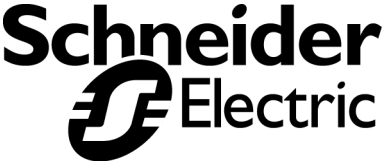
SEMI SAN
Elementary Function Block Library
User Guide

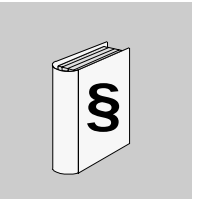
Version 1.0



31002879 01

Table of Contents

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death, serious injury, or equipment damage.

WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

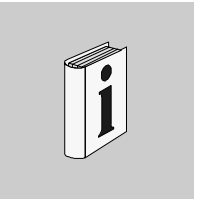
CAUTION

CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE

Electrical equipment should be serviced only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material. This document is not intended as an instruction manual for untrained persons.

About the Book



At a Glance

Document Scope

This documentation introduces you to the Semiconductor Equipment and Materials International (SEMI) trade association's Standard E54 Sensor/Actuator network (SAN) library of elementary function blocks (EFBs) and gives you detailed information for their use.

Validity Note

This documentation applies to Concept 2.5, SR2 or later.

Related Documents

Title of Documentation	Reference Number
Concept 2.5 Installation Instructions	840 USE 492 00
Concept 2.5 User Manual	840 USE 493 00
Concept EFB User Manual	840 USE 495 00
Concept 2.5 LL984 Block Library	840 USE 496 00
Momentum M1 Process Adapter and Option Adapter User Guide	870USE10100

Product Related Warnings

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

User Comments

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

SEMI SAN Function Block Library

1

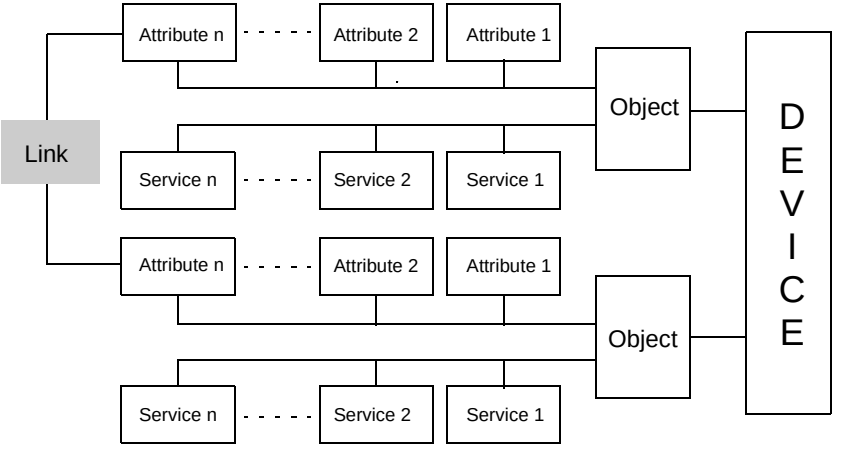
Introduction to the SEMI SAN Elementary Function Block Library

At a Glance

The SEMI SAN Elementary Function Block (EFB) Library lets Momentum M1E processors communicate with sensors and actuators that use the Sensor/Actuator network (SAN) as defined by the Semiconductor Equipment and Materials International trade association (SEMI) standard E54.
The Sensor/Actuator network (SAN), as defined by the SEMI standard E54, provides a standardized object-oriented interface to devices at the user/application layer. The M1E processors use their Ethernet communication port to communicate with these devices. M1E processors use either the Modbus or TCP/IP protocol to communicate.

Note: You should refer to, and be familiar with, the SEMI Standard E54 before using the SEMI SAN Elementary Function Block Library.

The following block diagram shows a typical SAN device configuration.



M1E Functions

The M1E processors function as either a client or a server on the Sensor/Actuator network. In the client mode you can read and write attribute data with a device. In the server mode you can create sensors and actuators. This EFB library is compatible with the PLC programming language, Concept 2.5, SR2 or later.
The SEMI SAN Elementary Function Block Library features are available with the following M1E processors only (they will not operate with any other Momentum processors):

- 171CCC96091 Ethernet and I/O bus communications ports
- 171CCC98091 Ethernet and Modbus RS485 communications ports

Note: These processors are not limited to use with the SEMI SAN Elementary Function Block Library.

Note: These processors are fully functional Momentum M1E controllers, except for either of two options available in the Code Generation Options dialog box.

- include diagnosis information
- include IEC upload information

Select Project | Code Generation Options ... .

SEMI SAN Library

The tables below give the names and a short description of the EFBs available in the SEMI SAN Elementary Function Block Library.
The following client side EFBs are provided in the SEMI SAN Elementary Function Block Library.

EFB Name	Function
SAN_READ	Issues a read attribute request to an object and analyzes the data response.
SAN_READ Types	SAN_R_INT, SAN_R_UINT, SAN_R_DINT, SAN_R_UDINT, SAN_R_REAL. These are similar to the SAN_READ EFB except the data output pin is type safe and the length pin is not required. Data output type differentiate them.
SAN_WRIT	Issues a write attribute request to an object and analyzes the data response.
SAN_WRIT Types	SAN_W_INT, SAN_W_UINT, SAN_W_DINT, SAN_W_UDINT, SAN_W_REAL. These are similar to the SAN_WRIT EFB except the data output pin is type safe and the length pin is not required. Data input type differentiate them.

The following server side EFBs are provided in the SEMI SAN Elementary Function Block Library.

EFB Name	Function
SAN_DEV	Acts as the collecting point for up to thirty objects belonging to a single device.
SAN_OBJ	Acts as the collecting point for all attributes belonging to a single device's object. The following three object types are supported: ● Sensor/Actuator/Controller Object (SAC) ● Device Manager Object (DM) ● Active Element Object (AE)
SAN_ATTR	Acts as the collecting point for all of the configuration items (data) that are associated with a single attribute.
SAN_ASMB	Acts as the collecting point for all of the configuration items (data) that are associated with a single attribute. It can pack several attributes of a device thereby reducing network traffic by allowing one read or write client access to all members of the assembly.
SAN_STRUCT	The SAN_STRUCT EFB complements the SAN_ATTR EFB by providing a method of structuring data. It provides the attribute ID for a structured object.
SAN_SMAP	Allows the entire SAN device setup table to be put into 4X registers in state RAM of the M1E processor.

