of CDM or SDMs support, due to the protocol, should be identified.

4 Referenced Documents

This section contains wording describing the referenced documents with additional references to protocol specification documentation and any support documentation as required for complete specification of the protocol (two subsections). A clearly defined mechanism is specified for obtaining any documentation that is not publicly available or available from SEMI.

5 Terminology

The various protocols and the CDM invariably use different terminology. A mapping is made between the CDM terminology and the terminology used in discussing the protocol. For example, a CDM Service implementation may be defined as an Action in the NCS. This terminology mapping is explicitly covered completely in the "Terminology" section. A mapping table is desirable that contains a column listing of the CDM terms in the same order as defined in the CDM document, mapped to terminology used in the NCS document. This NCS terminology and any additional terminology is defined in a subsection (e.g., Section 5.2).

6 Communication Protocol High Level Structure

A basic description of the protocol is provided in Section 6. The main purpose of Section 6 is to give the reader a general understanding of what will be specifically presented in the remainder of the subsections in Section 6.

The rest of the subsections in Section 6 should be devoted to describing the communications standard in terms of the OSI seven-layer model (one subsection for each layer) and any network management services.

Each of the OSI seven layers should be addressed and specified either directly or through reference; for each:

- If an OSI layer is not supported/defined, this is stated explicitly.
- Choices among protocol options are made, specified, and explained where appropriate. A complete summary of protocol choices specified and protocol options allowed should be summarized in a subsection of this section of the specification.

The Physical layer is specified either directly or through reference (signaling, bus arbitration, baud rate, transceivers, cabling, connectors, etc.).

The Data link layer is specified either directly or through reference (packet/frame specification and node addressing).

The Network layer is optional.

The Transport layer elements are specified either directly or through reference; message segmentation and reassembling should be supported as no message length limits are indicated in CDM or SDM's. This support should be provided as a transport layer service. Note that this functionality may alternatively be provided at the application layer by protocols unable to support it at the transport layer; if this is the case, it is explicitly stated in the description of transport layer support. Connection support would be specified here as appropriate.

The Session layer is optional.

The Presentation layer is optional.

The Application layer is specified. The document defines (explicitly or through reference) how objects are structured, identified, and addressed. Thus, it should also define how object attributes are addressed and how object request and notify services are addressed and communicated. This section also defines the application object to object communication mechanism which should include or reference a communication state model. This communication model should clearly indicate whether or not the protocol is connection oriented.

Network Management Services may be defined in a separate subsection.

7 Required Object Types

This section identifies specific objects that must be supported to enable the CDM. The document should identify how the CDM structure (Objects and relations) fits into the protocol application layer foundation. Any identifiable protocol object hierarchy should be introduced here. Specifically, the following are defined:

- The implementation of all CDM objects (DM, SAC, AE, and all subclasses) defined to the extent that these objects are defined in CDM documentation.
- Specific addressing/referencing for CDM objects (attributes, services, etc.) defined as necessary so as to specify access to CDM objects as required by the CDM standard specification. Attributes and services are further specified as necessary so that the protocol required to access specific attributes and services is fully defined. For example, a CDM service may be defined as a "new" SANCS service or mapped to an existing SANCS service (if an appropriate candidate has already been defined in the protocol specification).
- Any limitations among CDM object options are identified.
- Any additional capabilities required of, or optional for, CDM objects (e.g., attributes, services, and/or behavior) are identified.
- Any additional required objects (e.g., network management object) identified and their interaction with the CDM objects (e.g., connection management object) should be specified.

8 Protocol Compliance

This section should specify a method or reference for testing protocol compliance. It should contain in subsection 8.1 a "protocol specification sheet" that lists all protocol options that have been specified in the document or must be resolved by the user in order to achieve interoperability.

10 Template for an SANCS Ballot to Support an SDM

Section 1 Table of Contents

Request that the table of contents for the SANCS be updated to contain a pointer to the paragraphs which specify the SDM mappings which are being added by the ballot.

Section 2 <SDM> Object Mapping

Each of the subsections within this section should identify specific objects that must be supported to enable the SDM and should identify how the SDM structure (objects and relations) fits into the protocol application layer foundation.

Specifically, the following issues are addressed:

Terminology

- A mapping is made between the SDM terminology and the terminology used in discussing the protocol. This is explicitly covered in a "Terminology" section. A mapping table is desirable that contains a column listing of the SDM terms in the same order as defined in the CDM document, mapped to terminology used by the network technology.

Required Object Types

- Address implementation of all SDM objects to the extent that they are defined in SDM documentation.
- Identify and map the specific addressing/referencing for SDM object attributes, services, and parameters to the protocol. Note that, for example, this may be achieved by fully specifying a SDM service as a "new" NCS service or by mapping the service to an existing NCS service.

Protocol Compliance

- Provide methods or references (in addition to those identified in the section titled Template for Part 1 of an SANCS Ancillary Standard, Section 8) for testing protocol compliance of the SDM.

SEMI E54.3-0698 (Reapproved 0704)
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK SPECIFIC DEVICE MODEL FOR MASS FLOW DEVICE

This specification was technically reapproved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 14, 2004. Initially available at www.semi.org May 2004; to be published July 2004. Originally published June 1998.

1 Purpose

1.1 This specification is part of a suite of standards which specify the implementation of SEMI standards for the Sensor/Actuator Network. The specific purpose of this specification is to describe a network-independent application model comprised of device objects which are common to all Mass Flow Devices on a semiconductor equipment Sensor/Actuator communications network.

2 Scope

2.1 This specification specifically addresses the minimum attributes, services, and behavior a Mass Flow Controller (MFC) and Mass Flow Meter (MFM) device must support to be interoperable on the Sensor/Actuator Network.

2.2 This specification is intended to ensure a high-degree of device interoperability on the Sensor/Actuator Network, while still allowing flexibility for product differentiation and technology evolution.

2.3 The model specified in this specification is used in conjunction with the Sensor/Actuator Network Common Device Model (CDM) to completely describe the MFC or MFM as it appears from the network interface.

2.4 This specification, together with the Sensor/Actuator Network Standard, the Sensor/Actuator Network Common Device Model, and a Sensor/Actuator Network Communication Specification, form a complete interoperability specification for the MFC and MFM.

2.5 To comply with this specification, a device must implement and support, at a minimum, the required attributes, services, and behavior identified in these documents. Support for **optional** attributes, services, and behavior are not required to be compliant to this specification. Optional attributes, services, and behavior are specified in these documents to promote further device interoperability as features evolve and are adopted by more manufacturers. If optional attributes, services, and behavior are implemented for this device, they must be implemented as identified in this document.

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 Limitations

3.1 This specification is a companion to a suite of specifications which together make up the Sensor/Actuator Network Communication standard. Therefore, using portions of this specification that relate to network communications necessarily requires an understanding of the associated network specification.

3.2 As this document is a specification for the Mass Flow Device Model, it does not contain any definition of objects, attributes, services, or behavioral descriptions that are already defined in the Sensor/Actuator Network Common Device Model (CDM). Additional attributes, attribute assignments, services, and/or service parameters that are Mass Flow Device-specific and/or implementation-specific are contained in this specification.

3.3 While this specification is sufficient to completely describe the MFC or MFM as it appears from the network, it does not fully describe behavior of the MFC or MFM which is not visible from the network. This allows flexibility in implementation techniques and product differentiation between manufacturers. Manufacturer-specific objects may be defined by the manufacturer, but are, by definition, outside the scope of this standard.

3.4 This specification is compatible, but not compliant, with SEMI E39. This means that although this specification 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 specification are compatible with those of SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and, thus, are not applied to the Mass Flow Device Model.

3.5 Operation over the entire range specified for an attribute within a specific object instance is not a requisite for compliance with this specification.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E12 — Standard for Standard Pressure, Temperature, Density, and Flow Units Used in Mass Flow Meters and Mass Flow Controllers

SEMI E18 — Guideline for Temperature Specifications of the Mass Flow Controller

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

SEMI E52 — Practice for Referencing Gases and Gas Mixtures Used in Digital Mass Flow Controllers

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

4.2 *IEEE Document*[1]

IEEE 754 — Floating Point Definition

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

5 Terminology

5.1 *Terminology Defined in Standard for Sensor/Actuator Network Common Device Model (SEMI E54.1)*

5.1.1 Attribute

5.1.2 Behavior

5.1.3 Byte

5.1.4 Character

5.1.5 Device

5.1.6 Device Manager (DM) Object

5.1.7 Device Model

5.1.8 Instance

5.1.9 Nibble

5.1.10 Object

5.1.11 S, A, and C Objects

5.1.12 Sensor Actuator Controller (SAC) Object

5.1.13 Service

5.1.14 State Diagram

5.2 *Definitions*

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

5.2.2 *common device model (CDM)* — refers to Sensor/Actuator Network Common Device Model (SEMI E54.1).

5.2.3 *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

5.2.4 *data units* — an unsigned integer formatted as an enumerated byte 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 an appendix of this document.

5.2.5 *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
2	Month	Unsigned Short Integer(range of 1 to 12)
3	Day	Unsigned Short

1 Institute of Electrical and Electronics Engineers, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, New Jersey 08855-1331, USA. Telephone: 732.981.0060; Fax: 732.981.1721, Website: www.ieee.org

		Integer(range of 1 to 31)

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

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

5.2.8 *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.

5.2.9 *gas calibration* — a reference to a set of parameters or methods which are used to calibrate or correct the device for a particular gas type, range, and units.

5.2.10 *gas standard number* — a number that references a gas type. The number and its referenced gas type are defined in SEMI E52.

5.2.11 *gas standard symbol* — a text symbol that references a gas type. The symbol and its referenced gas type are defined in SEMI E52.

5.2.12 *last valid value (LVV)* — the most recent value successfully assigned to an attribute.

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

5.2.14 *long real (LREAL)* — a double floating point number, eight bytes long, as defined in IEEE 754.

5.2.15 *manufacturer* — in the context of this document, this refers to the manufacturer of the device.

5.2.16 *mass flow controller (MFC)* — a self-contained device, consisting of a mass flow transducer, control valve, and control and signal-processing electronics, commonly used in the semiconductor industry to measure and regulate the mass flow of gas.

5.2.17 *mass flow device (MFD)* — a device which is either a mass flow controller or mass flow meter.

5.2.18 *mass flow meter (MFM)* — a self-contained device, consisting of a mass flow transducer and signal-processing electronics, commonly used in the semiconductor industry to measure the mass flow of gas.

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

5.2.20 *programmed gas calibration* — a reference to a particular gas type, range, and units for which the device is currently calibrated.

5.2.21 *real (REAL)* — a floating point number, four bytes long, as defined by IEEE 754.

5.2.22 *short integer (SINT)* — an integer, one byte long, in the range -128 to 127.

5.2.23 *signed integer (INT)* — an integer, two bytes long, in the range -32768 to 32767.

5.2.24 *text string* — a string of one-byte characters. See Section 5.1 for a definition of a character.

5.2.25 *unsigned integer (UINT)* — an integer, two bytes long, in the range 0 to 65535.

5.2.26 *unsigned short integer (USINT)* — an integer, one byte long, in the range 0 to 255.

6 Requirements

6.1 In order to implement this standard in a Mass Flow Device, it is necessary to also implement SEMI E54.1 and one of the Sensor/Actuator Network Communication standards. See Section 2 for more information on a complete interoperability standard.

7 Conventions

7.1 This document embraces the Harel State Chart notation, the transition table definition format, the object attribute representation formats, service message definition formats, and behavior definition formats as specified in SEMI E54.1.

7.2 Figure 1 describes the convention for object representation used throughout this specification.

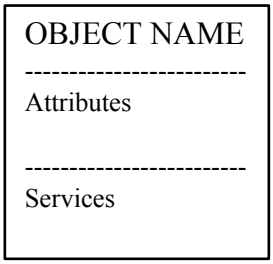


Figure 1
Object Representation

8 Device High Level Structure

8.1 *General Description* — The high level object view of a Mass Flow Meter (MFM) device and a Mass Flow Controller (MFC) Device is shown in Figure 2.

8.1.1 Note that the "MFM" and the "MFC" objects are depicted in Figure 2 only for the purposes of illustrating a high level view of the device and its component objects. In the context of this document, these objects are not addressable, do not have addressable attributes, do not have accessible services, nor do they exhibit any

defined behavior. These objects are only included in the figures to aid in the visualization of the device.

8.1.2 This document defines in detail all of the component objects unique to the MFC and the MFM devices. References, rather than definitions, are included for the DM, the SAC, and other objects defined in SEMI E54.1.

8.1.3 Many of the objects defined in this document inherit properties from other objects. The properties inherited include attribute, service, and behavior definitions. These other objects are specified here or in SEMI E54.1.

8.1.4 This document provides for future extensions, as well as manufacturer-specific enhancements, by reserving object attribute identifiers and object service identifiers. Specifically, all object definitions in this document specify or reserve the first 64 attribute identifiers (A1 through A64) and the first 64 service identifiers (S1 through S64), allowing manufacturers to specify identifiers beyond these ranges. Additionally, byte-enumerated attributes are specified or reserved from 0 to 63, allowing manufacturers to specify enumerations beyond this range (64 to 255).

8.1.5 *Mass Flow Meter (MFM) Device Description* — A Mass Flow Meter device profile is composed of the component objects and object relationships shown in Figure 2.

8.1.6 *Mass Flow Controller (MFC) Device Description* — A Mass Flow Controller device profile is composed of (1) all of the component objects and object relationships of the Mass Flow Meter Device in addition to (2) the component objects and object relationships as shown in Figure 2.

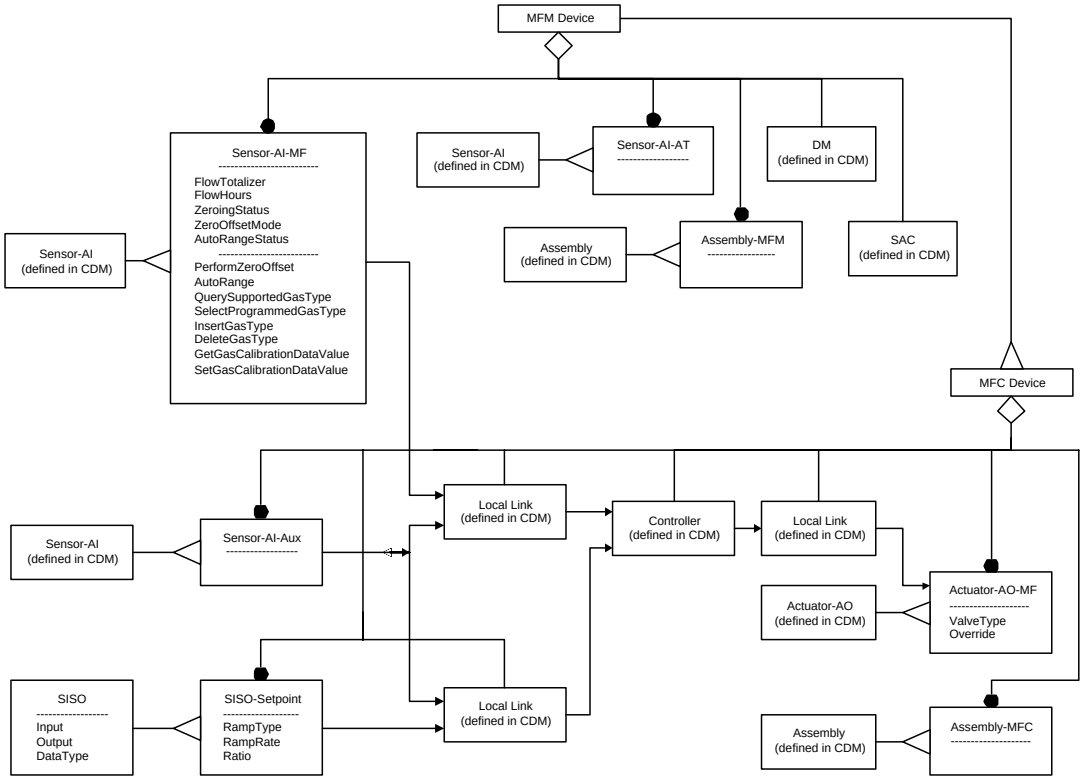


Figure 2
Mass Flow Meter Device and Mass Flow Controller Device High Level Structure

8.1.7 *General Requirements*

8.1.7.1 *Device Objects* — All 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.

Table 2 Mass Flow Device Objects

Referenced Document Section	*Object Name*	*Object Identifier*	*MFC Minimum Instances*	*MFM Minimum Instances*	*Maximum Instances*
8.2	Device Manager (DM)	MFD1	1	1	1
8.3	Sensor Actuator Controller (SAC)	MFD2	1	1	1
8.4	Sensor-AI-MF	MFD3	1	1	Manufacturer-Specified
8.5	Sensor-AI-AT	MFD4	0	0	Manufacturer-Specified
8.6	Assembly-MFM	MFD5	0	0	1
8.7	Sensor-AI-Aux	MFD6	0	0	Manufacturer-Specified
8.8	Actuator-AO-MF	MFD7	1	0	Manufacturer-Specified
8.9	Controller	MFD8	1	0	1
8.10	Local Link	MFD9	2	0	Manufacturer-Specified
8.11	SISO	MFD10	0	0	Manufacturer-Specified
8.12	SISO-Setpoint	MFD11	0	0	Manufacturer-Specified
8.13	Assembly-MFC	MFD12	0	0	Manufacturer-Specified
—	Reserved	MFD13–MFD64	–	–	—
—	Manufacturer-Specified	> MFD64	–	–	—

8.1.7.2 *Object Services* — Not all object services listed in this document can necessarily be requested over the network. They are included in this document because their behavior may generate network activity.

8.1.7.3 *Object Behavior* — For all service requests received over the network that are unsupported by the object, or contain a parameter value which is beyond the supported range, or which is otherwise invalid, a network-specific service error response is generated as specified in SEMI E54.1.

8.2 *Device Manager Object (DM)*

8.2.1 The Device Manager object instance is the device component responsible for managing and consolidating the device operation as specified in SEMI E54.1. The following sections specify the components of the DM object that are not specified in the Common Device Model.

8.2.2 *Device Manager Object Attributes* — Required and optional DM object instance attributes are listed in Table 3.

Table 3 DM Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	A1	R	Yes	Refer to CDM (SEMI E54.1).
Exception Detail Alarm	A13	R	No	Refer to CDM (SEMI E54.1).
Exception Detail Warning	A14	R	No	Refer to CDM (SEMI E54.1).
Reserved	A33–A64	—	—	Reserved for SDM future expansion.
Manufacturer-Specified	> A64	—	—	Manufacturer-Specific attributes

8.2.2.1 *Device Type* — An attribute which uniquely identifies the type of the device on the network. The device type attribute is assigned as follows:

Mass Flow Controller Device = "MFC"

Mass Flow Meter Device = "MFM"

8.2.2.2 *Exception Detail Alarm (Optional)* — An attribute which identifies the detailed alarm status of the device. Table 4 defines the bit assignments associated with the alarm exception detail.

Table 4 Exception Detail Alarm Bit Assignments

Bit	*Device-Specific Alarm [0]*
0	Reserved
1	Indicated Flow High
2	Indicated Flow Low
3	Flow Controller
4	Flow Valve Actuator
5	Reserved
6	Reserved
7	Reserved

8.2.2.3 *Exception Detail Warning (Optional)* — An attribute which identifies the detailed warning status of the device. Table 5 defines the bit assignments associated with the warning exception detail.

Table 5 Exception Detail Warning Bit Assignments

Bit	*Device-Specific Warning [0]*
0	Reserved
1	Indicated Flow High
2	Indicated Flow Low
3	Flow Controller
4	Flow Valve Actuator
5	Reserved
6	Reserved
7	Reserved

8.2.2.4 *Initial and Default Values*

Table 6 DM Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Device Type	MFC, MFM	MFC, MFM	MFM = Mass Flow Meter Device MFC = Mass Flow Controller Device
Exception Detail Alarm	0	0	
Exception Detail Warning	0	0	

8.3 *Sensor Actuator Controller Object (SAC)* — The Sensor Actuator Controller object instance is the device component responsible for coordinating the interaction of the mass flow device with the sensory/actuation/control environment as specified in SEMI E54.1.

8.4 *Sensor-AI-MF Object* — The Sensor-AI-MF object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-MF object instance is the device component responsible for retrieving a reading from a physical flow sensor, optionally correcting the reading with a manufacturer-specified algorithm, or algorithms, then making the value available to feed the Controller object instance via a Local Link object instance.

8.4.1 *Sensor-AI-MF Object Attributes*

Table 7 Sensor-AI-MF Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Flow Totalizer	A1	RW	No	REAL
Flow Hours	A2	R	No	ULINT
Zero Offset Mode	A5	RW	No	Enumerated Byte
Zeroing Status	A6	R	No	Enumerated Byte
Autorange Status	A7	R	No	Enumerated Byte
Reserved	A8–A64	—	—	Reserved for future expansion
Manufacturer-Specified	> A64	—	—	Manufacturer-Specific attributes

8.4.1.1 *Flow Totalizer (Optional)* — An attribute that maintains the volume of gas in standard cubic centimeters (SCC) that has flowed through the device since the last time the flow *totalizer* attribute value was set to zero.

8.4.1.2 *Flow Hours (Optional)* — An attribute which identifies the number of hours that the device has been flowing gas since the last time the *flow hours* attribute value was set to zero, as specified by the manufacturer. The attribute is an unsigned long integer with a resolution of 1 hour.

8.4.1.3 *Zero Offset Mode (Optional)* — An attribute which specifies the zero offset formula to be applied to produce the value attribute. This attribute is an enumerated byte that can take on one of the following values:

0 = Disable
1 = Enable Application of Formula
2 = Enable Automatic Zeroing
3–63 = Reserved
64–255 = Manufacturer-Specified

8.4.1.4 *Zeroing Status (Optional)* — An attribute which specifies whether the object is in the ZEROING substate. This attribute is an enumerated byte that can take on one of the following values:

0 = Not Zeroing
1 = Zeroing

8.4.1.5 *Autorange Status (Optional)* — An attribute which specifies whether the object is in the AUTORANGING sub-state. This attribute is an enumerated byte that can take on one of the following values:

0 = Not Autoranging
1 = Autoranging

8.4.1.6 *Initial and Default Values*

Table 8 Sensor-AI-MF Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Flow Totalizer	LVV	0	
Flow Hours	LVV	0	
Zero Offset Mode	LVV	Disable	
Zeroing Status	Not Zeroing	Not Zeroing	
Autorange Status	LVV	Not Autoranging	

8.4.2 *Sensor-AI-MF Object Services* — The services provided by the Sensor-AI-MF object instance are defined in SEMI E54.1. The Sensor-AI-MF object supports the additional services listed below.

8.4.2.1 All Gas Correction services are optional. Gas correction may be handled within the device as a pre-assigned gas calibration or may not be implemented at all. If these services are available, it is manufacturer-specific as to

whether the device supports references to *gas standard number*, *gas standard symbol*, or both. It is strongly recommended that, if a device supports only one method of referencing gas types, it should be *gas standard number*.

8.4.2.2 A device may support any of these services, in whole or in part, as specified by the manufacturer. A manufacturer may specify a subset of the service parameter values which is supported. The following table lists the services sup-ported by the Sensor-AI-MF object instance:

Table 9 Sensor-AI-MF Object Services

Service	*Service Identifier*	*Type*	*Description*
Perform Zero Offset	S1	R	Used to instruct the object to perform an automatic zeroing operation.
Query-Supported Gas Types	S2	R	Used to query the device to determine whether a specific gas calibration is supported.
Selected Programmed Gas Type	S3	R	Used to select the gas type and associated data to be used as the current programmed gas calibration.
Insert Gas Type	S4	R	Used to add a gas type to the list of available gas types.
Delete Gas Type	S5	R	Used to remove a gas type from the list of available gas types.
Get Gas Calibration Data Value	S6	R	Used to request the value of a specific gas calibration data value.
Set Gas Calibration Data Value	S7	R	Used to set the value of a specific gas calibration data value.
Autorange	S8	R	Used to enter the AUTORANGING state.
Reserved	S9–S64	—	Reserved for future expansion.
Manufacturer-Specified	> S64	—	Manufacturer-Specific services

8.4.2.3 *Perform Zero Offset (Optional)* — This service is used to instruct the Sensor-AI-MF object instance to perform a one-time automatic zeroing operation on the device or to reset the *offset* attribute value to zero. This service causes the automatic zeroing of the Sensor-AI-MF object instance *value* attribute by modifying the value of the *offset* attribute in order to yield a value of zero for the *value* attribute. Table 10 describes the parameters specified for this service.

Table 10 Perform Zero Service Parameter Definitions

Parameter	*Request/Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Command	M	—	Byte	Enumerated Byte: 0 = Set offset attribute to zero 1 = Calculate offset 2 = Cancel Zeroing 3–63 = Reserved 64–255 = Manufacturer-Specified

8.4.2.4 *Query-Supported Gas Types Service (Optional)* — This service is used to query the device to determine whether a specific gas type, range, and units is supported. Supported, in this context, implies that the device contains suitable gas calibration correction data or methods to correct the flow measurement for the specified gas type, range, and units. The query can be made by referencing either *gas standard number* or *gas standard symbol* (see Section 5 for a definition of "gas standard number" and "gas standard symbol"). The entire list of supported gas standard numbers or gas standard symbols (with ranges and units) can also be requested. The following table describes the parameters specified for this service:

Table 11 Query-Supported Gas Types Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Query Type	M	—	Byte	0 = specific gas type 1 = all currently supported gas calibrations 2 = currently programmed gas calibration
Gas Standard Number	C*	—	Unsigned Integer	0 = use standard gas symbol field n = gas standard number
Gas Standard Symbol	C*	—	Text String	null character = not specified text string = gas standard symbol
Full Scale Range	C*	—	Real	0 = not specified n = full scale range
Units	C*	—	UINT	Indication of the units associated with the full scale range.
Valid Flag	—	M	Byte	0 = Not Valid 1 = Valid
Size of List	—	M	Unsigned Integer	Number of gas calibrations in the list.
List of Gas Calibrations	—	M	Array of Structures	The list of gas calibrations.

* Parameter is Mandatory for Query Type = 0.

8.4.2.4.1 *Query Type* — This parameter is used to specify the type of service response. It is an enumerated byte that can take on the following values:

0 = specific gas type
1 = all currently supported gas calibrations
2 = currently programmed gas calibration
3–63 = Reserved
64–255 = Manufacturer-Specified

8.4.2.4.2 *Gas Standard Number* — This parameter is used to specify a gas standard number, for which device support is being queried. A value of "0" indicates that the following parameter "gas standard symbol" is used instead to reference the gas type.

8.4.2.4.3 *Gas Standard Symbol* — This parameter is used to specify a gas standard symbol, for which device support is being queried. If the gas standard number parameter has a value that is not zero, then this parameter will have the value of null character.

8.4.2.4.4 *Full Scale Range* — This parameter is used to specify the full scale range, for which device support is being queried. See Section 5 for a definition of "full scale range." A value of "0" queries the device to return the entire list of full scale ranges for the specified gas type.

8.4.2.4.5 *Units* — This parameter is used to specify the units associated with the full scale range specified in the service request. Its values are defined in SEMI E54.1. The supported list includes:

SCCM
SLM
Percent
Volts
Millivolts
Counts

8.4.2.4.6 *Valid Flag* — The first parameter of a Query-Supported Gas Type service response is a byte that indicates whether the requested gas type, range, and units are supported. If an entire list was requested, and at least one gas calibration exists, this byte will have the value of "Valid."

8.4.2.4.7 *Size of List* — This parameter is an unsigned integer that specifies the number of gas calibrations in the list that follows.

8.4.2.4.8 *List of Gas Calibrations* — This attribute is an array of structures that identifies the gas calibrations supported in the device. The format of the structure is *gas standard number*, *gas standard symbol* (zero if not supported), *full scale range*, and *units*.

8.4.2.5 *Select Programmed Gas Type Service (Optional)* — This service is used to select the gas type, range, and units of the programmed gas calibration to be used by the device (see Section 5 for a definition of "programmed gas calibration"). The following table describes the parameters specified for this service:

Table 12 Select Programmed Gas Type Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Gas Standard Number	M	—	UINT	0 = use gas standard symbol field n = gas standard number
Gas Standard Symbol	M	—	Text String	null character = not specified text string = gas standard symbol
Range Select Mode	M	—	Byte	0 = select highest full scale range 1 = use full scale range and units specified 2 = use full scale range and units greater than or equal to specified
Full Scale Range	C*	—	REAL	Full scale range of which the select references.
Units	C*	—	UINT	Indication of the units associated with the full scale range.

* Parameter is Mandatory for Range Select Mode = 1 and 2.

8.4.2.5.1 *Gas Standard Number* — This parameter is used to specify a gas standard number, for which device support is being queried. A value of "0" indicates that the following parameter "gas standard symbol" is used instead to reference the gas type.

8.4.2.5.2 *Gas Standard Symbol* — This parameter is used to specify a gas standard symbol, for which device support is being queried. If the gas standard number parameter has a value that is not zero, then this parameter will have the value of null character.

8.4.2.5.3 *Range Select Mode* — This parameter is used to specify the mode by which the programmed gas is selected. It is an enumerated byte that can take on the following values:

0 = Select highest full scale range
1 = Use full scale range and units specified
2 = Use full scale range and units greater than or equal to specified
3–63 = Reserved
64–255 = Manufacturer-Specified

8.4.2.5.4 *Full Scale Range* — This parameter is used to specify the full scale range to be selected as the current programmed gas calibration full scale range.

8.4.2.5.5 *Units* — This parameter specifies the units associated with the full scale range to be set as the current programmed gas calibration being used by the device. If the Full Scale Range parameter has the value "0," this parameter is not included in the request. Its values are defined in Section 8.4.2.2.

8.4.2.6 *Insert Gas Type Service (Optional)* — This service is used to add a gas calibration to the list of supported gas calibrations. This service will typically be invoked while calibrating the device, since it has not specified what, if any, gas calibration data or methods will be set as the new gas calibration. The Set Gas Calibration Data Value service will generally be required to set the gas calibration data or methods after invoking this service. The following table describes the parameters specified for this service:

Table 13 Insert Gas Type Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Gas Standard Number	M	—	UINT	0 = use gas standard symbol field n = gas standard number
Gas Standard Symbol	M	—	Text String	null character = not specified text string = gas standard symbol
Full Scale Range	M	—	REAL	n = full scale range
Units	M	—	UINT	Indication of the units associated with the full scale range.

8.4.2.6.1 *Gas Standard Number* — This parameter is used to specify a gas standard number, for which a gas calibration is to be added. A value of "0" indicates that the following parameter "gas standard symbol" is used instead to reference the gas type.

8.4.2.6.2 *Gas Standard Symbol* — This parameter is used to specify a gas standard symbol for which a gas calibration is to be added.

8.4.2.6.3 *Full Scale Range* — This parameter is used to specify the full scale range for which a gas calibration is to be added. See Section 5 for a definition of full scale range.

8.4.2.6.4 *Units* — This parameter is used to specify the units associated with the full scale range specified in the service request. Its values are defined in Section 8.4.2.4.

8.4.2.7 *Delete Gas Type Service (Optional)* — This service is used to remove a gas calibration from the list of supported gas calibrations. The gas calibration data or method used by the device, if the current programmed gas type is deleted, is manufacturer-specified. The following table describes the parameters specified for this service:

Table 14 Delete Gas Type Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Gas Standard Number	M	—	UINT	0 = use gas standard symbol field n = gas standard number
Gas Standard Symbol	M	—	Text String	null character = not specified text string = gas standard symbol
Full Scale Range	M	—	REAL	n = full scale range
Units	M	—	UINT	Indication of the units associated with the full scale range.

8.4.2.7.1 *Gas Standard Number* — This parameter is used to specify a gas standard number for which a gas calibration is to be deleted. A value of "0" indicates that the following parameter "gas standard symbol" is used instead to reference the gas type.

8.4.2.7.2 *Gas Standard Symbol* — This parameter is used to specify a gas standard symbol for which a gas calibration is to be deleted.

8.4.2.7.3 *Full Scale Range* — This parameter is used to specify the full scale range for which a gas calibration is to be deleted. See Section 5 for a definition of "full scale range."

8.4.2.7.4 *Units* — This parameter is used to specify the units associated with the full scale range specified in the service request. Its values are defined in Section 8.4.2.4.

8.4.2.8 *Get Gas Calibration Data Value Service (Optional)* — This service is used to retrieve the values associ-ated with a gas calibration. The mechanism for referencing a data value involves specifying the gas type and full scale range (a default is provided to specify the programmed gas calibration currently in use by the device), followed by an index value which is used to specify the particular data value that is being requested. The following table describes the parameters specified for this service:

Table 15 Get Calibration Data Value Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Gas Standard Number	M	—	INT	-1 = current programmed gas calibration 0 = use gas standard symbol field n = gas standard number
Gas Standard Symbol	M	—	Text String	null character = not specified text string = gas standard symbol
Full Scale Range	M	—	REAL	0 = not specified n = full scale range
Units	M	—	UINT	Indication of the units associated with the full scale range.
Data Index	M	—	Byte	The identifier of the particular gas calibration datum within the list of data values.
Size of List	—	M	Byte	This parameter specifies the number of values returned in the list.
Zero	—	C	REAL	zero offset
Span	—	C	REAL	span multiplier
Calibration Date	—	C	Date	The date of calibration for a particular gas type and full scale range.
Calibration Gas Standard Number	—	C	UINT	The gas standard number representing the gas used to calibrate the device.
Calibration Temperature	—	C	REAL	The standard temperature of calibration conditions.
Calibration Pressure	—	C	REAL	The standard pressure of the calibration conditions.
Manufacturer-Specified Parameters	—	C	Manufacturer-Specific	

8.4.2.8.1 *Gas Standard Number* — This parameter is used to specify a gas standard number for which a gas calibration data value is being requested. Two special values are allowed: A value of "0" indicates that the following parameter "gas standard symbol" is used instead to reference the gas type. A value of "-1" indicates that the current programmed gas calibration is requested.

8.4.2.8.2 *Gas Standard Symbol* — This parameter is used to specify a gas standard symbol for which a gas calibration data value is being requested. If the *gas standard number* parameter has a value that is not "0," then this parameter is set to the null character.

8.4.2.8.3 *Full Scale Range* — This parameter is used to specify the full scale range for which a gas calibration data value is being requested. If the *gas standard number* parameter has a value that is "-1," then this parameter is set to zero.

8.4.2.8.4 *Units* — This parameter specifies the units associated with the *full scale range parameter*. Its values are defined in Section 8.4.2.4. If the *gas standard number* parameter has a value that is "-1," then this parameter is set to 0.

8.4.2.8.5 *Data Index* — This parameter is used to specify the index of the value being requested. It is an enumerated byte that can take on the values listed in the following table:

Table 16 Gas Calibration Data Value Matrix

Data Value Name	*Data Index*	*Access Network*	*Required*	*Form*
All	0	—	No	
Zero	1	RW	No	REAL
Span	2	RW	No	REAL
Calibration Date	3	R	No	Date
Calibration Gas Standard Number	4	R	No	UINT
Calibration Temperature	5	R	No	REAL
Calibration Pressure	6	R	No	REAL
Reserved	7–63	—	—	
Manufacturer-Specified	64–255	—	—	

Table 17 Gas Calibration Data Initial and Default Values

Data Value Name	*Data Index*	*Initial Value*	*Default Value*	*Comment*
Zero	1	LVV	0	
Span	2	LVV	1	
Calibration Date	3	LVV	0, 0, 0	
Calibration Gas Standard Number	4	LVV	Manufacturer-Specific	
Calibration Temperature	5	LVV	0.0	Defined by SEMI E12 as 0.0°C. Other industries may specify a different value.
Calibration Pressure	6	LVV	101.32	Defined by SEMI E12 as 101.32 kPa. Other industries may specify a different value.
Reserved	7–63	—	—	
Manufacturer-Specified	64–255	—	—	

8.4.2.8.5.1 *All* — This value for the data index parameter requests the list of all data values associated with a gas calibration. The returned parameter values are ordered by index as defined in Table 15. The size of the list returned is manufacturer-specified, since it includes the Manufacturer-Specific values.

8.4.2.8.5.2 *Size of List* — This parameter is used to specify the number of parameters returned in the response list.

8.4.2.8.5.3 *Zero* — This value is used in conjunction with the span value to correct the flow measurement. It is expressed in terms of the units parameter.

8.4.2.8.5.4 *Span* — This value is used in conjunction with the zero value to correct the flow measurement. It is a dimensionless value.

8.4.2.8.5.5 *Calibration Date* — This value identifies the date the device was last calibrated for the specified gas type and full scale range.

8.4.2.8.5.6 *Calibration Gas Standard Number* — This value identifies the gas type used when the device was calibrated for the referenced gas type and full scale range. It may be the same gas as gas type, a surrogate gas, nitrogen, or some other gas.

8.4.2.8.5.7 *Calibration Temperature* — This value identifies the Standard Temperature with respect to calibration conditions. The units for this value are degrees Centigrade.

8.4.2.8.5.8 *Calibration Pressure* — This value identifies the Standard Pressure with respect to calibration conditions. The units for this value are KiloPascal.

8.4.2.9 *Set Gas Calibration Data Value Service (Optional)* — This service is used to set the values associated with a gas calibration. The mechanism for referencing a data value involves specifying the gas type and full scale range (a default is provided to specify the programmed gas calibration currently in use by the device), followed by an index value which is used to specify the particular data value associated with the referenced gas calibration that is

being set. The specific value to be set is the last parameter passed in the service request. The following table describes the parameters specified for this service:

Table 18 Set Calibration Data Value Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Gas Standard Number	M	—	INT	-1 = current programmed gas calibration 0 = use gas standard symbol field n = gas standard number
Gas Standard Symbol	M	—	Text String	null character = not specified text string = gas standard symbol
Full Scale Range	M	—	REAL	0 = not specified n = full scale range
Units	M	—	UINT	Indication of the units associated with the full scale range.
Data Index	M	—	Byte	The identifier of the particular gas calibration datum within the list of data values.
Size of List	M	—	Byte	This parameter specifies the number of values in the list that follows.
Zero	C	—	REAL	zero offset
Span	C	—	REAL	span multiplier
Calibration Date	C	—	Date	The date of calibration for a particular gas type and full scale range.
Calibration Gas Standard Number	C	—	UINT	The gas standard number representing the gas used to calibrate the device.
Calibration Temperature	C	—	REAL	The standard temperature of calibration conditions.
Calibration Pressure	C	—	REAL	The standard pressure of the calibration conditions.
Manufacturer-Specified Parameters	C	—	Manufacturer-Specific	

For a description of these parameters, see the preceding section.

8.4.2.10 *Autorange Service (Optional)* — This service is used to instruct the Sensor-AI-MF object instance to enter the AUTORANGING state from the NOT AUTORANGING state or to enter the NOT AUTORANGING state from the AUTORANGING state. The following table describes the parameters specified for this service:

Table 19 Autorange Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Command	M	—	Byte	Enumerated Byte: 0 = enter NOT AUTORANGING state 1 = enter AUTORANGING state 2–63 = Reserved 64–255 = Manufacturer-Specified

8.4.3 *Sensor-AI-MF Object Behavior* — The behavior exhibited by the Sensor-AI-MF object instance is inherited from the Sensor-AI object defined in SEMI E54.1. Additional specific behavior associated with the Sensor-AI-MF object is defined below.

8.4.3.1 *Sensor-AI-MF OPERATING Application Process* — A reading is retrieved from a physical flow sensor. This reading may be corrected with a manufacturer-specified algorithm. This corrected reading becomes the input to the offset and gain formula to generate the *value* attribute as referenced in SEMI E54.1.

8.4.3.1.1 For Gas Correction, the value retrieved is corrected with a manufacturer-specified algorithm using the correction values, parameters, coefficients, or methods for the programmed gas calibration.

8.4.3.2 *Sensor-AI-MF OPERATING Application Process–Flow Totalizer* — The value of the *flow totalizer* attribute is incremented at a rate of once every cubic centimeter of gas flow. Whenever the *flow totalizer* attribute reaches its maximum allowed value, it no longer is incremented and remains at the maximum value.

8.4.3.3 *Sensor-AI-MF OPERATING Application Process–Flow Hours* — The value of the *flow hours* attribute is incremented at a rate of once every hour. Whenever the *flow hours* attribute reaches its maximum allowed value, it no longer is incremented and remains at the maximum value.

8.4.3.4 *Sensor-AI-MF ZEROING Application Process* — The Zeroing application process is described as follows: Certain manufacturer-specified service requests may be sent to other objects. The value of *offset* is set such that *value* is nulled to zero. The Sensor-AI-MF object instance determines, using a manufacturer-specified method, when a Zeroing application process is completed.

8.4.3.4.1 If a Perform Zero Offset: Command = 2 (cancel) service request is received while in the ZEROING state, the original value of the *offset* attribute is restored, and the process is considered Failed. If a Perform Zero Offset: Command = 0 (set offset attribute to zero) service request is received while in the ZEROING state, the *offset* attribute is set to zero, and the process is considered Failed. If a Perform Zero Offset: Command = 1 (calculate offset) service request is received while in the ZEROING state, the process is simply restarted.

8.4.3.4.2 Upon completion, the appropriate Pass or Fail service response is reported to the requesting object instance, and an internal Operate service request to this object is generated.

8.4.3.4.3 The Sensor-AI-MF object instance supports the additional behavior sub-states (defined in Figure 3) within the OPERATING state. Table 20 defines the additional sub-states, and Table 21 defines the additional state transitions specified for this object.

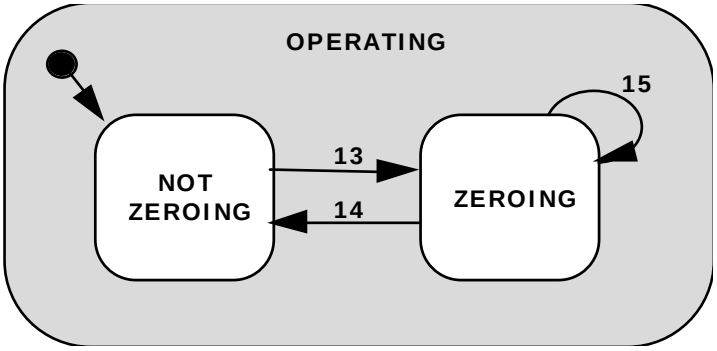


Figure 3
Sensor-AI-MF Object Behavior Additional Sub-States

Table 20 Sensor-AI-MF Object Behavior State Descriptions

State	*Description*
NOT ZEROING	The Sensor-AI-MF object instance is running the Sensor-AI-MF OPERATING application process.
ZEROING	The Sensor-AI-MF object instance is running the Sensor-AI-MF OPERATING application process. Additionally, the Sensor-AI-MF object instance is running the Sensor-AI-MF ZEROING application process.

Table 21 Sensor AI-MF Object Behavior State Transition Matrix

#	*Current State*	*Trigger*	*New State*	*Action*	*Comments*
4	NORMAL OPERATING	Get Attribute, Set Attribute, Restore Defaults request, Perform Zero Offset request (command ≠ 1)	NORMAL OPERATING	Get Attribute, Set Attribute, Restore Defaults appropriate response	Valid for all sub-states of NORMAL OPERATING.
13	NOT ZEROING	Perform Zero Offset request (command = 1)	ZEROING	Run the ZEROING application process. Set *zeroing status* to Zeroing.	
14	ZEROING	Operate request or Perform Zero Offset request (command ≠ 1)	NOT ZEROING	Resume the OPERATING application process. Set *zeroing status* to Not Zeroing. Report appropriate service response.	The mechanism for reporting is network-specific.
15	ZEROING	Perform Zero Offset request (command = 1)	ZEROING	Restart the ZEROING application process.	

8.4.3.5 *Sensor-AI-MF AUTORANGING Application Process* — The Sensor-AI-MF object instance supports the additional behavior sub-states (defined in Figure 4) within the OPERATING state. Table 22 defines the additional sub-states, and Table 23 defines the additional state transitions specified for this object.

8.4.3.5.1 The object instance is automatically selecting gas calibrations based on a manufacturer's specified method. This method may include a determination based on the value of the *value* attribute and/or on the value of the *setpoint value* attribute.

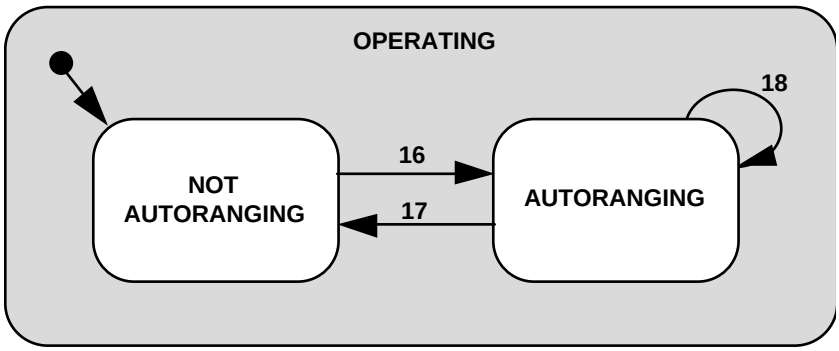


Figure 4
Sensor-AI-MF Object Behavior Additional Sub-States

Table 22 Sensor-AI-MF Object Behavior State Descriptions

State	*Description*
NOT AUTORANGING	The Sensor-AI-MF object instance is running the Sensor-AI-MF OPERATING application process.
AUTORANGING	The Sensor-AI-MF object instance is running the Sensor-AI-MF OPERATING application process. Additionally, the Sensor-AI-MF object instance is running the Sensor-AI-MF AUTORANGING application process (defined above).

Table 23 Sensor-AI-MF Object Behavior State Transition Matrix

#	*Current State*	*Trigger*	*New State*	*Action*	*Comments*
4	NORMAL OPERATING	Get Attribute, Set Attribute, Restore Defaults request, Autorange request (command = 0)	NORMAL OPERATING	Get Attribute, Set Attribute, Restore Defaults appropriate response	Valid for all sub-states of NORMAL OPERATING.
16	NOT AUTORANGING	Autorange request Command = 1	AUTORANGING	Run AUTORANGING application process. Set *Autoranging status* to Autoranging.	
17	AUTORANGING	Autorange request Command = 0	NOT AUTORANGING	Halt the Autoranging application process. Set *Autoranging status* to Not Autoranging.	
18	AUTORANGING	Autorange request Command = 1	AUTORANGING		No change.

8.5 *Sensor-AI-AT Object* — The Sensor-AI-AT object is an Ambient Temperature object instance which inherits attributes, services, and behavior from the Sensor-AI object as defined in SEMI E54.1. The Sensor-AI-AT object instance is the device component responsible for retrieving a reading from a physical temperature sensor and making the value available to the network.

8.5.1 *Sensor-AI-AT Object Attributes* — The inherited *value* attribute represents the ambient temperature measurement. There are no additional attributes defined for this object in this document.

8.5.2 *Sensor-AI-AT Object Services* — There are no additional services defined for this object in this document.

8.5.3 *Sensor-AI-AT Object Behavior* — The behavior exhibited by the Sensor-AI-AT object instance is defined in SEM E54.1. Additional specific behavior associated with the Sensor-AI-AT object is defined below.

8.5.3.1 *Sensor-AI-AT OPERATING Application Process* — A reading is retrieved from a physical temperature sensor. This reading may be corrected with a manufacturer-specified algorithm. This corrected reading becomes the input to the offset and gain formula to generate the *value* attribute as referenced in SEMI E54.1.

8.6 *Assembly-MFM Object* — The Assembly-MFM object inherits attributes, services, and behavior from the Assembly object. The Assembly object instance is the device component which provides a mechanism of grouping more than one attribute from one or more object instances into a single data structure for access over the network.

8.6.1 *Assembly-MFM Object Attributes*

Table 24 Assembly-MFM Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Data	A1	R	Yes	Structure as defined below.

8.6.1.1 *Data* — An attribute with the following list of attributes within its structure:

Table 25 Assembly-MFM Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	MFD3	A4	Sensor-AI-MF Value

8.7 *Sensor-AI-Aux Object* — The Sensor-AI-Aux object is an Auxiliary Input object instance which inherits attributes, services, and behavior from the Sensor-AI object as defined in SEMI E54.1. The Sensor-AI-Aux object instance is the device component responsible for retrieving a reading from a physical analog input and making the value available to the network.

8.7.1 *Sensor-AI-Aux Object Attributes* — The attributes provided by the Sensor-AI-Aux object instance are defined in SEMI E54.1. The *value* attribute represents the analog input measurement.

8.7.2 *Sensor-AI-Aux Object Services* — The services provided by the Sensor-AI object instance are defined in SEMI E54.1.

8.7.3 *Sensor-AI-Aux Object Behavior* — The behavior exhibited by the Sensor-AI-Aux object instance is defined in SEMI E54.1. Additional specific behavior associated with the Sensor-AI-Aux object is defined below.

8.7.3.1 *Sensor-AI-Aux OPERATING Application Process* — A reading is retrieved from a physical analog input. This reading may be corrected with a manufacturer-specified algorithm. This corrected reading becomes the input to the offset and gain formula to generate the *value* attribute as referenced in SEMI E54.1.

8.8 *Actuator-AO-MF Object* — The Actuator-AO-MF object instance inherits from the Actuator-AO object attributes, services, and behavior as defined in SEMI E54.1. The Actuator-AO-MF object instance is the component responsible for driving the physical mass flow control valve of the device. Override attributes are available to produce valve drives which either fully close or fully open the valve, irrespective of the value of the *setting* attribute.

8.8.1 *Actuator-AO-MF Object Attributes*

Table 26 Actuator-AO-MF Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Valve Type	A1	R	No	Enumerated Byte
Override	A2	RW	No	Enumerated Byte
Reserved	A3–A64	—	—	Reserved for future expansion.
Manufacturer-Specified	> A64	—	—	Manufacturer-Specific attributes.

8.8.1.1 *Valve Type (Optional)* — An attribute which specifies the type of valve present in the device. This attribute is an enumerated byte that can take on one of the following values:

0 = Solenoid

1 = Voice Coil

2 = Piezo Electric

3 = Thermal

4–63 = Reserved

64–255 = Manufacturer-Specified

8.8.1.2 *Override (Optional)* — An attribute which specifies an override to the controlled valve position derived from the Controller object instance. This attribute is an enumerated byte that can take on one of the following values:

0 = Normal — Normal control mode

1 = Flow Off — Valve Closed

2 = Purge — Valve Open

3 = Power Off — Valve to unpowered state

4–63 = Reserved

64–255 = Manufacturer-Specified

8.8.1.3 *Initial and Default Values*

Table 27 Actuator-AO-MF Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Valve Type	Default Value	Manufacturer-Specific	
Override	LVV	Normal	

8.8.2 *Actuator-AO-MF Object Services* — There are no additional services defined for this object in this document.

8.8.3 *Actuator-AO-MF Object Behavior* — The behavior exhibited by the Actuator-AO-MF object instance is defined in SEMI E54.1. The specific behavior associated with the Actuator-AO-MF object is defined below.

8.8.3.1 *Actuator-AO-MF OPERATING Application Process–Override* — The *override* attribute determines whether the position of the physical flow control valve will be overridden and, if so, to what position it will be driven. Otherwise, the physical flow control valve is driven by the converted signal derived from the Controller object instance.

8.9 *Controller Object* — The Controller object definition is provided in SEMI E54.1. This object instance is the device component which provides the closed-loop control of mass flow. There are no additional attributes, services, or behavior defined for this object in this document.

8.9.1 *Controller Object Attributes* — The following attribute table is provided to enhance the C Class Attribute table and provides a specific attribute "Form" definition.

Table 28 C Class Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Alarm Settling Time	CA21	RW	N	Real
Warning Settling Time	CA24	RW	N	Real

8.10 *Local Link Object* — The Local Link object is defined in SEMI E54.1. This object instance is the device component which provides a mechanism of linking two attributes within the device. There are no additional attributes, services, or behavior defined for this object in this document.

8.11 *SISO Object* — The SISO object provides a Single Input, Single Output function. At this level, only the input and output attributes are defined. An instance of this object makes no sense, because there is no transfer function defined, but rather, the attributes, services, and behavior of this object are inherited by the next level down (e.g., SISO-Setpoint Object).

8.11.1 *SISO Object Attributes*

Table 29 SISO Function Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Input	A1	RW	Yes	Data Type
Output	A2	R	Yes	Data Type
Data Type	A3	R	No	USINT
Reserved	A4–A32	—	—	For future revisions to this object.
Reserved	A33–A64			For next level object.
Manufacturer-Specified	> A64	—	—	For manufacturer specification.

8.11.1.1 *Input* — An attribute which specifies the input value, or independent variable, for the transfer function. The data type is specified by the *data type* attribute.

8.11.1.2 *Output* — An attribute whose value is the output, or dependent variable, of the transfer function. The data type is specified by the *data type* attribute.

8.11.1.3 *Data Type* — An attribute which specifies the data type of the *input* attribute and the *output* attribute. The format and values of this attribute are defined in SEMI E54.1.

8.11.1.4 *Initial and Default Values*

Table 30 Table 30 SISO Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Input	LVV	0	
Output	LVV	manufacturer-specified	
Data Type	LVV	manufacturer-specified	

8.11.2 *SISO Object Behavior* — The only behavior defined at this level is that of a continuously operating transfer function: The Output attribute value is calculated as a function of the Input attribute value. The specific transfer function is defined by lower level objects which inherit from this object.

Output = F(Input)

where: F is the function specified

8.12 *SISO-Setpoint Object* — The SISO-Setpoint Object inherits from the SISO object. The definitions added by the SISO-Setpoint object include the behavior associated with the transfer function.

8.12.1 *SISO-Setpoint Object Attributes*

Table 31 SISO-Setpoint Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Ramp Type	A33	RW	No	USINT
Ramp Rate	A34	RW	No	Data Type
Ratio	A35	RW	No	REAL
Reserved	A36–A64	—	—	Reserved for future expansion.

8.12.1.1 *Ramp Type (Optional)* — An attribute which specifies a mechanism by which the *output* attribute is ramped to the current value of the *input* attribute.

0 = Disable

1 = Time in seconds

2 = Amount per second

3–63 = Reserved

64–255 = Manufacturer-specified

8.12.1.2 *Ramp Rate (Optional)* — An attribute specified to define the ramp rate at which the SISO-Setpoint object instance tracks towards the current *input* value. The ramp rate specifies how quickly the *output* is ramped from the previous *output* value to the current *input* value. This attribute is expressed in terms of seconds or amount of change per second.

8.12.1.3 *Ratio (Optional)* — An attribute which specifies the ratio multiplier to be applied to the *input* attribute value prior to the application of the ramp transfer function.

8.12.1.4 *Initial and Default Values*

Table 32 SISO-Setpoint Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Ramp Type	LVV	Disable	
Ramp Rate	LVV	0	
Ratio	LVV	1.0	

8.12.2 *SISO-Setpoint Object Behavior* — The general behavior exhibited by the SISO-Setpoint object instance is defined in SEMI E54.1. The specific behavior associated with this object is defined below.

8.12.2.1 *SISO-Setpoint OPERATING Application Process–Ramp* — The *ramp type* and *ramp rate* attributes determine the conditions under which the value of the *output* attribute is calculated and modified by this application process. The *ramp rate* attribute is defined to express the rate at which the *output* attribute is ramped to the value of the *input* attribute based on the value of the *ramp type* attribute. Table 33 describes the behavior of the SISO-Setpoint object instance based on the value of the *ramp type* attribute.

Table 33 SISO-Setpoint Object Behavior - Ramp

Ramp Type Value	*SISO-Setpoint Object Instance Behavior*
Disabled	Achieve a new output value in one step.
Time in Seconds	Achieve a new output value in the time interval specified by the *ramp rate* attribute.
Amount per Second	Achieve a new output value in the amount per second increments specified by the *ramp rate* attribute.

8.12.2.2 *SISO-Setpoint OPERATING Application Process–Ratio* — Prior to the application of the Ramp calculation, a multiplier is applied to the value of the *input* attribute. The value of the *ratio* attribute is multiplied to the value of the *input* attribute, and the result becomes the value which is used in the ramp calculation.

The combined formula for the SISO-Setpoint object is as follows:

Output = F(x)

where: F is the function as defined by ramp type above

and: x = (ratio)(input)

8.13 *Assembly-MFC Object* — The Assembly-MFC object inherits attributes, services, and behavior from the Assembly object. The Assembly object instance is the device component which provides a mechanism of grouping more than one attribute from one or more object instances into a single data structure for communication over the network.

8.13.1 *Assembly-MFC Object Attributes*

Table 34 Assembly-MFC Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Data	A1	R	Yes	Structure as defined below.

8.13.1.1 *Data* — An attribute with the following list of attributes within its structure:

Table 35 Assembly-MFC Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	MFD8	A1	Controller Status
3	MFD3	A4	Sensor-AI-MF Value
4	MFD8	A4	Controller Setpoint
5	MFD8	A6	Controller Control Variable

SEMI E54.4-0704
STANDARD FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR DEVICENET

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North AmericanRegional Standards Committee on April 22, 2004. Initially available at www.semi.org June 2004; to be published July 2004. Originally published September 1997.

1 Purpose

1.1 This standard defines a communication specification based on the DeviceNet protocol to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI- specified device models (common and device specific) in a semiconductor manufacturing tool.

1.2 *Background and Motivation* — DeviceNet provides for networking between simple industrial devices (e.g., sensors and actuators) and higher level devices such as controllers. DeviceNet provides:

- A solution to low-level device networking.
- Access to intelligence present in low-level devices.
- Master/Slave and Peer-to-Peer capabilities.

1.2.1 DeviceNet is based on the Controller Area Network (CAN) technology. CAN defines a Media Access Control (MAC) methodology and physical signaling characteristics. DeviceNet wraps a communication model and protocol as well as a complete Physical Layer definition around CAN to provide a complete network definition.

1.3 This document enables communications between intelligent devices on a SEMI-compliant SAN by providing a presentation mapping of common and specific device network visible structure and behavior to a DeviceNet network.

2 Scope

2.1 This document specifies the protocol and services that compliant intelligent devices must support to interchange information over this semiconductor equipment sensor/actuator network.

2.2 This document specifies the utilization of the DeviceNet protocol to present externally visible device structure and behavior, specified in the Common Device Model (CDM) and appropriate Specific Device Models (SDM's), on a DeviceNet network.

2.3 This document is used in conjunction with a SEMI standard SAN Common Device Model specification and one or more SEMI standard-specific device model specifications (e.g., for a mass flow controller). Together, they describe the externally visible data structure and behavior of devices using the DeviceNet networking capability in a SEMI-compliant SAN system.

2.4 This standard, together with a sensor/actuator network interoperability guideline, the sensor/actuator network common device model, one or more sensor/actuator network specific device model documents, and the DeviceNet specifications, form a complete interoperability standard. The general sensor/actuator network document architecture is shown in the Sensor/Actuator Network Common Device Model document in Figure 1.

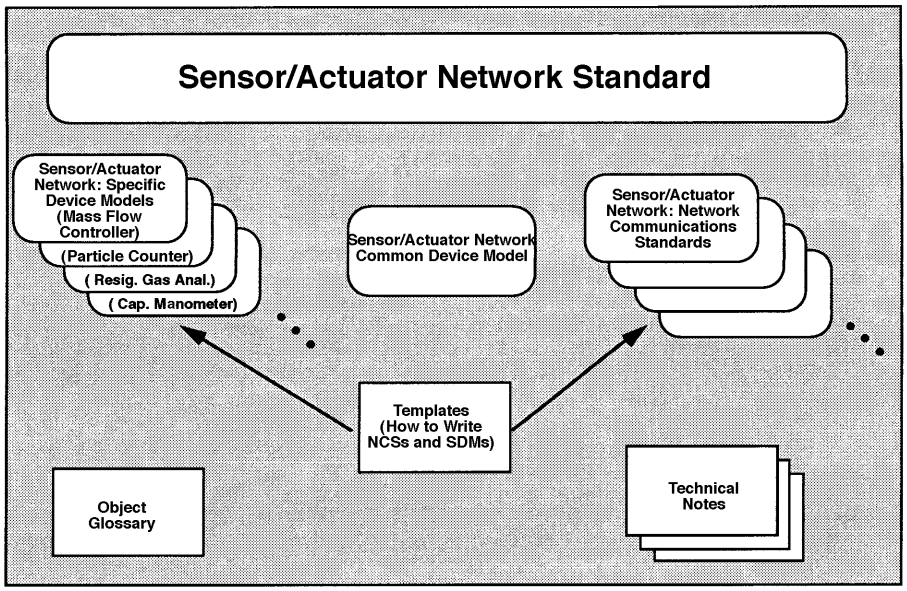


Figure 1
Sensor/Actuator Network Related Documents

2.5 *Document Structure* — The DeviceNet network communication standard complies with the SEMI SAN NCS template document structure; this structure is shown in Figure 2. The standard document is composed of two main parts. The first part (Sections 1 through 8) specifies the SAN enabling protocol as well as the presentation (i.e., mapping) of CDM object structure and behavior onto the network (referred to as the "CDM mapping"). The second part (Section 9) specifies the presentation (i.e., mapping) of SDM object structure and behavior onto the network for each SEMI-specified SDM (referred to as the "SDM mapping").

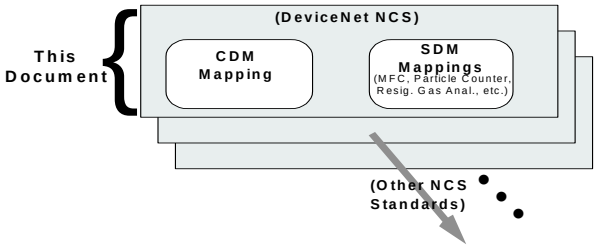


Figure 2
DeviceNet NCS Document Structure

2.6 *Adding SDM Mappings* — SDM mappings added to part two of this document are considered document additions and are balloted as such. An SDM mapping may only be balloted for addition to this document if the corresponding SEMI SDM has been standardized or is in the process of being balloted for standardization.

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 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on DeviceNet and is a companion document to the DeviceNet specification; thus, a complete specification of this standard necessarily includes the DeviceNet specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by DeviceNet. In order to avoid document consistency problems, information in the DeviceNet specification that relates to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the DeviceNet specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the DeviceNet protocol conformance testing specification. Conformance testing shall also include enhancements and limitations to the DeviceNet specification required by this standard.

4 Referenced Standards

4.1 *SEMI Standards*

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

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

4.2 *ISO Standards*[1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection

ISO 11898 — Road Vehicles — Interchange of Digital Information — Controller Area Network (CAN) for High-Speed Communications

4.3 *Other Documents*

DeviceNet Specification — ODVA[2]

Controller Area Network Specification — Version 2, R. Bosch GmbH, + Postfach 50 D-7000, Stuttgart 1, Germany, 1991

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

5 Terminology

Terminology that is common to all of the documents in this SAN standard may also be defined in the Sensor Actuator Network Standard. Terminology may be reproduced here which is defined in other SEMI documents.

5.1 *Abbreviations & Acronyms*

5.1.1 *CAN* — Controller area network

5.1.2 *CDM* — Common device model

5.1.3 *DM* — Device manager (object)

5.1.4 *DN* — DeviceNet

5.1.5 *NCS* — Network communication standard

5.1.6 *OSI* — Open systems interconnect

5.1.7 *OSS* — Object services standard

5.1.8 *SAC* — Sensor, actuator, controller (object)

5.1.9 *SAN* — Sensor/actuator network

5.1.10 *SDM* — Specific device model

5.2 *Device Component Definitions* — As this standard defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in the CDM document. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the DeviceNet specification. Note that Column 2 contains an equal sign "=" if the definition is used exactly as specified in the CDM specification.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*	*DeviceNet Equivalent*
Device	=	=
Device Model	=	=
Object	=, Class	=, Class
Instance	=	=
Attribute	=	=
Behavior	=	=
Service	=	=
State Diagram	=	=
Byte	=	=
Nibble	=	=
Character String	=	=

5.3 *DeviceNet Specific Definitions*

5.3.1 *class* — a set of objects that all represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values.

5.3.2 *controller area network (CAN)* — a protocol developed by the Bosch corporation for automotive in-vehicle networking. The CAN specification specifies OSI reference model layers 1 and 2, specifically the physical signaling and media access/data link protocols.

5.3.3 *device profile* — a DeviceNet specification for a device that contains an object model for the device type, the I/O data format for the device type, and the configuration data and the public interface(s) to that data.

5.3.4 *explicit message connections* — connections over a DeviceNet network that provide generic, multi-purpose communication paths between two devices. These connections often are referred to as just messaging connections. Explicit messages provide the typical request/response - oriented network communications.

1 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30, Website: www.iso.ch

2 Open DeviceNet Vendor Association, www.odva.org

5.3.5 *input/output connections* — connections over a DeviceNet network that provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data moves through these ports.

6 Communication Protocol High Level Structure

6.1 The DeviceNet protocol is loosely based on a three-layer architecture. These layers constitute a collapsed form of the OSI seven-layer architecture, mapping into the physical, data link, and application layers of the Reference Model; however, DeviceNet provides additional functionality (such as connection support) commonly attributed to other OSI layers. The high level protocol architecture is shown in Figure 3.

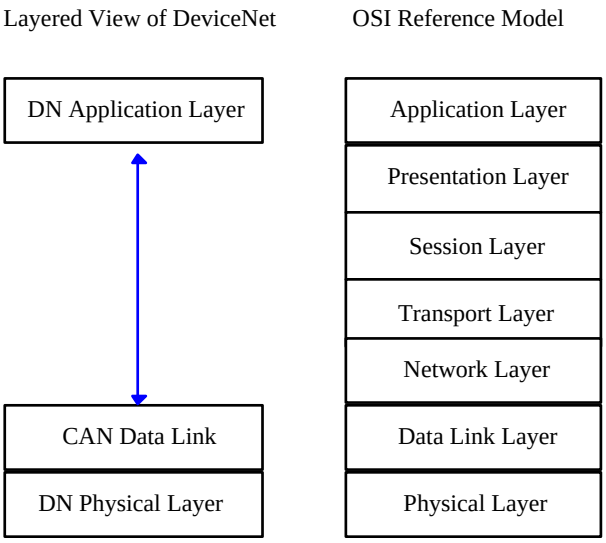


Figure 3
Layered View of DeviceNet

6.1.1 Note that Figure 3 represents a conceptual view of the device architecture. Conforming implementations must implement the services defined in this specification at each layer and must appear (from the network) to have implemented this architecture; however, an internal modular partitioning is not required. Implementations may sacrifice modularity in order to achieve high performance.

6.2 The DeviceNet physical layer is fully specified in Volume 1 of the DN Specification. Features of this layer include Trunkline - dropline configuration, simultaneous support for both network-powered and self-powered devices, and selectable data rates including at least 125k, 250k, and 500k baud. At the data link layer, the CAN specification defines a carrier sense multiple access mechanism for media access control that avoids collisions and sends frames reliably. The application layer is specified in Volumes 1 and 2 of the DN Specification and provides for the definition of DeviceNet applications as a collection of addressable objects. Two basic categories of objects exist: Communication Objects and Application Objects. Communication Objects manage and provide for the runtime exchange of messages across DeviceNet. Application Objects implement product-specific features and/or provide a logical interface to product-specific information of devices on a DeviceNet network.

6.3 In the remainder of this section, the protocol structure is described in more detail in terms of the OSI seven layer reference model, the object model environment, and network management specifications.

6.4 *Physical Layer* — The device shall comply with the DeviceNet physical layer specification as well as the "Interface Guidelines for DeviceNet Devices on Semiconductor Manufacturing Tools" which is published as part of the DeviceNet specification. This includes physical signaling (levels and baud rates - detailed in the CAN specification), transceivers, node isolation, media topology, cable specifications, network connectors and taps, and power considerations (load limits, system tolerances, and power supply options).

6.4.1 *Physical Layer Enhancements* — This standard supports the following physical layer enhancement to the DeviceNet specifications:

- The transmission media, including both trunk cable and drop cable, shall be DeviceNet Thin Cable. Specifications on this cable and its utilization in DeviceNet systems (e.g., maximum trunk and drop lengths) are included in the DeviceNet specification.

6.5 *Data Link Layer* — The device shall comply with the DeviceNet Data Link Layer Specifications (i.e., Controller Area Network Specification: Version 2). This includes the media access control mechanism and the logical link control mechanism. Addressing is currently limited to 11 bits.

6.6 *Network Layer* — There is no distinct network layer.

6.7 *Transport Layer* — There is no distinct transport layer. Some of the functionality of this layer is implemented in the Application Layer. Specific functions include: segmentation/reassembly for full message delivery and the establishment of node-to-node connections.

6.8 *Session Layer* — There is no distinct session layer.

6.9 *Presentation Layer* — There is no distinct presentation layer. Data types and data presentation in DeviceNet messages are specified as part of the DeviceNet object definitions and object attribute and service communication protocol.

6.10 *Application Layer* — The device shall comply with the DeviceNet application layer specification for defining and addressing objects, including their attributes and services, and enabling specified network behavior. The device shall comply with the object model specifications provided in the DeviceNet specification as well as the "Interface Guidelines for DeviceNet Devices on Semiconductor Manufacturing Tools" component of the DeviceNet specification. In addition, the device shall comply with the object specifications defined in Section 7 of this document.

6.10.1 *Object Models* — The DeviceNet protocol provides an object-oriented specification for creating, defining, and addressing objects explicitly, including their attributes and services (i.e., explicit messaging), and creating, defining, and communicating object attribute assemblies in an application-dependent format (i.e., input/output messaging). The device shall comply with the object model specifications provided in the DeviceNet documentation. In addition, the device shall comply with the object specifications defined in Section 7 of this document.

6.11 *Network Management* — The device shall comply with the DeviceNet network management specifications (e.g., physical layer bit rate, duplicate MAC ID detection, master-slave, and peer-to-peer network management). No (additional) network management functions are specified in this document.

7 Required Object Types

7.1 The DeviceNet specification identifies and describes objects (i.e., classes) that must exist in all DeviceNet-compliant devices. The Common Device Model specification additionally identifies two objects (namely the Device Manager (DM) and Sensor Actuator Controller (SAC) objects) that must exist in all SEMI-compliant SAN devices. The required object types for a SEMI-compliant SAN device, using the network communication specification described herein, necessarily comprise the union of the above to requirements.

7.2 A list of required and optional object types is given in Table 2. Note that the Sensor, Actuator, and Controller object types are not required and are indicated as optional in the CDM specification. These objects are aggregated together to form a SEMI- and DN- compliant device, as shown in Figure 4.

7.3 The mapping of the DM and SAC objects are combined into a single object in DeviceNet called the S-Device Supervisor (DS) object.

Table 2 Required Object Types

Object	*DN Class #**	*CDM Tag **	*Required by DN **	*Required by CDM **	*Required by NCS*
Identity	01	N.A.***	Yes	No	Yes
MR	02	N.A.	Yes	No	Yes
DN	03	N.A.	Yes	No	Yes
CNX	05	N.A.	Yes	No	Yes
DM(DS)	48	DmI0	No	Yes	Yes
SAC(DS)	48	SACI0	No	Yes	Yes
Sensor	****	SenIn	No	No	No
Actuator	****	ActIn	No	No	No
Controller	****	CntIn	No	No	No
(Other)	****	N.A.	No	No	No

* See DeviceNet specification for further information; values are hexadecimal.

** See CDM specification for further information.

*** Not applicable.

**** Application-dependent.

7.4 An embodiment of a specific device type represented as an aggregation of the object types listed in Table 2, that is compliant with both the CDM specification and the DN specification, is a candidate for a SEMI SDM as well as a DN Device Profile. Conversely, all SEMI SDM's and DN Device profiles specified for operation over a SEMI-compliant DN network must be an aggregation of the object types listed in Table 2 and be compliant with both the CDM specification and the DN specification.

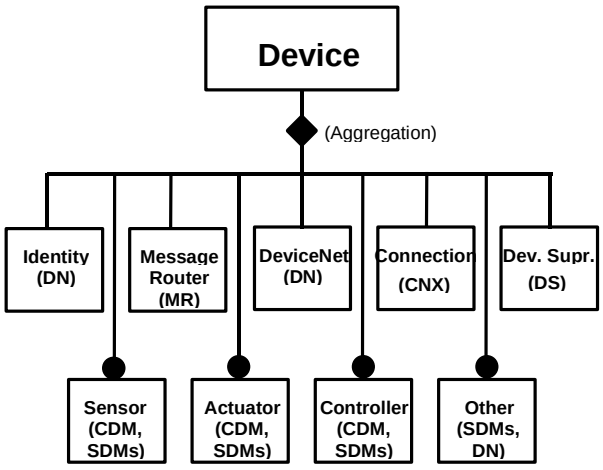


Figure 4
Aggregation of a Compliant Device

7.5 In the following sections, the presentation to the network of object addressing, object attributes, and object services for each of the object types listed in Table 2 and Figure 4 is described in detail.

7.6 *Identity Object* — This object provides identification of, and general information about, the device. The Identity Object must be present in all DeviceNet products. As specified by DeviceNet, each DeviceNet product shall support one (and only one) Identity object per physical connection to the DeviceNet communication link. Presentation of object identity, attributes, and services to the network for this object is described in the DeviceNet specification. Compliance with the DeviceNet specification shall constitute compliance with this NCS for the Identity Object.

7.7 *Message Router (MR) Object* — The MR Object provides a messaging connection point through which a service of any object class or instance residing in the physical device may be addressed. The MR object must be present in all DeviceNet products. As specified by DeviceNet, each DeviceNet product shall support one (and only one) Message Router object per physical connection to the DeviceNet communication link. Presentation of object identity, attributes, and services to the network for this object is described in the DeviceNet specification. Compliance with the DeviceNet specification shall constitute compliance with this NCS for the MR Object.

7.8 *DeviceNet (DN) Object* — The DN Object provides the configuration and status of a DeviceNet port. As specified by DeviceNet, each DeviceNet product shall support one (and only one) DN object per physical connection to the DeviceNet communication link. Presentation of object identity, attributes, and services to the network for this object is described in the DeviceNet specification. Compliance with the DeviceNet specification shall constitute compliance with this NCS for the DN Object.

7.9 *Connection (CNX) Object* — The CNX Object provides configuration and management of DeviceNet connections. A CNX object exists at each end of a DeviceNet connection (point-to-point or multicast). The CNX object handles the negotiation for connection establishment and manages a set of timers in order to handle cyclic traffic, connection timeout and fault containment and recovery. Several connection behaviors are supported including: explicit messaging, polled, cyclic, change-of-state and multicast messaging.

7.10 *S-Device Supervisor (DS) Object* — The DS object is the device component responsible for managing and consolidating the device operation as well as coordinating the interaction of the device with the sensory/actuation/control environment. Each device must support one (and only one) DS object. The DS

object combines both the DM object and the SAC object into a single object for DeviceNet. The DM and SAC, as well as their common required and optional attributes, services, and behavior, are described in the CDM standard. The presentation of object attributes and services to the DN network shall be as indicated in Table 3.

Table 3 Network Presentation of DM Object Attributes and Services

S-Device Supervisor Object - - Object ID == 48			
Attributes			
ID	**Name**		**CDM Tag**
3	Device Type		DmA1
4	Standard Revision Level		DmA2
5	Device Manufacturer Identifier		DmA3
6	Manufacturer Model Number		DmA4
7	Software or Firmware Revision Level		DmA5
8	Hardware Revision Level		DmA6
9	Serial Number (optional)		DmA7
10	Device Configuration (optional)		DmA8
11	Device Status		DmA9
-- *	Reporting Mode		DmA10
-- *	Exception Status Report Interval (optional)		DmA11
12	Exception Status		DmA12
13	Exception Detail Alarm (optional)		DmA13
14	Exception Detail Warning (optional)		DmA14
Services			
ID (hex)	**Name (SEMI)**	**Name (DN)**	**CDM Tag**
05	Reset	Reset	DmS1
4B	Abort	Abort	DmS2
4C	Recover	Recover	DmS3
0E	Get_Attribute	Get_Attribute_Single	DmS4
10	Set_Attribute	Set_Attribute_Single	DmS5
06	Execute	Start	DmS6
4E	Perform_Diagnostics	Perform_Diagnostics	DmS7

* There is no one-to-one mapping of the Reporting Mode attribute in DeviceNet. Instead, DeviceNet specifies several attributes, services and behaviors for its Connection Object to effect a superset of the behaviors associated with the DM attributes Reporting Mode and Exception Status Report Interval. See the DeviceNet specification for details.

7.10.1 Note that the format of DM object attributes is detailed in the CDM document; the presentation of DM object attributes to the DN network is detailed in Table 3 and the DN specification; the format of DM object services is detailed in the CDM document and the DN specification; and the presentation of the DM object services is detailed in Table 3 and the DN specification.

7.11 *Sensor Object, Actuator Object, Controller Object, and Other Object Types* — These object types are used collectively to model the type-specific structure and behavior of the device. The requirement and number of each of these object types in a device model is device type-specific. Further, the attributes, services, and behavior associated with each of these object classes and instances in a device is also device type-specific, but must be compliant with both SEMI and DN specifications. The specification of these object types for a specific device type can be found in the appropriate SDM. The method of presentation of object structure and behavior to the DN network for objects defined for, and associated with, a specific device type can be found in Section 9 of this document.

8 Protocol Compliance

8.1 A method of testing protocol compliance is required to verify implementation conformance to the standard. The test plan includes tests for duplicate address resolution, mandatory objects, etc.

8.2 The compliance test suite for this protocol necessarily includes the DeviceNet protocol compliance specification and test suite as well as the “Conformance Test Procedures for DeviceNet Devices on Semiconductor Manufacturing Tools” which is published as part of the DeviceNet specification. Additional compliance specification required for compliance to this NCS is provided as a set of Protocol Specification Sheets.

8.3 Any enhancements to DeviceNet presented in this document must be accepted by the Open DeviceNet Vendors Association (ODVA) and incorporated into the DeviceNet Specification before they can be implemented as a DeviceNet product.

8.4 *Protocol Specification Sheets* — Compliance to this NCS necessarily requires adherence to a set of protocol specification sheets. These sheets are included as Appendix 1 of this document and are included here for reference only. For compliance with this NCS, it is necessary to reference the latest revision of the Protocol Specification Sheets from ODVA. Note that these sheets provide statements of compliance for general device data, physical conformance data, communications data, required object implementation (see also Section 7 or this document), and optional object implementation. Note also that these specification sheets must conform with DeviceNet specifications for “Statement of Compliance” forms. Finally note that additional specification sheets shall be completed as necessary to specify compliance with objects defined in SDM’s and SDM mappings (see Section 9).

9 Specific Device Model Mappings

9.1 This section provides for the mapping of network-visible specific device structure and behavior, specified in a SEMI standard SDM specification, to the DN network. Each subsection is devoted to a single SDM specification (e.g., 9.1: Mass Flow Controller). Additional SDM mappings are added as sub-sections to this NCS specfication according to SEMI guidelines and the guidelines of the SEMI SAN Interoperability standard.

APPENDIX 1
PROTOCOL SPECIFICATION SHEETS

NOTICE: This appendix is not an official part of SEMI E54.4 and is not intended to modify or supercede the official standard. Determination of the suitability of the material is solely the responsibility of the user. This appendix contains a set of protocol specification sheets. Compliance to this NCS necessarily requires adherence to this set of protocol specification sheets (see Section 8.1).

A1-1 The protocol specification set of sheets contains the following components:

- A General Device Data/Physical ConformanceData/ DeviceNet Communication Data statement of compliance. (Form F_1, 1 page.)
- DeviceNet required object implementation statements of compliance (for Identity, Message Router, and DeviceNet objects, see Section 7). (Forms F_2 through F_5, 4 pages.)
- NCS - DeviceNet required object implementation statements of compliance (for Device Manager and Sensor/Actuator/Controller objects, see Section 7). (Form F_6, 2 pages.)
- Open object- and vendor- specific implementation templates (for utilization by SDMs, vendors, etc.). (Forms F_6 through F_7, 2 pages.)

A1-1.1 Note that these protocol specification sheets are aligned with DeviceNet specifications for “Statement of Compliance” documentation, available with the DeviceNet specification from ODVA. Detailed instructions on completing these forms are in this “Statement of Compliance” documentation. The set of protocol specification sheets begins on the following page.

DeviceNet **Statement of Compliance**

Complete this form using the definitions previously outlined.

					Fill in the blank or X the appropriate box
General Device Data	Conforms to DeviceNet Specification	Volume I - Release			_____
		Volume II - Release			_____
	Vendor Name				_____
	Device Profile Name				_____
	Product Catalog Number				_____
	Product Revision				_____
DeviceNet Physical Conformance Data	Network Power Consumption (Max)				_____A @ 11V dc (worst case)
	Connector Style	Open-Hardwired	❑	Sealed-Mini	❑
		Open-Pluggable	❑	Sealed-Micro	❑
	Isolated Physical Layer	Yes ❑			
		No ❑			
	LEDs Supported	Module	❑	Combo Mod/Net	❑
		Network	❑	I/O	❑
		DIP Switch	❑	Software-Settable	❑
		Other_____			
	Default MAC ID	_____			
	Communication Rate Setting	DIP Switch	❑	Software-Settable	❑
		Other_____			
	Communication Rates Supported	125k bit/s	❑	500k bit/s	❑
		250k bit/s	❑		
DeviceNet Communication Data	❑ Predefined Master/Slave Connection Set	Group 2 Client	❑	Group 2 Only Client	❑
		Group 2 Server	❑	Group 2 Only Server	❑
	❑ Dynamic Connections Supported (UCMM)	Group 1	❑	Group 3	❑
		Group 2	❑		
	Fragmented Explicit Messaging Implemented	Yes ❑			
		No ❑			
		If yes, Transmission Time Out_____ms			
		Typical Target Address		Class_____	
				Instance_____	
				Attribute_____	

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

F_ 1

DeviceNet			Statement of Compliance			
DeviceNet Required Object Implementation	**Identity Object 0x01**					
	Object Class		**ID**	**Description**	**Get**	**Set** **Value Limits**
	Attributes	Open	1	Revision	❑	❑ _____
			2	Max instance	❑	❑ _____
	❑ None Supported		6	Max ID of class attributes	❑	❑ _____
			7	Max ID of instance attributes	❑	❑ _____
			DeviceNet Services		**Parameter Options**	
	Services		❑	Get_Attribute_All	_____	
			❑	Reset	_____	
	❑ None Supported		❑	Get_Attribute_Single	_____	
			❑	Find_Next_Object_Instance	_____	
	Object Instance		**ID**	**Description**	**Get**	**Set** **Value Limits**
	Attributes	Open	1	Vendor	❑	❑ _____
			2	Product type	❑	❑ _____
	❑ None Supported		3	Product code	❑	❑ _____
			4	Revision	❑	❑ _____
			5	Status	❑	❑ _____
			6	Serial number	❑	❑ _____
			7	Product name	❑	❑ _____ _____
			8	State	❑	❑ _____
			DeviceNet Services		**Parameter Options**	
	Services		❑ Reset		_____	
			❑ Get_Attribute_All		_____	
	❑ None Supported					
	Vendor-Specific Additions		If yes, fill out the Vendor-Specific Additions form on page F_7.		Yes ❑ No ❑	

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

DeviceNet			Statement of Compliance			
DeviceNet			**Message Router Object 0x02**			
Required Object Implementation	**Object Class**		**ID**	**Description**	**Get**	**Set** **Value Limits**
	Attributes	Open	1	Revision	❑	❑ _____
			4	Optional attribute list	❑	❑ _____
	❑ None Supported		5	Optional service list	❑	❑ _____
			6	Max ID of class attributes	❑	❑ _____
			7	Max ID of instance attributes	❑	❑ _____
			DeviceNet Services		**Parameter Options**	
	Services		❑	Get_Attribute_all	_____	
			❑	Get_Attribute_Single	_____	
	❑ None Supported					
	Object Instance		**ID**	**Description**	**Get**	**Set** **Value Limits**
	Attributes	Open	1	Object list	❑	❑ _____
			2	Maximum connections supported	❑	❑ _____
	❑ None Supported		3	Number of active connections	❑	❑ _____
			4	Active connections list	❑	❑ _____
			DeviceNet Services		**Parameter Options**	
	Services		❑	Get_Attribute_All	_____	
			❑	Get_Attribute_Single	_____	
	❑ None Supported					
	Vendor-Specific Additions		If yes, fill out the Vendor-Specific Additions form on page F_7.		Yes ❑	No ❑

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

DeviceNet			Statement of Compliance		
DeviceNet Required Object Implementation	**DeviceNet Object 0x03**				
	Object Class		**ID**	**Description**	**Get Set Value Limits**
	Attributes	Open	1	Revision	❑ ❑ _____
	❑ None Supported				
			DeviceNet Services		**Parameter Options**
	Services		❑	Get_Attribute_Single	_____
	❑ None Supported				
	Object Instance		**ID**	**Description**	**Get Set Value Limits**
	Attributes	Open	1	MAC ID	❑ ❑ _____
			2	Baud rate	❑ ❑ _____
	❑ None Supported		3	BOI	❑ ❑ _____
			4	Bus-off counter	❑ ❑ _____
			5	Allocation information	❑ ❑ _____
			6	MAC ID switch changed	❑ ❑ _____
			7	Baud rate switch changed	❑ ❑ _____
			8	MAC ID switch value	❑ ❑ _____
			9	Baud rate switch value	❑ ❑ _____
			DeviceNet Services		**Parameter Options**
	Services		❑	Get_Attribute_Single	_____
			❑	Set_Attribute_Single	_____
	❑ None Supported		❑	Allocate M/S connection set	_____
			❑	Release M/S connection set	_____
	Vendor-Specific Additions		If yes, fill out Vendor-Specific		Yes ❑
			Additions form on page F_7.		No ❑

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

DeviceNet	Statement of Compliance					
DeviceNet Required Object Implementation	**Connection Object 0x05**					
	Object Class		**ID**	**Description**	**Get Set Value Limits**	
	Attributes ❑ None Supported	Open	1	Revision	❑ ❑	_____
			DeviceNet Services		**Parameter Options**	
	Services ❑None Supported		❑	Reset		_____
			❑	Create		_____
			❑	Delete		_____
			❑	Get_Attribute_Single		_____
			❑	Find_Next_Object_Instance		_____
	Total Active Connections Possible		_____			
	Object Instance		**Section**		**Information Max**	
	The Object Instance section must be completed for each combination of Instance type, Production trigger, Transport type, and Transport class supported		Instance type		Explicit Message ❑	_____
					Polled I/O ❑	_____
					Bit Strobed I/O ❑	_____
					Dynamic I/O ❑	_____
			Production trigger		Cyclic ❑	_____
					Change of State ❑	_____
					Application Trig. ❑	_____
			Transport type		Server ❑	_____
					Client ❑	_____
			Transport class		0 ❑	_____
					2 ❑	_____
					3 ❑	_____
			ID	**Description**	**Get Set Value Limits**	
	Attributes	Open	1	State	❑ ❑	_____
			2	Instance type	❑ ❑	_____
			3	Transport class trigger	❑ ❑	_____
			4	Produced connection ID	❑ ❑	_____
			5	Consumed connection ID	❑ ❑	_____
			6	Initial comm. characteristics	❑ ❑	_____
			7	Produced connection size	❑ ❑	_____
			8	Consumed connection size	❑ ❑	_____
			9	Expected packet rate	❑ ❑	_____
			12	Watchdog time-out action	❑ ❑	_____
			13	Produced connection path length	❑ ❑	_____
			14	Produced connection path	❑ ❑	_____
			15	Consumed connection path length	❑ ❑	_____
			16	Consumed connection path	❑ ❑	_____
			DeviceNet Services		**Parameter Options**	
	Services		❑	Reset		_____
			❑	Delete		_____
			❑	Apply_Attributes		_____
			❑	Get_Attribute_Single		_____
			❑	Set_Attribute_Single		_____
	Vendor-Specific Additions		If yes, fill out the Vendor-Specific Additions form on page F_7.		Yes ❑ No ❑	

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

F_ 5

DeviceNet Open Object Implementation

OBJECT NAME Device Manager **OBJECT ID** 64

		ID	Description	Get	Set	Value Limits
Object Class						
Attributes	Open	_____	_____	❑	❑	_____
		DeviceNet Services				**Parameter Options**
Services		_____				_____
Object Instance		**ID**	**Description**	**Get**	**Set**	**Value Limits**
Attributes	Open	31	Device Type	❑	❑	_____
		32	Standard Revision Level	❑	❑	_____
		33	Device Manufacturer ID	❑	❑	_____
		34	Manufacturer Model Number	❑	❑	_____
		35	Soft/Firmware Rev. Level	❑	❑	_____
		36	Hardware Revision Level	❑	❑	_____
		37	Serial Number	❑	❑	_____
		38	Device Configuration	❑	❑	_____
		39	Device Status	❑	❑	_____
		3A	Reporting Mode	❑	❑	_____
		3B	Exception Status Timer	❑	❑	_____
		3C	Exception Status	❑	❑	_____
		3D	Exception Detail Alarm	❑	❑	_____
		3E	Exception Detail Warning	❑	❑	_____
		DeviceNet Services				**Parameter Options**
Services		❑	Reset			_____
		❑	Abort			_____
		❑	Recover			_____
		❑	Get_Attribute_Single			_____
		❑	Set_Attribute_Single			_____
		❑	Execute			_____
		❑	Perform Diagnostics			_____
Meaning of Zero Length I/O Data Production_____						
Vendor-Specific Additions		If yes, fill out the Vendor-Specific Additions form on page F_7.		Yes ❑ No ❑		

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written by the use of Set_Attribute_Single service.

