NOTICE

This document contains information about one or more ABB products and may include a description of or a reference to one or more standards that may be generally relevant to the ABB products. The presence of any such description of a standard or reference to a standard is not a representation that all of the ABB products referenced in this document support all of the features of the described or referenced standard. In order to determine the specific features supported by a particular ABB product, the reader should consult the product specifications for the particular ABB product.

ABB may have one or more patents or pending patent applications protecting the intellectual property in the ABB products described in this document.

The information in this document is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from ABB, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license. This product meets the requirements specified in EMC Directive 2004/108/EC and in Low Voltage Directive 2006/95/EC.

TRADEMARKS

All rights to copyrights, registered trademarks, and trademarks reside with their respective owners.

Copyright © 2003-2014 by ABB.
All rights reserved.

Release: August 2014
Document number: 3BDS013988-600
# Table of Contents

## About This User Manual

General ..............................................................................................................................9

User Manual Conventions ...............................................................................................10
  Feature Pack ..................................................................................................................10
  Warning, Caution, Information, and Tip Icons ............................................................10

Terminology.....................................................................................................................11

Released User Manuals ...................................................................................................15

## Section 1 - Introduction

Product Overview ............................................................................................................19
  Product Scope..................................................................................................................22
  Communication Overview .............................................................................................22

Operating Overview ........................................................................................................24
  Start-up ..........................................................................................................................24
  Runtime .........................................................................................................................24
  Shutdown .......................................................................................................................25
  Mutual lifesign supervision ............................................................................................26
  Bus Cable Disconnect .................................................................................................26
  Basic Functions .............................................................................................................27

## Section 2 - Configuration

Initial Configuration .........................................................................................................33

Considerations ..................................................................................................................41
  Sending Commands and Subscribing for Values .......................................................42
  Capacity and Performance ............................................................................................47
  Compatibility ..................................................................................................................53
# Table of Contents

Multiple and Redundant OPC Servers ................................................................. 53  
Startup of Redundant OPC Servers ................................................................. 60  
Online Changes ............................................................................................. 60  

Customizing the OPC Server .......................................................................... 61  
   General ........................................................................................................ 61  

Working With AC 100 OPC Data ..................................................................... 62  
   General ........................................................................................................ 62  
   Connecting to a Remote AC 100 OPC Server ............................................ 62  
   How to View the Diagnostics ................................................................. 63  
   How to View System Status ................................................................. 65  
   How to View System Messages ............................................................ 67  

Working with the Alarms and Events ............................................................... 68  
   General ........................................................................................................ 68  
   Alarm and Event Texts ............................................................................ 70  
   Alarm and Event Severity ...................................................................... 72  
   Distribution of Acknowledge .................................................................. 73  

Working with Control Builder A (CBA) ........................................................... 73  
Tutorials ......................................................................................................... 74  
   Creating a Small Test Configuration Using Extended DB Elements ........ 74  
   Creating a Small Test Configuration Using DAT Based Objects ............. 104  

# Section 3 - Maintenance  
Preventive Maintenance ............................................................................... 123  
Error Messages .............................................................................................. 123  
Fault Finding and User Repair ....................................................................... 124  
   General ........................................................................................................ 124  
   Check ABB Services, CI Device Driver and Event Viewer ...................... 124  
   Hints on Known Failure Situations .......................................................... 128  

# Appendix A - Object Attributes and Events  
General .......................................................................................................... 129  
   Attribute Name Mapping ....................................................................... 130  
   Event Quality ........................................................................................... 131  

---

---
Table of Contents

AC 100 Event Categories ........................................................................................................132
Analog Input Object Type (AIS and DAT_AI) .................................................................133
    Attributes .......................................................................................................................133
    Events .........................................................................................................................138
Analog Output Object Type (AOS and DAT_AO) ..............................................................140
    Attributes .......................................................................................................................140
    Events .........................................................................................................................144
Digital Input Object Type (DIS and DAT_DI) .................................................................146
    Attributes .......................................................................................................................146
    Events .........................................................................................................................149
Digital Output Object Type (DOS and DAT_DO) ..............................................................149
    Attributes .......................................................................................................................149
    Events .........................................................................................................................153
Simple Memory Object Types (DAT(x), MB, MI, MIL, MR) ........................................153
    Attributes .......................................................................................................................153
    Events .........................................................................................................................155
Simple Memory Object Type (MBS) ..............................................................................157
    Attributes .......................................................................................................................157
    Events .........................................................................................................................158
System Status Object Types (PM, PM_SUM, CI, CI_SUM, I/O, OPC) .......................160
    Processor Module (PM) Attributes ...........................................................................160
    Processor Module (PM) Events ................................................................................163
    AC 100 Processor Module Summary Status (PM_SUM) Attributes .............166
    Communication Module (CI) Attributes .................................................................168
    Communication Module (CI) Events ...................................................................171
    AC 100 AF100 CI Module Summary Status (CI_SUM) Attributes ...............173
    AC 100 AF100 CI Module Summary Status (CI_SUM) Events .......................174
    AC 100 IO CI Summary Status (IO) Attributes .......................................................175
    AC 100 IO CI Summary Status (IO) Events ............................................................176
    OPC Station (OPC) Attributes ...............................................................................178
    OPC Station (OPC) Events .......................................................................................180

Appendix B - Setting up OPC Server to send DSPs
# Table of Contents

Configure DSP DB Element ................................................................. 184
Configure AF 100 OPC Station ............................................................. 185

**Appendix C - Extended Database Elements**
Overview ............................................................................................ 187
Extended DB Element Communication Between Stations .................... 189
Event Handling with Extended DB Elements ....................................... 190

**Appendix D - Installing the PCI Board CI527A**

**Appendix E - Product Verification**
About This User Manual

General

This user manual is intended for a user that is responsible for:

• Configuration of the AC 100 OPC Server
• Configuration and building of applications for controllers of the AC 100 Series utilizing the AC 100 OPC Server
• Runtime operation and maintenance of the AC 100 OPC Server.

The predecessor product of AC 100 OPC Server was named Advant OPC Server for Advant Fieldbus 100.

AC 100 OPC Server is a standard OPC Server for use with any standard OPC client.

For use of AC 100 OPC Server within Extended Automation System 800xA, a complete connectivity package 800xA for AC 100, including this OPC Server, is available.

Section 1, Introduction gives you an overview on the product and its operation.
Section 2, Configuration describes configuration considerations and procedures.
Section 3, Maintenance gives maintenance hints.
Appendix A, Object Attributes and Events lists object attributes and events in detail.
Appendix B, Setting up OPC Server to send DSPs outlines how to configure DSP sending.
Appendix C, Extended Database Elements gives an overview on the handling of Extended Database Elements.
The following subsections of this section are a guide to the conventions and the terminology used throughout this book. They further contain a list of related documentation.

User Manual Conventions

Microsoft Windows conventions are normally used for the standard presentation of material when entering text, key sequences, prompts, messages, menu items, screen elements, etc.

Feature Pack

The Feature Pack content (including text, tables, and figures) included in this User Manual is distinguished from the existing content using the following two separators:

Feature Pack Functionality ______________________________________________________________________

<Feature Pack Content>

___________________________________________________________________________________________

Feature Pack functionality included in an existing table is indicated using a table footnote (*):

*Feature Pack Functionality

Feature Pack functionality in an existing figure is indicated using callouts.

Unless noted, all other information in this User Manual applies to 800xA Systems with or without a Feature Pack installed.

Warning, Caution, Information, and Tip Icons

This User Manual includes Warning, Caution, and Information where appropriate to point out safety related or other important information. It also includes Tip to point
out useful hints to the reader. The corresponding symbols should be interpreted as follows:

- **Electrical warning icon** indicates the presence of a hazard which could result in *electrical shock*.

- **Warning icon** indicates the presence of a hazard which could result in *personal injury*.

- **Caution icon** indicates important information or warning related to the concept discussed in the text. It might indicate the presence of a hazard which could result in *corruption of software or damage to equipment/property*.

- **Information icon** alerts the reader to pertinent facts and conditions.

- **Tip icon** indicates advice on, for example, how to design your project or how to use a certain function.

Although Warning hazards are related to personal injury, and Caution hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, fully comply with all Warning and Caution notices.