Important

Note: The SEMI SAN EFB library is not intended to be used with the Concept simulator.

Note: Attributes, with data types having a length of less than 2 bytes or greater than 4 bytes, must use both a structured data type and either the SAN_READ or SAN_WRIT EFB.
Use either one of these two types of structured data type:

- Array
- Structure

Client Side Elementary Function Blocks (EFBs)

2

Overview of Client Side EFBs

Introduction

The EFBs used on the client side initiate requests to the server. This chapter gives you the technical information needed to introduce these EFBs into a sensor/actuator network with the M1E processors 171CCC96091 or 171CCC98091.
Also found in this chapter is a description of the Swapping EFBs, which are used with read and write style EFBs to correct byte order.

Note: The SEMI SAN EFB library is not intended to be used with the Concept simulator.

What's in this Chapter?

This chapter contains the following topics:

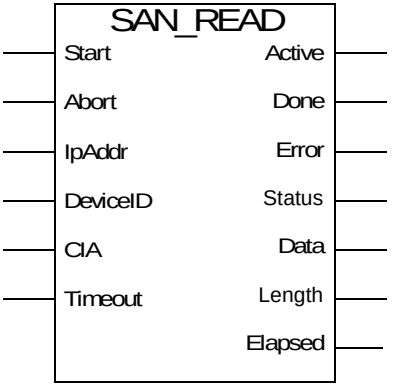
SAN_READ EFB

Functional Description

The SAN_READ Elementary Function Block (EFB) issues a read attribute request to an object. It analyzes the incoming data response to ensure that the data is in the correct format.

Representation

A block representation and a table listing the parameters of the SAN_READ EFB follow.
The following figure is a block representation of the SAN_READ EFB.



Parameter Description

The following table describes the Function Block Parameters.

Parameters	Data Type	Meaning
Start Input	BOOL	This initiates a read attribute operation when transitioning from FALSE to TRUE, and a request is not in progress, and Abort is FALSE. Use the Abort input to stop a pending request. **Start** is only evaluated when the EFB is not active, it must transition from TRUE to FALSE and then FALSE to TRUE when the **Active** pin is FALSE.
Abort input	BOOL	This cancels an ongoing request. It must be FALSE to start a read operation.
IpAddr Input	OMP_IP	This input parameter provides complete routing information for the specified OMP attribute on an Sensor/Actuator Network. This routing information conforms with the SEMI standards and is as follows: • ByteArr5[1] 1st byte of IP address • ByteArr5[2] 2nd byte of IP address • ByteArr5[3] 3rd byte of IP address • ByteArr5[4] 4th byte of IP address
DeviceID input	UINT	This represents ID of the device to be read.
CIA input	OMP_CIA	This represents the class ID, instance ID, and attribute ID of the attribute to be read.
Timeout input	TIME	Here you can enter the maximum time (Timeout limit) the EFB should try to complete a transaction, e.g. 200ms, 5s500ms. If the timeout limit is reached and the EFB operation is not complete, the timeout status code will be generated and Error will be set. If not connected or set to zero, the EFB will act as if there is no timeout.
Active Output	BOOL	This indicates the operation is in progress. It is TRUE while the read operation is in progress and then becomes FALSE.
Done Output	BOOL	This is set to FALSE at the start of a read operation and becomes TRUE when complete. It remains TRUE until the start of the next read operation.
Error Output	BOOL	This is TRUE when the read attribute fails. The 'Status' output provides additional detail.

Parameters	Data Type	Meaning
Status Output	WORD	This presents additional information in case a read operation fails. See *Error/Status Codes for SAN_READ, SAN_R_xx EFBs, p. 44*. It is 0 if SAN_READ is not started or if the read operation was successful.
Data Output	ANY	A variable (located or unlocated) of any data type may be connected to this pin. The byte length of the variable will be determined by the EFB. In case a read operation succeeds, the received data will be copied into the connected variable. In case the received data exceeds the size of the connected variable, the EFB will show an error and no data will be copied. **Note:** If this output is longer than one byte a swapping EFB must be used. If the data is an assembly or a structure each element must be separately swapped.
Length Output	UINT	After a read operation, ‘Length’ always contains the byte length of the received attribute. Even in cases where the variable connected to the 'Data' parameter is smaller than the received attribute causing an error, the Length will show the byte size of the returned data.
Elapsed output	TIME	This displays the time the EFB is taking to complete while it is active, and the total time to complete when Done or Error has occurred. By observing the Elapsed output, you can make decisions regarding network performance and the effect of changing the Start times of EFBs.

Function Blocks SAN_R_INT, SAN_R_UINT, SAN_R_DINT, SAN_R_UDINT, and SAN_R_REAL

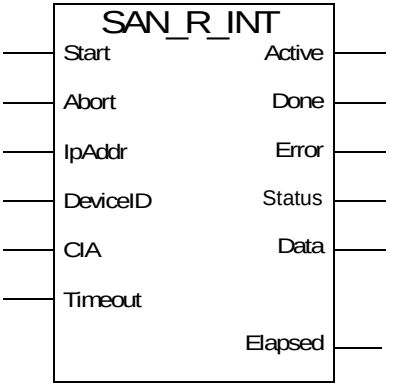
Functional Description

These function blocks are all similar to the SAN_READ EFB except the Data output pin is type safe and the Length pin is not required. The distinguishing data output types are found below.

Note: It is not necessary to use a swapping EFB on the DATA pin with these type safe EFBs.

Representation

The following figure is a block representation of the these EFBs.



Parameter Description

The following table describes the Data output types for individual EFBs.

EFB Name	Data Output Type	Description
SAN_R_INT	INT	Integer
SAN_R_UINT	UINT	Unsigned Integer
SAN_R_DINT	DINT	Double Integer
SAN_R_UDINT	UDINT	Unsigned Double Integer
SAN_R_REAL	REAL	Real

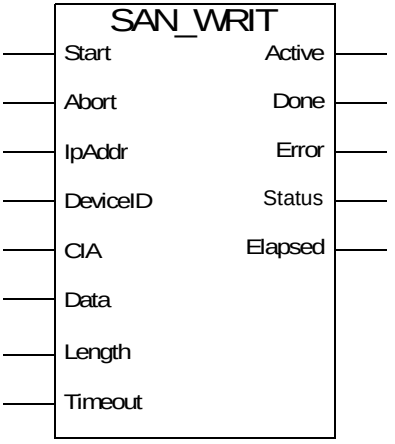
SAN_WRIT EFB

Functional Description

The SAN_WRIT Elementary Function Block (EFB) issues a write attribute request to an object. It analyzes the incoming data response to ensure that the data is in the correct format.