DeviceNet	OBJECT NAME Sensor Actuator Controller		OBJECT ID 66	
Open Object Implementation	**Object Class**	**ID Description**	**Get Set Value Limits**	
	Attributes Open	_____ _____	❑ ❑	_____
		_____ _____	❑ ❑	_____
		_____ _____	❑ ❑	_____
		DeviceNet Services	**Parameter Options**	
	Services	_____	_____	
		_____	_____	
		_____	_____	
	Object Instance	**ID Description**	**Get Set Value Limits**	
	Attributes Open	_____	❑ ❑	_____
		_____	❑ ❑	_____
		_____	❑ ❑	_____
		DeviceNet Services	**Parameter Options**	
	Services	❑ Reset	_____	
		❑ Abort	_____	
		❑ Recover	_____	
		❑ Get_Attribute_Single	_____	
		❑ Set_Attribute_Single	_____	
		_____	_____	
		_____	_____	
		_____	_____	
	Meaning of Zero Length I/O Data Production_____			
	Vendor-Specific Additions	If yes, fill out the Vendor-Specific Additions form on page F_7.	Yes ❑	
			No ❑	

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

DeviceNet	**OBJECT NAME**_____				**OBJECT ID**_____	
Open Object Implementation	**Object Class**		**ID Description**		**Get Set Value Limits**	
	Attributes	Open	_____	_____	❑ ❑	_____
			_____	_____	❑ ❑	_____
			_____	_____	❑ ❑	_____
			DeviceNet Services		**Parameter Options**	
	Service		_____		_____	
			_____		_____	
			_____		_____	
	Object Instance		**ID Description**		**Get Set Value Limits**	
	Attributes	Open	1	_____	❑ ❑	_____
			2	_____	❑ ❑	_____
			3	_____	❑ ❑	_____
			DeviceNet Services		**Parameter Options**	
	Services		_____		_____	
			_____		_____	
			_____		_____	
	Meaning of Zero Length I/O Data Production_____					
	Vendor-SpecificAdditions		If yes, fill out the Vendor-Specific		Yes ❑	
			Additions form on page F_7.		No ❑	

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set indicate that attribute value is written to by the use of Set_Attribute_Single service.

F_ 8

	❑ Extension to Open Object		
	❑ Vendor-Specific Object		
Vendor Specific Object Implementation	**OBJECT NAME**_____		**OBJECT ID**_____
	Object Class	**ID Description**	**Get Set Value Limits**
	Attributes	_____ _____	❑ ❑ _____
		_____ _____	❑ ❑ _____
		_____ _____	❑ ❑ _____
		Code (Hex) Service Description	**Parameter Type/Options**
	Services	_____ _____	_____
		_____ _____	_____
		_____ _____	_____
	Object Instance	**ID Description**	**Get Set Type/Value Limits**
	Attributes	_____ _____	❑ ❑ _____
		_____ _____	❑ ❑ _____
		_____ _____	❑ ❑ _____
		Code (Hex) Service Description	**Parameter Type/Options**
	Services	_____ _____	_____
		_____ _____	_____
		_____ _____	_____
	Meaning of Zero Length I/O Data Production_____		

X Get to indicate that attribute value is returned by the use of Get_Attribute_Single service.

X Set to indicate that attribute value is written to by the use of Set_Attribute_Single service.

F_ 9

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E54.5-0997 (Withdrawn 0704)
STANDARD FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR THE SMART DISTRIBUTED SYSTEM (SDS)

NOTE: This document was previously designated SEMI E60. Because this document is part of a suite of documents, its designation has been reassigned for ease of reference. Please note that the technical content of this document is unchanged from the 0697 version.

NOTICE: This document was balloted and approved for withdrawal in 2004.

1 Purpose

This standard defines a communication protocol based on the Smart Distributed System (SDS) to enable communications between intelligent devices on a sensor/actuator network (SAN) to be used in semiconductor manufacturing equipment.

1.1 *Background and Motivation* — SDS provides interconnection of smart control devices such as sensors, actuators, and controllers in a fast-response time, low-cost network for industrial use. SDS enables multiple devices to share a single bus, thereby significantly reducing the point-to-point wiring between controllers, sensors, and actuators. The SDS system is based on CAN, the controller area network used in the automotive industry and defined by Bosch.

2 Scope

This document specifies a SAN communications standard based on the Smart Distributed System (SDS) specification that is in compliance with the SEMI SAN Common Device Model specification.

2.1 This document specifies the protocol and services that compliant intelligent devices must support to interchange information over this semiconductor equipment sensor/actuator network.

2.2 This document is used in conjunction with a SEMI standard SAN Common Device Model specification and one or more SEMI standard specific device model specifications (e.g., for a mass flow controller). Together, the model documents describe the data structure and behavior that are characteristic of the various devices on the network. This SAN communications standard identifies the protocol for the interaction with such a device over the network to make the data structures and behavior available to other devices.

2.3 This standard, together with a sensor/actuator network interoperability guideline, the sensor/actuator network common device model, one or more sensor/actuator network specific device model documents, and the SDS specifications, form a complete interoperability standard. The general sensor/actuator network document architecture is shown in the Sensor/Actuator Network Common Device Model document in Figure 1.

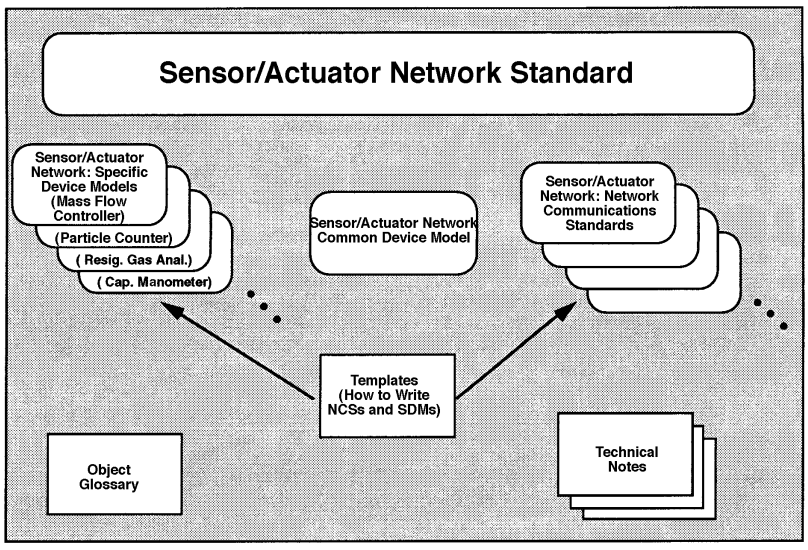


Figure 1
Sensor/Actuator Network Related Documents

The SDS Network communications standard document structure is shown in Figure 2. This standard specifies the mapping of the SAN common device model onto the SDS-specific network. In addition, the document will include mappings to specific device models (e.g., the mass flow controller). The latter mappings will be included in this document as the specific device models are defined.

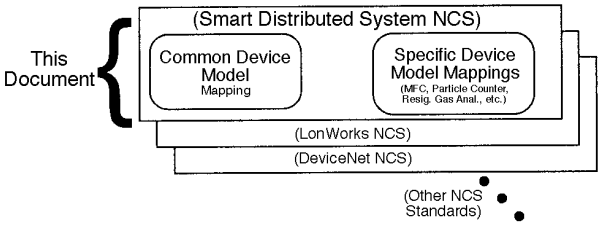


Figure 2
SDS Network Communications Standard Document Structure

3 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on SDS and is a companion document to the SDS specification; thus, a complete specification of this standard necessarily includes the SDS specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by SDS. In order to avoid document consistency problems, information in the SDS specification that relates to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the SDS specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the SDS protocol conformance testing specification. Conformance testing shall also include enhancements and limitations to the SDS specification required by this standard.

4 Referenced Standards

4.1 *SEMI Standards*

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

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

4.2 *ISO Standards*[1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection

ISO 11898 — Road Vehicles — Interchange of Digital Information — Controller Area Network (CAN) for High-Speed Communications

4.3 *Other Documents*

Doc 84-08420-1 (GS 052 103) — SMART DISTRIBUTED SYSTEM Application Layer Protocol Specification, Honeywell MICRO SWITCH, February 15, 1995

Doc 84-08421-A (GS 052 104) — SDS Physical Layer Specification, Honeywell MICRO SWITCH, December 15, 1994

Doc 85-08453-0 (GS 052 107) — SMART DISTRIBUTED SYSTEM Component Model Specification, Honeywell MICRO SWITCH, January 29, 1995

Doc GS 052 108 Issue 1 — SMART DISTRIBUTED SYSTEM Conformance Test Procedure Specification, Honeywell MICRO SWITCH, January 2, 1995

Controller Area Network Specification: Version 2, R. Bosch GmbH, + Postfach 50 D-7000, Stuttgart 1, Germany, 1991

5 Terminology

Terminology that is common to all of the documents in this SAN series may also be defined in the *Sensor Actuator Network Interoperability Guideline.* Terminology may be reproduced here which is defined in other SEMI documents.

5.1 *Device Component Definitions* — This standard is based on the concepts developed in the Sensor Actuator Network Common Device Model (SEMI E54) document and makes use of the following terms defined in it:

a) device

b) device model

c) object

d) instance

e) attribute

f) behavior

g) service

[1] International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland

h) state diagram

i) byte

j) nibble

k) character string

5.2 *SDS-Specific Definitions* — In addition to the above terms, the following terms are defined for SDS networks.

5.2.1 *actions* — operations that a client may request an Embedded Object to perform. The action typically modifies the state of the device. Actions are more powerful than writing to an attribute, in that multiple input arguments may be provided as part of an action request. Also, results of the action are typically returned. Actions, like attributes, have identifiers that are specific to the type of the object. However, these identifiers are independent from attribute or event identifiers. The "request" services specified in SEMI E54 are implemented as SDS actions.

5.2.2 *controller area network (CAN)* — a protocol developed by the Bosch corporation for automotive in-vehicle networking. The CAN specification specifies OSI reference model layers 1 and 2, specifically the physical signaling and media access/data link protocols.

5.2.3 *embedded objects* — each SDS Logical Device contains at least one, and at most 32, Embedded Objects. An SDS Embedded Object is an abstraction representing an addressable entity "embedded" within a Logical Device having specific application-related interface characteristics. These characteristics include a defined set of attributes, actions, and events that are specific to the Embedded Object. Combinations of these Embedded Objects provide a mechanism for describing an arbitrary sensor/actuator network device. The behaviors required by SEMI E54 are specified as part of the definition of attributes, actions, and events.

5.2.4 *events* — Are used by objects to asynchronously report the occurrence of events within an Embedded Object. Event definitions specify the event reports that a specific Embedded Object type may emit. Events also have type-specific identifiers. Event definitions include: a text string defining the semantics of the event, an event identifier numerical value and textual name, and a syntax definition. The "notification" services specified in SEMI E54 are implemented as SDS events.

5.2.5 *logical device* — It is possible to have one or more independent Logical Devices within the Physical Component. This provides the illusion of multiple devices on the network that actually are implemented on the same physical hardware. Logical Devices are distinguished within the Physical Component using unique SDS bus addresses. The Logical Device defines a separately addressable entity within a Physical Component that has an independent interface definition.

5.2.6 *physical component* — An SDS Physical Component is an abstraction representing a single physical package of hardware and software that is connected to the network. This is equivalent to the CDM SEMI E54 definition for "device." The Physical Component contains one or more Logical Devices.

Table 1 provides a mapping of SEMI E54 definitions to SDS-specific definitions. Column 2 contains an equal sign "=" if the definition is used exactly as specified in SEMI E54. Otherwise, the appropriate SDS-specific term(s) are identified.

Table 1 Table 1 Mapping of Terms Between SEMI E54 and SDS

SEMI E54 Term	*SDS Term*
Device	=
Device Model	Physical Component
Object	Embedded Object
Instance	=
Attribute	=
Behavior	Set of Attribute/Action/Event definitions
Service	Actions for Request Services and Events for Notification Services
State Diagram	=
Byte	=
Nibble	=
Character String	=

6 Communication Protocol High Level Structure

The SDS protocol is based on a three-layer architecture. These layers constitute a collapsed form of the OSI seven-layer architecture, mapping into the physical, datalink, and application layers of the Open Systems Interconnection Reference Model. The high level protocol architecture is shown in Figure 3.

Note that Figure 3 represents a conceptual view of the device architecture. Conforming implementations must implement the services defined in this specification at each layer and must appear (from the network) to have implemented this architecture; however, an internal modular partitioning is not required. Implementations may sacrifice modularity in order to achieve high performance.

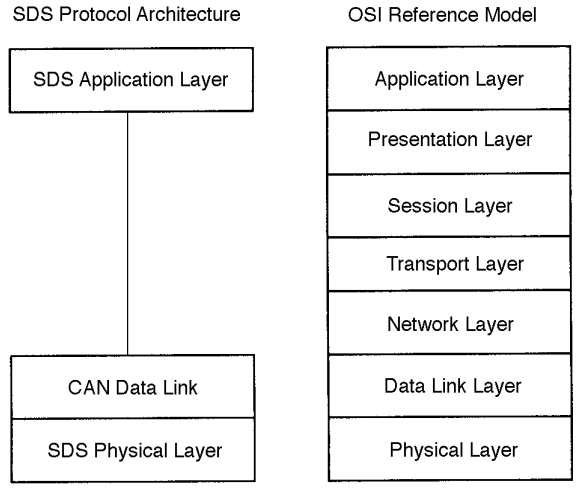


Figure 3
Protocol Architecture

SDS uses a three-layer protocol stack consisting of the physical, datalink, and application layers. The SDS physical layer uses two twisted pair to deliver power and data to smart devices at speeds ranging from 125 Kbps to 1 Mbps and distances up to 500 meters. At the data link layer, the CAN specification defines a carrier sense multiple access mechanism for media access control that avoids collisions and sends frames reliably. The application layer supports a concise set of services to set and get attributes of objects, to invoke operations on objects, and to report notifications from objects. These services are optimized for high performance in discrete control applications (e.g., 3 byte messages convey digital I/O state changes). In addition, a set of pre-defined and extensible component objects flexibly specifies all network-visible application level device capabilities.

6.1 *Physical Layer* — The device shall comply with the SDS physical layer specifications. These include physical signaling (levels and baud rates - detailed in the CAN specification), transceivers, node isolation, media topology, cable specifications, network connectors and taps, and power considerations (load limits, system tolerances, and power supply options).

6.2 *Data Link Layer* — The device shall comply with the SDS Data Link Layer Specifications (i.e., Controller Area Network Specification: Version 2). These include the media access control mechanism and the logical link control mechanism. Frame formats, interframe spacing, and error signaling shall comply with the CAN specifications. CAN frame identifiers shall be 11 bits in length.

6.3 *Network Layer* — There is no distinct network layer. A future extension of the SDS SEMI SAN protocol will support inter network messaging.

6.4 *Transport Layer* — There is no distinct transport layer. Specific functionality of this layer is implemented in the Application Layer. Functions include: segmentation and reassembly for large message delivery.

6.5 *Session Layer* — There is no distinct session layer.

6.6 *Presentation Layer* — There is no distinct presentation layer. Data types are specified as part of the SDS object definitions.

6.7 *Application Layer* — The device shall comply with the SDS application layer specification. This includes application object to application object communication mechanisms.

6.7.1 *Object Models* — The SDS protocol provides an object-oriented specification for defining and addressing objects, including their attributes, actions, and events. The device shall comply with the object model specifications provided in the SDS Component Model Specification. In addition, the device shall comply with the object specifications defined in Section 7 of this document.

6.8 *Network Management* — The device shall comply with the SDS system and network management specifications.

7 Required Object Types

The Common Device Model specification identifies the objects that must be supported in SEMI SAN-compliant devices. These objects are specified in Table 1 of SEMI E54 and include the following:

a) Device Manager (DM)

b) Sensor/Actuator/Controller (SAC)

c) Sensor (S_i)

d) Actuator (A_i)

e) Controller (C_i).

As specified in SEMI E54, a conforming device must support the DM, the SAC, and one or more of the remaining object types.

This section specifies the implementation of these required and optional objects in the SDS object model. SDS object types are defined for the DM and SAC objects. Sensor, actuator, or controller objects may be implemented using existing standard SDS object types, or new types may be specified, depending on the specific device model needs.

7.1 *Embedded Object Type Hierarchy* — SDS classifies Embedded Objects into specific object types (classes), and these types are organized into a class inheritance hierarchy. (For more information on classification and inheritance, see SEMI E39 -- Object Services Standard). Each SDS object type specifies the details of an object's attributes, actions, and events. The Embedded Object (an instance of a type) is a network-addressable entity within a logical device. The classification identifies common characteristics and enables reuse of object definitions. The top two levels of that hierarchy are shown in Figure 4 below.

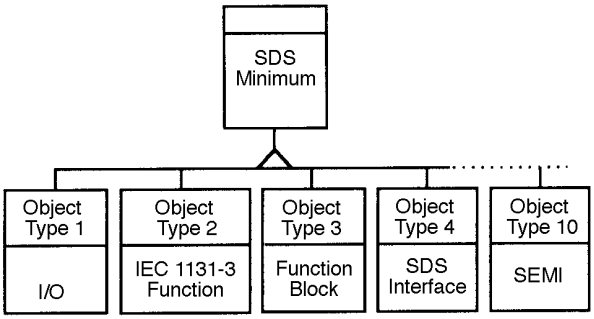


Figure 4
Top Levels of the SDS Embedded Object Type Hierarchy

The SDS object types are specified in detail in the SDS Component Modeling Specification. There are four primary Embedded Object types in the SDS type hierarchy. These include: I/O, IEC 1131-3 Function Object, Function Block, and SDS Interface. In addition, a new SEMI object type has been defined for use in developing classes necessary for SEMI SAN-compliant networks. Each of these object types is derived from a common type called SDS Minimum, which defines the minimum characteristics that are common to all SDS Embedded Objects.

All SDS object types inherit from the SDS Minimum object type, which provides the fundamental characteristics of an SDS device. The other top level SDS object types shown in Figure 4 are summarized below.

I/O Object (Type 1) — Defines the general characteristics of objects used for process I/O type functions. Subtypes of this type specify input and output objects with either analog or digital process variables. These may be used in SAN devices as the sensor or actuator objects.

IEC 1131-3 Function Object (Type 2) — Defines standard IEC 1131-3 Function Objects such as counters, timers, type conversion, bit-wise operations, character strings, etc.

Function Block Object (Type 3) — Collections of function objects that have a specific set of inputs and outputs (e.g., a predefined control algorithm). These may be used in the SAN devices as the controller objects.

SDS Interface Object (Type 4) — Models, network interfaces, gateways, etc.

SEMI Object (object Type 10) — Implement the additional object types necessary for SEMI SAN-compliant networks.

7.1.1 *SEMI Objects* — To implement SEMI-specific object types, the SDS object type hierarchy is augmented with type number 10 - SEMI object. This type, and the types derived from it, specify the new functionality necessary to implement this SEMI standard. Object type 10 is defined to include the common required attributes, services, and behaviors as described in the SAN Common Device Model standard SEMI E54. This definition is in terms of SDS Attributes, Actions, and Events.

In addition, there are derived types: types 10.1 implementing the SAC object and 10.2 implementing the Device Manager (DM) object (Figure 5). Object instances of these types are required in the Common Device Model document SEMI E54.

Each object type inheriting from the SEMI Object type 10 (e.g., 10.1 SAC object) automatically includes the features of the SEMI Object. Thus, both the SAC and the DM object have the reset, abort, and recover actions that are defined for the SEMI Object type 10. (See Section 7.2.) The Device Manager type augments the SEMI Object to provide the required characteristics over and above those specified in type 10 and SDS minimum.

As specific device models are defined, the SAC object type will be refined to be specific to that device. These object types are indicated in Figure 5 as object types SDM-1, SDM-2, etc. (examples include: Mass Flow Controller, Particle Counter, and Capacitance Manometer SAC objects). These object types have (at least) the set of attributes, actions, and events inherited from SDS Minimum, the SEMI Object type, and the SAC object type. Additionally, each SDM-n may have other, unique attributes, actions, and events which implement the desired behavior of a specific device SAC object.

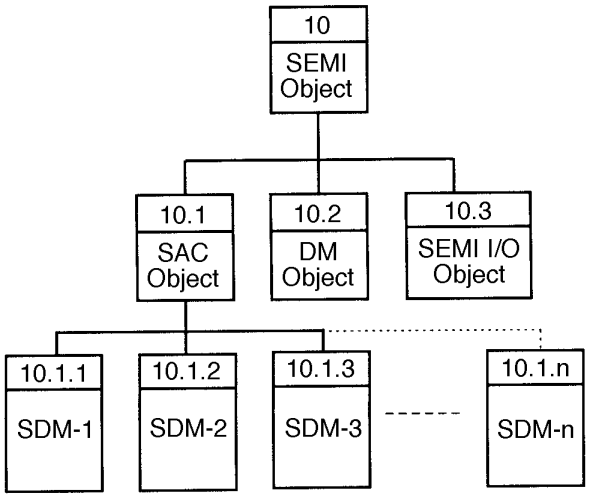


Figure 5
Second and Third Levels of the SEMI Object Type Hierarchy

The optional sensor and actuator objects described in the SEMI E54 specification may be instances of existing SDS object types (e.g., of type I/O object in Figure 4). In cases where a desired object type is not available under SDS object type 1, new I/O object types may be defined as sub-types of object type 10.3 SEMI I/O object.

7.1.2 *Example SEMI Device* — For clarification purposes, an example SEMI device is shown in Figure 6 (using Rumbaugh notation). It consists of three (or more) objects, including a DM, SAC, and binary input object. This is a complete and fully functional SEMI SAN-compliant digital input device.

In Figure 6, the Device is assigned an SDS address (e.g., 125), and each object is assigned an object instance number. The device address may be changed during installation. Each SDS message contains a device address and an object instance number.

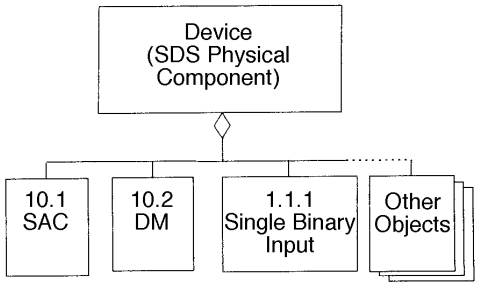


Figure 6
Example Device with Multiple Objects

7.2 *SEMI Object Type 10 Definition* — Table 2 formally specifies the SEMI Object (type 10). The three sections in this table specify SDS identifiers to code the required attributes, actions, or events. The first column is the SDS identifier number, the unique identifier used in SDS messaging. The second column is the SEMI E54-specified name associated with the ID#. The third column identifies the associated tag name (identifier) used in SEMI E54.

Table 2 Table 2 SEMI Object Common Level Model(Object Type 10)

Type 10 SEMI Object Common Structure and Behavior		
Attributes		
ID	Name	SEMI E54 tag
	None defined.	
80–144	SEMI Reserved Attribute IDs	Reserve
Actions		
ID	Name	SEMI E54 tag
80	Reset	SacS1/DmS1
81	Abort	SacS2/DmS1
82	Recover	SacS3/DmS3
83–95	SEMI Reserved Action IDs	Reserve
Events		
ID	Name	SEMI E54 tag
	None defined.	
80–95	SEMI Reserved Event IDs	Reserve

There are no attributes that are currently identified in SEMI E54 that are common to all SEMI objects. As a result, there are no attributes defined in the table. However, 64 attribute IDs have been reserved for use in other sub-types derived from type 10 in this document and in future revisions of this document.

SEMI E54 defines three service requests (reset, abort, and recover) that are common to the Device Manager and the SAC objects. These service requests are implemented as specific SDS Actions. Each service is mapped to a designated SDS action in Table 2. The actions are identified as ID# 80, 81, and 82 respectively. Additional action identifiers are reserved for use in sub-types of this type and for future use.

Not explicitly shown as Actions are the SEMI E54 Get and Set Attribute services. The equivalent behavior is provided by the SDS Read and Write Attribute services that are implicit for every attribute.

The execution of a service request may cause a change in the object's state variable. When a device receives an SDS action message indicating a service request, the object transitions to the appropriate state, after which an SDS response message is transmitted to the service

requester indicating the response to the request. For example, the SAC Object Instance Behavior State Transition Matrix in Table 4 of the SEMI E54 document defines the valid SAC state transitions.

Service Notifications (as defined in SEMI E54) are implemented with specific SDS event messages which are specified in the third part of the table. The SEMI object (type 10) has no events defined at this time. Additional event identifiers are reserved for use in sub-types of this type and for future use.

7.3 *Implementation of Sensor/Actuator/Controller Object Type (SAC)* — A single instance of a SAC object type is required in each SAN device. The actual object type used will typically be derived from the SAC object type rather than this exact type. The derived type will have added services and attributes that are device-specific (according to a Specific Device Model specification). The generic SAC type is, nevertheless, defined as a semantic type definition (sensor actuator controller object). The features of the SAC object include service requests (for Reset, Abort, Recover, etc.) that are mapped to SDS Action functions. Since these have been defined in the super type (object type 10), they are not re-defined here.

7.3.1 *SDS Object Model for Sensor/Actuator/Controller Object Type (SAC)* — The SDS object model for the SAC object type (Table 3) needs only to include the appropriate mapping for SAC-specific attributes and behavior (i.e., augmenting the SEMI Object Common Level - Table 8-1). The SAC Object Model contains no attributes, actions, or events beyond those specified in its super type (type 10).

An implementor of a specific device will normally derive a new SAC object type and add new attributes, actions, or events as necessary (i.e., 10.1.1 MFC SAC Object). Object type 10.1 is, in essence, a semantic grouping only—for a group of device-specific SAC object types implementing specific behavior.

The default object identifier that should be used for the SAC object in an SDS device is 1. The actual object identifier used may be reassigned by the implementor. The identifier may be determined on line via reading the object type attributes from all objects on the device.

Table 3 Table 3 SAC Object Model (Object Type 10.1)

Type 10.1 Sensor/Actuator/Controller Object Model		
Attributes		
ID	Name	SEMI E54 tag
	None defined.	
Actions		
ID	Name	SEMI E54 tag
	None defined.	
Events		
ID	Name	SEMI E54 tag
	None defined.	

7.4 *Implementation of Device Manager Object Type (DM)* — A single instance of this type is required on each device.

7.4.1 *SDS Object Model for Device Manager Object Type (DM)* — The SDS object model for the DM object type includes the appropriate mapping for DM-specific attributes and behavior to SDS-specific identifiers (Table 4).

Table 4 Table 4 DM Object Model (Object Type 10.2)

Type 10.2 Device Manager (DM) Object Model		
Attributes		
ID	Name	SEMI E54 tag
81	Device Type	DmA1
82	Standard Revision Level	DmA2
83	Device Manufacturer Identifier	DmA3
84	Manufacturer Model Number	DmA4
85	Software or Firmware Revision Level	DmA5
86	Hardware Revision Level	DmA6
87	Serial Number (optional)	DmA7
88	Device Configuration (optional)	DmA8
89	Device Status	DmA9
90	Reporting Mode	DmA10
91	Exception Status Timer (optional)	DmA11
92	Exception Status	DmA12
93	Exception Detail Alarm (optional)	DmA13
94	Exception Detail Warning (optional)	DmA14
Actions		
ID	Name	SEMI E54 tag
83	Execute	DmS6
84	PerformDiagnostics	DmS7
Events		
ID	Name	SEMI E54 tag
81	Publish Attribute	DmS8

The DM Object Model maps exactly to the specification for the DM in SEMI E54. Refer to that document for detailed descriptions for the attributes, actions, and events defined in Table 4. The SEMI E54 tag column indicates the identifier used in SEMI E54.

The default object identifier that should be used for the DM object in an SDS device is 2. The actual object identifier used may be reassigned by the implementer.

7.5 *Implementation of Sensor Object Type (S_i)* — Zero or more instances of this object type are permitted on each device.

Sensor object types are already defined within the SDS object hierarchy (e.g., Object Type 1 - I/O Device). Implementation of a device which is compliant with this type results in compliance with this standard.

If the existing sensor object types defined in the SDS specifications do not meet the requirements of the application, new SEMI-specific types may be defined deriving from type 10.3. If new types are required that are generic (i.e., not specific to SEMI), they should be derived from type 1 - I/O Device.

7.6 *Implementation of Actuator Object Type (A_i)* — Zero or more instances of this object type are permitted on each device.

Actuator object types are already defined within the SDS object hierarchy (e.g., Object Type 1 - I/O Device). Implementation of a device which is compliant with this type results in compliance with this standard.

If the existing actuator object types defined in the SDS specifications do not meet the requirements of the application, new SEMI-specific object types may be defined deriving from type 10.3. If new types are required that are generic (i.e., not specific to SEMI), they should be derived from type 1 - I/O Device.

7.7 *Implementation of Controller Object Type (C_i)* — Zero or more instances of this object type are permitted on each device.

Specific controller object types are defined within the SDS object hierarchy (e.g., Object type 3 - Function Block Object). Implementation of a device which is compliant with these types results in compliance with this standard.

If the controller types defined in the SDS specifications do not meet the requirements of the application, new SEMI-specific object types may be defined deriving from type 10. If new types are required that are generic (i.e., not specific to SEMI), they should be derived from type 3 - Function Block Object.

8 Protocol Compliance

A method of testing protocol compliance is required to verify conformance to this standard. The device must satisfy the SDS protocol conformance requirements as documented in the SDS specifications. The SDS partners group provides a conformance verification test procedure service to its members. When this SEMI Sensor/Actuator Network standard is incorporated into the SDS Partners specification set, this service may be used to verify compliance with the SEMI guidelines.

8.1 *Compliance Statement* — Addendum A includes a compliance statement form that should be completed by device implementors for compliance verification.

9 Specific Device Model Mappings

9.1 *Device Model for Mass Flow Controller* — This model will be specified after the MFC-specific device model (SDM) standard is complete.

9.2 *Device Model for Capacitance Manometer* — This model will be specified after the capacitance manometer SDM standard is complete.

9.3 *Device Model for Particle Counter* — This model will be specified after the particle counter SDM standdard is complete.

9.4 *Device Model for Residual Gas Analyzer* — This model will be specified after the residual gas analyzer SDM is defined.

APPENDIX 1
STATEMENT OF COMPLIANCE

NOTE: This appendix was approved as an official part of SEMI E54.5 by full letter ballot procedure.

This form is used to specify conformance options with respect to the SDS and common device model SEMI E54 specifications.

		Fill in the blank or the appropriate box. ❑ ❑			
General Device Data	Conforms to Smart Distributed System Specification:	Release _____			
	Vendor Name	_____			
	Vendor Partner ID#	_____			
	Catalog Listing	_____			
	Object Type	_____			
	Software Version	_____			
Smart Distributed System Physical Conformance Data	Network Power Consumption			_____mA @ 18 VDC	
	Connector Style	Open-Hardwired	❑	Sealed-Mini	❑
		Open-Pluggable	❑	Sealed-Micro	❑
		9 Pin	❑		
	Isolated Physical Layer	Yes		No	❑
		Opto	❑		
		Transformer	❑		
	Default Device Address	_____			
	Communication Data Rates Supported	125K bits/s	❑	500K bits/s	❑
		250k bits/s	❑	1M bits/s	❑
Smart Distributed System Logical Device & Object Data	Device Type	Master	❑		
		Slave	❑		
		Peer-to-Peer	❑		
	Number of Logical Devices	Default = 1			
	Object Class(es) on Logical Device #0	Minimum = 1			
		Object #0 _____			
		Object #1 _____			
		Object #2 _____			
		Object #3 _____			
		...			
		Object #31 _____			

If there are additional logical devices, they should be documented as above.

NOTICE: These standards do not purport to address safety issues, if any, associated with their use. It is the responsibility of the user of these standards to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use. SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E54.6-0997 (Withdrawn 0704)
STANDARD FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR LONWORKS

NOTE: This document was previously designated SEMI E61. Because this document is part of a suite of documents, its designation has been reassigned for ease of reference. Please note that the technical content of this document is unchanged from the 0697 version.

NOTICE: This document was balloted and approved for withdrawal in 2004.

1 Purpose

This standard defines a communication specification based on the LonWorks technology specification to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI-specified device models (common and device specific) in a semiconductor manufacturing tool.

This document specifies a mapping of the SEMI common device model (CDM) onto LonWorks technology using the *LonMark Interoperability Guidelines* established for LonWorks devices. The LonMark Interoperability Association may incorporate into the Interoperability Guidelines any enhancements presented in this document.

1.1 *Background and Motivation* — The LonWorks communications system provides interconnection of smart control devices such as sensors, actuators, and controllers in a fast-response time, low-cost network for industrial use. LonWorks enables multiple devices to share a single network, thereby significantly reducing the point-to-point wiring between controllers, sensors, and actuators. The LonWorks network communications standard (NCS) is based on the seven-layer LonTalk Protocol, implemented by the Neuron Chip, a physical layer transceiver, and an optional host processor. The LonTalk Protocol was developed by Echelon and may be freely licensed for implementation on any hardware platform. The SEMI NCS for LonWorks is based on the LonMark interoperability guidelines, which provide a framework for interoperable use of the LonTalk Protocol at layers 1-6, as well as at the application layer. Where the LonMark interoperability guidelines do not provide the functionality required by the SEMI CDM, the guidelines are extended with SEMI-specific requirements.

2 Scope

This document specifies a SAN communications standard, based on the LonWorks specification, that enables communication with SAN devices configured according to the SEMI SAN Common Device Model and appropriate Specific Device Model (SDM) specifications.

2.1 This document specifies the use of LonWorks technology for services that compliant intelligent devices must support in order to exchange information over this semiconductor equipment sensor/actuator network.

2.2 This document specifies the utilization of LonWorks technology to present externally visible device structure and behavior, specified in the CDM and appropriate SDMs, on a LonWorks network.

2.3 This document is used in conjunction with a SEMI standard SAN Common Device Model specification and one or more SEMI standard Specific Device Model specifications (e.g., for a mass flow controller). Together, the model documents describe the externally visible data structures and behavior of devices using LonWorks technology in a SEMI-compliant SAN system.

2.4 This standard, together with a sensor/actuator network interoperability guideline, the sensor/actuator network common device model, one or more sensor/actuator network specific device model documents, the LonTalk protocol specification, and the LonMark interoperability guidelines, specifies requirements for SEMI SAN implementations based on LonWorks technology. The general sensor/actuator network document architecture is shown in the Sensor/Actuator Network Common Device Model document in Figure 1.

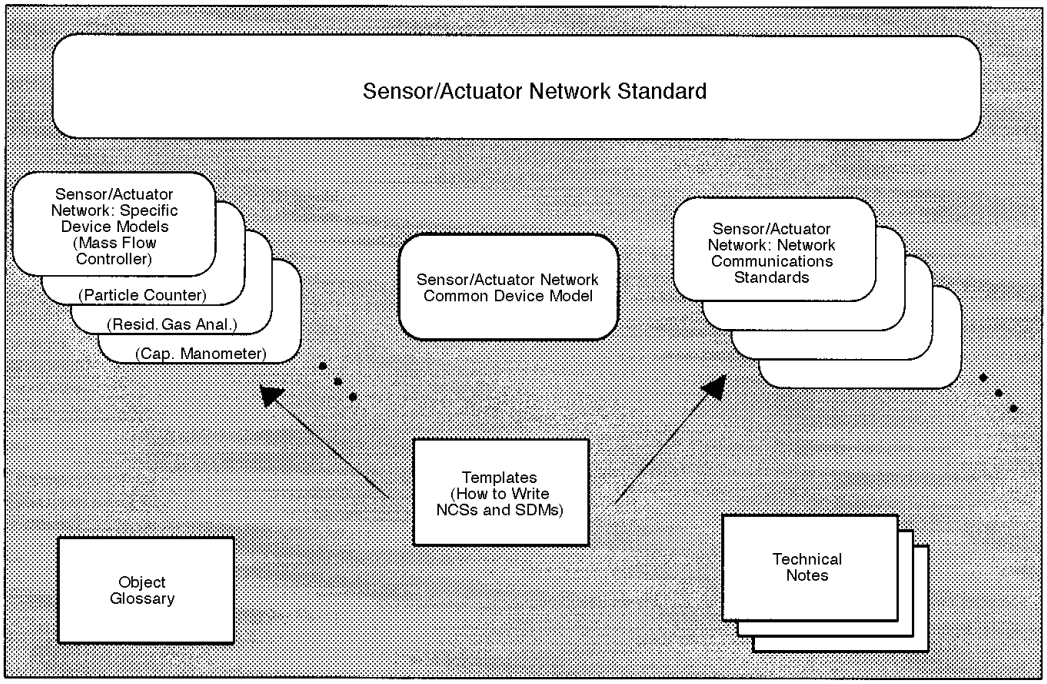


Figure 1
Sensor/Actuator Network Related Documents

2.5 *Document Structure* — The LonWorks network communications standard complies with the SEMI SAN NCS template document structure; this structure is shown in Figure 2. The standard document is composed of two main parts. The first part (Sections 1 through 8) specifies the SAN-enabling protocol as well as the presentation (i.e., mapping) of CDM object structure and behavior onto the network (referred to as the "CDM mapping"). The second part (Section 9) specifies the presentation (i.e., mapping) of SDM object structure and behavior onto the network for each SEMI-specified SDM (referred to as the "SDM mapping"). Device-type-specific items, such as connector deviations (see Section 6.1), may also be noted in Section 9.

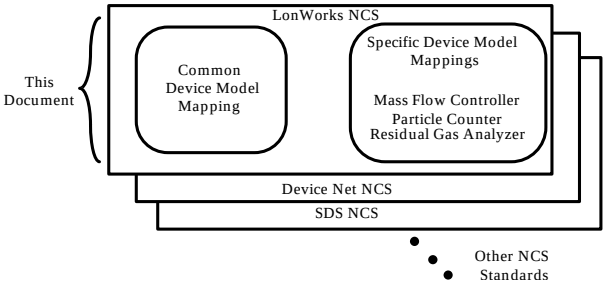


Figure 2
LonWorks Network Communications Standard Document Structure

3 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on LonWorks and is a companion document to the LonWorks protocol specification; thus, a complete specification of this standard necessarily includes the LonWorks specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by the *LonMark Interoperability Guidelines*. In order to avoid document consistency problems, information in the LonWorks technology standard that relates to this standard is not repeated in this document. This document is limited to describing enhancements or limitations of LonWorks technology and the LonMark interoperability guidelines that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the applicable LonMark interoperability conformance testing specification. Conformance testing shall include enhancements to the LonMark guidelines required by this standard.

4 Referenced Documents

4.1 *SEMI Documents*

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

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

4.2 *ISO Standard*[1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection

4.3 *Other Documents*

LonTalk Protocol Specification, Echelon Corporation, 4015 Miranda Avenue, Palo Alto, CA 94304 USA

LonMark Layers 1–6 Interoperability Guidelines, Echelon Corporation

LonMark Application Layer Interoperability Guidelines, Echelon Corporation

Neuron Chip Data Book, Echelon Corporation

Neuron C Programmer's Guide, Echelon Corporation

Standard Network Variable Type Master List and Programmer's Guide, Echelon Corporation

Standard Configuration Parameter Type Master List and Programmer's Guide, Echelon Corporation

5 Terminology

Terminology that is common to all of the documents in this SAN series may also be defined in the *Sensor Actuator Network Standard*. Terminology may be reproduced here which is defined in other SEMI documents.

5.1 *Acronyms*

5.1.1 *CDM* — Common device model

5.1.2 *DM* — DeviceManager

5.1.3 *DS* — Device status

5.1.4 *NCS* — Network communications standard

5.1.5 *NV* — Network variable

5.1.6 *OSI* — Open systems interconnect

5.1.7 *OSS* — Object services standard

5.1.8 *SAC* — Sensor, actuator, controller object

5.1.9 *SAN* — Sensor/actuator network

5.1.10 *SCPT* — Standard configuration parameter type

5.1.11 *SDM* — Specific device model

5.1.12 *SNVT* — Standard network variable type

5.2 *Device Component Definitions* — As this standard defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in the CDM document. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the LonWorks definitions. The symbol "=" indicates that the definition is used exactly as specified on the CDM specification.

In the following sections, additional clarification of some of these terms is provided in the context of the LonWorks protocol.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*	*LonWorks Equivalent*
Device	=	Device or node
Device Model	=	Functional profile
Object	=	LonMark object type
Instance	=	LonMark object instance
Attribute	=	Network variable or configuration property
Behavior	=	=
Service	=	Network variable function or application message
State Diagram	=	=
Byte	=	=
Nibble	=	=
Character String	=	String of ASCII characters or string of international characters

5.2.1 *attribute* — Attributes are either input network variables, output network variables, or configuration properties. Input and output network variables may be read and/or written by the device itself, and all attributes may be polled over the network. Additionally, input network variables and configuration properties may be updated over the network, and the receipt of such an update causes an event to be propagated to the device's application layer. This corresponds to a RW (Read and Write) attribute of the object owning the network variable. Output network variables may not be updated over the network. This corresponds to a RO (Read Only) attribute of the object owning the network variable. When the device itself updates one of its output network variables, the value of that variable may be propagated over the network to destination

1 International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211, Geneva 20, Switzerland

address(es) determined at installation time. Finally, configuration properties are attributes typically stored in non-volatile memory and preserved across device resets and power cycles.

5.2.2 *behavior* — Generic object behavior is specified by the *LonMark Application Layer Interoperability Guidelines*. Additional object-specific behavior is specified by means of functional profiles.

5.2.3 *device* — A device (or node) consists of one network transceiver which implements the physical layer of the LonTalk Protocol, one Neuron Chip with associated firmware which implements the other layers of the LonTalk Protocol, and input/output hardware implementing the physical interface of the device to external sensor and/or actuator hardware. A LonWorks device may optionally contain a host processor and associated software or firmware which implements the application layer of the LonTalk Protocol.

5.2.4 *device model* — The device model comprises several elements which fully describe the external interface of the device for an interoperable network. The interface is made of the following pieces: a Device Manager (DM) object; a Sensor/Actuator/Controller (SAC) object; LonMark objects such as sensors, actuators, and controllers; individual network variables; and configuration properties.

5.2.5 *instance* — Real devices may have zero or more instances of each of the defined LonMark objects and functional profiles. Object instances are identified by means of an instance number within the device.

5.2.6 *object* — LonMark objects are defined as a set of one or more network variable inputs and/or outputs, implemented as Standard Network Variable Types, and a set of configuration properties, implemented as Standard Configuration Property Types. LonMark objects form the basis of interoperability at the application layer. The LonMark objects describe standard formats for how information is input to, and output from, a device, and shared with other devices on the network.

5.2.7 *service* — Request services are represented by LonTalk messages delivered to the device application. Notification services are represented by LonTalk messages originated by the device application.

5.2.8 *state diagram* — In a LonWorks device, state is represented by the collection of values of local and network variables of the application program. Transitions between states are the result of external events (such as the receipt of a network variable update, or other I/O event), or internal events (such as the expiration of a timer).

5.3 *LonWorks-Specific Definitions* — In addition to the standard data type definitions for bit, nibble, byte, and character, the LonTalk Protocol defines a set of standard data representations for use as attribute values.

5.3.1 *binding* — Network variables on the same or different devices may be associated together by means of a network management service known as binding. Binding is permitted only if all the network variables in the set are of the same data type. The values of network variables that are bound together are propagated over the network by the LonTalk protocol. Table 2 shows the permitted combinations for updating and polling of network variables.

Table 2 Updating and Polling of Network Variables

Network Variable Class	*Update from Network*	*Update from Device*	*Poll from Network*	*Poll from Device*
Input	Yes	Yes	Yes	Yes[2]
Output	No	Yes[1]	Yes	No
Config'n	Yes	No	Yes	Yes[2]

NOTES:

1. When the device updates one of its own output network variables, input network variables that are bound to this output network variable receive an update from the network.

2. When the device polls one of its own input or configuration network variables, output network variables that are bound to this input or configuration network variable receive a poll from the network.

5.3.2 *configuration properties* — These are attributes of a LonMark object that are used to configure the application-specific behavior of the object, such as sensor gain and offset, linearization table, and sample rate. These attributes are typically updated when the device is installed, configured, or calibrated, and are stored in non-volatile memory.

5.3.3 *functional profile* — A functional profile is a set of one or more LonMark objects, together with semantic definitions relating the behavior of the object(s) to the network variable values. The collection of functional profiles and LonMark objects in a device corresponds to the device-specific model for that device. Each type of functional profile is identified by a type number which is allocated when the profile is standardized.

5.3.4 *network variable* — This is a network-visible data attribute of a device, with a well-defined data type. Network variables are either input variables, output variables, or configuration variables. The value of a network variable may be updated either by the device itself, or over the network by some other device. This corresponds to a SetAttribute operation. The value of a network variable may be polled over the network by

some other device, or retrieved by the device itself. This corresponds to a GetAttribute operation.

5.3.5 *object instance* — A device's external interface documentation specifies the type identifiers of the LonMark object instances contained within the device. Each instance is allocated an index number based on the order of the declaration of the instance in the device's external interface documentation. This documentation may be uploaded from the device, and completely specifies the functional profiles and LonMark objects contained within the device, as well as the network variables and configuration properties contained within each of the functional profiles.

5.3.6 *object type* — An object type is the definition of the attributes and behaviors of an abstract entity. Each type of LonMark object is identified by a type number which is allocated when the object type is standardized. A specific device type consists of instantiations of one or more of these object types. The term *LonMark object* is loosely used to refer to either a LonMark object type or to a specific instance of a LonMark object type. The attributes of a LonMark object are implemented as a collection of network variables of SNVT types and configuration properties of SCPT types.

5.3.7 *standard configuration parameter types* — These data types, also known as SCPTs, provide a data type definition and a semantic behavior for the configuration properties of LonMark objects. A list of all available SCPTs and details of their definitions is provided in the *SCPT Master List and Programmer's Guide.*

5.3.8 *standard network variable types* — These data types, also known as SNVTs, facilitate interoperability by providing a well-defined interface for communication between devices made by different manufacturers. A device may be installed in a network, and logically connected to other devices via network variables, as long as the data types match. A list of all available SNVTs and details of their definitions is provided in the *SNVT Master List and Programmer's Guide.*

6 Communication Protocol High Level Structure

The LonTalk Protocol is based on a seven-layer architecture. At each layer, there is a description of the services provided within that layer. The high level protocol architecture is shown in Figure 3.

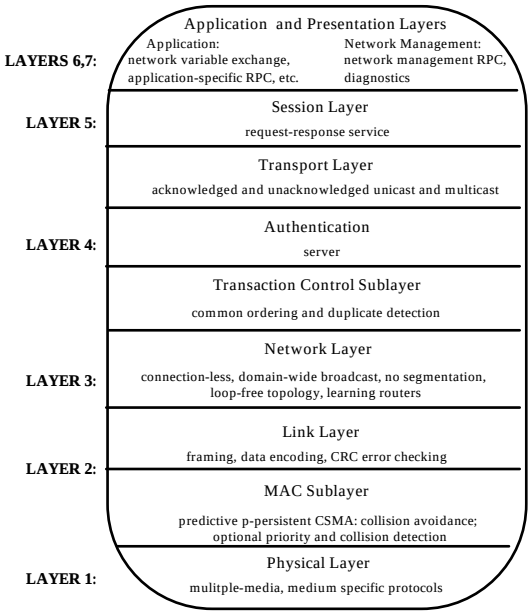


Figure 3
Layered View of the LonTalk Protocol

Note that Figure 3 represents a conceptual view of the device architecture. Implementations typically use the Neuron Chip and its associated firmware, which provide a conforming implementation of layers 2 through 6. The *LonMark Interoperability Guidelines* specify the protocol options to be used, most specifically at the physical layer (network transceivers) and at the application layer (object model).

6.1 *Physical Layer* — The device shall employ one of the LonMark-approved physical channels as specified in the *LonMark Layers 1–6 Interoperability Guidelines*. LonWorks-based SEMI SAN-compliant devices shall use, by default, a two-pin screw-terminal open pluggable connector (Weidmüller-Klippon SL2, Phoenix Combicon, or equivalent) for the network connection. The default connector specification may be overridden for specific device types if special requirements apply; any such overrides shall be noted in Section 9 of this document. This connection is polarity-insensitive. The requirements of semiconductor equipment may be met by one of the twisted pair channel specifications listed below.

6.1.1 *TP/XF-1250 Twisted Pair* — This twisted-pair channel operates at a bit rate of 1,250kbps and supports a bus topology using transformer-coupled transceivers.

6.1.2 *TP/FT-10 Twisted Pair* — This twisted-pair channel operates at a bit rate of 78kbps and supports both free topology and bus topology wiring, as well as optional link power.

The *LonMark Layers 1-6 Interoperability Guidelines* provide specific details of the characteristics of these

transceivers. This document also provides specifications of wiring types and interconnection topologies to be used for guaranteed device interoperability. Note that the LonTalk Protocol supports heterogeneous networks. Devices with dissimilar transceivers may be interconnected and communicate via routers or repeaters. Similarly, routers and repeaters may be used to extend a physical channel beyond the device count, wire length, or other physical limitations imposed by the chosen transceiver.

Multiple physical layer protocols and data encoding methods are used in the LonTalk Protocol. Differential Manchester encoding is used on twisted pair physical layers.

6.2 *Link Layer* — The device shall comply with the LonTalk protocol link layer specification. This layer includes the media access control sublayer. For a number of reasons, including simplicity and compatibility with the multicast protocol, the LonTalk protocol supports a simple connectionless service. Its functions are limited to framing, frame encoding, and error detection, with no error recovery by retransmission.

6.2.1 *Media Access Control Sublayer* — In order to deal with a variety of media in the potential absence of collision detection, the MAC (Media Access Control) sublayer employs a collision avoidance algorithm called Predictive *p*-persistent CSMA (Carrier Sense, Multiple Access).

6.3 *Network Layer* — The device shall comply with the LonTalk protocol network layer specification. This layer handles packet delivery within a single domain, with no provisions for inter-domain communication. The network service is connection-less, unacknowledged, and supports neither segmentation nor re-assembly of messages. The routing algorithms employed by the network layer to learn the topology assume a tree-like network topology; routers with configured tables may operate on topologies with physical loops, as long as the communication paths are logically tree-like. In this configuration, a packet may never appear more than once at the router on the side on which the packet originated. The unicast routing algorithm uses learning for minimal overhead and no additional routing traffic. Use of configured routing tables is supported for both unicast and multicast addresses.

6.4 *Transport Layer* — The device shall comply with the LonTalk protocol transport layer specification. The heart of the protocol hierarchy is the Transport and Session layers. A common Transaction Control sublayer handles transaction ordering and duplicate detection for both layers. The transport layer is connectionless and provides reliable message delivery to both single and multiple destinations. Authentication of the message sender's identity is provided as an optional feature. The authentication server requires only the Transaction Control sublayer to accomplish its function. The transport and session layer messages may be authenticated using all of the LonTalk addressing modes other than broadcast. The transport layer supports end-to-end acknowledged service and an unacknowledged/ repeated service.

6.5 *Session Layer* — The device shall comply with the LonTalk protocol session layer specification. This layer implements a simple Request-Response mechanism for access to remote servers. This mechanism provides a platform upon which application-specific remote procedure calls can be built. The LonTalk network management protocol, for example, is dependent on the Request-Response mechanism in the Session layer, even though it accesses the protocol via the application layer interface.

6.6 *Presentation Layer* — The device shall comply with the LonTalk protocol presentation layer specification. The Presentation layer and the Application layer taken together form the foundation of interoperability for LonTalk devices. The application layer provides all the usual services for sending and receiving messages, but it also contains the concept of network variables. The presentation layer provides information in the Application Protocol Data Unit (APDU) header for how the APDU is to be interpreted for network variable updates. This application-independent interpretation of the data allows data to be shared among devices without prior arrangement. With agreement on which network variables are to be used for sensors, actuators, etc., intelligent components from different manufacturers may work together without prior knowledge of each other's characteristics.

6.7 *Application Layer* — At the application layer, interoperability between LonWorks-based devices is facilitated through the use of LonMark objects and Standard Network Variable Types (SNVTs). LonMark objects build upon network variables and provide a concise application layer interface that incorporates semantic meaning for specific device functions. LonMark objects not only define which SNVTs to use to convey data, but also provide semantic meaning about the information being communicated. To aid in the specification of specific device models with well-defined functional behavior, collections of objects with defined relations can be aggregated and referenced as functional profiles. The Application Layer also includes the LonTalk file transfer protocol, which provides segmentation and reassembly of arbitrary length files of data. This service may be used to get and set object

attributes that exceed the network variable size limit of 31 bytes.

6.7.1 *Object Models* — The LonTalk Protocol provides an object-oriented specification for defining and addressing network variables and configuration properties, which are the representation of object attributes and events. The device shall comply with the object model specifications defined in Section 7 of this document.

6.7.2 *LonMark Object Structure* — The *LonMark Application Layer Interoperability Guidelines* define a number of object types. Each object type has a set of mandatory network variables, a set of optional network variables, a set of configuration properties (both mandatory and optional), and a manufacturer-defined section, which may be used for non-interoperable extensions to the object. This is illustrated in Figure 4. This notation is defined in the *LonMark Application Layer Interoperability Guidelines.*

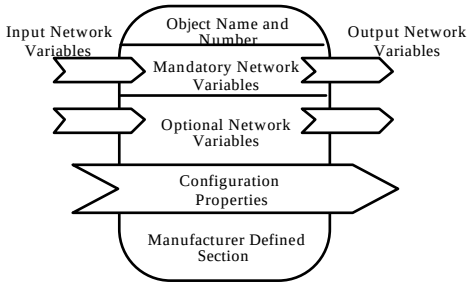


Figure 4
LonMark Object Structure[2]

The *LonMark Application Layer Interoperability Guidelines* provide for the definition of new Standard Network Variable Types, LonMark Object Types, and Functional Profiles. In the mapping of the SEMI CDM to the LonMark object structure in Section 7, extensions to the current SNVT list and Interoperability Guidelines are marked with an asterisk (*). Object type numbers are specified by the guidelines; a device may consist of one instance of a node object type, and one or more instances of LonMark object types, which are assigned sequential instance numbers starting from one.

6.8 *Network Management* — The LonTalk Protocol defines a complete network management and diagnostic protocol for LonWorks devices. This protocol is a layer above the Session layer (request/response service) and provides mechanisms for application downloading, device address assignment, distribution of destination addresses for implicit messaging, router configuration, and device-level diagnostics. The *LonMark Application Layer Interoperability Guidelines* define a device management layer for LonMark objects.

7 Required Object Types

The *LonMark Application Layer Interoperability Guidelines* describe sensor, actuator, and controller objects. A specific device may be implemented using these objects or functional profiles based on these objects. The Common Device Model specification additionally identifies two objects (namely the Device Manager (DM) and Sensor Actuator Controller (SAC) objects) that must exist in all SEMI-compliant SAN devices.

7.1 *Service Requests* — Common Device Model service requests are implemented as LonTalk foreign frame messages delivered to the application using the LonTalk request/response protocol. The transaction layer protocol ensures that response messages are correlated with the original request message. Tables 3 and 4 show the LonTalk APDU format for the Request/Indication message, and for the Response/Confirmation message respectively.

2 Diagram notation, the arrow-like symbol used in Figure 4 is defined in the LonMark Application Layer Interoperability Guidelines.

Table 3 SEMI SAN Request Message APDU Format

Field Name	*Size (bits)*	*Value*
Message Code	8	4D (hex). Indicates a SEMI SAN-compliant foreign frame message.
Object ID	16	Destination object's ID number. Based on the order of the declaration of the instance in the device's external interface documentation string.
Service Code	8	Defines the service being requested.
Request Parameters	optional	Service-specific request parameters.

Table 4 SEMI SAN Response Message APDU Format

Field Name	*Size (bits)*	*Value*
Message Code	8	Zero indicates successful execution of the requested service. Non-zero indicates failure. Values are request-specific.
Response Parameters	optional	Service-specific result parameters.

7.2 *Object Attributes* — The GetAttribute and SetAttribute service requests may also be implemented as network variable fetch, poll, and update requests addressed to the network variable corresponding to the specified attribute. This is appropriate when application-layer service responses are not required. A GetAttribute service request may be addressed directly to any network variable as a LonTalk request message, using the LonTalk protocol network management NV fetch mechanism. A GetAttribute service request may also be addressed to an output network variable as a LonTalk NV poll message, using the NV selection mechanism. A SetAttribute service request may be addressed to an input network variable as a LonTalk NV update message, using the NV selection mechanism. The confirmation of a SetAttribute (network variable update) is provided by the acknowledged service of the LonTalk protocol transport layer.

Each network variable in a LonMark object is identified by means of a self-documentation string stored in the device's memory. This string contains the object id of the object to which this variable belongs, and the sequence number of the network variable within its enclosing object. For the LonWorks NCS, this sequence number is identical to the numerical sequence number specified by the CDM tag.

Example: Suppose that the Device Manager object instance is declared as the second object instance in the device. It would, therefore, have object id 1. The Device Manager attribute Standard Revision Level has the tag DmA2. The self-documentation string for this network variable is, therefore, specified as "@1|2."

The Publish notification service is implicit when an output network variable is updated. The device propagates the value of the output network variable (equivalent to a read-only attribute) to any input network variable(s) to which it may be bound.

The LonTalk protocol only supports propagation of output network variables. In CDM terminology, this means that only read-only attributes may be published. If a specific device model requires publication of a read/write attribute, an output network variable whose value mirrors the value of the input (read/write) network variable may be introduced to the object definition.

7.3 *Sensor/Actuator/Controller Object (*)* — The SEMI CDM SAC object coordinates the functionality of Sensor, Actuator, and Controller objects in the device. A new object type is defined, which forms part of the LonMark Functional Profile for SEMI SAN-compliant devices based on LonWorks. Table 5 summarizes the services implemented by the SAC object.

Table 5 SAC Object Services

Service Name	*CDM Tag*	*Service Code*
Reset	SacS1	1
Abort	SacS2	2
Recover	SacS3	3

7.4 *Device Manager Object (*)* — The SEMI CDM Device Manager Object combines attributes of device self-documentation with an exception reporting mechanism. A new object type is, therefore, defined with the following mandatory network variables and behaviors. This object type forms part of the LonMark Functional Profile for SEMI SAN-compliant devices based on LonWorks. Table 6 summarizes the network variables that implement the attributes of the Device Manager object.

Table 6 Device Manager Object Network Variables

Name	*Storage Class*	*CDM Tag*	*Standard NV DataType*
Device Type	const	DmA1	SNVT_str_asc or SNVT_str_int
Standard Rev. Level	const	DmA2	SNVT_str_asc or SNVT_str_int
Device Mfgr. Identifier	const	DmA3	SNVT_str_asc or SNVT_str_int
Mfr. Model Number	const	DmA4	SNVT_str_asc or SNVT_str_int
S/W or F/W Rev. Level	const	DmA5	SNVT_str_asc or SNVT_str_int
Hardware Rev. Level	const	DmA6	SNVT_str_asc or SNVT_str_int
Serial Number	const	DmA7	SNVT_str_asc or SNVT_str_int
Device Config'n	const	DmA8	SNVT_str_asc or SNVT_str_int
Device Status	output	DmA9	SNVT_dev_status
Reporting Mode	config	DmA10	SCPT_rept_mode
Exception Status Rept Interval	config	DmA11	SCPT_exc_sts_t
Exception Status	output	DmA12	SNVT_exc_status
Exception Detail Alarm	output	DmA13	SNVT_exc_detail
Exception Detail Warning	output	DmA14	SNVT_exc_detail

7.4.1 *Device Manager Object Requests* — Table 7 summarizes the services implemented by the Device Manager object.

Table7 Device Manager Object Request Services

Service Name	*CDM Tag*	*Service Code*	*Request Parameters*	*Result Parameters*
Reset	DmS1	1		
Abort	DmS2	2		
Recover	DmS3	3		
GetAttribute#	DmS4	4	Attribute ID##	Attribute Value
SetAttribute###	DmS5	5	Attribute ID##, Attribute Value	
Execute	DmS6	6		
Perform Diagnostics	DmS7	7	Test ID####	

\# The GetAttribute service may also be implemented as a network variable poll.

The attribute ID is the numerical sequence number specified by the CDM tag for the attribute. This is the same as the LonMark member ID of the network variable in its owning object.

The SetAttribute service may also be implemented as a network variable update.

The Test ID parameter will be the first parameter in the Perform Diagnostics Request Parameters field.

The Publish (DmS8) notification service for the Device Manager exception status is implemented when the device updates the output network variable of type SNVT_exc_status. This causes the value of this network variable to be propagated across the network to other network variable(s) to which it may be bound. The implementation of the Device Manager object updates this output network variable according to the conditions specified by the Reporting Mode and Exception Status Reporting Interval configuration properties of the object.

7.4.2 *Device Manager Object Constant Output Network Variables* — Table 6 lists the constant output network variables of the Device Manager object. The type of each of these network variables is either SNVT_str_asc, a Standard Network Variable Type that can represent from 0 to 30 ASCII characters, or SNVT_str_int, a Standard Network Variable Type that can represent from 0 to 14 international 16-bit characters.

7.4.3 *Device Manager Object Configuration Properties* — The DM object has two configuration properties to control exception reporting as shown in Table 6. These parameters are of Standard Configuration Parameter Types (SCPTs).

The type SCPT_rept_mode(*) contains two four-bit fields specifying the reporting method for alarms and warning conditions. For example, in Neuron C, the application programming language used on the Neuron Chip, the declaration of SCPT_rept_mode is as follows:

```
typedef enum {
   REP_REQUEST                 = 0,
   REP_REQ_LATCHED             = 1,
   REP_EVT_TRIGD_ON            = 2,
   REP_EVT_TRIGD_ONOFF         = 3,
   REP_TIME_TRIGD              = 4,
   REP_EVT_ON_TIME_TRIGD       = 5,
   REP_EVT_ONOFF_TIME_TRIGD    = 6,
} rept_mode_t;
typedef struct {
   rept_mode_t alarm_rept_mode :4;
   rept_mode_t warn_rept_mode  :4;
} SCPT_rept_mode;
```

The type SCPT_exc_sts_t(*) is a 16-bit value representing times from 0.00 to 655.35 seconds, with a resolution of 0.01 seconds. This parameter is optional. The default reporting mode is REP_REQUEST.

7.4.4 *Device Manager Object Output Network Variables* — The Device Manager object has four output network variables as shown in Table 6. The data type SNVT_dev_status is an enumeration, corresponding to the device status attribute defined in Table 6 of the CDM. The values of this type are defined in Table 8.

Table 8 Device Status Enumeration Values

Value	*Enumeration Tag*
0	DS_UNKNOWN
1	DS_INIT_SELFTEST
2	DS_IDLE
3	DS_SELFTEST_EXCPT
4	DS_EXECUTING
5	DS_ABORT_1
6	DS_ABORT_2

The value DS_ABORT_1 corresponds to the *Abort from Idle* or *Executing* state, and the value DS_ABORT_2 corresponds to the *Abort from Initialized/Self Testing or Self Test Exception* state of the DM object.

The type SNVT_exc_status(*) is a union of two structures, depending on whether expanded or basic exception reporting mode is used. For example, in Neuron C, the declaration of SNVT_exc_status is as follows:

```
typedef union {
 struct {
  int  excpt_method : 1; //set to 0
  int  dev_spec     : 7;
} basic_method;
 struct {
  int  excpt_method  : 1; //set to 1
  int  warn_mfr_spec : 1;
  int  warn_dev_spec : 1;
  int  warn_dev_comn : 1;
  int  resvd         : 1;
  int  alrm_mfr_spec : 1;
  int  alrm_dev_spec : 1;
  int  alrm_dev_comn : 1;
 } expanded_method;
} SNVT_exc_status;
```

The type SNVT_exc_detail(*) is a sequence of three structures containing arrays. The LonWorks Network Communication Standard limits the size of each of these arrays to 9 bytes, so that the type fits within the network variable size limit of 31 bytes. For example, in Neuron C, the declaration of SNVT_exc_detail is as follows:

```
typedef struct {
   u_char comn_exc_size;
   int    resvd1          : 1;
   int    real_time       : 1;
   int    communic        : 1;
   int    RAM             : 1;
   int    EEPROM          : 1;
   int    EPROM           : 1;
   int    microproc       : 1;
   int    diagnostic      : 1;
/*-------------------------*/
   int    resvd2          : 1;
   int    reset           : 1;
   int    notify_mfr      : 1;
   int    maintenance     : 1;
   int    power_inputV    : 1;
   int    power_outptV    : 1;
   int    power_resvd     : 1;
   int    power_overC     : 1;
   u_char comn_exc_dtl[7];
/*-------------------------*/
   u_char dev_exc_size;
   u_char dev_exc_dtl[9];
/*-------------------------*/
   u_char mfr_exc_size;
   u_char mfr_exc_dtl[9];
} SNVT_exc_detail;
```

7.5 *Sensor, Actuator, and Controller Objects* — These objects are necessarily specific to the Specific Device Models. The *LonMark Application Layer Interoperability Guidelines* provide a framework for defining LonMark objects, together with specifications of generic sensor and actuator objects. Specific Device Models may employ these objects, and/or may define their own objects and Standard Network Variable Types for device-specific requirements. As long as the LonMark object definition guidelines are followed, these device-specific objects may be proposed to the LonMark Interoperability Association for incorporation within the LonMark guidelines.

8 Protocol Compliance

A method of testing protocol compliance is required to verify implementation conformance to the standard. By virtue of the fact that the intermediate layers of the LonTalk protocol are implemented in commercially available silicon, compliance verification is needed only at the physical and application layers. The LonMark Interoperability Association provides a compliance verification service to its members. When the SEMI Sensor/Actuator Network standard is incorporated into the LonMark guidelines, this service may be used to verify compliance with the SEMI guidelines.

8.1 *Interoperability Guidelines Checklist*

Applicant Name	
Product Name	
Standard Program ID	
Manufacturer ID	
Device Class	
Device Subclass	
Model Number	
Comm. Transceiver	
Standard Xcvr Type	
Network Connector	
Neuron Chip Clock Rate	
Oscillator Accuracy	
Network Buffer Size	
Receive Transactions	
SAC Object	
Mandatory NVs	
Optional NVs	
Configuration Properties	
Device Manager Object	
Mandatory NVs	
Optional NVs	
Configuration Properties	
Functional Profiles	
Mandatory NVs	
Optional NVs	
Configuration Properties	

9 Specific Device Type Information

9.1 This section provides for the mapping of network-visible specific device structure and behavior, specified in a SEMI standard SDM specification, to the LonWorks network. Each subsection is devoted to a single SDM specification (e.g., Section 9.1: Mass Flow Controller). Additional SDM mappings are added as subsections to this NCS specification according to SEMI guidelines and the guidelines of the SEMI SAN Interoperability standard. Device-type-specific items, such as overrides to the standard connector, may also be noted in these subsections.

SEMI E54.7-0999
STANDARD FOR SENSOR/ACTUATOR NETWORK COMMUNICATION FOR SERIPLEX

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on July 15, 1999. Initially available at www.semi.org August 1999; to be published September 1999.

1 Purpose

1.1 This standard defines a communication specification based on the Seriplex protocol to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI specified device models (common and device specific) in a semiconductor manufacturing tool.

1.2 *Background and Motivation*

1.2.1 Seriplex is a component level network which provides a simple, inexpensive, fast, and deterministic means of exchanging data among control level industrial devices (e.g., sensors and actuators) and higher level devices such as controllers. Seriplex provides:

- A solution to low-level device networking
- Access to intelligence present in low-level devices
- Networking between higher level controllers
- Master/Slave and Peer-to-Peer capabilities

1.2.2 Seriplex specifies a communication model and protocol as well as a complete Physical Layer definition.

1.2.3 This document enables communications between intelligent devices on a SEMI compliant SAN by providing a presentation mapping of common and specific device network visible structure and behavior to a Seriplex network.

2 Scope

2.1 This document specifies the protocol and services that compliant intelligent devices must support to exchange information over this semiconductor equipment sensor/actuator network.

2.2 This document specifies the utilization of the Seriplex protocol to present externally visible device structure and behavior, specified in the Common Device Model (CDM) and appropriate Specific Device Models (SDMs), on a Seriplex network.

2.3 This document is used in conjunction with a SEMI standard SAN Common Device Model specification, one or more SEMI standard Specific Device Model (SDM) specifications (e.g. for a mass flow controller) and the Seriplex Standard Specification. Together, they describe the Seriplex protocol, the externally visible data structures and behaviors of devices utilizing the Seriplex networking capability in a SEMI compliant SAN system.

2.4 This standard does not purport to address all of the safety issues associated with its use. It is the responsibility of the users of this standard to establish appropriate safety and health practices and determine the applicablity of regulatory limitations prior to use.

3 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on Seriplex and is a companion document to the Seriplex standard specification; thus a complete specification of this standard necessarily includes the Seriplex standard specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by Seriplex. In order to avoid document consistency problems, information in the Seriplex standard specification that relates to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the Seriplex standard specification that are imposed by this standard.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

4.2 *Other Documents*

Bulletin No. 8310PD9603 — Seriplex Standard Specification: August 1997, Technology Organization, Inc. Raleigh, NC, USA. [1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection[2]

NOTE 1: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

5.1 Terminology that is common to all of the documents in this SAN standard may also be defined in the Sensor/Actuator Network Standard. Terminology may be reproduced here which is defined in other SEMI documents.

5.2 *Abbreviations and Acronyms*

CDM — Common Device Model

DM — Device Manager (object)

NCS — Network Communication Standard

OSI — Open Systems Interconnect

OSS — Object Services Standard

SAC — Sensor, Actuator, Controller (Object)

SAN — Sensor/Actuator Network

SDM — Specific Device Model

STO — Seriplex Technology Organization

VDC — Volts, Direct Current

5.3 *Device Component Definitions*

5.3.1 As this standard defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in the SEMI E54.1 CDM document. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the Seriplex standard specification. Note that Column 2 contains an equal sign '=' if the definition is used exactly as specified in the CDM specification.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*	*Seriplex Equivalent*
Device	=	=
Device Model	=	=
Object	=, Class	=, Class
Instance	=	=
Attribute	=	=
Behavior	=	=
Service	=	=
State Diagram	=	=
Byte	=	=
Nibble	=	=
Character String	=	=

5.4 *Seriplex Specific Definitions*

class — a set of objects that all represent the same kind of system component. A class is a general-ization of an object. All objects in a class are identical in form and behavior, but may contain different attri-bute values.

master/slave mode — communication over a Seriplex network that provides exclusive control of data by a "master" or "host" device. All bus input data is reported exclusively to the host, and the host has exclusive control over the states of all bus output signals, with all bus I/O devices acting as 'slaves". Master/Slave mode provides the typical request/ response oriented network communications.

peer-to-peer mode — communication over a Seriplex network that provides sharing of bus input and output data directly among devices. This mode allows dedicated or broadcast data to be shared between a producing application and one or more consuming applications. Application specific I/O data moves though these devices.

seriplex — an open protocol maintained by the Seriplex Technology Organization (STO) as a standard means of interconnection for simple field devices. The Seriplex standard specification specifies OSI reference model layers 1, 2, 4 and 7 specifically the physical signaling, the media access/data link protocols, the transport capability of end-to-end transmission of data, and the application layer.

6 Communication Protocol High Level Structure

6.1 The Seriplex protocol is loosely based on a four-layer architecture. These layers constitute a collapsed form of the OSI seven layer architecture, mapping into the physical, data link, transport and application layers

1 Seriplex Technology Organization, P.O. Box 27446, Raleigh, NC 27611, www.seriplex.org

2 ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Genève 20, Switzerland

of the Reference Model. The high level protocol architecture is shown in Figure 1.

6.1.1 Note that Figure 1 represents a conceptual view of the device architecture. Conforming implementations must implement the services defined in this specification at each layer and must appear (from the network) to have implemented this architecture, however an internal modular partitioning is not required. Implementations may sacrifice modularity in order to achieve high performance.

6.1.2 The Seriplex physical layer is fully specified in the Seriplex Standard Specification. There are guideline specified for the topology of the bus cable, length of bus cable and number of bus nodes within the system. Typical configurations include: Daisy Chain, Trunk/Dropline, Tree, Loop, Star and Combinations of the above. Reference guidelines are specified for the cable length and node limits determined by a system's Clock rate and total data line capacity within a system. Bus Power Supply provides power for the Seriplex bus itself – that is, for the Seriplex bus communication circuitry within each bus device. The bus power supply normally provides a 24 VDC source for the bus. In general, the bus supply does not provide power to monitoring and control devices.

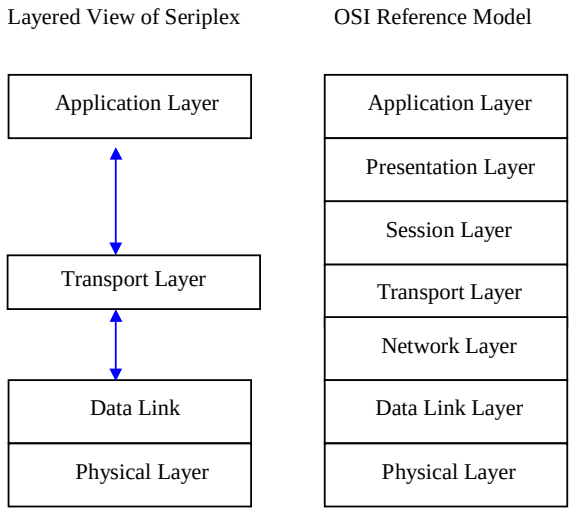


Figure 1
Layered View of Seriplex

6.1.3 At the data link layer, the Seriplex standard messaging specification defines a carrier sense multiple access mechanism for media access control that supports non-destructive collision resolution and sends frames reliably.

6.1.4 The application layer is specified in the Seriplex standard specification and provides for the definition of Seriplex applications as a collection of addressable objects. A subset of these objects may be addressed over the network (as defined by the implementation).

6.1.5 In the remainder of this section the protocol structure is described in more detail in terms of the OSI seven layer reference model, the object model environment and network management specifications.

6.2 *Physical Layer* — The device shall comply with the Seriplex physical layer specification (contained in Seriplex standard specification). This includes physical signaling (levels and data rates), transceivers, node isolation, media topology, cable specifications, network connectors and taps, and power considerations (load limits, system tolerances, and power supply options).

6.3 *Data Link Layer* — The device shall comply with the Seriplex standard specification for the Data Link Layer. This includes the media access control mechanism and the logical link control mechanism. Addressing is currently limited to 8 bits of source address and 8 bits of destination address. Bitwise arbitration is used to gain access to the network in cases where multiple nodes contend for the same message bandwidth.

6.4 *Network Layer* — There is no distinct network layer.

6.5 *Transport (Messaging) Layer* — The device shall comply with the Seriplex standard specification for the Messaging Layer. The messaging layer provides transparent transfer of data between objects in application-entities. Some of the functionality, of this layer is implemented in the Application Layer. Specific functions include: segmentation/re-assembly (fragmentation) for full message delivery.

6.6 *Session Layer* — There is no distinct session layer.

6.7 *Presentation Layer* — There is no distinct presentation layer. Object addressing and data presentation in Seriplex messages are specified as part of the Seriplex object definitions and object attribute and service communication protocol.

6.8 *Application Layer* — The device shall comply with the Seriplex application layer specification for defining and addressing objects, including their attributes and services, and enabling specified network behavior. The device shall comply with the object messaging and object model specifications included in the Seriplex standard specification. In addition the device shall comply with the object specifications defined in Section 7 of this document.

6.8.1 *Object Models* — The Seriplex protocol has been enhanced to provide an object-oriented specification for creating, defining and addressing objects explicitly, including their attributes and

services, and creating, defining and communicating object attributes in an application dependent format. The device shall comply with the object messaging and object model specifications included in the Seriplex standard specification documentation. In addition the device shall comply with the object specifications defined in Section 7 of this document.

6.8.2 *Alternate Method for the Communication Transmission of Attributes* — In order to take advantage of the Seriplex network's speed and deterministic characteristics, the Seriplex Standard Specification details a mechanism of transmitting data over the network by assigning, in advance, a sequence of serial frames to be used collectively to deliver specific sensor and/or actuator attribute data. This mechanism can be utilized as an alternative to the object messaging specification of the Seriplex Standard Specification to implement the behavior associated with the GetAttribute and SetAttribute services detailed in Section 7. This mechanism may be used to provide an efficient and optimum implementation of data transmission over the network.

6.9 *Network Management* — The device shall comply with the Seriplex network management specifications detailed in the Seriplex Standard Specification (e.g., physical layer bit rate, master/slave and peer-to-peer network management, etc.). No (additional) network management functions are specified in this document.

7 Required Object Types

7.1 At this time, the Seriplex Standard Specification does not require any specific objects to exist in a Seriplex device in order to be a compliant Seriplex device. The Seriplex Standard Specification will be extended to identify and describe objects (i.e. classes) that must exist in devices that are to be interoperable and interchangeable on a Seriplex SEMI compliant SAN network.

7.1.1 The Common Device Model specification identifies two objects (namely the Device Manager (DM) and Sensor Actuator Controller (SAC) objects) that must exist in all SEMI compliant SAN devices. The required object types for a SEMI compliant SAN device utilizing the network communication specification described herein, necessarily comprises the union of the above two requirements.

7.1.2 A list of required and optional object types is given in Table 2. Additional objects that are specified in a particular SDM are given identifiers in that SDM specification; Seriplex specific presentation information for these identifiers is given in Section 9 of this document.

7.1.3 An embodiment of a specific device type, represented as an aggregation of the object types listed in Table 2, that is compliant with both the CDM specification and the Seriplex specification, is a candidate for a SEMI SDM as well as a Seriplex device definition. Conversely, all SEMI SDM's and Seriplex device definitions specified for operation over a SEMI compliant Seriplex network must be an aggregation of the object types listed in Table 2, and be compliant with both the CDM specification and the Seriplex standard specification.

7.1.4 In the following sections the presentation to the network of object addressing, object attributes, and object services for each of the object types listed in Table 2 is described in detail. Refer to the CDM standard to determine if the object instance attribute and service is specified as required or optional. Unless otherwise noted, all attributes and services described are instance level attributes (as opposed to class level attributes). A class level attribute and service is accessed as instance number zero.

7.1.5 Note that the formats of object attributes and services are detailed in the CDM document; the presentation of object attributes and services to the Seriplex network is detailed in the tables contained in the following sub-sections and in the Seriplex standard specification.

7.2 *Device Manager (DM) Object* — The DM object is the device component responsible for managing and consolidating the device operation. Each device must support one (and only one) DM object. The DM object as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object instance attributes and services to the Seriplex network shall be as indicated in Table 3.

7.2.1 Note that the formats of DM object attributes are detailed in the CDM document; the presentation of DM object attributes to the Seriplex network is detailed in Table 3 and the Seriplex standard specification; the format of DM object services is detailed in the CDM document and the Seriplex standard specification; and the presentation of the DM object services is detailed in Table 3 and the Seriplex standard specification.

Table 2 Required and Optional Object Types

Object Name	*Seriplex Class ID/Instance ID (See Note 1)*	*CDM Tag (See Note 2)*	*Required by Seriplex (See Note 1)*	*Required by CDM (See Note 2)*	*Required by NCS*
Device Manager	1/1	DmI0	No	Yes	Yes
Sensor/ Actuator/ Controller	2/1	SacI0	No	Yes	Yes
Assembly	3/1 through i	Asm	No	No	No
Local Link	4/1 through j	Lnk	No	No	No
Sensor – AI	33/1 through k	Sai	No	No	No
Sensor – EI	34/1 through l	Sei	No	No	No
Sensor – BI	35/1 through m	Sbi	No	No	No
Actuator – AO	36/1 through n	Aao	No	No	No
Actuator – EO	37/1 through o	Aeo	No	No	No
Actuator – BO	38/1 through p	Abo	No	No	No
Controller	39/1 through q	Ca	No	No	No
Application Objects	129 through x/ 1 through r	(See Note 3)	No	No	No

NOTE 1: See Seriplex specification for further information; values are decimal; ‘i’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’ and ‘r’ represent arbitrary numbers (greater than or equal to 1) indicating that more than one instance may be supported. ‘x’ is a number greater than or equal to 129 indicating that one or more application object classes may be supported.

NOTE 2: See CDM specification for further information.

NOTE 3: Application Dependent objects as specified in SDM.

Table 3 DM Object Instance Attributes and Services

Device Manager Object (DM) *Class ID = 01, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Device Type	DmA1
02	Standard Revision Level	DmA2
03	Device Manufacturer Identifier	DmA3
04	Manufacturer Model Number	DmA4
05	Software or Firmware Revision Level	DmA5
06	Hardware Revision Level	DmA6
07	Serial Number	DmA7
08	Device Configuration	DmA8
09	Device Status	DmA9
12	Exception Status	DmA12
13	Exception Detail Alarm	DmA13
14	Exception Detail Warning	DmA14
15	Visual Indicator	DmA15
16	Alarm Enable	DmA16
17	Warning Enable	DmA17
18	Exception Detail Type	DmA18
19	Exception Detail Alarm Queue	DmA19
20	Exception Detail Warning Queue	DmA20
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	DmS1
03	Abort	DmS2
05	Recover	DmS3
07	Get Attribute	DmS4
09	Set Attribute	DmS5
11	Execute	DmS6
13	Perform Diagnostics	DmS7
15	Publish Attribute	DmS8
17	Lock	DmS9
19	Unlock	DmS10
21	Get Exception Queue	DmS11
23	Clear Exception Queue	DmS12

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. Each device must support one (and only one) SAC object. The SAC object as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object instance attributes and services to the Seriplex network shall be as indicated in Table 4.

Table 4 SAC Object Instance Attributes and Services

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Last Calibration Date	SacA1
02	Next Calibration Date	SacA2
03	Expiration Timer	SacA3
04	Expiration Warning Enable	SacA4
05	Run Hours	SacA5
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SacS1
03	Abort	SacS2
05	Recover	SacS3
07	Get Attribute	SacA4
09	Set Attribute	SacA5
25	Operate	SacA6
27	Restore Default	SacA7
29	Publish Attribute	SacA8

7.3.1 Note that the format of SAC object attributes is detailed in the CDM document; the presentation of SAC object attributes to the Seriplex network is detailed in Table 3 and the Seriplex standard specification; the format of SAC object services is detailed in the CDM document and the Seriplex standard specification; and the presentation of the SAC object services is detailed in Table 3 and the Seriplex standard specification.