### Terminology

A complete and comprehensive list of terms is included in *System 800xA System Guide Functional Description (3BSE038018*)*. The listing includes terms and definitions that apply to the 800xA System where the usage is different from commonly accepted industry standard definitions and definitions given in standard dictionaries such as Webster’s Dictionary of Computer Terms. Terms that uniquely apply to this User Manual are listed in the following table.
<table>
<thead>
<tr>
<th>Term/Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Advant Controller, the common term for the series of controllers used in ABB automation systems, for example AC 160, AC 110.</td>
</tr>
<tr>
<td>AC 100 Series</td>
<td>Term stands for AC 70, AC 110 and AC 160. Derivatives AC 80 and APC 2 exist.</td>
</tr>
<tr>
<td>ACC</td>
<td>AMPL Control Configuration. See CBA below.</td>
</tr>
<tr>
<td>Advant® OCS</td>
<td>The term refers to one or more of the open control systems from ABB, i.e., ABB Advant, ABB Master and ABB MOD 300.</td>
</tr>
<tr>
<td>AF100</td>
<td>Advant Fieldbus 100. AF100 is a high speed communication link intended for communication between controllers in the AC 400 and AC 100 series and other equipment adapted for the bus. The transmission rate is 1.5 Mbit/s and it supports three transmission media: twisted pair, coaxial and optical media.</td>
</tr>
<tr>
<td>AMPL</td>
<td>ABB Master® Programming Language. A function-block language with graphic representation which is especially developed for process control applications.</td>
</tr>
<tr>
<td>APB</td>
<td>Application Builder. The software module in CBA used to handle projects, nodes, circuits and type circuits.</td>
</tr>
<tr>
<td>APC</td>
<td>Application Controller, a control unit used for motor drives, and so on.</td>
</tr>
<tr>
<td>BCB</td>
<td>Bus Configuration Builder. The software module in CBA used to create and maintain the Bus Configuration Database.</td>
</tr>
<tr>
<td>BCD</td>
<td>Bus Configuration Database contains configuration data of nodes and busses on the AF100 network belonging to a project. The BCD is created and maintained with the CBA software and read by the OPC Server at start-up. File suffix is MDB.</td>
</tr>
</tbody>
</table>
## Terminology

<table>
<thead>
<tr>
<th>Term/Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCI</td>
<td>Bus Configuration Interface. Internal interface to the Bus Configuration Database for the AC 100 OPC Server. Implemented according to Microsoft’s Component Object Model (COM).</td>
</tr>
<tr>
<td>Board</td>
<td>Hardware device, in the context of AC 100 Series often the term module is used instead.</td>
</tr>
<tr>
<td>Bus Master</td>
<td>A bus master is a device which administrates the information flow on the Advant Fieldbus 100</td>
</tr>
<tr>
<td>CBA</td>
<td>Control Builder A. Application building software for Advant Controller 100 and 400 series. Replaces the AMPL Control Configuration (ACC) software. CBA includes Application Builder (APB), Function Chart Builder (FCB), and Bus Configuration Builder (BCB).</td>
</tr>
<tr>
<td>CI</td>
<td>Communication Interface</td>
</tr>
<tr>
<td>CI Board</td>
<td>A CI board is a Communication Interface Board, used to connect a PC to the Advant Fieldbus 100 bus.</td>
</tr>
<tr>
<td>Client Application</td>
<td>A client application is an application which subscribes for data and/or sends commands to objects in the automation system.</td>
</tr>
<tr>
<td>Clock Master</td>
<td>A clock master is a device which distributes time on the bus to synchronize clocks in nodes on the Advant Fieldbus 100.</td>
</tr>
<tr>
<td>COM</td>
<td>Component Object Model, a specification that defines how individual software components can interact and share data under Windows. Developed by Microsoft.</td>
</tr>
<tr>
<td>DAT Based Objects</td>
<td>Object types with, compared with the Extended DB Elements, limited object support.</td>
</tr>
<tr>
<td>DB element</td>
<td>DB element is an abbreviation for a data base element, which is part of a process control application. DB elements represent e.g. hardware boards (PM645...) or signals (AIS,DSP...).</td>
</tr>
<tr>
<td>Term/Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DCOM</td>
<td>Distributed COM. Extends COM to networks.</td>
</tr>
<tr>
<td>DSP</td>
<td>DataSet Peripheral, a block of data to be transmitted on Advant Fieldbus 100. In the controller the DataSet Peripheral is represented with a database element.</td>
</tr>
<tr>
<td>ESD</td>
<td>Electrostatic Discharge.</td>
</tr>
<tr>
<td>EVS</td>
<td>Event Sets, the object type for transmitting time tagged events on the Advant Fieldbus 100 network.</td>
</tr>
<tr>
<td>Extended DB Elements</td>
<td>These object types offer complete support for the different object attributes.</td>
</tr>
<tr>
<td>FCB</td>
<td>Function Chart Builder, the graphic configuration and programming tool in CBA.</td>
</tr>
<tr>
<td>Function Chart</td>
<td>A function chart is a diagram for the representation of process control programs. It contains graphical symbols (rectangles) for PC elements and lines for connections between element terminals.</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interface. See also MMI.</td>
</tr>
<tr>
<td>HSI</td>
<td>Human System Interface. See also MMI.</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode. LEDs are often used as indicators on control panels.</td>
</tr>
<tr>
<td>Object (Process)</td>
<td>Object (Process) is a physical or calculated process object, or a functional unit or type circuit containing all related inputs and outputs. An example of a process object is an analog input including value, limits, status, etc. There are two kinds of objects, Extended DB Elements and DAT based objects</td>
</tr>
<tr>
<td>MMI</td>
<td>Man Machine Interface. (Term used by default in this manual. See also HMI, HSI.)</td>
</tr>
<tr>
<td>OLE</td>
<td>Object Linking and Embedding. A technology, based on COM, developed by Microsoft.</td>
</tr>
<tr>
<td>OPC</td>
<td>OLE for Process Control.</td>
</tr>
</tbody>
</table>
Table 2 lists the documentation related to AC 100 OPC Server Configuration and Operation. A Release Note in .pdf format is included on the delivered installation media.

Table 2. Related Documentation

<table>
<thead>
<tr>
<th>Category</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Advant Fieldbus 100 User´s Guide (3BSE000506*)</td>
<td>Describes how to use the AF100 Network. It also provides a technical description of communication boards and modems available.</td>
</tr>
</tbody>
</table>
### Table 2. Related Documentation

<table>
<thead>
<tr>
<th>Category</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation</td>
<td>Advant® OCS, AC 100 OPC Server, Installation (3BSE077523*)</td>
<td>Describes how to install AC 100 OPC Server for third party clients.</td>
</tr>
<tr>
<td></td>
<td>Advant® OCS, Control Builder A Installation (3BDS013836*)</td>
<td>Describes how to install Control Builder A.</td>
</tr>
<tr>
<td>Software</td>
<td>Application Builder User's Guide (3BDS100560*)</td>
<td>Describes installation of Advant Engineering components and contains the operation instruction of Application Builder and Advant Engineering support functions.</td>
</tr>
<tr>
<td></td>
<td>Function Chart Builder User's Guide (3BDS100595*)</td>
<td>Describes how Function Chart Builder can be used to configure different target systems.</td>
</tr>
<tr>
<td></td>
<td>System 800xA, Configuration (3BDS011222*)</td>
<td>Describes configuration of System 800xA.</td>
</tr>
<tr>
<td></td>
<td>System 800xA, Tools (2PAA101888*)</td>
<td>Describes the procedure to access the ABB Start Menu.</td>
</tr>
</tbody>
</table>
Table 2. Related Documentation

<table>
<thead>
<tr>
<th>Category</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>AMPL Configuration Advant Controller 100 Series Ref. Manual (3BSE009626*)</td>
<td>Contains instructions about configuration and application programming, fault tracing and maintenance of AC 160/AC 110/AC 70.</td>
</tr>
<tr>
<td></td>
<td>PC Elements Control Builder A Advant Controller 160 Series Ref. Manual (3BDS005557*)</td>
<td>Contain instructions about configuration and application programming, fault tracing and maintenance of AC 160/AC 110/AC 70.</td>
</tr>
<tr>
<td></td>
<td>PC Elements Control Builder A Advant Controller 110 Series Ref. Manual (3BSE000504*)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PC Elements Control Builder A Advant Controller 70 Series Ref. Manual (3BSE009177*)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Base Elements Advant Controller 160 Reference Manual (3BDS005556*)</td>
<td>Contain instructions and the data sheets for all data base elements available for Advant Controller 160/110/70.</td>
</tr>
<tr>
<td></td>
<td>Data Base Elements Advant Controller 110 Reference Manual (3BDS100594*)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Base Elements Advant Controller 70 Reference Manual (3BDS100593*)</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td><a href="http://www.opcfoundation.org">www.opcfoundation.org</a></td>
<td>Web page for the OPC Foundation, where you can find information about the OPC specifications and links to other sources of information about OPC.</td>
</tr>
</tbody>
</table>
Section 1 Introduction

Product Overview

The AC 100 OPC Server enables bi-directional data exchange between Windows applications and stations on the Advant Fieldbus 100 network. It is configured from the Bus Configuration Database (BCD) created with the Control Builder A (CBA) programming and configuration tools for the AC 100 Series controllers.

Figure 1 visualizes:

- System 800xA, a third party MMI or another OPC client can communicate via AC100 OPC Server with AC 100 Series controllers on the AF 100 bus.
- To establish communication AC 100 OPC Server reads the Bus Configuration Database created from Control Builder A. The OPC Server uses base communication services of AF 100 Interface.
Control Builder A software can communicate with the AC 100 Series Controllers for diagnostic, test and download purposes using the AF 100 Interface directly.