Representation

A block representation and a table listing the parameters of the SAN_WRIT EFB follow.



Parameter Description

The following table describes the Function Block Parameters.

Parameters	Data Type	Meaning
Start Input	BOOL	This initiates a write attribute operation when transitioning from FALSE to TRUE, and a request is not in progress, and Abort is FALSE. Use the Abort input to stop a pending request. **Start** is only evaluated when the EFB is not active, it must transition from TRUE to FALSE and then FALSE to TRUE when the **Active** pin is FALSE.
Abort input	BOOL	This cancels an ongoing request. It must be FALSE to start a write operation.
IpAddr Input	OMP_IP	This input parameter provides complete routing information for the specified OMP attribute on an Sensor/Actuator Network. This routing information conforms with the SEMI standards and is as follows: • ByteArr5[1] 1st byte of IP address • ByteArr5[2] 2nd byte of IP address • ByteArr5[3] 3rd byte of IP address • ByteArr5[4] 4th byte of IP address
DeviceID input	UINT	This represents the ID of the device to be written to.
CIA input	OMP_CIA	This represents the class ID, instance ID, and attribute ID of the attribute to be written.
Timeout input	TIME	Enter the maximum number of milliseconds the EFB should try to complete a transaction. If the timeout limit is reached and the EFB operation is not complete, the timeout status code will be generated and Error will be set. If not connected or set to zero, the EFB will act as if there was no timeout.
Data Input	ANY	A variable (located or unlocated) of any data type may be connected to this pin. The byte length of the variable will be determined by the EFB internally. The amount of bytes taken from the connected variable is defined by parameter Length. The byte stream will start at the lowest physical address (little endian) of that variable. **Note:** Data to this pin must be swapped by a swapping EFB. If the data is an assembly or a structure each element must be separately swapped.

Parameters	Data Type	Meaning
Length Input	UINT	This specifies the number of bytes to be taken from the variable connected to the 'Data' parameter. If Length specifies more bytes than the variable connected to Data can provide, an error will occur and an appropriate error code will be set in the Status output. See *Error Codes for SAN_WRIT, SAN_W_xx EFBs, p. 46*.
Active Output	BOOL	This indicates the operation is in progress. It is TRUE while the write operation is in progress and then becomes FALSE.
Done Output	BOOL	This is set to FALSE at the start of a write operation and becomes TRUE when complete. It remains TRUE until the start of the next write operation.
Error Output	BOOL	This is TRUE when the write attribute fails. The 'Status' output provides additional detail.
Status Output	WORD	This presents additional information in case a write operation fails. See *Error Codes for SAN_WRIT, SAN_W_xx EFBs, p. 46*. It is 0 if SAN_WRIT is not started or if the write operation was successful.
Elapsed output	TIME	This displays the time the EFB is taking to complete while it is active, and the total time to complete when Done or Error has occurred. By observing the Elapsed output, the user can make decisions regarding network performance and the effect of changing the Start times of EFBs in the user program.

Function Blocks SAN_W_INT, SAN_W_UINT, SAN_W_DINT, SAN_W_UDINT, and SAN_W_REAL

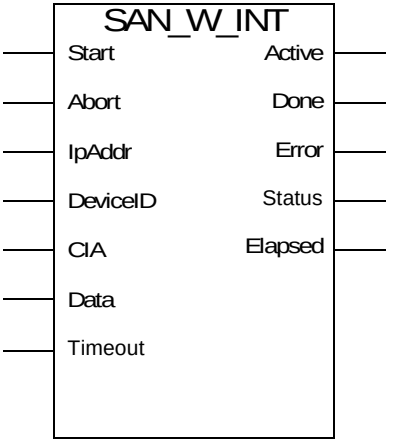
Functional Description

These function blocks are all similar to the SAN_WRIT EFB except the Data output pin is type safe and the Length pin is not required. The distinguishing data output types are found below.

Note: It is not necessary to use a swapping EFB on the DATA pin with these type safe EFBs.

Representation

The following figure is a block representation of the these EFBs.



Parameter Description

The following table describes the Data output types for individual EFBs.

EFB Name	Data Output Type	Description
SAN_W_INT	INT	Integer
SAN_W_UINT	UINT	Unsigned Integer
SAN_W_DINT	DINT	Double Integer
SAN_W_UDINT	UDINT	Unsigned Double Integer
SAN_W_REAL	REAL	Real

Note: For more detailed information see *SAN_WRIT EFB, p. 18*.

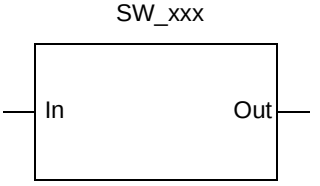
Swapping EFBs

Functional Description

Swapping EFBs correct the data byte sequence out of a READ EFB or into a WRITE EFB. There are five Swapping EFBs, each for a particular Data type. They all have one input, labeled In, and one output, labeled Out.

Block Structure

Here is a universal block structure of a Swapping EFB.



Swapping EFB List

Here is a list of all the Swapping EFBs. The names indicate which data type the particular Swapping EFB is used for.

- SW_INT
- SW_UINT
- SW_DINT
- SW_UDINT
- SW_REAL

How They Are Used

- **When used with a SAN_READ EFB** the Data input to the Swapping EFB comes from the data output of a SAN_READ Data pin, usually after the data has been input to a structure representing a SAN_ASMB or SAN STRUCT. In these cases a swapping EFB must be used on each individual member of the structure. The correct Concept byte order is then found on the output of the Swapping EFBs. Choose the proper Swapping EFB type by identifying data type.
- **When used with a SAN_WRIT EFB** the Data input to the Swapping EFB is a Concept register, variable, structure or constant. In the case of a structure, typically representing a SAN_ASMB or SAN_STRUCT, each member of a Concept-correct structure must be converted to a Wire-correct structure before the data is connected to the SAN_WRIT Data Input pin. The Swapping EFB Data output is connected to the Data input pin of a SAN_WRIT, only for elementary types, and represents a byte stream that is correct for transmission on the wire.