7.4 *Assembly Object (Asm)* — The Assembly (Asm) object instances may be used to provide for grouping more than one attribute from one or more object instances is a device into a single data structure for communication over the Seriplex network. The presentation of object instance attributes and services shall be as indicated in Table 5.

Table 5 Assembly Object Instance Attributes and Services

ASSEMBLY Object (Asm) *Class ID == 03, Instance ID = 01 through i*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Data	AsmA1
Services		
ID	*Service Name*	*CDM Tag*
07	Get Attribute	AsmS4
09	Set Attribute	AsmS5

7.5 *Local Link Object (Lnk)* — The Local Link (Lnk) object instances may be used to 'link" an attribute of one object instance to an attribute of another object instance. The presentation of object instance attributes and services are as indicated in Table 6.

Table 6 Local Link Object Instance Attributes and Services

Local Link Object (Asm) *Class ID = 04, Instance ID = 01 through j*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Source Object Class	LnkA1
02	Source Object Instance	LnkA2
03	Source Object Attribute	LnkA3
04	Destination Object Class	LnkA4
05	Destination Object Instance	LnkA5
06	Destination Object Attribute	LnkA6
07	Commit	LnkA7
Services		
ID	*Service Name*	*CDM Tag*
--	No services defined	--

7.6 *Sensor-AI Object (Sai)* — The presentation of the Sensor Analog Input (Sensor-AI) object instance attributes and services are as indicated in Table 7.

Table 7 Sensor-AI Object Instance Attributes and Services

Sensor-AI *Class ID = 33, Instance ID = 01 through k*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SaiA1
02	Status	SaiA2
03	Alarm Enable	SaiA3
04	Warning Enable	SaiA5
16	Value	Sai16
17	ReportInhibitTimer	Sai17
18	EnableReportRate	Sai18
19	ReportRate	Sai19
64	Offset	Sai64
65	Gain	Sai65
66	DataType	Sai66
67	DataUnits	Sai67
68	SafeState	Sai68
69	EnableReportDelta	Sai69
70	ReportDelta	Sai70
71	EnableReportROC	Sai71
72	AlarmTripPointHigh	Sai72
73	AlarmTrippointLow	Sai73
74	AlarmHystersis	Sai74
75	WarningTripPointHigh	Sai75
76	WarningTripPointLow	Sai76
77	WarningHystersis	Sai77
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	SaiS1
03	Abort	SaiS2
05	Recover	SaiS3
25	Operate	SaiS4
07	GetAttribute	SaiS5
09	SetAttribute	SaiS6
27	RestoreDefault	SaiS7

7.7 *Sensor-EI Object (Sei)* — The presentation of the Sensor Enumerated Input (Sensor-EI) object instance attributes and services are as indicated in Table 8.

Table 8 Sensor-EI Object Instance Attributes and Services

Sensor-EI *Class ID = 34, Instance = 01 through l*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SeiA1
02	Status	SeiA2
03	Alarm Enable	SeiA3
04	Warning Enable	SeiA5
16	Value	Sei16
17	ReportInhibitTimer	Sei17
18	EnableReportRate	Sei18
19	ReportRate	Sei19
64	DebounceControl	Sei64
65	AlarmStatus	Sei65
66	WarningStatus	Sei66
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	SeiS1
03	Abort	SeiS2
05	Recover	SeiS3
25	Operate	SeiS4
07	GetAttribute	SeiS5
09	SetAttribute	SeiS6
27	RestoreDefault	SeiS7

7.8 *Sensor-BI Object (Sbi)* — The presentation of the Sensor Binary Input (Sensor-BI) object instance attributes and services are as indicated in Table 9.

Table 9 Sensor-BI Object Instance Attributes and Services

Sensor-BI *Class ID = 35, Instance ID = 01 through m*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SbiA1
02	Status	SbiA2
03	Alarm Enable	SbiA3
04	Warning Enable	SbiA5
16	Value	Sbi16
17	ReportInhibitTimer	Sbi17
18	EnableReportRate	Sbi18
19	ReportRate	Sbi19
64	DebounceControl	Sbi64
65	AlarmStatus	Sbi65
66	WarningStatus	Sbi66
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	SbiS1
03	Abort	SbiS2
05	Recover	SbiS3
25	Operate	SbiS4
07	GetAttribute	SbiS5
09	SetAttribute	SbiS6
27	RestoreDefault	SbiS7

7.9 *Actuator-AO Object (Aao)* — The presentation of the Actuator Analog Output (Actuator-AO) object instance attributes and services are as indicated in Table 10.

Table 10 Actuator-AO Object Instance Attributes and Services

Actuator-AO *Class ID = 36, Instance = 01 through n*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AaoA1
02	Status	AaoA2
03	Alarm Enable	AaoA3
04	Warning Enable	AaoA5
16	Setting	Aao16
17	SafeState	Aao17
18	WatchRate	Aao18
19	Watchdog	Aao19
64	Offset	Aao64
65	Gain	Aao65
66	DataType	Aao66
67	DataUnits	Aao67
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	AaoS1
03	Abort	AaoS2
05	Recover	AaoS3
25	Operate	AaoS4
07	GetAttribute	AaoS5
09	SetAttribute	AaoS6
27	RestoreDefault	AaoS7

7.10 *Actuator-EO Object (Aeo)* — The presentation of the Actuator Enumerated Output (Actuator-EO) object instance attributes and services are as indicated in Table 11.

Table 11 Actuator-EO Object Instance Attributes and Services

Actuator-EO *Class ID = 37, Instance ID = 01 through o*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AeoA1
02	Status	AeoA2
03	Alarm Enable	AeoA3
04	Warning Enable	AeoA5
16	Setting	Aeo16
17	SafeState	Aeo17
18	WatchRate	Aeo18
19	Watchdog	Aeo19
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	AeoS1
03	Abort	AeoS2
05	Recover	AeoS3
25	Operate	AeoS4
07	GetAttribute	AeoS5
09	SetAttribute	AeoS6
27	RestoreDefault	AeoS7

7.11 *Actuator-BO Object (Abo)* — The presentation of the Actuator Binary Output (Actuator-BO) object instance attributes and services are as indicated in Table 12.

Table 12 Actuator-BO Object Instance Attributes and Services

Actuator-BO *Class ID = 38, Instance ID = 01 through p*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AboA1
02	Status	AboA2
03	Alarm Enable	AboA3
04	Warning Enable	AboA5
16	Setting	Abo16
17	SafeState	Abo17
18	WatchRate	Abo18
19	Watchdog	Abo19
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	AboS1
03	Abort	AboS2
05	Recover	AboS3
25	Operate	AboS4
07	GetAttribute	AboS5
09	SetAttribute	AboS6
27	RestoreDefault	AboS7

7.12 *Controller Object (CA)* — The presentation of the Controller (CA) object instance attributes and services are as indicated in Table 13.

Table 13 Controller-CA Object Instance Attributes and Services

Controller *Class ID = 39, Instance ID = 01 through q*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	CAA1
02	Status	CAA2
03	Alarm Enable	CAA3
04	Warning Enable	CAA4
16	Setpoint	CAA16
17	ProcessVariable	CAA17
18	ControlVariable	CAA18
19	DataType	CAA19
64	DataUnits	CAA20
65	AlarmSettleTime	CAA21
66	AlarmErrorBand	CAA22
67	WarningSettleTime	CAA23
68	WarningErrorBand	CAA24
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	CAS1
03	Abort	CAS2
05	Recover	CAS3
25	Operate	CAS4
07	GetAttribute	CAS5
09	SetAttribute	CAS6
27	RestoreDefault	CAS7

8 Protocol Compliance

8.1 A method of testing protocol compliance is required to verify implementation conformance to the standard. The Seriplex Technology Organization (STO)[3] has established a mechanism for self certification of devices on a Seriplex network. This certification includes procedures and reporting mechanisms to demonstrate conformance testing and interoperability testing of devices.

9 Specific Device Model Mappings

9.1 This section provides for the mapping of network visible specific device structure and behavior, specified in a SEMI standard SDM specification, to the Seriplex network. Each subsection is devoted to a single SDM specification. Additional SDM mappings are added as sub-sections to this NCS specification according to SEMI guidelines and the guidelines of the SEMI SAN Interoperability standard.

9.2 *Specific Device Model For Mass Flow Device*

9.2.1 This section details the network mapping required to support the Specific Device Model For Mass Flow Devices. Table 14 summarizes the Mass Flow Device Object types. Subsequent tables 15 to 24 details the attributes and services associated with each Mass Flow Device object type.

Table 14 Mass Flow Device Object Types

SDM Object Identifier	*Object Name*	*Seriplex Class ID*
MFD1 (DM)	Device Manager	1
MFD2 (SAC)	Sensor Actuator Controller	2
MFD3	Sensor-AI-MF	129
MFD4	Sensor-AI-AT	130
MFD5	Assembly-MFM	131
MFD6	Sensor-AI-Aux	132
MFD7	Actuator-AO-MF	133
MFD8	Controller	39
MFD9	Local Link	4
MFD10	SISO	134
MFD11	SISO-Setpoint	135
MFD12	Assembly-MFC	136

9.3 *Sensor-AI-MF* — The presentation of the Sensor Analog Input Mass Flow (Sensor-AI-MF) object instance attributes and services are as indicated in Table 15.

Table 15 Sensor-AI-MF Object Instance Attributes and Services

Sensor-AI-MF *Class ID = 129, Instance ID = 1 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Flow Totalizer	A1
129	Flow Hours	A2
130	Zero Offset Mode	A5
131	Zeroing Status	A6
132	Autorange Status	A7
Services		
ID	*Service Name*	*SDM Tag*
129	Perform Zero Offset	S1
131	Query-Supported Gas Types	S2
133	Selected Programmed Gas Type	S3
135	Insert Gas Type	S4
137	Delete Gas Type	S5
139	Get Gas Calibration Data Value	S6
141	Set Gas Calibration Data Value	S7
143	Autorange	S8

9.4 *Sensor-AI-AT* — The presentation of the Sensor Analog Input Ambient Temperature (Sensor-AI-AT) object instance attributes and services are as indicated in Table 16.

Table 16 Sensor-AI-AT Object Instance Attributes and Services

Sensor-AI-AT *Class ID = 130, Instance = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.5 *Assembly-MFM* The presentation of the Assembly Mass Flow Meter (Assembly-MFM) object instance attributes and services are as indicated in Table 17.

3 Seriplex Technology Organization, P.O. Box 27446, Raleigh, NC 27611, www.seriplex.org

Table 17 Assembly-MFM Object Instance Attributes and Services

Assembly-MFM *Class ID = 131, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.6 *Sensor-AI-Aux*— The presentation of the Sensor Analog Input Auxiliary (Sensor-AI-Aux) object instance attributes and services are as indicated in Table 18.

Table 18 Sensor-AI-Aux Object Instance Attributes and Services

Sensor-AI-Aux *Class ID = 132, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.7 *Actuator-AO-MF* — The presentation of the Actuator Analog Output Mass Flow (Actuator-AO-MF) object instance attributes and services are as indicated in Table 19.

Table 19 Actuator-AO-MF Object Instance Attributes and Services

Actuator-AO – MF *Class ID = 133, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Valve Type	A1
129	Override	A2
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.8 *Controller* — The presentation of the Controller (Ca) object instance attributes and services are as indicated in Table 20.

Table 20 Controller Object Instance Attributes and Services

Controller *Class ID = 39, Instance ID = 01 through q*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Alarm Settling Time	CaA21
129	Warning Settling Time	CaA24
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.9 *Local Link* — The presentation of the Local Link (Lnk) object instance attributes and services are as indicated in Table 21.

Table 21 Local Link Object Instance Attributes and Services

Local Link *Class ID = 4, Instance ID = 01 through j*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.10 *SISO* — The presentation of the Single Input Single Output (SISO) object instance attributes and services are as indicated in Table 22.

Table 22 SISO Object Instance Attributes and Services

SISO *Class ID = 134, Instance = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Input	A1
129	Output	A2
130	Data Type	A3
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.11 *SISO-Setpoint* — The presentation of the Single Input Single Output Setpoint (SISO-Setpoint) object instance attributes and services are as indicated in Table 23.

Table 23 SISO-Setpoint Object Instance Attributes and Services

SISO-Setpoint *Class ID = 135, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
161	Ramp Type	A33
162	Ramp Rate	A34
163	Ratio	A35
Services		
ID	*Service Name*	*SDM Tag*
---	No additional services defined	--

9.12 *Assembly-MFC* — The presentation of the Assembly Mass Flow Controller (Assembly-MFC) object instance attributes and services are as indicated in Table 24.

Table 24 Assembly-MFC Object Instance Attributes and Services

Assembly-MFC *Class ID = 136, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

SEMI E54.8-0305
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR PROFIBUS-DP

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the European Information & Control Committee. Current edition approved by the European Regional Standards Committee on October 7, 2004. Initially available at www.semi.org February 2005; to be published March 2005. Originally published September 1999.

1 Purpose

1.1 This specification is part of the SEMI Sensor/Actuator Network (SAN) suite of standards and defines a specific communications protocol based on the PROFIBUS-DP standard. This Network Communication Standard (NCS) taken together with the SEMI Sensor/Actuator Network standard suite and the PROFIBUS standard completely and unambiguously defines an open standard providing an industry specific solution to off-the-shelf interoperability of networked devices in semiconductor manufacturing equipment.

1.2 PROFIBUS is a vendor independent, open fieldbus standard for a wide range of applications in manufacturing, process and building automation. Vendor independence and openness are guaranteed by the IEC standards for PROFIBUS, IEC 61158 part 2 to part 6 and IEC 61784-1. PROFIBUS-DP is one version of PROFIBUS which is optimized for high speed and inexpensive connectivity between automation control systems and distributed I/O at the device level.

2 Scope

2.1 This document specifies a SAN communications standard based on the PROFIBUS-DP specification that is in compliance with SEMI E54.1. As such, it specifies the protocol, services, and behavior that compliant intelligent devices must support in order to interchange information over this SAN in a method compatible with SEMI E39.

2.2 In conjunction with a SEMI standard SAN Common Device Model (CDM) specification and one or more SEMI standard Specific Device Model (SDM) specifications (e.g., for a Mass Flow Controller, In-Situ Particle Monitor Devices or Endpoint Devices), this Network Communication Standard (NCS) with the related PROFIBUS-DP standard describe the data structures, interactions, and behavior that are characteristic of the various devices on the network. This composite model forms a complete interoperability standard for communications among intelligent sensors, actuators, and controllers in semiconductor manufacturing equipment.

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 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on PROFIBUS-DP and is a companion document to the PROFIBUS-DP specification; thus, a complete specification of this standard necessarily includes the PROFIBUS-DP specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 The specifications within are strictly enhancements that provide additional capabilities over and above those currently required by PROFIBUS-DP. Included throughout this document, primarily in §6, is information paraphrased from the PROFIBUS-DP specifications such as: protocol structure, capabilities, options, and limitations. This information is provided here for reference only and is not intended to provide specification definitions. In all such areas, refer to the PROFIBUS-DP specification documents for information. This document is limited to describing enhancements or limitations to the PROFIBUS-DP specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the PROFIBUS-DP protocol conformance testing specification. Conformance testing shall also include enhancements and limitations to the PROFIBUS-DP specification required by this standard.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

SEMI E54.10 — Specification for Sensor/Actuator Network Specific Device Model for an In-Situ Particle Monitor Device

SEMI E54.11 — Specific Device Model for Endpoint Devices

4.2 *ISO Standard*[1]

ISO 7498 OSI — Basic Reference Model for Open Systems Interconnection

4.3 *IEC Standards*[2]

IEC 61158-2 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Physical Layer specification

IEC 61158-3 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Data Link Layer service definition

IEC 61158-4 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Data Link Layer protocol specification

IEC 61158-5 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems –Application Layer service definition

IEC 61158-6 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Application Layer protocol specification

IEC 61784-1 — Digital data communication for measurement and control – Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems

4.4 *PROFIBUS Standards*[3]

PROFIBUS Profile Guidelines — Part 1: Identification & Maintenance Functions

GSD Specification for PROFIBUS

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *APDU* — Application Protocol Data Unit

5.1.2 *AREP* — Application Reference Endpoint

5.1.3 *ASE* — Application Service Element

5.1.4 *CDM* — Common Device Model

5.1.5 *DMPM* — Data Link Mapping Protocol Machine

---

1 International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211, Geneva 20, Switzerland

2 International Electrotechnical Commission, 3 rue de Varembé, Case Postale 131, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.919.02.11; Fax: 41.22.919.03.00, Website: www.iec.ch

3 Profibus International, Haid-und-Neu-Str. 7,76131 Karlsruhe, Germany, Telephone: 49 721 9658 590; Fax: 49 721 9658 589, Website: www.profibus.com

5.1.6 *DP* — Decentralized Periphery

5.1.7 *DPM1* — DP-Master Class 1

5.1.8 *DPM2* — DP-Master Class 2

5.1.9 *DPV1* — DP Extensions Version 1

5.1.10 *DSAP* — Destination SAP

5.1.11 *FDL* — Fieldbus Data Link

5.1.12 *FSPM* — Fieldbus Service Protocol Machine

5.1.13 *GSD* — Generic Data Description

5.1.14 *IM* — Identification and Maintenance

5.1.15 *NCS* — Network Communication Standard

5.1.16 *OSI* — Basic Reference Model for Open Systems Interconnection (ISO 7498)

5.1.17 *PDU* — Protocol Data Unit

5.1.18 *PHY* — Physical Layer

5.1.19 *SAN* — Sensor/Actuator Network

5.1.20 *SAP* — Service Access Point

5.1.21 *SDA* — Send Data with Acknowledge

5.1.22 *SDM* — Specific Device Model

5.1.23 *SDN* — Send Data with No acknowledge

5.1.24 *SRD* — Send and Request Data with reply

5.1.25 *SSAP* — Source SAP

5.2 *Terminology Defined in Sensor/Actuator Network Common Device Model (SEMI E54.1)*

5.2.1 attribute

5.2.2 behavior

5.2.3 byte

5.2.4 common device model

5.2.5 device

5.2.6 Device Manager (DM) Object

5.2.7 device model

5.2.8 instance

5.2.9 network communication standard

5.2.10 object

5.2.11 Sensor, Actuator and Controller (SAC) Object

5.2.12 service

5.2.13 specific device model

5.2.14 state diagram

5.3 *Terminology Mapping*

5.3.1 As this standard defines the mapping of CDM data structure and behavior over a network, it makes use of many of the terms in SEMI E54.1. Table 1 provides a mapping of fundamental terminology of the CDM document into this document which uses the terminology of PROFIBUS.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*
Device	DP Slave
Object	= (with ASE as class specification)
Instance	=
Attribute	=
Behavior	=
Service	=
State Diagram	Protocol Machine, State Machine
Byte	=, Unsigned8
Nibble	Specific field coding
Character String	Visible String

5.4 *Terminology Defined in PROFIBUS*

5.4.1 *Device Data Base* — an electronic file that provides a clear and comprehensive description of the characteristics of a device type in a precisely defined format. Also called a GSD File.

5.4.2 *Device Profile* — a Device Data Base Sheet, which specifies the characteristic features of a device, and a GSD File.

5.4.3 *Data Link Mapping Protocol Machine* — a protocol layer that provides an interface to the Application Layer Protocol Machines by translating service requests and responses between the Application Layer and the Fieldbus Data Link.

5.4.4 *DP-Master Class 1 (DPM1)* — a device that polls its assigned DP-Slave devices and handles user data exchange.

5.4.5 *DP-Master Class 2 (DPM2)* — a device that interacts as a configuration or diagnostic tool; usually a programming device.

5.4.6 *DP-Slave* — a device that is configured, managed, and polled by Master devices; a DP-Slave initiates no unsolicited communications.

5.4.7 *Fieldbus Data Link* — the PROFIBUS-DP model for the OSI Layer 2 definition.

5.4.8 *GSD File* — see Device Data Base.

5.4.9 *Service Access Point* — an addressable location in a device for the directing of service requests.

5.4.10 *Send Data with No acknowledge* — a service request that sends data with no reply.

5.4.11 *Send and Request Data with reply* — a service request that sends data followed by a reply by the receiving device.

5.4.12 *Slave Diagnostics* — a method of retrieving a specifically formatted Data Structure that represents the diagnostic status of a DP-Slave.

6 Communication Protocol High Level Structure

6.1 In a typical remote I/O configuration, single master architectures are used to optimize response times. In lower speed applications, multi-master architectures are also possible. PROFIBUS-DP uses the polling principle for communication (Master-Slave method).

6.1.1 Message transfer is organized in cycles. A message cycle mainly consists of a request-frame followed by a corresponding acknowledge/response-frame of the addressed station. An exception to this is the global-control function for synchronization and coordination of several remote I/O stations.

6.1.2 A brief description of the PROFIBUS-DP protocol as it relates to the ISO 7498 OSI model follows in the sections below. For protocol efficiency, PROFIBUS-DP does not define layers 3 to 6. Layer 7 is the interface between the Application Process and the communication stack.

NOTE 1: The information contained in this section is for reference only. It in no way represents specifications for PROFIBUS-DP. See related documentation for these specifications.

6.2 *Physical Layer — Layer 1*

6.2.1 There are three options specified for the Physical Layer (PHY): Manchester Coded Interface for Bus Powered Systems, RS-485 and Optical. See the PROFIBUS-DP standard for more information about these options.

6.3 *Data Link Layer — Layer 2*

6.3.1 *Data Transfer*

6.3.1.1 The Data Link Layer or Fieldbus Data Link (FDL) provides the functions for sending and receiving data over the network. Protocol Data Units (PDU) are packaged, delivered, and checked. Acknowledgements, responses, retries, and timeouts are used to guard against Line Protocol Errors (e.g., frame, overrun, and parity) and Transmission Protocol Errors (e.g., start and end delimiters, frame check, frame length, and response times).

6.3.1.2 A PDU is restricted to 246 bytes. In addition to the PDU, a transmission frame of variable length will contain 8 bytes of overhead; one of fixed length (8 bytes) will contain 6 bytes of overhead. Various acknowledgement and response frames are also defined.

6.3.1.3 To better understand the FDL, a summary of FDL data transfer services is given by the following list:

- Send Data with Acknowledge (SDA),
- Send Data with No Acknowledge (SDN), and
- Send and Request Data with Reply (SRD).

6.4 *Application Layer — Layer 7*

6.4.1 The PROFIBUS application layer is structured in a so called service definition and protocol specification. The service definition uses an object orientated approach and specifies the services for remote access and local functions together with their objects (the ASE is a class definition of these objects). The protocol specification includes both coding and state machines.

6.4.2 *Service Definition*

6.4.2.1 Application layer services are structured to reflect the needs of flexible configurable automation devices. A Device consists of a set of modules that are placed in slots (see Figure 2). Modules are addressed uniquely by the slot number. The module view can be a hardware oriented or reflects the software structure of the DP-Slave.

6.4.3 Slot 0 is used to address the DP Slave itself. Subslot 0 represents the module and contains no IO Data. The other object classes can have instances scattered over the modules addressed by slot number. Each module can contain IO Data, Context parameter, Diagnosis information, Process Data and Alarms. Process Data is a generic class which can contain different application specific parameters accessible by read and write services. There is a set of identification and maintenance parameters defined in a PROFIBUS Guideline.

6.4.3.1 Object classes are defined for:

- IO Data for periodic reporting,
- Context for configuration,
- Diagnosis for event collection,
- Process Data for polled access, and
- Alarms for asynchronous event reporting.

6.4.3.1.1 A set of services are defined for these object classes.

6.4.3.2 IO Data is handled mainly by buffered services which allow decoupling between application and communication. Client/Server service structure (request/response) is used for Record Data, Context and Diagnosis.

6.4.4 *Protocol Specification*

6.4.4.1 A DP-Slave shall get a FDL address before using it in the target configuration.

6.4.4.2 To access a DP-Slave the DP-Master has to check the availability with a Slave-Diag service. To establish the context a SetPrm service is issued first. A check is done to ensure that the appropriate device type with the required resources is accessed. The following ChkCfg service. Another Slave Diag service has been used to check the establishment of the application relationship.

6.4.4.3 After this start up procedure the data exchange of IO data can be done and alarms can be signaled from the DP Slave to the DP Master. The DP Master can invoke services to read and write process data that contain all kind of information e.g. produced units, calibration information, and batch information.

6.4.4.4 A context is monitored by the Data Exchange Service at the DP-Slave. The DP-Master monitors the communication by the receipt of the responses to the FDL services.

6.4.5 *Data Link Mapping Protocol Machine (DMPM)*

6.4.5.1 The DMPM performs the interpretation of service requests from, and responses to, the Application Protocol Machines. Table 2 is included to demonstrate the basic structure of the PROFIBUS-DP DMPM. For speed and efficiency, the PROFIBUS-DP protocol defines the Service Access Point (SAP). These are included in the transmission protocol to direct messages within the device for fast dedicated processing. Defined are Destination SAP (DSAP) and Source SAP (SSAP).

6.4.6 *I&M Functions*

6.4.6.1 "Identification & Maintenance Functions" (I&M functions) define general parameters and protocols. The main purpose of the I&M functions is to support the end user during various scenarios of a device's life cycle be it configuration, commissioning, parameterization, diagnosis, repair, firmware update, asset management, audit trailing, and alike. Well-defined uniform parameters and rules should enable the manufacturers to offer devices that behave in a uniform manner These profile guidelines take into account requirements from FDA (Food & Drug Administration) and others. The basic information offered can be characterized as "Type Plate" or "Boiler Plate". It consists of Manufacturer ID, Order ID, Serial Number, Hardware Revision, Software Revision, Revision Counter, and Profile ID.

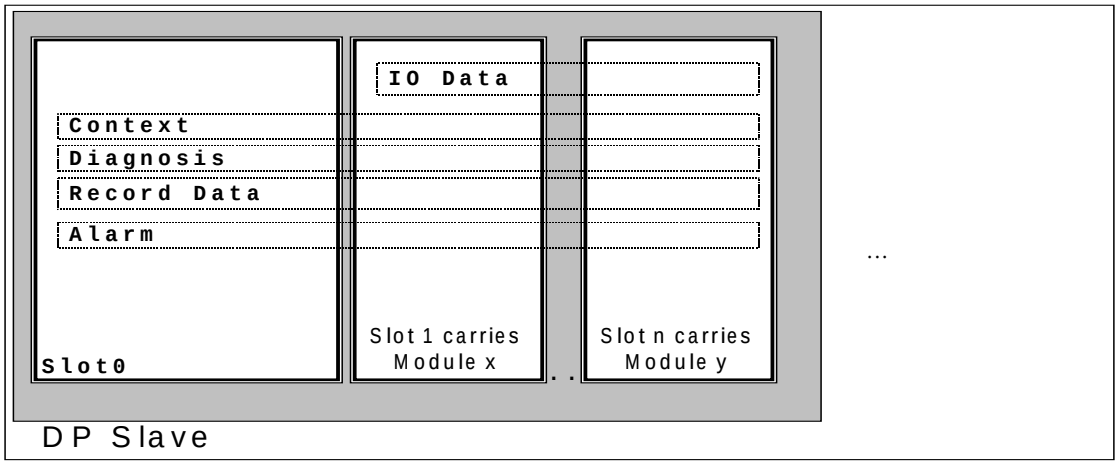


Figure 1
Device Model

Table 2 DMPM Primitive Functions

DMPM Function	*Description*	*SSAP*	*DSAP*	*FDL SRV*
Master-Slave				
Data_Exchange	Exchanges I/O Data	NIL	NIL	SRD
Check_Cfg	Sends Configuration to DP-Slave for verification.	62	62	SRD
Set_Prm	Sends Parametric Data to DP-Slave.	62	61	SRD
Slave_Diag	Retrieves the Diagnostic Data Structure from a DP-Slave.	62	60	SRD
Get_Cfg	Retrieves the Configuration Data Structure from a DP-Slave.	62	59	SRD
Global_Control	Controls the Operational and Synchronization of DP-Slaves.	62	58	SDN
DPM2-Slave				
RD_Outp	Retrieves the Status of the Outputs of the DP-Slave.	62	57	SRD
RD_Inp	Retrieves the Values of the Inputs of the DP-Slave.	62	56	SRD
Set_Slave_Add	Sets the Node Address of a DP-Slave.	62	55	SRD
Master-Master				
	Various Services for Master-Master Communications	54	54	SRD/SDN
Master-Slave — Extended Communications				
Read/Write	Acyclic Read/Write of DP-Slave Data	51	51	SRD
DPM2-Slave — Connection Configuration				
Initiate/Read/Write ...	Services for Control and Management of the MS2 Connection	50	x..49	SRD

6.4.7 *Service Access Point (SAP)*

6.4.7.1 The Service Access Point provides standard access addressing for messages. The FDL message frame includes fields for Source and Destination SAP. By directing a message to a particular Destination SAP, its context is immediately known. This provides a fast and interoperable environment for device messaging.

6.4.8 *Device Profile/Device Data Base*

6.4.8.1 PROFIBUS devices have different performance characteristics. Features differ in regard to available functionality (i.e., number of I/O signals and diagnostic messages) or possible bus parameters such as baud rate and time monitoring. These parameters vary individually for each device type and vendor. These parameters are usually documented in the technical manual. To achieve simple plug-and-play configuration of PROFIBUS, the characteristic features are specified in an electronic data sheet called a Device Data Base file or GSD file.

6.4.8.2 The GSD Files provide a clear and comprehensive description of the characteristics of a device type in a precisely defined format. These are prepared individually by the vendor for each type of device and made available to the user in the form of a Device Data Base Sheet and a GSD File. The device data base file is divided into three parts: General Specifications, Master Related Specifications, and Slave Related Specifications.

6.4.8.3 These GSD Files are maintained and managed by the PROFIBUS Trade Organization.

6.5 *Network Management*

6.5.1 The PROFIBUS-DP system is managed through several phases of operation. A Master device must have knowledge of the Device Profile for each of the Slave devices it will connect. The Device Data Base Files serve this purpose. Upon initialization, a Master will control a DP-Slave through three operational modes: Parameterization, Configuration, and I/O Data Exchange. In any operational mode, a Master may interrogate a DP-Slave for its Diagnostic information.

6.5.2 In the sections that follow, these operation modes are mapped to related SAN CDM behavior states.

7 Required Object Types

7.1 This section describes a general mapping of the SEMI SAN Object Model to the PROFIBUS-DP environment. Component definitions are clarified and the mapping of Attributes, Services, and Behaviors are specified.

7.2 *Object Model*

7.2.1 The Object Model defined in the CDM is represented in the PROFIBUS NCS. Specifically, the DM and SAC objects are mapped.

7.2.2 The CDM Objects are mapped to slots. These slots have a unique address space within the device. The Application Objects associated with the SDM standards are mapped in PROFIBUS-DP Device Data Base documents as defined above in the Device Profile/Device Data Base section. §9 specifies the mapping of SDM Objects in PROFIBUS-DP.

7.3 *Component Mapping Summary*

7.3.1 Figure 2 provides a summary of the components of the CDM object model as they relate to the components of PROFIBUS-DP.

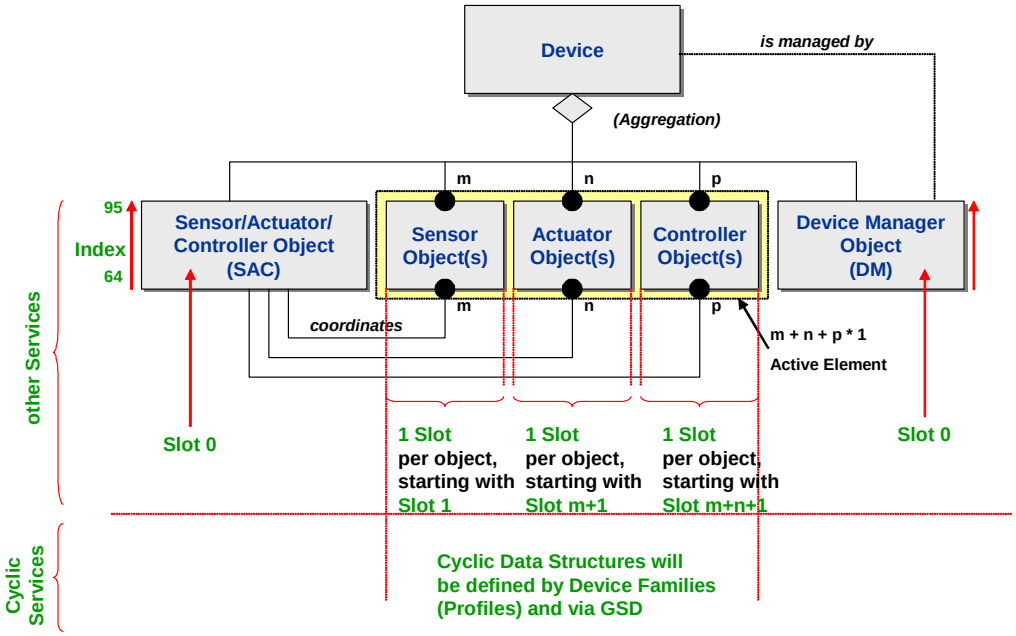


Figure 2
Component Mapping Summary

7.4 *Objects*

7.4.1 The required objects of the CDM are identified here. Additional objects that are contained in the SDM are given identifiers in the Device Profile. §9 specifies additional mapping information.

7.4.2 Table 3 lists the Object Identifiers specified for use in protocol messages. Slot numbers are the way for addressing Objects within PROFIBUS-DP.

Table 3 Object Identifiers

Slot ID	*Index*	*Object*
0	IM0-17 96	DM Object DM Control
0	64 65–95	SAC Control SAC Object
1–n	32–254	Application Objects as specified in §§ 7 and 9.

7.4.3 Assembly Objects and Local Link Objects which are objects embedded in the SAC Object is not explicitly mapped.

7.5 *Attributes*

7.5.1 *General*

7.5.1.1 All attributes are accessible via Get_Attribute and Set_Attribute services defined in the sections below. Additionally, attributes are accessible via different PROFIBUS-DP defined methods which are mapped in this document based on attribute type.

7.5.2 *DM Attributes*

7.5.2.1 All attributes are communicated with Read and Write Service and mapped to IM Blocks. Status attributes are modeled additionally as Channel Diagnosis. See Table 18 for detailed mapping.

7.5.2.2 Table 4 shows the DM status attribute mapping to Diagnosis.

Table 4 Channel Diagnosis mapping of DM Status

Parameter	*Value*
Identifier_Number	= SlotID of the Object
Channel_Number	0 = Global
Channel Type	0 = unspecified
IO Type	0 = unspecified
Error Type	9 = Error with Bit0 - 6 set in Common Exception Detail 1 21 = Error Calibration set in Common Exception Detail Attribute 1 17 = Error with Bit set in Common Exception Detail 2

7.5.3 *SAC Attributes*

7.5.3.1 The attributes of SAC are mapped to Process data objects of PROFIBUS.

7.5.3.2 The Slot number used is 0. The index is the numeric value of the SAC Attribute identifier with an offset of 64 (see Table 5).

Table 5 SAC Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail*
SacA1	65	Last Calibration Date	
SacA2	66	Next Calibration Date	
SacA3	67	Expiration Timer	
SacA4	68	Expiration Warning Enable	
SacA5	69	Run Hours	

7.5.4 *Active Element Attributes*

7.5.4.1 The attributes of Active Elements are mapped to Process data objects of PROFIBUS. These attributes are contained in any Sensor, Actuator and Controller Object.

7.5.4.2 The Slot number used is the instance number in the device. The index is the numeric value of the Active Element Attribute identifier with an offset of 64 (see Table 6).

Table 6 Active Element Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
nA1	65	Name	
nA2	66	Status	
nA3	67	Alarm Enable	
nA4	68	Warning Enable	

7.5.5 *Sensor Attributes*

7.5.5.1 The attributes of Sensors are mapped to Process data objects of PROFIBUS. These attributes are contained in any Sensor-AI, Sensor-EI and Sensor-BI Object.

7.5.5.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Attribute identifier with an offset of 64 (see Table 7).

Table 7 Sensor Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
nA16	80	Value	Can be mapped as Input Data Object
nA17	81	Report Inhibit Time	
nA18	82	Enable Report Rate	
nA19	83	Report Rate	

7.5.6 *Actuator Attributes*

7.5.6.1 The attributes of Actuators are mapped to Process data objects of PROFIBUS. These attributes are contained in any Actuator-AO, Actuator-EO and Actuator-BO Object.

7.5.6.2 The Slot number used is the instance number in the device. The index is the numeric value of the Actuator Attribute identifier with an offset of 64 (see Table 8).

Table 8 Actuator Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
nA16	80	Setting	Can be mapped as Output Data Object
nA17	81	Safe State	
nA18	82	Watch Rate	
nA19	83	Watch Dog	The value of this variable reflects the WD_Factor parameter of PROFIBUS

7.5.7 *Controller Attributes*

7.5.7.1 The attributes of Controllers are mapped to Process data objects of PROFIBUS.

7.5.7.2 The Slot number used is the instance number in the device. The index is the numeric value of the Controller Attribute identifier with an offset of 64 (see Table 9).

Table 9 Controller Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
CA16	80	Setpoint	
CA17	81	Process Variable	
CA18	82	Control Variable	
CA19	83	Data Type	
CA20	84	Data Unit	
CA21	85	Alarm Settle Time	
CA22	86	Alarm Error Band	
CA24	88	Warning Settle Time	
CA25	89	Warning Error Band	

7.5.8 *Sensor Analog Input Attributes*

7.5.8.1 The attributes of Sensor Analog Inputs are mapped to Process data objects of PROFIBUS.

7.5.8.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Analog Input Attribute identifier with an offset of 64 (see Table 10).

Table 10 Sensor Analog Input Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
SaiA64	128	Offset	
SaiA65	129	Gain	
SaiA66	130	Data Type	
SaiA67	131	Data Units	
SaiA68	132	Safe State	
SaiA69	133	Enable Report Delta	
SaiA70	134	Report Delta	
SaiA71	135	Enable Report ROC	
SaiA72	136	Report ROC	
SaiA73	137	Alarm Trip Point High	
SaiA74	138	Alarm Trip Point Low	
SaiA75	139	Alarm Hysteresis	
SaiA76	140	Warning Trip Point High	
SaiA77	141	Warning Trip Point Low	
SaiA78	142	Warning Hysteresis	

7.5.9 *Sensor Binary Input Attributes*

7.5.9.1 The attributes of Sensor Binary Inputs are mapped to Process data objects of PROFIBUS.

7.5.9.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Binary Input Attribute identifier with an offset of 64 (see Table 11).

Table 11 Sensor Binary Input Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
SbiA64	128	Debounce Control	
SbiA65	129	Alarm State	
SbiA66	130	Warning State	

7.5.10 *Sensor Enumerated Input Attributes*

7.5.10.1 The attributes of Sensor Enumerated Inputs are mapped to Process data objects of PROFIBUS.

7.5.10.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Enumerated Input Attribute identifier with an offset of 64 (see Table 12).

Table 12 Sensor Enumerated Input Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
SeiA64	128	Debounce Control	
SeiA65	129	Alarm State	
SeiA66	130	Warning State	

7.5.11 *Actuator Analog Output Attributes*

7.5.11.1 The attributes of Actuator Analog Outputs are mapped to Process data objects of PROFIBUS.

7.5.11.2 The Slot number used is the instance number in the device. The index is the numeric value of the Actuator Analog Output Attribute identifier with an offset of 64 (see Table 13).

Table 13 Actuator Analog Output Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
AaoA64	128	Offset	
AaoA65	129	Gain	
AaoA66	130	Data Type	
AaoA67	131	Data Units	

7.5.12 *Sensor Binary Input Threshold Attributes*

7.5.12.1 The attributes of Sensor Binary Input Thresholds are mapped to Process data objects of PROFIBUS.

7.5.12.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Binary Input Threshold Attribute identifier with an offset of 80 (see Table 14). The different mapping has to be done because there is an Overlap to the Sensor Binary Input Attributes enumeration part.

Table 14 Sensor Binary Input Threshold Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFIBUS Index*	*Attribute*	*Detail/Alternative Access Method*
SbithA64	144	Reading Valid	
SbithA65	145	State	
SbithA66	146	Status	

7.6 *Services*

7.6.1 *DM Services*

7.6.1.1 Get_Attribute and Set_Attribute are mapped to Read and Write services to the IM Objects.

7.6.1.2 The other services are mapped to a write at an UINT Object addressed with Index 96 as described in Table 3 and Table 15.

7.6.1.3 A read to Index 96 will return the last command executed successfully.

Table 15 DM Service Mapping

SEMI CDM Service ID	*Service*	*Write Value to Index 96*
DmS4	Get Attribute	Mapped as Read from Attribute Index
DmS5	Set Attribute	Mapped as Write to Attribute Index
DmS1	Reset	0
DmS2	Abort	1
DmS3	Recover	2
DmS6	Execute	3
DmS7	Perform Diagnostics	4
DmS8 – DmS12		Optional Services are not mapped in PROFIBUS

7.6.2 *SAC Services*

7.6.2.1 Get_Attribute and Set_Attribute are mapped to Read and Write services with Slot and Index as described in ¶¶ 7.2 and 7.4.

7.6.2.2 The other services are mapped to a write at an UINT Object addressed with Index 64 as described in Table 3 and Table 16.

7.6.2.3 A read to Index 64 will return the last command executed successfully (Exception: Restore Default is not mirrored).

Table 16 SAC Service Mapping

SEMI CDM Service ID	*Service*	*Write Value to Index 64*
SacS4	Get Attribute	Mapped as Read from Attribute Index
SacS5	Set Attribute	Mapped as Write to Attribute Index
SacS1	Reset	0
SacS2	Abort	1
SacS3	Recover	2
SacS6	Operate	3
SacS7	Restore Default	4
SacS8	Publish Attribute	Optional Service not mapped in PROFIBUS

7.6.3 *Active Element Services*

7.6.3.1 Get_Attribute and Set_Attribute are mapped to Read and Write services with Slot and Index as described in ¶¶ 7.2 and 7.4.

7.6.3.2 The other services are mapped to a write at an UINT Object addressed with Index 64 as described in Table 2 and Table 17.

7.6.3.3 A read to Index 64 will return the last command executed successfully (Exception: Restore Default is not mirrored).

Table 17 Active Element service mapping

SEMI CDM Service ID	*Service*	*Write Value to Index 64*
nS4	Get Attribute	Mapped as Read from Attribute Index
nS5	Set Attribute	Mapped as Write to Attribute Index
nS1	Reset	0
nS2	Abort	1
nS3	Recover	2
nS6	Operate	3
nS7	Restore Default	4

7.7 *PROFIBUS-DP Device Data Base*

7.7.1 The specification of the Device Data Base (or GSD file) for a given SDM is beyond the scope of this document. The PROFIBUS Trade Organization is responsible for the management of these files.

7.8 *PROFIBUS-DP I/O Data Exchange*

7.8.1 Input/Output attributes of the Application objects are communicated using the I/O Data Exchange method of PROFIBUS-DP. This method is described in the PROFIBUS-DP standard. A list of which attributes are accessible with this method is included in the PROFIBUS Device Profile for a given device type

Table 18 DM Object Attribute Identifiers

SEMI CDM	*PROFIBUS*	*Attribute*	*Detail /Alternative Access Method*
Attribute ID	*Attribute ID*		
DmA1	IM0, DEVICE_ID	Device Type	8 Octets significant
DmA2	IM0, REVISION_COUNTER	Standard Revision Level	Shall be an UINT type
DmA3	IM0, MANUFACTURER_ID	Device Manufacturer Identifier	Shall be an UINT type, the number is resolved via file from *www.profibus.com* to supply with web pages and phone numbers
DmA4	IM0, ORDER_ID	Manufacturer Model Number	
DmA5	IM0, SOFTWARE_REVISION	Software or Firmware Revision Level	
DmA6	IM0, HARDWARE_REVISION	Hardware Revision Level	
DmA7	IM0, SERIAL_NUMBER	Serial Number	16 Octets maximum
DmA8	IM3	Device Configuration	
DmA9	IM17, octet 10-11	Device Status	Additional as Status Diagnosis
DmA10	IM16, octet 10-11	Reporting Mode	
DmA11	IM16, octet 12-13	Exception Status Report Interval	
DmA12	IM17, octet 12-13	Exception Status	Additional as Status Diagnosis
DmA13	Slot 0 Index 0xe00b	Exception Detail Alarm	Additional as Channel Diagnosis
DmA14	Slot 0 Index 0xe00b	Exception Detail Warning	Additional as Channel Diagnosis
DmA15	Optional Attribute not mapped	Visual Indicator	N/A
DmA16	Context	Alarm Enable	Put in the SetPrm User Data
DmA17	Context	Warning Enable	Put in the SetPrm User Data
DmA18	Optional Attribute not mapped	Exception Detail Type	N/A
DmA19	Optional Attribute not mapped	Exception Detail Alarm Queue	N/A
DmA20	Optional Attribute not mapped	Exception Detail Warning Queue	N/A
DmA21	Optional Attribute not mapped	Date and Time	N/A
DmA22	Optional Attribute not mapped	Date and Time Type	N/A
DmA23–DmA31	Reseved	Reserved	N/A

8 Protocol Compliance

8.1 PROFIBUS International has established a qualified certification system, with test laboratories in Europe, Asia and USA, which includes conformance testing and interoperability testing (address list can be found at http://www.PROFIBUS.com/support.html). Certified products are listed with their certificate number in the PROFIBUS Electronic Product Guide.

8.2 GSD files of all PROFIBUS-DP devices that are tested for their conformity to the PROFIBUS standard are available in the GSD library on the World Wide Web Server of the PROFIBUS User Organization at http://www.PROFIBUS.com.

9 Specific Device Model Mappings

9.1 Every type of device must have an identifier number. Vendors must apply for an identifier number from the PROFIBUS User Organization for every Device Type. In order to receive a valid identifier number, a Device Profile must be submitted in the form of a GSD File and Device Data Base Sheet.

9.2 The Device Profile must specify the identifiers for Objects, Attributes and Services for CDM and SDM components, including data formats and bit mappings for specified parameters, as represented in this document.

9.3 The following sections specify mappings for Sensor Actuator Network Specific Device Models.

9.4 *Mass Flow Device*

9.4.1 Reference SEMI E54.3 for a complete specification of the SDM for Mass Flow Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.4.2 *Objects*

9.4.2.1 Consistent with SEMI E54.3 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.4.2.2 Notice that references for the Local Link Objects are not included; the existence of these objects are implied by behavior and not explicitly included. Therefore, these objects are not accessible from the network.

9.4.2.3 Table 19 shows the mapping of the SDM.

9.4.2.4 Objects specified in SEMI E54.3 are listed under the heading NCS Module reference. Every instance is allocated to a slot number. The configuration can be modeled in the GSD by using the Module reference as key for the module description.

Table 19 Mass Flow Device Object Identifiers

SDM Object Name	*SDM Object ID*	*NCS Module reference*
Sensor-AI-MF	MFD3	103
Sensor-AI-AT	MFD4	104
Assembly-MFM	MFD5	105
Sensor-AI-Aux	MFD6	106
Actuator-AO-MF	MFD7	107
Controller	MFD8	108
Local Link	MFD9	109
SISO	MFD10	110
SISO-Setpoint	MFD11	111
Assembly-MFC	MFD12	112

9.4.3 *Attributes*

9.4.3.1 The mapping of Attribute Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.4.3.2 Tables 20–24 show the mapping of the specific SDM attributes.

Table 20 Sensor AI-MF Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
A1	160	Flow Totalizer
A2	161	Flow Hours
A5	164	Zero Offset Mode
A6	165	Zeroing Status
A7	166	Autorange Status

Table 21 Actuator AO-MF Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
A1	160	Valve Type
A2	161	Override

Table 22 Controller Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
A1	160	Valve Type
A2	161	Override

Table 23 SISO Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
A1	65	Input
A2	66	Output
A3	67	Data Type

Table 24 SISO Setpoint Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
A33	97	Ramp Type
A34	98	Ramp Rate
A35	99	Ratio

9.4.4 *Services*

9.4.4.1 The mapping of Service Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.4.4.2 The additional Services of Sensor-AI-MF are mapped as described in Table 25.

Table 25 Sensor-AI-MF additional Service mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
S1	Perform Zero Offset	11
S2	Query Supported Gas Type	12
S3	Select Programmed Gas Type	13
S4	Insert Gas Type	14
S5	Delete Gas Type	15
S6	Get Gas Calibration Data Value	16
S7	Set Gas Calibration Data Value	17
S8	Autorange	18

9.5 *In-Situ Particle Monitor Device*

9.5.1 Reference SEMI E54.10 for a complete specification of the SDM for In-Situ Particle Monitor Device (ISPM).

9.5.2 *Objects*

9.5.2.1 Consistent with SEMI E54.10 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.5.2.2 Notice that references for the Local Link Objects are not included; the existence of these objects are implied by behavior and not explicitly included. Therefore, these objects are not accessible from the network.

9.5.2.3 Table 26 shows the mapping of the SDM.

9.5.2.4 Objects specified in SEMI E54.10 are listed under the heading NCS Module reference. Every instance is allocated to a slot number. The configuration can be modeled in the GSD by using the Module reference as key for the module description.

Table 26 In-Situ Particle Monitor Device Object Identifiers

SDM Object Name	*SDM Object ID*	*NCS Module reference*
Sensor-AI-LCS	ISPMD03	203
Sensor-AI-SLS	ISPMD04	204
Sensor-AI- MNS	ISPMD05	205
Sensor-AI-Counter	ISPMD16	216
Assembly-ISPM#1	ISPMD17	217
Assembly-ISPM#2	ISPMD18	218
Assembly-ISPM#3	ISPMD19	219
Assembly-ISPM#4	ISPMD20	220
Assembly-ISPM#5	ISPMD21	221
Assembly-ISPM#6	ISPMD22	222
Assembly-ISPM#7	ISPMD23	223
Assembly-ISPM#8	ISPMD24	224
Assembly-ISPM#9	ISPMD25	225

9.5.3 *Attributes*

9.5.3.1 The mapping of Attribute Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.5.3.2 Tables 27–32 show the mapping of the specific SDM attributes.

Table 27 DM Object Additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
DmA33	96	Gain
DmA34	97	Filter Bandwidth
DmA35	98	Tool State
DmA36	99	Laser Status
DmA37	100	Flow Path
DmA38	101	Volume
DmA39	102	Volume Units
DmA40	103	Leak Status
DmA41	104	Time Stamp

Table 28 SAC Object Additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
SacA65	129	Number of Bins
SacA66	130	Count Mode
SacA67	131	Duration

Table 29 Sensor AI-LCS Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
LcsA1	160	Reading Valid
LcsA2	161	Full Scale
LcsA3	162	Alarm Settling Time
LcsA4	163	Warning Settling Time

Table 30 Sensor AI-SLS Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
SlsA1	160	Reading Valid
SlsA2	161	Full Scale
SlsA3	162	Alarm Settling Time
SlsA4	163	Warning Settling Time

Table 31 Sensor AI-MNS Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
MnsA1	160	Reading Valid
MnsA2	161	Full Scale
MnsA3	162	Alarm Settling Time
MnsA4	163	Warning Settling Time

Table 32 Sensor AI-Counter Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
CounterA1	160	Reading Valid
CounterA2	161	Full Scale
CounterA3	162	Alarm Settling Time
CounterA4	163	Warning Settling Time
CounterA5	164	Upper Size
CounterA6	165	Lower Size

9.5.4 *Services*

9.5.4.1 The mapping of Service Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.5.4.2 The additional Services of DM are mapped as described in Table 33.

Table 33 DM additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 96*
DmS1	Laser On	11
DmS2	Laser Off	12

9.5.4.3 The additional Services of SAC are mapped as described in Table 34.

Table 34 SAC Additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
SacS1	Clear Counts	11

9.6 *Endpoint Device*

9.6.1 Reference SEMI E54.11 for a complete specification of the SDM for Endpoint Devices.

9.6.2 Date-and-Time data structure shall be mapped to PROFIBUS data type Time Of Day. An Offset of 4383 has to be added to Time Of Day day value to obtain the number of days since 1/1/72 (Time Of Day base is 1/1/84).

9.6.3 *Objects*

9.6.3.1 Consistent with SEMI E54.11 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.6.3.2 Notice that references for the Local Link Objects are not included; the existence of these objects are implied by behavior and not explicitly included. Therefore, these objects are not accessible from the network.

9.6.3.3 Table 35 shows the mapping of the SDM.

9.6.3.4 Objects specified in SEMI E54.11 are listed under the heading NCS Module reference. Every instance is allocated to a slot number. The configuration can be modeled in the GSD by using the Module reference as key for the module description.

9.6.3.5 The Sensor-BI-TH-EP can occur more than once. This shall be modeled in GSD in that way, that this Module ID can be placed in several slots.

Table 35 Endpoint Device Object Identifiers

SDM Object Name	*SDM Object ID*	*NCS Module reference*
Sensor-BI-TH-EP	EPD3	203
Assembly-EPD#1	EPD4	204
Assembly- EPD#2	EPD5	205
Assembly- EPD#3	EPD6	206
Assembly- EPD#4	EPD7	207

9.6.4 *Attributes*

9.6.4.1 The mapping of Attribute Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.6.4.2 Tables 36 and 37 show the mapping of the specific SDM attributes.

Table 36 DM Object additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
SacA65	129	Number of Endpoint Objects

Table 37 Sensor-BI-TH-EP Object additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFIBUS Index*	*Attribute*
EpA1	160	Minimum Time
EpA2	161	Maximum Time
EpA3	162	Target Time
EpA4	163	Elapsed Time
EpA5	164	Time Stamp
EpA6	165	Recipe Identifier
EpA7	166	Step Identifier

9.6.5 *Services*

9.6.5.1 The mapping of Service Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.6.5.2 The additional Services of SAC are mapped as described in Table 38.

Table 38 SAC additional Service mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
SacS33	Reset Endpoint	33
SacS34	Download Recipe	34
SacS35	Upload Recipe	35
SacS36	Calibrate	36

9.6.5.3 The additional Services of Sensor BI-EP are mapped as described in Table 39.

Table 39 Sensor BI-EP additional Service mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
EpS1	Endpoint On	11
EpS2	Endpoint Off	12
EpS3	Endpoint Start	13
EpS4	Endpoint Suspend	14
EpS5	Endpoint Resume	15

SEMI E54.9-0303
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMUNICATION FOR MODBUS/TCP OVER TCP/IP

This specification was technically approved by the Global Information and Control Committee and is the direct responsibility of the North American Information and Control Committee. Current edition approved by the North American Regional Standards Committee on October 25, 2002. Initially available at www.semi.org December 2002; to be published March 2003. Originally published February 2000; previously published October 2000.

1 Purpose

1.1 This standard defines a communication specification based on the Modbus/TCP protocol over a Transmission Control Protocol/Internet Protocol (TCP/IP) network to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI specified device models (common and device specific) in a semiconductor manufacturing tool.

1.2 *Background and Motivation*

1.2.1 Modbus/TCP over TCP/IP is a component level network which provides a simple, inexpensive, and fast means of exchanging data among control level industrial devices (e.g., sensors and actuators) and higher level devices such as controllers. Modbus/TCP over TCP/IP provides:

- A solution to low-level device networking,
- Access to intelligence present in low-level devices,
- Networking between higher level controllers, and
- Master/Slave and Peer-to-Peer communication capabilities.

1.2.2 Modbus/TCP specifies a communication model and protocol. The Physical, Data Link, and Network Layer definitions are defined by the network in which the Modbus/TCP protocol is embedded such as TCP/IP Ethernet.

1.2.3 This document enables communications between intelligent devices on a SEMI compliant SAN by providing a presentation mapping of common and specific device network visible structure and behavior to Modbus/TCP over a TCP/IP network.

2 Scope

2.1 This document specifies the protocol and services that compliant intelligent devices must support to exchange information over this semiconductor equipment sensor/actuator network.

2.2 This document specifies the utilization of the Modbus/TCP protocol to present externally visible device structure and behavior, specified in the Common Device Model (CDM) and appropriate Specific Device Models (SDMs), for the Modbus/TCP protocol over a TCP/IP network.

2.3 This document is used in conjunction with a SEMI standard SAN Common Device Model specification, one or more SEMI standard Specific Device Model (SDM) specifications (e.g., for a mass flow device), the Modicon Modbus Protocol Reference Guide and the Open Modbus/TCP Specification. Together, they describe the Modbus/TCP protocol, the externally visible data structures and behaviors of devices utilizing the Modbus/TCP networking capability in a SEMI compliant SAN system.

3 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on Modbus/TCP over a TCP/IP network and is a companion document to the Open Modbus/TCP Specification; thus a complete specification of this standard necessarily includes the Modbus/TCP and Modbus protocol specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by Modbus/TCP. In order to avoid document consistency problems, information in the Modbus/TCP standard specifications that relate to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the Modbus/TCP standard specifications that are imposed by 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.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services (OSS)

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

4.2 *Other Standards*

IEEE 802.3 — Telecommunication and Information Exchange between System Local and Metropolitan Networks Specific Requirement Part 3: Carrier Sense Multiple Access CSMA/CD Method and Physical Layer Specification, 1998[1]

IP RFC 791 — Reference for data transmission[2]

ISO 7498 — Basic Reference Model for Open Systems Interconnection[3]

PI-MBUS-300 Rev. E — Modicon Modbus Protocol Reference Guide, March 1993[4]

Specification — Open Modbus/TCP Specification Version 1.0 March 29, 1999[4]

NOTICE: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

5.1 Terminology that is common to all of the documents in this SAN standard may also be defined in the Sensor/Actuator Network standard. Terminology may be reproduced here which is defined in other SEMI standards.

5.2 *Abbreviations and Acronyms*

5.2.1 *CDM* — Common Device Model

5.2.2 *DM* — Device Manager (object)

5.2.3 *IP* — Internet Protocol

5.2.4 *NCS* — Network Communication Standard

5.2.5 *OSI* — Open Systems Interconnect

5.2.6 *OSS* — Object Services Standard

5.2.7 *SAC* — Sensor, Actuator, Controller (Object)

5.2.8 *SAN* — Sensor/Actuator Network

5.2.9 *SDM* — Specific Device Model

5.2.10 *TCP* — Transmission Control Protocol

5.3 *Device Component Definitions*

5.3.1 As this standard defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in SEMI E54.1. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the Modbus/TCP standard specifications.

NOTE 1: Column 2 contains an equal sign "=" if the definition is used exactly as specified in the CDM specification.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*	*Modbus/TCP Equivalent*
Device	=	=
Device Model	=	=
Object	=, Class, Instance	=, Class, Instance
Instance	=	=
Attribute	=	=
Behavior	=	=
Service	=	=
State Diagram	=	=
Byte	=	=
Nibble	=	=
Character String	=	=

5.4 *Modbus/TCP Specific Definitions*

5.4.1 *class* — a set of objects that represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values as well as additional attributes and services. Refer to SEMI E39 for further definition.

5.4.2 *master/slave* — communication over a Modbus network, which is referred to as "client/server", that provides exclusive control of data by a "master" or "host" device acting as a "client". All network input data is reported exclusively to the host when requested by the host, and the host has exclusive control over the states of all network output signals of all nodes acting as it's "slaves" or "servers". Master/Slave communication provides the typical request/response oriented network communications.

5.4.3 *modbus/TCP* — an open protocol established at The University of Michigan's Electronics Manufacturing Laboratory as a standard means of intercon-

1 IEEE, 3 Park Avenue, 17th Floor, New York, NY 10016 U.S.A. tel: 212 419 7900 fax: 212 752 4929. http://www.ieee.org/

2 http://src.doc.ic.ac.uk/computing/internet/rfc/rfc791.txt

3 ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Genève 20, Switzerland

4 http://www.modicon.com/openmbus/standards/standards.htm

nection for simple field devices. The Modbus/TCP over TCP/IP standard specifies OSI reference model layers 1, 2, 3, 4 and 7 specifically the physical signaling, the media access/data link protocols, internetworking capability, the transport capability of end-to-end transmission of data, and the application layer.

5.4.4 *peer-to-peer* — on Modbus/TCP over TCP/IP networks, messages formatted according to the Modbus/TCP protocol are embedded into the TCP packet structure that is used on the TCP/IP network. The Modbus protocol over TCP/IP supports the asynchronous or unsolicited bi-directional transmission of data between nodes. This type of communication is referred to as peer-to-peer.

6 Communication Protocol High Level Structure

6.1 The Modbus/TCP protocol over TCP/IP is loosely based on a five-layer architecture. These layers constitute a collapsed form of the OSI seven layer architecture, mapping into the physical, data link, network, transport, and application layers of the Reference Model. This section has been formatted to be aligned with the Basic Reference Model for OSI. The high-level protocol architecture is shown in Figure 1.

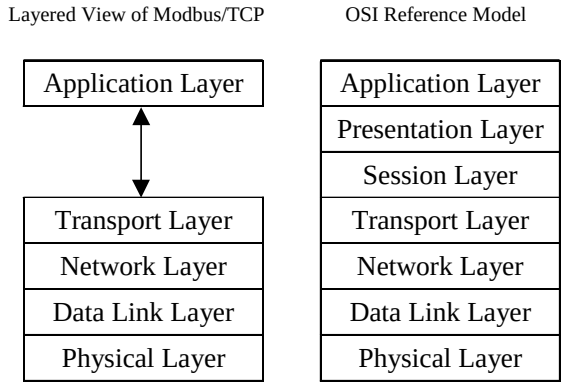


Figure 1
Layered View of Modbus/TCP Over TCP/IP

NOTE 2: Figure 1 represents a conceptual view of the communication device architecture. Conforming implementations must implement the services defined in this specification at each layer and must appear (from the network) to have implemented this architecture, however, an internal modular partitioning is not required. Implementations may sacrifice modularity in order to achieve high performance.

6.1.1 The application layer is specified in the Modicon Modbus Protocol Reference Guide and provides for the definition of Modbus applications as a collection of addressable objects. A subset of these objects may be addressed over the network (as defined by the implementation).

6.1.2 In the remainder of this section the protocol structure is described in more detail in terms of the OSI seven layer reference model, the object model environment and network management specifications.

6.2 *Physical Layer* — The device shall comply with a physical layer specification identified in the Modbus/TCP specification. The recommended and default Modbus/TCP physical layer is IEEE 802.3. If an accepted physical layer specification other than IEEE 802.3 is being used, this must be clearly specified in product literature. Physical layer specification includes physical signaling (levels and data rates), transceivers, node isolation, media topology, cable specifications, network connectors and taps, and power considerations (load limits, system tolerances, and power supply options).

6.3 *Data Link Layer* — The device shall comply with a data link layer specification of the Modbus/TCP specification. The recommended and default Modbus/TCP data link layer is IEEE 802.3. If an accepted data link layer specification other than IEEE 802.3 is being used, this must be clearly specified in product literature. Data link layer specification includes the media access control mechanism and the logical link control mechanism.

6.4 *Network Layer* — The device shall comply with a network layer specification of the Modbus/TCP specification. This specification is the IP or Internet Protocol as defined in the Modbus/TCP standard specification. The network layer specification includes network routing and internetworking.

6.5 *Transport (Messaging) Layer* — The device shall comply with the Modbus/TCP standard specification for the Transport Layer. This specification is the Transmission Control Protocol as defined in the Modbus/TCP standard specification. The transport layer provides transparent transfer of data between objects in application-entities. Some of the functionality of this layer is implemented in the Application Layer. Specific functions include: object data segmentation/reassembly (fragmentation) for full message delivery.

6.6 *Session Layer* — There is no distinct session layer.

6.7 *Presentation Layer* — There is no distinct presentation layer. Object addressing and data presentation in Modbus messages are specified as part of the Modbus object definitions and object attribute and service communication protocol. Data byte transmission ordering is defined in IP RFC 791, Appendix B.

6.8 *Application Layer* — The device shall comply with the Modbus/TCP application layer specification for defining and addressing objects, including their attributes and services, and enabling specified network behavior. The device shall comply with the object messaging and object model specifications included in the Modbus/TCP standard specifications. In addition, the device shall comply with the object specifications defined in Section 7 of this document.

6.8.1 *Object Models* — The Modbus/TCP protocol has been enhanced to provide an object-oriented specification for creating, defining, and addressing objects explicitly, including their attributes and services, and creating, defining, and communicating object attributes in an application dependent format. The device shall comply with the object messaging and object model specifications included in the Modbus/TCP standard specification documentation. In addition, the device shall comply with the object specifications defined in Section 7 of this document.

6.9 *Network Management* — The device shall comply with the Modbus/TCP and TCP/IP network management specifications detailed in the Modbus/TCP Standard Specifications (e.g., physical layer bit rate, master/slave and peer-to-peer network management, etc.). No (additional) network management functions are specified in this document.

7 Required and Optional Object Types

7.1 At this time, the Modbus/TCP standard specifications do not require any specific objects to exist in a Modbus/TCP device in order to be a compliant Modbus/TCP device. The Modbus/TCP standard specifications will be extended to identify and describe objects (i.e., classes) that must exist in devices that are to be interoperable and interchangeable on a Modbus/TCP SEMI compliant SAN network.

7.1.1 The Common Device Model (CDM) specification identifies two objects (namely the Device Manager (DM) and Sensor Actuator Controller (SAC) objects) that must exist in all SEMI compliant SAN devices.

7.1.2 The required object types for a SEMI compliant SAN device utilizing the network communication specification described herein, necessarily comprises, at minimum, the union of the Modbus/TCP object type requirements and the CDM specification requirements.

7.1.3 A list of required and optional object types is given in Table 2. Additional objects that are specified in a particular SDM are given identifiers in that SDM specification. Modbus specific presentation information for these identifiers is given in Section 9 of this document.

Table 2 Required and Optional Object Types

Object Name	*Modbus Class ID/Instance ID (See Note 1)*	*CDM Tag (See Note 2)*	*Required by Modbus (See Note 1)*	*Required by CDM (See Note 2)*	*Required by NCS*
Device Manager	1/1	DmI0	No	Yes	Yes
Sensor/Actuator/ Controller	2/1	SacI0	No	Yes	Yes
Assembly	3/1 through i	Asm	No	No	No
Local Link	4/1 through j	Lnk	No	No	No
Sensor – AI	33/1 through k	Sai	No	No	No
Sensor – EI	34/1 through l	Sei	No	No	No
Sensor – BI	35/1 through m	Sbi	No	No	No
Actuator – AO	36/1 through n	Aao	No	No	No
Actuator – EO	37/1 through o	Aeo	No	No	No
Actuator – BO	38/1 through p	Abo	No	No	No
Controller	39/1 through q	Ca	No	No	No
Application Objects	129 through x/ 1 through r	(See Note 3)	No	No	No

NOTE 1: See Modbus specification for further information; values are decimal: “i”, “j”, “k”, “l”, “m”, “n”, “o”, “p”, “q” and “r” represent arbitrary numbers (greater than or equal to 1) indicating that more than one instance may be supported. “x” is a number greater than or equal to 129 indicating that one or more application object classes may be supported.
NOTE 2: See CDM specification for further information.
NOTE 3: Application Dependent objects as specified in SDM.

7.1.4 An embodiment of a specific device type represented as an aggregation of the object types listed in Table 2 that is compliant with both the CDM specification and the Modbus/TCP specification, is a candidate for a SEMI SDM as well as a Modbus/TCP device definition. Conversely, all SEMI SDM's and Modbus/TCP device definitions specified for operation over a SEMI compliant Modbus/TCP network must be an aggregation of the object types listed in Table 2, and be compliant with both the CDM specification and the Modbus/TCP standard specifications.

7.1.5 In the following sections the presentation to the network of object addressing, object attributes, and object services for each of the object types listed in Table 2 is described in detail. Refer to the CDM standard to determine if the object instance attribute and service is specified as required or optional. Unless otherwise noted, all attributes and services described are instance level attributes (as opposed to class level attributes). A class level attribute and service is accessed as instance number zero.

NOTE 3: The formats of object attributes and services are detailed in the CDM document; the presentation of object attributes and services to the Modbus/TCP over a TCP/IP network is detailed in the tables contained in the following sub-sections and in the Modbus/TCP standard specifications.

7.2 *Device Manager (DM) Object* — The DM object instance is the device component responsible for managing and consolidating the device operation. Each device must support one (and only one) DM object. The DM object as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object instance attributes and services to the Modbus/TCP network shall be as indicated in Table 3. Note that all service ID values identified refer to the ID of the request or notification component of that service. Corresponding reply components to request/reply services shall have a service ID value equal to the request component ID plus one.

Table 3 DM Object Instance Attributes and Services

Device Manager Object (DM) *Class ID = 01, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Device Type	DmA1
02	Standard Revision Level	DmA2
03	Device Manufacturer Identifier	DmA3
04	Manufacturer Model Number	DmA4
05	Software or Firmware Revision Level	DmA5
06	Hardware Revision Level	DmA6
07	Serial Number	DmA7
08	Device Configuration	DmA8
09	Device Status	DmA9
12	Exception Status	DmA12
13	Exception Detail Alarm	DmA13
14	Exception Detail Warning	DmA14
15	Visual Indicator	DmA15
16	Alarm Enable	DmA16
17	Warning Enable	DmA17
18	Exception Detail Type	DmA18
19	Exception Detail Alarm Queue	DmA19
20	Exception Detail Warning Queue	DmA20
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	DmS1
03	Abort	DmS2
05	Recover	DmS3
07	Get Attribute	DmS4
09	Set Attribute	DmS5
11	Execute	DmS6
13	Perform Diagnostics	DmS7
15	Publish Attribute	DmS8
17	Lock	DmS9
19	Unlock	DmS10
21	Get Exception Queue	DmS11
23	Clear Exception Queue	DmS12

7.3 *Sensor, Actuator, Controller (SAC) Object* — The SAC object instance is the device component responsible for coordinating the interaction of the device with the sensory/actuation/control environment. Each device must support one (and only one) SAC object instance. The SAC object instance as well as its common required and optional attributes, services, and behavior are described in the CDM standard. The presentation of object instance attributes and services to the Modbus/TCP network shall be as indicated in Table 4.

Table 4 SAC Object Instance Attributes and Services

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Last Calibration Date	SacA1
02	Next Calibration Date	SacA2
03	Expiration Timer	SacA3
04	Expiration Warning Enable	SacA4

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
05	Run Hours	SacA5
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SacS1
03	Abort	SacS2
05	Recover	SacS3
07	Get Attribute	SacA4
09	Set Attribute	SacA5
25	Operate	SacA6
27	Restore Default	SacA7
29	Publish Attribute	SacA8

7.4 *Assembly Object (Asm)* — The Assembly (Asm) object instances may be used to provide for grouping more than one attribute from one or more object instances in a device into a single data structure for communication over the Modbus/TCP network. The presentation of object instance attributes and services shall be as indicated in Table 5.

Table 5 Assembly Object Instance Attributes and Services

ASSEMBLY Object (Asm) *Class ID == 03, Instance ID = 01 through i*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Data	AsmA1
Services		
ID	*Service Name*	*CDM Tag*
07	Get Attribute	AsmS4
09	Set Attribute	AsmS5

7.5 *Local Link Object (Lnk)* — The Local Link (Lnk) object instances may be used to "link" an attribute of one object instance to an attribute of another object instance. Refer to the CDM for further explanation and use of this object. The presentation of object instance attributes and services are as indicated in Table 6.

Table 6 Local Link Object Instance Attributes and Services

Local Link Object (Asm) *Class ID = 04, Instance ID = 01 through j*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Source Object Class	LnkA1
02	Source Object Instance	LnkA2
03	Source Object Attribute	LnkA3
04	Destination Object Class	LnkA4
05	Destination Object Instance	LnkA5
06	Destination Object Attribute	LnkA6
07	Commit	LnkA7
Services		
ID	*Service Name*	*CDM Tag*
--	No services defined.	--

7.6 *Sensor-AI Object (Sai)* — The presentation of the Sensor Analog Input (Sensor-AI) object instance attributes and services are as indicated in Table 7.

Table 7 Sensor-AI Object Instance Attributes and Services

Sensor-AI *Class ID = 33, Instance ID = 01 through k*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SaiA1
02	Status	SaiA2
03	Alarm Enable	SaiA3
04	Warning Enable	SaiA5
16	Value	Sai16
17	ReportInhibitTimer	Sai17
18	EnableReportRate	Sai18
19	ReportRate	Sai19
64	Offset	Sai64
65	Gain	Sai65
66	DataType	Sai66
67	DataUnits	Sai67
68	SafeState	Sai68
69	EnableReportDelta	Sai69
70	ReportDelta	Sai70
71	EnableReportROC	Sai71
72	ReportROC	Sai72
73	AlarmTripPointHigh	Sai73
74	AlarmTripPointLow	Sai74
75	AlarmHystersis	Sai75
76	WarningTripPointHigh	Sai76
77	WarningTripPointLow	Sai77
78	WarningHystersis	Sai78
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	SaiS1
03	Abort	SaiS2
05	Recover	SaiS3
25	Operate	SaiS4
07	GetAttribute	SaiS5

Sensor-AI *Class ID = 33, Instance ID = 01 through k*		
09	SetAttribute	SaiS6
27	RestoreDefault	SaiS7

7.7 *Sensor-EI Object (Sei)* — The presentation of the Sensor Enumerated Input (Sensor-EI) object instance attributes and services are as indicated in Table 8.

Table 8 Sensor-EI Object Instance Attributes and Services

Sensor-EI *Class ID = 34, Instance = 01 through l*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SeiA1
02	Status	SeiA2
03	Alarm Enable	SeiA3
04	Warning Enable	SeiA5
16	Value	Sei16
17	ReportInhibitTimer	Sei17
18	EnableReportRate	Sei18
19	ReportRate	Sei19
64	DebounceControl	Sei64
65	AlarmStatus	Sei65
66	WarningStatus	Sei66
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	SeiS1
03	Abort	SeiS2
05	Recover	SeiS3
25	Operate	SeiS4
07	GetAttribute	SeiS5
09	SetAttribute	SeiS6
27	RestoreDefault	SeiS7

7.8 *Sensor-BI Object (Sbi)* — The presentation of the Sensor Binary Input (Sensor-BI) object instance attributes and services are as indicated in Table 9.

Table 9 Sensor-BI Object Instance Attributes and Services

Sensor-BI *Class ID = 35, Instance ID = 01 through m*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SbiA1
02	Status	SbiA2
03	Alarm Enable	SbiA3
04	Warning Enable	SbiA5
16	Value	Sbi16
17	ReportInhibitTimer	Sbi17
18	EnableReportRate	Sbi18
19	ReportRate	Sbi19
64	DebounceControl	Sbi64
65	AlarmStatus	Sbi65
66	WarningStatus	Sbi66
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	SbiS1
03	Abort	SbiS2
05	Recover	SbiS3
25	Operate	SbiS4
07	GetAttribute	SbiS5
09	SetAttribute	SbiS6
27	RestoreDefault	SbiS7

7.9 *Actuator-AO Object (Aao)* — The presentation of the Actuator Analog Output (Actuator-AO) object instance attributes and services are as indicated in Table 10.

Table 10 Actuator-AO Object Instance Attributes and Services

Actuator-AO *Class ID = 36, Instance = 01 through n*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AaoA1
02	Status	AaoA2
03	Alarm Enable	AaoA3
04	Warning Enable	AaoA5
16	Setting	Aao16
17	SafeState	Aao17
18	WatchRate	Aao18
19	Watchdog	Aao19
64	Offset	Aao64
65	Gain	Aao65
66	DataType	Aao66
67	DataUnits	Aao67
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	AaoS1
03	Abort	AaoS2
05	Recover	AaoS3
25	Operate	AaoS4
07	GetAttribute	AaoS5

Actuator-AO *Class ID = 36, Instance = 01 through n*		
09	SetAttribute	AaoS6
27	RestoreDefault	AaoS7

7.10 *Actuator-EO Object (Aeo)* — The presentation of the Actuator Enumerated Output (Actuator-EO) object instance attributes and services are as indicated in Table 11.

Table 11 Actuator-EO Object Instance Attributes and Services

Actuator-EO *Class ID = 37, Instance ID = 01 through o*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AeoA1
02	Status	AeoA2
03	Alarm Enable	AeoA3
04	Warning Enable	AeoA5
16	Setting	Aeo16
17	SafeState	Aeo17
18	WatchRate	Aeo18
19	Watchdog	Aeo19
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	AeoS1
03	Abort	AeoS2
05	Recover	AeoS3
25	Operate	AeoS4
07	GetAttribute	AeoS5
09	SetAttribute	AeoS6
27	RestoreDefault	AeoS7

7.11 *Actuator-BO Object (Abo)* — The presentation of the Actuator Binary Output (Actuator-BO) object instance attributes and services are as indicated in Table 12.

Table 12 Actuator-BO Object Instance Attributes and Services

Actuator-BO *Class ID = 38, Instance ID = 01 through p*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AboA1
02	Status	AboA2
03	Alarm Enable	AboA3
04	Warning Enable	AboA5
16	Setting	Abo16
17	SafeState	Abo17
18	WatchRate	Abo18
19	Watchdog	Abo19
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	AboS1
03	Abort	AboS2
05	Recover	AboS3
25	Operate	AboS4
07	GetAttribute	AboS5
09	SetAttribute	AboS6
27	RestoreDefault	AboS7

7.12 *Controller Object (CA)* — The presentation of the Controller (CA) object instance attributes and services are as indicated in Table 13.

Table 13 Controller-CA Instance Object Attributes and Services

Controller *Class ID = 39, Instance ID = 01 through q*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	CAA1
02	Status	CAA2
03	Alarm Enable	CAA3
04	Warning Enable	CAA4
16	Setpoint	CAA16
17	ProcessVariable	CAA17
18	ControlVariable	CAA18
19	DataType	CAA19
64	DataUnits	CAA20
65	AlarmSettleTime	CAA21
66	AlarmErrorBand	CAA22
67	WarningSettleTime	CAA23
68	WarningErrorBand	CAA24
Services		
ID	*Service Name*	*SDM Tag*
01	Reset	CAS1
03	Abort	CAS2
05	Recover	CAS3
25	Operate	CAS4
07	GetAttribute	CAS5
09	SetAttribute	CAS6
27	RestoreDefault	CAS7

8 Protocol Compliance

8.1 A method of testing protocol compliance is required to verify implementation conformance to the standard. An independent compliance testing laboratory has been established at The University of Michigan's Electronics Manufacturing Laboratory for the testing of Modbus/TCP over TCP/IP solutions. This laboratory utilizes an established, documented and freely available mechanism for compliance certification of devices on a Modbus/TCP over a TCP/IP network. This certification includes procedures and reporting mechanisms to demonstrate conformance and interoperability of Modbus/TCP devices. Information on the conformance testing laboratory can be found on the World Wide Web.[5] Additional information on certification procedures can also be found on the World Wide Web.[6]

9 Specific Device Model Mappings

9.1 This section provides for the mapping of network visible specific device structure and behavior, specified in a SEMI SDM specification, to the Modbus/TCP network. Each subsection is devoted to a single Specific Device Model (SDM) specification. Additional SDM mappings are added as sub-sections to this NCS specification according to SEMI guidelines and the guidelines SEMI E54. Unless otherwise noted, all attributes and services described are instance level attributes (as opposed to class level attributes).

NOTE 4: The formats of object instance attributes and services are detailed in the associated SDM specification; the presentation of object attributes and services to the Modbus/TCP over a TCP/IP network is detailed in the tables contained in the following sub-sections and in the Modbus/TCP standard specifications.

NOTE 5: Relationships between object classes, including inheritance is defined in the associated SDM specification and the CDM specification.

9.1.1 The instance identifier of 1 through r, assigned to an object type, refers to the possibility of multiple instantiations of the object type. Refer to Table 2 of this document and the CDM document for a further explanation of object instance identifier assignments.

9.2 *Specific Device Model for Mass Flow Device* — These sections detail the network mapping required to support the Specific Device Model for Mass Flow Devices. Table 14 summarizes the Mass Flow Device Object types. Subsequent Tables 15 to 24 detail the instance attributes and services associated with each Mass Flow Device object type.

Table 14 Mass Flow Device Object Types

SDM Object Identifier	*Object Name*	*Modbus Class ID*
MFD1 (DM)	Device Manager	1
MFD2 (SAC)	Sensor Actuator Controller	2
MFD3	Sensor-AI-MF	129
MFD4	Sensor-AI-AT	130
MFD5	Assembly-MFM	131
MFD6	Sensor-AI-Aux	132
MFD7	Actuator-AO-MF	133
MFD8	Controller	39
MFD9	Local Link	4
MFD10	SISO	134
MFD11	SISO-Setpoint	135
MFD12	Assembly-MFC	136

9.2.1 *Sensor-AI-MF* — The presentation of the Sensor Analog Input Mass Flow (Sensor-AI-MF) object instance attributes and services are as indicated in Table 15.

Table 15 Sensor-AI-MF Object Instance Attributes and Services

Sensor-AI-MF *Class ID = 129, Instance ID = 1 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Flow Totalizer	A1
129	Flow Hours	A2
130	Zero Offset Mode	A5
131	Zeroing Status	A6
132	Autorange Status	A7
Services		
ID	*Service Name*	*SDM Tag*
129	Perform Zero Offset	S1
131	Query-Supported Gas Types	S2
133	Selected Programmed Gas Type	S3
135	Insert Gas Type	S4
137	Delete Gas Type	S5
139	Get Gas Calibration Data Value	S6
141	Set Gas Calibration Data Value	S7
143	Autorange	S8

9.2.2 *Sensor-AI-AT* — The presentation of the Sensor Analog Input Ambient Temperature (Sensor-AI-AT) object instance attributes and services are as indicated in Table 16.

5 http://www.eecs.umich.edu/~sbus

6 http://www.modicon.com/openmbus

Table 16 Sensor-AI-AT Object Instance Attributes and Services

Sensor-AI-AT *Class ID = 130, Instance = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined.	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.3 *Assembly-MFM* — The presentation of the Assembly Mass Flow Meter (Assembly-MFM) object instance attributes and services are as indicated in Table 17.

Table 17 Assembly-MFM Object Instance Attributes and Services

Assembly-MFM *Class ID = 131, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined.	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.4 *Sensor-AI-Aux* — The presentation of the Sensor Analog Input Auxiliary (Sensor-AI-Aux) object instance attributes and services are as indicated in Table 18.

Table 18 Sensor-AI-Aux Object Instance Attributes and Services

Sensor-AI-Aux *Class ID = 132, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined.	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.5 *Actuator-AO-MF* — The presentation of the Actuator Analog Output Mass Flow (Actuator-AO-MF) object instance attributes and services are as indicated in Table 19.

Table 19 Actuator-AO-MF Object Instance Attributes and Services

Actuator-AO – MF *Class ID = 133, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Valve Type	A1
129	Override	A2
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.6 *Controller* — The presentation of the extended Controller (Ca) object instance attributes and services are as indicated in Table 20.

Table 20 Controller Object Instance Attributes and Services

Controller *Class ID = 39, Instance ID = 01 through q*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Alarm Settling Time	CaA21
129	Warning Settling Time	CaA24
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.7 *Local Link* — The presentation of the extended Local Link (Lnk) object instance attributes and services are as indicated in Table 21.

Table 21 Local Link Object Instance Attributes and Services

Local Link *Class ID = 4, Instance ID = 01 through j*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined.	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.8 *SISO* — The presentation of the Single Input Single Output (SISO) object instance attributes and services are as indicated in Table 22.

Table 22 SISO Object Instance Attributes and Services

SISO *Class ID = 134, Instance = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Input	A1
129	Output	A2
130	Data Type	A3
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.2.9 *SISO-Setpoint* — The presentation of the Single Input Single Output Setpoint (SISO-Setpoint) object instance attributes and services are as indicated in Table 23.

Table 23 SISO-Setpoint Object Instance Attributes and Services

SISO-Setpoint *Class ID = 135, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
161	Ramp Type	A33
162	Ramp Rate	A34
163	Ratio	A35
Services		
ID	*Service Name*	*SDM Tag*
---	No additional services defined.	--

9.2.10 *Assembly-MFC* — The presentation of the Assembly Mass Flow Controller (Assembly-MFC) object instance attributes and services are as indicated in Table 24.

Table 24 Assembly-MFC Object Instance Attributes and Services

Assembly-MFC *Class ID = 136, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined.	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined.	--

9.3 *Specific Device Model For In-Situ Particle Monitor Device* — These sections detail the network mapping required to support the Specific Device Model for In-Situ Particle Monitor (ISPM) Devices. Table 25 summarizes the In-Situ Particle Monitor Device Object types. Subsequent Tables 26 to 40 details the attributes and services associated with each In-Situ Particle Monitor Device object type.

Table 25 In-Situ Particle Monitor Device Object Types

SDM Object Identifier	*Object Name*	*Modbus Class ID*
ISPMD1 (DM)	Device Manager	1
ISPMD2 (SAC)	Sensor Actuator Controller	2
ISPMD3	Sensor-AI-LCS	137
ISPMD4	Sensor-AI-SLS	138
ISPMD5	Sensor-AI-MNS	139
ISPM16	Sensor-AI-Counter	140
ISPMD17	Assembly-ISPM#1	141
ISPMD18	Assembly-ISPM#2	142
ISPMD19	Assembly-ISPM#3	143
ISPMD20	Assembly-ISPM#4	144
ISPMD21	Assembly-ISPM#5	145
ISPMD22	Assembly-ISPM#6	146
ISPMD23	Assembly-ISPM#7	147
ISPMD24	Assembly-ISPM#8	148
ISPMD25	Assembly-ISPM#9	149

9.3.1 *Device Manager (DM)* — The presentation of the extended ISPM Device Manager (DM) object attributes and services are as indicated in Table 26.