Within the Extended Automation System 800xA is the MMI supported by the complete integration package 800xA for AC 100 (including AC 100 OPC Server).

Figure 1 shows CBA and AC 100 OPC Server installed on different PCs. CBA can also be installed and run on the same PC as the AC 100 OPC Server is. The BCD can reside on a disk of the PC of the AC 100 OPC Server or on a PC accessible from this machine via a network supported by the Windows operating system.

The AC 100 OPC Server can be used by any OPC client application with support for the mandatory Data Access Custom and Automation Interface as well as the Alarms and Events Custom and Automation Interface according to the OPC Foundation standard.

The AC 100 OPC Server is developed based on the OPC Data Access Specification version 2.0 and the OPC Alarms and Events specification version 1.02 delivered by OPC foundation. It is not certified against these specifications.

**OPC Compatibility**

- The OPC Data Access Custom Interface Specification 1.0a
- The OPC Data Access Custom Interface Specification 2.0
- The OPC Data Access Automation Interface Specification 2.0
- The OPC Alarms and Events Custom Interface Specification 1.01
- The OPC Alarms and Events Custom Interface Specification 1.02
- The OPC Alarms and Events Automation Interface Specification 1.01

*Figure 2. OPC Compatibility*

**OPC Data Access Custom Interface**

The OPC Custom Interface is used for high-performance applications written in a compiled language such as C++. 
**OPC Data Access Automation Interface**

The OPC Automation Interface for higher level applications developed in for example Visual Basic for Applications.

**Alarms and Events Custom Interface**

The Alarms and Events Interface is used for broadcasting alarm and event information from servers to clients.

**Alarms and Events Automation Interface**

The OPC Alarms and Events Automation Interface for higher level applications developed in for example Visual Basic for Applications is supported. Sample code for building such applications can be fetched from the OPC foundation Web site [http://www.opcfoundation.org/](http://www.opcfoundation.org/).

**Optional IOPC BrowseServerAddressSpace**

The IOPCBrowseServerAdressSpace provides a way for clients to browse the available data items in the server.

You can find information about the specifications and the work of the OPC Foundation at the OPC Foundation home page: [www.opcfoundation.org](http://www.opcfoundation.org).

**Components**

The following components are needed to run AC 100 OPC Server:

- ABB 800xA Common 3rd party install
- ABB Central Licensing System (CLS)
- AC 100 OPC Server software
- Communication Interface board (CI527A)
- Connection hardware such as modems, cables and connection units.

Below is a conceptual overview of the AC 100 OPC Server and related software.
Product Scope

AC 100 OPC Server is a standard OPC interface for AC100 Series Controls. OPC is a software technology standard used by Windows based applications to access data from process control systems. The basic principle of OPC is that OPC client applications communicate with an OPC server via a standardized, open and therefore vendor independent interface

The AC 100 OPC Server provides OPC client applications, such as Human System Interface (MMI) products or custom-build applications, access to process objects in AC 100 Series Controllers (AC 160, AC 110 and AC 70) and Application Controllers (APC).

The Advant Fieldbus 100 is a high-performance bus specially designed for real-time applications. It features reliable, cyclic data transfer using DataSet peripherals (DSP) as well as event-driven background transfer of service data.

Communication Overview

There are two mechanisms for communication on the Advant Fieldbus 100:

- **DataSet Peripheral (DSP)**
  DSPs are broadcasted cyclically to all stations on the Advant Fieldbus 100 network. The DSP communication is configured and handled by the CI527A board. The AC 100 OPC Server software has functions to read and write data of these DSPs. The DSPs are used to transfer values for process objects on the Advant Fieldbus 100 network.

- **Service Data Protocol (SDP)**
  The SDP is used to transfer messages between applications in different stations on the Advant Fieldbus 100 network. SDP is used for service requests (command) to process objects, and for receiving time tagged events (Event Sets). SDP is also used by CBA when connecting to the Advant Controller 100 Series.
Figure 3. AC 100 OPC Server software conceptual overview
Operating Overview

This section briefly describes how the AC 100 OPC Server operates during start-up, runtime and shutdown.

Start-up

The AC 100 OPC Server is implemented as a Windows service and is automatically started when Windows is started. When AC 100 OPC Server is started:

- The CI527A board is initialized, for example, with station number, bus number.
- The Bus Configuration Database, containing Network configuration and DB elements from all stations, are loaded to the OPC Server.
- If selected, the latest alarm status is fetched from the redundant OPC Server.
- The CI527A board goes operational and process data exchange starts.

Runtime

During runtime, the DATs on the CI board are cyclically updated via DataSet Peripheral (DSP) from the other stations on the AF100 network. The cycle times for the DSP communication can be configured in FCB DB Element Editor.

Cyclic subscriptions of process values are defined by the OPC client application and handled by the AC 100 OPC Server. They have no connection to the DSP cycle time.
When you close the AC 100 OPC Server, the CI board goes to a passive state and the DataSet Peripheral transfer stops.

The CI board also goes to a passive state if the AC 100 OPC Server lifesign stops as described below.

---

1) The cycle time of the PCPGM, Control modules and SCANT of DSPs is defined in the application program of the controller.

*Figure 4. Cyclic DSP Communication*
Mutual lifesign supervision

The AC 100 OPC Server and the CI board provide each other with a lifesign. This mutual lifesign supervision is automatically enabled at start-up of the AC 100 OPC Server. The AC 100 OPC Server lifesign ensures that the CI board goes to a passive state in case of a program halt. The CI board lifesign automatically detects if there is a problem on the board.

Bus Cable Disconnect

If the AF100 bus cable is disconnected the CI board status changes from Operational to Active. The CI board will automatically recognize when the cable is connected again and resume communication, changing its board status back to Operational.
Basic Functions

The AC 100 OPC Server provides the following functionality:

- **Object Handling**
  The AC 100 OPC Server supports two groups of objects providing process data, Extended DB Elements and DAT based objects. The DAT based objects have limited functionality compared to Extended DB Elements, which provides an almost complete object support as for corresponding object types in the Advant Controller 400 series. For these groups of objects the AC 100 OPC Server provides the following object handling functions:

  - **Data Subscription**
    Values for process objects are received from the AF100 network using cyclic DataSet Peripheral (DSP).

    Subscription of other object attributes such as object description and engineering unit is also provided. These attributes are not transferred via the AF100 network, but retrieved from the BCD (Bus Configuration Database), which is created by CBA's Application Builder and populated by Function Chart Builder and Bus Configuration Builder.
Service requests

Service requests (commands) can be sent from the OPC client to change object attributes on the Advant Fieldbus 100 network. For DAT based objects the attribute Value can be changed in the controller and the Acknowledge operation is processed by the OPC Server itself. For Extended DB Elements, the attributes Value, Block, Deblock and Acknowledge are examples of attributes in the controller that can be changed. See also Appendix A, Object Attributes and Events.

Figure 5. Data subscription for an AI-signal using Extended Data Base Elements
Section 1  Introduction  

Basic Functions

Figure 6. Service request for an AI-signal using Extended DB Elements

- **Event Handling**

  The AC 100 OPC Server can be configured to receive time tagged events, via Event Set (EVS), from the AF100 network. The time tagged events contain object information such as the reason for the event and a time stamp. This information, together with user-defined object information such as object description, engineering unit, etc., can be used by the client application to generate events and alarms.

  For DAT based objects, alarm acknowledgment by the OPC client can be distributed on the AF100 network to other PCs equipped with the AC 100 OPC Server.
The table below shows the services which are available for the object type.

**Table 3. AC 100 OPC Server Services**

<table>
<thead>
<tr>
<th>Object type</th>
<th>Demand(^{(4)})</th>
<th>Cyclic</th>
<th>Event</th>
<th>Command(^{(5)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAT_Ai(^{(1)})/Ais(^{(2)})</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DAT_Ao(^{(1)})/Aos(^{(2)})</td>
<td>yes</td>
<td>yes</td>
<td>no / yes</td>
<td>yes</td>
</tr>
<tr>
<td>DAT_Di(^{(1)})/Dis(^{(2)})</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DAT_Do(^{(1)})/Dos(^{(2)})</td>
<td>yes</td>
<td>yes</td>
<td>no / yes</td>
<td>yes</td>
</tr>
<tr>
<td>DAT_Dat(^{(1)})</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>MB(^{(3)})</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MBS(^{(3)})</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MR, MI and MIL(^{(3)})</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

(1) These are DAT Based Elements object types. They are mapped on DAT's and transferred with DataSet Peripherals (DSP) for subscription and Service Data Protocol (SDP) for commands and events.

(2) These objects are the Extended DB version of the AI, AO, DI and DO objects. See Appendix A, Object Attributes and Events.

(3) These objects belong to the Extended DB Elements object group but function similar to DAT objects.

(4) SDP read

(5) SDP write
• **System Status**  
The AC 100 OPC Server supports access to status information via status attributes in objects representing:

– Extended DB Elements and DAT based elements
– Communication Interfaces (CI), Processor Modules (PM) and the OPC Server (OPC)
– Aggregated status information of CI boards and I/O boards (IO), of CI boards (CI_SUM) and of Processor Modules (PM_SUM).