Server Side Elementary Function Blocks (EFBs)

3

Overview of Server Side EFBs

Introduction

The SEMI SAN Elementary Function Block library has six EFBs that you can use on the server side of a sensor/actuator network. EFBs used on the server side respond to requests from clients. This chapter gives you the technical information needed for the introduction of these function blocks into a sensor/actuator network with M1E processors 171CCC96091 or 171CCC98091.

Note: Server Side EFBs are limited to five (5) concurrent devices.

What's in this Chapter?

This chapter contains the following topics:

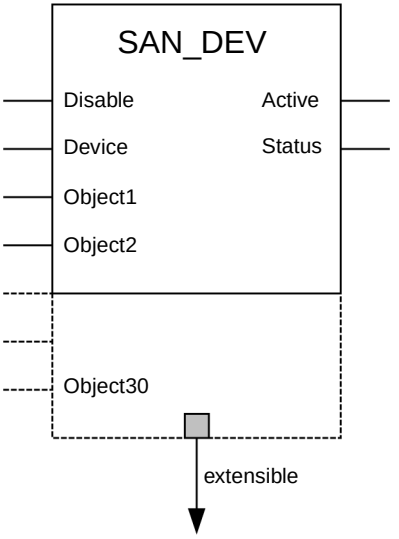
SAN_DEV EFB

Functional Description

The SAN_DEV Elementary Function Block (EFB) acts as the collecting point for all of the objects that belong to a single device. This EFB supports up to 30 objects for a single device.

Representation

The following figure is a block representation of the SAN_DEV EFB



Parameter Description

The following table describes the Function Block Parameters

Parameters	Data Type	Meaning
Disable Input	BOOL	If TRUE, the device and all attached objects are not visible to the internal OMP handler. If this input is not connected the OMP handler assumes FALSE and always 'sees' the device.
Device Input	BYTE	Assigns a logical device number to the SAN_DEV for use when the PLC is hosting multiple SAN_DEV objects. The same value also corresponds to the Unit-ID of the Modbus/TCP protocol, so that multiple devices on the same PLC could be addressed with different Unit-Ids in the Modbus/TCP message frame, according to the OMP standard of Modbus/TCP. If the PLC will host only one device this pin may be left open to automatically assign the SEMI default value 0 as the device's unit number.
Object Input	Object ID	This should be linked to the ObjectID output parameter of a SAN_OBJ. Up to 30 SAN_OBJs can be linked or attached to a SAN_DEV
Active Output	BOOL	If the value is TRUE then the device has been accepted by the PLC's internal OMP handler. If the value is FALSE then the output parameter 'Status' will contain more specific information.
Status Output	WORD	The Status Output delivers an error/status code. See *Error/Status Codes for SAN_DEV EFB, p. 48*.

SAN_OBJ EFB

Functional Description

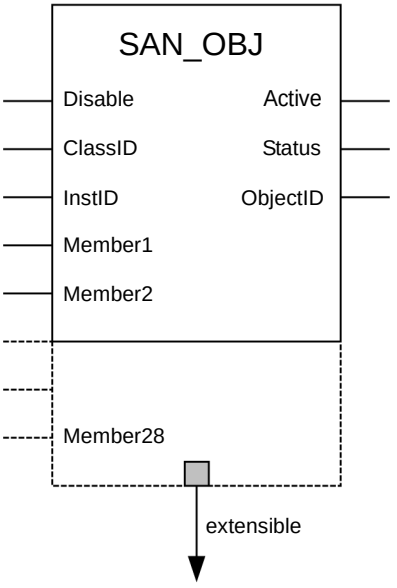
The SAN_OBJ EFB encapsulates a SEMI SAN object. This object represents one of the three types of SAN objects that are related to a device:
The following three object types are supported:

- Sensor/Actuator/Controller Object (SAC)
- Device Manager Object (DM)
- Any other Active Element Object (AE)

The SAN_OBJ acts as the collecting point for all attributes associated with an object. The class ID and instance ID of an object are assigned to the SAN_OBJ and up to 28 attributes may be assigned.

Representation

The following figure is a block representation of the SAN_OBJ EFB.



Parameter Description

The following table describes the Function Block Parameters

Parameters	Data Type	Meaning
Disable Input	BOOL	If TRUE the object, and all attached attributes, are not visible to the internal OMP handler. If this input is not connected the OMP handler assumes FALSE and always 'sees' the object.
ClassID Input	UINT	The ClassID identifies the object type that is to be read or written. The Object Types are defined in the SEMI Standard E54.1.
InstID Input	MEMBER_ID	This input defines the instance of the object to be read or written. Each SAN_OBJ instance must have a unique identifier (Most identifiers are defined in the SEMI Standard E54.1). **Example:** If a device has four analog values, each analog value is represented by a SAN_OBJ, and each SAN_OBJ has a unique Instance ID.
Member Input 1... 28 Input	Member ID	This is linked to the Member ID Output parameter of a SAN_ATTR Function Block. See *SAN_ATTR EFB, p. 30*.
Active Output	BOOL	This value is TRUE when the object has been accepted by the PLC's internal OMP handler. If the value is FALSE the output parameter 'Status' will contain more specific information.
Status Output	WORD	The Status Output provides a error/status code.
ObjectID	Object_ID	To assign this object to a certain device this parameter needs to be connected to one of the 'Object' parameters of a SAN-DEV EFB.

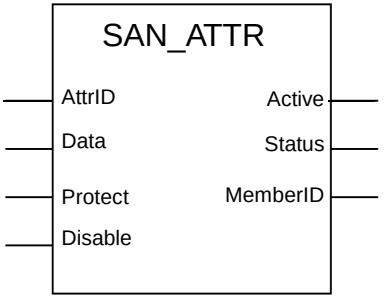
SAN_ATTR EFB

Functional Description

The SAN_ATTR EFB completes the SEMI SAN object model by providing the attribute ID for an object and collecting the actual data from the server PLC.

Representation

The following figure is a block representation of the SAN_ATTR EFB.



Parameter Description

The following table describes the Function Block Parameters.