Table 26 DM Object Instance Attributes and Services

Device Manager Object (DM) *Class ID = 01, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
128	Gain	DmA33
129	Filter Bandwidth	DmA34
130	Tool State	DmA35
131	Laser Status	DmA36
132	Flow Path	DmA37
133	Volume	DmA38
134	Volume Units	DmA39
135	Leak Status	DmA40
136	Time Stamp	DmA41
Services		
ID	*Service Name*	*CDM Tag*
33	Laser On	DmS1
35	Laser Off	DmS2

9.3.2 *Sensor Actuator Controller (SAC)* — The presentation of the extended ISPM Sensor Actuator Controller (SAC) object attributes and services are as indicated in Table 27.

Table 27 SAC Object Instance Attributes and Services

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
65	Number of Bins	SacA65
66	Count Mode	SacA66
67	Duration	SacA67
Services		
ID	*Service Name*	*CDM Tag*
33	Clear Counts	SacS33

9.3.3 *Sensor-AI-LCS* — The presentation of the Sensor Analog Input Laser Current Sensor (Sensor-AI-LCS) object attributes and services are as indicated in Table 28.

Table 28 Sensor-AI-LCS Object Instance Attributes and Services

Sensor-AI-LCS *Class ID = 137, Instance ID = 1 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	LcsA1
129	Full Scale	LcsA2
130	Alarm Settling Time	LcsA3
131	Warning Settling Time	LcsA4
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.4 *Sensor-AI-SLS* — The presentation of the Sensor Analog Input Stray Light Sensor (Sensor-AI-SLS) object attributes and services are as indicated in Table 29.

Table 29 Sensor-AI-SLS Object Instance Attributes and Services

Sensor-AI-SLS *Class ID = 138, Instance ID = 1 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	SlsA1
129	Full Scale	SlsA2
130	Alarm Settling Time	SlsA3
131	Warning Settling Time	SlsA4
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.5 *Sensor-AI-MNS* — The presentation of the Sensor Analog Input Medium Noise Sensor (Sensor-AI-MNS) object attributes and services are as indicated in Table 30.

Table 30 Sensor-AI-MNS Object Instance Attributes and Services

Sensor-AI-MNS *Class ID = 139, Instance ID = 1 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	MnsA1
129	Full Scale	MnsA2
130	Alarm Settling Time	MnsA3
131	Warning Settling Time	MnsA4
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.6 *Sensor-AI-Counter* — The presentation of the Sensor Analog Input Counter (Sensor-AI-Counter) object attributes and services are as indicated in Table 31.

Table 31 Sensor-AI-Counter Object Instance Attributes and Services

Sensor-AI-Counter *Class ID = 140, Instance ID = 1 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	CounterA1
129	Full Scale	CounterA2
130	Alarm Settling Time	CounterA3
131	Warning Settling Time	CounterA4
132	Upper Size	CounterA5
133	Lower Size	CounterA6
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.7 *Assembly-ISPM#1* — The presentation of the Assembly #1 In-Situ Particle Monitor (Assembly-ISPM#1) object attributes and services are as indicated in Table 32.

Table 32 Assembly-ISPM#1 Object Instance Attributes and Services

Assembly-ISPM#1 *Class ID = 141, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.8 *Assembly-ISPM#2* — The presentation of the Assembly #2 In-Situ Particle Monitor (Assembly-ISPM#2) object attributes and services are as indicated in Table 33.

Table 33 Assembly-ISPM#2 Object Instance Attributes and Services

Assembly-ISPM#2 *Class ID = 142, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.9 *Assembly-ISPM#3* — The presentation of the Assembly #3 In-Situ Particle Monitor (Assembly-ISPM#3) object attributes and services are as indicated in Table 34.

Table 34 Assembly-ISPM#3 Object Instance Attributes and Services

Assembly-ISPM#3 *Class ID = 143, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.10 *Assembly-ISPM#4* — The presentation of the Assembly #4 In-Situ Particle Monitor (Assembly-ISPM#4) object attributes and services are as indicated in Table 35.

Table 35 Assembly-ISPM#4 Object Instance Attributes and Services

Assembly-ISPM#4 *Class ID = 144, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.11 *Assembly-ISPM#5* — The presentation of the Assembly #5 In-Situ Particle Monitor (Assembly-ISPM#5) object attributes and services are as indicated in Table 36.

Table 36 Assembly-ISPM#5 Object Instance Attributes and Services

Assembly-ISPM#5 *Class ID = 145, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.12 *Assembly-ISPM#6* — The presentation of the Assembly #6 In-Situ Particle Monitor (Assembly-ISPM#6) object attributes and services are as indicated in Table 37.

Table 37 Assembly-ISPM#6 Object Instance Attributes and Services

Assembly-ISPM#6 *Class ID = 146, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.13 *Assembly-ISPM#7* — The presentation of the Assembly #7 In-Situ Particle Monitor (Assembly-ISPM#4) object attributes and services are as indicated in Table 38.

Table 38 Assembly-ISPM#7 Object Instance Attributes and Services

Assembly-ISPM#7 *Class ID = 147, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.14 *Assembly-ISPM#8* — The presentation of the Assembly #8 In-Situ Particle Monitor (Assembly-ISPM#8) object attributes and services are as indicated in Table 39.

Table 39 Assembly-ISPM#8 Object Instance Attributes and Services

Assembly-ISPM#8 *Class ID = 148, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.15 *Assembly-ISPM#9* — The presentation of the Assembly #9 In-Situ Particle Monitor (Assembly-ISPM#9) object attributes and services are as indicated in Table 40.

Table 40 Assembly-ISPM#9 Object Instance Attributes and Services

Assembly-ISPM#9 *Class ID = 149, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4 *Specific Device Model for Endpoint Device* — These sections detail the network mapping required to support the Specific Device Model for Endpoint Devices. Table 41 summarizes the Endpoint Device object types. Subsequent Tables 42 to 48 detail the attributes and services associated with each Endpoint Device object type.

Table 41 Endpoint Device Object Types

SDM Object Identifier	*Object Name*	*Modbus Class ID*
EPD1 (DM)	Device Manager	1
EPD2 (SAC)	Sensor Actuator Controller	2
EPD3	Sensor-BI-TH-EP	150
EPD4	Assembly-EPD#1	151
EPD5	Assembly-EPD#2	152
EPD6	Assembly-EPD#3	153
EPD7	Assembly-EPD#4	154

9.4.1 *Device Manager* — The presentation of the Device Manager object attributes and services that are specified in the CDM is specified in Section 7. The presentation of the extended Device Manager object attributes and services that are specified in the SDM for Endpoint Devices is specified in Table 42.

Table 42 DM Object Instance Attributes and Services

Device Manger (DM) *Class ID = 0001, Instance ID = 0001*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.2 *Sensor Actuator Controller* — The presentation of the Sensor Actuator Controller object attributes and services that are specified in the CDM is specified in Section 7. The presentation of the extended Sensor Actuator Controller object attributes and services that are specified in the SDM for Endpoint Devices is specified in Table 43.

Table 43 SAC Object Instance Attributes and Services

Sensor Actuator Controller (SAC) *Class ID = 0002, Instance ID = 0001*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
65	Number of Endpoint Objects	SacA65
Services		
ID	*Service Name*	*SDM Tag*
33	Reset Endpoint	S33
34	Download Recipe	S34
35	Upload Recipe	S35
36	Calibrate	S36

9.4.3 *Sensor-BI-TH-EP* — The presentation of the Sensor, Binary Input, Threshold type, Endpoint object attributes and services is specified in Table 44.

Table 44 Sensor-BI-TH-EP Object Instance Attributes and Services

Sensor-BI-TH-EP *Class ID = 0150, Instance ID = 0001 - 1024*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
129	Value	SbithepA16
130	Reading Valid	SbithA64
131	State	SbithA65
132	Status	SbithA66
133	Minimum Time	EpA1
134	Maximum Time	EpA2
135	Target Time	EpA3
136	Elapsed Time	EpA4
137	Time Stamp	EpA5
138	Recipe Identifier	EpA6
139	Step Identifier	EpA7
Services		
ID	*Service Name*	*SDM Tag*
129	Endpoint On	S1
130	Endpoint Off	S2
131	Endpoint Start	S3
132	Endpoint Suspend	S4
133	Endpoint Resume	S5

9.4.4 *Assembly-EPD#1* — The presentation of the Assembly #1 Endpoint Device object attributes and services is specified in Table 45.

Table 45 Assembly-EPD#1 Object Instance Attributes and Services

Assembly-EPD#1 *Class ID = 0151, Instance ID = 0001*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.5 *Assembly-EPD#2* — The presentation of the Assembly #2 Endpoint Device object attributes and services is specified in Table 46.

Table 46 Assembly-EPD#2 Object Instance Attributes and Services

Assembly-EPD#2 *Class ID = 0152, Instance ID = 0001*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.6 *Assembly-EPD#3* — The presentation of the Assembly #3 Endpoint Device object attributes and services is specified in Table 47.

Table 47 Assembly-EPD#3 Object Instance Attributes and Services

Assembly-EPD#3 *Class ID = 0153, Instance ID = 0001*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.7 *Assembly-EPD#4* — The presentation of the Assembly #4 Endpoint Device object attributes and services is specified in Table 48.

Table 48 Assembly-EPD#4 Object Instance Attributes and Services

Assembly-EPD#4 *Class ID = 0154, Instance ID = 0001*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

NOTICE: SEMI makes no warranties or representations as to the suitability of the standard set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E54.10-0600
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK SPECIFIC DEVICE MODEL FOR AN IN-SITU PARTICLE MONITOR DEVICE

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 2, 2000. Initially available at www.semi.org May 2000; to be published June 2000.

1 Purpose

1.1 This specification is part of a suite of standards that specify the implementation of SEMI standards for the Sensor/Actuator Network. The specific purpose of this specification is to describe a network independent application model comprised of device objects that are common to all In-Situ Particle Monitor Devices on a semiconductor equipment Sensor/Actuator communications network.

2 Scope

2.1 An In-Situ Particle Monitor (ISPM) is a device that measures and counts particles. These devices classify by size and count, particles in the environment (gaseous, liquid, or vacuum) utilizing a technique such as detecting light from a sample region of the environment's space. These counts are accumulated in bins and then reported. The number of bins varies by vendor and model.

2.2 This specification specifically addresses the minimum attributes, services, and behavior an In-Situ Particle Monitor (ISPM) device must support to be interoperable on the Sensor/Actuator Network.

2.3 This specification is intended to ensure a high-degree of device interoperability on the Sensor/Actuator Network, while still allowing flexibility for product differentiation and technology evolution.

2.4 The model specified in this specification is used in conjunction with SEMI E54.1 (Standard for Sensor/Actuator Network Common Device Model (CDM)) to completely describe the ISPM as it appears from the network interface.

2.5 This specification, together with SEMI E54, SEMI E54.1, and one of the Sensor/Actuator Network Communication Specifications, form a complete interoperability specification for the ISPM.

2.6 To comply with this specification, a device must implement and support, at a minimum, the required attributes, services, and behavior identified in these documents. Support for optional attributes, services, and behavior is not required to be compliant to this specification. Optional attributes, services, and behavior are specified in these documents to promote further device interoperability as features evolve and are adopted by more manufacturers. If optional attributes, services, and behavior are implemented for this device they must be implemented as identified in this document.

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

3 Limitations

3.1 This specification is a companion to a suite of specifications that together make up the Sensor/Actuator Network Communication standard. Therefore, using portions of this specification that relate to network communications necessarily requires an understanding of the associated network specification.

3.2 As this document is a specification for the In-Situ Particle Monitor Device Model, it does not contain any definition of objects, attributes, services, or behavioral descriptions that are already defined in SEMI E54.1. Additional attributes, attribute assignments, services, and/or service parameters that are ISPM Device specific and/or implementation specific are contained in this specification.

3.3 While this specification is sufficient to completely describe the ISPM as it appears from the network, it does not fully describe behavior of the ISPM which is not visible from the network. This allows flexibility in implementation techniques and product differentiation between manufacturers. Manufacturer specific objects may be defined by the manufacturer but are, by definition, outside the scope of this standard.

3.4 This specification is compatible, but not compliant with SEMI E39. This means that although this specification 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 specification are compatible with those of SEMI E39. However, SEMI E39 has specific requirements that are

intended for higher level applications and thus are not applied to the In-Situ Particle Monitor Device Model.

3.5 Operation over the entire range specified for an attribute within a specific object is not a requisite for compliance with this specification.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E54 — Sensor/Actuator Network Standard

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

NOTE 1: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *LCS* — an ISPM sensor named the Laser Current Sensor.

5.1.2 *MNS* — an ISPM sensor named the Median Noise Sensor.

5.1.3 *SLS* — an ISPM sensor named the Stray Light Sensor.

5.2 *Terminology Defined in This Document*

5.2.1 *In-Situ Particle Monitor (ISPM)* — a self-contained device, consisting of a laser that generates light, a light detector, counters, diagnostics and control and signal-processing electronics, commonly used in the semiconductor industry to measure and count particles in a specific area.

5.3 This document inherits the terminology defined by SEMI E54.1 [3].

5.3.1 *Attribute*

5.3.2 *Behavior*

5.3.3 *Boolean*

5.3.4 *Byte*

5.3.5 *Character*

5.3.6 *Common Device Model (CDM)*

5.3.7 *Data Type*

5.3.8 *Data Units*

5.3.9 *Device*

5.3.10 *Device Manager (DM) Object*

5.3.11 *Device Model*

5.3.12 *Double Integer (DINT*)

5.3.13 *Enumerated Byte*

5.3.14 *Full Scale Range*

5.3.15 *Instance*

5.3.16 *Last Valid Value (LVV)*

5.3.17 *Long Integer (LINT)*

5.3.18 *Long Real (LREAL)*

5.3.19 *Manufacturer*

5.3.20 *Nibble*

5.3.21 *Null Character*

5.3.22 *Object*

5.3.23 *Real (REAL)*

5.3.24 *S, A, and C Objects*

5.3.25 *Sensor Actuator Controller (SAC) Object*

5.3.26 *Service*

5.3.27 *Short Integer (SINT)*

5.3.28 *Signed Integer (INT)*

5.3.29 *State Diagram*

5.3.30 *Test String*

5.3.31 *Unsigned Double Integer (UDINT)*

5.3.32 *Unsigned Double Long Integer (UDLINT)*

5.3.33 *Unsigned Integer (UINT)*

5.3.34 *Unsigned Long Integer (ULINT)*

5.3.35 *Unsigned Short Integer (USINT)*

6 Requirements and Specifications

6.1 In order to implement this standard in an In-Situ Particle Monitor Device, it is necessary to also implement SEMI E54.1 and one of the Sensor/Actuator Network Communication Standards [2]. See Section 1.1 for more information on a complete interoperability standard.

6.2 This specification also requires the implementation of a Date_And_Time data structure used to represent the current device Date and Time. Table 1 defines the format of the Date_And_Time data type.

Table 1 Date_And_Time Format

Data Item	*Description*	*Range*
1	Number of milli-seconds since midnight	Unsigned Double Integer (UDINT)
2	Number of days since	Unsigned Integer

Data Item	*Description*	*Range*
	1/1/72	(UINT)

7 Conventions

7.1 This document embraces the conventions and notations stated in Section 6 of SEMI E54.1 [3].

8 Device High Level Structure

8.1 *General Description*

8.1.1 The high-level object view of an In-Situ Particle Monitor (ISPM) Device is shown in Figure 1.

8.1.2 Note that the "ISPM" device object is depicted in Figure 1 only for the purposes of illustrating a high level view of the device and its component objects. In the context of this document, this object is not addressable, does not have addressable attributes, does not have accessible services, and does not exhibit any defined behavior.

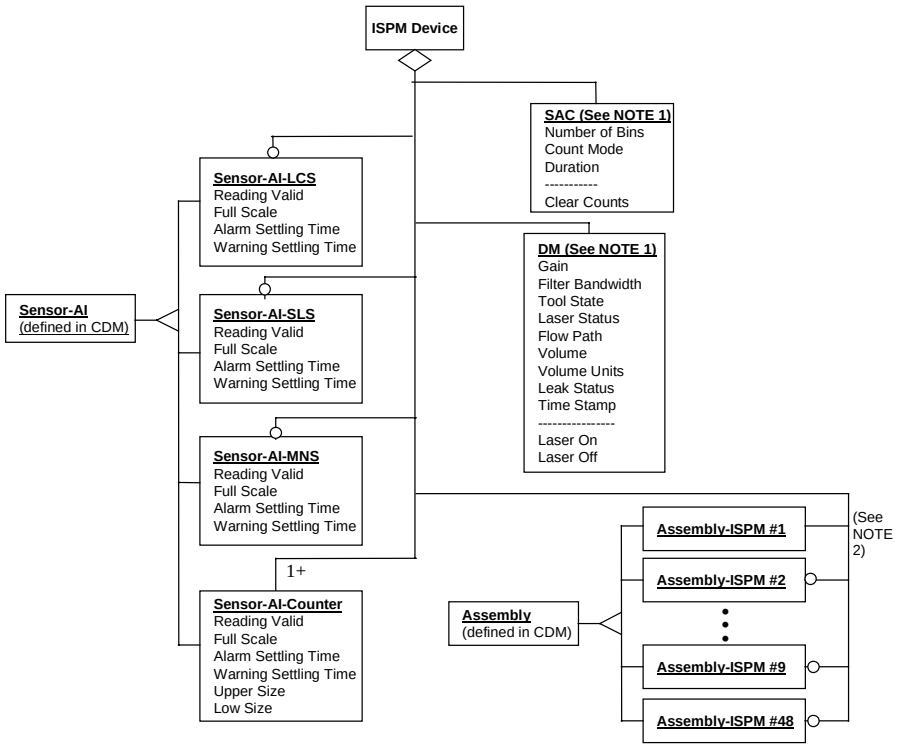
8.1.3 In the remainder of this section, this document defines in detail all of the component objects unique to the ISPM device. References, rather than definitions, are included for the DM, the SAC, and other objects defined in SEMI E54.1 [3].

8.1.4 Many of the objects defined in this document inherit properties from other objects. The properties inherited include attribute, services, and behavior definitions. These other objects are specified here or in SEMI E54.1 [3].

8.1.5 This document provides for future extensions as well as manufacturer specific enhancements by reserving object attribute identifiers and object service identifiers. Specifically, all object definitions in this document specify or reserve the first 64 attribute identifiers (A1 through A64) and the first 64 service identifiers (S1 through S64) allowing manufacturers to specify identifiers beyond these ranges. Additionally, byte enumerated attributes are specified or reserved from 0 to 63 allowing manufacturers to specify an enumeration beyond this range (64 to 255).

8.1.6 *In-Situ Particle Monitor (ISPM) Device Description*

8.1.6.1 An In-Situ particle Monitor device profile is composed of the component objects and object relationships shown in Figure 1.



NOTE 1: The DM and SAC are taken from the CDM. Additional attributes and services are added to support the ISPM device.

NOTE 2: Assembly-ISPM #1 and Assembly-ISPM #6 objects are required. Other Assembly-ISPM objects are optional.

Figure 1
In-Situ Particle Monitor Device High Level Structure

8.1.7 *General Requirements*

8.1.7.1 *Device Objects* — All objects are defined in terms of their object name and Class/Object identifier. Identifiers for all objects described in this document are summarized in Table 2.

Table 2 In-Situ Particle Monitor Device Objects

Referenced Document Section	*Object Name*	*Class/Object Identifier*	*Minimum #*	*Maximum #*
8.2	Device Manager (DM)	ISPMD1	1	1
8.3	Sensor Actuator Controller (SAC)	ISPMD2	1	1
8.5	Sensor-AI-LCS	ISPMD3	0	1
8.6	Sensor-AI-SLS	ISPMD4	0	1
8.7	Sensor-AI-MNS	ISPMD5	0	1
8.8	Sensor-AI-Counter	ISPMD16	1	1024
8.9	Assembly-ISPM#1	ISPMD17	1	1
8.9	Assembly-ISPM#2	ISPMD18	0	1
8.9	Assembly-ISPM#3	ISPMD19	0	1
8.9	Assembly-ISPM#4	ISPMD20	0	1
8.9	Assembly-ISPM#5	ISPMD21	0	1
8.9	Assembly-ISPM#6	ISPMD22	1	1
8.9	Assembly-ISPM#7	ISPMD23	0	1
8.9	Assembly-ISPM#8	ISPMD24	0	1
8.9	Assembly-ISPM#9	ISPMD25	0	1
8.9	Assembly-ISPM#48	ISPMD64	0	1
—	Reserved	ISPM26–ISPM63	—	—
—	Manufacturer Specified	> ISPM64	—	—

8.1.7.2 *Object Services* — Not all object services listed in this document can necessarily be requested over the network. They are included in this document because their behavior may generate network activity.

8.1.7.3 *Object Behavior* — For all service requests received over the network that are not supported by the object, or contain a parameter value which is beyond the supported range, or which is otherwise invalid, a network specific service error response is generated.

8.2 *Device Manager Object (DM)* — The Device Manager object is the device component responsible for managing and consolidating the device operation as specified in SEMI E54.1 [3]. The following sections specify the components of the DM object that are not specified in SEMI E54.1 or require further definition than specified in SEMI E54.1.

8.2.1 *Device Manager Object Attributes* — Required and optional DM object attributes are listed in Table 3.

Table 3 DM Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	A1	R	Yes	refer to CDM [1]
Exception Detail Alarm	A13	R	No	refer to CDM [1]
Exception Detail Warning	A14	R	No	refer to CDM [1]
Gain	A33	RW *	No	REAL
Filter Bandwidth	A34	RW *	No	REAL
Tool State	A35	R	No	USINT, enumerated
Laser Status	A36	R	Yes	USINT, enumerated
Flow Path	A37	RW	No	USINT
Volume	A38	R	No	REAL
Volume Units	A39	R	No	Byte, enumerated

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Leak Status	A40	R	No	Byte, enumerated
Time Stamp	A41	RW	No	Date_And_Time
Reserved	A42–A64	—	—	Reserved for SDM future expansion
Manufacturer Specified	> A64	—	—	Manufacturer Specific attributes

NOTE 1: "*" Indicates that the specific attribute is nonvolatile. Nonvolatile requires that the current attribute value be maintained through a component power cycle.

8.2.1.1 *Device Type* — An attribute that uniquely identifies the type of the device on the network. The device type attribute is assigned as follows:

In-Situ Particle Monitor Device = "ISPM"

8.2.1.2 *Exception Detail Alarm (Optional)* — An attribute which identifies the detailed alarm status of the device. Table 4 defines the bit assignments associated with the alarm exception detail.

Table 4 Exception Detail Alarm Bit Assignments

Bit	*Device Specific Alarm[0]*
0	Reserved
1	Interlock
2	Sensor Not Detected
3	Leak Detected
4	Reserved
5	Sensor Type Changed
6	Reserved
7	Reserved

Bit	*Device Specific Alarm[1]*
0	High Laser Current
1	High Stray Light
2	High Median Noise
3	High Calibration
4	High Fill Rate
5	High Flow Rate
6	Reserved
7	Reserved

Bit	*Device Specific Alarm[2]*
0	Low Laser Current
1	Low Stray Light
2	Low Median Noise
3	Low Calibration
4	Low Fill Rate
5	Low Flow Rate
6	Reserved
7	Reserved

8.2.1.3 *Exception Detail Warning (Optional)* — An attribute which identifies the detailed warning status of the device. Table 5 defines the bit assignments associated with the warning exception detail.

Table 5 Exception Detail Warning Bit Assignments

Bit	*Device Specific Warning[0]*
0	Reserved
1	0
2	0
3	0
4	Reserved
5	0
6	Reserved
7	Reserved

Bit	*Device Specific Warning[1]*
0	High Laser Current
1	High Stray Light
2	High Median Noise
3	High Calibration
4	High Fill Rate
5	High Flow Rate
6	Reserved
7	Reserved

Bit	*Device Specific Warning[2]*
0	Low Laser Current
1	Low Stray Light
2	Low Median Noise
3	Low Calibration
4	Low Fill Rate
5	Low Flow Rate
6	Reserved
7	Reserved

8.2.1.4 *Manufacturer Exception Detail Size (Optional)* — An attribute that specifies the number of exception detail bytes included in the alarm or warning details.

8.2.1.5 *Gain (Optional)* — An attribute that specifies the Gain value of the amplifier for the photodiode signal. The factory configured out-of-box value is determined by the manufacturer based on the sensor type.

8.2.1.6 *Filter Bandwidth (Optional)* — An attribute that determines the bandwidth in kilohertz of the amplifier for the photodiode signal. The factory configured out-of-box value is determined by the manufacturer based on the sensor type.

8.2.1.7 *Tool State (Optional)* — An attribute that identifies the current state of the tool that is utilizing the counter. In many cases, the ISPM device can detect and report the Tool State; this variable is utilized for this type of reporting. This attribute is enumerated and is specified by the manufacturer.

8.2.1.8 *Laser Status* — An attribute that records the current status of the laser. This attribute is enumerated. The possible enumeration and the requirement for support are as follows:

0 = LASER OFF (required)
1 = LASER TURNING ON (optional)
2 = LASER ON BUT NOT STABLE (optional)
3 = LASER ON AND STABLE (required)
4 = LASER INTERLOCKED (optional)
5 = SENSOR MISSING (required)
6–63 = Reserved
64–255 = Manufacturer Specified (optional)

8.2.1.9 *Flow Path (Optional)* — An attribute that identifies the current flow path (from multiple sources) being utilized. The source may be modified by a Host at anytime. Changing the "Flow path" attribute connects the ISPM device to a different source. When changing sources, data is not valid for a time interval whose length is a function of the distance to the source.

8.2.1.10 *Volume (Optional)* — An attribute that maintains the sample volume. This attribute is used primarily by liquid particle counters utilizing a sampling technique in a small container to extrapolate the particle count in the volume of interest. The value of this attribute is the size of the small container.

8.2.1.11 *Volume Units (Optional)* — An attribute that specifies the unit for the "Volume" attribute. This attribute is enumerated and can take on one of the following values: milliliters or gallons. (See Appendix 1 of SEMI E54.1 for assigned values.)

8.2.1.12 *Leak Status (Optional)* — An attribute that indicates which one of the potential leak locations is actually leaking. This attribute is enumerated and is specified by the manufacturer. The enumerated value corresponds to a specific leak location.

8.2.1.13 *Time Stamp (Optional)* — An attribute that records the time when the last counting event completed.

8.2.1.14 *Duration (Optional)* — An attribute that defines the interval of time during which the particle counts are collected and put into the bin assemblies.

8.2.1.15 *Initial and Default Values*

Table 6 DM Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Device Type	ISPM	ISPM	ISPM = In-Situ Particle Monitor Device
Exception Detail Alarm	0	0	
Exception Detail Warning	0	0	
Laser Status	0	0	Laser Off

8.2.2 *Device Manager Object Services* — The services provided by the Device Manager object are defined in SEMI E54.1 [1]. The Device Manager object supports the additional services listed below.

Table 7 Device Manager Object Services

Service	*Service Identifier*	*Type*	*Description*
Laser On	S1	R	Used to prompt the device to take the laser from the LASER OFF state to LASER ON AND STABLE state as defined by the Laser Status attribute.
Laser Off	S2	R	Used to prompt the device to take the laser immediately to the LASER OFF state as defined by the Laser Status attribute.
Reserved	S3–S64	—	Reserved for future expansion
Manufacturer Specified	> S64	—	Manufacturer Specific services

8.2.2.1 *Laser On (Optional)* — This service is used to prompt the device to take the laser from the LASER OFF state through supported laser states 1 and/or 2, to state 3, as defined by the Laser Status attribute. If the laser turns on successfully, a "success" response code is returned. If the sensor is missing or interlocked off, a "object state conflict" error response is returned and the laser remains in the appropriate state.

8.2.2.2 *Laser Off (Optional)* — This service is used to prompt the device to take the laser to the LASER OFF state immediately. If the laser turns off successfully, a "success" response is returned. If the sensor is in an interlocked state, a "object state conflict" error response is returned and the laser remains in the interlocked off state.

8.2.3 *Device Manager Object Behavior* — The behavior exhibited by the Device Manager object is defined in SEMI E54.1 [1]. Additional behavior is detailed below.

8.2.3.1 *Device Manager EXECUTING State Behavior* — Required sub-states within the EXECUTING state, descriptions of these substates, and a transition matrix associated with these sub-states are given in Figure 2, Table 8, and Table 9 respectively.

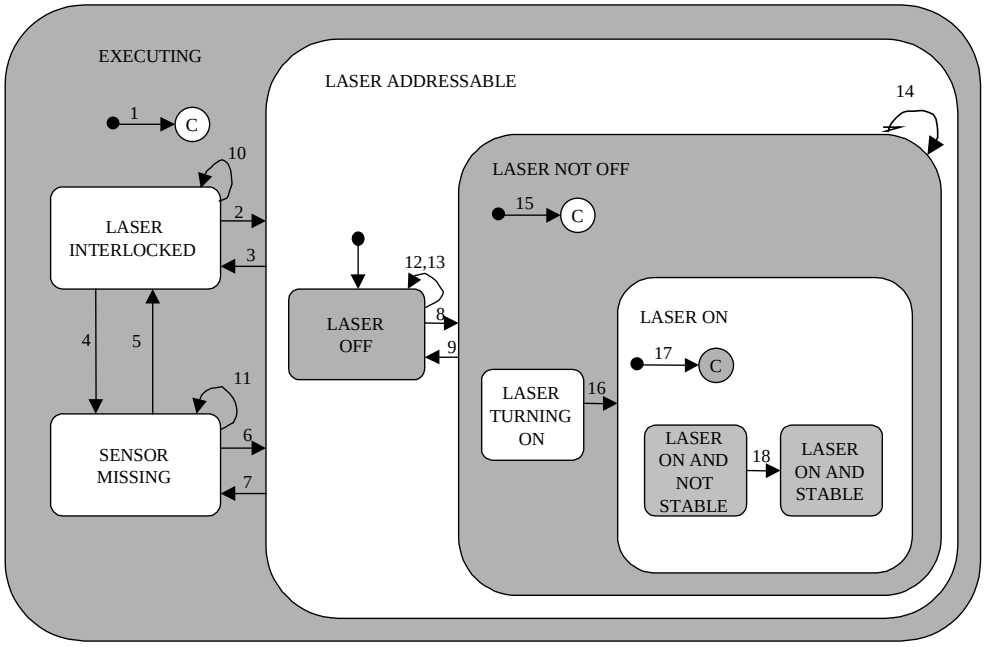


Figure 2
Device Manager Object Behavior Within the EXECUTING State

Table 8 Device Manager Behavior EXECUTING Sub-State Description

State	*Description*
EXECUTING	Laser is in one of the following enumerated states as indicated in the Laser Status Device Manager attribute: LASER INTERLOCKED (optional), LASER MISSING (optional), or LASER OFF. Device will respond to Laser On and Laser Off services as appropriate to move between sub-states within the EXECUTING and LASER ADDRESSABLE states.
LASER ADDRESSABLE	Laser is in one of the following enumerated states as indicated in the Laser Status Device Manager attribute: LASER OFF, LASER ON AND STABLE, LASER TURNING ON (optional), or LASER ON BUT NOT STABLE (optional). Device will respond to Laser On and Laser Off services as appropriate to move between sub-states within the LASER ADDRESSABLE state.
LASER OFF	This is the entry sub-state to LASER ADDRESSABLE; Laser is Off; Laser Status Device Manager attribute is in the enumerated state: LASER OFF. Device is not performing the "counting" process. (See NOTE 1.)
LASER ON AND STABLE	Sub-state of LASER ADDRESSABLE; device is "counting" (See NOTE 1); Laser Status Device Manager attribute is the enumerated state: LASER ON AND STABLE. Laser is performing the "counting" process.
LASER INTERLOCKED	Laser is Interlocked off; Laser Status Device Manager attribute is in the enumerated state: LASER INTERLOCKED. Device is not performing the "counting" process. (See NOTE 1.) Sensor may or may not be missing. Device cannot move to the LASER ADDRESSABLE or SENSOR MISSING states until the interlock is removed.
SENSOR MISSING	Laser sensor is missing; Laser Status Device Manager attribute is in the enumerated state: SENSOR MISSING. Device is not performing the "counting" process. (See NOTE 1.) Device cannot move to the LASER ADDRESSABLE state until the sensor is replaced.
LASER NOT OFF	Laser is in one of the enumerated states as indicated in the Laser Status Device Manager attribute: LASER TURNING ON (optional), LASER ON AND NOT STABLE (optional), or LASER ON AND STABLE. Device will respond to Laser Off service as appropriate to move between sub-states within the LASER ADDRESSABLE state.
LASER TURNING ON	An optional sub-state of LASER NOT OFF. This is a conditional entry sub-state to LASER NOT OFF. Device is preparing to turn on. Device is not performing the "counting" process. (See NOTE 1.)
LASER ON	Laser is in one of the enumerated states as indicated in the Laser Status Device Manager attribute: LASER ON AND NOT STABLE (optional) or LASER ON AND STABLE. Device will respond to Laser Off service as appropriate to move between sub-states within the LASER ADDRESSABLE state.
LASER ON AND NOT STABLE	Sub-state of LASER ON; device is not "counting" (See NOTE 1); Laser Status Device Manager attribute is the enumerated state: LASER ON AND NOT STABLE. Laser is not performing the "counting" process.

NOTE 1: The "counting" process is defined in Section 8.8.3.1.

Table 9 Device Manager Behavior EXECUTING Sub-State Transition Matrix (See NOTE 1.)

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
1	Entry into EXECUTING	Device detects interlock condition, and availability and state of laser.	Conditional: LASER ADDRESSABLE/ LASER OFF or LASER INTERLOCKED or SENSOR MISSING	If the sensor is in an interlocked or missing condition then state is LASER INTERLOCKED. If laser interlock and or laser missing is not supported then state is LASER OFF. Set Laser Status attribute to appropriate value.	Entry state depends on availability of laser and physical laser interlock setting on device.
2	LASER INTERLOCKED	Device detects removal of interlock condition	LASER MISSING	Set Laser Status attribute to	Device moves to LASER OFF state.

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
		and determines that sensor is not missing.		appropriate value.	
3	LASER ADDRESSABLE	Device detects that it has been set to a laser interlock condition.	LASER INTERLOCKED	Set Laser Status attribute to appropriate value.	Valid for all sub-states of LASER ADDRESSABLE.
4	LASER INTERLOCKED	Device detects removal of interlock condition and determines that sensor is missing.	SENSOR MISSING	Set Laser Status attribute to appropriate value.	
5	SENSOR MISSING	Device detects that it has been set to a laser interlock condition.	LASER INTERLOCKED	Set Laser Status attribute to appropriate value.	
6	SENSOR MISSING	Device detects replacement of sensor.	LASER ADDRESSABLE/ LASER OFF	Set Laser Status attribute to appropriate value.	Device moves to LASER OFF.
7	LASER ADDRESSABLE/ LASER OFF	Device detects that sensor is missing.	SENSOR MISSING	Set Laser Status attribute to appropriate value.	
8	LASER OFF	Laser On request	LASER ON AND STABLE	Take the object from the LASER OFF state through optional laser states LASER TURNING ON and LASER ON BUT NOT STABLE to the required LASER ON AND STABLE state, turning on the laser and stabilizing it respectively. Set the Laser Status attribute to the appropriate values throughout the transition. Issue Laser On response and begin the "counting" process. (See NOTE 2.)	LASER TURNING ON and "LASER ON BUT NOT STABLE" are optional intermediate states between LASER OFF and LASER ON AND STABLE. Laser must not be missing or in an interlocked off state. Service response is not issued until transition to LASER ON AND STABLE is completed.
9	LASER ON AND STABLE	Laser Off request	LASER OFF	Stop the "counting" process. (See NOTE 2.) Turn the laser off. Issue Laser Off response.	
10	LASER INTERLOCKED	Laser Off request or Laser On request	LASER INTERLOCKED	Error response	Object cannot move from this state until interlock is turned off.
11	SENSOR MISSING	Laser Off request or Laser On request	SENSOR MISSING	Error response	Object cannot move to LASER ADDRESSABLE state until sensor is replaced.
12	LASER OFF	Laser Off request	LASER OFF	Error response	Laser is already off.
13	LASER OFF	Laser On request	LASER OFF	Device attempts to take the laser from the "Laser Off" state through optional laser	Behavior associated with determining that laser will not turn on properly or

#	Current State	Trigger	New State	Action	Comment
				state “Laser Turning On” and to required state “Laser On But Not Stable”. Set the Laser Status attribute to the appropriate values throughout the transition attempt. Laser either won’t turn on properly or won’t stabilize. Turn laser back off and generate Error response.	won’t stabilize is manufacturer specific.
14	LASER ON		LASER ON AND STABLE	Error response	Laser is attempting to turn on.
15	Entry into LASER NOT OFF	Laser On request	Conditional: LASER NOT OFF/LASER TURNING ON or LASER ON/LASER ON AND NOT STABLE or LASER ON AND STABLE	Set Laser Status attribute to appropriate value.	Entry state depends on device support for optional states LASER TURNING ON and LASER ON / LASER ON AND NOT STABLE or LASER ON AND STABLE
16	LASER TURNING ON	Device detects that the laser is on.	LASER ON/LASER ON AND NOT STABLE or LASER ON AND STABLE	Set Laser Status attribute to appropriate value.	Behavior associated with determining that laser is in the process of turning on and is manufacturer specific.
17	Entry into LASER ON	Laser On request	Conditional: LASER ON/LASER ON AND NOT STABLE or LASER ON AND STABLE	Set Laser Status attribute to appropriate value.	Entry state depends on device support for optional states LASER ON AND NOT STABLE or LASER ON AND STABLE.
18	LASER ON AND NOT STABLE	Device determines that the laser is stable.	LASER ON AND STABLE	Set Laser Status attribute to appropriate value.	Laser is on. Behavior associated with determining that laser will turn on properly or won’t stabilize is manufacturer specific.

NOTE 1: Note that this matrix augments the Device Manager Behavior State Transition Matrix as defined in SEMI E54.1 [3]. All transitions are in addition to those specified in SEMI E54.1.

NOTE 2: The “counting” process is defined in Section 8.8.3.1.

8.3 *Sensor Actuator Controller Object (SAC)* — The Sensor Actuator Controller object is the device component responsible for coordinating the interaction of the ISPM device with the sensory/actuation/control environment as specified in SEMI E54.1 [3]. The following sections specify the components of the SAC object that are not specified in the Common Device model or require further definition than specified in the CDM.

8.3.1 *Sensor Actuator Controller Object Attributes* — The attributes provided by the Sensor Actuator Controller object are defined in SEMI E54.1 [3]. Table 10 contains the additional attributes required for the Sensor Actuator Controller object.

Table 10 SAC Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Number of Bins	SacA65	R*	Yes	UINT
Count Mode	SacA66	RW*	Yes	Enumerated, USINT
Duration	SacA67	R	No	REAL

NOTE 1: "*" Indicates that the specific attribute is nonvolatile. Nonvolatile requires that the current attribute value be maintained through a component power cycle.

8.3.1.1 *Number of Bins* — An attribute that specifies the number of counters in the device. This value represents the number of Sensor-AI-Counter objects in the device.

8.3.1.2 *Count Mode* — An attribute that specifies whether all count object values are cleared (to zero) when read. This attribute is an enumerated USINT that can take on one of the following values:

0 = The particle counter count is not affected by reading the count.
1 = The particle counter count is cleared to zero when the count ("value" attribute of the Counter object) is read or reported through a service request to or from an Assembly-ISPM object that contains that count.
2 = The particle counter count is cleared to zero when the count ("value" attribute of the Counter object) is read or reported (regardless of the connection) through a service request to or from either an Assembly-ISPM object that contains that count ("value" attribute), or the Counter object that contains the count.
3–63 = Reserved
64–255 = Manufacturer Specified

8.3.1.3 *Duration (Optional)* — An attribute that defines the interval of time during which the particle counts (value of *Value* object attribute) are collected and put into the bin assemblies.

8.3.1.4 *Initial and Default Values*

Table 11 SAC Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Number of Bins	Manufacturer Specified	Manufacturer Specified	
Count Mode	LVV	0	Do not clear particle counter value.
Duration	Manufacturer Specified	Manufacturer Specified	

8.3.2 *Sensor Actuator Controller Object Services* — The services provided by the Sensor Actuator Controller object are defined in SEMI E54.1 [1]. Table 12 contains the additional services required for the Sensor Actuator Controller object.

Table 12 SAC Object Services

Service	*Service Identifier*	*Type*	*Description*
Clear Counts	S33	R	Used to clear the "Value" attribute of all Sensor-AI-Counter sensor objects.
Reserved	S34–S64	—	Reserved for future expansion
Manufacturer Specified	> S64	—	Manufacturer Specific services

8.3.2.1 *Clear Counts (Optional)* — This service is used to instruct all of the Sensor-AI-Counter object to perform a one-time clear of their respective Sensor-AI-Counter "*Value*" attributes by setting the value to zero. There are no parameters required for this service.

8.3.3 *Sensor Actuator Controller Object Behavior*

8.3.3.1 The behavior exhibited by the Sensor Actuator Controller object is defined in SEMI E54.1 [3]. Additional behavior is detailed below.

8.3.3.2 The "*Value*" attribute is cleared (to zero) in each ISPM counter sensor object, after being read, if the "*Count Mode"* attribute has a value of 1. It may also be cleared by a "Clear Counts" service of the SAC object or the "Reset" service issued to the S-Analog Sensor class.

8.4 *Sensor-AI Object* — The Sensor-AI object is the device component responsible for coordinating the behavior common to all analog input sensor elements in the ISPM device as specified in SEMI E54.1 [3].

8.4.1 *Sensor-AI Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1 [1]. There are no additional attributes required for the Sensor-AI object.

8.4.2 *Sensor-AI Object Services* — The services provided by the Sensor-AI object are defined in SEMI E54.1 [3]. There are no additional services required for the Sensor-AI object.

8.4.3 *Sensor-AI Object Behavior* — The behavior exhibited by the Sensor-AI object is defined in SEMI E54.1 [3]. There is no additional behavior required for the Sensor-AI object.

8.5 *Sensor-AI-LCS Object* — The Sensor-AI-LCS (Laser Current Sensor) object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1 [3]. The Sensor-AI-LCS is the device component responsible for retrieving a reading from a physical laser current sensor, optionally correcting the reading with a manufacturer specified algorithm, or algorithms, then making the value available through the "*Value*" attribute.

8.5.1 *Sensor-AI-LCS Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1 [3]. The Sensor-AI object attribute content and its attribute extensions to support the Sensor-AI-LCS object are listed in Table 13.

Table 13 Sensor-AI and Sensor-AI-LCS Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Value	SaiA16	R	Yes	UINT
Gain	SaiA65	R	Yes	REAL
Data Type	SaiA66	R	Yes	USINT
Data Units	SaiA67	R	Yes	Enumerated, UINT milliAmps only value supported
Reading Valid	LcsA1	R	Yes	Enumerated, Byte
Full Scale	LcsA2	R	Yes	UINT
Alarm Settling Time	LcsA3	R	Yes	UINT
Warning Settling Time	LcsA4	R	Yes	UINT
Reserved	LcsA5–LcsA64	—	—	Reserved for future expansion.
Manufacturer Specified	> LcsA64	—	—	Manufacturer Specific attributes

8.5.1.1 *Value* — The attribute that maintains the laser current sensor value. This value is always read as milliAmps.

8.5.1.2 *Gain* — The attribute that specifies the gain of the laser current measuring circuit.

8.5.1.3 *Data Type* — The attribute which defines the format of the data associated with the "value" attribute.

8.5.1.4 *Data Units* — The attribute which specifies the units for the "*Value*" attribute. This attribute is an enumerated UINT that can take only one value:

MilliAmps (as assigned in Appendix 1 of SEMI E54.1).

8.5.1.5 *Reading Valid* — An attribute which specifies whether the "*Value*" attribute contains a valid value. This attribute is an enumerated byte that can take on one of the following values:

0 = Invalid, or
1 = Valid.

8.5.1.6 *Full Scale* — An attribute which specifies the maximum value allowed for the "*Value*" attribute.

8.5.1.7 *Alarm Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before an alarm is generated.

8.5.1.8 *Warning Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before a warning is generated.

8.5.1.9 *Initial and Default Values*

Table 14 Sensor-AI and Sensor-AI-LCS Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value	0	0	
Gain	LVV	1.0	
Data Type	LVV	UINT	
Data Units	LVV	MilliAmps	See Appendix 1 of SEMI E54.1.
Reading Valid	0	0	Invalid
Full Scale	LVV	0XFFFF	
Alarm Settling Time	LVV	1000	Milliseconds
Warning Settling Time	LVV	1000	Milliseconds

8.5.2 *Sensor-AI-LCS Object Services* — The services provided by the Sensor-AI-LCS are inherited from the Sensor-AI object defined in SEMI E54.1 [3]. There are no additional services required for the Sensor AI-LCS object.

8.5.3 *Sensor-AI-LCS Object Behavior* — The behavior exhibited by the Sensor-AI-LCS object is inherited from the Sensor-AI object defined in SEMI E54.1 [3]. Additional specific behavior associated with the Sensor-AI-LCS object is defined below.

8.5.3.1 *Sensor-AI-LCS OPERATING Application Process* — A reading is retrieved from a physical laser current sensor. This reading may be corrected with a manufacturer-specified algorithm. This corrected reading becomes the input to the gain formula to generate the *value* attribute as referenced in SEMI E54.1 [3]. This process is executed only when the Device Manager Object is in the EXECUTING state as defined in the Common Device Model. When not in the EXECUTING state, the value of the "*Reading Valid*" attribute shall be set to "Invalid". When in one of the sub-states of the EXECUTING state, the validity of the "*Value*" attribute shall be manufacturer specific.

8.6 *Sensor-AI-SLS Object* — The Sensor-AI-SLS (Stray Light Sensor) object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1 [3]. The Sensor-AI-SLS is the device component responsible for retrieving a reading from a physical stray light sensor, optionally correcting the reading with a manufacturer specified algorithm, or algorithms, then making the value available through the "*Value*" attribute.

8.6.1 *Sensor-AI-SLS Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1 [3]. The Sensor-AI object attribute content and its attribute extensions to support the Sensor-AI-SLS object is listed in Table 15.

Table 15 Sensor-AI and Sensor-AI-SLS Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Value	SaiA16	R	Yes	UINT
Gain	SaiA65	R	Yes	REAL
Data Type	SaiA66	R	Yes	USINT
Data Units	SaiA67	R	Yes	Enumerated, UINT milliVolts only value supported
Reading Valid	SlsA1	R	Yes	Enumerated, Byte

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Full Scale	SlsA2	R	Yes	UINT
Alarm Settling Time	SlsA3	R	Yes	UINT
Warning Settling Time	SlsA4	R	Yes	UINT
Reserved	SlsA5–SlsA64	—	—	Reserved for future expansion.
Manufacturer Specified	> SlsA64	—	—	Manufacturer Specific attributes

8.6.1.1 *Value* — The attribute that maintains the stray light sensor value. This value is always read as milliVolts.

8.6.1.2 *Gain* — The attribute that specifies the gain between the photodiode (or some other light detection source) and the stray light reading.

8.6.1.3 *Data Type* — The attribute which defines the format of the data associated with the "value" attribute.

8.6.1.4 *Data Units* — The attribute which specifies the units for the "*Value*" attribute. This attribute is an enumerated UINT and can take only one value:

MilliVolts (as assigned in Appendix 1 of SEMI E54.1).

8.6.1.5 *Reading Valid* — An attribute which specifies whether the "*Value*" attribute contains a valid value. This attribute is an enumerated byte that can take on one of the following values:

0 = Invalid, or
1 = Valid.

8.6.1.6 *Full Scale* — An attribute which specifies the maximum value allowed for the "*Value*" attribute.

8.6.1.7 *Alarm Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before an alarm is generated.

8.6.1.8 *Warning Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before a warning is generated.

8.6.1.9 *Initial and Default Values*

Table 16 Sensor-AI and Sensor-AI-SLS Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value	0	0	
Gain	LVV	1.0	
Data Type	LVV	UINT	
Data Units	LVV	MilliVolts	See Appendix 1 of SEMI E54.1.
Reading Valid	0	0	Invalid
Full Scale	LVV	0XFFFF	
Alarm Settling Time	LVV	1000	Milliseconds
Warning Settling Time	LVV	1000	Milliseconds

8.6.2 *Sensor-AI-SLS Object Services* — The services provided by the Sensor-AI-SLS object are inherited from the Sensor-AI object defined in SEMI E54.1 [3]. There are no additional services required for the Sensor AI-SLS object.

8.6.3 *Sensor-AI-SLS Object Behavior* — The behavior exhibited by the Sensor-AI-SLS object is inherited from the Sensor-AI object defined in SEMI E54.1 [3]. Additional specific behavior associated with the Sensor-AI-SLS object is defined below.

8.6.3.1 *Sensor-AI-SLS OPERATING Application Process* — A reading is retrieved from a physical stray light sensor. This reading may be corrected with a manufacturer-specified algorithm. This corrected reading becomes the input to the gain formula to generate the *value* attribute as referenced in SEMI E54.1 [3]. This process is executed only when the Device Manager Object is in the EXECUTING state as defined in the Common Device Model. When

not in the EXECUTING state, the value of the "*Reading Valid*" attribute shall be set to "Invalid". When in one of the sub-states of the EXECUTING state, the validity of the "*Value*" attribute shall be manufacturer specific.

8.7 *Sensor-AI-MNS Object* — The Sensor-AI-MNS (Median Noise Sensor) object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1 [3]. The Sensor-AI-MNS is the device component responsible for retrieving a reading from a physical median noise sensor, optionally correcting the reading with a manufacturer specified algorithm, or algorithms, then making the value available through the "*Value*" attribute.

8.7.1 *Sensor-AI-MNS Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1 [3]. The Sensor-AI object attribute content and its attribute extensions to support the Sensor-AI-MNS object is listed in Table 17.

Table 17 Sensor-AI and Sensor-AI-MNS Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Value	SaiA16	R	Yes	UINT
Gain	SaiA65	R	Yes	REAL
Data Type	SaiA66	R	Yes	USINT
Data Units	SaiA67	R	Yes	Enumerated, UINT milliVolts only value supported
Reading Valid	MnsA1	R	Yes	Enumerated, Byte
Full Scale	MnsA2	R	Yes	UINT
Alarm Settling Time	MnsA3	R	Yes	UINT
Warning Settling Time	MnsA4	R	Yes	UINT
Reserved	MnsA5–MnsA64	—	—	Reserved for future expansion.
Manufacturer Specified	> MnsA64	—	—	Manufacturer Specific attributes

8.7.1.1 *Value* — The attribute that maintains the median noise sensor value. This is the noise level at the output of the sensor amplifier. The value is calculated by taking the median value of the last manufacturer specific number of rolling samples of the sensor amplifier output. This value is always read as milliVolts.

8.7.1.2 *Gain* — The attribute that specifies the gain of the median noise measuring circuit.

8.7.1.3 *Data Type* — The attribute which defines the format of the data associated with the "value" attribute.

8.7.1.4 *Data Units* — The attribute which specifies the units for the "*Value*" attribute. This attribute is an enumerated UINT and can take only one value:

MilliVolts (as assigned in Appendix 1 of SEMI E54.1).

8.7.1.5 *Reading Valid* — An attribute which specifies whether the "*Value*" attribute contains a valid value. This attribute is an enumerated byte that can take on one of the following values:

0 = Invalid, or
1 = Valid.

8.7.1.6 *Full Scale* — An attribute which specifies the maximum value allowed for the "*Value*" attribute.

8.7.1.7 *Alarm Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before an alarm is generated.

8.7.1.8 *Warning Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before a warning is generated.

8.7.1.9 *Initial and Default Values*

Table 18 Sensor-AI and Sensor-AI-MNS Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value	0	0	
Gain	LVV	1.0	
Data Type	LVV	UINT	
Data Units	LVV	MilliVolts	See Appendix 1 of SEMI E54.1.
Reading Valid	0	0	Invalid
Full Scale	LVV	0XFFFF	
Alarm Settling Time	LVV	1000	Milliseconds
Warning Settling Time	LVV	1000	Milliseconds

8.7.2 *Sensor-AI-MNS Object Services* — The services provided by the Sensor-AI-MNS object are inherited from the Sensor-AI object defined in SEMI E54.1 [3]. There are no additional services required for the Sensor AI-MNS object.

8.7.3 *Sensor-AI-MNS Object Behavior* — The behavior exhibited by the Sensor-AI-MNS object is inherited from the Sensor-AI object defined in SEMI E54.1 [3]. Additional specific behavior associated with the Sensor-AI-MNS object is defined below.

8.7.3.1 *Sensor-AI-MNS OPERATING Application Process* — A reading is retrieved from the physical median noise sensor. This reading is maintained in a rolling count of manufacturer specified samples and then the median value of the manufacturer specified samples is calculated. This median reading becomes the input to the gain formula to generate the *value* attribute as referenced in SEMI E54.1 [3]. This process is executed only when the Device Manager Object is in the EXECUTING state as defined in SEMI E54.1. When not in the EXECUTING state, the value of the "*Reading Valid*" attribute shall be set to "Invalid". When in one of the sub-states of the EXECUTING state, the validity of the "*Value*" attribute shall be manufacturer specific.

8.8 *Sensor-AI-Counter Object* — The Sensor-AI-Counter object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1 [3]. The Sensor-AI-Counter is the device component responsible for maintaining a count of the number of particles deposited in each bin and then making the value available through the "*Value*" attribute.

8.8.1 *Sensor-AI-Counter Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1 [3]. The Sensor-AI object attribute content and its attribute extensions to support the Sensor-AI-Counter object is listed in Table 19.

Table 19 Sensor-AI and Sensor-AI-Counter Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Value	SaiA16	R	Yes	UDINT
Data Type	SaiA66	R	Yes	USINT
Data Units	SaiA67	R	Yes	Enumerated, UINT
Reading Valid	CounterA1	R	Yes	Enumerated, Byte
Full Scale	CounterA2	R	Yes	UDINT
Alarm Settling Time	CounterA3	R	Yes	UINT
Warning Settling Time	CounterA4	R	Yes	UINT
Upper Size	CounterA5	RW (See NOTE 1.)	Yes	REAL
Lower Size	CounterA6	RW (See NOTE 1.)	Yes	REAL
Reserved	CounterA7–CounterA64	—	—	Reserved for future expansion.
Manufacturer Specified	> CounterA64	—	—	Manufacturer Specific attributes

NOTE 1: Consult the manufacturer's specification for specific "Write" capabilities of this attribute.

8.8.1.1 *Value* — The attribute that maintains the particle count in the bin.

8.8.1.2 *Data Type* — The attribute that defines the format of the data associated with the "*Value*" attribute.

8.8.1.3 *Data Units* — The attribute which specifies the units for the "*Value*" attribute. This attribute is an enumerated UINT that can take on one of the following values: raw counts, count per second, counts per milliliter, and counts per gallon. (See Appendix 1 of SEMI E54.1.)

8.8.1.4 *Reading Valid* — An attribute which specifies whether the "*Value*" attribute contains a valid value. This attribute is an enumerated byte that can take on one of the following values:

0 = Invalid, or
1 = Valid.

8.8.1.5 *Full Scale* — An attribute which specifies the maximum value allowed for the "*Value*" attribute.

8.8.1.6 *Alarm Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before an alarm is generated.

8.8.1.7 *Warning Settling Time* — An attribute which specifies the maximum time in milliseconds that the "*Value*" attribute may take to stabilize before a warning is generated.

8.8.1.8 *Upper Size* — An attribute which specifies the upper bound of the particle size, in microns, included in the bin counts.

8.8.1.9 *Lower Size* — An attribute which specifies the lower bound of particle size, in microns, included in the bin counts.

8.8.1.10 *Initial and Default Values*

Table 20 Sensor-AI and Sensor-AI-Counter Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value	0	0	
Data Type	LVV	UDINT	
Data Units	LVV	0	Raw counts
Reading Valid	0	0	Invalid
Full Scale	LVV	0XFFFFFFFF	
Alarm Settling Time	LVV	1000	Milliseconds
Warning Settling Time	LVV	1000	Milliseconds
Upper Size	LVV	Manufacturer Specified	Manufacturer Specific
Lower Size	LVV	Manufacturer Specified	Manufacturer Specific

8.8.2 *Sensor-AI-Counter Object Services* — The services provided by the Sensor-AI-Counter object are defined in SEMI E54.1 [1]. There are no additional services required for the Sensor AI-Counter object.

8.8.3 *Sensor-AI-Counter Object Behavior* — The behavior exhibited by the Sensor-AI-Counter object is inherited from the Sensor-AI object defined in SEMI E54.1. Additional specific behavior associated with the Sensor-AI-Counter object is defined below.

8.8.3.1 *Sensor-AI-Counter OPERATING Application Process*

8.8.3.1.1 A reading is retrieved from the physical median noise sensor. This reading is filtered through an analysis of a manufacturer specified number of samples. The *value* attribute is set to this processed reading. This process is called "counting" and is executed only when the Device Manager Object is in the EXECUTING state as defined in the Common Device Model.

8.8.3.1.2 Whenever an Interlock alarm, Sensor Not Detected alarm, Leak Detected alarm, and Sensor Type Change alarm is active the ISPM device shall stop "counting" until the alarm clears (see Section 8.2).

8.8.3.1.3 The ISPM is a device that measures and counts particles. As part of counting, particles are classified by size and count and accumulated in bins over time intervals then reported. The number of bins varies by vendor and device model.

8.8.3.1.4 The *value* attributes for all Sensor-AI-Counter objects, shall be held at zero unless the device laser is on and stable (i.e., Laser Status attribute of Device Manager Object has a value indicating "LASER ON AND STABLE"). The *value* attribute of any Sensor-AI-Counter object may be cleared (to zero) in a counter sensor after being read; the conditions under which this clearing behavior occurs may be further specified in the associated Network Communication Standard or by the manufacturer. The *value* shall also be cleared (to zero) by a valid "Clear Counts" service from the SAC object or the "Reset" service issued to the Sensor AI Class Object.

8.8.3.1.5 When attempting to set the "Upper Size" attribute to a value above the capability of the sensor, or to a value equal to or below the current "Lower Size" attribute value, an error response shall be generated indicating an invalid operation has been attempted. Similarly, an attempt to set the "Lower Size" attribute to a value below the capability of the sensor, or to a value equal to or above the current "Upper Size" attribute value, an error response shall be generated indicating an invalid operation has been attempted.

8.9 *Assembly-ISPM Objects* — Assembly-ISPM objects inherit attributes, services, and behavior from the Assembly object. The Assembly object is the device component which provides a mechanism of grouping more than one attribute from one or more objects into a single data structure for communication over the network.

8.9.1 Table 21 identifies the Assembly-ISPM objects defined for the ISPM device.

Table 21 Assembly List

Object	*Access Network*	*Required*	*Form*
Assembly-ISPM#1	R	Yes	Status and Counters (Counters ≤ 1024); Default Assembly
Assembly-ISPM#2	R	No	Status, Count and Counters (Counters ≤ 1024)
Assembly-ISPM#3	R	No	Status Diagnostics, and Counters (Counters ≤ 1024)
Assembly-ISPM#4	R	No	Status, Tool State, Time Stamp, Duration, and Counters (Counters ≤ 1024)
Assembly-ISPM#5	R	No	Status, Tool State, Time Stamp, Duration, Diagnostics, and Counters (Counters ≤ 1024)
Assembly-ISPM#6	R	Yes	Status
Assembly-ISPM#7	R	No	Exception Detail Alarm
Assembly-ISPM#8	R	No	Exception Detail Warning
Assembly-ISPM#9	R	No	Exception Detail Alarm and Exception Detail Warning
Assembly-ISPM#40	RW	No	Device Configuration

8.9.2 *Assembly-ISPM Objects Attributes* — Table 22 provides a list of attributes common to all Assembly-ISPM object types.

Table 22 Assembly-ISPM Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Data (See NOTE 1.)	A1	RW	Yes	Structure as defined below

NOTE 1: Inherited from the Assembly object as shown in Figure 1.

8.9.2.1 *Data Attribute Format for Assembly-ISPM Objects* — The Data attribute of all Assembly-ISPM objects is a structured attribute containing an ordered list of attributes within its structure. In the following tables (23 through 31), the structure of the Data attribute for each of the Assembly-ISPM object types is defined.

Table 23 Assembly-ISPM #1 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #1 Value
3	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #2 Value
.	ISPMD16	SaiA16	Sensor-AI-Counter #3 to Sensor-AI-Counter-ISPM #N-1 Value
N≤1024 (See NOTE 1.)	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #N Value

NOTE 1: "N" is the value of the Sensor-AI-Counter object attribute "Number of Bins". Note also that the number of bins reported in an assembly is fixed throughout the life of the device, and is specified by the manufacturer.

Table 24 Assembly-ISPM #2 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	ISPMD2	SacA65	SAC Number of Bins
3	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #1 Value
4	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #2 Value
.	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #3 to Sensor-AI-Counter-ISPM #N-1 Value
N≤1024 (See NOTE 1.)	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #N Value

NOTE 1: "N" is the value of the Sensor-AI-Counter object attribute "Number of Bins". Note also that the number of bins reported in an assembly is fixed throughout the life of the device, and is specified by the manufacturer.

Table 25 Assembly-ISPM #3 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	ISPMD3	SaiA16	Sensor-AI-LCS Value (Laser Current Diagnostic)
3	ISPMD4	SaiA16	Sensor-AI-SLS Value (Stray Light Diagnostic)
4	ISPMD5	SaiA16	Sensor-AI-MNS Value (Median Noise Diagnostic)
5	–	–	Reserved (2 bytes)
6	ISPMD2	SacA65	SAC Number of Bins
7	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #1 Value
8	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #2 Value
.	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #3 to Sensor-AI-Counter-ISPM #N-1 Value

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
N ≤ 1024 (See NOTE 1.)	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #N Value

NOTE 1: "N" is the value of the Sensor-AI-Counter object attribute "Number of Bins". Note also that the number of bins reported in an assembly is fixed throughout the life of the device, and is specified by the manufacturer.

Table 26 Assembly-ISPM #4 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	DM1	A35	Device Manager Tool State
3	DM1	A41	Device Manager Time Stamp
4	ISPMD2	SacA67	SAC Duration
6	ISPMD2	SacA65	Sensor-AI-Counter Number of Bins
7	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #1 Value
8	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #2 Value
.	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #3 to Sensor-AI-Counter-ISPM #N-1 Value
N ≤ 1024 (See NOTE 1.)	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #N Value

NOTE 1: "N" is the value of the Sensor-AI-Counter object attribute "Number of Bins". Note also that the number of bins reported in an assembly is fixed throughout the life of the device, and is specified by the manufacturer.

Table 27 Assembly-ISPM #5 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	DM1	A35	Device Manager Tool State
3	DM1	A41	Device Manager Time Stamp
4	ISPMD2	SacA67	SAC Duration
5	ISPMD3	SaiA16	Sensor-AI-LCS Value (Laser Current Diagnostic)
6	ISPMD4	SaiA16	Sensor-AI-SLS Value (Stray Light Diagnostic)
7	ISPMD5	SaiA16	Sensor-AI-MNS Value (Median Noise Diagnostic)
8	–	–	Reserved (2 bytes)
9	ISPMD2	SacA65	SAC Number of Bins
10	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #1 Value
11	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #2 Value

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
.	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #3 to Sensor-AI-Counter-ISPM #N-1 Value
N ≤ 1024*	ISPMD16	SaiA16	Sensor-AI-Counter-ISPM #N Value

NOTE 1: "N" is the value of the Sensor-AI-Counter object attribute "Number of Bins". Note also that the number of bins reported in an assembly is fixed throughout the life of the device, and is specified by the manufacturer.

Table 28 Assembly-ISPM #6 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status

Table 29 Assembly-ISPM #7 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	DM1	A13	Device Manager Exception Detail Alarm

Table 30 Assembly-ISPM #8 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	DM1	A14	Device Manager Exception Detail Warning

Table 31 Assembly-ISPM #9 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A12	Device Manager Exception Status
2	DM1	A13	Device Manager Exception Detail Alarm
3	DM1	A14	Device Manager Exception Detail Warning

Table 32 Assembly-ISPM #40 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM1	A33	Device Manager Gain
2	DM1	A13	Device Manager Filter Bandwidth
3	ISPMD2	SacA65	SAC Number of Bins
4	ISPMD3	SaiA73	Sensor-AI-LCS Alarm Trip Point High (See NOTE 1.)
5	ISPMD3	SaiA76	Sensor-AI-LCS

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
			Warning Trip Point High (See NOTE 1.)
6	ISPMD3	SaiA74	Sensor-AI-LCS Alarm Trip Point Low (See NOTE 1.)
7	ISPMD3	SaiA77	Sensor-AI-LCS Warning Trip Point Low (See NOTE 1.)
8	ISPMD4	SaiA73	Sensor-AI-SLS Alarm Trip Point High (See NOTE 1.)
9	ISPMD4	SaiA76	Sensor-AI-SLS Warning Trip Point High (See NOTE 1.)
10	ISPMD4	SaiA74	Sensor-AI-SLS Alarm Trip Point Low (See NOTE 1.)
11	ISPMD4	SaiA77	Sensor-AI-SLS Warning Trip Point Low (See NOTE 1.)
12	ISPMD5	SaiA73	Sensor-AI-MNS Alarm Trip Point High (See NOTE 1.)
13	ISPMD5	SaiA76	Sensor-AI-MNS Warning Trip Point High (See NOTE 1.)
14	ISPMD5	SaiA74	Sensor-AI-MNS Alarm Trip Point Low (See NOTE 1.)
15	ISPMD5	SaiA77	Sensor-AI-MNS Warning Trip Point Low (See NOTE 1.)

NOTE 1: Objects to which these attributes are linked are optional. If the associated object is not supported in the ISPM device, the data field for the associated attribute in this assembly is set to zero and should be interpreted as having no meaning.

SEMI E54.11-1105
SPECIFIC DEVICE MODEL FOR ENDPOINT DEVICES

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 September 8, 2005. It was available at www.semi.org in October 2005 and on CD-ROM in November 2005. Originally published in 2001.

1 Purpose

1.1 This specification is part of a suite of standards that specify the implementation of SEMI standards for the Sensor/Actuator Network. The specific purpose of this specification is to describe a network independent application model comprised of device objects that are common to all Endpoint Devices on a semiconductor equipment Sensor/Actuator communication network.

2 Scope

2.1 An Endpoint Device (EPD) is a device that measures and monitors process characteristics to determine when a specific threshold or event has been obtained usually to signal the completion of a process or process step. These endpoint devices are, but not limited to, devices that may classify a process endpoint by determining the size and count of particles in the process environment, detecting and determining optical light from a sample region of the environment's space, or determining motor current of an equipment component.

2.2 This specification specifically addresses the minimum attributes, services and behavior an Endpoint Device (EPD) must support to be interoperable on the Sensor/Actuator Network.

2.3 This specification is intended to ensure a high-degree of device interoperability on the Sensor/Actuator Network, while still allowing flexibility for product differentiation and technology evolution.

2.4 The model specified in this specification is used in conjunction with the Sensor/Actuator Network Common Device Model (CDM) to completely describe the Endpoint Device (EPD) as it appears from the network interface.

2.5 This specification, together with the Sensor/Actuator Network Standard, the Sensor/Actuator Network Common Device Model, and a Sensor/Actuator Network Communication Specification, form a complete interoperability specification for the EPD.

2.6 To comply with this specification, a device must implement and support, at a minimum, the required attributes, services, and behavior identified in these documents. Support for optional attributes, services and behavior is not required to be compliant to this specification. Optional attributes, services, and behavior are specified in these documents to promote further device interoperability as features evolve and are adopted by more manufacturers. If optional attributes, services, and behavior are implemented for this device they must be implemented as identified in this document.

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 Limitations

3.1 This specification is a companion to a suite of specifications that together make up the Sensor/Actuator Network Communication standard. Therefore, using portions of this specification that relate to network communications necessarily requires an understanding of the associated network specification.

3.2 As this document is a specification for the Endpoint Device Model, it does not contain any definition of objects, attributes, services, or behavioral descriptions that are already defined in the Sensor/Actuator Network Common Device Model (CDM). Additional attributes, attribute assignments, services and/or service parameters that are Endpoint Device specific and/or implementation specific are contained in this specification.

3.3 While this specification is sufficient to completely describe the EPD as it appears from the network, it does not fully describe behavior of a specific endpoint device type which is not visible from the network. This allows

flexibility in implementation techniques and product differentiation between manufacturers. Manufacturer specific objects may be defined by the manufacturer but are, by definition, outside the scope of this standard.

3.4 This specification is compatible, but not compliant with SEMI E39. This means that although this specification 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 specification are compatible with those of SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and thus are not applied to the Endpoint Device Model.

3.5 Operation over the entire range specified for an attribute within a specific object is not a requisite for compliance with this specification.

4 Referenced Standards and Documents

4.1 *SEMI Standards*

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

SEMI E54 — Sensor/Actuator Network Standard

SEMI E54.1— Standard for Sensor/Actuator Network Common Device Model

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *EP* — an EPD sensor named Endpoint.

5.2 *Definitions*

5.2.1 *Endpoint Device (EPD)* — a self-contained device, consisting of device specific signal-processing electronics, which is capable of monitoring and measuring the occurrence of a process endpoint.

5.2.2 *Endpoint Detection Event* — consists of the device operation of monitoring, measuring, analyzing, waiting, and reporting endpoint.

5.3 This document inherits the terminology defined by SEMI E54.1

5.3.1 *Attribute*

5.3.2 *Behavior*

5.3.3 *Boolean (BOOL)*

5.3.4 *Byte*

5.3.5 *Character*

5.3.6 *Common Device Model (CDM)*

5.3.7 *Data Type*

5.3.8 *Data Units*

5.3.9 *Device*

5.3.10 *Device Manager (DM) Object*

5.3.11 *Device Model*

5.3.12 *Double Integer (DINT)*

5.3.13 *Enumerated Byte*

5.3.14 *Full Scale Range*

5.3.15 *Instance*

5.3.16 *Last Valid Value (LVV)*

5.3.17 *Long Integer (LINT)*

5.3.18 *Long Real (LREAL)*

5.3.19 *Manufacturer*

5.3.20 *Nibble*

5.3.21 *Null Character*

5.3.22 *Object*

5.3.23 *Real (REAL)*

5.3.24 *S, A, and C Objects*

5.3.25 *Sensor Actuator Controller (SAC) Object*

5.3.26 *Service*

5.3.27 *Signed Integer (INT)*

5.3.28 *Short Integer (SINT)*

5.3.29 *State Diagram*

5.3.30 *Test String*

5.3.31 *Unsigned Double Integer (UDINT)*

5.3.32 *Unsigned Double Long Integer (UDLINT)*

5.3.33 *Unsigned Integer (UINT)*

5.3.34 *Unsigned Long Integer (ULINT)*

5.3.35 *Unsigned Short Integer (USINT)*

6 Requirements and Specifications

6.1 In order to implement this standard in an Endpoint Device, it is necessary to also implement SEMI E54.1 and one of the Sensor/Actuator Network Communication Standards (SEMI E54.4–E54.9). See §3 for more information on a complete interoperability standard.

6.2 This specification also requires the implementation of a Date_And_Time data structure used to represent the current device Date and Time. Table 1 defines the format of the Date_And_Time data type.

Table 1 Date_And_Time Format

Data Item	*Description*	*Range*
1	Number of milliseconds since midnight	Unsigned Double Integer (UDINT)
2	Number of days since 1/1/72	Unsigned Integer (UINT)

7 Conventions

7.1 This document embraces the conventions and notations stated in §6 of SEMI E54.1.

8 Device High Level Structure

8.1 *General Description*

8.1.1 The high-level object view of an Endpoint Device (EPD) profile is shown in Figure 1.

8.1.2 Note that the profile for an "EPD Device" object is depicted in Figure 1 only for the purposes of illustrating a high level view of the device and its component objects. In the context of this document, this object is not addressable, does not have addressable attributes, does not have accessible services and, does not exhibit any defined behavior.

8.1.3 In the remainder of this section, this document defines in detail the component objects unique to the EPD device. References, rather than definitions, are included for the DM, the SAC and other objects defined in SEMI E54.1.

8.1.4 Many of the objects defined in this document inherit properties from other objects. The properties inherited include attribute, service and behavior definitions. These other objects are specified here or in SEMI E54.1.

8.1.5 This document provides for future extensions as well as manufacturer specific enhancements by reserving object attribute identifiers and object service identifiers. Specifically all object definitions in this document specify or reserve the first 64 attribute identifiers (A1 through A64) and the first 64 service identifiers (S1 through S64) allowing manufacturers to specify identifiers beyond these ranges. Additionally, byte enumerated attributes are specified or reserved from 0 to 63 allowing manufacturers to specify an enumeration beyond this range (64 to 255).

8.1.6 *Endpoint Device (EPD) Description* — An Endpoint device profile is composed of the component objects and object relationships shown in Figure 1.

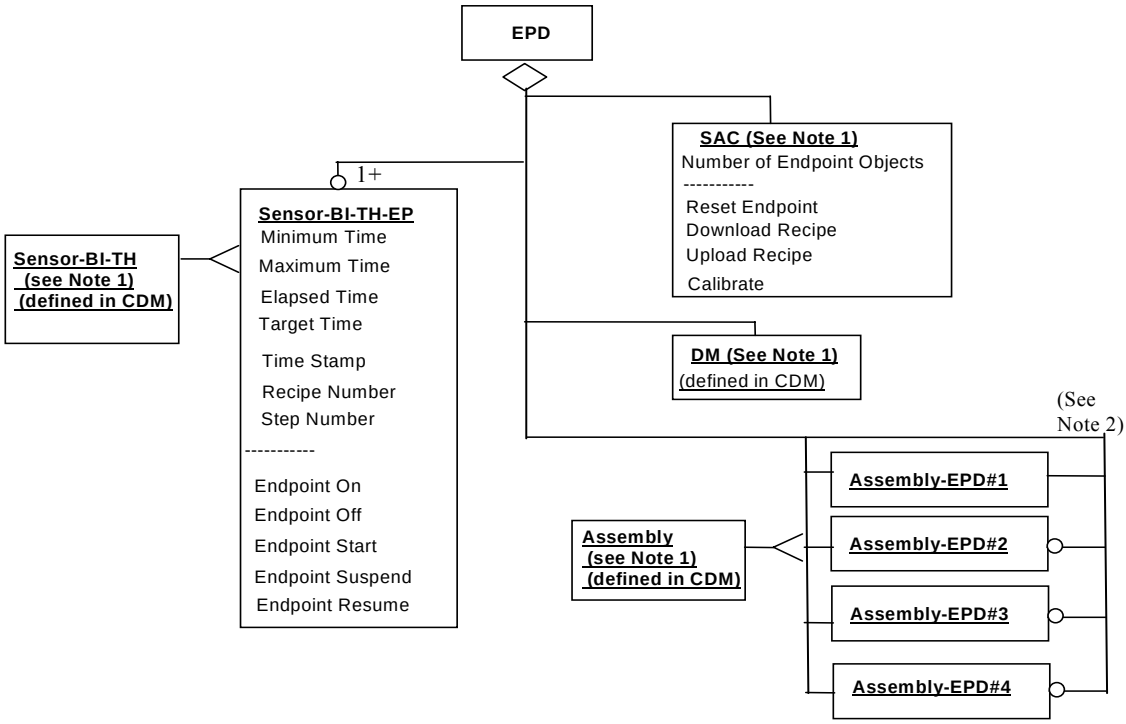


Figure 1
Endpoint Device High Level Structure

NOTE 1: The Sensor-BI-TH, DM, SAC, and Assembly are defined in the CDM. Additional attributes and services are added to support the EPD.

NOTE 2: Assembly-EPD #1 object is required. Other Assembly-EPD objects are optional.

8.1.7 *General Requirements*

8.1.7.1 *Device Objects* — All objects are defined in terms of their object name and Class/Object identifier. Identifiers for all objects described in this document are summarized in Table 2.

Table 2 Endpoint Device Objects

Referenced Document Section	*Object Name*	*Class/Object Identifier*	*Minimum #*	*Maximum #*
8.2	Device Manager (DM)	EPD1	1	1
8.3	Sensor Actuator Controller (SAC)	EPD2	1	1
8.5	Sensor-BI-TH-EP	EPD3	1	1024
8.6	Assembly-EPD#1	EPD4	1	1
8.6	Assembly-EPD#2	EPD5	0	1
8.6	Assembly-EPD#3	EPD6	0	1
8.6	Assembly-EPD#4	EPD7	0	1
—	Reserved	EPD7 – EPD63	-	—
—	Manufacturer Specified	>EPD64	-	—

8.1.7.2 *Object Services* — Not all object services listed in this document can necessarily be requested over the network. They are included in this document because their behavior may generate network activity.

8.1.7.3 *Object Behavior* — A network specific service error response is generated for all service requests received over the network that are not supported by the object, or contain a parameter value which is beyond the supported range, or which is otherwise invalid.

8.2 *Device Manager Object (DM)* — The Device Manager object is the device component responsible for managing and consolidating the device operation as specified in SEMI E54.1. The following sections specify the components of the DM object that are not specified in the Common Device Model or require further definition than specified in SEMI E54.1.

8.2.1 *Device Manager Object Attributes* — Required and optional DM object attributes are listed in Table 3.

Table 3 DM Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	A1	R	Yes	Refer to SEMI E54.1.
Exception Detail Alarm	A13	R	No	Refer to SEMI E54.1.
Exception Detail Warning	A14	R	No	Refer to SEMI E54.1.
Reserved	A33-A64	—	—	Reserved for SDM future expansion
Manufacturer Specified	>A64	—	—	Manufacturer Specific attributes

8.2.1.1 *Device Type* — An attribute that uniquely identifies the type of the device on the network. The device type attribute is assigned as follows:

Endpoint Device = “EPD”

If the “Endpoint Device” functionality is implemented by another Sensor/Actuator Network Specific Device Model, the ‘*Device Type*’ attribute value may be specified by the manufacturer to identify the other Specific Device Model.

8.2.1.2 *Exception Detail Alarm (Optional)* — An attribute that identifies the detailed alarm status of the device. Table 4 defines the bit assignments associated with the alarm exception detail.

Table 4 Exception Detail Alarm Bit Assignments

Bit	*Device Specific Alarm[0]*
0	Unexpected Conditions Detected
1	Reserved
2	Sensor Not Detected
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Endpoint Failure

8.2.1.3 *Exception Detail Warning (Optional)* — An attribute that identifies the detailed warning status of the device. Table 5 defines the bit assignments associated with the warning exception detail.

Table 5 Exception Detail Warning Bit Assignments

Bit	*Device Specific Warning[0]*
0	Unexpected Conditions Detected
1	Reserved
2	0
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Endpoint Warning

8.2.1.4 *Manufacturer Exception Detail Size (Optional)* — An attribute that specifies the number of exception detail bytes included in the alarm or warning details.

8.2.1.5 *Initial and Default Values*

Table 6 DM Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Device Type	EPD	EPD	EPD = Endpoint Device
Exception Detail Alarm	0	0	
Exception Detail Warning	0	0	

8.2.2 *Device Manager Object Services* — The services provided by the Device Manager object are defined in SEMI E54.1. There are no additional services required for the Device Manager object.

8.2.3 *Device Manager Object Behavior* — The behavior exhibited by the Device Manager object is defined in SEMI E54.1. There is no additional behavior specified for the Device Manager object.

8.3 *Sensor Actuator Controller Object (SAC)* — The Sensor Actuator Controller object is the device component responsible for coordinating the interaction of the EPD device with the sensory/actuation/control environment as specified in SEMI E54.1. The following sections specify the components of the SAC object that are not specified in the Common Device Model or require further definition than specified in the Common Device Model.

8.3.1 *Sensor Actuator Controller Object Attributes* — The attributes provided by the Sensor Actuator Controller object are defined in SEMI E54.1. Table 7 contains the additional attributes required for the Sensor Actuator Controller object.

Table 7 SAC Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Number of Endpoint Objects	SacA65	R	No	UINT

8.3.1.1 *Number of Endpoint Objects* — An attribute that specifies the number of endpoint objects in the device. This value represents the number of Sensor-BI-TH-EP objects in the device. If this attribute is not supported then the default value of 1 Endpoint object is supported.

8.3.1.2 *Initial and Default Values*

Table 8 SAC Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Number of Endpoint Objects	Manufacturer Specified	Manufacturer Specified	There must be at least one Endpoint object.

8.3.2 *Sensor Actuator Controller Object Services* — The services provided by the Sensor Actuator Controller object are defined in SEMI E54.1. Table 9 contains the additional services required for the Sensor Actuator Controller object.

Table 9 SAC Object Services

Service	*Service Identifier*	*Type*	*Description*
Reset Endpoint	S33	R	Used to 'Reset' all of the Sensor-BI-TH-EP sensor objects.
Download Recipe	S34	R	Used to download the Sensor-BI-TH-EP sensor objects with manufacturer specific recipe data.
Upload Recipe	S35	R	Used to upload, from the device, manufacturer specific recipe data.
Calibrate	S36	R	Used to execute a device manufacture specific calibration procedure.
Reserved	S37–S64	—	Reserved for future expansion
Manufacturer Specified	>S64	—	Manufacturer Specific services

8.3.2.1 *Reset Endpoint (Optional)* — This service is used to instruct all of the Sensor-BI-TH-EP objects to perform a 'Reset' of their endpoint monitoring and measuring events. A one-time reset of their respective Sensor-BI-TH-EP attributes is performed. There are no parameters required for this service.

8.3.2.2 *Download Recipe (Optional)* — This service is used to set the recipe parameters associated with the endpoint sensor or a specific Sensor-BI-TH-EP object. The format and type of data comprising recipe data and the mechanism implemented to interrupt recipe data and distribute its contents to the appropriate Sensor-BI-TH-EP object is manufacturer specific. The parameter 'Recipe Data' may be a formatted data structure or a list of individual data items. The following table describes the parameters specified for this service.

Table 10 Download Recipe Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Recipe Data #1	M	U	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Recipe Data #2	M	U	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Recipe Data #3 through Recipe Data #N-1	M	U	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Recipe Data #N	M	U	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.

8.3.2.3 *Upload Recipe (Optional)* — This service is used to read the recipe parameters associated with the endpoint sensor or a specific Sensor-BI-TH-EP object. The format and type of data comprising recipe data and the mechanism implemented to assemble recipe data to be read is manufacturer specific. The parameter 'Recipe Data' may be a formatted data structure or a list of individual data items. The following table describes the parameters specified for this service.

Table 11 Upload Recipe Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Recipe Data #1	M	M	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Recipe Data #2	M	M	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Recipe Data #3 through Recipe Data #N-1	M	M	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Recipe Data #N	M	M	Manufacturer Specific	The recipe parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.

8.3.2.4 *Calibrate (Optional)* — This service is used to set the calibration parameters associated with the endpoint sensor or a specific Sensor-BI-TH-EP object. The format and type of data comprising calibration data and the mechanism implemented to interrupt calibration data and execute a calibration algorithm is manufacturer specific. The parameter 'Calibration Data' may be a formatted data structure or a list of individual data items. The following table describes the parameters specified for this service.

Table 12 Calibration Service Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Calibration Data #1	C	C	Manufacturer Specific	The calibration parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Calibration Data #2	C	C	Manufacturer Specific	The calibration parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.
Calibration Data #3 through Calibration Data #N-1	C	C	Manufacturer Specific	The Calibration parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Calibration Data #N	C	C	Manufacturer Specific	The calibration parameter format is manufacturer specific. The implementation behavior required to support this service is also manufacturer specific.

8.3.3 *Sensor Actuator Controller Object Behavior*

8.3.3.1 The behavior exhibited by the Sensor Actuator Controller object is defined in SEMI E54.1. Additional behavior is detailed below.

8.3.3.2 The "*Reset Endpoint"* service will issue an "Endpoint Restart" object service to each Sensor-BI-TH-EP sensor object.

8.4 *Sensor-BI-TH Object* — The Sensor-BI object is the device component responsible for coordinating the behavior common to all Boolean input threshold sensor elements in the EPD device as specified in SEMI E54.1.

8.4.1 *Sensor-BI-TH Object Attributes* — The attributes provided by the Sensor-BI-TH object are defined in SEMI E54.1. There are no additional attributes required for the Sensor-BI-TH object.

8.4.2 *Sensor-BI-TH Object Services* — The services provided by the Sensor-BI-TH object are defined in SEMI E54.1. There are no additional services required for the Sensor-BI-TH object.

8.4.3 *Sensor-BI-TH Object Behavior* — The behavior exhibited by the Sensor-BI-TH object is defined in SEMI E54.1. There is no additional behavior required for the Sensor-BI-TH object.

8.5 *Sensor-BI-TH-EP Object* — The Sensor-BI-TH-EP (Endpoint) object inherits the attributes, services, and behavior of the Sensor-BI-TH as defined in SEMI E54.1. The Sensor-BI-TH-EP is the device component responsible for retrieving a reading, or readings, from the device specific signal-processing sensors, optionally processing the readings with a manufacturer specified algorithm, or algorithms, and then making the endpoint result available through the "*Value*" attribute.

8.5.1 *Sensor-BI-TH-EP Object Attributes* — The attributes provided by the Sensor-BI-TH object are defined in SEMI E54.1. The Sensor-BI-TH object attribute content and its attribute extensions to support the Sensor-BI-TH-EP object are listed in the table below.

Table 13 Sensor-BI, Sensor-BI-TH, and Sensor-BI-TH-EP Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Value	SbiA16	R	Yes	BOOL
Reading Valid	SbithA64	R	Yes	BOOL
State	SbithA65	R	No	Enumerated, Byte
Status	SbithA66	R	No	Enumerated, Byte
Minimum Time	EpA1	RW*	No	UDINT
Maximum Time	EpA2	RW*	No	UDINT
Target Time	EpA3	RW*	No	UDINT
Elapsed Time	EpA4	R	No	UDINT
Time Stamp	EpA5	R	No	Date_And_Time
Recipe Identifier	EpA6	RW*	No	Text String
Step Identifier	EpA7	RW*	No	Text String
Reserved	EpA8 –EpA64	—	—	Reserved for future expansion
Manufacturer Specified	> EpA64	—	—	Manufacturer Specific attributes

[#1] "*" Indicates that the specific attribute is nonvolatile. Nonvolatile requires that the current attribute value be maintained through a component power cycle.