The objects of type CI, PM, OPC, IO, CI_SUM, PM_SUM are called system status objects.

System status information for all objects (and any other information exposed via these objects) in the nodes reachable by the AC 100 OPC Server can be displayed using a third party OPC viewer. See How to View System Status on page 65.

• **Alarms and Events Interface**  
Alarms and events are distributed on the AF100 bus and are accessible according to the OPC Foundation standard (see Figure 2) for Alarms and Events Interface.

• **Time Synchronization**  
The AC 100 OPC Server can act as clock master, slave or backup node on the AF100 bus. All Personal Computers running AC 100 OPC Server in an AF 100 network must be configured to the same time zone and must be time synchronized.

• **Hardware Configuration and Diagnostics**  
It is possible to view and change the configuration parameters for the CI527A communication board.

The Mutual Lifesign Supervision function in AC 100 OPC Server supervises the CI527A board and automatically tries to restart it in case of board failure. If the AC 100 OPC Server has an uncontrolled stop the CI527A board goes to passive state and the DSP transmission stops.
• **Support for Control Builder A (CBA)**
  All object configuration are made in the CBA. The configuration data are stored in the Bus Configuration Database (BCD) where it is accessible by the AC 100 OPC Server via the Bus Configuration Interface (an internal interface, installed by Control Builder A or manually).
  The AF100 Interface (included in the AC 100 OPC Server), used together with the Function Chart Builder (FCB), provides remote engineering of nodes on the AF100 bus.

• **Online Changes**
  AC 100 OPC Server recognizes and processes online changes in the BCD.

• **Application Messages**
  Application messages of the AC 100 OPC Server are presented in the Windows Event Viewer as Application Events.

• **System Messages**
  System messages caused by the AC 100 OPC Server and the CI board drivers are presented in the Windows Event Viewer as System Events.

• **Support for Multiple and Redundant OPC Servers.**
  Alarm acknowledge information for redundant and multiple OPC servers can be distributed. Alarm & Event synchronization for redundant OPC servers is performed at start-up.
Section 2  Configuration

Initial Configuration

Configuration Wizard

Use the **Configuration Wizard** to do all the configuration of the OPC server. There are two Configuration Wizards, one for System 800xA and one for OPC standalone systems, that is, the OPC server is not used together with other 800xA products. Follow these steps for the configuration:

1. If System 800xA is already installed, select the **Configuration Wizard** from the ABB Start Menu.

2. Select **AF100 Network** and click **Next**.

![Configuration Wizard Dialog - Choose AF100 Network Settings](image_url)

*Figure 7. Configuration Wizard Dialog - Choose AF100 Network Settings*
3. Set **Bus number** and **Bus Length**, click **Next** to continue.

**Bus Number**
0-255. Use the same bus number as you use in the CBA application.

**Bus Length**
You can choose between three pre-defined bus length intervals (2000, 8500 and 15000 meters). To know which bus length to select you need to know the details of your AF100 network installation. The selection is up to maximum length, so for most systems 2000 meters will be sufficient.

**Note:** AC110 and AC160 support only 2000 meters

![Configuration Wizard](image)

*Figure 8. Configure PCI Device Parameters*

4. Set the **Bus Parameters** and select the **Time Synchronization**.

Set **Station Number** to a unique number.

**Bus Parameters:**
Select the **Bus Master** check box. There are only a few exceptions with large AF100 networks where it might be an advantage not to check it.
Check the **Redundant Line Support** if redundant AF100 cables are used.

**Time Synchronization:**

Time synchronization enables the Advant Fieldbus 100 user to time synchronize all stations connected to the Advant Fieldbus 100 network. Possible configuration modes are: None, Master, Backup or Slave.

- **None**: No time synchronization is performed against the AF100 bus.
- **Slave**: The AC 100 OPC Server keeps the PC clock synchronized to the time received from AF100.
- **Master or Backup**: AC 100 OPC Server permanently monitors AF100 for a time master.

  If a time master on AF100 bus is detected, it will act as a **Slave**.

  Otherwise, it will act as **time master**, that is, it will retrieve the time from the Windows operating system and distribute it on AF100 bus.

  The AF100 Diagnostic tool displays the configured mode as well as the current state of time synchronization.

Both time synchronization and OPC Server redundancy requires that Bus Master is checked to work correctly.

Only one station on the AF100 bus may be set to time synchronization Master. Only one AC 100 OPC Server may be configured to time synchronization Backup.

If a Backup node becomes Master or vice versa this is not saved permanently. At restart the AC 100 OPC Server will go back to the original configured time synchronization mode.

Select the **Keep server running** check box to keep the server software and the communication board CI527A running even when all clients applications have been closed. This will speed up client restarts.
Figure 9. AF100 Bus Configuration

5. Click Next and then click Finish in the Apply Settings dialog.

6. In the Configuration Wizard, select AC 100 OPC Server Definitions and click Next. The OPC Server Configuration dialog appears.
7. Enter the path to the Bus Configuration Database (BCD) and some other applications.

The **BCD path** is a file path available in the Application Builder (part of Control Builder A) accessible through **Help > On Project**.

The path consists of the shown Project Root extended by a backslash (“\”) and the Project Name. In the example shown in Figure 11 the BCD path is `c:\proj\acptt17`.

If the path is placed on a network disk you must define a user login for the Advant OPC Server that has access rights to that disk. If you use a distributed BCD, the path must be specified using UNC notation, for example, `\Server1\projects`.

Placing the BCD on a network device is convenient during commissioning for redundant OPC server, which have in this case always the same data source. Restarting the OPC Server is not possible, if the network device is not available. As the redundancy is threatened by a single failure, it is
recommended to copy the BCD to a physical drive of the OPC Server PC, after the commissioning phase.

![On Project dialog](image)

Figure 11. Application Builder dialog Help>On Project

Set the **Application Version Supervision** to the interval length in seconds that you want between each check of Controller database changes (due to CBA changes).

Set the **Event enabling cycle time** to the interval required between each Enable Event message sent to controllers. The Enable Event message is required by the restarted controller to send alarms and events to the OPC Server.

Select the **Enable** check box in **Dynamic AIS Alarm Limits** if you want the OPC Server to always have AIS-alarm limit items (for example, HI2_LIM) synchronized with the controller.

If the check box is not selected, the AIS alarm limits will be read only at start-up.
The OPC-server will read the limits from the controller when an AIS-alarm limit item is added to an OPC-group. In run-time the alarm limits are updated via controller sent Event Sets. The disadvantage with usage of this feature is increased bus load due to Service Data traffic and in worst case, subscription timeouts in client applications. If there are more than ten AIS Bargraphs present in the one process display the suggestion is not to use this feature, or use bargraphs without limit presentation.

If there is a redundant pair of OPC Servers in your configuration specify the IP-address of the other OPC Server at **Redundant OPC Server IP**.

8. Click **Next** and then click **Finish** to set your choices.
9. In the **Configuration Wizard**, select **AC 100 OPC Server Station Filtering** and click **Next**.

![Figure 12. Configuration Wizard Dialog - Choose OPC Server Filtering](image)

10. The OPC Server Filtering dialog will appear. Select the stations to be visible in the OPC Server.\(^1\)
Stations of type OPC cannot be filtered. They are always exposed by the AC 100 OPC Server.

11. Click **Next** and then click **Finish** to set your choices. The system prompts for a restart of the PC (see **Figure 14**).

---

1. The stations listed in the ‘Filtering of OPC Server’ dialog are the stations stored in the BCD for the corresponding bus number. This dialog is empty until the BCD path is defined and the computer is restarted. The controller stations in the BCD come from nodes (with DB Elements for AF 100 communication interfaces) created in Application Builder and from OPC stations created in Bus Configuration Builder. You can see these stations also in the left window of Bus Configuration Builder under the appropriate bus number.
Considerations

The basics of configuration and application programming of Advant Controller 100 Series and their derivatives are described in *AMPL Configuration Advant Controller 100 Series Reference Manual*.

Additionally consider the following design aspects when creating configuration data and building applications:

- **Network configuration**
  Consider the limitation described in subsection *Capacity and Performance* on page 47.

- **DAT elements**
  DAT based elements must be referenced from a configured SEND DSP. They only transfer the value.

- **Extended DB elements**
  If you use Extended DB elements you have higher functionality (see Appendix A, *Object Attributes and Events* and Appendix C, *Extended Database Elements*) and the application building in the controller is simplified. The amount of data transferred on the bus is increased if you use Extended DB elements compared with DAT based elements. Extended DB elements transfer object value and status cyclically on the bus while DAT based elements only transfer value. Note that not all controllers do support Extended DB. See also subsection *Compatibility* on page 53.
Sending Commands and Subscribing for Values

In the AC 100 OPC Server you have two possibilities to communicate on the Advant Fieldbus 100:

- Dataset Peripheral (DSP)
- Service Data Protocol (SDP).

An overview of the DSP and SDP communication is described in subsection Communication Overview on page 22.