Parameters	Data Type	Meaning
AttrID Input	UINT	The AttrID input assigns the attribute identifier to the attribute, according to the SEMI standard E54.1.
Data Input	ANY	An IEC variable (located or unlocated) of any primitive data type may be connected to this. (Structured data must use the SAN_STRUCT mechanism EFB). The EFB will then internally discover the byte length of the particular variable. This variable holds the attribute's actual data. The variable's actual data may be read or written by a client using the SAN_READ and SAN_WRIT EFBs. See *Function Blocks SAN_R_INT, SAN_R_UINT, SAN_R_DINT, SAN_R_UDINT, and SAN_R_REAL, p. 17* and *Function Blocks SAN_W_INT, SAN_W_UINT, SAN_W_DINT, SAN_W_UDINT, and SAN_W_REAL, p. 21*.
Protect Input	BOOL	If the SEMI specification defines the attribute to be Read-Only this input must be TRUE. If the attribute is Read-Write, this input must be FALSE or left unconnected.
Disable Input	BOOL	If TRUE the attribute is not visible to the internal OMP handler. If this input is not connected, the OMP handler assumes FALSE and always 'sees' the attribute (and variable).
Active Output	BOOL	This value is TRUE when the attribute has been accepted by the PLC's internal OMP handler. If the value is FALSE the output parameter 'Status' will contain more specific information.
Status Output	WORD	The Status Output provides an error/status code about the device. See *Error/Status Codes for SAN_ATTR EFB, p. 50*.
MemID Output	Member ID	To assign this attribute to a certain object, this parameter needs to be connected to one of the 'Member..' inputs of a SAN_OBJ or SAN_STRUCT, and if applicable SAN_ASMB.

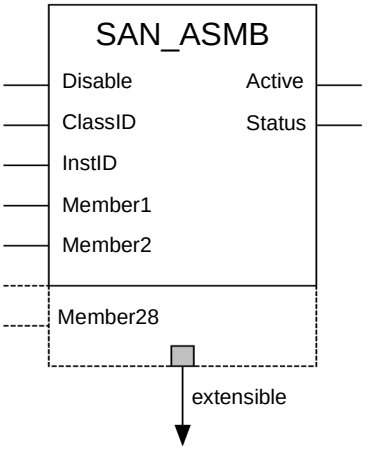
SAN_ASMB EFB

Functional Description

The SAN_ASMB Elementary Function Block (EFB) allows the packing of multiple attributes into one read or write operation. The attributes do not have to come from the same SAN_OBJ. Assemblies are defined within the SEMI specification for a device. The EFB has inputs for class ID and instance ID; the attribute ID of all assemblies is 1 by definition.

Representation

The following figure is a block representation of the SAN_ASMB EFB.



Parameter Description

The following table describes the Function Block Parameters.

Parameters	Data Type	Meaning
Disable Input	BOOL	If TRUE the assembly, and all attached attributes, are not visible to the internal OMP handler. If this input is not connected the OMP handler assumes FALSE and always 'sees' the assembly.
ClassID Input	UINT	This assigns the class identifier to the assembly, according to the SEMI standard E54.1.
InstID Input	UINT	This input defines the instance identifier of the object pointing to a block of data to be read or written. The Instance Identifiers are defined in the SEMI Standard E54.1.
Member1 ... 28 Input	Member ID	This should be linked to the MemID output parameter of a SAN_ATTR EFB or a SAN_STRUCT EFB. See *SAN_ATTR EFB, p. 30*or *SAN_STRUCT EFB, p. 34*.
Active Output	BOOL	This value is TRUE when the assembly has been accepted by the PLC's internal OMP handler. If the value is FALSE the output parameter 'Status' will contain more specific information. See *Error/Status Codes for SAN_ASMB EFB, p. 51*.
Status Output	WORD	The Status Output provides a error/status code. See *Error/Status Codes for SAN_ASMB EFB, p. 51*.

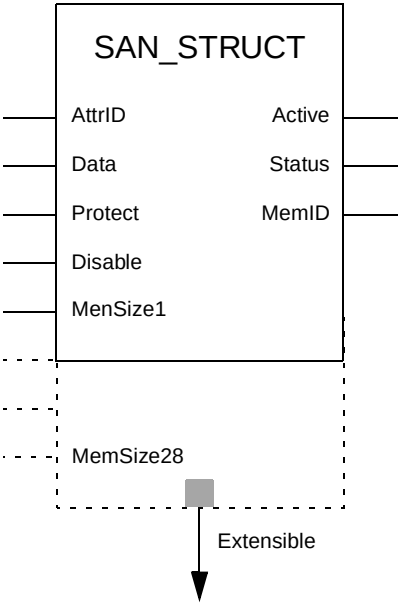
SAN_STRUCT EFB

Functional Description

The SAN_STRUCT Elementary Function Block complements the SAN_ATTR EFB by providing a method of structuring data. It provides the attribute ID for a structured object.

Representation

The following figure is a block representation of the SAN_STRUCT EFB.



Parameter Description

The following table describes the Function Block Parameters

Parameters	Data Type	Meaning
AttrID Input	UINT	The AttrID input assigns the attribute identifier to the structure, according to the SEMI standard E54.1.
Data	ANY	The Data input is of type ANY. Structured data is connected to this pin. The EFB will internally discover the byte length of the structure. The sum of the values in MemSize must equal the byte length of the structure. This variable holds the attribute's actual data. The variable's actual data may be read or written by a client using the SAN_READ and SAN_WRIT EFBs.
Disable Input	BOOL	If TRUE the attribute is not visible to the internal OMP handler. If this input is not connected, the OMP handler assumes FALSE and always 'sees' the attribute (and variable).
Protect Input	BOOL	If the SEMI specification defines the attribute to be Read-Only this input must be TRUE. If the attribute is Read-Write, this input must be FALSE or left unconnected.
MemSize1 Input	Byte	Each MemSize input defines the size of an element of the structure connected to the Data pin. MemSize1 is for the first element defined in the .DTY file. The total of all MemSize pins must equal the byte length of the structure connected to the Data pin. A zero is an invalid entry.
Active Output	BOOL	This value is TRUE when the attribute has been accepted by the PLC's internal OMP handler. If the value is FALSE the output parameter 'Status' will contain more specific information.
Status Output	WORD	This delivers an error code. See *Error/Status Codes for SAN_STRUCT EFB, p. 52*.
MemID Output	MEMBER_ID	To assign this attribute to a certain object, this parameter needs to be connected to one of the 'Member..' inputs of a SAN_OBJ or SAN_ASMB.