8.5.1.1 *Value* — The attribute that maintains the current endpoint event result. The value of the *'Value'* attribute is read as a Boolean (True or False) endpoint detection event result.

8.5.1.2 *Reading Valid* — An attribute which specifies whether the "*Value*" attribute contains a valid value. This attribute is Boolean that can take on one of the following values:

0 = INVALID
1 = VALID

The 'Reading Valid' attribute is identified as an optional attribute of the SBITH object class but is identified as a required attribute for the SBITHEP object class.

8.5.1.3 *State* — An attribute that records the current state of the endpoint object. This attribute is an enumerated byte. The possible enumeration and the requirement for support are as follows:

0 = ENDPOINT OFF (required)
1 = ENDPOINT IN PROCESS (required)
2 = ENDPOINT IDLE (optional)
3 = ENDPOINT SUSPENDED (optional)
4 = ENDPOINT FAILURE (optional)
5-63 = Reserved
64-255 = Manufacturer Specified (optional)

8.5.1.4 *Status* — An attribute which specifies whether endpoint detection event reporting is active based upon the services 'Endpoint On', 'Endpoint Start', and 'Endpoint Off or On'. This attribute is an enumerated byte that can take on one of the following values:

0 = Endpoint Off
1 = Endpoint On

8.5.1.5 *Minimum Time* — An attribute that specifies the minimum time in milliseconds for an endpoint detection event. No attempt to report an endpoint detection event will take place until the minimum time specified has expired.

8.5.1.6 *Maximum Time* — An attribute that specifies the maximum time in milliseconds for an Endpoint detection event before an alarm is reported.

8.5.1.7 *Target Time* — An attribute that specifies the expected time in milliseconds for an Endpoint detection event.

8.5.1.8 *Elapsed Time* — An attribute that specifies in milliseconds the amount of time that has elapsed since the beginning of the current endpoint detection event. This attribute will behave as a count up timer that is frozen when the endpoint event is detected.

8.5.1.9 *Time Stamp* — An attribute that specifies the time when the endpoint detection event completed.

8.5.1.10 *Recipe Identifier* — An attribute that specifies a manufacturer specific endpoint algorithm or algorithms to be utilized to determine the current endpoint detection event. The interpretation of this attribute is manufacturer specific.

8.5.1.11 *Step Identifier* — An attribute that specifies a process recipe step that is associated with the current endpoint detection recipe and/or event. The interpretation of this attribute is manufacturer specific.

8.5.1.12 *Initial and Default Values*

Table 14 Sensor-BI and Sensor-BI-EP Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value	FALSE	FALSE	
Reading Valid	0	0	Invalid Reading
State	LVV	0	ENDPOINT OFF
Status	0	Endpoint Off	
Minimum Time	LVV	0	Milliseconds
Maximum Time	LVV	0	Milliseconds
Target Time	LVV	0	Milliseconds

Attribute	*Initial Value*	*Default Value*	*Comment*
Elapsed Time	LVV	0	Milliseconds
Time Stamp	LVV	Manufacture Specified	Date_And_Time
Recipe Identifier	LVV	Manufacture Specified	
Step Identifier	LVV	Manufacturer Specified	

8.5.2 *Sensor-BI-EP Object Services* — The services provided by the Sensor-BI-EP are inherited from the Sensor-BI object defined in SEMI E54.1. The Sensor-BI-EP object supports the additional services listed in Table 15 below.

Table 15 Sensor-BI-EP Object Services

Service	*Service Identifier*	*Type*	*Description*
Endpoint On	S1	R	Used to prompt the endpoint object to go from the ENDPOINT OFF state to ENDPOINT ON/ENDPOINT IDLE state as defined by the State and Status attribute.
Endpoint Off	S2	R	Used to prompt the endpoint object to go immediately to the ENDPOINT OFF state as defined by the State and Status attribute. All endpoint detection processes are aborted and all active timers are stopped.
Endpoint Start	S3	R	Used to prompt the endpoint object to go from the ENDPOINT OFF or ENDPOINT IDLE (optional) state to the ENDPOINT IN PROCESS state and begin the endpoint detection event process.
Endpoint Suspend	S4	R	Used to prompt the endpoint object to go to the ENDPOINT SUSPENDED state from the ENDPOINT IN PROCESS state. All endpoint detection event processes and active timers are suspended.
Endpoint Resume	S5	R	Used to prompt the endpoint object to go from the ENDPOINT SUSPENDED state to the ENDPOINT IN PROCESS state. All endpoint detection event processes and active timers resume from where they were suspended.
Reserved	S6–S64	—	Reserved for future expansion.
Manufacturer Specified	> S64	—	Manufacturer Specific services.

8.5.2.1 *Endpoint On (Required)* — This service is used to prompt the endpoint object to go from the ENDPOINT OFF state to the ENDPOINT ON / ENDPOINT IDLE state as defined by the State attribute. If the device does not support the ENDPOINT IDLE state then the endpoint object goes immediately to the ENDPOINT ON/ENDPOINT IN PROCESS state. If the State attribute is already set to ENDPOINT IDLE or ENDPOINT IN PROCESS, the endpoint object is set to the appropriate state. If the device turns on successfully, a "success" response is returned. If the device fails to turn on properly, a "fail" response is returned and the endpoint object remains in the appropriate state.

8.5.2.2 *Endpoint Off (Required)* — This service is used to prompt the endpoint object to go immediately to the ENDPOINT OFF state from the ENDPOINT ON state as defined by the State attribute. All endpoint detection event processes are aborted and all active timers are stopped. If the state attribute is already set to ENDPOINT OFF, the endpoint object remains in the appropriate state. If the device turns off successfully, a "success" response is returned. If the device fails to turn off properly, a "fail" response is returned and the endpoint object remains in the current state.

8.5.2.3 *Endpoint Start (Optional)* — This service is used to prompt the endpoint object to go from the ENDPOINT OFF or ENDPOINT IDLE state to the ENDPOINT IN PROCESS state and begin the endpoint detection event process with the initial endpoint attribute parameter values. If the State attribute is not currently in the ENDPOINT OFF or ENDPOINT IDLE state, an "object state conflict" error response is returned and the endpoint event remains in the appropriate state.

8.5.2.4 *Endpoint Suspend (Optional)* — This service is used to prompt the endpoint object to go from the ENDPOINT IN PROCESS state to the ENDPOINT SUSPENDED state and suspend the endpoint detection event process. All active timers are suspended and held at their current values. If the State attribute is not currently in the ENDPOINT IN PROCESS state, an "object state conflict" error response is returned and the endpoint object

remains in the appropriate state. The Endpoint Suspend service is conditional on the Endpoint Resume service being supported.

8.5.2.5 *Endpoint Resume (Optional)* — This service is used to prompt the endpoint object to go from the ENDPOINT SUSPENDED state to the ENDPOINT IN PROCESS state and resume the endpoint detection event process. The process resumes using the endpoint attribute parameter values and timer readings that were saved when the endpoint was suspended by the Endpoint Suspend service request. If the Endpoint Resume service is issued while the State attribute is not in the ENDPOINT SUSPENDED state, an "object state conflict" error response is returned and the endpoint event remains in its existing state. The Endpoint Resume service is conditional on the Endpoint Suspend service being supported.

8.5.3 *Sensor-BI-TH-EP Object Behavior* — The behavior exhibited by the Sensor-BI-TH-EP object is inherited from the Sensor-BI-TH object defined in SEMI E54.1. Additional specific behavior associated with the Sensor-BI-TH-EP object is defined below.

8.5.3.1 *Sensor-BI-EP OPERATING Application Process* — When in the ENDPOINT ON / ENDPOINT IN PROCESS state, the '*Value'* attribute is set to FALSE and the '*Reading Value'* attribute is set to VALID. *A* reading, or readings, may be retrieved from the device physical signal processing electronics. This reading may be filtered, analyzed, and corrected with a manufacturer-specified algorithm. This corrected reading becomes the input to the endpoint formula to generate the 'V*alue'* attribute as referenced in SEMI E54.1. This process is called "endpoint detection" and is executed only when the Sensor-BI-TH-EP object is in the OPERATING state as defined in SEMI E54.1. When an Endpoint event is detected, the '*Value'* attribute is set to TRUE and the '*Reading Valid'* attribute is VALID. When not in the OPERATING state, the value of the '*Reading Valid*' attribute shall be set to INVALID. When in one of the sub-states of the OPERATING state, the validity of the '*Value*' attribute shall be manufacturer specific. Required sub-states within the OPERATING state, descriptions of these sub-states, and a transition matrix associated with these sub-states are given in Figure 2, Table 16 and Table 17 respectively.

8.5.3.1.1 Whenever a 'Sensor Not Detected' alarm, 'Endpoint Failure' alarm or 'Unexpected Conditions Detected' alarm is active, the EPD device shall turn off the endpoint detection event process and hold the device in the ENDPOINT OFF state until the alarm clears (see ¶8.2).

8.5.3.1.2 A device can concurrently monitor and report many endpoint detection events. The number of endpoint detection events varies by vendor and device model.

8.5.3.1.3 The 'Value' attributes for all Sensor-BI-TH-EP objects shall be held at 'FALSE' until a valid endpoint detection event is determined. An endpoint detection event is initiated when the endpoint service request 'Endpoint Start' or Endpoint On (if the state ENDPOINT ON / ENDPOINT IDLE is not supported) is received and the endpoint object can successfully transition to the ENDPOINT ON / ENDPOINT IN PROCESS state.

8.5.3.1.4 When attempting to set the "Minimum Time", "Maximum Time", and "Target Time" attribute to a value above the capability of the endpoint detection event an error response shall be generated indicating an invalid operation has been attempted.

8.5.3.1.5 When attempting to set the "Recipe Identifier" or "Step Identifier" attribute to an identifier outside the range of the endpoint detection event an error response shall be generated indicating an invalid operation has been attempted.

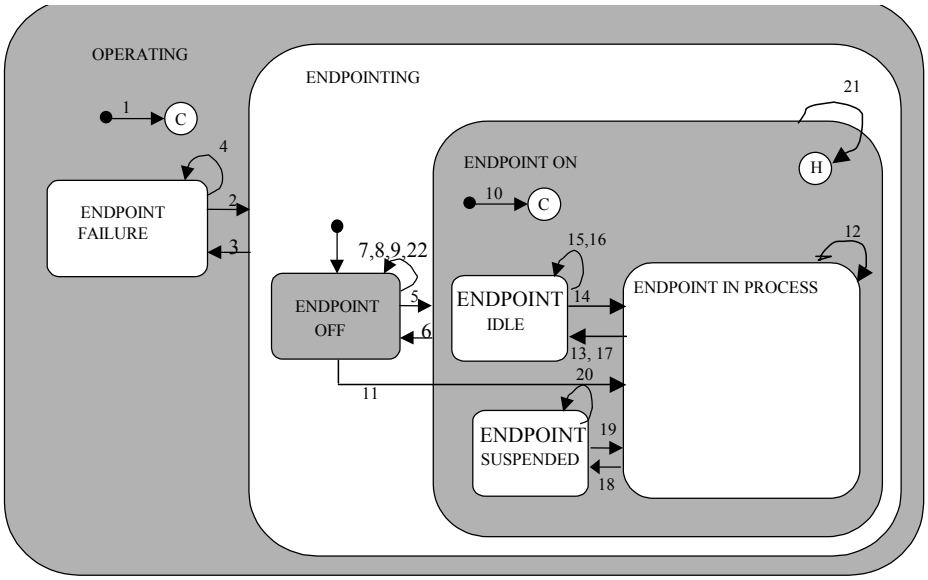


Figure 2
Sensor-BI-TH-EP Object Behavior Within the Operating State

Table 16 Sensor-BI-TH-EP Behavior OPERATING Sub-state Description

State	*Description*
OPERATING	Endpoint is in one of the following enumerated states as indicated in the Sensor-BI-TH-EP State attribute: ENDPOINT FAILURE (optional), ENDPOINT OFF, ENDPOINT IDLE (optional), ENDPOINT IN PROCESS, or ENDPOINT SUSPENDED (optional). Endpoint will respond to Endpoint On and Endpoint Off services as appropriate to move between sub-states within the ENDPOINTING and ENDPOINT ON states.
ENDPOINTING	Endpoint is in one of the following enumerated states as indicated in the Sensor-BI-TH-EP State attribute: ENDPOINT OFF, ENDPOINT IDLE, ENDPOINT SUSPENDED, and ENDPOINT IN PROCESS. Endpoint will respond to Endpoint On, Endpoint Off, Endpoint Start, Endpoint Suspend, and Endpoint Resume services as appropriate to move between sub-states within the ENDPOINTING state.
ENDPOINT OFF	This is a sub-state to ENDPOINTING; Endpoint Off is the status of the Sensor-BI-TH-EP Status attribute and ENDPOINT OFF is the enumerated state of the Sensor-BI-TH-EP State attribute. Endpoint object is **NOT** performing the "endpoint" detection event process. The endpoint object may be downloaded with new recipe parameters when in this state.
ENDPOINT ON	This is a sub-state of ENDPOINTING; Endpoint On is the status of the Sensor-BI-TH-EP Status attribute and ENDPOINT IDLE, ENDPOINT SUSPENDED or ENDPOINT IN PROCESS is the enumerated state of the Sensor-BI-TH-EP State attribute. Endpoint object may not be performing the "endpoint" detection event process.
ENDPOINT IDLE	This is a sub-state to ENDPOINT ON. Endpoint On is the status of the SENSOR-BI-TH-EP Status attribute and ENDPOINT IDLE is the enumerated state of the Sensor-BI-TH-EP State attribute. Endpoint object is not performing the "endpoint" detection event process. The endpoint object may be downloaded with new recipe parameters.
ENDPOINT IN PROCESS	This is a sub-state to ENDPOINT ON. Endpoint On is the status of the Sensor-BI-TH-EP Status attribute and ENDPOINT IN PROCESS is the enumerated state of the Sensor-BI-TH-EP State attribute. Endpoint object is performing the "endpoint" detection event process. The endpoint object may not be downloaded with new recipe parameters when in this state.
ENDPOINT SUSPENDED	This is a sub-state to ENDPOINT ON; Endpoint On is the status of the Sensor-BI-TH-EP Status attribute and ENDPOINT SUSPENDED is the enumerated state of the Sensor-BI-TH-EP State attribute. Endpoint object is **NOT** performing the "endpoint" detection event process. The endpoint object may not be downloaded with new recipe parameters when in this state.

State	*Description*
ENDPOINT FAILURE	Endpoint electronics failure; Sensor-BI-TH-EP State attribute is in the enumerated state: ENDPOINT FAILURE. Endpoint is not performing the "endpoint" detection event process. Endpoint cannot move to the ENDPOINTING state until the electronics are repaired or replaced. A Perform Diagnostics or Reset service request to the Device Manager object may be sent to validate that the failure has been cleared. The endpoint object may not be downloaded with new recipe parameters when in this state.

Table 17 Sensor-BI-TH-EP Behavior EXECUTING Sub-state Transition Matrix*

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
1	Entry into OPERATING	Power Up.	Conditional: ENDPOINTING or ENDPOINT FAILURE (optional)	If the endpoint electronics are in a failure condition than State is ENDPOINT FAILURE (optional) or ENDPOINTING. If endpoint failure checking is not supported the state is ENDPOINTING / ENDPOINT OFF. Set Status attribute to appropriate value.	Entry state depends on availability of endpoint electronics checking.
2	ENDPOINT FAILURE	Perform Diagnostics or Reset service request received and device determines that the failure no longer exists or the Endpoint device detects replacement of electronics that caused the failure.	ENDPOINTING / ENDPOINT OFF	Set Status attribute to appropriate value.	Endpoint moves to ENDPOINT OFF state.
3	ENDPOINTING	Endpoint device detects an endpoint electronics failure.	Conditional: ENDPOINT FAILURE or ENDPOINTING / EDNDPOINT OFF	Set State attribute to appropriate value.	The setting of the Status attribute is manufacturer specific.
4	ENDPOINT FAILURE	Endpoint Off, Endpoint On, Endpoint Start, Endpoint Resume, Endpoint Suspend, Download Recipe, Upload Recipe, and Calibrate request.	ENDPOINT FAILURE	Error response.	Object cannot move to ENDPOINTING / ENDPOINT OFF state until the electronics are repaired.

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
5	ENDPOINT OFF	Endpoint On request.	Conditional: ENDPOINT ON / ENDPOINT IDLE or ENDPOINT ON / ENDPOINT IN PROCESS	If ENDPOINT IDLE supported: Take the object from the ENDPOINT OFF state to the ENDPOINT IDLE state and wait for an Endpoint Start request. Recipe parameter downloads are allowed. If ENDPOINT IDLE is not supported: Take the object from the ENDPOINT OFF state immediately to the ENDPOINT IN PROCESS state and begin the endpoint detection event process. Set the Status attribute to the appropriate values throughout the transition. Issue an Endpoint On response.	ENDPOINT IDLE is an optional intermediate state between ENDPOINT OFF and ENDPOINT IN PROCESS. Endpoint electronics must not be in failure. Service response is not issued until transition to ENDPOINT IDLE or ENDPOINT IN PROCESS is completed.
6	ENDPOINT ON	Endpoint Off request.	ENDPOINT OFF	Stop the "endpoint" detection event process. Issue an Endpoint Off response.	This results in the same state transition as when a valid Abort service request is issued to the device.
7	ENDPOINT OFF	Endpoint Off request.	ENDPOINT OFF	Error response.	Endpoint is already off.
8	ENDPOINT OFF	Endpoint Suspend, or Endpoint Resume request.	ENDPOINT OFF	Error response.	Endpoint event process remains off.
9	ENDPOINT OFF	Endpoint On or Endpoint Start request. Endpoint is unable to turn on properly.	ENDPOINT OFF	Endpoint attempts to take the object from the ENDPOINT OFF state to an ENDPOINT ON state. Endpoint won't turn on properly. Turn "endpoint" detection event process back off and generate an Error response.	Behavior associated with determining that the endpoint will not turn on properly is manufacturer specific.
10	Entry into ENDPOINT ON	None.	Entry state for ENDPOINT ON state. Conditional: ENDPOINT ON / ENDPOINT IDLE or ENDPOINT ON / ENDPOINT IN PROCESS	ENDPOINT IDLE state entered if supported; otherwise ENDPOINT IN PROCESS state entered.	Entry depends on endpoint electronics functioning properly.

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
11	ENDPOINT OFF	Endpoint Start request.	ENDPOINT ON / ENDPOINT IN PROCESS	Go immediately to the ENDPOINT ON / ENDPOINT IN PROCESS state and begin the endpoint detection event process. Issue an Endpoint Start response.	Behavior associated with determining that the endpoint will not turn on properly is manufacturer specific.
12	ENDPOINT IN PROCESS	Endpoint On, Endpoint Start, Endpoint Resume, Download Recipe, Upload Recipe.	ENDPOINT IN PROCESS	Error response.	Endpoint is already in process.
13	ENDPOINT IN PROCESS	Endpoint detection completed and idle state supported.	ENDPOINT IDLE	Endpoint event detected. Endpoint event reported.	
14	ENDPOINT IDLE	Endpoint Start Request	ENDPOINT IN PROCESS	Endpoint event detection process is started. Issue an Endpoint Started success response.	
15	ENDPOINT IDLE	Download Recipe, and Upload Recipe Request.	ENDPOINT IDLE	Valid recipes are Downloaded or Uploaded and a services response is generated.	
16	ENDPOINT IDLE	Endpoint Suspend and Endpoint Resume request.	ENDPOINT IDLE	Error response.	Endpoint is not in process so it can not be suspended.
17	ENDPOINT IN PROCESS	Endpoint detection failed.	Conditional: ENDPOINTING ON / ENDPOINT IDLE or ENDPOINTING / ENDPOINT OFF	Endpoint event not detected within the allotted endpoint active timers. Endpoint event failure reported.	
18	ENDPOINT IN PROCESS	Endpoint Suspend request and service supported.	ENDPOINT SUSPENDED	Take the object from the ENDPOINT IN PROCESS state to ENDPOINT SUSPENDED state, turning off the endpoint event process. All active endpoint timers are held at their current values. Set the Status attribute to the appropriate value. Issue an Endpoint Suspended response and stop the “endpoint” detection process.	Endpoint event process is temporally suspended. All timers are suspended.

#	Current State	Trigger	New State	Action	Comment
19	ENDPOINT SUSPENDED	Endpoint Resume request.	ENDPOINT IN PROCESS	Take the object from ENDPOINT SUSPENDED state to ENDPOINT IN PROCESS. Begin the "endpoint" detection event process form the point it was suspended. All active endpoint timers are reinstated to their last held values. Issue an Endpoint Resume response.	Endpoint event process is resumed from the point it was suspended.
20	ENDPOINT SUSPENDED	Endpoint On, Endpoint Start, Endpoint Suspend , Download Recipe, Upload Recipe request.	ENDPOINT SUSPENDED	Error response.	Endpoint event process remains suspended.
21	ENDPOINT ON	Endpoint On, Endpoint Start, Endpoint Resume, Download Recipe, and Upload Recipe request.	ENDPOINT ON	Error response.	Endpoint detection event process remains on and continues endpoint process from its current attribute settings.
22	ENDPOINT OFF	Download Recipe or Upload Recipe request.	ENDPOINT OFF	Valid recipes are Downloaded or Uploaded and a services response is generated.	

8.6 *Assembly-EP Objects* — Assembly-EP objects inherit attributes, services, and behavior from the Assembly object. The Assembly object is the device component that provides a mechanism of grouping more than one attribute from one or more objects into a single data structure for communication over the network.

Table 18 identifies the Assembly-EP objects defined for the EPD device.

Table 18 Assembly List

Object	Access Network	Required	Form
Assembly-EPD#1	R	Yes	Status; Default Assembly
Assembly-EPD#2	R	No	Value, Reading Valid
Assembly-EPD#3	R	No	Status, Number of Endpoint Objects and Value (Values ≤ 1024)
Assembly-EPD#4	R	No	Status, Exception Detail Alarm and Exception Detail Warning

8.6.1 *Assembly-EP Objects Attributes* — Table 19 provides a list of attributes common to all Assembly-EP object types.

Table 19 Assembly-EP Object Attributes

Attribute Name	Attribute Identifier	Access Network	Required	Form
Data [#1]	A1	RW	Yes	Structure as defined below.

[#1] Inherited from the Assembly object as shown in Figure 1.

8.6.1.1 *Data Attribute Format for Assembly-EP Objects* — The Data attribute of all Assembly-EP objects is a structured attribute containing an ordered list of attributes within its structure. In the following tables (20 through 23), the structure of the Data attribute for each of the Assembly-EP object types is defined.

Table 20 Assembly-EPD #1 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	EPD1	A12	Device Manager Exception Status

Table 21 Assembly-EPD #2 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	EPD3	SbithepA16	Sensor-BI-TH-EP #1 Value Reading Valid

Table 22 Assembly-EPD #3 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	EPD1	A12	Device Manager Exception Status
2	EPD2	SacA65	SAC Number of Endpoint Objects
3	EPD3	SbithepA16	Sensor-BI-TH-EP #1 Value
4	EPD3	SbithepA16	Sensor-BI-TH-EP #2 Value
.	EPD3	SbithepA16	Sensor-BI-YH-EP #3 to Sensor-BI-TH-EP #N-1 Value
$N \leq 1024$ [#1]	EPD3	SbithepA16	Sensor-BI-TH-EP #N Value

[#1] 'N' is the value of the Sensor-BI-EP object attribute "Number of Endpointing Objects". Note also that the number of endpoints reported in an assembly is fixed throughout the life of the device, and is specified by the manufacturer.

Table 23 Assembly-EPD #4 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	EPD1	A12	Device Manager Exception Status
2	EPD1	A13	Device Manager Exception Detail Alarm
3	EPD1	A14	Device Manager Exception Detail Warning

SEMI E54.12-0701[E]
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR CC-LINK

This specification was technically approved by the Global Information and Control Committee and is the direct responsibility of the North American Global Information and Control Committee. Current edition approved by the North American Information and Control Committee on April 30, 2001. Initially available at www.semi.org May 2001; to be published July 2001.

[E] This standard was editorially modified in September 2001 to correct a typographical error and the omission of a required disclaimer. Section 2.3 was added and changes were made to Section 7.6.2.6.

1 Purpose

1.1 This specification is part of the SEMI Sensor/Actuator Network (SAN) suite of standards and defines a specific communications protocol based on the CC-Link standard. This Network Communication Standard (NCS) taken together with the SEMI Sensor/Actuator Network standard suite and the CC-Link standard completely and unambiguously defines an open standard providing an industry specific solution to off-the-shelf interoperability of networked devices in semiconductor manufacturing equipment.

1.2 CC-Link is a vendor independent, open device level network standard. Vendor independence and openness are guaranteed by the CC-Link Partner Association.

2 Scope

2.1 This document specifies a SAN communications standard based on the CC-Link specification that is in compliance with SEMI E54.1. As such, it specifies the protocol, services and behavior that compliant intelligent devices must support in order to interchange information over this SAN in a method compatible with SEMI E39.

2.2 In conjunction with a SEMI standard SAN Common Device Model (CDM) specification and one or more SEMI standard Specific Device Model (SDM) specifications (e.g., for a mass flow controller), this Network Communication Standard (NCS) with the related CC-Link standard describe the data structures, interactions and behavior that are characteristic of the various devices on the network. This composite model forms a complete interoperability standard for communications among intelligent sensors, actuators and controllers in semiconductor manufacturing equipment.

2.3 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 health practices and determine the applicability or regulatory limitations prior to use.

3 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on CC-Link; thus, a complete specification of this standard necessarily includes the CC-Link specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 The specifications within are strictly enhancements that provide additional capabilities over and above those currently required by CC-Link. Included throughout this document, primarily in Section 6, is information paraphrased from the CC-Link specifications — such as: protocol structure, capabilities, options and limitations. This information is provided here for reference only and is not intended to provide specification definitions. In all such areas, refer to the CC-Link specification documents for information. This document is limited to describing enhancements or limitations to the CC-Link specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the CC-Link protocol conformance testing specification. Conformance testing shall also include enhancements and limitations to the CC-Link specification required by this standard.

4 Referenced Standards

NOTE 1: Unless otherwise indicated, all documents cited shall be the latest published versions.

4.1 *SEMI Standards*

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

SEMI E54.1 — Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

4.2 *ISO[1] Standards*

7498 OSI — Basic Reference Model for Open Systems Interconnection

4.3 *CC-Link Partner Association[2]*

CC-Link Specification, Version 1.11 (or later)

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *CDM* — Common Device Model

5.1.2 *NCS* — Network Communication Standard

5.1.3 *OSI* — Basic Reference Model for Open Systems Interconnection (ISO 7498)

5.1.4 *SAN* — Sensor/Actuator Network

5.1.5 *SDM* — Specific Device Model

5.2 *Definitions from Sensor/Actuator Network Common Device Model (SEMI E54.1)*

5.2.1 *Attribute*

5.2.2 *Behavior*

5.2.3 *Byte*

5.2.4 *Common Device Model*

5.2.5 *Device*

5.2.6 *Device Manager (DM) Object*

5.2.7 *Device Model*

5.2.8 *Instance*

5.2.9 *Network Communication Standard*

5.2.10 *Object*

5.2.11 *Sensor, Actuator and Controller (SAC) Object*

5.2.12 *Service*

5.2.13 *Specific Device Model*

5.2.14 *State Diagram*

5.3 *Definitions*

5.3.1 *broadcast polling method* — polling to each station and the data communication are executed by the same packet, and the data is transmitted to all of the stations in this method.

5.3.2 *cyclic transmission* — function to transmit the data from master station to all stations periodically, then for each station to transmit the response data to master station.

5.3.3 *intelligent device station* — station which can send cyclic transmission and transient transmission to master station.

5.3.4 *local station* — station which can send cyclic transmission and transient transmission to master station and other local stations.

5.3.5 *master station* — station that controls all stations on CC-Link. One (and only one) master station per system is required.

5.3.6 *profiles* — Application Object Model specifications.

5.3.7 *remote device station* — station that handles bit data and word data.

5.3.8 *remote I/O station* — station that handles only bit data.

5.3.9 *remote station* — generic name of remote I/O station and remote device station.

5.3.10 *slave station* — generic name of station other than master station.

5.3.11 *station* — equipment which can be connected with CC-Link and is assigned a station number of 0–64.

5.3.12 *transient transmission* — function to transmit the non-periodic data generated in master station, local station, and intelligent device station.

6 Communication Protocol High Level Structure

6.1 Message transfer is organized in cycles. A message cycle mainly consists of a request-frame followed by a corresponding acknowledge/response-frame of the addressed station.

6.2 A brief description of the CC-Link protocol as it relates to the ISO 7498 OSI model follows in the sections below. For protocol efficiency, CC-Link does not define layers 3 to 6.

NOTE 2: The information contained in this section is for reference only. It in no way represents specifications for CC-Link. See related documentation for these specifications.

6.3 *Physical Layer - Layer 1* — the Physical Layer conforms to the EIA RS-485 standard. See the CC-Link standard for more information.

6.4 *Data Link Layer - Layer 2* — the Data Link Layer conforms to the HDLC standard. See the CC-Link standard for more information.

6.5 *Application Layer - Layer 7* — the Application Layer defines services and protocols for Network

1 ISO - International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211, Geneva 20, Switzerland

2 CC-Link Partner Association - <www.cc-link.org>

Management, Cyclic Transmission and Non-Cyclic Transmission (or Transient Transmission). Also, Application Object Models are specified as "Profiles". See the CC-Link standard for more information.

6.6 Version 1.11 of the CC-Link Standard introduces two methods for request/response messaging.

6.7 Devices that support the CC-Link Transient Transmission capability, use this method, together with a messaging protocol to transmit Service Requests and Responses.

6.8 Devices that do not support the Transient Transmission capability, use the Cyclic Transmission method. A protocol is defined for changing the context of the cyclic data from I/O to a messaging protocol for Service Requests and Responses.

6.9 See the CC-Link standard for more information.

7 Required Object Types

7.1 This section describes a general mapping of the SEMI SAN Object Model to the CC-Link environment. Component definitions are clarified and the mapping of Attributes, Services and Behaviors are specified.

7.2 *Object Model* — The Object Model defined in the CDM is represented in the CC-Link NCS. Specifically, the DM and SAC objects are mapped.

7.2.1 Section 9 specifies the mapping of SDM Objects in CC-Link.

7.3 *Objects* — The required objects of the CDM are identified here. Additional objects that are contained in the SDM are given identifiers in the Device Profile. Section 9 specifies additional mapping information.

7.3.1 Table 1 lists the Object Identifiers specified for use in protocol messages.

Table 1 Object Identifiers

Object ID	*Object*
0	Invalid
1	DM Object
2	SAC Object
3–n	Application Objects as specified in Section 9

7.4 *Attributes* — All attributes are accessible via Get_Attribute and Set_Attribute services defined in the sections below.

7.4.1 *Attribute Identifiers* — Every object specified in the CDM and SDMs uses tags to identify its attributes. These tags are formatted with letters (identifying the object) followed by an upper case "A", followed by a numerical identifier. The Attribute ID used in the CC-Link NCS is simply the numerical portion of these tags.

7.5 See Table 2 for a list of DM attributes.

Table 2 DM Object Attribute Identifiers

SEMI CDM Attribute ID	*CC-Link Attribute ID*	*Attribute*
DmA1	1	Device Type
DmA2	2	Standard Revision Level
DmA3	3	Device Manufacturer Identifier
DmA4	4	Manufacturer Model Number
DmA5	5	Software or Firmware Revision Level
DmA6	6	Hardware Revision Level
DmA7	7	Serial Number
DmA8	8	Device Configuration
DmA9	9	Device Status
DmA12	12	Exception Status
DmA13	13	Exception Detail Alarm
DmA14	14	Exception Detail Warning
DmA15	15	Visual Indicator
DmA16	16	Alarm Enable
DmA17	17	Warning Enable

7.6 *Services*

7.6.1 *Service Identifiers* — The required services of the CDM are identified here. Additional services that are contained in the SDM are given identifiers in the Device Profile. Table 3 specifies the required services and ID numbers.

Table 3 Service Identifiers

Service ID	*Service*
0	Invalid
1	Reset
2	Abort
3	Recover
4	Get Attribute
5	Set Attribute
6	Execute
7	Perform Diagnostics

7.6.2 *Specified Services* — The following sections define the details associated with each of the services required by the CDM.

7.6.2.1 *Reset* — The Reset Request specifies no parameters. In addition to an explicit Reset Service Request, CC-Link specifies others methods whereby a Slave device can be reset.

7.6.2.2 *Abort* — The Abort Service Request specifies no parameters.

7.6.2.3 *Recover* — The Recover Service Request specifies no parameters.

7.6.2.4 *Get Attribute* — The Get Attribute Request specifies two parameters: the Object ID and the Attribute ID. Each are currently defined in the range 1–255. Both are expandable to 65,535.

7.6.2.5 *Set Attribute* — The Set Attribute Request specifies three parameters: the Object ID, the Attribute ID and the Attribute Value to set. The Object ID and the Attribute ID are defined the same as for the Set Attribute Service. The length of the Attribute Value is based on the specification of the attribute.

7.6.2.6 *Execute* — The Execute Service Request specifies no parameters.

7.6.2.7 *Perform Diagnostics* — The Perform Diagnostic Request specifies one parameter: Test ID. The Test ID parameter is one byte in length.

8 Protocol Compliance

8.1 The CC-Link Partner Association has established a qualified certification system, with test laboratories in Japan that include conformance testing and interoperability testing.

9 Specific Device Model Mappings

9.1 The following sections specify mappings for Sensor Actuator Network Specific Device Models.

9.2 *Mass Flow Device* — Reference SEMI E54.3 for a complete specification of the SDM for Mass Flow Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.2.1 *Objects* — Consistent with SEMI E54.3 and Section 7.3 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.2.1.1 Table 4 shows the mapping of the SDM Object Instances specified in SEMI E54.3 (Instance numbers are listed under heading Inst. in the table) and the CC-Link Object ID (listed under ID in the table).

Table 4 MFD Object Identifiers

SDM Object Name	*SDM Object ID*	*Inst.*	*ID*
Sensor-AI-MF	MFD3	1	3
Actuator-AO-MF	MFD7	1	4
Controller	MFD8	1	5
SISO-Setpoint	MFD11	1	6
Sensor-AI-AT	MFD4	1	7

9.2.1.2 Additional objects may be defined by the manufacturer in the Device Profile for a given device.

9.2.2 *Attributes* — The mapping of Attribute Tags and Identifiers is defined in Section 7.4.1 for the CDM. The same method applies here for the SDM.

9.2.3 *Services* — The mapping of Service Tags and Identifiers is defined in Section 7.6.1 for the CDM. The same method applies here for the SDM.

9.3 *In-Situ Particle Monitor* — Reference SEMI E54.10 for a complete specification of the SDM for In-Situ Particle Monitor Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.3.1 *Objects* — Consistent with SEMI E54.10 and Section 7.3 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.3.1.1 Table 5 shows the mapping of the SDM Object Instances specified in SEMI E54.3 (Instance numbers are listed under heading Inst. in the table) and the CC-Link Object ID (listed under ID in the table).

Table 5 ISPM Object Identifiers

SDM Object Name	*SDM Object ID*	*Inst.*	*ID*
Sensor-AI-LCS	ISPMD3	1	3
Sensor-AI-SLS	ISPMD4	1	4
Sensor-AI-MNS	ISPMD5	1	5
Assembly-ISPM#1	ISPMD17	1	6
Assembly-ISPM#2	ISPMD18	1	7
Assembly-ISPM#3	ISPMD19	1	8
Assembly-ISPM#4	ISPMD20	1	9
Assembly-ISPM#5	ISPMD21	1	10
Assembly-ISPM#6	ISPMD22	1	11
Assembly-ISPM#7	ISPMD23	1	12
Assembly-ISPM#8	ISPMD24	1	13
Assembly-ISPM#9	ISPMD25	1	14
Assembly-ISPM#40	ISPMD64	1	15
Sensor-AI-Counter	ISPMD16	1	16
Sensor-AI-Counter	ISPMD16	n	15 + n
Sensor-AI-Counter	ISPMD16	1024	1039

9.3.1.2 Additional objects may be defined by the manufacturer in the Device Profile for a given device.

9.3.2 *Attributes* — The mapping of Attribute Tags and Identifiers is defined in Section 7.4.1 for the CDM. The same method applies here for the SDM.

9.3.3 *Services* — The mapping of Service Tags and Identifiers is defined in Section 7.6.1 for the CDM. The same method applies here for the SDM.

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. Mitsubishi has filed a statement with SEMI asserting that licenses will be made available to applicants throughout the world for the purpose of implementing this standard without unfair discrimination. Attention is also drawn to the possibility that some elements of this standard may be subject to patented technology or copyrighted items other than those identified above. Semiconductor Equipment and Materials International (SEMI) shall not be held responsible for identifying any or all such patented technology or copyrighted items. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights and the risk of infringement of such rights are entirely their own responsibility.

SEMI E54.13-0303
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR ETHERNET/IP™

This specification was technically approved by the Global Information and Control Committee and is the direct responsibility of the North American Information and Control Committee. Current edition approved by the North American Regional Standards Committee on November 22, 2002. Initially available at www.semi.org January 2003; to be published March 2003.

1 Purpose

1.1 *Introduction*

1.1.1 This standard defines a communication specification based on the EtherNet/IP[1] (Ethernet/Industrial Protocol) network to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI specified device models (common and device specific) in a semiconductor manufacturing tool.

1.2 *Motivation*

1.2.1 EtherNet/IP is a communication system suitable for use in industrial environments. EtherNet/IP allows intelligent devices to exchange time-critical application information. These devices include simple I/O devices such as sensors/actuators, as well as complex control devices such as robots, programmable logic controllers, and process controllers.

1.2.2 EtherNet/IP uses CIP (Control and Information Protocol), the common network, transport and application layers also shared by DeviceNet (SEMI E54.4). EtherNet/IP provides:

- A cost effective solution to bridge factory Ethernet (IEEE 802.3) networks to SEMI E54.4 DeviceNet low-level device networks,
- Access to intelligence present in low-level devices, and
- Producer/Consumer model for Master/Slave and Peer-to-Peer application relationships.

1.3 *Background*

1.3.1 EtherNet/IP makes use of standard Ethernet and TCP/IP technology to transport CIP communications packets. The result is a common, open application layer on top of open and highly popular Ethernet and TCP/IP protocols.

1.3.2 EtherNet/IP provides a producer/consumer model for the exchange of time-critical control data. The producer/consumer model allows the exchange of application information between a sending device (e.g., the producer) and many receiving devices (e.g., the consumers) without the need to send the data multiple times to multiple destinations. For EtherNet/IP, this is accomplished by making use of the CIP network and transport layers along with IP Multicast technology. Many EtherNet/IP devices can receive the same produced piece of application information from a single producing device.

1.3.3 EtherNet/IP makes use of standard IEEE 802.3 technology; there are no non-standard additions that attempt to improve determinism. Rather, EtherNet/IP recommends the use of commercial switch technology, with 100 Mbps bandwidth and full-duplex operation, to provide for more deterministic performance.

2 Scope

2.1 *Specification* — This document specifies a Sensor/Actuator Network Communications Standard (NCS) based on the EtherNet/IP specification that enables communication with SAN devices configured according to SEMI SAN Common Device Model (CDM) and appropriate Specific Device Model (SDM) specifications.

2.2 *Use* — This document is used in conjunction with a SEMI standard SAN CDM specification and one or more SEMI standard SDM specifications (e.g. for a mass flow controller). Together, they describe the externally visible data structure and behavior of devices utilizing the EtherNet/IP networking capability in a SEMI compliant SAN system. The general sensor/actuator network document architecture is described in the SEMI E54.0 Sensor/Actuator Network Standard (the root SAN document).

2.3 *Document Structure* — The EtherNet/IP network communication standard complies with the SEMI SAN NCS template document structure, as described in SEMI E54.0. The standard document is composed of two main parts. The first part (Sections 1 through 8) specifies the SAN enabling protocol as well as the presentation (i.e., mapping) of CDM object structure and behavior onto the network (referred to as the "CDM mapping"). The second part (Section 9) specifies the presentation (i.e., mapping) of SDM object

1 EtherNet/IP is a trademark of Open DeviceNet Vendor Association (ODVA)

structure and behavior onto the network for each SEMI specified SDM (referred to as the "SDM mapping").

2.4 *Adding SDM Mappings* — SDM mappings added to part two of this document are considered document additions and are balloted as such. An SDM mapping may only be balloted for addition to this document if the corresponding SEMI SDM has been standardized.

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 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on EtherNet/IP and is a companion document to the EtherNet/IP specification; thus a complete specification of this standard necessarily includes the EtherNet/IP specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by EtherNet/IP. In order to avoid document consistency problems, information in the EtherNet/IP specification that relates to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the EtherNet/IP specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the EtherNet/IP conformance testing specification. Conformance testing shall also include enhancements and limitations to the EtherNet/IP specification required by this standard.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services (OSS)

SEMI E54 — Sensor/Actuator Network Standard

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.2 — Guide for Writing Sensor/Actuator Network (SAN) Standard Ballots

4.2 *ISO Standards*[2]

ISO 7498 — Basic Reference Model for Open Systems Interconnection.

ISO/IEC 8802-3 — Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (Ethernet IEEE 802.3).

4.3 *Other Documents*

ODVA[3] EtherNet/IP Specification — Volume I and Volume II, Release 1.0.

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

5 Terminology

5.1 Terminology that is common to all of the documents in this SAN standard may also be defined in SEMI E54. Terminology may be reproduced here which is defined in other SEMI documents.

5.2 *Abbreviations and Acronyms*

5.2.1 *CDM* — Common Device Model

5.2.2 *CIP* — Control and Information Protocol

5.2.3 *CM* — Connection Manager object

5.2.4 *DM* — Device Management object

5.2.5 *EIP* — EtherNet/IP

5.2.6 *EL* — Ethernet Link object

5.2.7 *IP* — Internet Protocol

5.2.8 *MR* — Message Router object

5.2.9 *NCS* — Network Communication Standard

5.2.10 *OSI* — Open Systems Interconnect

5.2.11 *OSS* — Object Services Standard

5.2.12 *PDU* — Protocol Data Unit

5.2.13 *SAC* — Sensor, Actuator, Controller object

5.2.14 *SAN* — Sensor/Actuator Network

5.2.15 *SDM* — Specific Device Model

5.2.16 *S-DS* — S-Device Supervisor object

5.2.17 *TCP* — Transport Control Protocol

5.3 *Device Component Definitions*

---

2 International Organization for Standardization. ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30. Website: /www.iso.ch

3 Open DeviceNet Vendor Association, 20423 State Road 7 #F6, Boca Raton, FL 33498-6797; Phone: (1) 561-477-7966; Website: www.odva.org.

5.3.1 As this standard defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in the CDM document. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the EtherNet/IP specification. Note that Column 2 contains an equal sign "=" if the definition is used exactly as specified in the CDM specification.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*	*EtherNet/IP Equivalent*
Device	=	=
Device Model	=	Device Profile
Object	=, Class	=, Class
Instance	=	=
Attribute	=	=
Behavior	=	=
Service	=	=
Behavior State Diagram	=	State Transition Diagram
State Transition Matrix	=	State Event Matrix

5.4 *EtherNet/IP Specific Definitions*

5.4.1 *class* — A set of objects that all represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values.

5.4.2 *Control and Information Protocol (CIP)* — The common network, transport and application layers shared by EtherNet/IP and DeviceNet.

5.4.3 *device profile* — An EtherNet/IP specification for a device that contains an object model for the device type, the I/O data format for the device type, and the configuration data and the public interface(s) to that data.

5.4.4 *encapsulation* — The technique used by layered protocols in which a layer adds header information to the protocol data unit (PDU) from the layer above. As an example, in Internet terminology, a packet would contain a header from the data link layer, followed by a header from the network layer (IP), followed by a header from the transport layer (TCP), followed by the application protocol data.

5.4.5 *Ethernet* — A 10/100-Mb/s standard for LANs, initially developed by Xerox, and later refined by Digital, Intel and Xerox (DIX). All hosts are connected to the network media where they contend for network access using a Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

5.4.6 *EtherNet/IP* — EtherNet/IP (Ethernet/Industrial Protocol) is a networked communications protocol that specifies the encapsulation of CIP over TCP/IP.

5.4.7 *Explicit Message Connections* — Connections over an EtherNet/IP network that provide generic, multi-purpose communication paths between two devices. These connections often are referred to as just *Messaging* Connections. Explicit Messages provide the typical request/response–oriented network communications.

5.4.8 *Input/Output Connections* — Connections over an EtherNet/IP network that provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data moves though these ports.

6 Communication Protocol High Level Structure

6.1 Though the protocol specification is called "EtherNet/IP", Ethernet is technically not required. EtherNet/IP may be used on any media that supports transmission of the Internet Protocol (IP). When Ethernet is used, it shall comply with the IEEE 802.3 specification.

NOTE 1: Conformance testing of EtherNet/IP products is limited to those products implemented on Ethernet.

6.2 The following sections give a brief description of the EtherNet/IP specification in terms of the seven layer architecture described by the ISO Basic Reference Model for Open Systems Interconnection (the OSI model).

6.3 *Physical Layer* — The physical layer requirements include specifications for topology, media, connectors, shielding and grounding. Where copper media is used, the EtherNet/IP specification is based on the ANSI/TIA/EIA-568-B Category 5/5E cable/connector specifications. A Commercial grade and Industrial grade level of conformance are specified. The signaling and coupling are as specified in the IEEE 802.3 / TP-PMD standard.

6.4 *Data Link Layer* — For compatibility with EtherNet/IP conformance testing capabilities, the device must comply with the Ethernet Data Link Layer Specifications as specified by IEEE 802.3. Otherwise, any data link layer that supports the IP is acceptable.

6.5 *Network Layer* — The network layer for the EtherNet/IP protocol is defined by the Internet Protocol (IP version 4) (RFC 791).

6.6 *Transport Layer* — The transport layer requires the support of Transmission Control Protocol (TCP) (RFC 793) as well as User Datagram Protocol (UDP) (RFC 768).

6.7 *Session Layer* — Many of the services and functions of the session layer for the EtherNet/IP protocol are divided and handled, in varying capacities, by the TCP/UDP protocol specifications and the EtherNet/IP Application Layer specifications. The first defines transport level connection sessions, while the latter defines application level sessions in terms of Explicit Connection and I/O Connection management.

6.8 *Presentation Layer* — The presentation layer is defined in terms of standard data types and specific encoding definitions for application layer data.

6.9 *Application Layer* — The device shall comply with the EtherNet/IP application layer specification for defining and addressing objects, including their attributes and services, and enabling specified network behavior. The device shall comply with the object model specifications provided in the EtherNet/IP specification.

6.10 *Object models* — EtherNet/IP provides an object-oriented specification for creating, defining and addressing objects explicitly, including their attributes and services (i.e., explicit messaging), and creating, defining and communicating object attribute assemblies in an application dependent format (i.e., input/output messaging). The device shall comply with the object model specifications provided in the EtherNet/IP documentation. In addition, the device shall comply with the object specifications defined in Section 7 of this document.

7 Required Object Types

7.1 The EtherNet/IP specification identifies and describes objects (i.e., classes) that must exist in all EtherNet/IP compliant devices. The Common Device Model (CDM) specification additionally identifies two objects, the Device Management (DM) and Sensor Actuator Controller (SAC) objects, that must exist in all SEMI compliant SAN devices. The required object types for a SEMI compliant SAN device utilizing the network communication specification described herein, necessarily comprises the union of the above two requirements.

7.2 A list of required and optional object types is given in Table 2. Note that the Sensor, Actuator, and Controller object types are not required, and are indicated as optional in the CDM specification. These objects are aggregated together to form a SEMI and EtherNet/IP compliant device.

7.3 The two required object types from the CDM (the DM and the SAC) are implemented as mapped components in EtherNet/IP; they are mapped to the SEMI-specific S-Device Supervisor object class as described in this section.

7.4 The listed object types are:

- Identity,
- Message Router (MR),
- Connection (Conn),
- Connection Manager (CM),
- Ethernet Link (EL),
- TCP/IP Interface (TCP/IP),
- S-Device Supervisor (S-DS),
- Device Management (DM),
- Sensor Actuator Controller (SAC),
- Sensor,
- Actuator, and
- Controller.

7.5 Also, although not a network addressable object, each device shall support the Unconnected Message Manager (UCMM) as defined by EtherNet/IP.

Table 2 Object Types

Object	*EIP Class ID (See Note 1.)*	*CDM Tag (See Note 2.)*	*Required by*		
			EIP (See Note 1.)	*CDM (See Note 2.)*	*NCS*
Identity	01	—	Yes	No	Yes
MR	02	—	Yes	No	Yes
Conn	05	—	Yes	No	Yes
CM	06	—	Conditional (See Note 1.)	No	Conditional (See Note 1.)
EL	F5	—	Conditional (See Note 1.)	No	Conditional (See Note 1.)
TCP/IP	F6	—	Yes	No	Yes
S-DS	30	—	No	No	Yes
DM	—	DmI0	No	Yes	Yes (See Note 4.)
SAC	—	SACI0	No	Yes	Yes (See Note 4.)
Sensor	31	SenIn	No	No	No
Actuator	32	ActIn	No	No	No

Object	*EIP Class ID (See Note 1.)*	*CDM Tag (See Note 2.)*	*Required by*		
			EIP (See Note 1.)	*CDM (See Note 2.)*	*NCS*
Controller	33	CntIn	No	No	No
(Other)	***	—	No	No	No

NOTE 1: See EtherNet/IP specification for further information; values are hexadecimal.

NOTE 2: See CDM specification for further information.

NOTE 3: Application Dependent.

NOTE 4: The DM and SAC Objects are implemented as mappings to the S-Device Supervisor object for EtherNet/IP.

7.6 See the EtherNet/IP specification for the implementation detail of the following object classes:

- Identity
- Message Router (MR)
- Connection (Conn)
- Connection Manager (CM)
- Ethernet Link (EL)
- TCP/IP Interface (TCP/IP)
- S-Device Supervisor (S-DS)

7.7 The Sensor, Actuator and Controller object types are utilized collectively to model the type specific structure and behavior of the device. The requirement and number of each of these object types in a device model is device type specific. Further, the attributes, services and behavior associated with each of these object classes and instances in a device is also device type specific, but must be compliant with both SEMI and EtherNet/IP specifications. The specification of these object types for a specific device type can be found in the appropriate SDM. The method of presentation of object structure and behavior to the EtherNet/IP network for objects defined for and associated with a specific device type can be found in Section 9 of this document.

7.8 The following sections describe the implementation detail for the two required objects from the CDM. The EtherNet/IP specification essentially combines the Device Management (DM) Object and the Sensor Actuator Controller (SAC) Object into a single Object: the S-Device Supervisor (S-DS) Object.

7.9 *Device Management (DM) Object* — The DM object is the device component responsible for managing and consolidating the device operation. The DM object as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object attributes and services to the EtherNet/IP network shall be as indicated in Table 3 and Table 4 respectively. Each attribute and service is shown with its corresponding identifier (ID) for the S-DS Object.

Table 3 Network Presentation of DM Object Attributes

Device Management Object Attributes		
CDM Tag	*Name*	*S-DS Attribute ID (See Note 1.)*
DmA1	Device Type	03
DmA2	Standard Revision Level	04
DmA3	Device Manufacturer Identifier	05
DmA4	Manufacturer Model Number	06
DmA5	Software or Firmware Revision Level	07
DmA6	Hardware Revision Level	08
DmA7	Serial Number	09
DmA8	Device Configuration	0A
DmA9	Device Status	0B
DmA10	Reporting Mode	N.A. (See Note 2.)
DmA11	Exception Status Timer	N.A. (See Note 2.)
DmA12	Exception Status	0C
DmA13	Exception Detail Alarm	0D
DmA13	Exception Detail Warning	0E

NOTE 1: Attribute Identifier Values are hexadecimal.

NOTE 2: Reporting is controlled in EtherNet/IP by the configuration of connection related object instances. See the EtherNet/IP specification for detail.

Table 4 Network Presentation of DM Object Services

Device Management Object Services			
CDM Tag	*Name (SEMI)*	*Name (S-DS)*	*S-DS Service ID (See Note 1.)*
DmS1	Reset	Reset	05
DmS2	Abort	Abort	4B
DmS3	Recover	Recover	4C
DmS4	Get Attribute	Get_Attribute _Single	0E
DmS5	Set Attribute	Set_Attribute _Single	10
DmS6	Execute	Start	06
DmS7	Perform Diagnostics	Perform _Diagnostics	4E

NOTE 1: Service Identifier Values are hexadecimal.

7.10 *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. The SAC object as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object attributes and services to the EtherNet/IP network shall be as indicated in Table 5 and Table 6 respectively. Each attribute and service is shown with its corresponding identifier (ID) for the S-DS Object.

Table 5 Network Presentation of SAC Object Attributes

SAC Object Services		
CDM Tag	*Name*	*S-DS Attribute ID (See Note 1.)*
SacA1	Last Calibration Date	13
SacA2	Next Calibration Date	14
SacA3	Expiration Timer	15
SacA4	Expiration Warning Enable	16
SacA5	Run Hours	17

NOTE 1: Attribute Identifier Values are hexadecimal.

Table 6 Network Presentation of SAC Object Services

SAC Object Services			
CDM Tag	*Name (SEMI)*	*Name (S-DS)*	*S-DS Service ID (See Note 1.)*
SacS1	Reset	Reset	05
SacS2	Abort	Abort	4B
SacS3	Recover	Recover	4C
SacS4	Get Attribute	Get_Attribute _Single	0E
SacS5	Set Attribute	Set_Attribute _Single	10
SacS6	Operate	Start	06
SacS7	Restore Default	N.A. (See Note 2.)	—
SacS8	Publish Attribute	N.A. (See Note 3.)	—

NOTE 1: Service Identifier Values are hexadecimal.

NOTE 2: Restoring defaults in EtherNet/IP is managed at the device level with services defined for the Identity Object.

NOTE 3: Reporting is controlled in EtherNet/IP by the configuration of connection related object instances. See the EtherNet/IP specification for detail.

8 Protocol Compliance

8.1 A method of testing protocol compliance is required to verify implementation conformance to the standard.

8.2 The compliance test suite for this protocol is documented along with the EtherNet/IP specification. ODVA has established an independent test center that perform a complete EtherNet/IP protocol compliance test suite. Visit <http://www.odva.org> for more information.

9 Specific Device Model Mappings

9.1 The following sections specify mappings for Sensor Actuator Network Specific Device Models.

9.2 *Mass Flow Device* — Reference SEMI E54.3 for a complete specification of the SDM for Mass Flow Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.2.1 *Objects* — Table 7 shows the mapping of the SDM objects specified in SEMI E54.3 and the EtherNet/IP objects.

Table 7 MFD Object Identifiers

SDM Object Name	*SDM Object ID*	*EIP Object Name*	*EIP ID (See Note 1.)*
Sensor-AI-MF	MFD3	S-Analog Sensor	31
Sensor-AI-AT	MFD4	S-Analog Sensor	31
Assembly-MFM	MFD5	Assembly	04
Sensor-AI-Aux	MFD6	S-Analog Sensor	31
Actuator-AO-MF	MFD7	S-Analog Actuator	32
Controller	MFD8	S-Single Stage Controller	33
Local Link	MFD9	N.A. (See Note 2.)	—
SISO	MFD10	N.A. (See Note 2.)	—
SISO Setpoint	MFD11	N.A. (See Note 2.)	—
Assembly-MFC	MFD12	Assembly	04

NOTE 1: Object Identifier Values are hexadecimal.

NOTE 2: The EtherNet/IP device model provides a different mechanism for data flow based on the configuration of EPATH type attributes. See the EtherNet/IP specification for detail.

9.2.1.1 Additional objects may be defined by the manufacturer in the Device Profile for a given device.

9.2.1.2 All objects listed in Table 7 provide a one-to-one correlation, with respect to Attributes and Services, between the SEMI SAN and EtherNet/IP. Therefore,

no mapping is necessary for attributes and services here in this standard.

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature, respecting any materials or equipment mentioned herein. These standards are subject to change without notice.

By publication of this standard, Semiconductor Equipment and Materials International (SEMI) takes no position respecting the validity of any patent rights or copyrights asserted in connection with any items mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights are entirely their own responsibility.

SEMI E54.14-0305
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR PROFINET

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the European Equipment Automation Committee. Current edition approved by the European Regional Standards Committee on November 10, 2004. Initially available at www.semi.org February 2005; to be published March 2005.

1 Purpose

1.1 This specification is part of the SEMI Sensor/Actuator Network (SAN) suite of standards and defines a specific communications protocol based on the PROFINET standard. This Network Communication Standard (NCS) taken together with the SEMI Sensor/Actuator Network standard suite and the PROFINET standard completely and unambiguously defines an open standard providing an industry specific solution to off-the-shelf interoperability of networked devices in semiconductor manufacturing equipment.

1.2 PROFINET is a vendor independent, open field bus standard for a wide range of applications in manufacturing, process and building automation.

1.3 The application model of PROFINET is compatible with the model of PROFIBUS. The same mapping rules apply.

1.4 PROFINET is optimized for high speed and inexpensive connectivity between automation control systems and distributed I/O at the device level.

2 Scope

2.1 This document specifies a SAN communications standard based on the PROFINET specification that is in compliance with SEMI E54.1. As such, it specifies the protocol, services, and behavior that compliant intelligent devices must support in order to interchange information over this SAN in a method compatible with SEMI E39.

2.2 In conjunction with a SEMI standard SAN Common Device Model (CDM) specification and one or more SEMI standard Specific Device Model (SDM) specifications (e.g., for a mass flow controller), this Network Communication Standard (NCS) with the related PROFINET standard describe the data structures, interactions, and behavior that are characteristic of the various devices on the network. This composite model forms a complete interoperability standard for communications among intelligent sensors, actuators, and controllers in semiconductor manufacturing equipment.

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 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on PROFINET and is a companion document to the PROFINET specification, including, by reference, the PROFINET standard; thus, a complete specification of this standard necessarily includes the PROFINET specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 The specifications within are strictly enhancements that provide additional capabilities over and above those currently required by PROFINET. Included throughout this document, primarily in §6, is information paraphrased from the PROFINET specifications such as: protocol structure, capabilities, options, and limitations. This information is provided here for reference only and is not intended to provide specification definitions. In all such areas, refer to the PROFINET specification documents for information. This document is limited to describing enhancements or limitations to the PROFINET specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the PROFINET protocol conformance testing specification. Conformance testing shall also include enhancements and limitations to the PROFINET specification required by this standard.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

SEMI E54.10 — Specification for Sensor/Actuator Network Specific Device Model for an In-Situ Particle Monitor Device

SEMI E54.11 — Specific Device Model for Endpoint Devices

4.2 *IEC Standards*[1]

IEC 61158-5 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Application Layer service definition (reference standard for the model in use)

IEC 61158-6 — Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Application Layer protocol specification (reference standard for the model in use)

4.3 *IEEE Standards*[2]

IEEE 802.3 — Telecommunications and information exchange between systems — Local and metropolitan area networks — Part 3:Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

IEEE 802.1D — Telecommunications and information exchange between systems — Local and metropolitan area networks — Common specifications — Media Access Control (MAC) Bridges

IEEE 802.1Q — Telecommunications and information exchange between systems — Local and metropolitan area networks — Common specifications —Virtual Bridged Local Area Networks

4.4 *ISO Standards*[3]

ISO 7498 Information Technology — Open Systems Interconnection — Basic Reference Model

4.5 *PROFINET Standards*[4]

PROFINET IO — Application Layer protocol specification

PROFINET IO — Application Layer service definition

PROFIBUS Profile Guidelines — Part 1: Identification & Maintenance Functions

PROFINET — Discovery and Configuration Protocol

PROFINET — Installation Guideline

GSD Specification for PROFINET

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *APDU* — Application Protocol Data Unit

---

1 International Electrotechnical Commission 3, rue de Varembé, Case Postale 131, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.919.02.11; Fax: 41.22.919.03.00, Website: www.iec.ch

2 IEEE , The Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, New York, NY 10016-5997, USA, Website: www.ieee.org

3 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30, Website: www.iso.ch

4 Profibus International, Haid-und-Neu-Str. 7,76131 Karlsruhe, Germany, Telephone: 49 721 9658 590; Fax: 49 721 9658 589, Website: www.profibus.com

5.1.2 *AREP* — Application Reference Endpoint

5.1.3 *ASE* — Application Service Element

5.1.4 *CDM* — Common Device Model

5.1.5 *DCE* — Distributed Computing Environment

5.1.6 *DCP* — Discovery and Configuration Protocol

5.1.7 *DHCP* — Dynamic Host Configuration Protocol

5.1.8 *DM* — Device Manager

5.1.9 *GSD* — Generic station description

5.1.10 *IM* — Identification and Maintenance

5.1.11 *IO* — Input Output

5.1.12 *IP* — Internet Protocol

5.1.13 *NCS* — Network Communication Standard

5.1.14 *OSI* — Basic Reference Model for Open Systems Interconnection (ISO 7498)

5.1.15 *PDU* — Protocol Data Unit

5.1.16 *PHY* — Physical Layer

5.1.17 *RPC* — Remote Procedure Call

5.1.18 *SAC* — Sensor, actor, controller (object)

5.1.19 *SAN* — Sensor/actor network

5.1.20 *SAP* — Service Access Point

5.1.21 *SDM* — Specific Device Model

5.1.22 *UDP* — User Datagram Protocol

5.2 *Terminology Defined in Sensor/Actuator Network Common Device Model (SEMI E54.1)*

5.2.1 *attribute*

5.2.2 *behavior*

5.2.3 *byte*

5.2.4 *common device model*

5.2.5 *device*

5.2.6 *Device Manager (DM) Object*

5.2.7 *device model*

5.2.8 *instance*

5.2.9 *network communication standard*

5.2.10 *object*

5.2.11 *Sensor, Actuator and Controller (SAC) Object*

5.2.12 *service*

5.2.13 *specific device model*

5.2.14 *state diagram*

5.3 *Terminology Mapping*

5.3.1 As this standard defines the mapping of CDM data structure and behavior over a network it makes use of many of the terms in SEMI E54.1. Table 1 provides a mapping of fundamental terminology of the CDM document into this document which uses the terminology of PROFINET IO.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*
Device	IO Device
Object	=(with ASE as class specification)
Instance	=
Attribute	=
Behavior	=
Service	=
State Diagram	Protocol Machine, State Machine
Byte	=, Unsigned8
Nibble	Specific field coding
Character String	Visible String

5.4 *PROFINET Specific Definitions*

5.4.1 *Context Management* — network-accessible information (communication objects) that supports managing the operation of the system, including the application layer.

NOTE 1: Managing includes functions such as controlling, monitoring, and diagnosing.

5.4.2 *Device Data Base* — an electronic file that provides a clear and comprehensive description of the characteristics of a device type in a precisely defined format. Also called a GSD File.

5.4.3 *Device Profile* — a Device Data Base Sheet, which specifies the characteristic features of a device, and a GSD File.

5.4.4 *channel* — single physical or logical link of an input or output application object of a server to the process.

5.4.5 *channel related diagnosis* — information concerning a specific element of an input or output application object, provided for maintenance purposes.

5.4.6 *diagnosis data object* — object(s) which contains diagnosis information referenced by device/slot/subslot/diagnosis identifier.

5.4.7 *GSD File* — see Device Data Base.

5.4.8 *Index* — address of a record data object.

5.4.9 *IO Controller* — a device that manages its assigned IO Devices and handles user data exchange; usually a programmable controller.

5.4.10 *IO data object* — object designated to be transferred cyclically for the purpose of processing and referenced by device/slot/Subslot.

5.4.11 *IO Device* — a device that is configured and managed by IO Controllers and IO Supervisors; an IO Device initiates no unsolicited communications.

5.4.12 *IO Supervisor* — a device that interacts as a configuration or diagnostic tool; usually a programming device.

5.4.13 *module* — hardware or logical component of a physical device.

5.4.14 *provider* — node or source sending data to one or many consumers.

5.4.15 *record data object* — object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing and referenced by device/slot/subslot/index.

5.4.16 *resource* — processing or information capability.

5.4.17 *server* — a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request — b) object which provides services to another (client) object.

5.4.18 *service* — operation or function that an object and/or object class performs upon request from another object and/or object class.

5.4.19 *Service Access Point* — an addressable location in a device for the directing of service requests.

5.4.20 *slot* — address of a structural unit within an IO device.

NOTE 2: Within a modular device, a slot typically addresses a physical module. Within compact devices, a slot typically addresses a logical function or virtual module.

5.4.21 *slot related diagnosis* — information dedicated to modules for maintenance purpose.

5.4.22 *submodule* — hardware or logical component of a module.

5.4.23 *subslot* — address of a structural unit within a slot.

NOTE 3: A subslot may address a physical interface for submodules within a module. Generally, a subslot is a second level to structure data within a device.

5.4.24 *vendor ident number* — central administrative number assigned by the PNO.

6 Communication Protocol High Level Structure

6.1 The PROFINET IO protocol constitutes a collapsed form of the OSI seven layer architecture. PROFINET IO uses physical, data link, network, transport, and application layers of the Reference Model. For real time communication a three layer approach with physical, data link and application is specified. This section has been formatted to be aligned with the Basic Reference Model for OSI. Figure 1 gives an overview of the architecture.

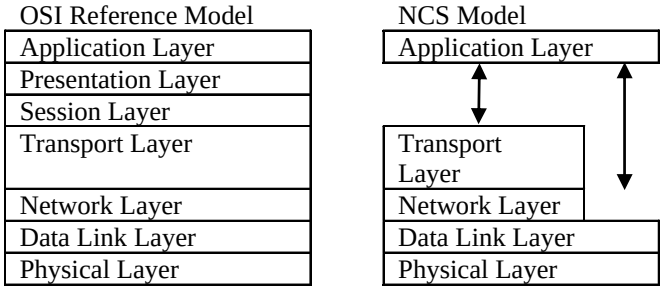


Figure 1
Layering of PROFINET in Relation to OSI

6.2 *General Behavior*

6.2.1 In a typical remote I/O configuration, single IO controller architectures are used to optimize response times. In lower speed applications, multi IO controller architectures are also possible.

6.2.2 Message transfer is organized in cycles. A message cycle mainly consists of a set of Output Data Frames of the IO controller and a set of Input Data Frames of several IO devices. Every Output Data Frame is associated to an Input Data Frame. The monitoring of an IO application relationship is done by the receiver of that frames.

6.2.3 A brief description of the PROFINET protocol as it relates to the ISO 7498 OSI model follows in the sections below. For protocol efficiency, PROFINET does not define layers 3 to 7 for cyclic IO data transfer and alarms. Layer 7 is the interface between the Application Process and the communication stack.

NOTE 4: The information contained in this section is for reference only. It in no way represents specifications for PROFINET. See related documentation for these specifications.

6.3 *Physical Layer — Layer 1*

6.3.1 The Physical Layer of IEEE 802.3 is adopted. There are two recommendations specified for the Physical Layer (PHY): twisted pair (100 Base TX) and optical (100 Base FX) in PROFINET. See IEEE 802.3 standard for more information about these options. More Details are specified in the PROFINET Installation Guideline. If an accepted physical standard other than IEEE 802.3 is being used it has be clearly specified in the product documentation.

6.4 *Data Link Layer — Layer 2*

6.4.1 *Data Transfer*

6.4.1.1 The Data Link Layer provides the functions for sending and receiving data over the network. Protocol Data Units (PDU) are packaged, delivered, and checked. Checks are used to guard against Line Protocol Errors (e.g., frame, overrun, and coding violations). IEEE 802.3 Data Link MAC sublayer is the preferred Data link technology of PROFINET. If an accepted data link standard other than IEEE 802.3 is being used it has to be clearly specified in the product documentation.

6.4.1.2 A PDU is restricted to 1500 bytes. A protocol overhead of 28 Bytes is needed for addressing, error detection and protocol selection.

6.4.2 *Conveyance of Data*

6.4.2.1 In IEEE 802 networks bridges (switches) are used to transport data. These bridge functions are defined in IEEE 802.1D and IEEE 802.1Q (priority options). Real time extensions can be used to enhance performance and availability of bridged networks.

6.5 *Network Layer — Layer 3*

6.5.1 While real time data transfer uses a protocol with no network layer, all other services are using IP (Internet Protocol) as network layer.

6.6 *Transport Layer — Layer 4*

6.6.1 All non-real time services are using UDP (User Datagram Protocol) as transport layer.

6.7 *Session Layer — Layer 5*

6.7.1 There is no distinct Session Layer in this CDM.

6.8 *Presentation Layer — Layer 6*

6.8.1 There is no distinct Presentation Layer in this CDM. The Encoding is part of the application layer.

6.9 *Application Layer — Layer 7*

6.9.1 The PROFINET IO application layer is structured in a so called service definition and protocol specification. The service definition uses an object orientated approach and specifies the services for remote access and local functions together with their objects (the ASE is a class definition of these objects). The protocol specification includes both coding and state machines.

6.9.2 *Service Definition*

6.9.2.1 Application layer services are structured to reflect the needs of flexible configurable automation devices. A Device consists of a set of modules that are placed in slots (see Figure 2). Modules are addressed uniquely by the slot number. The module view can be a hardware oriented or reflects the software structure of the IO Device.

6.9.2.2 Each module contains submodules that contain objects of different classes. Submodules are the addresses of the submodules.

6.9.2.3 Slot 0 is used to address the IO Device itself. Subslot 0 represents the module and contains no IO Data. The other object classes can have instances scattered over the modules and their submodules addressed by slots and their subslots respectively. Each submodule (except submodule 0) can contain IO Data, Context parameter, Diagnosis information, Record Data and Alarms. Record Data is a generic class which can contain different application specific parameters accessible by read and write services. There is a set of identification and maintenance parameters defined in a PROFIBUS Guideline that are mandatory within PROFINET.

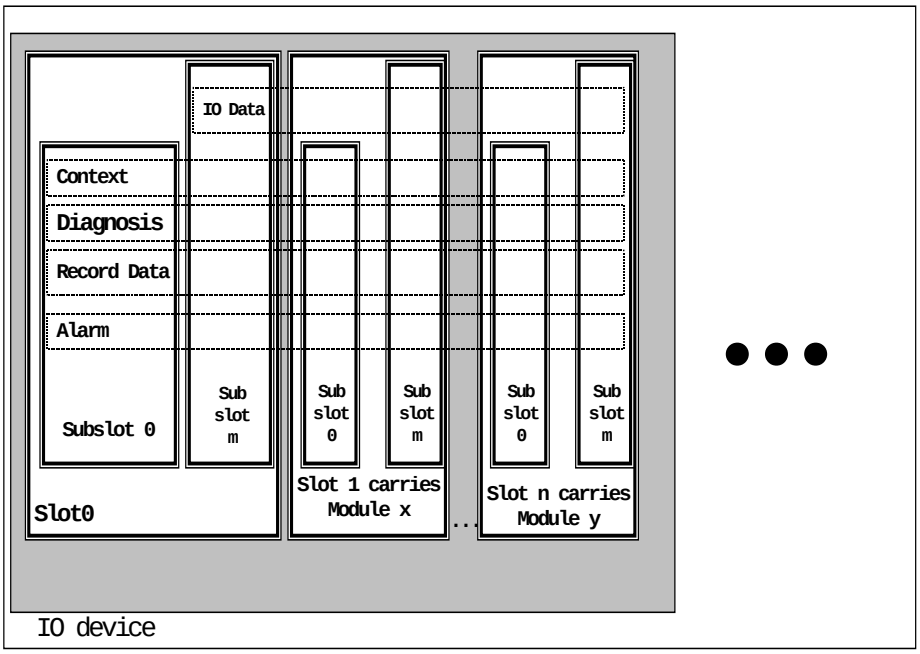


Figure 2
Device Model

6.9.2.4 Object classes are defined for:

- IO Data for periodic reporting,
- Context for configuration,
- Diagnosis for event collection,
- Record Data for polled access, and
- Alarms for asynchronous event reporting.

A set of services are defined for these object classes.

6.9.2.5 IO Data are handled mainly by buffered services which allow decoupling between application and communication. Client/Server service structure (request/response) is used for Context for configuration, Diagnosis for event collection, Record Data for polled access and Alarms for asynchronous event reporting.

6.9.2.6 Cyclic functionalities offered in PROFINET IO:

- exchange of IO data with related IO devices

6.9.2.7 Acyclic functionalities offered in PROFINET IO:

- read diagnosis from IO devices
- configuration of IO devices
- write parameter data to IO devices (startup or application parameter)
- treatment of configuration and diagnosis requests of an engineering device
- initiate connections to IO devices by means of context management
- acyclic access to record data of IO devices
- treatment of alarms from IO devices

- sending of alarms to IO devices

6.9.3 *Protocol Specification*

6.9.3.1 An IO device has no IP-address and logical name when it is shipped.

6.9.3.2 A set up procedure has to allocate name and IP-address prior to operation.

6.9.3.3 To access an IO Device a controller has to establish the context first with a connect service. A check is done to ensure that the appropriate device type with the required resources is accessed. Resources can be locked with the connect service.

6.9.3.4 After a positive confirmation of the connect service the IO device may be loaded with the parameter required by the controller application (e.g. warning limits, measure range, filter time). A control service is issued at the end of parameterization.

6.9.3.5 There is a control service indicating that the IO Device application is ready to enter the operate state. The start up is completed with this service.

6.9.3.6 After this start up procedure the data exchange of IO data can be done and alarms can be signaled from the IO Device to the IO Controller. The IO Controller can invoke services to read and write record data that contain all kind of information e.g. produced units, calibration information, and batch information.

6.9.3.7 A Release service will terminate the context.

6.9.3.8 A context is monitored by the receipt of Input Data at the IO controller and by the receipt of Output Data at the IO device site. A timeout error of one timer unit is allowed which makes it possible to use a single timer resource with granularity in the range of a millisecond.

6.9.3.9 Figure 3 shows a sequence diagram of the start up procedure.

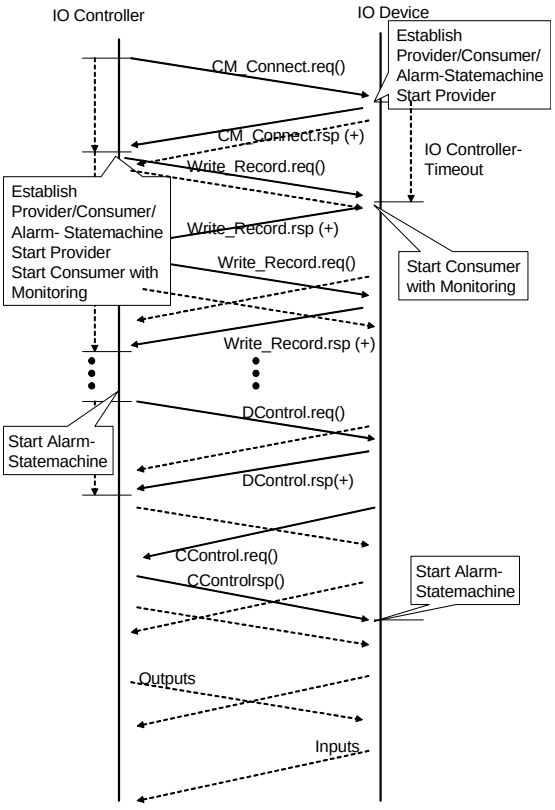


Figure 3
Start Up Sequence

6.9.4 *Communication Management*

6.9.4.1 DCP (Discovery and Configuration Protocol) is used for basic management of stations (especially setting of IP-address and Device name). DHCP (Dynamic Host Configuration Protocol) can be used as configuration option.

6.9.4.2 Connectionless DCE RPC is used as an interface between transport and application.

6.9.4.3 "Identification & Maintenance Functions" (I&M functions) define general parameters and protocols. The main purpose of the I&M functions is to support the end user during various scenarios of a device's life cycle be it configuration, commissioning, parameterization, diagnosis, repair, firmware update, asset management, audit trailing, and alike. Well-defined uniform parameters and rules should enable the manufacturers to offer devices that behave in a uniform manner These profile guidelines take into account requirements from FDA (Food & Drug Administration) and others. The basic information offered can be characterized as "Type Plate" or "Boiler Plate". It consists of Manufacturer ID, Order ID, Serial Number, Hardware Revision, Software Revision, Revision Counter, Profile ID.

7 Required Object Types

7.1 This section describes a general mapping of the SEMI SAN Object Model to the PROFINET environment. Component definitions are clarified and the mapping of Attributes, Services, and Behaviors are specified.

7.2 *Object Model*

7.2.1 The Object Model defined in the CDM is represented in the PROFINET NCS. Specifically, the DM and SAC objects are mapped.

7.2.2 The CDM Objects are mapped to slots. These slots have a unique address space within the device. The Specific Application Objects are mapped in PROFINET Device Data Base. §9 specifies the mapping of SDM Objects in PROFINET.

7.3 *Component Mapping Summary*

7.3.1 Figure 4 provides a summary of the components of the CDM object model as they relate to the components of PROFINET.

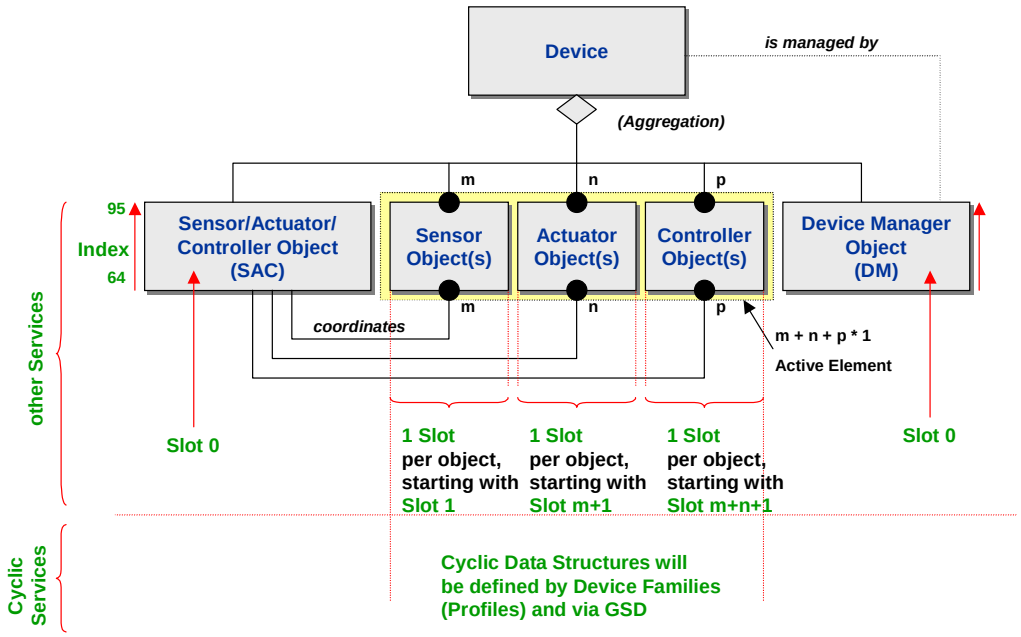


Figure 4
Component Mapping Summary

7.4 *Objects*

7.4.1 The required objects of the CDM are identified here. Additional objects that are contained in the SDM are given identifiers in the Device Profile. §9 specifies additional mapping information.

7.4.2 Table 2 lists the Object Identifiers specified for use in protocol messages. Slot numbers is the mechanism used for addressing Objects within PROFINET. Subslot IDs can be used for sub addressing. Within this NCS the subslot 1 shall be used as default subslot.

7.4.3 Assembly Objects and Local Link Objects which are objects embedded in the SAC Object are not explicitly mapped in PROFINET.

Table 2 Object Identifiers

Slot ID	*Index*	*Object*
0	IM0-17 96	DM Object Attributes DM Control
0	64 65-95	SAC Control SAC Object Attributes
1–z	64 65-254	Application Object Control Application Objects Attributes as specified in §7 and §9

7.5 *Attributes*

7.5.1 *General*

7.5.1.1 All attributes are accessible via Get_Attribute and Set_Attribute services defined in the sections below. The Get_Attribute and Set_Attribute services shall be mapped to PROFINET Read and Write services for Record Data objects. Additionally, attributes are accessible via different PROFINET defined methods that are mapped in this document based on attribute type.

7.5.2 *DM Attributes*

7.5.2.1 All attributes are communicated with Read and Write Service and mapped to IM Blocks. Status attributes are modeled additionally as Diagnosis. See Table 17 for detailed mapping.

7.5.2.2 Table 3 shows the DM status attribute mapping to Diagnosis.

Table 3 Diagnosis Mapping of DM Status

Parameter	*Value*
Slot	0 = Device
Subslot	1 = Device
Channel	0 = Global
Channel Type	0 = unspecified
IO Type	0 = unspecified
Error Type	9 = Error with Bit0 - 6 set in Common Exception Detail 1 21 = Error Calibration set in Common Exception Detail Attribute 1 17 = Error with Bit set in Common Exception Detail 2

7.5.3 *SAC Attributes*

7.5.3.1 The attributes of SAC are mapped to Record data objects of PROFINET.

7.5.3.2 The Slot number used is 0. The index is the numeric value of the SAC Attribute identifier with an offset of 64 (see Table 4).

Table 4 SAC Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail*
SacA1	65	Last Calibration Date	
SacA2	66	Next Calibration Date	
SacA3	67	Expiration Timer	
SacA4	68	Expiration Warning Enable	
SacA5	69	Run Hours	

7.5.4 *Active Element Attributes*

7.5.4.1 The attributes of Active Elements are mapped to Record data objects of PROFINET. These attributes are contained in any Sensor, Actuator and Controller Object.

7.5.4.2 The Slot number used is the instance number in the device. The index is the numeric value of the Active Element Attribute identifier with an offset of 64 (see Table 5).

Table 5 Active Element Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
nA1	65	Name	
nA2	66	Status	
nA3	67	Alarm Enable	
nA4	68	Warning Enable	

7.5.5 *Sensor Attributes*

7.5.5.1 The attributes of Sensors are mapped to Record data objects of PROFINET. These attributes are contained in any Sensor-AI, Sensor-EI and Sensor-BI Object.

7.5.5.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Attribute identifier with an offset of 64 (see Table 6).

Table 6 Sensor Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
nA16	80	Value	Can be mapped as Input Data Object
nA17	81	Report Inhibit Time	
nA18	82	Enable Report Rate	
nA19	83	Report Rate	

7.5.6 *Actuator Attributes*

7.5.6.1 The attributes of Actuators are mapped to Record data objects of PROFINET. These attributes are contained in any Actuator-AO, Actuator-EO and Actuator-BO Object.

7.5.6.2 The Slot number used is the instance number in the device. The index is the numeric value of the Actuator Attribute identifier with an offset of 64 (see Table 7).

Table 7 Actuator Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
nA16	80	Setting	Can be mapped as Output Data Object
nA17	81	Safe State	
nA18	82	Watch Rate	
nA19	83	Watch Dog	The value of this variable reflects the WD_Factor para-meter of PROFINET

7.5.7 *Controller Attributes*

7.5.7.1 The attributes of Controllers are mapped to Record data objects of PROFINET.

7.5.7.2 The Slot number used is the instance number in the device. The index is the numeric value of the Controller Attribute identifier with an offset of 64 (see Table 8).

Table 8 Controller Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
CA16	80	Setpoint	
CA17	81	Process Variable	
CA18	82	Control Variable	
CA19	83	Data Type	
CA20	84	Data Unit	
CA21	85	Alarm Settle Time	
CA22	86	Alarm Error Band	
CA24	88	Warning Settle Time	
CA25	89	Warning Error Band	

7.5.8 *Sensor Analog Input Attributes*

7.5.8.1 The attributes of Sensor Analog Inputs are mapped to Record data objects of PROFINET.

7.5.8.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Analog Input Attribute identifier with an offset of 64 (see Table 9).

Table 9 Sensor Analog Input Object Attribute Identifier

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
SaiA64	128	Offset	
SaiA65	129	Gain	
SaiA66	130	Data Type	
SaiA67	131	Data Units	
SaiA68	132	Safe State	
SaiA69	133	Enable Report Delta	
SaiA70	134	Report Delta	
SaiA71	135	Enable Report ROC	
SaiA72	136	Report ROC	
SaiA73	137	Alarm Trip Point High	
SaiA74	138	Alarm Trip Point Low	
SaiA75	139	Alarm Hysteresis	
SaiA76	140	Warning Trip Point High	
SaiA77	141	Warning Trip Point Low	
SaiA78	142	Warning Hysteresis	

7.5.9 *Sensor Binary Input Attributes*

7.5.9.1 The attributes of Sensor Binary Inputs are mapped to Record data objects of PROFINET.

7.5.9.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Binary Input Attribute identifier with an offset of 64 (see Table 10).

Table 10 Sensor Binary Input Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
SbiA64	128	Debounce Control	
SbiA65	129	Alarm State	
SbiA66	130	Warning State	

7.5.10 *Sensor Enumerated Input Attributes*

7.5.10.1 The attributes of Sensor Enumerated Inputs are mapped to Record data objects of PROFINET.

7.5.10.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Enumerated Input Attribute identifier with an offset of 64 (see Table 11).

Table 11 Sensor Enumerated Input Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
SeiA64	128	Debounce Control	
SeiA65	129	Alarm State	
SeiA66	130	Warning State	

7.5.11 *Actuator Analog Output Attributes*

7.5.11.1 The attributes of Actuator Analog Outputs are mapped to Record data objects of PROFINET.

7.5.11.2 The Slot number used is the instance number in the device. The index is the numeric value of the Actuator Analog Output Attribute identifier with an offset of 64 (see Table 12).