You can use both SDP and DSP to send commands and subscribe for values from the controller. You find recommendations on when to use the different methods in this subsection below.

The differences between these two methods are summarized in the Table 4.

Table 4. Command Methods

<table>
<thead>
<tr>
<th>Command method</th>
<th>Advantages (+)</th>
<th>Disadvantages (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Data Protocol</td>
<td>Multiple OPC Servers can send commands to the same object without any additional logic in the controller.</td>
<td>No guarantee when the command is sent. Depends on the load in the PC and on the AF 100.</td>
</tr>
<tr>
<td>(SDP)</td>
<td></td>
<td>Very low transmission speed, approximately 1..2 values per second can be transmitted with the SDP.</td>
</tr>
<tr>
<td></td>
<td>A command can be “broadcasted” to multiple controllers (all configured with a receiving DSP for this command).</td>
<td></td>
</tr>
<tr>
<td>DataSet Peripheral</td>
<td>High transmission speed, commands are sent directly, independent of the load in the PC.</td>
<td>Needs additional logic in the controllers to handle multiple OPC Servers sending value changes to the same object.</td>
</tr>
<tr>
<td>(DSP)</td>
<td>A command can be “broadcasted” to multiple controllers (all configured with a receiving DSP for this command).</td>
<td>Requires additional DSPs (one for sending the value and one for receiving the mirrored value from the controller).</td>
</tr>
</tbody>
</table>
Section 2  Configuration

Sending Commands and Subscribing for Values

- **Cyclic subscriptions and bus load**
  Subscription frequency is set in the client application. Subscription frequency that is higher than the frequency at which the DSPs are updated on the CI board, has no effect. Avoid this to minimize the AF100 Communication handler’s load. See Capacity and Performance on page 47 for calculation of bus load.

The cycle time of the PCPGM, Control module and SCANT should be considered internally in the controller see Figure 4. This affects the cycle time of the whole system.

**DSPs Sent by the Controller**

The controller sends the DSP containing the object value. The AC 100 OPC Server and other nodes on the bus receive the DSP and can subscribe for the object data. All AC 100 OPC Server nodes on the same bus can send commands to update the corresponding object value. The commands are sent via the SDP. You do not need to create any additional logic in the controller or in the AC 100 OPC Server node. The SDP can directly be used to send commands to all objects defined in receiving DSPs in the AC 100 OPC Server. Sending DSPs from controller nodes can be used for both Extended DB elements and DAT based elements.

![Figure 15. Commands sent via the Service Data Protocol for Extended DB Elements](image-url)
Only one DAT_DI/DO object can be mapped to a DAT(B) object (record of 32 bits) with this solution since the command sent via the Service Data Protocol affects all bits of the DAT.

**When to Use -- DSPs Sent by the Controller**

- This is the most common way to work with all object types and it will give you minimum of engineering work.
- When several AC 100 OPC Server nodes should be able to change an object value.
- This is the only supported communication feature when you are working with Extended DB Elements AIS, AOS, DIS, DOS since these objects have a DSP transferred object status and this status can only be written by the controller.
- This is preferred if the objects should generate events since the controller that sends the value also sends the event.

*Figure 16. Commands sent via the Service Data Protocol for a DAT Based Object*
When Not to Use -- DSPs Sent by the Controller

- When commands must be sent cyclically.\(^1\)

The method **DSPs Sent by the Controller** is the most common and recommended method.

The two methods **DSPs Sent by the AC 100 OPC Server** and **SDP Without Read DSP Communication** are more special cases and can be skipped during first reading of this manual. You can continue with **Packing and Scaling Values with DAT(II) Objects**.

**DSPs Sent by the AC 100 OPC Server**

The AC 100 OPC Server sends the DSP containing the object value. The controller nodes on the bus receive the DSP. Commands sent from the AC 100 OPC Server are “broadcasted” to all nodes with corresponding receiving DSPs. It is only the node with the sending DSP that can send commands to the corresponding objects.

Be aware of that the OPC Server sending DSPs are initialized to zero and “broadcasted” on the AF100 bus immediately at the OPC Server startup.

If the AC 100 OPC Server also subscribes for data from the object you better should define a sending DSP in the controller.

---

1. When using SDP to send commands there is a physical limit of eight SDP messages per second. Also note that the AC 100 OPC Server node needs a number of SDP telegrams on a cyclic basis for network supervision.
With this solution, you can use all bits of the DAT(B) to map different DAT_DI/DO objects.

See Appendix B, Setting up OPC Server to send DSPs for information how to send DSPs.

**When to Use -- DSPs Sent by the AC 100 OPC Server**

- When the OPC Client must send commands on a cyclic basis to one of the following objects:
  DAT_DI, DAT_DO, DAT_AI, DAT_AO, MR, MI, MIL, MB.

**When Not to Use -- DSPs Sent by the AC 100 OPC Server**

- When you use the Extended DB elements AIS, AOS, DIS, DOS since these objects use a DSP transferred object status which cannot be written by the AC 100 OPC Server node.
- When several AC 100 OPC Server nodes should be able to change an object value. This cannot be done without additional logic in the controller.

Redundant OPC Servers cannot send the same DSP from both OPC Servers, as it is not allowed to have 2 sending DSPs with the same identification (STATION, IDENT). The DSP would only be sent by one of the OPC Servers.

- When you are working with objects that generate events. The AC 100 OPC Server nodes cannot generate events on AF100, it must be done by a controller node.

Values could be sent from the AC 100 OPC Server node to a controller which generates events. If the DSP is slow and/or the DSP communication is lost the object value observed in the OPC node might be in a disturbed state (object in alarm state) but without a corresponding alarm.

**SDP Without Read DSP Communication**

You can use the SDP to read object data without sending the data in a DSP. The SDP will send a read command to the controller and the controller returns the data.

This feature can only be used together with the Extended DB elements MI, MIL and MR and with objects of type MB without event handling.
You can specify that the object only use SDP in the FCB. This is made when you create the object by setting the terminal DS_COM=NONE and MMI_USE=YES.

**When to Use -- SDP Without Read DSP Communication**

- When there are no more free DSPs to send data with, the object value is almost static and almost never subscribed for.

**When Not to Use -- SDP Without Read DSP Communication**

- In all normal use of the system.
- When you must be able to send commands from the AC 100 OPC Server without delay. All outgoing commands will be queued if you read many values with SDP.
- When you have event handling enabled. Because the alarm lists in the OPC client usually subscribe for the object values cyclically at a high rate.

**Packing and Scaling Values with DAT(II) Objects**

In a DSP configuration with value references of type I (DAT(I)) you can only use eight value references (objects) per DSP. In a configuration with the DAT(II) object type you can pack two integers into each value reference. Each DSP can then send 16 integers (16-bits allow values within the range -32768..+32767). Each one of these integers can be scaled by one of the following factors [0.001; 0.01; 0.1; 1; 10; 100; 1000] and the scaled result is represented by a floating point value. This packing and scaling of the value is only supported by the DAT_DAT MMI object.

**When to Use -- Packing and Scaling Values with the DAT(II) Objects**

When you need to transfer a lot of data and need to reduce the amount of DSPs.

This feature is only supported for APC2 and the AC 80.

**Capacity and Performance**

The performance of the AC 100 OPC Server depends on the client application program and the Windows system’s load.
The Bus

One Advant Fieldbus 100 bus (coax cable) can have up to 79 nodes (stations 1 ... 79) connected. On Advant Fieldbus 100 with twisted pair up to 32 nodes are possible per segment, several segments may be connected via modems to handle up to 79 nodes.

The bus load on the AF 100 has to be checked after each change in the bus configuration.

The capacity of the AC 100 OPC Server is limited by the CI board capacity. The capacity of the CI board is indicated in Table 5 below.

Table 5. Advant Fieldbus 100 Interface Capacity

<table>
<thead>
<tr>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total no. of DataSet Peripherals</td>
<td>3950</td>
</tr>
<tr>
<td>Cycle times depending on configuration in AC 100 series</td>
<td>1, 2, 4,... 4096 ms</td>
</tr>
</tbody>
</table>

As each DataSet Peripheral can reference up to eight DAT elements, the maximum number of DATs is 31600.

The Advant Fieldbus 100 runs a data rate of 1.5 MBits/sec. Some of the information is, however, used for preambles, cyclic redundancy checks and protocol overhead; as a result, the net data rate is somewhat less.

The transfer times for DataSet Peripherals depend on the number of attached DAT elements (see Table 6).

Table 6. DataSet Peripheral Transfer time

<table>
<thead>
<tr>
<th>DSP size</th>
<th>Transfer time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 DAT element</td>
<td>0.088</td>
</tr>
<tr>
<td>2 DAT elements</td>
<td>0.108</td>
</tr>
<tr>
<td>3-4 DAT elements</td>
<td>0.156</td>
</tr>
<tr>
<td>5-8 DAT elements</td>
<td>0.252</td>
</tr>
</tbody>
</table>
The AF100 bus load has to be calculated when the system is configured. For a complete description of bus load calculation refer to the *Advant Fieldbus 100 User’s Guide*. The Bus Configuration Builder in the CBA can also be used to calculate the bus load.