SAN_SMAP EFB

Functional Description

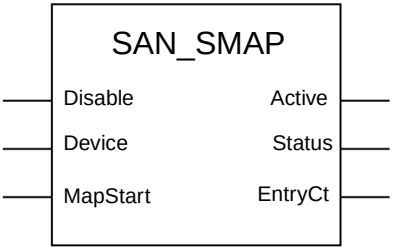
The SAN_SMAP Elementary Function Block (EFB) allows the entire Sensor/ Actuator Network (SAN) device setup table to be put into 4x registers in state RAM of the M1E processor. This method of configuring the SAN device setup allows the downloading of different setups without a programming change to the M1E's logic program. The name SMAP relates to StateRAM-MAP because the whole device setup table (MAP) is stored in state RAM (4x).
The SAN_SMAP EFB identifies to the OMP handler the start of a contiguous area of 4X register memory. This memory will contain a set of structures identifying the class, instance, and attribute IDs of all SAN attributes in use.
The SAN_SMAP is used as an alternate to using the SAN_DEV, SAN_OBJ, SAN_ATTR approach. Only one approach or the other should be used. The data for a SAN_SMAP can be created by a third party configuration tool and written to this 4x memory.

Note: This device's setup table must be rescanned after a power cycle or the download of a new application

Note: The scan bit (bit 0) of the Control Code (first register) must be held TRUE until the SMAP EFB indicates the table has been successfully scanned by the OMP handler.

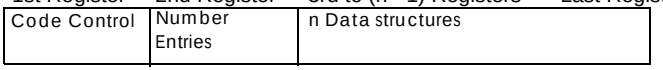
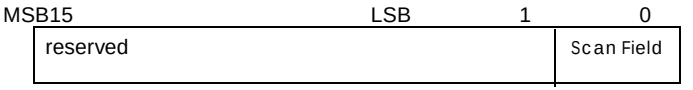
Note: The SAN_SMAP uses state RAM instead of IEC program space. To use the SAN_SMAP function block, the user may have to increase the size of the program data space in the M1E processor. The default setting is 16386 bytes.

Representation The following figure is a block representation of the SAN_SMAP EFB.



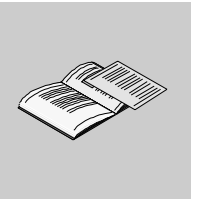
Parameter Description The following table describes the Function Block Parameters

Parameters	Data Type	Meaning
Disable Input	BOOL	If TRUE then the device, and all attached objects, are not visible to the internal OMP handler. If this input is not connected, the OMP handler assumes FALSE and always 'sees' the device.
Device Input	BYTE	The Device input assigns a logical device number to the SAN_DEV for use when the PLC is hosting multiple SAN_DEV objects. The same value also corresponds to the Unit-ID of the Modbus/TCP protocol, so that multiple devices on the same PLC could be addressed with different Unit-Ids in the Modbus/TCP message frame, according to the OMP standard of Modbus/TCP. If the PLC will host only one device this pin may be left open to automatically assign the value 0 as the device's unit number.

Parameters	Data Type	Meaning
MapStart Input	UINT	This value tells the EFB where the SAN device setup table starts in the 4x register area. If MapStart contains 1 the EFB would expect the device setup to start at 4x00001. The address is the first register of a contiguous block of memory. The data elements within this area represent a 'scattering' of memory locations.  The first register contains a control code. The Control Code register is a bit mask that tells the OMP handler when to scan the map for data, as follows:  If the Scan flag is set, the SAN_SMAP EFB will scan the table and publish the contents to the PLC's internal OMP handler. **Note**: This scanning will take place every scan cycle as long as this bit is set. To save processing time, set this bit to zero when changes will no longer be made. **Note**: The device setup table has to be re-scanned after a power-cycle (unless battery backup is available) or the download of a new application. The second register in the device setup contains the number of entries (attribute descriptors) that will start in the third register. Every entry in the device setup has the following structure: Class (UINT) : class identifier Instance (UINT) : instance identifier Attribute (UINT) : attribute identifier Settings (UINT) : characteristics bit mask bit 0 = 0 : read/write access bit 0 = 1 : read only access NumEntries (UINT) : Number of attributes Length_1 (UINT) : length of attribute in bytes Offset_1 (UINT) : 4x offset of attribute Length_n (UINT) : length of last attribute Offset_n (UINT) : 4x offset of last attribute If the NumEntries is 1, the entry is treated as an attribute, if it is greater than one it is treated as an assembly or structure, if it is 0 there is an error and the entire SAN_SMAP is invalid. If an attribute is a structure, each member must be entered as a primitive type, not one entry for the entire structure. The second entry in the device setup table will immediately follow the first one and so on.
Active Output	BOOL	This value is TRUE when the SAN_SMAP has been accepted by the PLC's internal OMP handler. If the value is FALSE the output parameter 'Status' will contain more specific information.

Parameters	Data Type	Meaning
Status Output	WORD	The Status Output provides a error/status code*Error/Status Codes for SAN_SMAP EFB, p. 53*.
EntryCT Output	UINT	The EntryCt Output indicates the number of entries of the device setup table that were scanned. If an error occurs during the scan, the EntryCT output indicates which entry caused the error.

Appendices



At A Glance

Overview

The following contains detailed information on the Error/Status Codes applicable to the SEMI SAN Function Block Library..

What's in this Appendix?

The appendix contains the following chapters:

Error/Status Codes

A

At a Glance

Overview

The following contains detailed information on the Error/Status codes applicable to the SEMI SAN Function Block Library. Codes for EFBs, TCP, and MODBUS are included.

What's in this Chapter?

This chapter contains the following topics:

Error/Status Codes for SAN_READ, SAN_R_xx EFBs

Error Code Table The following table describes the error codes for the SAN_READ, SAN_R_xx EFBs.