Table 12 Actuator Analog Output Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
AaoA64	128	Offset	
AaoA65	129	Gain	
AaoA66	130	Data Type	
AaoA67	131	Data Units	

7.5.12 *Sensor Binary Input Threshold Attributes*

7.5.12.1 The attributes of Sensor Binary Input Thresholds are mapped to Record data objects of PROFINET.

7.5.12.2 The Slot number used is the instance number in the device. The index is the numeric value of the Sensor Binary Input Threshold Attribute identifier with an offset of 80 (see Table 13). The different mapping has to be done because there is an Overlap to the Sensor Binary Input Attributes enumeration part.

Table 13 Sensor Binary Input Threshold Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Index*	*Attribute*	*Detail/Alternative Access Method*
SbithA64	144	Reading Valid	
SbithA65	145	State	
SbithA66	146	Status	

7.6 *Services*

7.6.1 *DM Services*

7.6.1.1 Get_Attribute and Set_Attribute are mapped to Read and Write services to the IM Objects.

7.6.1.2 The other services are mapped to a write at an UINT Object addressed with Index 96 as described in Table 2 and Table 14.

7.6.1.3 A read to Index 96 will return the last command executed successfully.

Table 14 DM Service Mapping

SEMI CDM Service ID	*Service*	*Write Value to Index 96*
DmS4	Get Attribute	Mapped as Read from Attribute Index
DmS5	Set Attribute	Mapped as Write to Attribute Index
DmS1	Reset	0
DmS2	Abort	1
DmS3	Recover	2
DmS6	Execute	3
DmS7	Perform Diagnostics	4
DmS8 – DmS12		Optional Services are not mapped in PROFINET

7.6.2 *SAC Services*

7.6.2.1 Get_Attribute and Set_Attribute are mapped to Read and Write services with Slot and Index as described in ¶¶ 7.2 and 7.4.

7.6.2.2 The other services are mapped to a write at an UINT Object addressed with Index 64 as described in Table 2 and Table 15.

7.6.2.3 A read to Index 64 will return the last command executed successfully (Exception: Restore Default is not mirrored).

Table 15 SAC Service Mapping

SEMI CDM Service ID	*Service*	*Write Value to Index 64*
SacS4	Get Attribute	Mapped as Read from Attribute Index
SacS5	Set Attribute	Mapped as Write to Attribute Index
SacS1	Reset	0
SacS2	Abort	1
SacS3	Recover	2
SacS6	Operate	3
SacS7	Restore Default	4
SacS8	Publish Attribute	Optional Service not mapped in PROFINET

7.6.3 *Active Element Services*

7.6.3.1 Get_Attribute and Set_Attribute are mapped to Read and Write services with Slot and Index as described in ¶¶ 7.2 and 7.4.

7.6.3.2 The other services are mapped to a write at an UINT Object addressed with Index 64 as described in Table 2 and Table 16.

7.6.3.3 A read to Index 64 will return the last command executed successfully (Exception: Restore Default is not mirrored).

Table 16 Active Element Service Mapping

SEMI CDM Service ID	*Service*	*Write Value to Index 64*
nS4	Get Attribute	Mapped as Read from Attribute Index
nS5	Set Attribute	Mapped as Write to Attribute Index
nS1	Reset	0
nS2	Abort	1
nS3	Recover	2
nS6	Operate	3
nS7	Restore Default	4

7.7 *PROFINET Device Data Base*

7.7.1 The specification of the Device Data Base (or GSD file) for a given SDM is beyond the scope of this document. The PROFIBUS Trade Organization is responsible for the management of these files.

Table 17 DM Object Attribute Identifiers

SEMI CDM Attribute ID	*PROFINET Attribute ID*	*Attribute*	*Detail/Alternative Access Method*
DmA1	IM0, DEVICE_ID	Device Type	8 Octets significant
DmA2	IM0, REVISION_COUNTER	Standard Revision Level	Shall be an UINT type
DmA3	IM0, MANUFACTURER_ID	Device Manufacturer Identifier	Shall be an UINT type, the number is resolved via file from www.profibus.com to supply with web pages and phone numbers
DmA4	IM0, ORDER_ID	Manufacturer Model Number	
DmA5	IM0, SOFTWARE_REVISION	Software or Firmware Revision Level	
DmA6	IM0, HARDWARE_REVISION	Hardware Revision Level	
DmA7	IM0, SERIAL_NUMBER	Serial Number	16 Octets maximum
DmA8	IM3	Device Configuration	
DmA9	IM17, octet 10-11	Device Status	Additional as Status Diagnosis
DmA10	IM16, octet 10-11	Reporting Mode	
DmA11	IM16, octet 12-13	Exception Status Report Interval	
DmA12	IM17, octet 12-13	Exception Status	Additional as Status Diagnosis
DmA13	Slot 0 Index 0xe00b	Exception Detail Alarm	Additional as Diagnosis Alarm
DmA14	Slot 0 Index 0xe00b	Exception Detail Warning	Additional as Diagnosis Alarm
DmA15	Optional Attribute not mapped	Visual Indicator	N/A
DmA16	Context	Alarm Enable	Put into a Parameter Data Record
DmA17	Context	Warning Enable	Put into a Parameter Data Record
DmA18	Optional Attribute not mapped	Exception Detail Type	N/A
DmA19	Optional Attribute not mapped	Exception Detail Alarm Queue	N/A
DmA20	Optional Attribute not mapped	Exception Detail Warning Queue	N/A
DmA21	Optional Attribute not mapped	Date and Time	N/A
DmA22	Optional Attribute not mapped	Date and Time Type	N/A
DmA23-DmA31	Reserved	Reserved	N/A

8 Protocol Compliance

8.1 PROFIBUS International has established a qualified certification system, with test laboratories in Europe, Asia and North America, which includes conformance testing and interoperability testing (address list can be found at

http://www.PROFIBUS.com/support.html[5]). Certified products are listed with their certificate number in the Electronic Product Guide.

8.2 GSD files of all PROFINET devices that are tested for their conformity to the PROFIBUS standard are available in the GSD library on the World Wide Web Server of the PROFIBUS User Organization at http://www.PROFIBUS.com.

9 Specific Device Model Mappings

9.1 Every type of device shall have an identifier number. Vendors must apply for an identifier number from the PROFIBUS User Organization. A Device Profile shall be submitted in the form of a GSD File and Device Data Base Sheet.

9.2 The Device Profile must specify the identifiers for Objects, Attributes and Services for CDM and SDM components, including data formats and bit mappings for specified parameters, as represented in this document.

9.3 The following sections specify mappings for Sensor Actuator Network Specific Device Models.

9.4 *Mass Flow Device*

9.4.1 Reference SEMI E54.3 for a complete specification of the SDM for Mass Flow Devices (MFD).

9.4.2 *Objects*

9.4.2.1 Consistent with SEMI E54.3 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.4.2.2 Notice that references for the Local Link Objects are not included; the existence of these objects are implied by behavior and not explicitly included. Therefore, these objects are not accessible from the network.

9.4.2.3 Table 18 shows the mapping of the SDM.

9.4.2.4 Objects specified in SEMI E54.3 are listed under the heading NCS Module ID. Every instance is allocated to a slot number. The configuration can be modeled in the GSD by using the Module ID as key for the module description.

Table 18 Mass Flow Device Object Identifiers

SDM Object Name	*SDM Object ID*	*NCS Module ID*
Sensor-AI-MF	MFD3	103
Sensor-AI-AT	MFD4	104
Assembly-MFM	MFD5	105
Sensor-AI-Aux	MFD6	106
Actuator-AO-MF	MFD7	107
Controller	MFD8	108
Local Link	MFD9	109
SISO	MFD10	110
SISO-Setpoint	MFD11	111
Assembly-MFC	MFD12	112

9.4.3 *Attributes*

9.4.3.1 The mapping of Attribute Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.4.3.2 Tables 19–23 show the mapping of the specific SDM attributes.

5 Website maintained by PROFIBUS International.

Table 19 Sensor AI-MF Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
A1	160	Flow Totalizer
A2	161	Flow Hours
A5	164	Zero Offset Mode
A6	165	Zeroing Status
A7	166	Autorange Status

Table 20 Actuator AO-MF Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
A1	160	Valve Type
A2	161	Override

Table 21 Controller Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
A1	160	Valve Type
A2	161	Override

Table 22 SISO Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
A1	65	Input
A2	66	Output
A3	67	Data Type

Table 23 SISO Setpoint Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
A33	97	Ramp Type
A34	98	Ramp Rate
A35	99	Ratio

9.4.4 *Services*

9.4.4.1 The mapping of Service Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.4.4.2 The additional Services of Sensor-AI-MF are mapped as described in Table 24.

Table 24 Sensor-AI-MF Additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
S1	Perform Zero Offset	11
S2	Query Supported Gas Type	12
S3	Select Programmed Gas Type	13
S4	Insert Gas Type	14
S5	Delete Gas Type	15
S6	Get Gas Calibration Data Value	16
S7	Set Gas Calibration Data Value	17
S8	Autorange	18

9.5 *In-Situ Particle Monitor Device*

9.5.1 Reference SEMI E54.10 for a complete specification of the SDM for In-Situ Particle Monitor Device (ISPM).

9.5.2 *Objects*

9.5.2.1 Consistent with SEMI E54.10 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.5.2.2 Notice that references for the Local Link Objects are not included; the existence of these objects are implied by behavior and not explicitly included. Therefore, these objects are not accessible from the network.

9.5.2.3 Table 25 shows the mapping of the SDM.

9.5.2.4 Objects specified in SEMI E54.10 are listed under the heading NCS Module ID. Every instance is allocated to a slot number. The configuration can be modeled in the GSD by using the Module ID as key for the module description.

Table 25 In-Situ Particle Monitor Device Object Identifiers

SDM Object Name	*SDM Object ID*	*NCS Module ID*
Sensor-AI-LCS	ISPMD03	203
Sensor-AI-SLS	ISPMD04	204
Sensor-AI- MNS	ISPMD05	205
Sensor-AI-Counter	ISPMD16	216
Assembly-ISPM#1	ISPMD17	217
Assembly-ISPM#2	ISPMD18	218
Assembly-ISPM#3	ISPMD19	219
Assembly-ISPM#4	ISPMD20	220
Assembly-ISPM#5	ISPMD21	221
Assembly-ISPM#6	ISPMD22	222
Assembly-ISPM#7	ISPMD23	223
Assembly-ISPM#8	ISPMD24	224
Assembly-ISPM#9	ISPMD25	225

9.5.3 *Attributes*

9.5.3.1 The mapping of Attribute Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.5.3.2 Tables 26–31 show the mapping of the specific SDM attributes.

Table 26 DM Object Additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
DmA33	96	Gain
DmA34	97	Filter Bandwidth
DmA35	98	Tool State
DmA36	99	Laser Status
DmA37	100	Flow Path
DmA38	101	Volume
DmA39	102	Volume Units
DmA40	103	Leak Status
DmA41	104	Time Stamp

Table 27 SAC Object Additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
SacA65	129	Number of Bins
SacA66	130	Count Mode
SacA67	131	Duration

Table 28 Sensor AI-LCS Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
LcsA1	160	Reading Valid
LcsA2	161	Full Scale
LcsA3	162	Alarm Settling Time
LcsA4	163	Warning Settling Time

Table 29 Sensor AI-SLS Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
SlsA1	160	Reading Valid
SlsA2	161	Full Scale
SlsA3	162	Alarm Settling Time
SlsA4	163	Warning Settling Time

Table 30 Sensor AI-MNS Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
MnsA1	160	Reading Valid
MnsA2	161	Full Scale
MnsA3	162	Alarm Settling Time
MnsA4	163	Warning Settling Time

Table 31 Sensor AI-Counter Object Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
CounterA1	160	Reading Valid
CounterA2	161	Full Scale
CounterA3	162	Alarm Settling Time
CounterA4	163	Warning Settling Time
CounterA5	164	Upper Size
CounterA6	165	Lower Size

9.5.4 *Services*

9.5.4.1 The mapping of Service Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.5.4.2 The additional Services of DM are mapped as described in Table 32.

Table 32 DM Additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 96*
DmS1	Laser On	11
DmS2	Laser Off	12

9.5.4.3 The additional Services of SAC are mapped as described in Table 33.

Table 33 SAC Additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
SacS1	Clear Counts	11

9.6 *Endpoint Device*

9.6.1 Reference SEMI E54.11 for a complete specification of the SDM for Endpoint Devices.

9.6.2 Date-and-Time data structure shall be mapped to PROFINET data type Time Of Day. An Offset of 4383 has to be added to Time Of Day day value to obtain the number of days since 1/1/72 (Time Of Day base is 1/1/84).

9.6.3 *Objects*

9.6.3.1 Consistent with SEMI E54.11 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.6.3.2 Notice that references for the Local Link Objects are not included; the existence of these objects are implied by behavior and not explicitly included. Therefore, these objects are not accessible from the network.

9.6.3.3 Table 34 shows the mapping of the SDM.

9.6.3.4 Objects specified in SEMI E54.11 are listed under the heading NCS Module ID. Every instance is allocated to a slot number. The configuration can be modeled in the GSD by using the Module ID as key for the module description.

9.6.3.5 The Sensor-BI-TH-EP can occur more than once. This shall be modeled in GSD in that way, that this Module ID can be placed in several slots.

Table 34 Endpoint Device Object Identifiers

SDM Object Name	*SDM Object ID*	*NCS Module ID*
Sensor-BI-TH-EP	EPD3	203
Assembly-EPD#1	EPD4	204
Assembly- EPD#2	EPD5	205
Assembly- EPD#3	EPD6	206
Assembly- EPD#4	EPD7	207

9.6.4 *Attributes*

9.6.4.1 The mapping of Attribute Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.6.4.2 Tables 35 and 36 show the mapping of the specific SDM attributes.

Table 35 DM Object Additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
SacA65	129	Number of Endpoint Objects

Table 36 Sensor-BI-TH-EP Object Additional Attribute Identifiers

SEMI SDM Attribute ID	*PROFINET Index*	*Attribute*
EpA1	160	Minimum Time
EpA2	161	Maximum Time
EpA3	162	Target Time
EpA4	163	Elapsed Time
EpA5	164	Time Stamp
EpA6	165	Recipe Identifier
EpA7	166	Step Identifier

9.6.5 *Services*

9.6.5.1 The mapping of Service Tags and Identifiers is defined in §7 for SAC. The same method applies here for the SDM.

9.6.5.2 The additional Services of SAC are mapped as described in Table 37.

Table 37 SAC Additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
SacS33	Reset Endpoint	33
SacS34	Download Recipe	34
SacS35	Upload Recipe	35
SacS36	Calibrate	36

9.6.5.3 The additional Services of Sensor BI-EP are mapped as described in Table 38.

Table 38 Sensor BI-EP Additional Service Mapping

SEMI SDM Service ID	*Service*	*Write Value to Index 64*
EpS1	Endpoint On	11
EpS2	Endpoint Off	12
EpS3	Endpoint Start	13
EpS4	Endpoint Suspend	14
EpS5	Endpoint Resume	15

10 Related Documents

10.1 *IEEE Standards*[6]

IEEE 802.3 — Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

IEEE 802.1D — Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common Specification – Media Access Control (MAC) Bridges

IEEE 802.1Q — Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common Specification – Virtual Bridged Local Area Networks

10.2 *Internet Engineering Task Force Standards*[7]

RFC 768 — UDP: User Datagram Protocol; IETF, available at <http://www.ietf.org>

RFC 791 — IP: Internet Protocol; IETF, available at <http://www.ietf.org>

RFC 1541 — DHCP: Dynamic Host Configuration Protocol; IETF, available at <http://www.ietf.org>

10.3 *Other Documents*

C706, CAE Specification DCE1.1: Remote Procedure Call (RPC); OSF available at <http://www.opengroup.org/onlinepubs/9629399/toc.htm>

6 Institute of Electrical and Electronics Engineers, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, New Jersey 08855-1331, USA. Telephone: 732.981.0060; Fax: 732.981.1721, Website: www.ieee.org

7 Internet Engineering Task Force, c/o Corporation for National Research Initiatives 1895 Preston White Drive, Suite 100 Reston, VA 20191-5434, USA. Telephone: 1.703.620.8990, Fax 1.703.620.9071, Website: www.ietf.org

SEMI E54.15-0305
SENSOR/ACTUATOR NETWORK COMMUNICATION SPECIFICATION FOR SafetyBUS p

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on December 10, 2005. Initially available at www.semi.org February 2005; to be published March 2005.

1 Purpose

1.1 This standard defines a communication specification based on the SafetyBUS p protocol to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI specified device models (common and device specific) in a semiconductor manufacturing equipment.

1.2 *Background and Motivation*

1.2.1 SafetyBUS p is a device level network which provides a simple, inexpensive, fast, and deterministic means of exchanging data among industrial devices (e.g., sensors and actuators) and higher level devices such as controllers (e.g., programmable control systems) while at the same time providing for the communication of information at a high priority level to support reliable operations. SafetyBUS p provides:

- A solution to low-level device networking,
- Access to intelligence present in low-level devices,
- Networking between higher level controllers,
- Master/Slave and Peer-to-Peer communication capabilities, and
- Serial communication of device-critical information.

1.2.2 SafetyBUS p specifies a communication model and protocol. The Physical and Data Link Layer definitions are defined by the Controller Area Network (CAN) technology. The CAN specification provides the Media Access Control (MAC) methodology and physical signaling characteristics.

1.2.3 This document enables communications between intelligent devices on a SEMI compliant SAN by providing a presentation mapping of common and specific device network visible structure and behavior to SafetyBUS p network.

1.2.4 This Network Communication Standard specification for SafetyBUS p is not intended to be a safety guideline for using SafetyBUS p technology. This document specifies a SAN communications standard specification based on the Object Communication Specification (OCS) for the SafetyBUS p Protocol. This SAN is structured in compliance with SEMI E54.1.

2 Scope

2.1 This document specifies the protocol and services that compliant intelligent devices shall support to exchange information over the SafetyBUS p semiconductor equipment sensor/actuator network.

2.2 This document specifies the utilization of the SafetyBUS p protocol to present externally visible device structure and behavior, specified in the Common Device Model (CDM) and appropriate Specific Device Models (SDMs), on a SafetyBUS p network.

2.3 This document is used in conjunction with a SEMI standard SAN Common Device Model specification, one or more SEMI standard Specific Device Model (SDM) specifications (e.g. for a mass flow device), the SafetyBUS p Object Communications Specification Reference Guide. Together, they describe the SafetyBUS p protocol, the externally visible data structures and behaviors of devices utilizing the SafetyBUS p networking capability in a SEMI compliant SAN system.

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 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on SafetyBUS p and is a companion document to the Object Communication Specification for the SafetyBUS p Protocol; thus a complete specification of this standard necessarily includes the SafetyBUS p protocol specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included.

3.2 This standard specifies enhancements that provide additional capabilities over and above those currently required by SafetyBUS p. In order to avoid document consistency problems, information in the Object Communication Specification for the SafetyBUS p Protocol that relate to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the SafetyBUS p specifications that are imposed by this standard specification.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services (OSS)

SMI E54 — Sensor/Actuator Network Standard

SEMI E54.1 — Standard For Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification For Sensor/Actuator Network Specific Device Model For Mass Flow Device

SEMI E54.10 — Specification For Sensor/Actuator Network Specific Device Model For An In-Situ Particle Monitor Device

SEMI E54.11 — Specification For Sensor/Actuator Network Specific Device Model For An Endpoint Device

4.2 *ISO Standards* [1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection

4.3 *SafetyBUS p Club International* [2]

Specification — Object Communication Specification (OCS) for the SafetyBUS p Protocol Version 1.1 April 09, 2004 may be accessed on the SafetyBUS p Club International web site http://www.safetybus.com/semi/

Reference Guide — Pilz Automation Technology – Guide to Programmable Safety Systems, February 2002 Volume 2, 1st Edition

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

5 Terminology

5.1 Terminology that is common to all of the documents in this SAN standard may also be defined in the Sensor /Actuator Network Standard (see reference SEMI E54 ¶4.1). Terminology may be reproduced here which is defined in other SEMI documents.

5.2 *Abbreviations and Acronyms*

5.2.1 *CAN* — Controller Area Network

5.2.2 *CDM* — Common Device Model

5.2.3 *DM* — Device Manager (object)

---

1 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30, Website: /www.iso.ch

2 SafetyBUS p Club International Partner Association, Robert-Bosch-Str. 30, 73760 Ostfildern, Germany ,Telephone (+49) 711 3409-118, Fax: (+49) 711 3409-449, website: www.safetybus.com

5.2.4 *ISO* — International Standards Organization

5.2.5 *MAC* — Media Access Control

5.2.6 *NCS* — Network Communication Standard

5.2.7 *OCS* — Object Communications Specification

5.2.8 *OSI* — Open Systems Interconnect

5.2.9 *SAC* — Sensor, Actuator, Controller (Object)

5.2.10 *SAN* — Sensor/Actuator Network

5.2.11 *SDM* — Specific Device Model

5.3 *Device Component Definitions*

5.3.1 As this standard specification defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in the SEMI E54.1 CDM document. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the SafetyBUS p standard specifications. Note that Column 2 contains an equal sign "=" if the definition is used exactly as specified in the CDM specification.

Table 1 Mapping of Common Device Model to NCS Terminology

CDM Term	*NCS Equivalent*	*SafetyBUS p Equivalent*
Device	=	=
Device Model	=	=
Class	=	Server Class
Object	=, Class, Instance	=, Server Class, Instance
Instance	=	=
Attribute	=	=
Behavior	=	=
Service	=	=
State Diagram	=	=
Byte	=	=
Nibble	=	=
Character String	=	=

5.4 *SafetyBUS p Specific Definitions*

5.4.1 *Server Class* — subset of devices that offer similar functions and provide fixed defined functionality in a uniform way.

5.4.2 *master/slave* — communication over a SafetyBUS p network provides exclusive control of data by a "master" or "host" device. All network input data is reported exclusively to the host when requested by the host, and the host has exclusive control over the states of all network output signals of all nodes acting as its "slaves". Master/Slave communication provides the typical request/response oriented network communications.

5.4.3 *peer-to-peer* — On SafetyBUS p networks, messages formatted according to the SafetyBUS p protocol are embedded into the SafetyBUS p packet structure that is used on the CAN network. The SafetyBUS p protocol over CAN supports the asynchronous or unsolicited bi-directional transmission of data between nodes. This type of communication is referred to as peer-to-peer.

5.4.4 *SafetyBUS p* — an open protocol maintained by Pilz GmbH & Co and distributed by SafetyBUS p Club International as a reliable and standard means of interconnection for simple field devices. The SafetyBUS p standard wraps a communication model and protocol as well as CAN specifications for OSI reference model layers 1 and 2, to provide a complete network definition. The OSI reference model layer 7 specifies the application layer.

6 Communication Protocol High Level Structure

6.1 The SafetyBUS p protocol is loosely based on three layer architecture. These three layers constitute a collapsed form of the seven layer OSI architecture which map into the physical, data link, and application layers of the OSI Basic Reference Model (see ¶4.2). The high-level protocol architecture is shown in Figure 1.

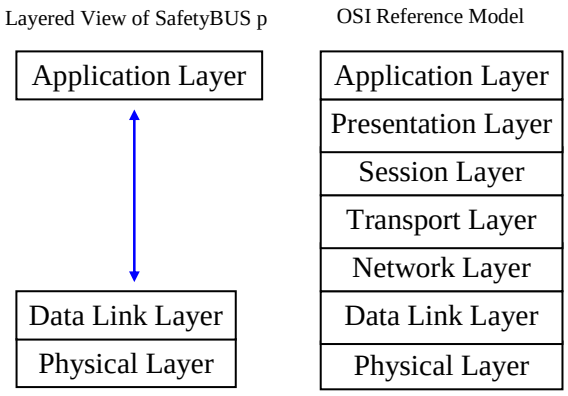


Figure 1
Layered View of SafetyBUS p

6.1.1 Note that Figure 1 represents a conceptual view of the technology architecture. Conforming implementations must implement the services defined in this specification at each layer and must appear (from the network) to have implemented this architecture, however an internal modular partitioning is not required. Implementations may sacrifice modularity in order to achieve high performance.

6.1.2 The application layer is specified in the SafetyBUS p OCS (see reference ¶4.3) and provides for the definition of SafetyBUS p applications as a collection of addressable objects. A subset of these objects may be addressed over the network (as defined by the implementation).

6.1.3 In the remainder of this section the protocol structure is described in more detail in terms of the OSI seven layer reference model, the object model environment and network management specifications.

6.2 *Physical Layer* — The device shall comply with a physical layer specification identified in the Controller Area Network (CAN) specification. Physical layer specification includes physical signaling (levels and data rates), transceivers, node isolation, media topology, cable specifications, network connectors and taps, and power considerations (load limits, system tolerances, and power supply options).

6.3 *Data Link Layer* — The device shall comply with a data link layer specification the Controller Area Network (CAN) specification. Data link layer specification includes the media access control mechanism and the logical link control mechanism.

6.4 *Network Layer* — There is no distinct Network layer.

6.5 *Transport (Messaging) Layer* — There is no distinct Transport layer.

6.6 *Session Layer* — There is no distinct Session layer.

6.7 *Presentation Layer* — There is no distinct Presentation layer.

6.8 *Application Layer* — The device shall comply with the SafetyBUS p application layer specification for defining and addressing objects, including their attributes and services, and enabling specified network behavior. The device shall comply with the object messaging and object model specifications included in the SafetyBUS p OCS. In addition the device shall comply with the object specifications defined in §7 of this document.

6.8.1 *Object Models* — The SafetyBUS p protocol includes an object-oriented specification for addressing objects explicitly, including their attributes and services, and communicating object attributes in an application dependent format. The device shall comply with the object messaging and object model specifications included in the

SafetyBUS p OCS documentation. In addition the device shall comply with the object specifications defined in §7 of this document.

6.9 *Network Management* — The device shall comply with the SafetyBUS p and CAN network management specifications detailed in the SafetyBUS p standard and SafetyBUS p Guide to Programmable Safety Systems Standard Specifications (e.g., physical layer bit rate, master/slave and peer-to-peer network management, etc.). No (additional) network management functions are specified in this document.

7 Required And Optional Object Types

7.1 The SafetyBUS p standard specification does not require any specific objects to exist in a SafetyBUS p device in order to be a compliant SafetyBUS p device. The SafetyBUS p standard specification is extended in this standard to identify and describe objects (i.e. classes) that shall exist in devices that are to be interoperable and interchangeable on a SafetyBUS p SEMI compliant SAN network.

7.1.1 The Common Device Model (CDM) specification (see reference SEMI E54.1 ¶4.1) identifies two objects (namely the Device Manager (DM) and Sensor Actuator Controller (SAC) objects) that shall exist in all SEMI compliant SAN devices.

7.1.2 The required object types for a SEMI compliant SAN device utilizing the network communication specification described herein necessarily comprises, at minimum, the union of the SafetyBUS p object type requirements and the CDM specification requirements.

7.1.3 A list of required and optional object types is given in Table 2. Additional objects that are specified in a particular SDM are given identifiers in that SDM specification; SafetyBUS p specific presentation information for these identifiers is given in Section 9 of this document.

Table 2 Required and Optional Object Types

Object Name	*SafetyBUS p Class ID / Instance ID* [#1]	*CDM Tag* [#2]	*Required by SafetyBUS p* [#1]	*Required by CDM* [#2]	*Required by NCS*
Device Manager	1/1	DmI0	No	Yes	Yes
Sensor/ Actuator/ Controller	2/1	SacI0	No	Yes	Yes
Assembly	3/1 through i	Asm	No	No	No
Local Link	4/1 through j	Lnk	No	No	No
Sensor – AI	33/1 through k	Sai	No	No	No
Sensor – EI	34/1 through l	Sei	No	No	No
Sensor – BI	35/1 through m	Sbi	No	No	No
Actuator – AO	36/1 through n	Aao	No	No	No
Actuator – EO	37/1 through o	Aeo	No	No	No
Actuator – BO	38/1 through p	Abo	No	No	No
Controller	39/1 through q	C	No	No	No
Sensor – BI-TH	40/I through s	Sbith	No	No	No
Application Objects	129 through x/1 through r	(3)	No	No	No

[#1] See SafetyBUS p specification for further information; values are decimal; ‘i’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’, ‘r’ and ‘s’ represent arbitrary numbers (greater than or equal to 1) indicating that more than one instance may be supported. ‘x’ is a number greater than or equal to 129 indicating that one or more application object classes may be supported.

[#2] See CDM specification for further information

[#3] Application Dependent objects’ tags as specified in SDM

7.1.4 An embodiment of a specific device type represented as an aggregation of the object types listed in Table 2 that is compliant with both the CDM specification and the SafetyBUS p specification, is a candidate for a SEMI SDM as well as a SafetyBUS p device definition. Conversely, all SEMI SDMs and SafetyBUS p device definitions specified for operation over a SEMI compliant SafetyBUS p network must be an aggregation of the object types listed in Table 2, and be compliant with both the CDM specification and the SafetyBUS p standard specifications.

7.1.5 In the following sections the presentation to the network of object addressing, object attributes and object services for each of the object types listed in Table 2 is described in detail. Refer to the CDM standard to determine if the object instance attribute and service is specified as required or optional. Unless otherwise noted, all attributes and services described are instance level attributes (as opposed to class level attributes). A class level attribute and service is accessed as instance number zero.

7.1.6 Note that the formats of object attributes and services are detailed in the CDM document; the presentation of object attributes and services to the SafetyBUS p network is detailed in the tables contained in the following sub-sections and in the SafetyBUS p standard specifications.

7.2 *Device Manager (DM) Object* — The DM object instance is the device component responsible for managing and consolidating the device operation. Each device must support one (and only one) DM object. The DM object as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object instance attributes and services to the SafetyBUS p network shall be as indicated in Table 3. Note that all service ID values identified refer to the ID of the request or notification component of that service. Corresponding reply components to request/reply services shall have a service ID value equal to the request component ID plus one.

Table 3 DM Object Instance Attributes and Services

Device Manager Object (DM) *Class ID = 01, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Device Type	DmA1
02	Standard Revision Level	DmA2
03	Device Manufacturer Identifier	DmA3
04	Manufacturer Model Number	DmA4
05	Software or Firmware Revision Level	DmA5
06	Hardware Revision Level	DmA6
07	Serial Number	DmA7
08	Device Configuration	DmA8
09	Device Status	DmA9
10	Reporting Mode	DmA10
11	Exception Status Report	DmA11
12	Exception Status	DmA12
13	Exception Detail Alarm	DmA13
14	Exception Detail Warning	DmA14
15	Visual Indicator	DmA15
16	Alarm Enable	DmA16
17	Warning Enable	DmA17
18	Exception Detail Type	DmA18
19	Exception Detail Alarm Queue	DmA19
20	Exception Detail Warning Queue	DmA20
21	Date and Time	DmA21
22	Date and Time Type	DmA22

Services		
ID	*Service Name*	*CDM Tag*
01	Reset	DmS1
03	Abort	DmS2
05	Recover	DmS3
07	Get Attribute	DmS4
09	Set Attribute	DmS5
11	Execute	DmS6
13	Perform Diagnostics	DmS7
15	Publish Attribute	DmS8
17	Lock	DmS9
19	Unlock	DmS10
21	Get Exception Queue	DmS11
23	Clear Exception Queue	DmS12

7.3 *Sensor, Actuator, Controller (SAC) Object* — The SAC object instance is the device component responsible for coordinating the interaction of the device with the sensory/actuation/control environment. Each device must support one (and only one) SAC object instance. The SAC object instance as well as its common required and optional attributes, services and behavior are described in the CDM standard. The presentation of object instance attributes and services to the SafetyBUS p network shall be as indicated in Table 4.

Table 4 Object Instance Attributes and Services

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Last Calibration Date	SacA1
02	Next Calibration Date	SacA2
03	Expiration Timer	SacA3
04	Expiration Warning Enable	SacA4
05	Run Hours	SacA5
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SacS1
03	Abort	SacS2
05	Recover	SacS3
07	Get Attribute	SacS4
09	Set Attribute	SacS5
25	Operate	SacS6
27	Restore Default	SacS7
29	Publish Attribute	SacS8

7.4 *Assembly Object (Asm)* — The Assembly (Asm) object instances may be used to provide for grouping more than one attribute from one or more object instances in a device into a single data structure for communication over the SafetyBUS p Network. The presentation of object instance attributes and services shall be as indicated in Table 5.

Table 5 Assembly Object Instance Attributes and Services

ASSEMBLY Object (Asm) *Class ID = 03, Instance ID = 01 through i*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Data	AsmA1
Services		
ID	*Service Name*	*CDM Tag*
07	Get Attribute	AsmS4
09	Set Attribute	AsmS5

7.5 *Local Link Object (Lnk)* — The Local Link (Lnk) object instances may be used to 'link" an attribute of one object instance to an attribute of another object instance. Refer to the CDM for further explanation and use of this object. The presentation of object instance attributes and services are as indicated in Table 6.

Table 6 Local Link Object Instance Attributes and Services

Local Link Object (Lnk) *Class ID = 04, Instance ID = 01 through j*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Source Object Class	LnkA1
02	Source Object Instance	LnkA2
03	Source Object Attribute	LnkA3
04	Destination Object Class	LnkA4
05	Destination Object Instance	LnkA5
06	Destination Object Attribute	LnkA6
07	Commit	LnkA7
Services		
ID	*Service Name*	*CDM Tag*
--	No services defined	--

7.6 *Sensor-AI Object (Sai)* — The presentation of the Sensor Analog Input (Sensor-AI) object instance attributes and services are as indicated in Table 7.

Table 7 Sensor-AI Object Instance Attributes and Services

Sensor-AI (Sai) *Class ID = 33, Instance ID = 01 through k*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SaiA1
02	Status	SaiA2
03	Alarm Enable	SaiA3
04	Warning Enable	SaiA4
16	Value	SaiA16
17	ReportInhibitTimer	SaiA17
18	EnableReportRate	SaiA18
19	ReportRate	SaiA19
64	Offset	SaiA64
65	Gain	SaiA65
66	DataType	SaiA66
67	DataUnits	SaiA67
68	SafeState	SaiA68
69	EnableReportDelta	SaiA69
70	ReportDelta	SaiA70
71	EnableReportROC	SaiA71
72	ReportROC	SaiA72
73	AlarmTripPointHigh	SaiA73
74	AlarmTrippointLow	SaiA74
75	AlarmHystersis	SaiA75
76	WarningTripPointHigh	SaiA76
77	WarningTripPointLow	SaiA77
78	WarningHystersis	SaiA78
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SaiS1
03	Abort	SaiS2
05	Recover	SaiS3
25	Operate	SaiS4
07	GetAttribute	SaiS5
09	SetAttribute	SaiS6
27	RestoreDefault	SaiS7

7.7 *Sensor-EI Object (Sei)* — The presentation of the Sensor Enumerated Input (Sensor-EI) object instance attributes and services are as indicated in Table 8.

Table 8 Sensor-EI Object Instance Attributes and Services

Sensor-EI (Sei) *Class ID = 34, Instance = 01 through l*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SeiA1
02	Status	SeiA2
03	Alarm Enable	SeiA3
04	Warning Enable	SeiA4
16	Value	SeiA16
17	ReportInhibitTimer	SeiA17
18	EnableReportRate	SeiA18
19	ReportRate	SeiA19
64	DebounceControl	SeiA64
65	AlarmState	SeiA65
66	WarningState	SeiA66
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SeiS1
03	Abort	SeiS2
05	Recover	SeiS3
25	Operate	SeiS4
07	GetAttribute	SeiS5
09	SetAttribute	SeiS6
27	RestoreDefault	SeiS7

7.8 *Sensor-BI Object (Sbi)* — The presentation of the Sensor Binary Input (Sensor-BI) object instance attributes and services are as indicated in Table 9.

Table 9 Sensor-BI Object Instance Attributes and Services

Sensor-BI (Sbi) *Class ID = 35, Instance ID = 01 through m*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SbiA1
02	Status	SbiA2
03	Alarm Enable	SbiA3
04	Warning Enable	SbiA4
16	Value	SbiA16
17	ReportInhibitTimer	SbiA17
18	EnableReportRate	SbiA18
19	ReportRate	SbiA19
64	DebounceControl	SbiA64
65	AlarmState	SbiA65
66	WarningState	SbiA66

Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SbiS1
03	Abort	SbiS2
05	Recover	SbiS3
25	Operate	SbiS4
07	GetAttribute	SbiS5
09	SetAttribute	SbiS6
27	RestoreDefault	SbiS7

7.9 *Sensor-BI-TH Object (Sbith)* — The presentation of the Sensor Binary Input Threshold (Sensor-BI-TH) object instance attributes and services are as indicated in Table 10.

Table 10 Sensor-BI-TH Object Instance Attributes and Services

Sensor-BI (Sbith) *Class ID = 40, Instance ID = 01 through m*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	SbiA1
02	Status	SbiA2
03	Alarm Enable	SbiA3
04	Warning Enable	SbiA4
16	Value	SbiA16
17	ReportInhibitTimer	SbiA17
18	EnableReportRate	SbiA18
19	ReportRate	SbiA19
64	DebounceControl	SbiA64
65	AlarmState	SbiA65
66	WarningState	SbiA66
67	Reading Valid	SbithA64
68	State	SbithA65
69	Status	SbithA66
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	SbiS1
03	Abort	SbiS2
05	Recover	SbiS3
25	Operate	SbiS4
07	GetAttribute	SbiS5
09	SetAttribute	SbiS6
27	RestoreDefault	SbiS7

7.10 *Actuator-AO Object (Aao)* — The presentation of the Actuator Analog Output (Actuator-AO) object instance attributes and services are as indicated in Table 11.

Table 11 Actuator-AO Object Instance Attributes and Services

Actuator-AO (Aao) *Class ID = 36, Instance = 01 through n*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AaoA1
02	Status	AaoA2
03	Alarm Enable	AaoA3
04	Warning Enable	AaoA4
16	Setting	AaoA16
17	SafeState	AaoA17
18	WatchRate	AaoA18
19	Watchdog	AaoA19
64	Offset	AaoA64
65	Gain	AaoA65
66	DataType	AaoA66
67	DataUnits	AaoA67
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	AaoS1
03	Abort	AaoS2
05	Recover	AaoS3
25	Operate	AaoS4
07	GetAttribute	AaoS5
09	SetAttribute	AaoS6
27	RestoreDefault	AaoS7

7.11 *Actuator-EO Object (Aeo)* — The presentation of the Actuator Enumerated Output (Actuator-EO) object instance attributes and services are as indicated in Table 12.

Table 12 Actuator-EO Object Instance Attributes and Services

Actuator-EO (Aeo) *Class ID = 37, Instance ID = 01 through o*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AeoA1
02	Status	AeoA2
03	Alarm Enable	AeoA3
04	Warning Enable	AeoA4
16	Setting	AeoA16
17	SafeState	AeoA17
18	WatchRate	AeoA18
19	Watchdog	AeoA19

Services		
ID	*Service Name*	*CDM Tag*
01	Reset	AeoS1
03	Abort	AeoS2
05	Recover	AeoS3
25	Operate	AeoS4
07	GetAttribute	AeoS5
09	SetAttribute	AeoS6
27	RestoreDefault	AeoS7

7.12 *Actuator-BO Object (Abo)* — The presentation of the Actuator Binary Output (Actuator-BO) object instance attributes and services are as indicated in Table 13.

Table 13 Actuator-BO Object Instance Attributes and Services

Actuator-BO (Abo) *Class ID = 38, Instance ID = 01 through p*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	AboA1
02	Status	AboA2
03	Alarm Enable	AboA3
04	Warning Enable	AboA4
16	Setting	AboA16
17	SafeState	AboA17
18	WatchRate	AboA18
19	Watchdog	AboA19
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	AboS1
03	Abort	AboS2
05	Recover	AboS3
25	Operate	AboS4
07	GetAttribute	AboS5
09	SetAttribute	AboS6
27	RestoreDefault	AboS7

7.13 *Controller Object (C)* — The presentation of the Controller (C) object instance attributes and services are as indicated in Table 14.

Table 14 Controller-C Instance Object Attributes and Services

Controller (C) *Class ID = 39, Instance ID = 01 through q*		
Attributes		
ID	*Attribute Name*	*CDM Tag*
01	Name	CA1
02	Status	CA2
03	Alarm Enable	CA3
04	Warning Enable	CA4
16	Setpoint	CA16
17	ProcessVariable	CA17
18	ControlVariable	CA18
19	DataType	CA19
20	DataUnits	CA20
21	AlarmSettleTime	CA21
22	AlarmErrorBand	CA22
24	WarningSettleTime	CA24
25	WarningErrorBand	CA25
Services		
ID	*Service Name*	*CDM Tag*
01	Reset	CS1
03	Abort	CS2
05	Recover	CS3
25	Operate	CS4
07	GetAttribute	CS5
09	SetAttribute	CS6
27	RestoreDefault	CS7

8 Protocol Compliance

8.1 A method of testing protocol compliance is required to verify implementation conformance to the standard. SafetyBUS p Club International e.V. has established and maintains a mechanism for compliance certification of devices on a SafetyBus p network. This certification includes procedures and reporting mechanisms to demonstrate conformance testing and interoperability testing of SafetyBUS p devices. The SafetyBUS p Club International e.V. organization can be contacted at the web site http://www.safetybus.com to obtain the latest certification procedures.

9 Specific Device Model Mappings

9.1 This section provides for the mapping of network visible specific device structure and behavior, specified in a SEMI standard SDM specification, to the SafetyBUS p network. Each subsection is devoted to a single Specific Device Model (SDM) specification. Additional SDM mappings are added as sub-sections to this NCS specification according to SEMI guidelines and the guidelines of the SEMI SAN Interoperability standard. Unless otherwise noted, all attributes and services described are instance level attributes (as opposed to class level attributes).

9.1.1 Note that the formats of object instance attributes and services are detailed in the associated SDM specification; the presentation of object attributes and services to the SafetyBUS p network is detailed in the tables contained in the following sub-sections and in the SafetyBUS p standard specifications. Note that relationships between object classes, including inheritance are defined in the associated SDM specification and the CDM specification.

9.1.2 The instance identifier of 1 through s, assigned to an object type, refers to the possibility of multiple instantiations of the object type. Refer to Table 2 of this document and the CDM document for a further explanation of object instance identifier assignments.

9.2 *Specific Device Model for Mass Flow Device*

9.2.1 These sections detail the network mapping required to support the Specific Device Model for Mass Flow Device (see SEMI E54.3 ¶4.1). Table 15 summarizes the Mass Flow Device Object types. Subsequent Tables 16 to 25 details the instance attributes and services associated with each Mass Flow Device object type.

Table 15 Mass Flow Device Object Types

SDM Object Identifier	*Object Name*	*SafetyBUS p Class ID*
MFD1 (DM)	Device Manager	1
MFD2 (SAC)	Sensor Actuator Controller	2
MFD3	Sensor-AI-MF	129
MFD4	Sensor-AI-AT	130
MFD5	Assembly-MFM	131
MFD6	Sensor-AI-Aux	132
MFD7	Actuator-AO-MF	133
MFD8	Controller	39
MFD9	Local Link	4
MFD10	SISO	134
MFD11	SISO-Setpoint	135
MFD12	Assembly-MFC	136

9.2.2 *Sensor-AI-MF* — The presentation of the Sensor Analog Input Mass Flow (Sensor-AI-MF) object instance attributes and services are as indicated in Table 16.

Table 16 Sensor-AI-MF Object Instance Attributes and Services

Sensor-AI-MF *Class ID = 129, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Flow Totalizer	A1
129	Flow Hours	A2
130	Zero Offset Mode	A5
131	Zeroing Status	A6
132	Autorange Status	A7
Services		
ID	*Service Name*	*SDM Tag*
129	Perform Zero Offset	S1
131	Query-Supported Gas Types	S2
133	Selected Programmed Gas Type	S3
135	Insert Gas Type	S4
137	Delete Gas Type	S5
139	Get Gas Calibration Data Value	S6
141	Set Gas Calibration Data Value	S7
143	Autorange	S8

9.2.3 *Sensor-AI-AT* — The presentation of the Sensor Analog Input Ambient Temperature (Sensor-AI-AT) object instance attributes and services are as indicated in Table 17.

Table 17 Sensor-AI-AT Object Instance Attributes and Services

Sensor-AI-AT *Class ID = 130, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.4 *Assembly-MFM* — The presentation of the Assembly Mass Flow Meter (Assembly-MFM) object instance attributes and services are as indicated in Table 18.

Table 18 Assembly-MFM Object Instance Attributes and Services

Assembly-MFM *Class ID = 131, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.5 *Sensor-AI-Aux* — The presentation of the Sensor Analog Input Auxiliary (Sensor-AI-Aux) object instance attributes and services are as indicated in Table 19.

Table 19 Sensor-AI-Aux Object Instance Attributes and Services

Sensor-AI-Aux *Class ID = 132, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.6 *Actuator-AO-MF* — The presentation of the Actuator Analog Output Mass Flow (Actuator-AO-MF) object instance attributes and services are as indicated in Table 20.

Table 20 Actuator-AO-MF Object Instance Attributes and Services		
Actuator-AO-MF *Class ID = 133, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Valve Type	A1
129	Override	A2

Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.7 *Controller* — The presentation of the extended Controller (C) object instance attributes and services are as indicated in Table 21.

Table 21 Controller Object Instance Attributes and Services

Controller *Class ID = 39, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Alarm Settling Time	CA21
129	Warning Settling Time	CA24
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.8 *Local Link* — The presentation of the extended Local Link (Lnk) object instance attributes and services are as indicated in Table 22.

Table 22 Local Link Object Instance Attributes and Services

Local Link *Class ID = 4, Instance ID = 01 through j*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.9 *SISO* — The presentation of the Single Input Single Output (SISO) object instance attributes and services are as indicated in Table 23.

Table 23 SISO Object Instance Attributes and Services

SISO *Class ID = 134, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Input	A1
129	Output	A2
130	Data Type	A3
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.2.10 *SISO-Setpoint* — The presentation of the Single Input Single Output Setpoint (SISO-Setpoint) object instance attributes and services are as indicated in Table 24.

Table 24 SISO-Setpoint Object Instance Attributes and Services

SISO-Setpoint *Class ID = 135, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
161	Ramp Type	A33
162	Ramp Rate	A34
163	Ratio	A35
Services		
ID	*Service Name*	*SDM Tag*
---	No additional services defined	--

9.2.11 *Assembly-MFC* — The presentation of the Assembly Mass Flow Controller (Assembly-MFC) object instance attributes and services are as indicated in Table 25.

Table 25 Assembly-MFC Object Instance Attributes and Services

Assembly-MFC *Class ID = 136, Instance ID = 01 through r*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3 *Specific Device Model For In-Situ Particle Monitor Device* — These sections detail the network mapping required to support the Specific Device Model for In-Situ Particle Monitor (ISPM) Devices (see SEMI E54.10 ¶4.1). Table 26 summarizes the In-Situ Particle Monitor Device Object types. Subsequent Tables 27 to 42 details the attributes and services associated with each In-Situ Particle Monitor Device object type.

Table 26 In-Situ Particle Monitor Device Object Types

SDM Object Identifier	*Object Name*	*SafetyBUS p Class ID*
ISPMD1 (DM)	Device Manager	1
ISPMD2 (SAC)	Sensor Actuator Controller	2
ISPMD3	Sensor-AI-LCS	137
ISPMD4	Sensor-AI-SLS	138
ISPMD5	Assembly-AI-MNS	139
ISPMD16	Sensor-AI-Counter	140
ISPMD17	Assembly-ISPM#1	141
ISPMD18	Assembly-ISPM#2	142
ISPMD19	Assembly-ISPM#3	143
ISPMD20	Assembly-ISPM#4	144
ISPMD21	Assembly-ISPM#5	145
ISPMD22	Assembly-ISPM#6	146
ISPMD23	Assembly-ISPM#7	147
ISPMD24	Assembly-ISPM#8	148
ISPMD25	Assembly-ISPM#9	149
ISPMD64	Assembly-ISPM#48	150

9.3.1 *Device Manager* — The presentation of the extended ISPM Device Manager (DM) object instance attributes and services are as indicated in Table 27.

Table 27 DM Object Instance Attributes and Services

Device Manager Object (DM) *Class ID = 01, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Gain	A33
129	Filter Bandwidth	A34
130	Tool State	A35
131	Laser Status	A36
132	Flow Path	A37
133	Volume	A38
134	Volume Units	A39
135	Leak Status	A40
136	Time Stamp	A41
Services		
ID	*Service Name*	*SDM Tag*
33	Laser On	S1
35	Laser Off	S2

9.3.2 *Sensor Actuator Controller (SAC)* — The presentation of the extended ISPM Sensor Actuator Controller (SAC) object attributes and services are as indicated in Table 28.

Table 28 SAC Object Instance Attributes and Services

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
65	Number of Bins	SacA65
66	Count Mode	SacA66
67	Duration	SacA67
Services		
ID	*Service Name*	*SDM Tag*
33	Clear Counts	S33

9.3.3 *Sensor-AI-LCS* — The presentation of the Sensor Analog Input Laser Current Sensor (Sensor-AI-LCS) object instance attributes and services are as indicated in Table 29.

Table 29 Sensor-AI-LCS Object Instance Attributes and Services

Sensor-AI-LCS *Class ID = 137, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	LcsA1
129	Full Scale	LcsA2
130	Alarm Settling Time	LcsA3
131	Warning Settling Time	LcsA4

Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.4 *Sensor-AI-SLS* — The presentation of the Sensor Analog Input Stray Light Sensor (Sensor-AI-SLS) object instance attributes and services are as indicated in Table 30.

Table 30 Sensor-AI-SLS Object Instance Attributes and Services

Sensor-AI-SLS *Class ID = 138, Instance ID = 1*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	SlsA1
129	Full Scale	SlsA2
130	Alarm Settling Time	SlsA3
131	Warning Settling Time	SlsA4
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.5 *Sensor-AI-MNS* — The presentation of the Sensor Analog Input Medium Noise Sensor (Sensor-AI-MNS) object instance attributes and services are as indicated in Table 31.

Table 31 Sensor-AI-MNS Object Instance Attributes and Services

Sensor-AI-MNS *Class ID = 139, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	MnsA1
129	Full Scale	MnsA2
130	Alarm Settling Time	MnsA3
131	Warning Settling Time	MnsA4
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.6 *Sensor-AI-Counter* — The presentation of the Sensor Analog Input Counter (Sensor-AI-Counter) object instance attributes and services are as indicated in Table 32.

Table 32 Sensor-AI-Counter Object Instance Attributes and Services

Sensor-AI-Counter *Class ID = 140, Instance ID = 01 through 1024*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Reading Valid	CounterA1
129	Full Scale	CounterA2
130	Alarm Settling Time	CounterA3
131	Warning Settling Time	CounterA4
132	Upper Size	CounterA5
133	Lower Size	CounterA6
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.7 *Assembly-ISPM#1* — The presentation of the Assembly #1 In-Situ Particle Monitor (Assembly-ISPM#1) object instance attributes and services are as indicated in Table 33.

Table 33 Assembly-ISPM#1 Object Instance Attributes and Services

Assembly-ISPM#1 *Class ID = 141, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.8 *Assembly-ISPM#2* — The presentation of the Assembly #2 In-Situ Particle Monitor (Assembly-ISPM#2) object instance attributes and services are as indicated in Table 34.

Table 34 Assembly-ISPM#2 Object Instance Attributes and Services

Assembly-ISPM#2 *Class ID = 142, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.9 *Assembly-ISPM#3* — The presentation of the Assembly #3 In-Situ Particle Monitor (Assembly-ISPM#3) object instance attributes and services are as indicated in Table 35.

Table 35 Assembly-ISPM#3 Object Instance Attributes and Services

Assembly-ISPM#3 *Class ID = 143, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.10 *Assembly-ISPM#4* — The presentation of the Assembly #4 In-Situ Particle Monitor (Assembly-ISPM#4) object instance attributes and services are as indicated in Table 36.

Table 36 Assembly-ISPM#4 Object Instance Attributes and Services

Assembly-ISPM#4 *Class ID = 144, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.11 *Assembly-ISPM#5* — The presentation of the Assembly #5 In-Situ Particle Monitor (Assembly-ISPM#5) object instance attributes and services are as indicated in Table 37.

Table 37 Assembly-ISPM#5 Object Instance Attributes and Services

Assembly-ISPM#5 *Class ID = 145, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.12 *Assembly-ISPM#6* — The presentation of the Assembly #6 In-Situ Particle Monitor (Assembly-ISPM#6) object instance attributes and services are as indicated in Table 38.

Table 38 Assembly-ISPM#6 Object Instance Attributes and Services

Assembly-ISPM#6 *Class ID = 146, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.13 *Assembly-ISPM#7* — The presentation of the Assembly #7 In-Situ Particle Monitor (Assembly-ISPM#7) object instance attributes and services are as indicated in Table 39.

Table 39 Assembly-ISPM#7 Object Instance Attributes and Services

Assembly-ISPM#7 *Class ID = 147, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.14 *Assembly-ISPM#8* — The presentation of the Assembly #8 In-Situ Particle Monitor (Assembly-ISPM#8) object instance attributes and services are as indicated in Table 40.

Table 40 Assembly-ISPM#8 Object Instance Attributes and Services

Assembly-ISPM#8 *Class ID = 148, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.15 *Assembly-ISPM#9* — The presentation of the Assembly #9 In-Situ Particle Monitor (Assembly-ISPM#9) object instance attributes and services are as indicated in Table 41.

Table 41 Assembly-ISPM#9 Object Instance Attributes and Services

Assembly-ISPM#9 *Class ID = 149, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.3.16 *Assembly-ISPM#48* — The presentation of the Assembly #48 In-Situ Particle Monitor (Assembly-ISPM#48) object instance attributes and services are as indicated in Table 42.

Table 42 Assembly-ISPM#48 Object Instance Attributes and Services

Assembly-ISPM#48 *Class ID = 150, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4 *Specific Device Model For Endpoint Device* — These sections detail the network mapping required to support the Specific Device Model for Endpoint (EPD) Devices (see reference SEMI E54.11 ¶4.1). Table 43 summarizes the Endpoint Device Object types. Subsequent Tables 44 to 50 details the attributes and services associated with each Endpoint Device object type.

Table 43 Endpoint Device Object Types

SDM Object Identifier	*Object Name*	*SafetyBUS p Class ID*
EPD1	Device Manager (DM)	1
EPD2	Sensor Actuator Controller (SAC)	2
EPD3	Sensor-BI-TH-EP	137
EPD4	Assembly-EPD#1	138
EPD5	Assembly-EPD#2	139
EPD6	Assembly-EPD#3	140
EPD7	Assembly-EPD#4	141

9.4.1 *Device Manager* — The presentation of the extended EPD Device Manager (DM) object instance attributes and services are as indicated in Table 44.

Table 44 DM Object Instance Attributes and Services

Device Manager Object (DM) *Class ID = 01, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
--	No additional attributes defined	--
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.2 *Sensor Actuator Controller (SAC)* — The presentation of the extended EPD Sensor Actuator Controller (SAC) object attributes and services are as indicated in Table 45.

Table 45 SAC Object Instance Attributes and Services

Sensor, Actuator, Controller Object (SAC) *Class ID = 02, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
65	Number of Endpoint Objects	SacA65
Services		
ID	*Service Name*	*SDM Tag*
33	Reset Endpoint	S33
34	Download Recipe	S34
35	Upload Recipe	S35
36	Calibrate	S36

9.4.3 *Sensor-BI-TH-EP* — The presentation of the Sensor Binary Input Threshold Endpoint (Sensor-BI-TH-EP) object instance attributes and services are as indicated in Table 46.

Table 46 Sensor-BI-TH-EP Object Instance Attributes and Services

Sensor-BI-TH-EP *Class ID = 137, Instance ID = 01 through 1024*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
128	Minimum Time	EpA1
129	Maximum Time	EpA2
130	Target Time	EpA3
131	Elapsed Time	EpA4
132	Time Stamp	EpA5
133	Recipe Identifier	EpA6
134	Step Identifier	EpA7

Services		
ID	*Service Name*	*SDM Tag*
33	Endpoint On	S1
34	Endpoint Off	S2
35	Endpoint Start	S3
36	Endpoint Suspend	S4
37	Endpoint Resume	S5

9.4.4 *Assembly-EPD#1* — The presentation of the Assembly #1 Endpoint Device (Assembly-EPD#1) object instance attributes and services are as indicated in Table 47.

Table 47 Assembly-EPD#1 Object Instance Attributes and Services

Assembly-EPD#1 *Class ID = 138, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.5 *Assembly-EPD#2* — The presentation of the Assembly #2 Endpoint Device (Assembly-EPD#2) object instance attributes and services are as indicated in Table 48.

Table 48 Assembly-EPD#2 Object Instance Attributes and Services

Assembly-EPD#2 *Class ID = 139, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.6 *Assembly-EPD#3* — The presentation of the Assembly #3 Endpoint Device (Assembly-EPD#3) object instance attributes and services are as indicated in Table 49.

Table 49 Assembly-EPD#3 Object Instance Attributes and Services

Assembly-EPD#3 *Class ID = 140, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

9.4.7 *Assembly-EPD#4* — The presentation of the Assembly #4 Endpoint Device (Assembly-EPD#4) object instance attributes and services are as indicated in Table 50.

Table 50 Assembly-EPD#4 Object Instance Attributes and Services

Assembly-EPD#4 *Class ID = 141, Instance ID = 01*		
Attributes		
ID	*Attribute Name*	*SDM Tag*
01	Data	A1
Services		
ID	*Service Name*	*SDM Tag*
--	No additional services defined	--

SEMI E54.16-0705
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK COMMUNICATIONS FOR LONWORKS

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 7, 2005. It was available at www.semi.org in June 2005 and on CD-ROM in July 2005.

NOTICE: This document replaced SEMI E54.6 in 2005.

1 Purpose

1.1 This standard specification defines a communication specification based on the ANSI/EIA/CEA-709.1 Control Networking Protocol (LONWORKS) to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate in accordance with SEMI-specified device models (common and device specific) on semiconductor manufacturing equipment.

1.2 This document specifies a mapping of the SEMI Common Device Model (CDM) onto LONWORKS technology using the *LONMARK Interoperability Guidelines* established for LONWORKS devices. The LONMARK International Association will review and approve the enhanced SEMI LONMARK functional profiles presented in this network communication document and incorporate the required SEMI profiles into the *LONMARK Interoperability Guidelines*.

1.3 *Background and Motivation* — The LONWORKS communications system provides interconnection of smart control devices such as sensors, actuators, and controllers in a fast-response time, low-cost network for industrial use. LONWORKS enables multiple devices to share a single network, thereby significantly reducing the point-to-point wiring between controllers, sensors, and actuators. The LONWORKS network communications standard (NCS) is based on the seven-layer ANSI/EIA/CEA-709.1 (LONWORKS) protocol, implemented by the Neuron Chip, a physical layer transceiver, and an optional host processor. The ANSI/EIA/CEA-709.1 (LONWORKS) control networking protocol was developed by Echelon and may be freely licensed for implementation on any hardware platform. The SEMI NCS for LONWORKS is based on the LONMARK interoperability guidelines, which provide a framework for interoperable use of the ANSI/EIA/CEA-709.1 (LONWORKS) protocol at layers 1-6, as well as at the application layer. Where the LONMARK interoperability guidelines do not provide the functionality required by the SEMI Common Device Model (CDM) and the SEMI Specific Device models (SDM), the guidelines are extended with SEMI-specific enhanced profile requirements.

2 Scope

2.1 This document specifies a SAN communications specification standard, based on the ANSI/EIA/CEA-709.1 Control Networking Protocol (LONWORKS) specification, that enables communication with SAN devices configured according to the SEMI SAN Common Device Model and appropriate SEMI SAN Specific Device Model (SDM) specifications.

2.2 This document specifies the use of LONWORKS technology for services that compliant intelligent devices must support in order to exchange information over this semiconductor equipment sensor/actuator network.

2.3 This document specifies the utilization of LONWORKS technology to present externally visible device structure and behavior, specified in the CDM and appropriate SDMs, on a LONWORKS network.

2.4 This document is used in conjunction with the SEMI SAN Common Device Model specification and one or more SEMI SAN Specific Device Model specifications (e.g., for a mass flow controller). Together, the model documents describe the externally visible data structures and behavior of devices using LONWORKS technology in a SEMI-compliant SAN system.

2.5 This specification, together with the sensor/actuator network common device model, one or more sensor/actuator network specific device model documents, the ANSI/EIA/CEA-709.1 (LONWORKS) protocol specification, and the LONMARK interoperability guidelines, specifies requirements for SEMI SAN implementations based on LONWORKS technology.

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

3 Limitations

3.1 This document specifies a semiconductor equipment SAN based solely on LONWORKS technology and is a companion document to the ANSI/EIA/CEA-709.1 (LONWORKS) control networking protocol specification; thus, a complete specification of this standard necessarily includes the LONWORKS specifications. There are other semiconductor equipment SAN communications options. The specifications for these options are not included here.

3.2 This document specifies enhancements that provide additional capabilities over and above those currently required by the LONMARK *Interoperability Guidelines*. In order to avoid document consistency problems, information in the LONWORKS technology specifications that relate to this standard is not repeated in this document. This document is limited to describing enhancements or limitations of LONWORKS technology and the LONMARK interoperability guidelines that are imposed by this specification.

3.3 A complete specification of the conformance testing procedure shall include the applicable LONMARK interoperability conformance testing specification. Conformance testing shall include enhancements to the LONMARK interoperability guidelines required by this standard specification.

4 Referenced Standards and Documents

4.1 It must be noted that unless otherwise indicated, all documents cited below shall be the latest published versions.

4.2 *SEMI Standards*

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

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

SEMI E54 – Sensor/Actuator Network Standard

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 – Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

SEMI E54.10 – Specification for Sensor/Actuator Network Specific Device Model for An In-Situ Particle Monitor Device

SEMI E54.11 – Specification for Sensor/Actuator Specific Device Model for An Endpoint Device

4.3 *ISO Standard*[1]

ISO 7498 — Basic Reference Model for Open Systems Interconnection

ISO/CEN EN14908 Specification for Control Networking Protocol

ANSI/EIA/CEA-709.1 (LONWORKS) Control Networking Protocol Specification, Rev B-2000

4.4 *Other Documents*

LONMARK Layers 1–6 Interoperability Guidelines, LONMARK International.

LONMARK Application Layer Interoperability Guidelines, LONMARK International.

Neuron Chip Data Book, Toshiba and Cypress Corporation.

Smart Transceiver Data Book, Echelon Corporation.

Neuron C Programmer's Guide, Revision 2, Echelon Corporation

Neuron C Reference Guide, Revision 2, Echelon Corporation

Standard Device Resource Report, http://types.lonmark.org, LONMARK International.

1 International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211, Geneva 20, Switzerland

Standard Program ID Definitions, http://types.lonmark.org/spid, LONMARK International.

SCPT Master List and Programmer's Guide, http:// types.lonmark.org/scpt, LONMARK International.

SNVT Master List and Programmer's Guide, http:// types.lonmark.org/snvt, LONMARK International.

5 Terminology

5.1 Terminology that is common to all of the documents in this SAN series may also be defined in the Sensor Actuator Network Standard (see reference SEMI E54 §4.1). Terminology may be reproduced here which is defined in other SEMI documents.

5.2 *Abbreviations and Acronyms*

5.2.1 *APDU* – Application Protocol Device Unit

5.2.2 *CDM* — Common Device Model

5.2.3 *CP* — Configuration Parameter

5.2.4 *DM* — Device Manager

5.2.5 *DS* — Device Status

5.2.6 *NCS* — Network Communications Standard

5.2.7 *Neuron* — Neuron Chip is the 8 bit hardware implementation of the ANSI/EIA/CEA-709.1 (LONWORKS) Control Networking Protocol

5.2.8 *NV* — Network Variable

5.2.9 *NVI* — Network Variable Input

5.2.10 *NVO* — Network Variable Output

5.2.11 *OSI* — Open Systems Interconnect

5.2.12 *OSS* — Object Services Standard

5.2.13 *RO* — Read Only

5.2.14 *RW* — Read/Write

5.2.15 *SAC* — Sensor, Actuator, Controller (object)

5.2.16 *SAN* — Sensor/Actuator Network

5.2.17 *SCPT* — Standard Configuration Parameter Type

5.2.18 *SDM* — Specific Device Model

5.2.19 *SNVT* — Standard Network Variable Type

5.2.20 *SPID* — Standard Program ID

5.3 *Device Component Definitions* — As this standard defines the presentation or mapping of CDM data structure and behavior over a network, it makes use of many of the terms in the SEMI E54.1 - CDM document. Table 1 provides a mapping of fundamental terminology of the CDM document into this document and the LONWORKS definitions. The symbol "=" indicates that the definition is used exactly as specified on the CDM specification.

5.4 In the following sections, additional clarification of some of these terms is provided in the context of the ANSI/EIA/CEA-709.1 (LONWORKS) protocol.

Table 1 Mapping of CDM to NCS Terminology

CDM Term	*NCS Equivalent*	*LONWORKS Equivalent*
Device	=	Device or node
Device Model	=	Device interface
Object	=, Class	Functional profile, Class
Instance	=	Functional block
Attribute	=	Network variable or configuration property
Behavior	=	=
Service	=	Network variable function or application message
State Diagram	=	=
Byte	=	=
Nibble	=	=
Character String	=	String of ASCII characters or string of international characters

5.4.1 *attribute* — Attributes are either input network variables (NVI), output network variables (NVO), or configuration properties (CP). Input and output network variables and configuration properties may be read and/or written by the device itself, and all attributes may be polled over the network. Additionally, input network variables and configuration properties may be updated over the network, and the receipt of such an update may cause an event to be propagated to the device's application layer. This corresponds to a RW (Read and Write) attribute of the object owning the network variable. Output network variables may not be updated over the network. This corresponds to a RO (Read Only) attribute of the object owning the network variable. When the device itself updates one of its output network variables, the value of that variable may be propagated over the network to destination addresses determined at installation time. Finally, configuration properties are attributes typically stored in non-volatile memory and preserved across device resets and power cycles.

5.4.2 *behavior* — Generic object behavior is specified by the *LONMARK Application Layer Interoperability Guidelines*. Additional object-specific behavior is specified by means of functional profiles.

5.4.3 *device* — A device (or node) typically consists of one network transceiver which implements the physical layer of the ANSI/EIA/CEA-709.1 (LONWORKS) protocol, one Neuron Chip with associated firmware which implements the other layers of the LONTALK protocol, and input/output hardware implementing the physical interface of the device to external sensor and/or actuator hardware. A LONWORKS device may optionally contain a host processor and associated software or firmware which implements the application layer of the ANSI/EIA/CEA-709.1 (LONWORKS) protocol.

5.4.4 *device model* — The device model comprises several elements which fully describe the external interface of the device for an interoperable network. The interface is made of the following pieces: a Device Manager (DM) object; a Sensor/Actuator/Controller (SAC) object; functional blocks such as sensors, actuators, and controllers; individual network variables; and configuration properties.

5.4.5 *instance* — Real devices may have zero or more instances of each of the defined functional blocks. Object instances are identified by means of an instance number within the device.

5.4.6 *object* — Functional blocks defined as a set of one or more network variable inputs and/or outputs, implemented as Standard Network Variable Types, and a set of configuration properties, implemented as Standard Configuration Property Types. Functional blocks form the basis of interoperability at the application layer. The functional blocks describe standard formats for how information is input to, and output from, a device, and shared with other devices on the network.

5.4.7 *service* — Request services are represented by ANSI/EIA/CEA-709.1input network variables (NVI) being set and delivered to the device application. Notification services are represented by ANSI/EIA/CEA-709.1output network variables being set by the device application and delivered over the network.

5.4.8 *state diagram* — In a LONWORKS device, state is represented by the collection of values of local and network variables of the application program. Transitions between states are the result of external events (such as the receipt of a network variable update, or other I/O event), or internal events (such as the expiration of a timer).

5.5 *LONWORKS-Specific Definitions* — In addition to the standard data type definitions for bit, nibble, byte, and character, the ANSI/EIA/CEA-709.1 (LONWORKS) protocol defines a set of standard data representations for use as attribute values.

5.5.1 *binding* — Network variables on the same or different devices may be associated together by means of a network management service known as binding. Binding is permitted only if all the network variables in the set are of the same data type. The values of network variables that are bound together are propagated over the network by the ANSI/EIA/CEA-709.1 (LONWORKS) protocol. Table 2 shows the permitted combinations for updating and polling of network variables.

Table 2 Updating and Polling of Network Variables

Network Variable Class	*Update from Network*	*Update from Device*	*Poll from Network*	*Poll from Device*
Input	Yes	Yes	Yes	Yes[#2]
Output	No	Yes[#1]	Yes	No

[#1] When the device updates one of its own output network variables, input network variables that are bound to this output network variable receive an update from the network.

[#2] When the device polls one of its own input or configuration network variables, output network variables that are bound to this input or configuration network variable receive a poll from the network.

5.5.2 *configuration properties* — These are attributes of a functional block that are used to configure the application-specific behavior of the functional block, such as sensor gain and offset, linearization table, and sample rate. These attributes are typically updated when the device is installed, configured, or calibrated, and are stored in non-volatile memory.

5.5.3 *device interface* — A device interface is a specification of one or more functional blocks, together with semantic definitions relating the behavior of the functional block(s) to the network variable values. The collection of functional blocks in a device corresponds to the SEMI SAN device-specific model for that device. Each type of device interface is identified by a standard program ID (SPID).

5.5.4 *network variable* — This is a network-visible data attribute of a device, with a well-defined data type. Network variables are either input variables, output variables, or configuration network variables. The value of a network variable may be updated either by the device itself, or over the network by some other device. This corresponds to the SEMI specific service SetAttribute operation. The value of a network variable may be polled over the network by some other device, or retrieved by the device itself. This corresponds to the SEMI specific service GetAttribute operation.

5.5.5 *functional block* — A device's external interface documentation specifies the type identifiers of the functional blocks contained within the device. This documentation may be uploaded from the device, and completely specifies the functional profiles implemented by the device, as well as the network variables and configuration properties contained within each of the functional blocks.

5.5.6 *functional profile* — A functional profile is the definition of the attributes and behaviors of an abstract entity. Each type of functional block is identified by a functional profile number. A specific device type consists of instantiations of one or more of these functional profiles. A functional block is implemented as a collection of network variables and configuration properties.

5.5.7 *standard configuration property types* — These data types, also known as SCPTs, provide a data type definition and a semantic behavior for the configuration properties of functional blocks. A list of all available SCPTs and details of their definitions is provided in the *SCPT Master List and Programmer's Guide*.

5.5.8 *standard network variable types* — These data types, also known as SNVTs, facilitate interoperability by providing a well-defined interface for communication between devices made by different manufacturers. A device may be installed in a network, and logically connected to other devices via network variables, as long as the data

types match. A list of all available SNVTs and details of their definitions is provided in the *SNVT Master List and Programmer's Guide*.

6 Communication Protocol High Level Structure

6.1 The ANSI/EIA/CEA-709.1 (LONWORKS) protocol is based on a seven-layer architecture. At each layer, there is a description of the services provided within that layer. The high level protocol architecture is shown in Figure 1.

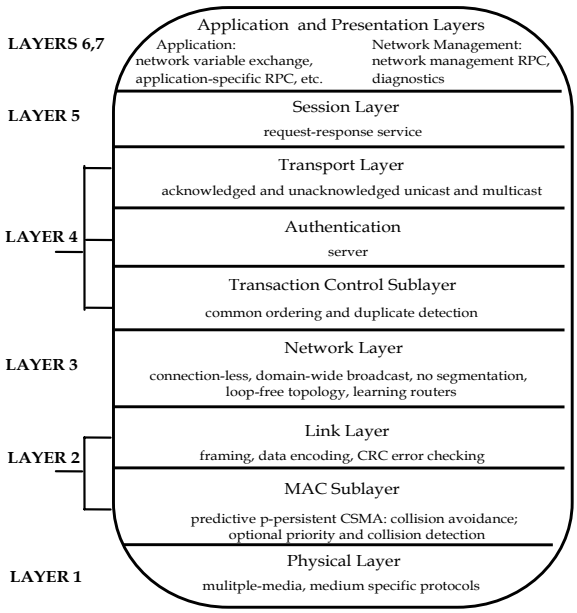


Figure 1
Layered View of the ANSI/EIA/CEA-709.1 (LONWORKS) Protocol

6.1.1 Figure 1 represents a conceptual view of the device architecture. Implementations typically use the Neuron Chip and its associated firmware, which provide a conforming implementation of layers 2 through 6. The *LonMark Interoperability Guidelines* specify the protocol options to be used, most specifically at the physical layer (network transceivers) and at the application layer (object model).

6.2 *Physical Layer —* The device shall employ one of the LONMARK-approved physical channels as specified in the *LONMARK Layers 1–6 Interoperability Guidelines*. LONWORKS-based SEMI SAN-compliant devices shall use, by default, a two-pin screw-terminal open pluggable connector (Weidmüller-Klippon SL2, Phoenix Combicon, or equivalent) for the network connection. The default connector specification may be overridden for specific device types if special requirements apply; any such overrides shall be noted in §9, *Specific Device Type Information*, of this document. This connection is polarity-insensitive. The requirements of semiconductor equipment may be met by one of the twisted pair channel specifications listed below.

6.2.1 *TP/XF-1250 Twisted Pair —* This twisted-pair channel operates at a bit rate of 1,250kbps and supports a bus topology using transformer-coupled transceivers.

6.2.2 *TP/FT-10 Twisted Pair —* This twisted-pair channel operates at a bit rate of 78kbps and supports both free topology and bus topology wiring, as well as optional link power.

6.2.3 The *LONMARK Layers 1-6 Interoperability Guidelines* provide specific details of the characteristics of these transceivers. This document also provides specifications of wiring types and interconnection topologies to be used for guaranteed device interoperability. Note that the ANSI/EIA/CEA-709.1 (LONWORKS) protocol supports heterogeneous networks. Devices with dissimilar transceivers may be interconnected and communicate via routers or repeaters. Similarly, routers and repeaters may be used to extend a physical channel beyond the device count, wire length, or other physical limitations imposed by the chosen transceiver.

6.2.4 Multiple physical layer protocols and data encoding methods are used in the ANSI/EIA/CEA-709.1 (LONWORKS) protocol. Differential Manchester encoding is used on twisted pair physical layers.

6.3 *Link Layer* — The device shall comply with the ANSI/EIA/CEA-709.1 (LONWORKS) protocol link layer specification. This layer includes the media access control sublayer. For a number of reasons, including simplicity and compatibility with the multicast protocol, the ANSI/EIA/CEA-709.1 (LONWORKS) protocol supports a simple connectionless service. Its functions are limited to framing, frame encoding, and error detection.

6.3.1 *Media Access Control Sub-layer* — In order to deal with a variety of media in the potential absence of collision detection, the MAC (Media Access Control) sub-layer employs a collision avoidance algorithm called Predictive *p*-persistent CSMA (Carrier Sense, Multiple Access).

6.4 *Network Layer* — The device shall comply with the ANSI/EIA/CEA-709.1 (LONWORKS) protocol network layer specification. This layer handles packet delivery within a single domain, with no provisions for inter-domain communication. The network service is connection-less, unacknowledged, and supports neither segmentation nor re-assembly of messages. The routing algorithms employed by the network layer to learn the topology assume a tree-like network topology; routers with configured tables may operate on topologies with physical loops, as long as the communication paths are logically tree-like. In this configuration, a packet may never appear more than once at the router on the side on which the packet originated. The unicast routing algorithm uses learning for minimal overhead and no additional routing traffic. Use of configured routing tables is supported for both unicast and multicast addresses.

6.5 *Transport Layer* — The device shall comply with the ANSI/EIA/CEA-709.1 (LONWORKS) protocol transport layer specification. The heart of the protocol hierarchy is the Transport and Session layers. A common Transaction Control sub-layer handles transaction ordering and duplicate detection for both layers. The transport layer is connectionless and provides reliable message delivery to both single and multiple destinations. Authentication of the message sender's identity is provided as an optional feature. The authentication server requires only the Transaction Control sub-layer to accomplish its function. The transport and session layer messages may be authenticated using all of the ANSI/EIA/CEA-709.1addressing modes other than broadcast. The transport layer supports end-to-end acknowledged service and an unacknowledged/ repeated service.

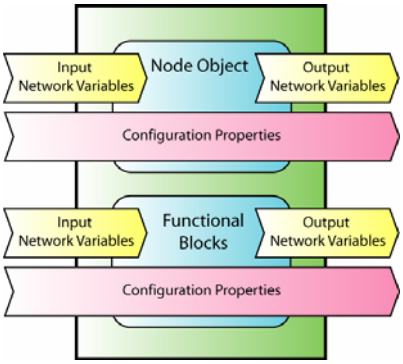
6.6 *Session Layer* — The device shall comply with the ANSI/EIA/CEA-709.1 (LONWORKS) protocol session layer specification. This layer implements a simple Request-Response mechanism for access to remote servers. This mechanism provides a platform upon which application-specific remote procedure calls can be built. The ANSI/EIA/CEA-709.1 (LONWORKS) network management protocol, for example, is dependent on the Request-Response mechanism in the Session layer, even though it accesses the protocol via the application layer interface.

6.7 *Presentation Layer* — The device shall comply with the ANSI/EIA/CEA-709.1 (LONWORKS) protocol presentation layer specification. The Presentation layer and the Application layer taken together form the foundation of interoperability for LONWORKS devices. The application layer provides all the usual services for sending and receiving messages, but it also contains the concept of network variables. The presentation layer provides information in the Application Protocol Data Unit (APDU) header for how the APDU is to be interpreted for network variable updates. This application-independent interpretation of the data allows data to be shared among devices without prior arrangement. With agreement on which network variables are to be used for sensors, actuators, etc., intelligent components from different manufacturers may work together without prior knowledge of each other's characteristics.

6.8 *Application Layer* — At the application layer, interoperability between LONWORKS-based devices is facilitated through the use of functional blocks and Standard Network Variable Types (SNVTs). Functional blocks build upon network variables and provide a concise application layer interface that incorporates semantic meaning for specific device functions. Functional blocks not only define which SNVTs to use to convey data, but also provide semantic meaning about the information being communicated. To aid in the specification of specific device models with well-defined functional behavior, functional block interfaces and semantics are defined by functional profiles. The Application Layer also includes the LONWORKS file transfer protocol, which provides segmentation and reassembly of arbitrary length files of data. This service may be used to get and set object attributes that exceed the network variable size limit of 31 bytes.

6.8.1 *Object Models* — The ANSI/EIA/CEA-709.1 (LONWORKS) protocol provides an object-oriented specification for defining and addressing network variables and configuration properties, which are the representation of object attributes and events. The device shall comply with the object model specifications defined in §7 of this document.

6.8.2 *Functional Block Structure* — The *LONMARK Application Layer Interoperability Guidelines* define a structure for *functional profiles*. Each functional profile may have a set of mandatory network variables, a set of optional network variables, a set of configuration properties (both mandatory and optional), and a manufacturer-defined section, which may be used for non-interoperable extensions to the profile. This is illustrated in Figure 2. This notation is defined in the *LONMARK Application Layer Interoperability Guidelines.*



NOTE: Diagram notation, the arrow-like symbol used in Figure 2 is defined in the LonMark Application Layer Interoperability Guidelines.

Figure 2
LonMark Object Structure

6.8.3 The *LONMARK Application Layer Interoperability Guidelines* provide for the definition of Standard Network Variable Types, Standard Configuration Property Types, and Functional Profiles. In the mapping of the SEMI CDM to the LONMARK object structure in §7, extensions to the current SNVT list and LONMARK Interoperability Guidelines are marked with an asterisk (*). Functional profile numbers are specified by the guidelines; a device may consist of one instance of a Node Object type, and one or more instances of other functional profiles.

6.9 *Network Management* — The ANSI/EIA/CEA-709.1 (LONWORKS) protocol defines a complete network management and diagnostic protocol for LONWORKS devices. This protocol is a layer above the Session layer (request/response service) and provides mechanisms for application downloading, device address assignment, distribution of destination addresses for implicit messaging, router configuration, and device-level diagnostics. The *LONMARK Application Layer Interoperability Guidelines* define a device management layer for functional blocks.

7 Required and Optional Object Types

7.1 The LONMARK guidelines do not require any specific objects to exist in a device in order to be a compliant LONMARK device, except that a Node object functional block is required for devices that contain multiple functional blocks (SEMI compliant devices will typically have a Node object functional block). New LONMARK standard profiles are defined in this standard to identify and describe functional blocks that shall exist in devices that are to be interoperable and interchangeable on a LONMARK SEMI compliant SAN network.

7.1.1 LONMARK International Association publishes functional profiles for various sensor, actuator, and controller objects. A specific device may be implemented using functional blocks based on these profiles. The Common Device Model specification additionally identifies two functional blocks (namely the Device Manager (DM) and Sensor Actuator Controller (SAC) objects) that must exist in all SEMI compliant SAN devices. The required functional profiles for a SEMI compliant SAN device utilizing the network communication specification described herein necessarily comprise, at minimum, the union of the LONMARK functional profile requirements and the CDM specification requirements.

7.1.2 A list of required and optional object types is given in Table 3. Additional objects that are specified in a particular SDM are given identifiers in that SDM specification. The LONMARK specific presentation information for these identifiers is given in §9 of this document.

Table 3 Common Device Model Required and Optional Object Types

SEMI Object Name	*LONMARK SEMI Class ID/Instance ID*[#1]	*CDM Tag*[#2]	*LONMARK Functional Profile Name*	*Required By LONMARK*[#1]	*Required By CDM*[#2]	*Required By NCS*
(DM) Device Manager[#4]	0 / 1	DmIO	Node Object	Yes	Yes	Yes
(SAC) Sensor/Actuator/Controller[#4]	0 / 1	SacIO	Node Object	Yes	Yes	Yes
Assembly	180.81 / 1 through i	Asm	Manufacturer-specific Members	No	No	No
Local Link	Turnaround Connection[#3]	Lnk	Turnaround Connection[#3]	No	No	No
Sensor-AI	180.11 / 1 through k	Sai	SFPTsemiSensorAI	No	No	No
Sensor-EI	180.22 / 1 through l	Sei	SFPTsemiSensorEI	No	No	No
Sensor-BI	180.18 / 1 through m	Sbi	SFPTsemiSensorBI	No	No	No
Actuator-AO	180.31 / 1 through n	Aao	SFPTsemiActuatorAO	No	No	No
Actuator-EO	180.34 / 1 through o	Aeo	SFPTsemiActuatorEO	No	No	No
Actuator-BO	180.33 / 1 through p	Abo	SFPTsemiActuatorBO	No	No	No
Controller	180.51 / 1 through q	C	SFPTsemiController	No	No	No
Sensor-BI-TH	180.19 / 1 through s	Sbith	SFPTsemiSensorBITH	No	No	No

[#1] LonMark groups SEMI Class IDs as reference 180 and designates specific object name categories as .XX. See LONMARK Application Layer Interoperability Guideline and http://types.lonmark.org.

[#2] See E54.1 – CDM specification for further information

[#3] A turnaround connection is not a profile, but is a protocol-specific method for creating a link

[#4] The LonMark Application Layer maps the SEMI Device Manger (DM) and Sensor/Actuator/Controller (SAC) objects to the LonMark Node object.

7.1.3 *Service Requests Code* — Most service requests are implemented as network variable updates addressed to the network variable corresponding to the specified attribute. The exceptions to this mapping are requests that require more than a 31 byte response. In that case, the request is implemented as a network variable update containing a file request to an **nviSemiReq** input to the Node Object functional block of a device. This input is defined by the **SNVT_semi_req** type as follows:

```
typedef enum {
 struct {
   CMD_RESET                  = 1,
   CMD_ABORT                  = 2,
   CMD_RECOVER                = 3,
   CMD_GET_ATTRIBUTE          = 4,
   CMD_SET_ATTRIBUTE          = 5,
   CMD_OERATE                 = 6,
   CMD_RESTOTE_DEFAULT        = 7,
   CMD_PUBLISH_ATTRIBUTE      = 8,
   CMD_LOCK                   = 9,
```

```
  CMD_UNLOCK                = 10,
  CMD_GET_EXCEPTION_QUEUE   = 11,
  CMD_CLEAR_EXCEPTION_QUEUE = 12,
  CMD_EXECUTE               = 13,
  CMD_PERFORM_DIAGNOSTICS   = 14,
} semi_request_t;
typedef struct {
  unsigned int flblock_index;
  semi_request_t service_code;
  unsigned int parameter;          // Optional parameter
} SNVT_semi_req;
```

7.1.4 A device accepts a **SNVT_semi_req** request by updating the **nvoSemiStat** and **nvoFileStat** outputs of the Node Object function block. The **nvoSemiStat** output is defined by the **SNVT_semi_status** type as follows:

```
typedef enum {
 struct {
  STAT_SUCCESS              = 0,
  STAT_UNSUPPORTED_SERVICE  = 1,
  STAT_UNSUPPORTED_ATTRIBUTE = 2,
  STAT_SET_ATTRIBUTE_ERROR  = 3,
  STAT_GET_ATTRIBUTE_ERROR  = 4,
  STST_CALIBRATION_ERROR    = 5,
  STAT_SERVICE_REQUEST_ERROR = 6,
  STAT_DIAGNOSTIC_ERROR     = 7,
} semi_status_t;
typedef struct {
  semi_status_t status;
  unsigned long selected_file;
  struct address {           // Address of the requesting device
    unsigned short domain_id[6];
    unsigned short domain_length;
    unsigned short subnet;
    unsigned short node;
  }
} SNVT_semi_status;
```

7.1.5 If the **nvoSemiStat** and **nvoFileStat** outputs indicate the request was accepted, the requesting device then fetches the requested data by transferring the file with the file index reported by the **nvoSemiStat** response.