The bus load depends on the DataSet Peripheral configuration and can be calculated using the following formula:

$$\text{BusLoad} = \sum \left(\frac{\text{Nbr} \cdot \text{Ttr}}{\text{cT}} \right) \cdot 100$$

This formula provides the bus load as a percentage, where the sum is taken over all sending DSPs and:

- $\text{Nbr}$ = number of DSPs (of the same data size and $\text{cT}$)
- $\text{Ttr}$ = transfer time in ms (from Table 6)
- $\text{cT}$ = desired cycle time in ms (1, 2, 4, 8, 16...4096).

In order to guarantee that message transfer is possible, at least 25% of the bus bandwidth is reserved for message transfer (SDP - Event Set and command signals). Up to 70% of the bandwidth may be used freely for DataSet Peripheral communication.

**Command Response Time**

The average for a synchronous OPC write from an OPC client to an object attribute in a controller is about 0.5 seconds. The time is measured from when the sending starts until the changed value is acknowledged back to the client.
**OPC Server Performance - Principles**

There are a number of terms which must be known before you can understand how to utilize the OPC server in the best possible way.

An OPC client sets up so called **OPC groups of OPC items**. Each OPC group provides a list of OPC items (each item normally corresponds to an Object attribute).

The OPC server builds up a memory **Cache** for each OPC group. All static attributes are read from the so called BCD file, see Table 1 for more information about the BCD file. Those attributes are stored memory resident. Dynamic attributes are fetched from the Device. The OPC Server transfers the dynamic data from the Device to the Cache according to a cyclicity specified in the OPC group, see Figure 18. As an example - for an AIS object the VALUE and STATUS attributes are transferred cyclically. As long as the Group is valid the OPC server will continue to refresh the Cache with the specified interval (10 ms up to minutes). The recommended Cache refresh time is 1 second for most applications.

The OPC client can request data from the Cache or directly from the Device. When data is requested directly from **Device**, a forced reading is done from the

---

*Figure 18. OPC Subscription Principals*

An OPC client sets up so called **OPC groups** of **OPC items**. Each OPC group provides a list of OPC items (each item normally corresponds to an Object attribute).

The OPC server builds up a memory **Cache** for each OPC group. All static attributes are read from the so called BCD file, see Table 1 for more information about the BCD file. Those attributes are stored memory resident. Dynamic attributes are fetched from the Device. The OPC Server transfers the dynamic data from the Device to the Cache according to a cyclicity specified in the OPC group, see Figure 18. As an example - for an AIS object the VALUE and STATUS attributes are transferred cyclically. As long as the Group is valid the OPC server will continue to refresh the Cache with the specified interval (10 ms up to minutes). The recommended Cache refresh time is 1 second for most applications.

The OPC client can request data from the Cache or directly from the Device.
Device, the refresh cycle time should then be set very high. As a request towards the Device is between 12 -20 times more CPU consuming than the Cache request, access directly towards the Device should be limited to a few critical objects only.

The OPC client can request data either synchronously or asynchronously. When it subscribes **synchronously** it waits for the subscribed data to be delivered until it continues the execution. The time it takes to read the data can be measured.

When the OPC client subscribes **asynchronously** it issues a request and is then interrupted (callback) each time data arrives. If you combine asynchronous reading with a **cyclic subscription interval** you will get all the OPC items from the OPC server at the first refresh but afterwards you will only receive data when they are changed.

Asynchronous subscription with a cyclic subscription interval towards the cache (typically 1 second) is the recommended subscription for most applications. With such a subscription the data is never older than 1 second and you minimize the data flow between the OPC client and the OPC server to the changed data (once the first set of data is transferred). As most attributes do not change at all during the subscription the load on the PCs where the OPC server and OPC clients reside is minimized.
OPC Server Performance - Subscription Field Tests

A simple OPC client giving little additional overhead was used in the tests. To help you to evaluate the OPC server performance an application that reads OPC items synchronously as fast as possible was executed. This means that during the tests the CPU load was very close to 100%.

The following parameters were calculated by reading 5000 items from about 200 AIS and DIS objects. The cyclic refresh time was specified to 1000 ms:

1. Required CPU MilliSeconds for the OPC client and the OPC server to handle OPC items read from Cache. **Result:** 29 msec per 1000 items.

2. Required CPU MilliSeconds for the OPC client and the OPC server to handle OPC items read from Device. **Result:** 40 msec per 1000 items.

By adding additional tests it was possible to translate the tests to a similar OPC client for example a Process Display with AIS objects. The results can be estimated according to **Figure 19**. All OPC items (attributes) are read for each AIS object.

![Figure 19. Subscription Performance](image)

---

**Figure 19. Subscription Performance**
Compatibility

The Table 7 shows compatibility for program versions in different controllers and supported AF100 options.

Table 7. Supported Options for Different Controllers

<table>
<thead>
<tr>
<th>Options</th>
<th>Controller/Controller version</th>
<th>AC 70</th>
<th>AC 110/ver. 2.0</th>
<th>AC 110/ver. 2.1 or later</th>
<th>AC 160/ver. 1.0 or later</th>
<th>APC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended DB elements</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DAT based Objects</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SDP</td>
<td></td>
<td>Yes (for commands only)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (for commands only)</td>
</tr>
<tr>
<td>Event handling</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AF100/Optical Bus</td>
<td></td>
<td>No(1)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No(1)</td>
</tr>
<tr>
<td>AF100/Coaxial Bus</td>
<td></td>
<td>No(1)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AF100/Twisted Pair Bus</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No(1)</td>
</tr>
</tbody>
</table>

(1) Modems to convert between bus types are available.

Multiple and Redundant OPC Servers

It is possible to start an own OPC Server on each of several PCs. Each OPC Server is an additional station on the AF 100 bus. These OPC Servers are normally not aware of each other. They can process any desired area of the plant, maybe different areas or maybe identical areas.

MMI OPC clients can attach to theses OPC Servers as desired. An MMI OPC client can provide the functionality to select two OPC Servers as redundant for each other:
If one OPC Server fails, the MMI client may switch automatically to the other OPC Server and continue to display values smoothly.

In the case of a redundant OPC Server these two systems will cover the same area of the plant. This means that the two OPC Servers must have

- the same stations selected in Configuration Wizard dialog Station Filtering of AC 100 OPC Server,
- identical data regarding the filtered stations in their BCD data bases (or the same BCD data base)
- and identical values for all the dynamic items.

This is normally the case, since all the dynamic values are fetched from the (same set of) controllers.

![Figure 20. Station Filtering](image)

**Bus Configuration Data Base in Distributed Configurations**

In the following configuration the BCD has to be accessed by several PCs: CS1 - Connectivity Server 1 (PC running AC 100 OPC Server)
CS2 - Connectivity Server 2 (2nd PC running AC 100 OPC Server)

AES - Advant Engineering Station

For information on distributed configurations see also the *System 800xA, Configuration (3BDS011222)*.

The Application Builder Project and hence the access data base BCD file could be located on:

- Advant Engineering Station (not recommended)
  - Both CS1 and CS2 will access the BCD via Network.
  - Please specify as LogOn User for AC100 OPC Server the SystemServiceAccount with the same password on AES, CS1 and CS2.
  - In Configuration Wizard the BCD Path has to be entered for both Connectivity Servers CS1 and CS2 using UNC notation, e.g. \AES_name\OPCproj\Project_name.
  - With this solution an online change is automatically detected by both Connectivity Servers, but a single failure of AES will lead to a failure of both Connectivity Servers.

- Connectivity Server 1 (recommended)
  - CS1 will have direct access to the BCD and any online-change will be detected automatically by AC100 OPC Server.
  - CS2 will access a local copy of the BCD. This ensures that a single point of failure at one of the Connectivity Servers will not affect the availability of the other.

To update the BCD on CS2 please take the following steps:
- Open **Computer Management > Services**.
- Double click on ABB AC 100 OPC Server.
- Change Startup type to Disabled, click **Apply**.
- Stop the server from **Stop** button.
- Copy the BCD file from CS1 to CS2.
- Change Startup type back to Automatic, click **Apply**.
It is not supported to copy the MDB file when the OPC server is running.

In case of an online change the situation arises that the databases of the two servers are not identical since their update process is not synchronized. This can lead to temporal inconsistencies between the OPC Servers which will not be corrected but will disappear after the changes have been applied to the second server.

Further they have to expose the same and accurate alarm and event information to MMI. This means that all information of an alarm visible to the operator, e.g. event time, event message, condition, ack time, etc, are equal on both servers. It also means that an exposed alarm matches the actual alarm event, regarding type, time, condition, etc, e.g. correct alarm for a limit transgression.

All Personal Computers running AC 100 OPC Server in an AF 100 network must be configured to the same time zone and must be time synchronized.