Status (Hex)	(Dec)	Description
0x0000	0	Either success or 'Start' is not set
0x80	32768+	OMP specific response error, refer to OMP-Modbus/TCP specification for details. (reproduced below)
0x81xx		The response contained a Modbus error code. The error code is anded with 8100 hex. (see below for codes)
0x8801	34817	The EFB could not gain access to the ethernet. All OMP paths have been taken by other EFBs. Add a TON delay to the start if synchronized.
0x8803	34819	The EFB could not gain access to the ethernet. This PLC does not implement the OMP handler.
0x8804	34820	The EFB could not connect to the assigned IP address, check for valid address
0x8805	34821	The EFB could not send to the assigned IP address.
0x8806	34822	Fatal error: The EFB could not gain the OMP handler
0x8807	34823	Fatal error: The EFB could not gain access to the OMP handler
0x8808	34824	The response sent back was not Modbus/TCP compatible (MB length < 3 byte)
0x8809	34825	SEMI SAN is not supported by the remote host.
0x880A	34826	The response sent back was not compliant to the SEMI SAN specification (fragment < 9 byte)
0x880B	34827	The response sent back used message fragmentation which is not supported by this PLC. This is also generated if the CIA pin is not hooked up.
0x880C	34828	The response contained the wrong service code(0x08 was expected)
0x880D	34829	The received attribute is too long to fit into the variable connected to pin 'Data'. Look at the parameter 'Length' to see how big the received attribute actually is.
0x880E	34830	The response sent back was too small (<= 9 bytes)
0x880F	34831	The data size was not equal to the specific EFB data type. The wrong EFB was used.
0x8811	34833	The connection was interrupted or reset by the OMP server
0x8812	34834	The connection was closed for unknown reason.

Status (Hex)	(Dec)	Description
0x8901	35073	The EFB was aborted by the user (via the Abort pin)
0x8902	35074	The user-set timeout occurred during connection.
0x8903	35075	The user-set timeout occurred after connection and before completion.

Error Codes for SAN_WRIT, SAN_W_xx EFBs

Error Code Table The following table describes the error codes for the SAN_WRIT, SAN_W_xx EFBs.

Status (Hex)	(Dec)	Description
0x0000	0	Either success or 'Start' is not set
0x80xx	32768+	OMP specific response error, refer to OMP-Modbus/TCP specification for further detail. (reproduced after SAN_READ)
0x81xx		The response contained a Modbus error code. The error code is anded with 8100 hex. (see below for codes)
0x8801	34817	The EFB could not gain access to the ethernet. All available OMP paths have been taken by other EFBs. Add a TON delay to the start if synchronized.
0x8803	34819	The EFB could not gain access to the ethernet. This PLC does not implement the OMP handler.
0x8804	34820	The EFB could not connect to the assigned IP address check for valid address.
0x8805+	34821	The EFB could not send to the assigned IP address.
0x8806	34822	Fatal error: The EFB could not gain access to the OMP handler
0x8807	34823	Fatal error: The EFB could not gain access the OMP handler
0x8808	34824	The response sent back was not even Modbus/TCP compatible (MB length < 3 byte)
0x8809	34825	The SEMI SAN (OMP on Modbus/TCP) is not supported by the remote host.
0x880A	34826	The response sent back was not compliant to the SEMI SAN specification (fragment < 9 byte)
0x880B	34827	The response sent back used message fragmentation which is not supported by this PLC. This is also generated if the CIA pin is not hooked up.
0x880C	34828	The response contained the wrong service code (0x08 was expected)
0x880E	34830	The response sent back is too small (<= 9 bytes)
0x880F	34831	The data block to be sent is too large, since message fragmentation is not supported by this PLC. Make sure the 'Length' parameter is smaller than 189 and also smaller or equal the size of the variable connected to 'Data'
0x8811	34833	The connection was interrupted or reset by the OMP server
0x8812	34834	The connection was closed for unknown reason.
0x8813	34835	The data length is set to 0

Status (Hex)	(Dec)	Description
0x8901	35073	The EFB was aborted by the user (via the Abort pin)
0x8902	35074	The user-set timeout occurred, via Timeout input.

Error/Status Codes for SAN_DEV EFB

Error Code Table

The following table descibes the error codes for the SAN_DEV EFB.

Status (Hex)	(Dec)	Description
0x4000	16384	Inactive, parameter 'Disable' has been set
0x4001	16385	Device has been published and accepted by the PLC's internal OMP handler
0x8001	32769	Device could not be published to the OMP handler. Check whether the PLC supports OMP.
0x8002	32770	PLC's internal OMP handler is filled, no more Devices (max. is 5 concurrent devices)

Error/Status Codes for SAN_OBJ EFB

Error Code Table

The following table descibes the error codes for the SAN_OBJ EFB.

Status (Hex)	(Dec)	Description
0x4000	16384	Inactive, parameter 'Disable' has been set
0x4002	16386	Object has been published and accepted by the PLC's internal OMP handler
0x8002	32770	Object could not be linked to the device. Make sure the ObjectID output is connected to one of the Objectxx inputs of the device EFB. Also SAN_DEV is disabled or OMP not installed

Error/Status Codes for SAN_ATTR EFB

Error Code Table The following table descibes the error codes for the SAN_ATTR EFB.

Status (Hex)	(Dec)	Description
0x4000	16384	Inactive, parameter 'Disable' has been set
0x4003	16387	Attribute has been published and accepted by the PLC's internal OMP handler
0x4111	16667	The attribute's link process is still ongoing. This should happen for a maximum of one scan only.
0x8003	32771	Attribute could not be linked to the device.Make sure that the MemID Output is connected to a Memberxx Input of an object EFB SAN_OBJ. All higher objects must be enabled. OMP not implemented.

Error/Status Codes for SAN_ASMB EFB

Error Code Table

The following table descibes the error codes for the SAN_ASMB EFB.

Status (Hex)	(Dec)	Description
0x4000	16384	Inactive, parameter 'Disable' has been set
0x4004	16988	Assembly object was published and accepted by the PLC's internal OMP handler.
0x4111	16667	Assembly object's link process is still ongoing. This should happen for a maximum of one scan only.
0x8004	92772	Assembly object could not be linked to the device. Make sure the associated instances of SAN_ATTR, SAN_OBJ, and SAN__DEV are active and linked properly. OMP not implemented.

Error/Status Codes for SAN_STRUCT EFB

Error Code Table

The following table descibes the error codes for the SAN_STRUCT EFB.