7.1.6 *Object Attributes* — The SEMI GetAttribute and SEMI SetAttribute service requests are implemented as network variable fetch, poll, and update requests addressed to the network variable corresponding to the specified attribute. This is appropriate when responses greater than 31 bytes are not required. A GetAttribute service request may be addressed directly to any network variable as a LonTalk request message, using the LonTalk protocol network management NV fetch mechanism. A GetAttribute service request may also be addressed to an output network variable as a LonTalk NV poll message, using the NV selection mechanism. A SetAttribute service request

may be addressed to an input network variable as a LonTalk NV update message, using the NV selection mechanism. The confirmation of a SetAttribute (network variable update) is provided by the acknowledged service of the LonTalk protocol transport layer.

7.1.7 *Sequence Numbers* — Each network variable in a functional block is identified by means of a self-documentation string stored in the device's memory. This string contains the functional block index of the functional block to which this variable belongs, and the Sequence Number of the network variable within its functional block. For the LONWORKS NCS, this sequence number is identical to the numerical sequence number specified by the CDM tag. As an example, using the Device Manager object instance declared as the second object instance in the device it has an Instance Id of 1. The Device Manager attribute Standard Revision Level has the tag DmA2. The self-documentation string for this network variable is, therefore, specified as "@1|2."

7.1.8 *Publish Service* — The Publish notification service is implicit when an output network variable is updated. The device propagates the value of the output network variable (equivalent to a read-only attribute) to any input network variable(s) to which it may be bound. The ANSI/EIA/CEA-709.1 (LONWORKS) protocol only supports propagation of output network variables. In CDM terminology, this means that only read-only attributes may be published. If a specific device model requires publication of a read/write attribute, an output network variable whose value mirrors the value of the input (read/write) network variable may be introduced to the object definition.

7.2 *Sensor/Actuator/Controller Object (*)* — The SEMI CDM SAC object coordinates the functionality of Sensor, Actuator, and Controller objects in the device. The SAC object is mapped to the LONMARK Node object functional block. Each device must support one (and only one) Node object functional block. The SAC functional block as well as its common required and optional attributes, services and behavior are described in document SEMI E54.1 the CDM standard specification. Extensions to the current SNVT list and LONMARK Interoperability Guidelines are marked with an asterisk (*).

7.2.1 Extensions to the Node Object functional profile are defined in the LONMARK Interoperability Guidelines, which form part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS technology. Table 4 summarizes the network variables that implement the attributes of the SAC object. Table 5 summarizes the services implemented by the SAC object.

Table 4 Sensor/Actuator/Controller (SAC) Object Network Variables

Sensor/Actuator/Controller Object (SAC) *Profile ID = 00, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	CDM Tag	Standard NV or CP Data Type
1	Last Calibration Date	CP	SacA1	SCPTlastCalibrationDate (*)[#2]
2	Next Calibration Date	CP	SacA2	SCPTnextCalibrationDate (*)[#2]
3	Expiration Timer	NVO	SacA3	SNVT_time_hour
4	Expiration Warning Enable	CP	SacA4	SCPTexpirationWarningEnable (*)[#2]
5	Run Hours	NVO	SacA5	SNVT_time_hour
128	Attribute ID[#1] (*)[#2]	NVI	SacA65	SNVT_member (*)[#2]

[#1] This input is used to specify an attribute for the Restore Default and Publish Attribute services.

[#2] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 5 Sensor/Actuator/Controller (SAC) Object Services

Sensor/Actuator/Controller Object (SAC) *Profile ID = 00, Instance ID = 01*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	SacS1	Node Object RQ_RESET	
2	Abort	SacS2	Node Object RQ_ABORT (*)[#2]	
3	Recover	SacS3	Node Object RQ_RECOVER (*)[#2]	
4	GetAttribute	SacS4	Read NV or CP	NV or CP Value
5	SetAttribute	SacS5	Write NV or CP	

Sensor/Actuator/Controller Object (SAC) *Profile ID = 00, Instance ID = 01*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
6	Operate	SacS6	Node Object RQ_NORMAL	
7	Restore Default[#1]	SacS7	Node Object RQ_RESTORE (*)[#2]	NVO Value
8	Publish Attribute[#1]	SacS8	Node Object RQ_PROPAGATE (*)[#2]	NVO Value

[#1] These services may use the new Attribute ID input to the Node Object functional block.

[#2] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.3 Device Manager Object (*) — The Device Manager object is the device component responsible for managing and consolidating the device operation. The Device Manager object is mapped to the LONMARK Node object functional block. Each device must support one (and only one) Node Object functional block. The DM functional block as well as its common required and optional attributes, services and behavior are described in SEMI E54.1 the CDM standard specification. Extensions to the current SNVT list and LONMARK Interoperability Guidelines are marked with an asterisk (*).

7.3.1 The SEMI CDM Device Manager functional block, which is mapped to the LONMARK Node object, combines attributes of device self-documentation with an exception reporting mechanism. A new functional profile is, therefore, defined with the following mandatory configuration parameters, network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS. Table 6 summarizes the network variables that implement the attributes of the Device Manager functional block.

Table 6 Device Manager (DM) Object Network Variables

Device Manager Object (DM) *Profile ID = 00, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
11	Device Type	CP	DmA1	SCPTdeviceType (*)[#5]
12	Standard Revision Level	CP	DmA2	SCPTdeviceRevLevel (*)[#5]
13	Device Manufacture Identifier	CP	DmA3	SCPTdeviceManufacturer (*)[#5]
14	Manufacturer Model Number	CP	DmA4	SCPTdeviceModelNumber (*)[#5]
15	S/W or F/W Revision Level	CP	DmA5	SCPTappRevLevel (*)[#5]
16	Hardware Revision Level	CP	DmA6	SCPThardwareRevLevel (*)[#5]
17	Serial Number	CP	DmA7	SCPTserialNumber
18	Device Configuration	CP	DmA8	SCPTdeviceConfiguration (*)[#5]
19	Device Status	NVO	DmA9	SNVT_dev_status (*)[#5]
20	Reporting Mode	CP	DmA10	SCPT_rept_mode (*)[#5]
21	Exception Status Report Interval	CP	DmA11	SCPT_exc_sts (*)[#5]
22	Exception Status	NVO	DmA12	SNVT_alarm_2
23	Exception Detail Alarm	NVO	DmA13	SNVT_exc_detail (*)[#5]
24	Exception Detail Warning	NVO	DmA14	SNVT_exc_detail (*)[#5]
25	Visual Indicator	CP	DmA15	SCPTvisualIndicator (*)[#5]
26	Alarm Enable	NVI	DmA16	SNVT_obj_request
27	Warning Enable	NVI	DmA17	SNVT_obj_request (*-add RQ_WARNING_NOTIFY_ENABLED and RQ_WARNING_NOTIFY_DISABLED)[#5]

Device Manager Object (DM) *Profile ID = 00, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
28	Exception Detail Type	CP	DmA18	SCPTexceptionDetailType (*)[#5]
29	Exception Detail Alarm Queue	NVO	DmA19	SNVT_exc_detail (*)[#5]
30	Exception Detail Warning Queue	NVO	DmA20	SNVT_exc_detail (*)[#5]
31	Date and Time	NVO	DmA21	SNVT_time_p (*)[#5]
32	Date and Time Type	CP	DmA22	SCPTdateTimeType (*)[#5]
66	Test ID[#2] (*)[#5]	NVI	DmA34	SNVT_test_id (*)[#5]
67	Password[#3] (*)[#5]	NVI	DmA35	SNVT_password (*)[#5]
68	Exception Queue[#4] (*)[#5]	NVI	DmA36	SNVT_exc_que (*)[#5]
128	Attribute ID[#1] (*)[#5]	NVI	DmA33	SNVT_member (*)[#5]

[#1] This input is used to specify an attribute for the Restore Default and Publish Attribute services.

[#2] This input is used to specify a test ID for the Perform Diagnostics service.

[#3] This input is used to specify a password for the Unlock service.

[#4] This input is used to specify a password for the Get and Clear Exception Queue services.

[#5] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.3.2 *Device Manager Functional Block Services* — Table 7 summarizes the services implemented by the Device Manager functional block.

Table 7 Device Manager (DM) Object Services

Device Manager Object (DM) *Profile ID = 00, Instance ID = 01*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	DmS1	Node Object RQ_RESET	
2	Abort	DmS2	Node Object RQ_ABORT (*)[#1]	
3	Recover	DmS3	Node Object RQ_RECOVER (*)[#1]	
4	GetAttribute	DmS4	Read NV or CP	NV or CP Value
5	SetAttribute	DmS5	Write NV or CP	
13	Execute	DmS6	Node Object RQ_ENABLE	
14	Perform Diagnostics	DmS7	Node Object RQ_SELF_TEST	
8	Publish Attribute	DmS8	Node Object RQ_PROPAGATE (*)[#1]	NVO Value
9	Lock	DmS9	Node Object RQ_LOCK (*)[#1]	
10	Unlock	DmS10	Node Object RQ_UNLOCK (*)[#1]	
11	Get Exception Queue	DmS11	Node Object RQ_GET_EXC_QUEUE (*)[#1]	
12	Clear Exception Queue	DmS12	Node Object RQ_CLEAR_EXC_QUEUE (*)[#1]	

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.3.3 The Publish (DmS8) notification service for the Device Manager exception status is implemented when an RQ_PROPAGATE request is received on the nviRequest input to the Node Object functional block. A specific network variable may be specified on the SNVT_member input. This causes the value of this network variable to be propagated across the network to other network variable(s) to which it may be bound. The implementation of the Device Manager functional block updates the specified output network variable according to the conditions

specified by the Reporting Mode and Exception Status Reporting Interval configuration properties of the functional block.

7.3.4 *Device Manager Object Configuration Properties* — The DM object has configuration properties to control exception reporting as shown in Table 8. These parameters are Standard Configuration Property Types (SCPTs).

7.3.4.1 The type SCPT_rept_mode(*) contains two four-bit fields specifying the reporting method for alarms and warning conditions. For example, in Neuron C, the application programming language used on the Neuron Chip, the declaration of SCPT_rept_mode is as follows:

```
typedef enum {
   REP_REQUEST                = 0,
   REP_REQ_LATCHED            = 1,
   REP_EVT_TRIGD_ON           = 2,
   REP_EVT_TRIGD_ONOFF        = 3,
   REP_TIME_TRIGD             = 4,
   REP_EVT_ON_TIME_TRIGD      = 5,
   REP_EVT_ONOFF_TIME_TRIGD   = 6,
} rept_mode_t;
typedef struct {
   rept_mode_t alarm_rept_mode :4;
   rept_mode_t warn_rept_mode  :4;
} SCPT_rept_mode;
```

7.3.4.2 The type SCPT_exc_sts(*) is a 16-bit value representing times from 0.00 to 655.35 seconds, with a resolution of 0.01 seconds. This parameter is optional. The default reporting mode is REP_REQUEST.

7.3.4.3 *Device Manager Functional Block Output Network Variables* — The Device Manager functional block of the CDM defines the attribute variable Device Status. The data type SNVT_dev_status is an enumeration. One field of the union is an enumeration, corresponding to the device status attribute defined by the table Device Status Attribute Values of the CDM. The values of this type are defined in Table 8.

Table 8 Device Status Attribute Variable Enumeration Values

Value	*Enumeration Tag*
0	DS_UNKNOWN
1	DS_INIT_SELFTEST
2	DS_IDLE
3	DS_SELFTEST_EXCPT
4	DS_EXECUTING
5	DS_ABORT_1
6	DS_ABORT_2

7.3.4.4 The application programming language used to declare SNVT_dev_status is as follows:

```
typedef enum {
   DS_UNKNOWN                 = 0,
   DS_INIT_SELFTEST           = 1,
   DS_IDLE                    = 2,
   DS_SELFTEST_EXCPT          = 3,
   DS_EXECUTING               = 4,
```

```
   DS_ABORT_1                  = 5,
   DS_ABORT_2                  = 6,
} dev_status_t;
```

7.3.4.4.1 The value DS_ABORT_1 corresponds to the *Abort from Idle* or *Executing* state, and the value DS_ABORT_2 corresponds to the *Abort from Initialized/Self Testing or Self Test Exception* state of the DM object.

7.3.4.5 The type SNVT_exc_que(*) contains four fields specifying the exception queue and exception elements to retrieve. For example, in Neuron C, the application programming language used on the Neuron Chip, the declaration of SNVT_exc_que_mode is as follows:

```
typedef enum {
   REP_ALARM          = 0,
   REP_WARNING        = 1,
} que_t;
typedef struct {
   que_t exception_type :1;
   int element_number   :1;
    SNVT_state exception_list[13];
    int exception_clear  :1;
} SNVT_exc_que;
```

7.3.4.6 The type SNVT_exc_detail(*) is a sequence of three structures containing arrays. The ANSI/EIA/CEA-709.1 protocol limits the size of each of these arrays to 9 bytes, so that the type fits within the network variable size limit of 31 bytes. For example, in Neuron C, the declaration of SNVT_exc_detail is as follows:

```
typedef struct {
   u_char comn_exc_size;
   int    calibration      : 1;
   int    real_time        : 1;
   int    communic         : 1;
   int    RAM              : 1;
   int    EEPROM           : 1;
   int    EPROM            : 1;
   int    microproc        : 1;
   int    diagnostic       : 1;
/*--------------------------*/
   int    resvd1           : 1;
   int    reset            : 1;
   int    notify_mfr       : 1;
   int    maintenance      : 1;
   int    power_inputV     : 1;
   int    power_outptV     : 1;
   int    power_resvd      : 1;
   int    power_overC      : 1;
   u_char comn_exc_dtl[7];
```

```
/*--------------------------*/
   u_char dev_exc_size;
   u_char dev_exc_dtl[9];
/*--------------------------*/
   u_char mfr_exc_size;
   u_char mfr_exc_dtl[9];
} SNVT_exc_detail;
```

7.4 *Sensor, Actuator, and Controller Functional Blocks* — These functional blocks are necessarily specific to the Specific Device Models. The *LONMARK Application Layer Interoperability Guidelines* provide a framework for defining functional profiles. Specific Device Models may employ these functional profiles, and/or may define their own functional profiles and Standard Network Variable Types for device-specific requirements. As long as the functional profile definition guidelines are followed, these profiles may be proposed to the LONMARK Interoperability Association as new standard profiles.

7.5 Assembly Object (Asm) – The Assembly (Asm) object instances may be used to provide for grouping more than one attribute from one or more functional blocks in a device into a single data structure for communication over the LonWorks network. The presentation of Assembly object instance attributes are indicated in Table 9. The presentation of Assembly object services are indicated in Table 10.

Table 9 Assembly Object Network Variables

Assembly Object (Asm) ***Profile ID = 180.81, Instance ID = 01 through i***				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Data	File	AsmA1	LONMARK file transfer NVs and messaging interface.

Table 10 Assembly Object Network Services

Assembly Object (Asm) ***Profile ID = 180.81, Instance ID = 01 through i***				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
4	GetAttribute	AsmS4	File transfer read	File transfer – Context Specific.
5	SetAttribute	AsmS5	File transfer write – Context Specific	

7.6 *Local Link Object (Lnk)* — The Local Link (Lnk) object instances may be used to "link" an attribute of one object instance to an attribute of another object instance. Refer to SEMI E54.1 – CDM standard for further explanation and use of this object. The presentation of Local Link object instance attributes are indicated in Table 11. The presentation of Local Link object services are indicated in Table 12.

Table 11 Local Link Object Network Variables

Local Link Object (Lnk)				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Source Object Class	NV Config Table/Address Table[#1]	LnkA1	SNVTlnk_context. The structure of this SNVT in context specific.
2	Source Object Instance	NV Config Table/Address Table[#1]	LnkA2	SNVTlnk_context. The structure of this SNVT in context specific.
3	Source Object Attribute	NV Config Table/Address Table[#1]	LnkA3	SNVTlnk_context. The structure of this SNVT in context specific.
4	Destination Object Class	NV Config Table/Address Table[#1]	LnkA4	SNVTlnk_context. The structure of this SNVT in context specific.
5	Destination Object Instance	NV Config Table/Address Table[#1]	LnkA5	SNVTlnk_context. The structure of this SNVT in context specific.
6	Destination Object Attribute	NV Config Table/Address Table[#1]	LnkA6	SNVTlnk_context. The structure of this SNVT in context specific.
7	Commit	NV Config Table/Address Table[#1]	LnkA7	SNVT_switch (TRUE, FALSE).

[#1] This is not an NV or CP. It is configuration information contained within the device's NV configuration table and address table that specifies a network variable connection between two or more network variables on the device. It can only be used to connect network variables of the same type. The link is created as soon as the tables are updated, so the Commit operation is not a separate step.

Table 12 Local Link Object Network Services

Local Link Object (Lnk)				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
4	GetAttribute	AsmS4	Read NV config table and address table.	Attribute Value
5	SetAttribute	AsmS5	Write NV config table and address table.	

7.7 *Sensor-AI Object (Sai)* — The presentation of the Sensor Analog Input (Sensor-AI) object instance attributes are as indicated in Table 13. The presentation of the Sensor-AI object services is indicated in Table 14.

Table 13 Sensor-AI Object Instance Network Variables

Sensor-AI Object (Sai) ***Profile ID = 180.11, Instance ID = 01 through k***				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	SaiA1	Typedef unsigned char SCPTname (*)[#5]
2	Status	NVO	SaiA2	nvoStatus output of the Sensor-AI Object
3	Alarm Enable	NVI	SaiA3	nviRequest input of the Sensor-AI Object

Sensor-AI Object (Sai) *Profile ID = 180.11, Instance ID = 01 through k*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
4	Warning Enable	NVI	SaiA4	nviRequest input of the Sensor-AI Object
16	Value	NVO	SaiA16	Typedef (*)[#5] SNVT_xxx[#4]
17	ReportInhibitTimer	CP	SaiA17	SCPTminSendTime
18	EnableReportRate	CP	SaiA18	SCPTmaxSendTime[#1]
19	ReportRate	CP	SaiA19	SCPTmaxSendTime
64	Offset	CP	SaiA64	SCPToffset
65	Gain	CP	SaiA65	SCPTgain
66	DataType	CP	SaiA66	SCPTnvType
67	DataUnits	CP	SaiA67	SCPTnvType
68	SafeState	CP	SaiA68	Typedef unsigned byte SCPTsafeState (*)[#5]
69	EnableReportDelta	CP	SaiA69	SCPTsndDelta[#2]
70	ReportDelta	CP	SaiA70	SCPTsndDelta
71	EnableReportROC	CP	SaiA71	Typedef BOOL SCPTreportROC[#3] (*)[#5]
72	ReportROC	CP	SaiA72	SCPTreportROC (*)[#5]
73	AlarmTripPointHigh	CP	SaiA73	SCPThighLimit2
74	AlarmTripPointLow	CP	SaiA74	SCPTlowLimit2
75	AlarmHystersis	CP	SaiA75	SCPThystLow2 and SCPThystHigh2
76	WarningTripPointHigh	CP	SaiA76	SCPThighLimit1
77	WarningTripPointLow	CP	SaiA77	SCPTlowLimit1
78	WarningHystersis	CP	SaiA78	SCPThystLow1 and SCPThystHigh1

[#1] A SCPTmaxSendTime value of 0 is used to inhibit report rate reporting.

[#2] A SCPTsndDelta value of 0 is used to inhibit value delta reporting.

[#3] A SCPTreportROC value of 0 is used to inhibit ROC reporting.

[#4] The form and content of this SNVT is context-specific.

[#5] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 14 Sensor-AI Object Instance Network Services

Sensor-AI Object (Sai) *Profile ID = 180.11, Instance ID = 01 through k*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	SaiS1	Sensor-AI Object RQ_RESET	
2	Abort	SaiS2	Sensor-AI Object RQ_ABORT (*)[#1]	
3	Recover	SaiS3	Sensor-AI Object RQ_RECOVER (*)[#1]	
6	Operate	SaiS4	Sensor-AI Object RQ_NORMAL	
4	GetAttribute	SaiS5	Read NV or CP	NV or CP Value
5	SetAttribute	SaiS6	Sensor-AI Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	SaiS7	Sensor-AI Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.7.1 *Sensor-AI Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.7.1.1 *SNVT_xxx* — The *value* of the sensor (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.7.2 *Sensor-AI Object Configuration Parameters* — The Sensor-AI object has configuration properties to control exception reporting as shown in Table 15. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.8 *Sensor-EI Object (Sei)* — The presentation of the Sensor Enumerated Input (Sensor-EI) object instance attributes are as indicated in Table 15. The presentation of object services shall be indicated in Table 16.

Table 15 Sensor-EI Object Instance Network Variables

Sensor-EI Object (Sei) ***Profile ID = 180.22, Instance ID = 01 through l***				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	SeiA1	Typedef unsigned char SCPTname (*)[#3]
2	Status	NVO	SeiA2	nvoStatus output of the Sensor-EI Object
3	Alarm Enable	NVI	SeiA3	nviRequest input of the Sensor-EI Object
4	Warning Enable	NVI	SeiA4	nviRequest input of the Sensor-EI Object
16	Value	NVO	SeiA16	Typedef (*)[#3] SNVT_xxx[#2]
17	ReportInhibitTimer	CP	SeiA17	SCPTminSendTime
18	EnableReportRate	CP	SeiA18	SCPTmaxSendTime[#1]
19	ReportRate	CP	SeiA19	SCPTmaxSendTime
64	DebounceControl	CP	SeiA64	SCPTdebounce
65	AlarmState	CP	SeiA65	Typedef BOOL SCPTalarmState (*)[#3]
66	WarningState	CP	SeiA66	Typedef BOOL SCPTwarningState (*)[#3]

[#1] A SCPTmaxSendTime value of 0 is used to inhibit report rate reporting.

[#2] The form and content of this SNVT is context-specific.

[#3] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 16 Sensor-EI Object Instance Network Services

Sensor-EI Object (Sei) ***Profile ID = 180.22, Instance ID = 01 through l***				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	SeiS1	Sensor-AI Object RQ_RESET	
2	Abort	SeiS2	Sensor-AI Object RQ_ABORT (*)[#1]	
3	Recover	SeiS3	Sensor-AI Object RQ_RECOVER (*)[#1]	
6	Operate	SeiS4	Sensor-AI Object RQ_NORMAL	
4	GetAttribute	SeiS5	Read NV or CP	NV or CP Value
5	SetAttribute	SeiS6	Sensor-AI Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	SeiS7	Sensor-AI Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.8.1 *Sensor-EI Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.8.1.1 *SNVT_xxx* — The *value* of the sensor (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.8.2 *Sensor-EI Object Configuration Parameters* — The Sensor-EI object has configuration properties to control exception reporting as shown in Table 17. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.9 *Sensor-BI Object (Sbi)* — The presentation of the Sensor Binary Input (Sensor-BI) object instance attributes are as indicated in Table 17. The presentation of Sensor-BI object services is indicated in Table 18.

Table 17 Sensor-BI Object Instance Network Variables

Sensor-BI Object (Sbi) ***Profile ID = 180.18, Instance ID = 01 through m***				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	SbiA1	Typedef unsigned char SCPTname (*)[#3]
2	Status	NVO	SbiA2	nvoStatus output of the Sensor-BI Object
3	Alarm Enable	NVI	SbiA3	nviRequest input of the Sensor-BI Object
4	Warning Enable	NVI	SbiA4	nviRequest input of the Sensor-BI Object
16	Value	NVO	SbiA16	Typedef (*)[#3] SNVT_xxx[#2]
17	ReportInhibitTimer	CP	SbiA17	SCPTminSendTime
18	EnableReportRate	CP	SbiA18	SCPTmaxSendTime[#1]
19	ReportRate	CP	SbiA19	SCPTmaxSendTime
64	DebounceControl	CP	SbiA64	SCPTdebounce
65	AlarmState	CP	SbiA65	Typedef BOOL SCPTalarmState (*)[#3]
66	WarningState	CP	SeiA66	Typedef BOOL SCPTwarningState (*)[#3]

[#1] A SCPTmaxSendTime value of 0 is used to inhibit report rate reporting.

[#2] The form and content of this SNVT is context-specific.

[#3] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 18 Sensor-BI Object Instance Network Services

Sensor-BI Object (Sbi) ***Profile ID = 180.18, Instance ID = 01 through m***				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	SbiS1	Sensor-BI Object RQ_RESET	
2	Abort	SbiS2	Sensor-BI Object RQ_ABORT (*)[#1]	
3	Recover	SbiS3	Sensor-BI Object RQ_RECOVER (*)[#1]	
6	Operate	SbiS4	Sensor-BI Object RQ_NORMAL	
4	GetAttribute	SbiS5	Read NV or CP	NV or CP Value
5	SetAttribute	SbiS6	Sensor-BI Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	SbiS7	Sensor-BI Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.9.1 *Sensor-BI Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.9.1.1 *SNVT_xxx* — The *value* of the sensor (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.9.2 *Sensor-BI Object Configuration Parameters* — The Sensor-BI object has configuration properties to control exception reporting as shown in Table 19. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.10 *Sensor-BI-TH Object (Sbith)* — The presentation of the Sensor Binary Input Threshold (Sensor-BI-TH) object instance attributes are as indicated in Table 19. The presentation of Sensor-BI-TH object services is indicated in Table 20.

Table 19 Sensor-BI-TH Object Instance Network Variables

Sensor-BI-TH Object (Sbith) ***Profile ID = 180.19, Instance ID = 01 through s***				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	SbiA1	Typedef unsigned char SCPTname (*)[#4]
2	Status	NVO	SbiA2	nvoStatus output of the Sensor-BI-TH Object
3	Alarm Enable	NVI	SbiA3	nviRequest input of the Sensor-BI-TH Object
4	Warning Enable	NVI	SbiA4	nviRequest input of the Sensor-BI-TH Object
16	Value	NVO	SbiA16	Typedef (*)[#4] SNVT_xxx[#2]
17	ReportInhibitTimer	CP	SbiA17	SCPTminSendTime
18	EnableReportRate	CP	SbiA18	SCPTmaxSendTime[#1]
19	ReportRate	CP	SbiA19	SCPTmaxSendTime
64	DebounceControl	CP	SbiA64	Typedef unsigned char SCPTname (*)[#4]
65	AlarmState	CP	SbiA65	nvoStatus output of the Sensor-BI-TH Object
66	WarningState	CP	SbiA66	nviRequest input of the Sensor-BI-TH Object
67	ReadingValid	NVO	sbithA64	SNVT_switch[#3]
68	State	NVO	sbithA65	nvoStatus output of the Sensor-BI-TH Object
69	Status	NVO	sbithA66	nvoStatus output of the Sensor-BI-TH Object

[#1] A SCPTmaxSendTime value of 0 is used to inhibit report rate reporting.

[#2] The form and content of this SNVT is context-specific.

[#3] An invalid value output is used to indicate an invalid reading.

[#4] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 20 Sensor-BI-TH Object Instance Network Services

Sensor-BI-TH Object (Sbith) ***Profile ID = 180.19, Instance ID = 01 through s***				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	SeiS1	Sensor-BI-TH Object RQ_RESET	

Sensor-BI-TH Object (Sbith) *Profile ID = 180.19, Instance ID = 01 through s*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
2	Abort	SeiS2	Sensor-BI-TH Object RQ_ABORT (*)[#1]	
3	Recover	SeiS3	Sensor-BI-TH Object RQ_RECOVER (*)[#1]	
6	Operate	SeiS4	Sensor-BI-TH Object RQ_NORMAL	
4	GetAttribute	SeiS5	Read NV or CP	NV or CP Value
5	SetAttribute	SeiS6	Sensor-BI-TH Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	SeiS7	Sensor-BI-TH Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.10.1 *Sensor-BI-TH Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.10.1.1 *SNVT_xxx* — The *value* of the sensor (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.10.2 *Sensor-BI-TH Object Configuration Parameters* — The Sensor-BI-TH object has configuration properties to control exception reporting as shown in Table 19. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.11 *Actuator-AO Object (Aao)* — The presentation of the Actuator Analog Output (Actuator-AO) object instance attributes are as indicated in Table 21. The presentation of Actuator-AO object services is indicated in Table 22.

Table 21 Actuator-AO Object Instance Network Variables

Actuator-AO Object (Aao) *Profile ID = 180.31, Instance ID = 01 through n*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	AaoA1	Typedef unsigned char SCPTname (*)[#3]
2	Status	NVO	AaoA2	nvoStatus output of the Actuator-AO Object
3	Alarm Enable	NVI	AaoA3	nviRequest input of the Actuator-AO Object
4	Warning Enable	NVI	AaoA4	nviRequest input of the Actuator-AO Object
16	Setting	NVI	AaoA16	Typedef (*)[#3] SNVT_xxx[#2]
17	SafeState	CP	AaoA17	Typedef BOOL SCPTsafeState (*)[#3]
18	WatchRate	CP	AaoA18	SCPTmaxRcvTime
19	Watchdog	CP	AaoA19	SCPTmaxRcvTime[#1]
64	Offset	CP	AaoA64	SCPToffset
65	Gain	CP	AaoA65	SCPTgain
66	DataType	CP	AaoA66	SCPTnvType
67	DataUnits	CP	AaoA67	SCPTnvType

[#1] A value of 0 is used to disable the watchdog timeout.

[#2] The form and content of this SNVT is context-specific.

[#3] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 22 Actuator-AO Object Instance Network Services

Actuator-AO Object (Aao) *Profile ID = 180.31, Instance ID = 01 through n*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	AaoS1	Actuator-AO Object RQ_RESET	
2	Abort	AaoS2	Actuator-AO Object RQ_ABORT (*)[#1]	
3	Recover	AaoS3	Actuator-AO Object RQ_RECOVER (*)[#1]	
6	Operate	AaoS4	Actuator-AO Object RQ_NORMAL	
4	GetAttribute	AaoS5	Read NV or CP	NV or CP Value
5	SetAttribute	AaoS6	Actuator-AO Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	AaoS7	Actuator-AO Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.11.1 *Actuator-AO Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.11.1.1 *SNVT_xxx* — The *setting* of the actuator (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.11.2 *Actuator-AO Object Configuration Parameters* — The Actuator-AO object has configuration properties to control exception reporting as shown in Table 21. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.12 *Actuator-EO Object (Aeo)* — The presentation of the Actuator Enumerated Output (Actuator-EO) object instance attributes are as indicated in Table 23. The presentation of Actuator-EO object services is indicated in Table 24.

Table 23 Actuator-EO Object Instance Network Variables

Actuator-EO Object (Aeo) *Profile ID = 180.34, Instance ID = 01 through o*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	AeoA1	Typedef unsigned char SCPTname (*)[#3]
2	Status	NVO	AeoA2	nvoStatus output of the Actuator-EO Object
3	Alarm Enable	NVI	AeoA3	nviRequest input of the Actuator-EO Object
4	Warning Enable	NVI	AeoA4	nviRequest input of the Actuator-EO Object
16	Setting	NVI	AeoA16	Typedef (*)[#3] SNVT_xxx[#2]
17	SafeState	CP	AeoA17	Typedef BOOL SCPTsafeState (*)[#3]
18	WatchRate	CP	AeoA18	SCPTmaxRcvTime
19	Watchdog	CP	AeoA19	SCPTmaxRcvTime[#1]

[#1] A value of 0 is used to disable the watchdog timeout.

[#2] The form and content of this SNVT is context-specific.

[#3] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 24 Actuator-EO Object Instance Network Services

Actuator-EO Object (Aeo) *Profile ID = 180.34, Instance ID = 01 through o*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	AeoS1	Actuator-EO Object RQ_RESET	
2	Abort	AeoS2	Actuator-EO Object RQ_ABORT (*)[#1]	
3	Recover	AeoS3	Actuator-EO Object RQ_RECOVER (*)[#1]	
6	Operate	AeoS4	Actuator-EO Object RQ_NORMAL	
4	GetAttribute	AeoS5	Read NV or CP	NV or CP Value
5	SetAttribute	AeoS6	Actuator-EO Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	AeoS7	Actuator-EO Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.12.1 *Actuator-EO Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.12.1.1 *SNVT_xxx* — The *setting* of the actuator (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.12.2 *Actuator-EO Object Configuration Parameters* — The Actuator-EO object has configuration properties to control exception reporting as shown in Table 23. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.13 *Actuator-BO Object (Abo)* — The presentation of the Actuator Binary Output (Actuator-BO) object instance attributes are as indicated in Table 25. The presentation of Actuator-BO object services is indicated in Table 26.

Table 25 Actuator-BO Object Instance Network Variables

Actuator-BO Object (Abo) *Profile ID = 180.33, Instance ID = 01 through p*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	AboA1	Typedef unsigned char SCPTname (*)[#3]
2	Status	NVO	AboA2	nvoStatus output of the Actuator-BO Object
3	Alarm Enable	NVI	AboA3	nviRequest input of the Actuator-BO Object
4	Warning Enable	NVI	AboA4	nviRequest input of the Actuator-BO Object
16	Setting	NVI	AboA16	Typedef (*)[#3] SNVT_xxx[#2]
17	SafeState	CP	AboA17	Typedef BOOL SCPTsafeState (*)[#3]
18	WatchRate	CP	AboA18	SCPTmaxRcvTime
19	Watchdog	CP	AboA19	SCPTmaxRcvTime[#1]

[#1] A value of 0 is used to disable the watchdog timeout.

[#2] The form and content of this SNVT is context-specific.

[#3] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 26 Actuator-BO Object Instance Network Services

Actuator-BO Object (Abo) *Profile ID = 180.33, Instance ID = 01 through p*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	AboS1	Actuator-BO Object RQ_RESET	
2	Abort	AboS2	Actuator-BO Object RQ_ABORT (*)[#1]	
3	Recover	AboS3	Actuator-BO Object RQ_RECOVER (*)[#1]	
6	Operate	AboS4	Actuator-BO Object RQ_NORMAL	
4	GetAttribute	AboS5	Read NV or CP	NV or CP Value
5	SetAttribute	AboS6	Actuator-BO Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	AboS7	Actuator-BO Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.13.1 *Actuator-BO Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.13.1.1 *SNVT_xxx* — The *setting* of the actuator (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.13.2 *Actuator-BO Object Configuration Parameters* — The Actuator-BO object has configuration properties to control exception reporting as shown in Table 25. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

7.14 *Controller Object (C)* — The presentation of the Controller Output (C) object instance attributes are as indicated in Table 27. The presentation of Controller object services is indicated in Table 28.

Table 27 Controller-C Object Instance Network Variables

Controller Object (C) *Profile ID = 180.51, Instance ID = 01 through q*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
1	Name	CP	CA1	Typedef unsigned char SCPTname (*)[#2]
2	Status	NVO	CA2	nvoStatus output of the Controller Object
3	Alarm Enable	NVI	CA3	nviRequest input of the Controller Object
4	Warning Enable	NVI	CA4	nviRequest input of the Controller Object
16	Setpoint	CP	CA16	Typedef unsigned char SCPTname (*)[#2]
17	ProcessVariable	NVI	CA17	Typedef (*)[#2] SNVT_xxx[#1]
18	ControlVariable	NVO	CA18	Typedef (*)[#2] SNVT_xxx[#1]
19	DataType	CP	CA19	SCPTnvType
20	DataUnits	CP	CA20	SCPTnvType

Controller Object (C) *Profile ID = 180.51, Instance ID = 01 through q*				
Sequence Number	*Name*	*Storage Class*	*CDM Tag*	*Standard NV or CP Data Type*
21	AlarmSettleTime	CP	CA21	SCPTalrmSetT2
22	AlarmErrorBand	CP	CA22	SCPTalrmErrorBand (*)[#2]
24	WarningSettleTime	CP	CA24	SCPTalrmSetT1
25	WarningErrorBand	CP	CA25	SCPTwarningErrorBand (*)[#2]

[#1] The form and content of this SNVT is context-specific.

[#2] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 28 Controller-C Object Instance Network Services

Controller Object (C) *Profile ID = 180.51, Instance ID = 01 through q*				
Service Request Code	*Service Name*	*CDM Tag*	*Request Parameters*	*Result Parameters*
1	Reset	CS1	Controller Object RQ_RESET	
2	Abort	CS2	Controller Object RQ_ABORT (*)[#1]	
3	Recover	CS3	Controller Object RQ_RECOVER (*)[#1]	
6	Operate	CS4	Controller Object RQ_NORMAL	
4	GetAttribute	CS5	Read NV or CP	NV or CP Value
5	SetAttribute	CS6	Controller Object RQ_RESTORE (*)[#1]	NVO Value
7	RestoreDefault	CS7	Controller Object RQ_PROPAGATE (*)[#1]	NVO Value

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

7.14.1 *Controller Object Network Variables* — A new functional profile is defined with the following mandatory network variables and behaviors. This functional profile forms part of the LONMARK Functional Profiles for SEMI SAN-compliant devices based on LONWORKS.

7.14.1.1 *SNVT_xxx* — The *setting* of the actuator (associated with the object instance) that is to be made available on the network. The form and content of this SNVT is context-specific.

7.14.2 *Controller Object Configuration Parameters* — The Controller object has configuration properties to control exception reporting as shown in Table 27. These parameters are Standard Configuration Property Types (SCPTs). Extensions to the current SCPT list and Interoperability Guidelines are marked by an asterisk (*).

8 Protocol Compliance

8.1 A method of testing protocol compliance is required to verify implementation conformance to the standard. By virtue of the fact that the intermediate layers of the ANSI/EIA/CEA-709.1 (LONWORKS) protocol are implemented in commercially available silicon, compliance verification is needed only at the physical and application layers. The LONMARK Interoperability Association provides a compliance verification service to its members. When the SEMI Sensor/Actuator Network functional profiles are approved and incorporated by LONMARK International Association, this service may be used to verify compliance with the SEMI guidelines.

8.2 The certification procedure and checklist may be viewed and obtained by accessing web site http://www.lonmark.org.

8.3 Following the LonMark Interoperability Guidelines during product design will allow easy product certification. Application-specific functional profile(s) are available to download from the User Guides section to ease the specification process. All questions about guidelines and certification should be directed to “cert@lonmark.org”. Design assistance may be obtained by referencing “lm_confm.pdf”.

9 Specific Device Type Information

9.1 This section provides for the mapping of network-visible specific device model structure and behavior, specified in a SEMI standard Specific Device Model specification, to the ANSI/EIA/CEA-709.1 (LONWORKS) protocol. Each subsection is devoted to a single Specific Device Model specification. As additional SEMI SDM specifications are created, additional SDM mappings are added as subsections to this NCS specification. Unless otherwise noted, all attributes and services described are instance level attributes (as opposed to class level attributes). Device-type-specific items, such as overrides to the standard connector, may also be noted in these subsections.

9.1.1 Note that the formats of object instance attributes and services are detailed in the associated Specific Device Model specification. The presentation of the attributes and services to a LONWORKS network is detailed in the tables contained in the following sub-sections and in the *LONMARK Application Layer Interoperability Guidelines* document. Note that relationships between object classes, including inheritance are defined in the associated SDM specification and the CDM specification.

9.1.2 The instance identifier format of 1 through n, assigned to an object type, refers to the possibility of multiple instantiations of the object type. Refer to Table 3 of this document and the CDM document for a further explanation of object instance assignments.

9.2 *Specific Device Model For Mass Flow Device* — These sections detail the network mapping required to support the Specific Device Model For Mass Flow Device (see reference SEMI E54.3 of §4.1). Table 29 summarizes the Mass Flow Device object types. Subsequent Table 30 to Table 49 detail the instance attributes and services associated with each Mass Flow Device object type.

Table 29 Mass Flow Device Object Types

SEMI SDM Object Identifier	*Object Name*	*LONMARK Profile ID*	*LONMARK Functional Profile Name*
MFD1 (DM)	Device Manager	0	Node Object
MFD2 (SAC)	Sensor/Actuator/Controller	0	Node Object
MFD3	Sensor-AI-MF	180.13	SFPTsemiSensorAIMF
MFD4	Sensor-AI-AT	180.12	SFPTsemiSensorAIAT
MFD5	Assembly-MFM	180.82	SFPTsemiAssemblyMFM
MFD6	Sensor-AI-Aux	180.14	SFPTsemiSensorSensorAIAux
MFD7	Actuator-AO-MF	180.32	SFPTsemiActuatorAOMF
MFD8	Controller	180.51	SFPTsemiController
MFD9	Local Link	--	Turnaround Connection[#1]
MFD10	SISO	180.71	SFPTsemiSISO
MFD11	SISO-Setpoint	180.72	SFPTsemiSISOSetpoint
MFD12	Assembly-MFC	180.83	SFPTsemiAssemblyMFC

[#1] A turnaround connection is not a profile, but is a protocol-specific method of creating a link

9.2.1 *The presentation of the Mass Flow Device - Device Manager(DM) Object* — is as defined by SEMI E54.1, CDM standard specification. There are no additional object instance attributes or services required by this SDM.

9.2.2 *The presentation of the Mass Flow Device – Sensor/Actuator Controller (SAC) Object* — is as defined by SEMI E54.1, CDM standard specification. There are no additional object instance attributes or services required by this SDM.

9.2.3 *Sensor-AI-MF* — The presentation of the Sensor Analog Input Mass Flow (Sensor-AI-MF) object instance attributes are as indicated in Table 30. The presentation of Sensor-AI-MF object services is indicated in Table 31.

Table 30 Sensor-AI-MF Object Instance Network Variables

Sensor-AI-MF ***Profile ID = 180.13, Instance ID = 01 through r***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Flow Totalizer	NVO	A1	SNVT_vol_f
129	Flow Hours	NVO	A2	SNVT_time_hour
130	Zero Offset Mode	CP	A5	SCPTzeroOffsetMode (*)[#1]
131	Zeroing Status	NVO	A6	SNVT_zeroing_stat (*)[#1]
132	Autorange Status	NVO	A7	SNVT_autorange_stat (*)[#1]

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 31 Sensor-AI-MF Object Instance Network Services

Sensor-AI-MF ***Class ID = 180.13, Instance ID = 01 through r***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
129	Perform Zero Offset	S1	Controller Object RQ_ZERO_OFFSET (*)[#1]	
130	Query-Supported Gas Types	S2	Controller Object RQ_QUERY_GAS_TYPES (*)[#1]	LONMARK file transfer – Context Specific
131	Selected Programmed Gas Type	S3	Controller Object RQ_SELECT_GAS_TYPES (*)[#1]	LONMARK file transfer – Context Specific
132	Insert Gas Type	S4	Controller Object RQ_INSERT_GAS_TYPES (*)[#1]	LONMARK file transfer – Context Specific
133	Delete Gas Type	S5	Controller Object RQ_DELETE_GAS_TYPES (*)[#1]	LONMARK file transfer – Context Specific
134	Get Gas Calibration Data Value	S6	Controller Object RQ_GET_CAL_DATA (*)[#1]	LONMARK file transfer – Context Specific
135	Set Gas Calibration Data Value	S7	Controller Object RQ_SET_CAL_DATA (*)[#1]	LONMARK file transfer – Context Specific
136	Autorange	S8	Controller Object RQ_AUTORANGE (*)[#1]	

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

9.2.4 *Service Requests Code* — The service request is implemented as a network variable update containing a file request to an **nviSemiReq** input to the Sensor-AI-MF Object functional block of a device. This input is defined by the **SNVT_semi_req** type as follows:

```
typedef enum {
 struct {
  CMD_PERFORM_ZERO_OFFSET                = 129,
  CMD_QUERY_SUPPORTED_GAS_TYPES          = 130,
  CMD_SELECTED_PROGRAMMED_GAS            = 131,
```

```
  CMD_INSERT_GAS_TYPE                    = 132,
  CMD_DELETE_GAS_TYPE                    = 133,
  CMD_GET_GAS_CALIBRATION_DATA_VALUE     = 135,
  CMD_SET_GAS_CALIBRATION_DATA_VALUE     = 135,
  CMD_AUTORANGE                          = 136,
} semi_request_t;
typedef struct {
  unsigned int flblock_index;
  semi_request_t service_code;
  unsigned int parameter;          // Optional parameter
} SNVT_semi_req;
```

9.2.5 *Sensor-AI-AT* — The presentation of the Sensor Analog Input Ambient Temperature (Sensor-AI-AT) object instance attributes are as indicated in Table 32. The presentation of object services shall be indicated in Table 33.

Table 32 Sensor-AI-AT Object Instance Network Variables

Sensor-AI-AT ***Profile ID = 180.12, Instance ID = 00 through r***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
--	No Additional Attributes Defined	--	--	

Table 33 Sensor-AI-AT Object Instance Network Services

Sensor-AI-AT ***Profile ID = 180.12, Instance ID = 00 through r***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.6 *Assembly-MFM* — The presentation of the Assembly Mass Flow Meter (Assembly-MFM) object instance attributes are as indicated in Table 34. The presentation of object services shall be indicated in Table 35.

Table 34 Assembly-MFM Object Instance Network Variables

Assembly-MFM ***Class ID = 180.82, Instance ID = 00, 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface

Table 35 Assembly-MFM Object Instance Network Services

Assembly-MFM ***Profile ID = 180.82, Instance ID = 00, 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.7 *Sensor-AI-Aux* — The presentation of the Sensor Analog Input Auxiliary (Sensor-AI-Aux) object instance attributes are as indicated in Table 36. The presentation of object services shall be indicated in Table 37.

Table 36 Sensor-AI-Aux Object Instance Network Variables

Sensor-AI-Aux ***Profile ID = 180.14, Instance ID = 00 through r***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
--	No Additional Attributes Defined	--	--	

Table 37 Sensor-AI-Aux Object Instance Network Services

Sensor-AI-Aux ***Profile ID = 180.14, Instance ID = 00 through r***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.8 *Actuator-AO-MF* — The presentation of the Actuator Analog Output Mass Flow (Actuator-AO-MF) object instance attributes are as indicated in Table 38. The presentation of object services shall be indicated in Table 39.

Table 38 Actuator-AO-MF Object Instance Network Variables

Actuator-AO-MF ***Profile ID = 180.32, Instance ID = 00 through r***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Valve Type	A1	CP	Typedef unsigned byte SCPTvalveType(*)[#1]
129	Override	A2	CP	Typedef unsigned byte SCPTvalveOperatingModeE(*)[#1]

[#1]: (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 39 Actuator-AO-MF Object Instance Network Services

Actuator-AO-MF ***Profile ID = 180.32, Instance ID = 00 through r***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.9 *Controller-C* — The presentation of the Controller (C) object instance attributes are as indicated in Table 40. The presentation of Controller object services are indicated in Table 41.

Table 40 Controller -C Object Instance Network Variables

Controller-C ***Profile ID = 180.51, Instance ID = 00, 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Alarm Settling Time	CA21	CP	SCPTalrmSetT2
129	Warning Settling Time	CA24	CP	SCPTalrmSetT1

Table 41 Controller-C Object Instance Network Services

Controller-C ***Profile ID = 180.51, Instance ID = 00, 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.10 *Local Link-Lnk* — The presentation of the Local Link (Lnk) object instance attributes are as indicated in Table 42. The presentation of Local Link object services are indicated in Table 43.

Table 42 Local Link Object Instance Network Variables

Local Link (Lnk)				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
--	No Additional Attributes Defined	--	--	

Table 43 Local Link Object Instance Network Services

Local Link (Lnk)				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.11 *SISO* — The presentation of the Single Input Single Output (SISO) object instance attributes are as indicated in Table 44. The presentation of SISO object services are indicated in Table 45.

Table 44 SISO Object Instance Network Variables

SISO ***Profile ID = 180.71, Instance ID = 00 through r***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Input	NVI	A1	Typedef (*)[#1] SNVT_xxx
129	Output	NVO	A2	Typedef (*)[#1] SNVT_xxx

SISO *Profile ID = 180.71, Instance ID = 00 through r*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
130	Data Type	CP	A3	SCPTnvType

[#1]: (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 45 SISO Object Instance Network Services

SISO *Profile ID = 180.71, Instance ID = 00 through r*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.12 SISO-Setpoint – The presentation of the Single Input Single Output Setpoint (SISO-Setpoint) object instance attributes are as indicated in Table 46. The presentation of SISO-Setpoint object services are indicated in Table 47.

Table 46 SISO-Setpoint Object Instance Network Variables

SISO - Setpoint *Profile ID = 180.72, Instance ID = 00 through r*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Ramp Type	CP	A1	Typedef usint SCPTramptType (*)[#1]
129	Ramp Rate	CP	A2	Typedef (*)[#1] SNVT_xxx
130	Ratio	CP	A3	Typedef SCPTratio (*)[#1]

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 47 SISO Object Instance Network Services

SISO - Setpoint *Profile ID = 180.72, Instance ID = 00 through r*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.2.13 *Assembly-MFC* — The presentation of the Assembly Mass Flow Controller (Assembly-MFC) object instance attributes are as indicated in Table 48. The presentation of Assembly-MFC object services are indicated in Table 49.

Table 48 Assembly-MFC Object Instance Network Variables

Assembly-MFC *Profile ID = 180.83, Instance ID = 00 through r*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface

Table 49 Assembly-MFC Object Instance Network Services

Assembly-MFC *Profile ID = 180.83, Instance ID = 00 through r*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.3 *Specific Device Model For In-Situ Particular Monitor Device* — These sections detail the network mapping required to support the Specific Device Model for In-Situ Particle Monitor Devices (see reference SEMI E54.10 of §4.1). Table 50 summarizes the In-Situ Particle Monitor Device object types. Subsequent Table 51 to Table 82 details the attributes and services associated with each In-Situ Particle Monitor Device object type.

Table 50 In-Situ Particle Monitor Device Object Types

SEMI SDM Object Identifier	*Object Name*	*LONMARK Profile ID*	*LONMARK Functional Profile Name*
ISPMD1 (DM)	Device Manager	0	Node Object
ISPMD2 (SAC)	Sensor/Actuator/Controller	0	Node Object
ISPMD3	Sensor-AI-LCS	180.16	SFPTsemiSensorAILCS
ISPMD4	Sensor-AI-SLS	180.17	SFPTsemiSensorAISLS
ISPMD5	Sensor-AI-MNS	180.21	SFPTsemiSensorAIMNS
ISPMD16	Sensor-AI-Counter	180.15	SFPTsemiSensorAICounter
ISPMD17	Assembly-ISPM#1	180.84	SFPTsemiAssemblyISPM1
ISPMD18	Assembly-ISPM#2	180.85	SFPTsemiAssemblyISPM2
ISPMD19	Assembly-ISPM#3	180.86	SFPTsemiAssemblyISPM3
ISPMD20	Assembly-ISPM#4	180.87	SFPTsemiAssemblyISPM4
ISPMD21	Assembly-ISPM#5	180.88	SFPTsemiAssemblyISPM5
ISPMD22	Assembly-ISPM#6	180.89	SFPTsemiAssemblyISPM6
ISPMD23	Assembly-ISPM#7	180.90	SFPTsemiAssemblyISPM7
ISPMD24	Assembly-ISPM#8	180.91	SFPTsemiAssemblyISPM8
ISPMD25	Assembly-ISPM#9	180.92	SFPTsemiAssemblyISPM9
ISPMD64	Assembly-ISPM#48	180.93	SFPTsemiAssemblyISPM48

9.3.1 *Device Manager* — The presentation of the extended ISPM Device manager (DM) object instance attributes are as indicated in Table 51. The presentation of the extended ISPM DM object services are indicated in Table 52.

Table 51 Extended Device Manager Object Instance Network Variables

Device Manager (DM) *Profile ID = 00, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
129	Gain	CP	A33	SCPTgain
130	Filter Bandwidth	CP	A34	SCPTfilterBandwidth (*)[#1] 1
131	Tool State	NVO	A35	Typedef unsigned byte SNVT_tool_state (*)[#1]

Device Manager (DM) *Profile ID = 00, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
132	Laser Status	NVO	A36	Typedef unsigned byte SNVT_laser_status (*)[#1]
133	Flow Path	CP	A37	Typedef unsigned int SCPTflowPath (*)[#1]
134	Volume	NVO	A38	SNVT_vol_f
135	Volume Units	CP	A39	SCPTnvType
136	Leak Status	NVO	A40	Typedef unsigned byte SNVT_leak_status (*)[#1]
137	Time Stamp	NVO	A41	SNVT_time_stamp
138	LaserSwitch	NVI	A42	Typedef unsigned int SNVT_switch (*)[#1]

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 52 Extended Device Manager Object Instance Network Services

Device Manager (DM) *Profile ID = 00, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
33	Laser On	S1	LaserSwitch: state=1	
34	Laser Off	S2	LaserSwitch: state=0	

9.3.2 *Service Requests Code* — The service request is implemented as a network variable update containing a file request to an **nviSemiReq** input to the Extended Device Manager Object functional block of a device. This input is defined by the **SNVT_semi_req** type as follows:

```
typedef enum {
 struct {
   CMD_LASER_ON                = 33,
   CMD_LASER_OFF               = 34,
} semi_request_t;
typedef struct {
  unsigned int flblock_index;
  semi_request_t service_code;
  unsigned int parameter;           // Optional parameter
} SNVT_semi_req;
```

9.3.3 *Sensor Actuator Controller* — The presentation of the extended ISPM Sensor Actuator Controller (SAC) object instance attributes are as indicated in Table 53. The presentation of SAC object services are indicated in Table 54.

Table 53 Extended SAC Object Instance Network Variables

Sensor Actuator Controller (SAC) *Profile ID = 00, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
139	Number of Bins	CP	SacA65	Typedef unsigned int SCPTnumBins (*)[#1]
140	Count Mode	CP	SacA66	Typedef unsigned byte SCPTcountMode (*)[#1]
141	Duration	CP	SacA67	Typedef float SCPTduration (*)[#1]

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 54 Extended SAC Object Instance Network Services

Sensor Actuator Controller (SAC) *Profile ID = 00, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
35	Clear Counts	S33	SAC Object RQ_CLEAR_COUNTS(*)[#1]	

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

9.3.4 *Service Requests Code* — The service request is implemented as a network variable update containing a file request to an **nviSemiReq** input to the Extended SAC Object functional block of a device. This input is defined by the **SNVT_semi_req** type as follows:

```
typedef enum {
 struct {
   CMD_CLEAR_COUNTS      = 35,
} semi_request_t;
typedef struct {
  unsigned int flblock_index;
  semi_request_t service_code;
  unsigned int parameter;           // Optional parameter
} SNVT_semi_req;
```

9.3.5 *Sensor-AI-LCS* — The presentation of the Sensor Analog Input Laser Sensor (Sensor-AI-LCS) object instance attributes are as indicated in Table 55. The presentation of Sensor-AI-LCS object services are indicated in Table 56.

Table 55 Sensor-AI-LCS Object Instance Network Variables

Sensor-AI-LCS *Profile ID = 180.16, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Reading Valid	NVO	LcsA1	SNVT_switch
129	Full Scale	CP	LcsA2	SCPTmaxRange
130	Alarm Settling Time	CP	LcsA3	SCPTalrmSetT2
131	Warning Settling Time	CP	LcsA4	SCPTalrmSetT1

Table 56 Sensor-AI-LCS Object Instance Network Services

Sensor-AI-LCS *Profile ID = 180.16, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.6 *Sensor-AI-SLS* — The presentation of the Sensor Analog Input Stray Light Sensor (Sensor-AI-SLS) object instance attributes are as indicated in Table 57. The presentation of Sensor-AI-SLS object services are indicated in Table 58.

Table 57 Sensor-AI-SLS Object Instance Network Variables

Sensor-AI-SLS *Profile ID = 180.17, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Reading Valid	NVO	SlsA1	SNVT_switch
129	Full Scale	CP	SlsA2	SCPTmaxRange
130	Alarm Settling Time	CP	SlsA3	SCPTalrmSetT2
131	Warning Settling Time	CP	SlsA4	SCPTalrmSetT1

Table 58 Sensor-AI-SLS Object Instance Network Services

Sensor-AI-SLS *Profile ID = 180.17, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.7 *Sensor-AI-MNS* — The presentation of the Sensor Analog Input Medium Noise Sensor (Sensor-AI-MNS) object instance attributes are as indicated in Table 59. The presentation of Sensor-AI-MNS object services are indicated in Table 60.

Table 59 Sensor-AI-MNS Object Instance Network Variables

Sensor-AI-MNS *Profile ID = 180.21, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Reading Valid	NVO	MnsA1	SNVT_switch
129	Full Scale	CP	MnsA2	SCPTmaxRange
130	Alarm Settling Time	CP	MnsA3	SCPTalrmSetT2
131	Warning Settling Time	CP	MnsA4	SCPTalrmSetT1

Table 60 Sensor-AI-MNS Object Instance Network Services

Sensor-AI-MNS ***Profile ID = 180.21, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.8 *Sensor-AI-Counter* — The presentation of the Sensor Analog Input Counter (Sensor-AI-Counter) object instance attributes are as indicated in Table 61. The presentation of Sensor-AI-Counter object services are indicated in Table 62.

Table 61 Sensor-AI-Counter Object Instance Network Variables

Sensor-AI-Counter ***Profile ID = 180.15, Instance ID = 01 through 1024***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Reading Valid	NVO	CounterA1	SNVT_switch
129	Full Sacle	CP	CounterA2	SCPTmaxRange
130	Alarm Settling Time	CP	CounterA3	SCPTalrmSetT2
131	Warning Settling Time	CP	CounterA4	SCPTalrmSetT1
132	Upper Size	CP	CounterA5	SCPTminRange
133	Lower Size	CP	CounterA6	SCPTmaxRange

Table 62 Sensor-AI-Counter Object Instance Network Services

Sensor-AI-Counter ***Profile ID = 180.15, Instance ID = 01 through 1024***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.9 *Assembly-ISPM#1* — The presentation of the Assembly #1 In-Situ Particle Monitor (Assembly-ISPM#1) object instance attributes is as indicated in Table 63. The presentation of Assembly-ISPM#1 object services are indicated in Table 64.

Table 63 Assembly-ISPM#1 Object Instance Network Variables

Assembly-ISPM#1 ***Profile ID = 180.84, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 64 Assembly-ISPM#1 Object Instance Network Services

Assembly-ISPM#1 ***Profile ID = 180.84, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.10 *Assembly-ISPM#2* — The presentation of the Assembly #2 In-Situ Particle Monitor (Assembly-ISPM#2) object instance attributes is as indicated in Table 65. The presentation of the Assembly-ISPM#2 object services are indicated in Table 66.

Table 65 Assembly-ISPM#2 Object Instance Network Variables

Assembly-ISPM#2 ***Profile ID = 180.85, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 66 Assembly-ISPM#2 Object Instance Network Services

Assembly-ISPM#2 ***Profile ID = 180.85, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.11 *Assembly-ISPM#3* — The presentation of the Assembly #3 In-Situ Particle Monitor (Assembly-ISPM#3) object instance attributes is as indicated in Table 67. The presentation of Assembly-ISPM#3 object services are indicated in Table 68.

Table 67 Assembly-ISPM#3 Object Instance Network Variables

Assembly-ISPM#3 ***Profile ID = 180.86, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 68 Assembly-ISPM#3 Object Instance Network Services

Assembly-ISPM#3 ***Profile ID = 180.86, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.12 *Assembly-ISPM#4* — The presentation of the Assembly #4 In-Situ Particle Monitor (Assembly-ISPM#4) object instance attributes is as indicated in Table 69. The presentation of Assembly-ISPM#4 object services are indicated in Table 70.

Table 69 Assembly-ISPM#4 Object Instance Network Variables

Assembly-ISPM#4 Profile ID = 180.87, Instance ID = 01				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 70 Assembly-ISPM#4 Object Instance Network Services

Assembly-ISPM#4 Profile ID = 180.87, Instance ID = 01				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.13 *Assembly-ISPM#5* — The presentation of the Assembly #5 In-Situ Particle Monitor (Assembly-ISPM#5) object instance attributes is as indicated in Table 71. The presentation of Assembly-ISPM#5 object services are indicated in Table 72.

Table 71 Assembly-ISPM#5 Object Instance Network Variables

Assembly-ISPM#5 Profile ID = 180.88, Instance ID = 01				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 72 Assembly-ISPM#5 Object Instance Network Services

Assembly-ISPM#5 Profile ID = 180.88, Instance ID = 01				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.14 *Assembly-ISPM#6* — The presentation of the Assembly #6 In-Situ Particle Monitor (Assembly-ISPM#6) object instance attributes is as indicated in Table 73. The presentation of Assembly-ISPM#6 object services are indicated in Table 74.

Table 73 Assembly-ISPM#6 Object Instance Network Variables

Assembly-ISPM#6 Profile ID = 180.89, Instance ID = 01				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 74 Assembly-ISPM#6 Object Instance Network Services

Assembly-ISPM#6 Profile ID = 180.89, Instance ID = 01				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.15 *Assembly-ISPM#7* – The presentation of the Assembly #7 In-Situ Particle Monitor (Assembly-ISPM#7) object instance attributes is as indicated in Table 75. The presentation of Assembly_ISPM#7 object services are indicated in Table 76.

Table 75 Assembly-ISPM#7 Object Instance Network Variables

Assembly-ISPM#7 Profile ID = 180.90, Instance ID = 01				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 76 Assembly-ISPM#7 Object Instance Network Services

Assembly-ISPM#7 Profile ID = 180.90, Instance ID = 01				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.16 *Assembly-ISPM#8* — The presentation of the Assembly #8 In-Situ Particle Monitor (Assembly-ISPM#8) object instance attributes is as indicated in Table 77. The presentation of Assembly-ISPM#8 object services are indicated in Table 78.

Table 77 Assembly-ISPM#8 Object Instance Network Variables

Assembly-ISPM#8 Profile ID = 180.91, Instance ID = 01				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 78 Assembly-ISPM#8 Object Instance Network Services

Assembly-ISPM#8 *Profile ID = 180.91, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.17 *Assembly-ISPM#9* — The presentation of the Assembly #9 In-Situ Particle Monitor (Assembly-ISPM#9) object instance attributes is as indicated in Table 79. The presentation of Assembly-ISPM#9 object services are indicated in Table 80.

Table 79 Assembly-ISPM#9 Object Instance Network Variables

Assembly-ISPM#9 *Profile ID = 180.92, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 80 Assembly-ISPM#9 Object Instance Network Services

Assembly-ISPM#9 *Profiles ID = 180.92, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.3.18 *Assembly-ISPM#48* — The presentation of the Assembly #48 In-Situ Particle Monitor (Assembly-ISPM#48) object instance attributes is as indicated in Table 81. The presentation of Assembly-ISPM#48 object services are indicated in Table 82.

Table 81 Assembly-ISPM#48 Object Instance Network Variables

Assembly-ISPM#48 *Profiles ID = 180.93, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 82 Assembly-ISPM#48 Object Instance Network Services

Assembly-ISPM#48 *Profile ID = 180.93, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.4 *Specific Device Model For Endpoint Device* — These sections detail the network mapping required to support the Specific Device Model for Endpoint Devices (see reference SEMI E54.11 of §4.1). Table 83 summarizes the

Endpoint Device object types. Subsequent Table 84 to Table 97 details the attributes and services associated with each Endpoint Device object type.

Table 83 Endpoint Device Object Types

SEMI SDM Object Identifier	*Object Name*	*LONMARK Profile ID*	*LONMARK Functional Profile Name*
EPD1 (DM)	Device Manager	0	Node Object
EPD2 (SAC)	Sensor Actuator Controller	0	Node Object
EPD3	Sensor-BI-TH-EP	180.20	SFPTsemiSensorBITHEP
EPD4	Assembly-EPD#1	180.94	SFPTsemiAssemblyEPD1
EPD5	Assembly-EPD#2	180.95	SFPTsemiAssemblyEPD2
EPD6	Assembly-EPD#3	180.96	SFPTsemiAssemblyEPD3
EPD7	Assembly-EPD#4	180.97	SFPTsemiAssemblyEPD4

9.4.1 *Device Manager* — The presentation of the extended EPD Device manager (DM) object instance attributes are as indicated in Table 84. The presentation of the extended EPD DM object services are indicated in Table 85.

Table 84 Extended Device Manager Object Instance Network Variables

Device Manager (DM) ***Profile ID = 00, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
--	No Additional Attributes Defined		--	

Table 85 Extended Device Manager Object Instance Network Services

Device Manager (DM) ***Profile ID = 00, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined	--		

9.4.2 *Sensor Actuator Controller* — The presentation of the extended EPD Sensor Actuator Controller (SAC) object instance attributes are as indicated in Table 86. The presentation of extended EPD SAC object services are indicated in Table 87.

Table 86 Extended SAC Object Instance Network Variables

Sensor Actuator Controller (SAC) ***Profile ID = 00, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
129	Number of Endpoint Objects	CP	SacA65	Typedef unsigned int SCPTnumEPObj. (*)[#2]
130	Endpoint Service Request	NVI	SacA128	SNVT_ep_Service. (*)[#1]

[#1] This input is used to specify an attribute for the Endpoint services.

[#2] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 87 Extended SAC Object Instance Network Services

Sensor Actuator Controller (SAC) *Profile ID = 00, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
33	Reset Endpoint	S33	SAC Object – REQ_Reset	
34	Download Recipe	S34	LonMark file transfer – Context Specific	
35	Upload Recipe	S35		LonMark file transfer NVs and messaging interface – Context Specific.
36	Calibrate	S36	LonMark file transfer – Context Specific	

9.4.3 *Service Requests Code* — The service request is implemented as a network variable update containing a file request to an **nviSemiReq** input to the Extended SAC Object functional block of a device. This input is defined by the **SNVT_semi_req** type as follows:

```
typedef enum {
 struct {
   CMD_RESET_ENDPOINT          = 33,
   CMD_DOWNLOAD_RECIPE         = 34,
   CMD_UPLOAD_RECIPE           = 35,
   CMD_CALIBRATE               = 36,
} semi_request_t;
typedef struct {
  unsigned int flblock_index;
  semi_request_t service_code;
  unsigned int parameter;           // Optional parameter
} SNVT_semi_req;
```

9.4.4 *Sensor-BI-TH-EP* — The presentation of the Sensor Binary Input Threshold Endpoint (Sensor-BI-TH-EP) object instance attributes are as indicated in Table 88. The presentation of Sensor-BI-TH-EP object services are indicated in Table 89.

Table 88 Sensor-BI-TH-EP Object Instance Network Variables

Sensor-BI-TH-EP *Profile ID = 180.20, Instance ID = 01 through 1024*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
128	Minimum Time	CP	EpA1	SCPTalrmSetT2
129	Maximum Time	CP	EpA2	SCPTalrmSetT2
130	Target Time	CP	EpA3	SCPTalrmSetT2
131	Elapsed Time	NVO	EpA4	SNVTalrmSetT2
132	Time Stamp	NVO	EpA5	SNVT_time_stamp
133	Recipe Identifier	CP	EpA6	Typedef char() SCPTrecipeID (*)[#1]
134	Step Identifier	CP	EpA7	Typedef char () SCPTstepID (*)[#1]

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

Table 89 Sensor-BI-TH-EP Object Instance Network Services

Sensor-BI-TH-EP *Profile ID = 180.20, Instance ID = 01 through 1024*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
33	Endpoint On	S1	Node Object RQ_ENDPOINT_ON(*)[#1]	
34	Endpoint Off	S2	Node Object RQ_ENDPOINT_OFF(*)[#1]	
35	Endpoint Start	S3	Node Object RQ_ENDPOINT_START(*)[#1]	
36	Endpoint Suspend	S4	Node Object RQ_ENDPOINT_SUSPEND(*)[#1]	
37	Endpoint Resume	S5	Node Object RQ_ENDPOINT_RESUME(*)[#1]	

[#1] (*) LonMark interoperability extension. Refer to §6.8.3 for further definition.

9.4.5 *Service Requests Code* — The service request is implemented as a network variable update containing a file request to an **nviSemiReq** input to the Sensor-AI-MF Object functional block of a device. This input is defined by the **SNVT_semi_req** type as follows:

```
typedef enum {
 struct {
   CMD_ENDPOINT_ON             = 33,
   CMD_ENDPOINT_OFF            = 34,
   CMD_ENDPOINT_START          = 35,
   CMD_ENDPOINT_SUSPEND        = 36,
   CMD_ENDPOINT_RESUME         = 37,
} semi_request_t;
typedef struct {
   unsigned int flblock_index;
   semi_request_t service_code;
   unsigned int parameter;           // Optional parameter
} SNVT_semi_req;
```

9.4.6 *Assembly-EPD#1* — The presentation of the Assembly #1 Endpoint (Assembly-EPD#1) object instance attributes is as indicated in Table 90. The presentation of Assembly-EPD#1 object services are indicated in Table 91.

Table 90 Assembly-EPD#1 Object Instance Network Variables

Assembly-EPD#1 *Profile ID = 180.94, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 91 Assembly-EPD#1 Object Instance Network Services

Assembly-EPD#1 ***Profile ID = 180.94, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.4.7 *Assembly-EPD#2* — The presentation of the Assembly #2 Endpoint (Assembly-EPD#2) object instance attributes is as indicated in Table 92. The presentation of Assembly-EPD#2 object services are indicated in Table 93.

Table 92 Assembly-EPD#2 Object Instance Network Variables

Assembly-EPD#2 ***Profile ID = 180.95, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 93 Assembly-EPD#2 Object Instance Network Services

Assembly-EPD#2 ***Profile ID = 180.95, Instance ID = 01***				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.4.8 *Assembly-EPD#3* — The presentation of the Assembly #3 Endpoint (Assembly-EPD#3) object instance attributes is as indicated in Table 94. The presentation of Assembly-EPD#3 object services are indicated in Table 95.

Table 94 Assembly-EPD#3 Object Instance Network Variables

Assembly-EPD#3 ***Profile ID = 180.96, Instance ID = 01***				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 95 Assembly-EPD#3 Object Instance Network Services

Assembly-EPD#3 ***Profile ID = 180.96, Instance ID = 01***				
Service Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

9.4.9 *Assembly-EPD#4* — The presentation of the Assembly #4 Endpoint (Assembly-EPD#4) object instance attributes is as indicated in Table 96. The presentation of Assembly-EPD#4 object services are indicated in Table 97.

Table 96 Assembly-EPD#4 Object Instance Network Variables

Assembly-EPD#4 *Profile ID = 180.97, Instance ID = 01*				
Sequence Number	*Name*	*Storage Class*	*SDM Tag*	*Standard NV or CP Data Type*
1	Data	File	A1	LonMark file transfer NVs and messaging interface.

Table 97 Assembly-EPD#4 Object Instance Network Services

Assembly-EPD#4 *Profile ID = 180.97, Instance ID = 01*				
Service Request Code	*Service Name*	*SDM Tag*	*Request Parameters*	*Result Parameters*
--	No Additional Services Defined			

SEMI E54.17-0705
SPECIFICATION OF SENSOR/ACTUATOR NETWORK FOR A-LINK

This specification was technically approved by the global Information & Control Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on May 20, 2005. It was available at www.semi.org in June 2005 and on CD-ROM in July 2005.

1 Purpose

1.1 This document defines a communication specification based on the A-LINK protocol to enable communications between intelligent devices on a sensor/actuator network (SAN) that operate according to SEMI E54 device models on semiconductor manufacturing equipment.

1.2 This document gives interoperability with SEMI E54 common/specific device model based Sensor/Actuator devices.

2 Scope

2.1 This document specifies how Sensor / Actuator / Controller devices interoperate on the network specific for the A-LINK Public specification referenced in the §4, as a part of equipment's control system.

2.2 This document defines relation with main part of SEMI E54 including Common Device Model (CDM) and existing Specific Device Models (SDMs). This document is to be used with the CDM and one or more SDMs, as well as the A-LINK Public Specification.

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 Limitations

3.1 This document specifies a semiconductor equipment SAN based only A-LINK and is a companion document to the A-LINK specification; therefore a complete specification of this standard includes the A-LINK specifications. There are other semiconductor SAN communication options. The specifications for these options are not included here.

3.2 This standard specifies enhancements that provide additional capability over and above those currently required by A-LINK. In order to avoid document inconsistency problem, information in the A-LINK specification that relates to this standard is not repeated in this document. This document is limited to describing enhancements or limitations to the A-LINK specification that are imposed by this standard.

3.3 A complete specification of the conformance testing procedure shall include the A-LINK protocol conformance testing specification. Conformance testing shall also include enhancements and limitations to the A-LINK specification required by this standard.

4 Referenced Standards and Documents

4.1 *SEMI Standards*

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Specification for Sensor/Actuator Network Specific Device Model for Mass Flow Device

SEMI E54.10 — Specification for Sensor/Actuator Network Specific Device Model for an In-situ Particle Monitor Device

SEMI E54.11 — Specific Device Model for Endpoint Devices

4.2 *OSI Standard*

ISO 7498 OSI — Basic Reference Model for Open Systems Interconnection

4.3 *A-LINK Documents*[1]

A-LINK Public Specification ver. 1.2 — Basic Reference Model for Open Systems Interconnection

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *AUF* — A-LINK User Forum

5.1.2 *CDM* — Common Device Model

5.1.3 *NCS* — Network Communication Standard

5.1.4 *OSI* — Basic Reference Model for Open Systems Interconnection (ISO 7498)

5.1.5 *PHY* — Physical Layer

5.1.6 *SAN* — Sensor/Actuator Network

5.1.7 *SDM* — Specific Device Model

5.1.8 *UI* — User Interface

5.2 This section includes the terms defined in SEMI E54.1 Sensor/Actuator Network Common Device Model.

5.2.1 *Attribute*

5.2.2 *Behavior*

5.2.3 *Byte*

5.2.4 *Class*

5.2.5 *Common Device Model*

5.2.6 *Device*

5.2.7 *Device Manager (DM) Object*

5.2.8 *Device Model*

5.2.9 *Instance*

5.2.10 *Network Communication Standard*

5.2.11 *Object*

5.2.12 *Sensor, Actuator and Controller (SAC) Object*

5.2.13 *Service*

5.2.14 *Specific Device Model*

5.2.15 *state diagram*

5.3 *Terminology Defined in A-LINK*

5.3.1 *AUF* — A-LINK User Forum.[1] A kind of consortium for A-LINK users to recommend improvement to A-LINK trade organization[2], share A-LINK applications, provide A-LINK compliance test and etc.

5.3.2 *AN MS* — a station that accesses and exchanges data by polling its assigned Networked-Slave stations.

5.3.3 *AN SS* — a station that is managed and accessed by AN MS.

1 http://www.a-linkuf.com

2 http://www.algosystem.co.jp

5.3.4 *Device Profile* — a kind of electric table to provide the characteristic features of a device including configuration and capabilities.

5.3.5 *Service Access Element* — an addressable location in a device for the directing of service requests.

5.3.6 *Data Exchange* — a capability to communicate such data for a device as raw/processed sensed data and variable settings.

5.3.7 *Directive* — a capability to instruct such basic functions to a device as reset or abort.

5.3.8 *DP Facility* — a capability to manage diagnostic matters.

5.3.9 *Unite Inventory* — a capability to realize and communicate the Device Profile.

5.3.10 *Unite States* — a capability to communicate state of a device.

6 Communication Protocol High Level Structure

6.1 In a typical remote I/O configuration, single master architectures are used to optimize response times. In complicated applications, multi-master architectures are also possible. A-LINK uses the polling principle for communication.

6.1.1 Message transfer is organized in cycles. A message cycle mainly consists of a request-frame followed by a corresponding acknowledge/response-frame of the addressed station. An exception to this is the global-control function for synchronization and coordination of several remote I/O stations.

6.1.2 A brief description of the A-LINK protocol as it relates to the ISO 7498 OSI model follows in the sections below. For protocol efficiency, A-LINK does not define layers 3 to 7. However, since the OSI model specifies Layer 7 as the interface between the Application Process and the communication stack, it is appropriate to discuss several aspects of the A-LINK standard at this level.

NOTE 1: The information contained in this section is for reference only. It in no way represents specifications for A-LINK. See related documentation for these specifications.

6.2 *Physical Layer* — Layer 1

6.2.1 The bottom Physical Layer is established by RS-485. See the A-LINK standard for more information for detail.

6.3 *Data Link Layer* — Layer 2

6.3.1 *Data Transfer*

6.3.1.1 The Data Link Layer provides the functions for sending and receiving data over the network. Data Elements are packaged, delivered, and checked. Acknowledgements, responses, retries, and timeouts are used to guard against Line Protocol Errors (e.g., frame, overrun, and parity) and Transmission Protocol Errors (e.g., start and end delimiters, frame check, frame length, and response times).

6.4 *Network Layer* — Layer 3

6.4.1 There is no distinct network layer.

6.5 *Transport Layer* — Layer 4

6.5.1 There is no distinct transport layer. Such functions as disassembling of message into transport unit and reassembling for transportation are implemented in the Application Layer.

6.6 *Session Layer* — Layer 5

6.6.1 There is no distinct session layer.

6.7 *Presentation Layer* — Layer 6

6.7.1 There is no distinct presentation layer. Such functions as data representation conformance are implemented in the Application Layer.

6.8 *Application Layer* — Layer 7

6.8.1 The Device shall comply with A-LINK application layer specification for defining and addressing objects including functions deferred in lower layers as described above.

6.8.2 *User Interface (UI)*

6.8.2.1 The UI provides the user with access to functionality of the A-LINK protocol as a part of application layer.

7 Required Object Types

7.1 This section describes a general mapping of the SEMI SAN Object Model to the A-LINK environment. Component definitions are clarified and the mapping of Attributes, Services, and Behaviors are specified.

7.2 *Object Model*

7.2.1 The Object Model defined in the CDM is represented in the A-LINK NCS. Especially the DM, SAC, and abstract or fundamental application objects are mapped.

7.2.2 The Application Objects associated with the SDM standards are mapped in A-LINK User Forum Public Specification documents. §9 specifies the mapping of SDM Objects in A-LINK.

7.3 *Component Mapping Summary*

7.3.1 Table 1 provides a summary of the components of the CDM object model as they relate to the components of A-LINK.

7.4 *Objects*

7.4.1 The required objects of the CDM are identified here. Additional objects that are contained in the SDM are given identifiers in the Device Profile. §9 specifies additional mapping information.

7.4.2 Table 1 lists the Object Identifiers specified for use in protocol messages.

Table 1 Object Identifiers

Object Name	*A-LINK Class/Object ID*	*CDM Object ID Tag*	*CDM Attribute/Service ID Tag Prefix*
Device Manager	1	DmI0	Dm
SAC	2	SacI0	Sac
Assembly	3	AsmIn	Asm
Local Link	4	LnkIn	Lnk
Sensor–AI	9	SenIn	Sai
Sensor–EI	10	SenIn	Sei
Sensor–BI	11	SenIn	Sbi
Sensor–BI–TH	12	SenIn	Sbith
Actuator–AO	17	ActIn	Aao
Actuator–EO	18	ActIn	Aeo
Actuator–BO	19	ActIn	Abo
Controller	24	CntIn	C
Application Objects	>31	-	-

7.5 *Attributes*

7.5.1 All attributes are accessible via Get_Attribute and Set_Attribute services defined in the sections below. Attributes are also accessible via different A-LINK peculiar instructions which are additionally mapped in this document based on attribute type.

7.5.1.1 The attributes of the DM object consists of distinctive information, states, and setups. Distinctive information attributes are retrieved with the Unite Inventory. Inquiring state attributes are provided by getting status data. Setup attributes are written by setting configuration data.

7.5.1.2 The attributes of Application objects are divided into two types: Input/Output, settings, status and Configuration. Input/Output attributes are acquired from or given through Data Exchange capability of A-LINK. Configurable attributes can be retrieved with Unite Inventory capability.

7.5.1.3 See Table 2 for a list of DM attributes and their related alternative access instructions.

7.5.1.4 *A-LINK Settings*

7.5.1.4.1 Attributes related to general settings are accessed through Unite Inventory communication.

7.5.1.4.2 Structure of the Unite Inventory for a given SDM is beyond the scope of this document. The A-LINK Trade Organization is responsible for the management of this information.

7.5.1.5 *A-LINK States and Diagnostics*

7.5.1.5.1 Attributes related to device state are categorized and handled as Unite State information. For example two attributes of the DM object listed in Table 2 that are identified with an alternative access instruction of Unite States are mapped into the A-LINK Unite States as specified in this section. See the A-LINK standard for a description of the Unite States.

7.5.1.5.2 These two attributes are mapped into the Unite State data structure. Additional diagnostic data may be included with DP Facility as specified by A-LINK.

7.5.2 *A-LINK I/O Data Exchange*

7.5.2.1 Input/Output attributes of the Application objects are communicated using the I/O Data Exchange instruction of A-LINK. This instruction is described in the A-LINK standard. A list of which attributes are accessible with this instruction is included in the A-LINK Device Profile for a given device type.

7.5.3 *A-LINK Device Configuration*

7.5.3.1 Configuration attributes of the DM object and Application objects are communicated using the Unite Inventory Service of A-LINK. This service is described in the A-LINK standard. A list of which Application object attributes are accessible with this instruction is included in the A-LINK Device Profile for a given device type.

7.5.3.2 *Attribute Identifiers*

7.5.3.2.1 Every class object specified in the CDM uses attribute identifier tags to identify its attributes. The tags are formed with alphabetic part of its object identifier tag, a character ‘A’ for impressing attribute and a numerical identifier of the attribute. The numerical identifier is assigned to the Attribute ID used in the A-LINK NCS for classes DM, SAC, Assembly, Local Link, AE, S, A and C, as described in following tables: i.e. from Table 2 through Table 8.

7.5.3.2.2 Except Sensor-Binary Input Threshold (SBITH) class, the Attribute ID used in the A-LINK for derived classes of Sensor and Actuator classes is assigned with the same manner as above.

7.5.3.2.3 For Sensor-Binary Input Threshold (SBITH) class, the Attribute ID used in the A-LINK starts from 128 to prevent duplication with the numerical identifiers in CDM assigned and reserved to is parent class, SBI.

7.5.3.2.4 The A-LINK attribute ID is used to identify attributes for access via A-LINK message requests, which are explained in later sections.

7.5.4 *A-LINK Attribute Mappings for CDM*

7.5.4.1 *Device Manager (DM) Object* — The DM object is the device component responsible for managing and consolidating the device operation as defined in the CDM standard. Attribute values including DM state are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 2.

Table 2 DM Object Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
DmA1	1	Device Type	Unite Inventory
DmA2	2	Standard Revision Level	Unite Inventory

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
DmA3	3	Device Manufacturer Identifier	Unite Inventory
DmA4	4	Manufacturer Model Number	Unite Inventory
DmA5	5	Software or Firmware Revision Level	Unite Inventory
DmA6	6	Hardware Revision Level	Unite Inventory
DmA7	7	Serial Number	Unite Inventory
DmA8	8	Device Configuration	Unite Inventory
DmA9	9	Device Status	Unite States
DmA10	10	Reporting Mode	Unite Inventory
DmA11	11	Exception Status Report Interval	Unite Inventory
DmA12	12	Exception Status	Unite States
DmA13	13	Exception Detail Alarm	Unite Inventory
DmA14	14	Exception Detail Warning	Unite Inventory
DmA15	15	Visual Indicator	Unite Inventory
DmA16	16	Alarm Enable	Unite Inventory
DmA17	17	Warning Enable	Unite Inventory
DmA18	18	Exception Detail Type	Unite Inventory
DmA19	19	Exception Detail Alarm Queue	Unite Inventory
DmA20	20	Exception Detail Warning Queue	Unite Inventory
DmA21	21	Date and Time	Unite Inventory
DmA22	22	Date and Time Type	Unite Inventory

7.5.4.2 *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 as defined in the CDM standard. Attribute values including SAC state are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 3.

Table 3 SAC Object Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
SacA1	1	Last Calibration Date	DP Facility
SacA2	2	Next Calibration Date	DP Facility
SacA3	3	Expiration Timer	DP Facility
SacA4	4	Expiration Warning Enable	DP Facility
SacA5	5	Run Hours	DP Facility

7.5.4.3 *Assembly Object (Asm)* — The Assembly (Asm) object instances may be used to provide for grouping more than one attribute from one or more object instances as defined in the CDM standard. Attribute values are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 4.

Table 4 Assembly Object Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
AsmA1	1	Data	Data Exchange

7.5.4.4 *Local Link Object (Lnk)* — The Local Link (Lnk) object instances may be used to ‘link” an attribute of one object instance to an attribute of another object instance as defined in the CDM standard. Attribute values are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 4.

Table 5 Local Link Object Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
LnkA1	1	Source Object Class	Unite Inventory
LnkA2	2	Source Object Instance	Unite Inventory
LnkA3	3	Source Object Attribute	Unite Inventory
LnkA4	4	Destination Object Class	Unite Inventory
LnkA5	5	Destination Object Instance	Unite Inventory
LnkA6	6	Destination Object Attribute	Unite Inventory
LnkA7	7	Commit	Unite Inventory

7.5.4.5 *Active Element (AE) Class* — The AE class is an abstract class generic to any device component as defined in the CDM standard. Attribute values including AE state are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in

7.5.4.6 These attributes are inherited by any sensor, actuator and controller objects, and such further derived objects as Sensor-AI and Actuator-BO objects.

Table 6 AE Class Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
nA1	1	Name	Unite Inventory
nA2	2	Status	Unite States
nA3	3	AlarmEnable	Unite Inventory
nA4	4	WarningEnable	Unite Inventory

[#1] Prefix n in CDM attribute ID represents one of Sai, Sei, Sbi, Sbith, Aao, Aeo, Abo and C.

7.5.4.7 *Sensor (S) Class* — The S class is an abstract class generic to any sensors on device component as defined in the CDM standard. Attribute values are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 7. These attributes are inherited by any sensor objects as Sensor-AI object. They also inherit AE class attributes. Attribute identifiers on A-LINK for such direct and indirect inheritor classes of S class as Sensor-AI are assigned as ¶7.5.3.2 and Data Exchange communication could give alternative access for the attributes.

Table 7 S Class Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
nA16	16	Value	Data Exchange
nA17	17	ReportInhibitTimer	Data Exchange
nA18	18	EnableReportRate	Data Exchange
nA19	19	ReportRate	Data Exchange

[#1] Prefix n in CDM attribute ID represents one of Sai, Sei, Sbi or Sbith.

7.5.4.8 *Actuator (A) Class* — The A class is an abstract class generic to any actuators on device component as defined in the CDM standard. Attribute values are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 8. These attributes are inherited by any sensor objects as Actuator-AO object. They also inherit AE class attributes. Attribute identifiers on A-LINK for inheritor classes Sensor-AO are assigned as ¶7.5.3.2 and Data Exchange communication could give alternative access for the attributes.