**Data Access**

All OPC Servers receive the same information from the controllers via the AF100 Bus and expose the same values. An OPC client (for example MMI) can take this information from the current active server and can switch automatically to the redundant server in case of a server failure.

**Alarm & Event**

As with normal data communication all servers receive the same information from the controller, therefore they are all equally informed about process and system events.

Exceptions are:

- The acknowledge for an alarm is received only by one OPC server (sent by MMI) and this acknowledge needs to be distributed to the other OPC Server. For Extended and DAT-based objects this has to be configured in BCB by using
Edit > Properties on the corresponding (ACK sending) AF 100 OPC Station and entering the desired (ACK receiving) stations: All AF 100 OPC Stations then share the same knowledge about the state of acknowledged DAT-based objects.

Note that a PC running AC 100 OPC Server is represented in BCB as an AF100 OPC Station object on the AF 100 bus.

---

**Figure 21. Configuring distribution of ACKs**

- Soft alarms are events for the system status objects. They are generated individually on each OPC Server by reading certain status words received from the controller. This process is not synchronized between the OPC servers.

The time stamps of the soft alarms can show time differences of up to 10 seconds. Duplicated alarms may occur during startup of a redundant OPC Server.

- If an OPC Server fails, restarts and reconnects to the network it has no information about the currently active alarms. It needs this information from
the running server. This is done by the OPC Server starting up. It connects to the active OPC Server as Alarm & Event client and reads these informations before it starts normal operation.

To enable Alarm & Event synchronization at startup the TCP/IP address of the redundant OPC server has to be entered for each OPC Server using the Configuration Wizard’s AC 100 OPC Server Definitions item as shown in the example of Figure 22.

![Configuration Wizard]

**Figure 22. Redundant AC 100 OPC Server Configuration**

It is not possible to add another, redundant OPC server at runtime to a single server configuration in order to build a redundant configuration. The running server’s configuration must be updated - the redundant server’s IP address must be specified via Configuration Wizard (see Figure 22) - which requires a restart of the OPC Server. A restart of the OPC Server is done when you restart the PC.

It is the responsibility of the OPC Servers to expose equal and correct data (Data Access and Alarm & Event) and to ensure that no events are lost for a single point of failure.
Known exception: When the time is switched backwards for daylight saving time and the redundant OPC Server is starting up (refresh from primary OPC Server), it may happen that alarms are lost. This is a quite seldom case because the same alarm on the same object must happen exactly at the same time before and after switching time backwards.

It is the MMI OPC client's responsibility to display this data correctly. For example, the OPC client has to make sure that it does not display the same alarm or event twice. The OPC client has to decide from which OPC server the data is received and displayed. And it is also the OPC client's responsibility to switch from one server to the other if one server fails. The OPC client has to handle possible gaps in the displayed data flow, e.g. trends, which might occur during the switch over.

![Diagram](image.png)

**Figure 23. Redundant OPC Server Synchronization Support**

Multiple, non-redundant OPC servers which expose the same set of dynamic data items of a controller can only use the distribution of acknowledge information. Alarm & Event-synchronization at start-up can only be used by redundant OPC servers.
Startup of Redundant OPC Servers

Redundant OPC Servers have to be started up one after the other. The second OPC Server must not be started before the first has connected successfully to the MMI OPC client. If they startup concurrently the Event and Alarm List may not be the same in the OPC servers.

Online Changes

The AC 100 OPC Server supports online changes in the BCD. It supervises BCD and also each controller for possible changes in the loaded application.

If Application Builder and Connectivity Server run on different PCs, the project created with Application Builder should physically be located on the Connectivity Server. This ensures that the BCD is always accessible for the OPC Server. For further information refer to Bus Configuration Data Base in Distributed Configurations.

With few exceptions, any BCD change (while OPC Server is running) is recognized by the Application Version Change (AVC) function of the OPC Server- after some time (default: 2 minutes) and automatically incorporated in the OPC internal data structure and then exposed to OPC clients.

Events are generated if changes are detected, see Table 31 in Appendix A, Object Attributes and Events.

Online changes in PM (i.e. DESCRIPTION) and CI are not incorporated. Application Version Changed (AVC) events are generated only covering other online changes.

Hence the final event description 'AVC: BCD matches AC' is a little bit misleading, since BCD might not match OPC with respect to a (changed) PM or a CI.

To incorporate these PM and CI changes really you have to stop and restart OPC Server. A restart of the OPC Server is done when you restart the PC.

Application version change is only shown for the Primary CPU and not for the Backup CPU.
Customizing the OPC Server

General

You have now read through the considerations in the subsections

- **Sending Commands and Subscribing for Values** on page 42
- **Capacity and Performance** on page 47
- **Compatibility** on page 53.
- **Multiple and Redundant OPC Servers** on page 53
- **Online Changes** on page 60.

Now we will learn how to configure and customize the AC 100 OPC Server system. During the life cycle of your automation system there might also be a need to customize the configuration of the OPC server. The following sections include instructions how to do this.

- Subsection **Working With AC 100 OPC Data** on page 62 contains information on different types of objects: `OPCServer`, `OPCGroup`, `OPCItem`, and how to connect to a remote AC 100 OPC Server using DCOM and about using a `distributed bus configuration database`.

- Subsection **Working with the Alarms and Events** on page 68 contains information on changes you can do to the alarm and event handling.

- Subsection **Working with Control Builder A (CBA)** on page 73 contains general information about the CBA tool.

- Going through the tutorials in subsection **Tutorials** on page 74 will give you a deeper understanding of the AC 100 OPC Server.
Working With AC 100 OPC Data

General

The AC 100 OPC Server provides the mandatory OPC Custom interface supporting clients written in a compiled language such as C++. An OPC Server contains several types of objects: OPCServer, OPCGroup and OPCItem. The OPCGroup object provide a way for clients to organize data. For example the group could represent items in a particular display or report. The OPC Items represent connections to data sources within the server. A data item could for example be the value for a specific input signal in the controller.

OPC clients access the AC 100 OPC Server as any other OPC server.

The version dependent programmatic identifier (ProgID) for the AC 100 OPC Server is ABB.AC100.1.

Specific for the AC 100 OPC Server is the item syntax defined as follows:

<objectname>(.<attribute>)

Examples:

AIS3_5.NAME
AOS5_1.VALUE
DIS1_5.VALUE

The object names used by the AC 100 OPC Server must be unique, which means that all database (DB) elements in all controllers on same bus with the terminal MMI_USE = YES must be assigned unique names.

You can find a detailed description of object attributes in Appendix A, Object Attributes and Events.

Connecting to a Remote AC 100 OPC Server

When you install the OPC server, the executable OPCenum.exe from OPC foundation is automatically installed on the server. You can read on the OPC foundation internet pages (www.OPCfoundation.org) in the document OPC Common Definition and Interfaces version 1.0 how to utilize it to set up distributed access to the OPC server.
In short you use the Interface IOPCServerList to obtain a list of servers either locally or on a remote host. This interface basically provides the functionality of the Component Categories Manager. It has been defined by OPC, because access to the Component Categories Manager does not work for remote machines.

How to View the Diagnostics

To view the internal status and diagnostics information, select **AC 100 Diagnostics** from the ABB Start Menu.

![AF100 Diagnostics 5.0.0 Beta2](image)

**Figure 24. Diagnostics - Station Status**

**Diagnostics**

Note that the Diagnostics tool accesses the CI board directly. It has no access to BCD and displays data in a 'raw' physical manner.
• **Station Number**
  Station number for the current station. Must be a number 1-80 that is unique on the bus.

• **Bus Number**
  0 - 255. Available busses for the station.

• **Board Id**
  The Board Id shows the type, version and revision of the interface board.

• **Board Status**
  – **Bus Master**
    Bus Master configuration ON or OFF - should always be ON.
  – **Signal Configuration**
    The Signal Configuration shows that configuration data is being transferred to the CI board and normal runtime operation is disabled.
  – **Redundant Line Support**
    The Redundant Line Support is checked if redundant modems are used.
  – **Line 1 Active**
    If Line 1 Active is checked, the line 1 is chosen from the two redundant lines.
  – **Line 2 Active**
    If Line 2 Active is checked, line 2 is used.
  – **Reconfiguration**
    The reconfiguration shows when errors or discrepancies are detected when the CI is scanning bus administration lists from other bus masters.

• **Board State**
  The Board State indicates the current mode of the CI board.
  – **Operational** - the communication board is running
  – **Ready** - the board is configured but not operational
  – **Not Ready** - the board is not operational and the configuration has been cleared
Section 2  Configuration

How to View System Status

- Error - the board is passive and not operating properly.

- **Time Synchronization**
  Possible modes are None, Slave, Master or Backup.

- **Board Errors**
  Errors from the CI board are displayed in the Board Error window.
  Double click on a device error to get more information about the error.

- **Reachable Stations and PMs**
  All reachable stations are displayed in the Reachable Stations and PMs window. By double clicking one of the reachable stations, detailed station status can be viewed, with more detailed information about the error.

  The status is marked with green, yellow and red meaning:
  - green - station operational and no errors.
  - yellow - minor station error, which does not prevent the application from running.
  - red - station not operational due to fatal error or blocked.