Status (Hex)	(Dec)	Description
0x4000	16384	Inactive, parameter 'Disable' has been set
0x4111	16667	The Structure's link process is still ongoing. This should happen for a maximum of one scan only.
0x8003	32771	Structure could not be linked to the device. Make sure that the MemID Output is connected to a Memberxx Input of an object EFB SAN_OBJ. All higher objects must be enabled. OMP not implemented.
0x8006	32772	One of the MemSize data pins was empty, before the last used pin
0x8007	32773	One of the MemSize data pins contained a zero size.
0x8008	32774	The size of the ANY input was not equal to the sum of MemSize values

Error/Status Codes for SAN_SMAP EFB

Error Code Table The following table describes the error codes for the SAN_SMAP EFB.

Status (Hex)	(Dec)	Description
0x4000	16384	Inactive, input 'Disable' is set
0x4001	16385	Device has been published to the internal OMP handler with all configured entries.
0xA001	40962	The EFB could not gain access to the PLC's internal Object Messaging Protocol handler (OMP). Shouldn't happen.
0xA002	40963	The EFB could not gain access to the PLC's internal Object Messaging Protocol handler (OMP). This PLC does not have any internal OMP handler.
0xA003	40964	Fatal: EFB lost access to PLC's internal OMP handler.
0xA004	40965	First bit of control register was not set on the very first scan for the SAN_EFB, so that no device could be published to the internal OMP handler
0xA006		Scan bit of Control register is not set, normal operation
0xB001	45057	Noentries would fit into the 4x registers starting at MapStart
0xB002	45058	MapStart contains 0, which is invalid.
0xB003	45059	Not enough 4x registers left to store the configured amount of entries.
0xC001	49153	Not enough registers for attributes, look at Output 'EntryCt' to see which entry caused the Problem.
0xC004	49156	insufficient internal memory, use one more instance of SAN_SMAP to cover the rest of the device setup table. The output 'EntryCt' tells which entry did not fit.
0xC008	49160	Fatal: Previous created object could not be found. (Should not happen.)
0xC009	49161	A Length entry is 0. The output 'EntryCt' tells which entry did not fit.
0xC00A	49162	NumEntries member of SMAP is zero, it must be at least one.

TCP Error Codes for Read and Write EFBs

Error Code Table

The following table describes the error codes for the SEMI_SAN TCP.

Status (Hex)	(Dec)	Description
0x8001	32769	Invalid Service code. Can also be generated if connected with wrong IP address or wrong device.
0x8002	32770	Invalid service code Parameter
0x8003	32771	Invalid attribute
0x8004	32772	Attribute out of range
0x8005	32773	Not valid in this range
0x8006	32774	Fragmentation error
0x8007	32775	Fragments from multiple messages
0x80FF	33023	Unspecified error

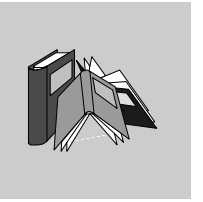
MODBUS Error Codes for Read and Write EFBs

Error Code Table

The following table describes the error codes for MODBUS.

Status (Hex)	(Dec)	Description
0x8101	33025	Illegal Function, bad Modbus function code. This code will also be returned if the OMP server PLC is not running, or does not support SEMI SAN. This code is also returned by an attempt to write to a read-only attribute.
0x8102	33026	Illegal data address
0x8103	33027	Illegal data value
0x8104	33028	Illegal response length
0x8105	33029	Acknowledge, for use in programming commands
0x8106	33030	Slave device busy or OMP Server had no more paths available, try again (Possibly too many client EFB's are triggered simultaneously).
0x8107	33031	Negative Acknowledge, for use in programming commands
0x8108	33032	Memory parity error
0x810A	33034	Gateway path not available
0x810B	33035	Gateway target device failed to respond

Glossary



A

attribute An attribute is a common component for all objects. It represents a certain value or data related to the object.

C

client The client sends and receives information to and from the device. The send and receive processes are encapsulated as services.
A client communicating on a Sensor/Actuator network always issues requests to one or more server(s) on the same network. A request consists of data that identifies the device's object and the object's services. Some services require additional parameters, as part of the request. The client should check the required response for successful completion.

D

device In this application, a device is automation equipment used for measurement and control specific to the manufacturing process of semiconductor components.

I

IP Address — Internet Protocol Address; The unique unit or node 32-bit address assigned to a device connected to the Ethernet Communication Network, Internet, or an intranet.

O

object — An object provides either internal services to the device, or the inputting and outputting of data.
There are several types of objects, such as the following:
- The objects that perform internal management functions include:
  - Sensor/Actuator/Controller Object (SAC) that coordinates the interaction among all of the Active Element Objects
  - Device Manager Object (DM) that performs overall device management
- The objects that provide process oriented Attributes and Services are called Active Element (AE) objects and include the following:
  - Sensor Objects, which provide data such as Values, Report Inhibit Timer, and Report Rate output from the process measurement
  - Actuator Objects, which receive data (attribute) input from the M1E processor. The data includes Settings, Safe State, WatchDog Timer
  - Controller Objects, which include Data Types, Data Units, Setpoint, Control Variable, and Process Variable

object messaging protocol (OMP) handler — The OMP is the firmware in the M1E processor that supports SEMI SAN communications.

object-oriented — Small pieces of data and programming code encapsulated for use in a program, and carrying properties that conform to a standard.

object-oriented software — A software language that operates on objects versus traditional text-based programs. Each object is a portable software module that can be linked with other objects to form an application program.

P

protocol The rules for standardized formats for the data transfer, data packets, and network management tools used to communicate with devices on a network. Ethernet can support multiple protocols depending on the types of devices and the type of information that is communicated.

S

Sensor/Actuator Network (SAN) A common device networking methodology at the user/application layer. Its main focus is devices. The SAN is independent of any particular network protocol layer. With the Momentum M1E processor, Schneider's Modbus/TCP protocol layer is used for the Sensor/Actuator Network.

server The server builds the non-standardized link between the Sensor/Actuator network (standardized) and the actual process-connected device. This makes the actual device appear to the network as an aggregation of objects.
A server on a Sensor/Actuator network hosts one or more devices from a communications point of view. The server represents at least the SAN gateway to the device which the client wants to interrogate. The server may or may not also host the actual device. After receiving a request (for a service) from a client, it checks whether the object and the object's service exist If so, it passes the request and parameters to that object. This process is purely internal and may be implemented differently with any server-device combination. Every request will generate a response.