Table 8 A Class Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
nA16	16	Setting	Data Exchange
nA17	17	SafeState	Data Exchange
nA18	18	WatchRate	Data Exchange
nA19	19	WatchDog	Data Exchange

[#1] Prefix n in CDM attribute ID represents one of Aao, Aeo or Abo.

7.5.4.9 *Controller (C) Object* — The C object contains structure and behavior common to all controller element instances on device component as defined in the CDM standard. Attribute values are also as same as defined in the CDM standard. The presentation of object instance attributes to the A-LINK network shall be as indicated in Table 9. It inherits AE class attributes.

Table 9 C Class Attribute Identifiers for A-LINK

Attribute ID		*Attribute Name*	*Alternative Access*
CDM	*A-LINK*		
CA16	16	Setpoint	Data Exchange
CA17	17	ProcessVariable	Data Exchange
CA18	18	ControlVariable	Data Exchange
CA19	19	DataType	Data Exchange
CA20	20	DataUnits	Data Exchange
CA21	21	AlarmSettleTime	Data Exchange
CA22	22	AlarmErrorBand	Data Exchange
CA24	24	WarningSettleTime	Data Exchange
CA25	25	WarningErrorBand	Data Exchange

7.6 *Services*

7.6.1 A-LINK specifies standard mechanisms for the communication of data over the network. These mechanisms are used to communicate attributes specified in the device on the A-LINK. Also they are used to request instructions specific for component in the device on the A-LINK.

7.6.2 *Service Requests and Response*

7.6.2.1 The A-LINK attribute communications and instructions require specific definitions. They are described in the following sections.

7.6.2.1.1 *Service Request and Response Protocol*

7.6.2.1.1.1 All service request messages, except Get_Attribute and Set_Attribute, are sent to a device using Directive service functions of A-LINK. The responses to these message requests are specified by A-LINK.

7.6.2.1.1.2 The Service Request message is formatted, as defined by A-LINK, with the following information:

Location := Object ID

7.6.2.1.1.3 This service request message has no other information requested with the message. Response of the request has following information:

Location := Object ID

Status := Response Information

NOTE 2: The Location information for the response message is optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on class and/or device. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.1.2 *Set_Attribute Protocol*

7.6.2.1.2.1 The Service Request message for the Set-Attribute is sent to a device using writing capability for following service functions of A-LINK: Unite Inventory service, Data Exchange service or DP Facility service. Using service function is dependent on attribute and object class as well. The dependency is given by Object Service Identifier tables in following subsections of ¶7.6.

7.6.2.1.2.2 The service request message with Unite Inventory service function of A-LINK has a couple of information items. Information items for the request and its response are given below respectively:

(Request)
Location := Object ID :optional
Data Name ID := Attribute ID
Data Value := Attribute Value

(Response)
Location := Object ID ;optional
Data Name ID := Attribute ID
Data Value := Attribute Value
Status := Response Information

NOTE 3: The items of Data Name ID and Data Value for the response message are optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on attribute and/or device. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.1.2.3 The service request message with Data Exchange service function of A-LINK has three information items. Information items for the request and its response are given below respectively:

(Request)
Location := Object ID
Data Name ID := Attribute ID
Data Value := Attribute Value

(Response)
Location := Object ID
Data Name ID := Attribute ID
Data Value := Attribute Value
Status := Response Information

NOTE 4: The items of Location, Data Name ID and Data Value for the response message are optional (conditional) information. Representation of the Status information is implementation specific. The Response Iinformation is dependent on attribute and/or class.

7.6.2.1.2.4 The service request message with DP Facility service function of A-LINK has a couple of information items. Information items for the request and its response are given below respectively:

(Request)
Information Name ID := Attribute ID
Information Value := Attribute Value

(Response)
Information Name ID := Attribute ID
Information Value := Attribute Value
Status := Response Information

NOTE 5: The items of Information Name ID and Information Value for the response message are optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on attribute and/or device. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.1.3 *Get_Attribute Protocol*

7.6.2.1.3.1 The Service Request message for the Get-Attribute is sent to a device using reading capability for following service functions of A-LINK: Unite Inventory service, Unite State service, Data Exchange service or DP Facility service. Using service function is dependent on attribute and object class as well. The dependency is given by Object Service Identifier tables in following subsections of ¶7.6.

7.6.2.1.3.2 The service request message with Unite Inventory service function of A-LINK has an information item. Information items for the request and its response are given below respectively:

(Request)
Location := Object ID :optional
Data Name ID := Attribute ID

(Response)
Location := Object ID :optional
Data Name ID := Attribute ID
Data Value := Attribute Value
Status := Response Information

NOTE 6: The item of Data Name ID for the response message is optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on attribute and/or device. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.1.3.3 The service request message with Unite State service function of A-LINK has an information item. Information items for the request and its response are given below respectively:

(Request)
Top State Name ID := Object ID

(Response)
Top State Name ID := Object ID
Data Value := Attribute Value
Status := Response Information

NOTE 7: The item of Top State Name ID for the response message is optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on attribute and/or device. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.1.3.4 The service request message with Data Exchange service function of A-LINK has a couple of information items. Information items for the request and its response are given below respectively:

(Request)
Location := Object ID
Data Name ID := Attribute ID

(Response)
Location := Object ID
Data Name ID := Attribute ID
Data Value := Attribute Value
Status := Response Information

NOTE 8: The items of Location and Data Name ID for the response message are optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on attribute and/or class. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.1.3.5 The service request message with DP Facility service function of A-LINK has an information item. Information items for the request and its response are given below respectively:

(Request)
Information Name ID := Attribute ID

(Response)
Information Name ID := Attribute ID

Information Value := Attribute Value
Status := Response Information

NOTE 9: The item of Information Name ID for the response message is optional (conditional) information. Representation of the Status information is implementation specific. The Response Information is dependent on attribute and/or device. However if the first item of the response information, i.e. Response Code, is zero, it always means successful. Additional data may follow.

7.6.2.2 *Service Identifiers* — The A-LINK services are mapped for CDM as following tables.

7.6.2.2.1 *Device Manager (DM) Object* — Services of the DM object are mapped as shown in Table 10.

Table 10 DM Object Service Identifiers for A-LINK

Service ID		*Service Name*	*Alternative Instruction*
CDM	*A-LINK*		
DmS1	1	Reset	Directive
DmS2	2	Abort	Directive
DmS3	3	Recover	Directive
DmS4	4	GetAttribute	Unite Inventory / Unite States
DmS5	5	SetAttribute	Unite Inventory
DmS6	6	Execute	Directive
DmS7	7	PerformDiagnostics	Directive
DmS8	8	PublishAttribute	Directive
DmS9	9	Lock	Directive
DmS10	10	Unlock	Directive
DmS11	11	Get Exception Queue	Directive
DmA12	12	Clear Exception Queue	Directive

7.6.2.2.2 *Sensor Actuator Controller (SAC) Object* — Services of the SAC object are mapped as shown in Table 11.

Table 11 SAC Object Service Identifiers for A-LINK

Service ID		*Service Name*	*Alternative Instruction*
CDM	*A-LINK*		
SacS1	1	Reset	Directive
SacS2	2	Abort	Directive
SacS3	3	Recover	Directive
SacS4	4	GetAttribute	Unite Inventory / Unite States
SacS5	5	SetAttribute	Unite Inventory
SacS6	6	Operate	Directive
SacS7	7	Restore Default	Directive
SacS8	8	Publish Attribute	Directive

7.6.2.2.3 *Active Element (AE) Object* — Services of the AE object are mapped as shown in Table 12.

7.6.2.2.3.1 Because AE class is inherited to any sensor, actuator or controller objects such as Sensor-AI and Actuator-BO, these services are equipped by such objects.

Table 12 AE Object Service Identifiers for A-LINK

Service ID		*Service Name*	*Alternative Instruction*
CDM	*A-LINK*		
nS1	1	Reset	Directive
nS2	2	Abort	Directive
nS3	3	Recover	Directive
nS4	4	Operate	Directive
nS5	5	GetAttribute	Data Exchange

Service ID		Service Name	Alternative Instruction
CDM	A-LINK		
nS6	6	SetAttribute	Data Exchange
nS7	7	Restore Default	Directive

[#1] Prefix n in CDM service ID represents one of Sai, Sei, Sbi, Sbith, Aao, Aeo, Abo and C.

8 Protocol Compliance

8.1 A method to testing protocol compliance is required to verify implementation conformance to this standard. A-LINK User Forum (AUF) has established a qualified certification mechanism of conformance testing and interoperability testing. The first laboratory is constituted in Japan. A-LINK conformance test information can be found at http://www.a-linkuf.com.

9 Specific Device Model Mappings

9.1 Every type of device must have an identifier number. Vendors must apply for an identifier number from the A-LINK User Organization for every Device Type.

9.1.1 The Device Profile must specify the identifiers for Objects, Attributes and Services for CDM and SDM components, including data formats and bit mappings for specified parameters, as represented in this document.

9.1.2 The following sections specify mappings for Sensor Actuator Network Specific Device Models.

9.2 *Mass Flow Device*

9.2.1 *MFD Device* — Reference SEMI E54.3 for a complete specification of the SDM for Mass Flow Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.2.2 *Objects* — Consistent with SEMI E54.3 and ¶7.4 above, the DM and SAC objects are identified as Object

9.2.2.1 *Mapping* — Table 13 shows the mapping of the SDM Object specified in SEMI E54.3.

Table 13 MFD Object Identifiers

SDM Object Name	SDM Object ID	ID	Instance
Device Manager (DM)	MFD1	1	1
Sensor Actuator Contriller (SAC)	MFD2	2	1
Sensor-AI-MF	MFD3	34	1 or supplier specific
Sensor-AI-AT	MFD4	35	0 / 0 or supplier specific
Assembly-MFM	MFD5	32	0 or 1
Sensor-AI-Aux	MFD6	36	0 / 0 or supplier specific
Actuator-AO-MF	MFD7	37	0 / 1 or supplier specific
Controller	MFD8	24	0 / 1
Local Link	MFD9	4	0 / 2 or supplier specific
SISO	MFD10	38	0 / 0 or supplier specific
SISO-Setpoint	MFD11	39	0 / 0 or supplier specific
Assembly-MFC	MFD12	33	0 / 0 or supplier specific

[#1] / = numbers for MFM / MFC

9.2.3 *Attributes and Services*

9.2.3.1 *Attributes* — The mapping of Attribute tags and Identifiers is defined in ¶7.5.3.2 for the CDM. Basically the same definitions are applied for attributes in the Mass Flow Device SDM.

9.2.3.2 *Services* — The mapping of Service tags and Identifiers is defined in ¶7.6.2.2 for the CDM. Basically the same definitions are applied for attributes in the Mass Flow Device SDM.

9.2.3.3 *MFD Specific Attributes and Services* — Some objects appear in MFD SDM use additional attributes and/or additional services.

9.2.3.3.1 *Sensor AI-MF Object* — Additional attributes and services are mapped as following tables.

Table 14 Sensor AI-MF Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
A1	192	Flow Totalizer
A2	193	Flow Hours
A5	196	Zero Offset Mode
A6	197	Zeroing Status
A7	198	Autorange Status

Table 15 Sensor AI-MF Object Service Identifiers

Service ID		*Service Name*
SDM	*A-LINK*	
S1	16	Perform Zero Offset
S2	17	Query Supported Gas type
S3	18	Select Programmed Gas Type
S4	19	Insert Gas Type
S5	20	Delete Gas Type
S6	21	Get Gas Calibration Data Value
S7	22	Set Gas Calibration Data Value
S8	23	Autorange

9.2.3.3.2 *Actuator AO-MF Object* — Additional attributes are mapped as following table.

Table 16 Actuator AO-MF Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
A1	192	Valve Type
A2	193	Override

9.2.3.3.3 *Controller Object* — This object has no additional attribute and/or service but form of following a couple of attributes is restricted as Real.

9.2.3.3.4 *SISO Object* — This object is specific for MFD SDM. Attributes of the object are mapped as following table.

Table 17 SISO Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
A1	192	Input
A2	193	Output
A3	194	Data Type

9.2.3.3.5 *SISO Setpoint Object* — This object is specific for MFD SDM and inherits SISO object. Attributes of the derived object are mapped as following table.

Table 18 SISO Setpoint Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
A33	224	Ramp type
A34	225	Ramp Rate
A35	226	Ratio

9.3 *In-Situ Particle Monitor*

9.3.1 *ISPM Device* — Reference SEMI E54.10 for a complete specification of the SDM for In-Situ Particle Monitor Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.3.2 *Objects* — Consistent with SEMI E54.10 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.3.2.1 *Mapping* — Table 19 shows the mapping of the SDM Object Instances specified in SEMI E54.10 (Instance numbers are listed under heading Inst. in the table) and the A-LINK Object ID (listed under ID in the table).

Table 19 ISPM Object Identifiers

SDM Object Name	*SDM Object ID*	*ID*	*Instance*
Device Manager (DM)	ISPMD1	1	1
Sensor Actuator Controller (SAC)	ISPMD2	2	1
Sensor-AI-LCS	ISPMD3	32	0 or 1
Sensor-AI-SLS	ISPMD4	33	0 or 1
Sensor-AI-MNS	ISPMD5	34	0 or 1
Sensor-AI-Counter	ISPMD16	45	0 or n[*]
Assembly-ISPM#1	ISPMD17	35	1
Assembly-ISPM#2	ISPMD18	36	0 or 1
Assembly-ISPM#3	ISPMD19	37	0 or 1
Assembly-ISPM#4	ISPMD20	38	0 or 1
Assembly-ISPM#5	ISPMD21	39	0 or 1
Assembly-ISPM#6	ISPMD22	40	1
Assembly-ISPM#7	ISPMD23	41	0 or 1
Assembly-ISPM#8	ISPMD24	42	0 or 1
Assembly-ISPM#9	ISPMD25	43	0 or 1
Assembly-ISPM#48	ISPMD64	44	0 or 1

[#1] Minimum number of the n is one and possible maximum number is 1024, dependent on supplier.

9.3.2.2 Additional objects may be defined by the manufacturer in the Device Profile for a given device.

9.3.3 *Attributes and Services*

9.3.3.1 *Attributes* — The mapping of Attribute tags and Identifiers is defined in ¶7.5.3.2 for the CDM. Basically the same definitions are applied for attributes in the In-Situ Particle Monitor SDM.

9.3.3.2 *Services* — The mapping of Service tags and Identifiers is defined in ¶7.6.2.2 for the CDM. Basically the same definitions are applied for attributes in the In-Situ Particle Monitor SDM.

9.3.3.3 *ISPM Specific Attributes and Services* — Some objects appear in ISPM SDM use additional attributes and/or additional services.

9.3.3.3.1 *DM Object* — Additional attributes and services are mapped as following tables.

Table 20 DM Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
A33	33	Gain
A34	34	Filter Bandwidth
A35	35	Tool State
A36	36	Laser Status
A37	37	Flow Path
A38	38	Volume
A39	39	Volume Units
A40	40	Leak Status
A41	41	Time Stamp

Table 21 DM Object Service Identifiers

Service ID		*Service Name*
SDM	*A-LINK*	
S1	33	Laser On
S2	34	Laser Off

9.3.3.3.2 *SAC Object* — Additional attributes and services are mapped as following tables.

Table 22 SAC Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
SacA65	33	Number of Bins
SacA66	34	Count Mode
SacA67	35	Duration

Table 23 SAC Object Service Identifiers

Service ID		*Service Name*
SDM	*A-LINK*	
S33	33	Clear Counts

9.3.3.3.3 *Sensor AI-LCS Object* — Additional attributes are mapped as following table.

Table 24 Sensor AI-LCS Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
LcsA1	192	Reading Valid
LcsA2	193	Full Scale
LcsA3	194	Alarm Setting Time
LcsA4	195	Warning Setting Time

9.3.3.3.4 *Sensor AI-SLS Object* — Additional attributes are mapped as following table.

Table 25 Sensor AI-SLS Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
SlsA1	192	Reading Valid
SlsA2	193	Full Scale
SlsA3	194	Alarm Setting Time
SlsA4	195	Warning Setting Time

9.3.3.3.5 *Sensor AI-MNS Object* — Additional attributes are mapped as following table.

Table 26 Sensor AI-MNS Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
MnsA1	192	Reading Valid
MnsA2	193	Full Scale
MnsA3	194	Alarm Setting Time
MnsA4	195	Warning Setting Time

9.3.3.3.6 *Sensor AI-Counter Object* — Additional attributes are mapped as following table.

Table 27 Sensor AI-Counter Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
CounterA1	192	Reading Valid
CounterA2	193	Full Scale
CounterA3	194	Alarm Setting Time
CounterA4	195	Warning Setting Time
CounterA5	196	Upper Size
CounterA6	197	Lower Size

9.4 *Endpoint Detector*

9.4.1 *EPD Device* — Reference SEMI E54.11 for a complete specification of the SDM for Endpoint Devices. Accordingly, the following mapping rules apply to the identification tags for the Objects, Attributes and Services of this model.

9.4.2 *Objects* — Consistent with SEMI E54.11 and ¶7.4 above, the DM and SAC objects are identified as Object 1 and Object 2, respectively.

9.4.2.1 Table 28 shows the mapping of the SDM Object Instances specified in SEMI E54.11 (Instance numbers are listed under heading Inst. in the table) and the A-LINK Object ID (listed under ID in the table).

Table 28 EPD Object Identifiers

SDM Object Name	*SDM Object ID*	*ID*	*Instance*
Device Manager (DM)	EPD1	1	1
Sensor Actuator Controller (SAC)	EPD2	2	1
Sensor-BI-TH-EP	EPD3	36	0 or n*
Assembly-EPD#1	EPD4	32	1
Assembly-EPD#2	EPD5	33	0 or 1
Assembly-EPD#3	EPD6	34	0 or 1
Assembly-EPD#4	EPD7	35	0 or 1

[#1] Minimum number of the n is one and possible maximum number is 1024.

9.4.2.2 Additional objects may be defined by the manufacturer in the Device Profile for a given device.

9.4.3 *Attributes and Services*

9.4.3.1 *Attributes* — The mapping of Attribute tags and Identifiers is defined in ¶7.5.3.2 for the CDM. Basically the same definitions are applied for attributes in the Endpoint Detector SDM.

9.4.3.2 *Services* — The mapping of Service tags and Identifiers is defined in ¶7.6.2.2 for the CDM. Basically the same definitions are applied for attributes in the Endpoint Detector SDM.

9.4.3.3 *EPD Specific Attributes and Services* — Some objects appear in EPD SDM use additional attributes and/or additional services.

9.4.3.3.1 *SAC Object* — Additional attributes and services are mapped as following tables.

Table 29 SAC Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
SacA65	33	Number of Endpoint Objects

Table 30 SAC Object Service Identifiers

Service ID		*Service Name*
SDM	*A-LINK*	
S33	33	Reset Endpoint
S34	34	Download Recipe
S35	35	Upload Recipe
S36	36	Calibrate

9.4.3.3.2 *Sensor-BI-TH-EP Object* — Additional attributes and services are mapped as following tables.

Table 31 Sensor-BI-TH-EP Object Attribute Identifiers

Attribute ID		*Attribute Name*
SDM	*A-LINK*	
EpA1	192	Minimum Time
EpA2	193	Maximum Time
EpA3	194	Target Time
EpA4	195	Elapsed Time
EpA5	196	Time Stamp
EpA6	197	Recipe Identifier
EpA7	198	Step Identifier

Table 32 Sensor-BI-TH-EP Object Service Identifiers

Service ID		*Service Name*
SDM	*A-LINK*	
S1	16	Endpoint On
S2	17	Endpoint Off
S3	18	Endpoint Start
S4	19	Endpoint Suspend
S5	20	Endpoint Resume

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature, respecting any materials or equipment mentioned herein. These standards are subject to change without notice.

By publication of this standard, Semiconductor Equipment and Materials International (SEMI) takes no position respecting the validity of any patent rights or copyrights asserted in connection with any items mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights are entirely their own responsibility.

SEMI E54.18-0306
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK SPECIFIC DEVICE MODEL FOR VACUUM PUMP DEVICE

This standard was technically approved by the global Information and Control Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on November 29, 2005. It was available at www.semi.org in February 2006 and on CD-ROM in March 2006.

1 Purpose

1.1 This specification is part of a suite of standards which specify the implementation of SEMI standards for the Sensor/Actuator Network. The specific purpose of this specification is to describe a network-independent application model comprised of device objects which are common to all Vacuum Pump Devices on a semiconductor equipment Sensor/Actuator communications network.

2 Scope

2.1 This specification for a vacuum pump device specifically addresses the minimum attributes, services, and behavior a Vacuum Pump Device (VPD) device must support to be interoperable on the Sensor/Actuator communication network. A Basic Vacuum Pump (BVP) device is the simplest form of a mechanical roughing pump, a turbo molecular pump and a cryogenic pump identified as a Vacuum Pump Device. The attributes, services and behaviors described for the BVP are required for each pump type. Additional attributes, services and behaviors unique to each of the pump types will be detailed in future versions of this specification.

2.2 This specification is intended to ensure a high-degree of device interoperability on the Sensor/Actuator communication network, while still allowing flexibility for product differentiation and technology evolution.

2.3 The Vacuum Pump Device model specified in this specification is used in conjunction with the Sensor/Actuator Network Common Device Model (CDM) to completely describe the Basic Vacuum Pump as it appears from the network interface.

2.4 This specification, together with the Sensor/Actuator Network Standard, the Sensor/Actuator Network Common Device Model, and a Sensor/Actuator Network Communication Specification, form a complete interoperability specification for the Vacuum Pump Device (VPD).

2.5 To comply with this specification, a device must implement and support, at a minimum, the required attributes, services, and behavior identified in this document. Support for optional attributes, services, and behavior are not required to be compliant to this specification. Optional attributes, services, and behavior are specified in these documents to promote further device interoperability as features evolve and are adopted by more manufacturers. If optional attributes, services, and behavior are implemented for this device, they must be implemented as identified in this document.

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 Limitations

3.1 This specification is a companion to a suite of specifications which together o tonyamake up the Sensor/Actuator Network Communication standard. Therefore, using portions of this specification that relate to network communications necessarily requires an understanding of the associated network specification.

3.2 As this document is a specification for the Vacuum Pump Specific Device Model, it does not contain any definition of objects, attributes, services, or behavioral descriptions that are already defined in the Sensor/Actuator Network Common Device Model (CDM). Additional attributes, attribute assignments, services, and/or service parameters that are Vacuum Pump Device-specific and/or implementation-specific are contained in this specification.

3.3 While this specification is sufficient to completely describe the Basic Vacuum Pump Device as it appears from the communication network, it does not fully describe behavior of the BVP which is not visible from the specific network. This allows flexibility in implementation techniques and product differentiation between manufactures. Manufacture-specific value-add objects, attributes and services may be defined by the manufacturer, but are, by definition, outside the scope of this standard specification.

3.4 This specification is compatible, but not compliant, with SEMI E39. This means that although this specification 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 specification are compatible with those of SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and, thus, are not applied to the Vacuum Pump Device Model.

3.5 Operation over the entire range specified for an attribute within a specific object instance is not a requisite for compliance with this specification.

4 Referenced Standards and Documents

4.1 *SEMI Standards*

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

SEMI E54 — Sensor/Actuator Network Standard

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *VPD* — electronic pump components named Vacuum Pump Device

5.1.2 *BVP* — electronic pump components named Basic Vacuum Pump

5.1.3 *RVP* — electronic pump components named Roughing Vacuum Pump

5.1.4 *TVP* — electronic pump components named Turbo Molecular Vacuum Pump

5.1.5 *CVP* — electronic pump components named Cryogenic Vacuum Pump

5.1.6 *RM* — routine maintenance

5.2 *Definitions*

5.2.1 *Basic Vacuum Pump Device (BVP)* — a self-contained device, consisting of device specific electronics, which is capable of pumping air and other gasses for the purposes of generating negative pressures on its inlet port.

5.2.2 *Roughing Vacuum Pump (RVP)* — a vacuum pump for reducing pressure from atmospheric to a value at which another pumping system can begin to operate. The other pumping system may be a booster attached to the inlet of the roughing pump.

5.2.3 *Turbo Molecular Vacuum Pump (TVP)* — an axial flow turbine for operation in the molecular flow range designed to impart momentum change to gas molecules in a preferential direction from pump inlet to outlet.

5.2.4 *Cryogenic Vacuum Pump (CVP)* — an entrapment vacuum pump operating by the condensation, adsorption, and/or trapping of gas molecules on surfaces cooled to sufficiently low temperatures.

5.3 This document inherits the following terminology defined by the Standard For Sensor/Actuator Network Common Device Model, SEMI E54.1

5.3.1 Attribute

5.3.2 Behavior

5.3.3 Boolean (BOOL)

5.3.4 Byte

5.3.5 Character

5.3.6 Common Device Model (CDM)

5.3.7 Data Type

5.3.8 Data Units

5.3.9 Device

5.3.10 Device Manager (DM) Object

5.3.11 Device Model

5.3.12 Double Integer (DINT)

5.3.13 Enumerated Byte (ENUM)

5.3.14 Full Scale Range

5.3.15 Instance

5.3.16 Last Valid Value (LVV)

5.3.17 Long Integer (LINT)

5.3.18 Long Real (LREAL)

5.3.19 Manufacturer

5.3.20 Nibble

5.3.21 Null Character

5.3.22 Object

5.3.23 Real (REAL)

5.3.24 Sensor (S), Actuator (A), and Controller (C) Objects

5.3.25 Sensor Actuator Controller (SAC) Object

5.3.26 Service

5.3.27 Signed Integer (INT)

5.3.28 Short Integer (SINT)

5.3.29 State Diagram

5.3.30 Text String

5.3.31 Unsigned Double Integer (UDINT)

5.3.32 Unsigned Double Long Integer (UDLINT)

5.3.33 Unsigned Integer (UINT)

5.3.34 Unsigned Long Integer (ULINT)

5.3.35 Unsigned Short Integer (USINT)

6 Requirements

6.1 In order to implement this standard in a Basic Vacuum Pump Device, it is necessary to also implement SEMI E54.1 and one of the Sensor/Actuator Network Communication standards. See §3 for more information on a complete interoperability standard.

7 Conventions

7.1 This document utilizes the conventions specified in the Common Device Model specification document, SEMI E54.1.

8 Device High Level Structure

8.1 *General Description* — A high level object view of a Basic Vacuum Pump (BVP) device is shown in Figure 1. The Basic Vacuum Pump (BVP) is described as the most basic form a VPD may be configured on a Sensor/Actuator communication network. The BVP objects are depicted in Figure 1 only for the purpose of illustrating a high level view of the device and its component objects. In the context of this document, some of these objects are not addressable, do not have addressable attributes, do not have accessible services, nor have any defined behavior. All objects are included in Figure 1 to aid in the visualization of the BVP device.

8.1.1 This document defines in detail the component objects unique to the BVP device. References, rather than definitions, are included for the DM, the SAC, and other objects defined in SEMI E54.1.

8.1.2 Many of the objects defined in this document inherit properties from other objects. The properties inherited include attribute, service, and behavior definitions. These other objects are specified here or in SEMI E54.1.

8.1.3 This document is structured as to allow future extensions to the VPD to describe and include a Roughing Vacuum Pump, a Turbo Molecular Vacuum Pump, and a Cryogenic Vacuum Pump. This document also provides the capability to allow manufacturer-specific enhancements, by reserving object attribute identifiers and object service identifiers. Specifically, all object definitions in this document specify or reserve the first 64 attribute identifiers (A1 through A64) and the first 64 service identifiers (S1 through S64), allowing manufacturers to specify

identifiers beyond these ranges. Additionally, byte-enumerated attributes are specified or reserved from 0 to 63, allowing manufacturers to specify enumerations beyond this range (64 to 255).

8.2 *Basic Vacuum Pump (BVP) Device Description* — The Basic Vacuum Pump device profile is composed of the component objects and object relationships shown in Figure 1.

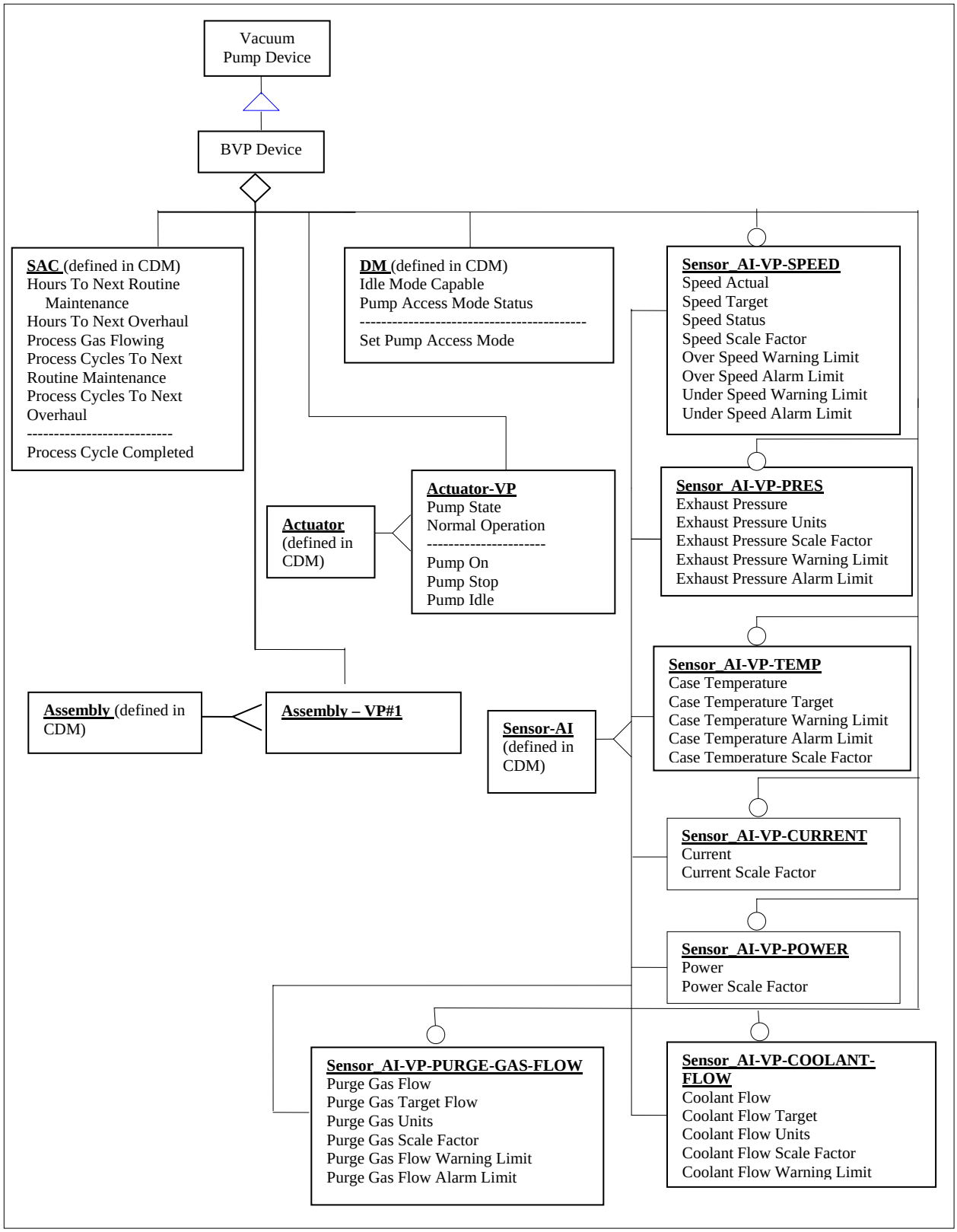


Figure 1
Vacuum Pump Device High Level Structure

8.2.1 *General Requirements* — This section defines in detail the requirements and component objects unique to the BVP device. Additional requirements for future extensions to the VPD to describe and include a Roughing Vacuum Pump, a Turbo Molecular Vacuum Pump, and a Cryogenic Vacuum Pump may be provided as extensions and revisions to this section and document.

8.2.1.1 *Device Objects* — All objects are defined in terms of their object name and instance identifier. Identifiers for all objects described in this document are summarized in Table 1.

Table 1 Basic Vacuum Pump Device Objects

Referenced Document Section	*Object Name*	*Object Identifier*	*BVP Minimum Instances*	*BVP Maximum Instances*
8.2.2	Sensor Actuator Controller (SAC)	VPD1	1	1
8.2.3	Device Manager (DM)	VPD2	1	1
8.2.4	Actuator-VP	VPD3	1	1
8.2.5	Sensor-AI-VP-SPEED	VPD4	0	Manufacturer-Specified
8.2.6	Sensor-AI-VP-PRES	VPD5	0	Manufacturer-Specified
8.2.7	Sensor-AI-VP-TEMP	VPD6	0	Manufacturer-Specified
8.2.8	Sensor-AI-VP-CURRENT	VPD7	0	Manufacturer-Specified
8.2.9	Sensor-AI-VP-POWER	VPD8	0	Manufacturer-Specified
8.2.10	Sensor-AI-VP-COOLANT-FLOW	VPD9	0	Manufacturer-Specified
8.2.11	Sensor-AI-VP-PURGE-GAS-FLOW	VPD10	0	Manufacturer-Specified
8.2.12	Assembly-VP#1	VPD11	1	1
—	Reserved	VPD12 –VPD64	—	—
—	Manufacturer-Specified	> VPD64	—	—

8.2.1.2 *Object Services* — Not all object services listed in this document can necessarily be requested over the network. They are included in this document because their behavior may generate network activity.

8.2.1.3 *Object Behavior* — When a service request is received over the network that is not supported by the object, or contains a parameter value which is beyond the supported range, or which is otherwise invalid, a network-specific service error response is generated as specified in SEMI E54.1 Appendix 2.

8.2.2 *Sensor Actuator Controller Object (SAC)* — The Sensor Actuator Controller object is the device component responsible for coordinating the interaction of the VPD device with the sensory/actuation/control environment as specified in SEMI E54.1. The following sections specify the components of the SAC object that are not specified in the Common Device Model or require further definition than specified in the Common Device Model.

8.2.2.1 *Sensor Actuator Controller Object Attributes* — The attributes provided by the Sensor Actuator Controller object are defined in SEMI E54.1. Additional required and optional SAC object instance attributes are listed in Table 2.

Table 2 SAC Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Hours To Next Routine Maintenance	SacA65	R	No	INT
Hours To Next Overhaul	SacA66	R	No	INT
Process Gas Flowing	SacA67	RW	No	Enumerated Byte
Process Cycles To Next Routine Maintenance	SacA68	R	No	INT
Process Cycles To Next Overhaul	SacA69	R	No	INT

8.2.2.1.1 *Hours To Next Routine Maintenance (Optional)* — An attribute that specifies the time in hours until the next scheduled routine maintenance cycle. This attribute value is decremented by one each time an hour expires and may become a negative value. A negative value indicates that the maintenance cycle has expired and has been exceeded by the negative hours indicated. When the most negative number is reached it is held and is no longer decremented. The Exception Detail Warning[0] Bit[6] and/or Alarm[0] Bit[6] may optionally be set to alert that pump maintenance is required when this attribute decrements to a manufacturer specified value. This attribute is initialized by a manufacturer specific procedure which is outside the scope of this specification.

8.2.2.1.2 *Hours To Next Overhaul (Optional)* — An attribute that specifies the time in hours until the next scheduled device overhaul cycle. An overhaul is an event that requires the physical removal of the pump for servicing. This attribute value is decremented by one each time an hour expires and may become a negative value. A negative value indicates that the overhaul cycle has expired and has been exceeded by the negative hours indicated. When the most negative number is reached it is held and is no longer decremented. The Exception Detail Warning[0] Bit[6] and/or Alarm[0] Bit[6] may optionally be set to alert that pump maintenance is required when this attribute decrements to a manufacturer specified value. This attribute is initialized by a manufacturer specific procedure which is outside the scope of this specification.

8.2.2.1.3 *Process Gas Flowing (Optional)* — An attribute which uniquely identifies to the vacuum pump device that process gas is flowing. This attribute is typically set to indicate to the vacuum pump device that it is pumping a process gas. The interpretation of this information is manufacturer specific. It may be used for example, to help determine the manufacturer's suggested routine maintenance cycle for the pump device. This attribute is an enumerated byte that can take on one of the following values:

- 0 = No process gas is flowing
- 1 = Process gas is flowing
- 2–63 = Reserved
- 64–255 = Manufacturer specific

8.2.2.1.4 *Process Cycles To Next Routine Maintenance (Optional)* — An attribute that records the number of process cycles that can be completed before the device requires scheduled routine maintenance. This attribute is maintained by the pump device based on the number of times the "Process Cycle Completed" service is invoked by the tool together with any other manufacturer specific operating conditions monitored by the pump device. The management of this attribute and the value assigned to this attribute is manufacturer specific. For example, the pump device may monitor its current consumption, temperature, process gas flowing attribute and the process cycle count to determine how many process cycles can be completed before scheduled maintenance is required. The Exception Detail Warning[0] Bit[4,5 and/or 6] (see Table 8), and/or Exception Detail Alarm[0] Bit[6] (see Table 6) may optionally be set to alert that pump maintenance is required when this attribute reaches the manufacturer specified value. This attribute is initialized by a manufacturer specific procedure which is outside the scope of this specification. The definition of a 'Process Cycle' is manufacture specific and is outside the scope of this specification.

8.2.2.1.5 *Process Cycles To Next Overhaul (Optional)* — An attribute that records the number of process cycles that can be completed before the pump device requires a scheduled overhaul. An overhaul is an event that requires the physical removal of the pump for servicing. This attribute is maintained by the pump device based on the number of times that the "Process Cycle Completed" service is invoked by the tool together with any other manufacturer specific conditions monitored by the pump device. The management of this attribute and the value assigned to this attribute is manufacturer specific. For example, the pump device may monitor its current consumption, temperature, process gas flowing attribute and the process cycle count to determine how many process cycles can be completed before an overhaul is required. The Exception Detail Warning[0] Bit[4,5 and/or 6] (see Table 8), and/or Exception Detail Alarm[0] Bit[6] (see Table 6) may optionally be set to alert that pump maintenance is required when this attribute reaches the manufacturer specified value. This attribute is initialized by a manufacturer specific procedure which is outside the scope of this specification. The definition of a 'Process Cycle' is manufacture specific and is outside the scope of this specification.

8.2.2.1.6 *Initial and Default Values*

Table 3 SAC Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Hours To Next Routine Maintenance	LVV	0	
Hours To Next Overhaul	LVV	0	
Process Gas Flowing	0	0	Value based on device "Process Gas Flow" conditions
Process Cycles To Next Routine Maintenance	LVV	0	
Process Cycles To Next Overhaul	LVV	0	

8.2.2.2 *Sensor Actuator Controller Object Services* — The services provided by the Sensor Actuator Controller object are defined in SEMI E54.1. Table 4 contains additional services required for the Sensor Actuator Controller object.

Table 4 SAC Object Services

Service	*Service Identifier*	*Type*	*Description*
Process Cycle Completed	SacS33	R	Used to maintain device scheduled maintenance counters.

8.2.2.2.1 *Process Cycle Completed (Optional)* — This service is used to indicate that a process cycle has been completed. The device uses this to maintain the 'Process Cycles To Next Routine Maintenance' (refer to ¶8.2.2.1.4) and 'Process Cycles To Next Overhaul' (refer to ¶8.2.2.1.5) attributes. There are no parameters specified for this service.

8.2.2.3 *Sensor Actuator Controller Object Behavior* — The behavior exhibited by the Sensor Actuator Controller object is defined in SEMI E54.1. Additional behavior required for the Sensor Actuator Controller object is detailed below.

8.2.2.3.1 The "Process Cycles To Next Routine Maintenance" and "Process Cycles To Next Overhaul" attributes is updated each time the service "Process Cycle Completed" is invoked by the tool. These attributes are managed and set based upon manufacturer specific operating conditions. These attributes may assume a negative value to indicate that the specific maintenance cycle has been exceeded.

8.2.3 *Device Manager Object (DM)* — The Device Manager Object instance is the device component responsible for managing and consolidating the device operation as specified in SEMI E54.1. The following sections specify the components of the DM object that are not specified in the Common Device Model.

8.2.3.1 *Device Manager Object Attributes* — Additional required and optional DM object instance attributes as well as additional requirements placed on attributes defined in SEMI E54.1 are listed in Table 5.

Table 5 Device Manager Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	DmA1	R	Yes	Refer to CDM (SEMI E54.1).
Exception Detail Alarm	DmA13	R	Yes	Refer to CDM (SEMI E54.1).
Exception Detail Warning	DmA14	R	Yes	Refer to CDM (SEMI E54.1).
Idle Mode Capable	DmA33	R	Yes	BOOL
Pump Access Mode Status	DmA34	R	Yes	Enumerated Byte

8.2.3.1.1 *Device Type* — An attribute which uniquely identifies the type of the device on the network. The device type attribute is assigned as follows:

- Basic Vacuum Pump Device = “BVP”
- Roughing Vacuum Pump Device = “RVP”
- Turbo Molecular Vacuum Pump = “TVP”
- Cryogenic Vacuum Pump = “CVP”

8.2.3.1.2 *Exception Detail Alarm* — An attribute which identifies the detailed alarm status of the device. All pumps must support the “General” alarm bit defined below, and optionally other alarm bits. Table 6 and Table 7 define the bit assignments associated with the alarm exception detail for the Vacuum Pump Device. An alarm condition is cleared by a manufacturer specified action. Alarm bits are set and cleared according to manufacturer specified actions. For example, filtering, hysterisis, or timing algorithms may be applied before an alarm condition is set. Definition of these techniques is outside the scope of this specification.

Table 6 Vacuum Pump Device Exception Detail Alarm [0] Bit Assignments

Bit	*Device-Specific Alarm [0]*	*Description*
0	General	The pump is in an alarm condition. At least one alarm condition exists (either one of the other alarm bits is set, or a manufacturer-specific alarm condition exists)
1	Mains Failure	Insufficient Power
2	Device Failure	Physical Fault
3	Startup Timeout	Unable to start up within the manufacturer specified time
4	Reserved	Reserved
5	Reserved	Reserved
6	RM	Pump device is in need of routine maintenance
7	Emergency Stop	Emergency stop button has been pressed

Table 7 Vacuum Pump Device Exception Detail Alarm [1] Bit Assignments

Bit	*Device-Specific Alarm [1]*	*Description*
0	Over Temperature	Pump is operating too hot
1	Over Speed	Pump is running too fast, mechanical failure
2	Under Speed	Pump is running too slow, overload, mechanical failure
3	Coolant Flow	Insufficient flow
4	Purge Gas Flow	Insufficient flow
5	Exhaust Pressure	Exhaust pressure too high
6	Reserved	Reserved
7	Reserved	Reserved

8.2.3.1.3 *Exception Detail Warning* — An attribute which identifies the detailed warning status of the device. A warning indicates that a condition exists that may lead to an alarm condition if not addressed. Table 8 and Table 9 define the bit assignments associated with the warning exception detail for the Vacuum Pump Device. Warning bits are set and cleared according to manufacturer specified actions. For example, filtering, hysterisis, or timing algorithms may be applied before a warning condition is set. Definition of these techniques is outside the scope of this specification.

Table 8 Vacuum Pump Device Exception Detail Warning [0] Bit Assignments

Bit	*Device-Specific Warning [0]*	*Description*
0	General	At least one warning condition exists (either one of the other warning bits is set, or a manufacturer-specific warning condition exists)
1	Mains Failure	Insufficient Power

Bit	*Device-Specific Warning [0]*	*Description*
2	Device Failure	Physical Fault
3	Startup Timeout	Unable to start up within the manufacturer specified time
4	Do Not Start Process	Do not start another process
5	Stop Process Now	Stop the current process safely now
6	RM	Pump device is nearing need of routine maintenance
7	Reserved	Reserved

Table 9 Pump Device Exception Detail Warning [1] Bit Assignments

Bit	*Device-Specific Warning [1]*	*Description*
0	Over Temperature	Pump is operating too hot
1	Over-Speed	Pump is running too fast, mechanical failure
2	Under-Speed	Pump is running too slow, overload, mechanical failure
3	Coolant Flow	Insufficient flow
4	Purge Gas Flow	Insufficient flow
5	Exhaust Pressure	Exhaust pressure too high
6	Reserved	Reserved
7	Reserved	Reserved

8.2.3.1.4 *Idle Mode Capable* — An attribute which specifies whether "Idle" mode is supported by the vacuum pump device. This attribute is a Boolean that can take on one of the following values:

- 0 = Idle mode not supported
- 1 = Idle mode supported

8.2.3.1.5 *Pump Access Mode Status* — An attribute which uniquely identifies the current access mode status of the vacuum pump device. See ¶8.3.2.1 for a full description of the pump access modes. This attribute is an enumerated byte that can take on one of the following values:

- 0 = Both (Remote and Local access)
- 1 = Local access only
- 2 = Remote access only
- 3 – 63 = Reserved
- 64 – 255 = Manufacturer Specific

8.2.3.1.6 *Initial and Default Values*

Table 10 Device Manager Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Device Type	BVP RVP TVP CVP	BVP RVP TVP CVP	BVP = Basic Vacuum Pump Device RVP = Roughing Vacuum Pump Device TVP = Turbo Molecular Vacuum Pump Device CVP = Cryogenic Vacuum Pump Device
Exception Detail Alarm	0	0	Refer to Table 6 and Table 7.
Exception Detail Warning	0	0	Refer to Table 8 and Table 9.
Idle Mode Capable	Manufacturer Specified	Manufacturer Specified	Value based on device specific Idle Mode support.
Pump Access Mode Status	0	0	Pump startup in "Both" access mode.

8.2.3.2 *Device Manager Object Services* — The services provided by the Device Manager object are defined in SEMI E54.1. Table 11 contains additional services required for the Device Manager object.

Table 11 Device Manager Object Services

Service	*Service Identifier*	*Type*	*Description*
Set Pump Access Mode	DmS33	R	Used to define and restrict communication to the device.

8.2.3.2.1 *Set Pump Access Mode (Required)* — This service is used to prompt the Device Manager object to place the pump in Remote, Local or Both access modes. Local access is defined as operation of the pump from a position in physical proximity to the pump. Remote access is defined as operation of the pump from a position physically removed from the pump via the sensor/actuator network. "Remote" access mode is used to prompt the device to restrict all "Local" access to the device. This access mode guarantees that the device will respond only to commands received from the network. "Local" access mode is used to prompt the device to restrict remote access to the device. This service guarantees that the device will not respond to commands received from the network. While in "Local" mode, the pump device may optionally respond to network requests for the value of an attribute, but not to requests to change an attribute. "Both" access mode allows Remote and Local access to the device. The device may only be brought out of Local access mode by a manufacturer specified procedure at the device. All active pumping processes are continued. All active timers and counters are continued and maintained. Table 12 describes the access parameter specified for this service.

Table 12 Set Pump Access Mode Service Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Access Mode Selector	M	—	Byte	Enumerated Byte: 0 = Both (Remote and Local) 1 = Local Access 2 = Remote Access 3 – 63 = Reserved 64 – 255 = Manufacturer Specific

8.2.3.3 *Device Manager Object Behavior* — The behavior exhibited by the Device Manager object is defined in SEMI E54.1. The Device Manager object supports the additional sub-states defined in Figure 2 within the EXECUTING state. Table 13 defines the additional sub-states, and Table 14 defines the additional state transitions specified for the Device Manager object.

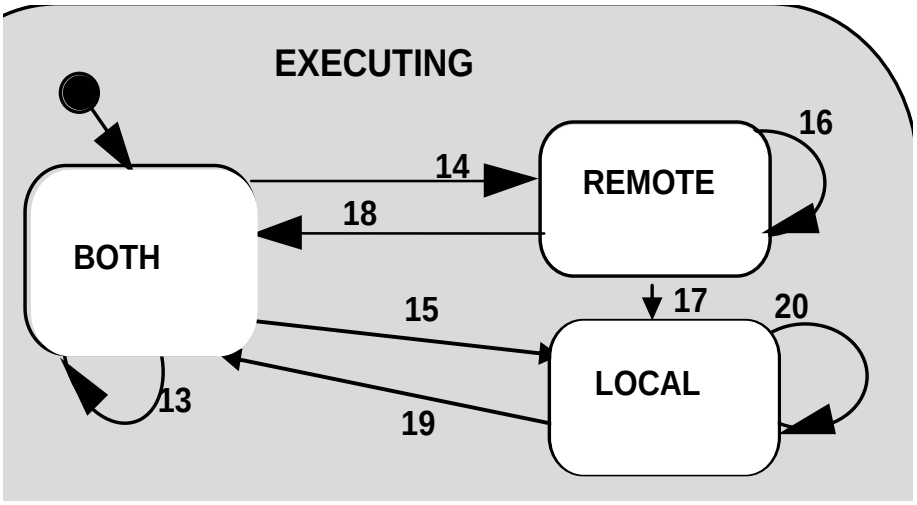


Figure 2
Device Manager Object Behavior Additional Sub-States

Table 13 Device Manager Object Additional Behavior State Descriptions

State	*Description*
BOTH	The Device Manager object instance is EXECUTING in NORMAL OPERATING state and allows Remote and Local control of the device.
REMOTE	The Device Manager object instance is EXECUTING in NORMAL OPERATING state and allows ONLY Remote control of the device.
LOCAL	The Device Manager object instance is EXECUTING in NORMAL OPERATING state and allows ONLY Local control of the device.

Table 14 Device Manager Object Additional Behavior State Transition Matrix

#	*Current State*	*Trigger*	*New State*	*Action*	*Comments*
13	BOTH	Perform Set Pump Mode Access request (selector = 0)	BOTH	Stay in device Remote and Local access mode.	Local and Remote control of the device.
14	BOTH	Perform Set Pump Mode Access request (selector = 2)	REMOTE	Device set to Remote access only. Update Pump Access Mode Status attribute.	Local control of the device not allowed.
15	BOTH	Perform Set Pump Mode Access request (selector = 1	LOCAL	Device set to Local access only. Update Pump Access Mode Status attribute.	Remote control of the device not allowed.
16	REMOTE	Perform Set Pump Mode Access request (selector = 2)	REMOTE	Device remains in Remote access.	Local control of the device not allowed.
17	REMOTE	Perform Set Pump Mode Access request (selector = 1)	LOCAL	Device set to Local access only. Update Pump Access Mode Status attribute.	Remote control of the device not allowed.
18	REMOTE	Perform Set Pump Mode Access request (selector = 0)	BOTH	Device set to Both (Remote and Local) access. Update Pump Access Mode Status attribute.	Local and Remote control of the device allowed.
19	LOCAL	External manufacturer specific action taken at the pump device.	BOTH	Device set to Both (Remote and Local) access. Update Pump Access Mode Status attribute.	Local and Remote control of the device.
20	LOCAL	External manufacturer specific action taken at the pump device.	LOCAL	Device remains in Local access.	Remote control of the device not allowed.

8.2.4 *Actuator-VP Object* — The Actuator-VP object inherits the attributes, services, and behavior of the Actuator object as defined in SEMI E54.1. The Actuator-VP is the device component responsible for retrieving a reading, or readings, from the device specific vacuum pump actuator and then making the result available through the object attribute value.

8.2.4.1 *Actuator-VP Object Attributes* — The attributes provided by the Actuator object are defined in SEMI E54.1. The Actuator-VP object attribute content and its attribute extensions to the Actuator object are listed in the Table 15.

Table 15 Actuator-VP Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Pump State	ActA16	R	Yes	Enumerated Byte
Normal Operation	ActVPA1	R	No	BOOLEAN
Manufacturer Specified	> ActVPA64	—	—	Manufacturer Specific attributes

8.2.4.1.1 *Pump State* — An attribute which uniquely identifies the current operating state of the vacuum pump device. This attribute is an enumerated byte that can take on one of the following values:

- 0 = PUMP STOP
- 1 = PUMP ON
- 2 = PUMP IDLE
- 3 - 63 = Reserved
- 64 - 255 = Manufacturer specific

8.2.4.1.2 *Normal Operation (Optional)* — A Boolean attribute which is TRUE if the pump device is in the PUMP ON state and has achieved stable operating conditions as specified by the manufacturer. When the pump device is switched from PUMP STOP or PUMP IDLE to PUMP ON using the "Pump On" service, the pump device will report the state change promptly, but it may take some time to reach stable operating conditions. The "Normal Operation" attribute, if supported, is set to TRUE only when a stable PUMP ON condition has been achieved. The criteria for stability are manufacturer specific and may include attributes such as speed, temperature and pressure. If the pump device does not reach normal operation in a manufacturer specified time, then the warning bit 3 of the Exception Detail Warning [0] attribute (see Table 8) or the alarm bit 3 of the Exception Detail Alarm [0] attribute (see Table 6) may optionally be set. When the pump device is switched to the PUMP STOP or PUMP IDLE state, the "Normal Operation" attribute is set to FALSE. This attribute can be used to maximize the benefit of idle mode operation by promptly reporting when the pump is ready for process after exiting idle mode. This attribute shall take on one of the following values:

- 0 = FALSE — Pump is not in PUMP ON state or has not stabilized
- 1 = TRUE — Pump is in PUMP ON state and has achieved stable operation

8.2.4.1.3 *Initial and Default Values*

Table 16 Actuator-VP Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Pump State	0	0	Device will power up initially in the PUMP STOP state, but interruptions in power are handled as specified by the pump manufacturer and consistent with safe operation.
Normal Operation	0	0	

8.2.4.2 *Actuator-VP Object Services* — The services provided by the Actuator object instance are defined in SEMI E54.1. Table 17 contains the additional services required for the Actuator-VP object.

Table 17 Actuator-VP Object Services

Service	*Service Identifier*	*Type*	*Description*
Pump On	ActVPS1	R	Used to prompt the Actuator-VP object to take the device from the PUMP STOP or PUMP IDLE state to the PUMP ON state and begin or resume the pumping process.
Pump Stop	ActVPS2	R	Used to prompt the Actuator-VP object to immediately take the device to the PUMP STOP state from PUMP ON or PUMP IDLE state. All active pumping processes are aborted and all active timers are stopped.
Pump Idle	ActVPS3	R	Used to prompt the Actuator-VP object to take the device to the PUMP IDLE state from the PUMP ON or PUMP STOP state. All active pumping processes are suspended.
Reserved	ActVPS4 – ActVPS64	—	Reserved for future expansion
Manufacturer Specified	> ActVPS64	—	Manufacturer Specific services

8.2.4.2.1 *Pump On (Required)* — This service is used to prompt the Actuator-VP object to take the pump to the PUMP ON state immediately. There are no additional parameters associated with this service. This service is used to

prompt the Actuator-VP object to go from the PUMP STOP or PUMP IDLE state to the PUMP ON state. The Pump State attribute is set to PUMP ON and the Normal Operation attribute is set to FALSE. If the device turns on successfully, a "success" response is returned (SEMI E54.1, Appendix 2) and the Normal Operation attribute is set to TRUE when the device reaches stable operation. If the device fails to turn on properly, a "fail" response (SEMI E54.1, Appendix 2) is returned and the Actuator-VP object enters the PUMP STOP state.

8.2.4.2.2 *Pump Stop (Required)* — This service is used to prompt the Actuator-VP object to take the pump to the PUMP STOP state immediately from the PUMP IDLE or PUMP ON state. There are no additional parameters associated with this service. The Pump State attribute is set to PUMP STOP. All pumping event processes are aborted and all active timers are stopped. If the Pump State attribute is already set to PUMP STOP, the Actuator-VP object remains in the PUMP STOP state. If the device turns off successfully, a "success" response is returned (SEMI E54.1, Appendix 2) and the Normal Operation attribute is set to FALSE. If the device fails to turn off properly, a "fail" response (SEMI E54.1, Appendix 2) is returned and the Actuator-VP object enters the PUMP STOP state.

8.2.4.2.3 *Pump Idle (Optional)* — This service is used to prompt the Actuator-VP object to take the pump to the PUMP IDLE state immediately from the PUMP STOP or PUMP ON state. All active pumping processes are suspended. All active timers are suspended and held at their current values. The Pump State attribute is set to PUMP IDLE. If the device enters idle successfully, a "success" response is returned (SEMI E54.1, Appendix 2) and the Normal Operation attribute is set to FALSE. If the pump is not currently ready to operate, a "fail" error response (SEMI E54.1, Appendix 2) is returned and the pump enters the PUMP STOP state. Table 18 describes the Idle Mode Selector parameter specified for this service.

Table 18 Pump Idle Service Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Data Type*	*Description*
Idle Mode Selector	M	—	Byte	Enumerated Byte: 0 = Normal Idle 1 – 63 = Reserved 64 – 255 = Manufacturer Specific

8.2.4.3 *Actuator-VP Object Additional Behavior* — The behavior exhibited by the Actuator object is defined in SEMI E54.1. Actuator-VP object behavior is detailed Figure 3 and Table 19 and Table 20. A device in an alarm condition shall not be allowed to run and shall be placed in the PUMP STOP state. The pump device shall not enter the PUMP IDLE or PUMP ON state while an alarm condition exists.

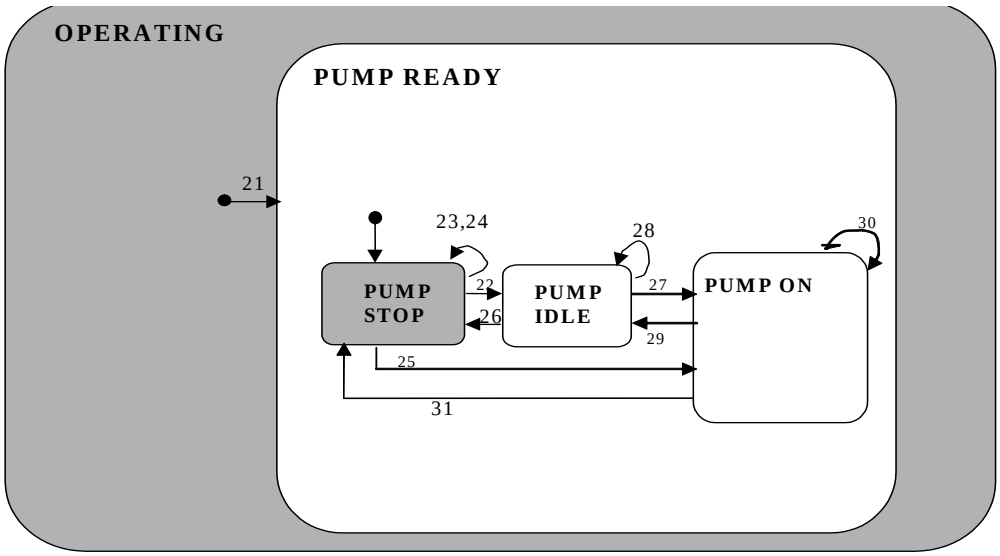


Figure 3
Actuator-VP Object Behavior Within the Pump OPERATING State

Table 19 Actuator-VP Object Behavior OPERATING Sub-state Description

State	*Description*
OPERATING	Pump is powered and able to communicate. Pump is in one of the following enumerated states as indicated in the Actuator-VP "Pump State" attribute: PUMP STOP, PUMP IDLE, PUMP ON. Pump will respond to Pump On, Pump Idle and Pump Stop services as appropriate to move between sub-states within the OPERATING/PUMP READY state.
PUMP READY	Pump is operating and either pumping or ready to pump. Pump is in one of the following enumerated states as indicated in the Actuator-VP "Pump State" attribute: PUMP STOP, PUMP IDLE, and PUMP ON. Pump will respond to Pump On, Pump Stop, Pump Idle services as appropriate to move between states within the PUMP READY state.
PUMP STOP	The pump is ready and not performing any pumping process. This is a sub-state to PUMP READY; PUMP STOP is the setting of the Actuator-VP "Pump State" attribute. Actuator-VP object is **NOT** performing the "pumping" process.
PUMP ON	The pump is ready and performing the pumping process in the normal on condition. This is a sub-state of PUMP READY; PUMP ON is the setting of the Actuator-VP "Pump State" attribute. Actuator-VP object is performing the "pumping" process.
PUMP IDLE	The pump is ready and performing the pumping process in the manufacturer specified idle condition. This is a sub-state to PUMP READY. PUMP IDLE is the setting of the Actuator-VP "Pump State" attribute. Actuator-VP object is performing the "pumping" process as determined by the manufacturer specific idle process.

Table 20 Actuator-VP Behavior OPERATING Sub-state Transition Matrix

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
21	Entry into OPERATING	Device Power Up	PUMP READY / PUMP STOP	Perform any manufacturer specific power-up diagnostics and initialization. Set 'Pump State' attribute to PUMP STOP.	Entry state shall be PUMP STOP.
22	PUMP STOP	Pump Idle service request	PUMP IDLE	Set 'Pump State' attribute to PUMP IDLE.	Behavior associated with determining that the pump will idle properly is manufacturer specific.
23	PUMP STOP	Pump Stop service request	PUMP STOP	Error response.	Pump is already in PUMP STOP state.
24	PUMP STOP	Pump On or Pump Idle service request. Pump is in an alarm condition or is unable to turn on properly.	PUMP STOP	Error response.	Behavior associated with determining that the pump will not turn on properly is manufacturer specific.
25	PUMP STOP	Pump On service request	PUMP ON	Set 'Pump State' attribute to PUMP ON. Set 'Normal Operation' attribute to TRUE when stable if supported.	Behavior associated with determining that the pump will turn on properly is manufacturer specific.
26	PUMP IDLE	Pump Stop service request or alarm condition	PUMP STOP	Set 'Pump State' attribute to PUMP STOP.	Behavior associated with determining that the pump is in an alarm condition is manufacturer specific.

#	Current State	Trigger	New State	Action	Comment
27	PUMP IDLE	Pump On service request	PUMP ON	Pump is Ready and started. Set 'Pump State' attribute to PUMP ON. Set 'Normal Operation' attribute to TRUE when stable if supported.	Behavior associated with determining that the pump will turn on properly is manufacturer specific.
28	PUMP IDLE	Pump Idle service request	PUMP IDLE	Error response. Stay in IDLE state.	Pump is already in idle.
29	PUMP ON	Pump Idle service request	PUMP IDLE	Stop pumping process and enter Idle pump process. Set 'Pump State' attribute to PUMP IDLE. Set 'Normal Operation' attribute to FALSE if supported.	Behavior associated with determining that the pump will idle properly is manufacturer specific.
30	PUMP ON	Pump On service request	PUMP ON	Error response.	Pumping is already on.
31	PUMP ON	Pump Stop service request or alarm condition	PUMP STOP	Stop pumping process and enter OFF pump process. Set 'Pump State' attribute to PUMP STOP. Set 'Normal Operation' attribute to FALSE if supported.	Behavior associated with determining that the pump will turn off properly is manufacturer specific.

8.2.5 *Sensor-AI-VP-SPEED Object (Optional)* — The Sensor-AI-VP-SPEED object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-SPEED is the device component responsible for retrieving a reading, or readings, from the device specific speed sensors and then making the result available through the object attribute value. This object can report an over or under speed warning condition by setting bits 1 or 2 respectively in the "Exception Detail Warning [1] (see Table 9). This object can report an over or under speed alarm condition by setting bits 1 or 2 respectively in the "Exception Detail Alarm [1] (see Table 7).

8.2.5.1 Sensor-AI-VP-SPEED Object Attributes — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-SPEED object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 21.

Table 21 Sensor-AI-VP-SPEED Object Attributes

Attribute Name	Attribute Identifier	Access Network	Required	Data Form
Value (Speed Actual)	SaiA16	R	Yes	INT
Speed Target	SaiVPSpdA1	R	No	INT
Speed Scale Factor	SaiVPSpdA2	R	No	INT
Over Speed Warning Limit	SaiVPSpdA3	R	No	INT
Over Speed Alarm Limit	SaiVPSpdA4	R	No	INT
Under Speed Warning Limit	SaiVPSpdA5	R	No	INT
Under Speed Alarm Limit	SaiVPSpdA6	R	No	INT
Reserved	SaiVPSpdA7 – SaiVPSpdA64	—	—	Reserved for future expansion
Manufacturer Specified	> SaiVPSpdA64	—	—	Manufacturer Specific attributes

8.2.5.1.1 *Value (Speed Actual)* — An attribute that is maintained by the pump device to report the current pump device speed value while controlling pumping operations. The value is represented in "percent of rated full speed".

8.2.5.1.2 *Speed Target (Optional)* — An attribute that specifies the target speed the pump device will control to and maintain when operating properly if the pump device is speed control capable. The value is represented in "percent of rated full speed". The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.5.1.3 *Speed Scale Factor (Optional)* — An attribute that specifies the scale factor for the "Speed Actual" and "Speed Target" attributes. As an example, a value of 10 indicates that the Speed Actual and Speed Target attributes are to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.5.1.4 *Over Speed Warning Limit (Optional)* — An attribute that specifies the pump over speed warning limit value. The value is represented in "percent of rated full speed". If the "Speed Actual" is at or above the value of this attribute, then a warning will be reported by setting bit 1 of the Exception Detail Warning [1] attribute (see Table 9). The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.5.1.5 *Over Speed Alarm Limit (Optional)* — An attribute that specifies the pump over speed alarm limit value. The value is represented in "percent of rated full speed". The value is represented in "percent of rated full speed".If the "Speed Actual" is at or above the value of this attribute, then an alarm will be reported by setting bit 1 of Exception Detail Alarm [1] attribute (see Table 7). The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.5.1.6 *Under Speed Warning Limit (Optional)* — An attribute that specifies the pump under speed warning limit value. The value is represented in "percent of rated full speed". The value is represented in "percent of rated full speed".If the "Speed Actual" is at or below the value of this attribute, then a warning will be reported by setting bit 2 of Exception Detail Warning [1] attribute (see Table 9). The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.5.1.7 *Under Speed Alarm Limit (Optional)* — An attribute that specifies the pump under speed alarm limit value. The value is represented in "percent of rated full speed". The value is represented in "percent of rated full speed".If the "Speed Actual" is at or below the value of this attribute, then an alarm will be reported by setting bit 2 of Exception Detail Alarm [1] (see Table 7). The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.5.1.8 *Initial and Default Values*

Table 22 Sensor-AI-VP-SPEED Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Speed Actual	0	0	
Speed Target	LVV	Manufacturer Specified	
Speed Scale Factor	LVV	Manufacturer Specified	
Over Speed Warning Limit	LVV	Manufacturer Specified	
Over Speed Alarm Limit	LVV	Manufacturer Specified	
Under Speed Warning Limit	LVV	Manufacturer Specified	
Under Speed Alarm Limit	LVV	Manufacturer Specified	

8.2.5.2 *Sensor-AI-VP-SPEED Object Services* — There are no additional services required for the Sensor-AI-VP-SPEED object.

8.2.5.3 *Sensor-AI-VP-SPEED Object Behavior* — There is no additional behavior required for the Sensor-AI-VP-SPEED object.

8.2.6 *Sensor-AI-VP-PRES Object (Optional)* — The Sensor-AI-VP-PRES object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-PRES is the device component responsible for retrieving a reading, or readings, from the device specific pressure sensors and then making the result available through the attribute value. This object can report a pressure warning condition by setting bit 5 in the

"Exception Detail Warning [1] (see Table 9). This object can report a pressure alarm condition by setting bit 5 in the "Exception Detail Alarm [1] (see Table 7).

8.2.6.1 *Sensor-AI-VP-PRES Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-PRES object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 23.

Table 23 Sensor-AI-VP-PRES Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Value (Exhaust Pressure)	SaiA16	R	Yes	INT
Exhaust Pressure Units	SaiVPPresA1	R	No	Enumerated Byte, refer to CDM appendix
Exhaust Pressure Scale Factor	SaiVPPresA2	R	No	INT
Exhaust Pressure Warning Limit	SaiVPPresA3	R	No	INT
Exhaust Pressure Alarm Limit	SaiVPPresA4	R	No	INT
Reserved	SaiVPPresA5 – SaiVPPresA64	—	—	Reserved for future expansion
Manufacturer Specified	> SaiVPPresA64	—	—	Manufacturer Specific attributes

8.2.6.1.1 *Value (Exhaust Pressure)* — An attribute that is maintained by the pump device to report the current exhaust pressure value as measured at the outlet of the pump.

8.2.6.1.2 *Exhaust Pressure Scale Factor (Optional)* — An attribute that specifies the scale factor for the "Exhaust Pressure" and "Exhaust Pressure Target" attributes. As an example, a value of 10 indicates that the Exhaust Pressure and Exhaust Pressure Target attributes are to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.6.1.3 *Exhaust Pressure Units (Optional)* — An attribute that specifies the unit for the "Exhaust Pressure", , "Exhaust Pressure Target", "Exhaust Pressure Warning Limit" and "Exhaust Pressure Alarm Limit" attributes. This attribute is enumerated and can be assigned one of the values specified in Appendix 1 of SEMI E54.1. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.6.1.4 *Exhaust Pressure Warning Limit (Optional)* — An attribute that specifies the pump exhaust pressure warning limit value. If the "Exhaust Pressure" is at or above the value of this attribute, then a warning will be reported by setting bit 5 in the "Exception Detail Warning [1] (see Table 9). The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.6.1.5 *Exhaust Pressure Alarm Limit (Optional)* — An attribute that specifies the pump exhaust pressure alarm limit value. If the "Exhaust Pressure" is at or above the value of this attribute, then an alarm will be reported by setting bit 5 in the "Exception Detail Alarm [1] (see Table 7). The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.6.1.6 *Initial and Default Values*

Table 24 Sensor-AI-VP-PRES Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value (Exhaust Pressure)	0	0	
Exhaust Pressure Units	LVV	Manufacturer Specified	Manufacturer specified from CDM appendix
Exhaust Pressure Scale Factor	LVV	Manufacturer Specified	
Exhaust Pressure Warning Limit	LVV	Manufacturer Specified	
Exhaust Pressure Alarm Limit	LVV	Manufacturer Specified	

8.2.6.2 *Sensor-AI-VP-PRES Object Services* — There are no additional services required for the Sensor-AI-VP-PRES object.

8.2.6.3 *Sensor-AI-VP-PRES Object Behavior* — There is no additional behavior required for the Sensor-AI-VP-PRES object.

8.2.7 *Sensor-AI-VP-TEMP Object (Optional)* — The Sensor-AI-VP-TEMP object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-TEMP is the device component responsible for retrieving a reading, or readings, from the device specific temperature sensor and then making the result available through the attribute value. This object can report an over temperature warning condition by setting bit 0 in the "Exception Detail Warning [1] (see Table 9). This object can report an over temperature alarm condition by setting bit 0 in the "Exception Detail Alarm [1] (see Table 7).

8.2.7.1 *Sensor-AI-VP-TEMP Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-TEMP object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 25.

Table 25 Sensor-AI-VP-TEMP Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Value (Case Temperature)	SaiA16	R	Yes	INT
Case Temperature Target	SaiVPTempA1	R	No	INT
Case Temperature Scale Factor	SaiVPTempA2	R	No	INT
Case Temperature Warning Limit	SaiVPTempA3	R	No	INT
Case Temperature Alarm Limit	SaiVPTempA4	R	No	INT
Reserved	SaiVPTempA5 – SaiVPTempA64	—	—	Reserved for future expansion
Manufacturer Specified	>SaiVPTempA64	—	—	Manufacturer Specific attributes

8.2.7.1.1 *Value (Case Temperature)* — An attribute that is maintained by the pump device to report the current device case temperature during pump operations. The value is represented in "Degrees Centigrade (C)".

8.2.7.1.2 *Case Temperature Target (Optional)* — An attribute that specifies the target case temperature the pump device shall control to and maintain when operating properly if the pump device is case temperature control capable. The value is represented in "Degrees Centigrade (C)". The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.7.1.3 *Case Temperature Scale Factor (Optional)* — An attribute that specifies the scale factor for the "Case Temperature" and "Case Temperature Target" attributes. As an example, a value of 10 indicates that the Case Temperature and Case Temperature Target attributes are to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.7.1.4 *Case Temperature Warning Limit (Optional)* — An attribute that specifies the pump case temperature warning limit value. The value is represented in "Degrees Centigrade (C)". If the Case Temperature is at or above the value of this attribute, then a warning will be reported by setting bit 0 in the "Exception Detail Warning [1] (see Table 9). The setting of this attribute is manufacture specific and is outside the scope of this specification.

8.2.7.1.5 *Case Temperature Alarm Limit (Optional)* —. An attribute that specifies the pump case temperature alarm limit value. The value is represented in "Degrees Centigrade (C)". If the Case Temperature is at or above the value of this attribute, then an alarm will be reported by setting bit 0 in the "Exception Detail Alarm [1] (see Table 7). The setting of this attribute is manufacture specific and is outside the scope of this specification.

8.2.7.1.6 *Initial and Default Values*

Table 26 Sensor-AI-VP-TEMP Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value (Case Temperature)	LVV	0	
Case Temperature Target	LVV	Manufacturer Specified	
Case Temperature Scale Factor	LVV	Manufacturer Specified	
Case Temperature Warning Limit	LVV	Manufacturer Specified	
Case Temperature Alarm Limit	LVV	Manufacturer Specified	

8.2.7.2 *Sensor-AI-VP-TEMP Object Services* — There are no additional services required for the Sensor-AI-VP-TEMP object.

8.2.7.3 *Sensor-AI-VP-TEMP Object Behavior* — There is no additional behavior required for the Sensor-AI-VP-TEMP object.

8.2.8 *Sensor-AI-VP-CURRENT Object (Optional)* — The Sensor-AI-VP-CURRENT object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-CURRENT is the device component responsible for retrieving a reading, or readings, from the device specific current sensors and then making the result available through the attribute value.

8.2.8.1 *Sensor-AI-VP-CURRENT Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-CURRENT object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 27.

Table 27 Sensor-AI-VP-CURRENT Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Value (Current)	SaiA16	R	Yes	INT
Current Scale Factor	SaiVPCurrentA1	R	No	INT
Reserved	SaiVPCurrentA2 – SaiVPCurrentA64	—	—	Reserved for future expansion
Manufacturer Specified	>SaiVPCurrentA64	—	—	Manufacturer Specific attributes

8.2.8.1.1 *Value (Current)* — An attribute that is maintained by the pump device to report the current device electrical current value during pump operations. The value is represented in “amps”.

8.2.8.1.2 *Current Scale Factor (Optional)* — An attribute that specifies the scale format for the “Current” attribute. As an example, a value of 10 indicates that the Current attribute is to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.8.1.3 *Initial and Default Values*

Table 28 Sensor-AI-VP-CURRENT Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value (Current)	LVV	0	
Current Scale Factor	LVV	Manufacturer Specified	

8.2.8.2 *Sensor-AI-VP-CURRENT Object Services* — There are no additional services required for the Sensor-AI-VP-CURRENT object.

8.2.8.3 *Sensor-AI-VP-CURRENT Object Behavior* — There is no additional behavior required for the Sensor-AI—VP-CURRENT object.

8.2.9 *Sensor-AI-VP-POWER Object (Optional)* — The Sensor-AI-VP-POWER object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-POWER is the device component responsible for retrieving a reading, or readings, from the device specific power sensors and then making the result available through the attribute value.

8.2.9.1 *Sensor-AI-VP-POWER Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-POWER object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 29.

Table 29 Sensor-AI-VP-POWER Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Value (Power)	SaiA16	R	Yes	INT
Power Scale Factor	SaiVPPowerA1	R	No	INT
Reserved	SaiVPPowerA2 – SaiVPPowerA64	—	—	Reserved for future expansion
Manufacturer Specified	>SaiVPPowerA64	—	—	Manufacturer Specific attributes

8.2.9.1.1 *Value (Power)* — An attribute that is maintained by the pump device to report the current device power value during pump operations. The value is represented in “kilowatts”.

8.2.9.1.2 *Power Scale Factor (Optional)* — An attribute that specifies the scale factor for the “Power” attribute. As an example, a value of 10 indicates that the Power attribute is to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.9.1.3 *Initial and Default Values*

Table 30 Sensor-AI-VP-POWER Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value (Power)	LVV	0	
Power Scale Factor	LVV	Manufacturer Specified	

8.2.9.2 *Sensor-AI-VP-POWER Object Services* — There are no additional services required for the Sensor-AI-VP-POWER object.

8.2.9.3 *Sensor-AI-VP-POWER Object Behavior* — There is no additional behavior required for the Sensor-AI—VP-POWER object.

8.2.10 *Sensor-AI-VP-COOLANT-FLOW Object (Optional)* — The Sensor-AI-VP-COOLANT-FLOW object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-COOLANT-FLOW is the device component responsible for retrieving a reading, or readings, from the device specific flow sensors and then making the result available through the attribute value. This object can report an insufficient flow warning condition by setting bit 3 in the “Exception Detail Warning [1] (see Table 9). This object can report an insufficient flow alarm condition by setting bit 3 in the “Exception Detail Alarm [1] (see Table 7).

8.2.10.1 *Sensor-AI-VP-COOLANT-FLOW Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-COOLANT-FLOW object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 31.

Table 31 Sensor-AI-VP-COOLANT-FLOW Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Value (Coolant Flow)	SaiA16	R	Yes	INT
Coolant Flow Target	SaiVPFlowA1	R	No	INT
Coolant Flow Units	SaiVPFlowA2	R	No	Enumerated Byte, refer to CDM appendix
Coolant Flow Scale Factor	SaiVPFlowA3	R	No	INT

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Coolant Flow Warning Limit	SaiVPFlowA4	R	No	INT
Coolant Flow Alarm Limit	SaiVPFlowA5	R	No	INT
Reserved	SaiVPFlowA6 – SaiVPFlowA64	—	—	Reserved for future expansion
Manufacturer Specified	>SaiVPFlowA64	—	—	Manufacturer Specific attributes

8.2.10.1.1 *Value (Coolant Flow)* — An attribute that is maintained by the pump to report the current device coolant flow value during pump operations.

8.2.10.1.2 *Coolant Flow Target (Optional)* — An attribute that specifies the coolant flow target the pump device shall control to and maintain when operating properly if the pump device is coolant flow control capable. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.10.1.3 *Coolant Flow Units (Optional)* — An attribute that specifies the unit for the "Coolant Flow", "Coolant Flow Target", "Coolant Flow Warning Limit" and "Coolant Flow Alarm Limit" attributes. This attribute is enumerated and can be assigned one of the values specified in Appendix 1 of SEMI E54.1. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.10.1.4 *Coolant Flow Scale Factor (Optional)* — An attribute that specifies the scale factor for the Coolant Flow and Coolant Flow Target attributes. As an example, a value of 10 indicates that the Coolant Flow and Coolant Flow Target attributes are to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.10.1.5 *Coolant Flow Warning Limit (Optional)* — An attribute that specifies the pump coolant flow warning limit value. If the "Coolant Flow" is at or below the value of this attribute, then a warning will be reported by setting bit 3 in the "Exception Detail Warning [1] (see Table 9). The setting of this attribute is manufacture specific and is outside the scope of this specification.

8.2.10.1.6 *Coolant Flow Alarm Limit (Optional)* — An attribute that specifies the pump coolant flow alarm limit value. If the "Coolant Flow" is at or below the value of this attribute, then an alarm will be reported by setting bit 3 in the "Exception Detail Alarm [1] (see Table 7). The setting of this attribute is manufacture specific and is outside the scope of this specification.

8.2.10.1.7 *Initial and Default Values*

Table 32 Sensor-AI-VP-COOLANT-FLOW Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value (Coolant Flow)	LVV	0	
Coolant Flow Target	LVV	Manufacturer Specified	
Coolant Flow Units	LVV	Manufacturer Specified	Manufacturer specified from CDM appendix
Coolant Flow Scale Factor	LVV	Manufacturer Specified	
Coolant Flow Warning Limit	LVV	Manufacturer Specified	
Coolant Flow Alarm Limit	LVV	Manufacturer Specified	

8.2.10.2 *Sensor-AI-VP-COOLANT-FLOW Object Services* — There are no additional services required for the Sensor-AI-VP-COOLANT-FLOW object.

8.2.10.3 *Sensor-AI-VP-COOLANT-FLOW Object Behavior* — There is no additional behavior required for the Sensor-AI-VP-COOLANT-FLOW object.

8.2.11 *Sensor-AI-VP-PURGE-GAS-FLOW Object (Optional)* — The Sensor-AI-VP-PURGE-GAS-FLOW object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-VP-PURGE-GAS-FLOW is the device component responsible for retrieving a reading, or readings, from the device specific flow sensors and then making the result available through the attribute value. This object can report an insufficient flow warning condition by setting bit 4 in the "Exception Detail Warning [1] (see Table 9). This object can report an insufficient flow alarm condition by setting bit 4 in the "Exception Detail Alarm [1] (see Table 7).

8.2.11.1 *Sensor-AI-VP-PURGE-GAS-Flow Object Attributes* — The attributes provided by the Sensor-AI object are defined in SEMI E54.1. The Sensor-AI-VP-PURGE-GAS-FLOW object attribute content and its attribute extensions to the Sensor-AI object are listed in the Table 33.

Table 33 Sensor-AI-VP-PURGE-GAS-FLOW Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Value (Purge Gas Flow)	SaiA16	R	Yes	INT
Purge Gas Flow Target	SaiVPPurgeGasFlowA1	R	No	INT
Purge Gas Flow Units	SaiVPPurgeGasFlowA2	R	No	Enumerated Byte, refer to CDM appendix
Purge Gas Flow Scale Factor	SaiVPPurgeGasFlowA3	R	No	INT
Purge Gas Flow Warning Limit	SaiVPPurgeGasFlowA4	R	No	INT
Purge Gas Flow Alarm Limit	SaiVPPurgeGasFlowA5	R	No	INT
Reserved	SaiVPPurgeGasFlowA6 – SaiVPPurgeGasFlowA64	—	—	Reserved for future expansion
Manufacturer Specified	>SaiVPPurgeGasFlowA64	—	—	Manufacturer Specific attributes

8.2.11.1.1 *Value (Purge Gas Flow)* — An attribute that is maintained by the pump device to report the current device purge gas flow value during pump operations.

8.2.11.1.2 *Purge Gas Flow Target (Optional)* — An attribute that specifies the purge gas flow target the pump device will control to and maintain when operating properly if the pump device is purge gas flow control capable. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.11.1.3 *Purge Gas Flow Units (Optional)* — An attribute that specifies the unit for the "Purge Gas Flow" and "Purge Gas Flow Target" attributes. This attribute is enumerated and can be assigned one of the values specified in Appendix 1 of SEMI E54.1. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.11.1.4 *Purge Gas Flow Scale Factor (Optional)* — An attribute that specifies the scale factor for the "Purge Gas Flow" and "Purge Gas Flow Target" attributes. As an example, a value of 10 indicates that the Purge Gas Flow and Purge Gas Flow Target attributes are to be divided by 10. The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.11.1.5 *Purge Gas Flow Warning Limit (Optional)* — An attribute that specifies the pump purge gas flow warning limit value. If the "Purge Gas Flow" is at or below the value of this attribute, then a warning will be reported by setting bit 4 in the "Exception Detail Warning [1] (see Table 9). . The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.11.1.6 *Purge Gas Flow Alarm Limit (Optional)* — An attribute that specifies the pump purge gas flow alarm limit value. If the "Purge Gas Flow" is at or below the value of this attribute, then an alarm will be reported by setting bit 4 in the "Exception Detail Alarm [1] (see Table 7). . The setting of this attribute is manufacturer specific and is outside the scope of this specification.

8.2.11.1.7 *Initial and Default Values*

Table 34 Sensor-AI-VP-PURGE-GAS-FLOW Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Value (Purge Gas Flow)	LVV	0	
Purge Gas Flow Target	LVV	Manufacturer Specified	
Purge Gas Flow Units	LVV	Manufacturer Specified	Manufacturer specified from CDM appendix
Purge Gas Flow Scale Factor	LVV	Manufacturer Specified	
Purge Gas Flow Warning Limit	LVV	Manufacturer Specified	
Purge Gas Flow Alarm Limit	LVV	Manufacturer Specified	

8.2.11.2 *Sensor-AI-VP-PURGE-GAS-FLOW Object Services* — There are no additional services required for the Sensor-AI-VP-PURGE-GAS-FLOW object.

8.2.11.3 *Sensor-AI-VP-PURGE-GAS-FLOW Object Behavior* — There is no additional behavior required for the Sensor-AI-VP- PURGE-GAS-FLOW object.

8.2.12 *Assembly-VP#1 Object* — Assembly-VP#1 object inherit attributes, services, and behavior from the Assembly object. The Assembly object is the device component that provides a mechanism of grouping more than one attribute from one or more objects into a single data structure for communication over the network. Table 35 identifies the Assembly-VP objects defined for the BVP device.

Table 35 Assembly-VP Object List

Object	*Access Network*	*Required*	*Form*
Assembly-VP#1 (Default Assembly)	R	Yes	Device Status, Pump State, Alarm Exception Detail, Warning Exception Detail

8.2.12.1 *Assembly-VP#1 Object Attributes* — Table 36 provides a list of attributes common to all Assembly-VP#1 object type.

Table 36 Assembly-VP#1 Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Data Form*
Data (Inherited from the Assembly object as shown in Figure 1)	A1	R	Yes	Structure as defined below

8.2.12.1.1 *Data* — The data attribute of the Assembly-VP#1 object is a structured attribute containing an ordered list of attributes within its structure. In the following Table 37, the structure of the Data attribute for the Assembly-VP#1 object type is defined.

Table 37 Assembly-VP#1 Object Data List

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
1	DM	DmA12	Device Manager Exception Status
2	VPD3	ActA16	Actuator-VP Pump State
3	DM	DmA13	Device Manager Exception Detail Alarm

Data Index	*Source Object ID*	*Source Attribute ID*	*Description*
4	DM	DmA14	Device Manager Exception Detail Warning

8.2.12.2 *Assembly-VP#1 Object Services* — There are no additional services required for the Assembly-VP#1 object.

8.2.12.3 *Assembly-VP#1 Object Behavior* — There is no additional behavior required for the Assembly-VP#1 object.