**How to View System Status**

To view the system status information (and any other information) of objects in the nodes reachable via the OPC Server you can use free third party OPC viewers available as downloads in the world wide web.

Below you can find examples using the viewer Matrikon OPC Explorer (available on www.matrikon.com) to view the data of attributes of system status objects PM and CI:
Figure 25. Example: View of a PM Object
Figure 26. Example: View of a CI object

To be able to view several different PM- and CI objects, corresponding to different controllers on the same AF100 bus, you need to have unique names for each PM and CI object. The unique object names have to be entered at the corresponding DB elements in Function Chart Builder DB Element Editor.

The IO object for controllers and the OPC- and CI board objects for the OPC Server station will automatically be read from the AF100 bus when appearing. PM_SUM and CI_SUM status objects also will be created automatically. These objects are not entered in the CBA. See, Appendix A, Object Attributes and Events for more information about attributes that corresponds to the system status objects.

How to View System Messages

Select Windows Event Viewer. Use the Application log to view messages sent by the AC 100 OPC Server.
You should customize the Event Viewer to show information that is suitable for you when monitoring the AC 100 OPC Server. See Preventive Maintenance on page 123 for more details on this.

**Working with the Alarms and Events**

**General**

**Kinds of Time-tagged Events**

Advant Controller 100 Series controllers in general may generate *calculated* events:

*Calculated* events are time tagged events which are configured by DB elements AIC and DIC.

Events may also be generated by other sources, which are not supported by all controller types:

- Events from I/O modules, such as DI650/DI651/DI652
- Events received in MVI Data Blocks and converted into the internal event format by means of DB elements AIMVB and DIMVB
- Events from extended DB elements, such as AIS, DIS.

DB elements DIC, AIC, DIS, AIS, DIS650/DIS651/DIS652, DIMVB, AIMVB used for configuration of event detection functions are called **Event Channel** elements.

**Calculated Events**

DB elements DIC and AIC allow the surveillance of connected PC signals (of the PC section in the controller) of type Boolean or Real for events. Event detection is *calculated* according to the parameters specified in these DB elements with a resolution down to 10 ms.

An event is triggered for a Boolean PC signal when the signal changes from 0 to 1 or vice versa.

For a PC signal of type Real an event is triggered when the signal exceeds one of the two low or high limits or moves back within the approved range.
**Event Configuration**

Events detected on controller are buffered in up to 32 event queues, each event queue can store up to 1000 events, depending on controller type. If events are generated while the event queue is full the new events overwrite the oldest events. The event queue is represented by the EVS DB element, in case of creation of an extended DB element the EVS is automatically generated.

You customize the Controller objects alarms and events by using the configuration features of Function Chart Builder DB Editor. The OPC server reads these configuration data at start-up (from the BCD file).

Examples on configuration data that you can change are

- alarm and event treats, see Appendix A Object Attributes and Events.
- alarm and event texts, see **Alarm and Event Texts** on page 70.
- alarm and event severity, see **Alarm and Event Severity** on page 72.
- distribution of acknowledge for DAT-based objects, see **Distribution of Acknowledge** on page 73.

**General Features**

Time-tagged events from a controller are transferred to one or several event receivers over Advant Fieldbus 100. An event receiver can be:

- An Advant Controller
- A personal computer running an OPC Server

The number of event receivers on an AF100 Bus is limited to 10.

**Event Set Elements**

The transfer of time-tagged events is configured by Event Set (EVS) elements.

An Event Set element of type “send” (configured by database element EVS(S)) groups a set of Event Channels (AIC, DIC, and so on, as applicable) for transmission of events. Each Event Set can handle up to 32 Event Channels of different types which can be mixed in arbitrary order.
An EVS(S) element collects events from its referenced Event Channel elements and sends the events to the event receivers when those request them.

An AC 100 Series Controller can handle up to 32 Event Set elements EVS(S).

**Alarm and Event Texts**

You can change the alarm and event texts by using the CBA software.

The following objects/events can get their strings from DB Element Editor of Function Chart Builder in CBA:

- **AIS:** Above H2Lim
  Below H2Lim
  Above H1Lim
  Below H1Lim
  Below L1Lim
  Above L1Lim
  Below L2Lim
  Above L2Lim

- **DIS, DAT_DI:** Value differs from Normal Position (this is always the T1VAL string).
  Value goes back to Normal Position (this is always the T0VAL string).

- **AOS:** Above H1Lim
  Below H1Lim.
  Above L1Lim
  Below L1Lim

- **DOS:** Value differs from Normal Position (this is always the T1VAL string).
  Value goes back to Normal Position (this is always the T0VAL string).

- **MBS:** Value differs from Normal Position (this is always the T1VAL string).
  Value goes back to Normal Position (this is always the T0VAL string).
The DB Element Editor of Function Chart Builder in CBA supplies default values.

Alarm and event texts and severity values that are not specific for an individual object, for example “Update Block”, are also defined in the BCD (Bus Configuration Database). They can additionally be accessed and changed via Edit >Alarm & Event Texts for AF100OPC in Bus Configuration Builder (BCB) of CBA, similar as shown in Figure 27.

![Figure 27. Changing the Alarm texts in BCB](image-url)
Using this tool all Alarm & Event texts in the BCD can be accessed and changed. For simple events the key is constructed according to <event>_STR, for condition events according to <event>_ON_STR and <event>_OFF_STR.

<event> is the event identification given in the first column of the tables describing the events in Appendix A, Object Attributes and Events.

The severity for an Off condition will get the same severity as the specified severity of the corresponding On condition. This will be the case regardless if you specify different severities for the On and Off condition. Example: If the MAN_AUTO_ON_STR has severity 450 the MAN_AUTO_OFF_STR will get the same severity.

BUSNO and STNNO must be set to -1 and the event TEXT string may not exceed 19 characters.

Any changes made by editing the BCD in this manner will be lost if the BCD is deleted or becomes corrupt. It is therefore recommended that you use a load file to set the values.

For the changes to become valid, you need to restart the OPC server for AF100. A restart of the OPC Server is done when you restart the PC.

Alarm and Event Severity

You can change the alarm and event severity for each type of event for each object instance by using the DB Element Editor of Function Chart Builder in CBA software. For an AIS for example, choose the MMI1 item from the Analog Input drop-down list. Change the default values to a suitable severity, see Figure 67.

You should have a plan for using severities in general before you start changing the severity from the defaults (recommended by OPC foundation). It is rare that only one single object of a type has other severities than the others, normally it is a group of objects, for example fire indicators.

For the changes to become valid, you need to restart the OPC server for AF100. A restart of the OPC Server is done when you restart the PC.
Distribution of Acknowledge

Whenever an alarm is acknowledged, this may automatically be distributed to other OPC Servers. You can explicitly define such a distribution using the BCB:

1. Select the appropriate bus and create an AF100 OPC Station for each OPC server node from which alarms may be acknowledged.
2. Edit the Station Properties (Edit > Properties) of each of these AF100 OPC Stations and enter the station numbers of OPC server nodes to which acknowledgments are to be sent, as already shown in Figure 21.

Working with Control Builder A (CBA)

Use the following manuals or users guides (see Released User Manuals on page 15):

The basics of configuration and application programming of Advant Controller 100 Series and their derivatives are described in


CBA tool functions are described in

Application Builder User’s Guide,
Function Chart Builder User’s Guide and
Bus Configuration Builder User’s Guide.

DB elements are described in

Data Base Elements Advant Controller 70 Reference Manual,
Data Base Elements Advant Controller 110 Reference Manual and

The following tutorials base on informations given in these manuals:

- Creating a Small Test Configuration Using Extended DB Elements
- Creating a Small Test Configuration Using DAT Based Objects
Creating a Small Test Configuration Using Extended DB Elements

By following the steps of this tutorial, you will create a small basic application that you can use to test whether the AC 100 OPC Server is up and running after the installation procedure is completed.

To carry out this tutorial, the following applications and hardware must be installed and working:

- AC 100 OPC Server
- CBA version 1.2/0 or later, including Application Builder, Function Chart Builder and Bus Configuration Builder.
- One AC 110 (or AC 160) controller version 2.1 or later with, at least, the Extended PC-elements SW 2.1 (OPT 1) and Events SW 2.1 (OPT3) options installed.

This tutorial assumes that this is a new installation with no previous node or network configuration. During the tutorial, a “procedure bar” is placed at strategic positions, indicating where in the procedure the tutorial is at a given moment.
Creating a new project

1. Start the Application Builder program by selecting CBA Application Builder from the ABB Start Menu.
2. Select the New > Project item in the File menu.
3. In the Project Name field, fill in following information.

4. Set Project Name to **Myproj**.

5. Set Project Root to **C:\proj** (or as appropriate) and click **OK**.

*Figure 29. Application Builder Start Window*
6. Three windows are displayed:
   – The Node List window
   – The Circuit List window
   – The Type Circuit List window.

These windows show the configuration done so far for the selected project. For a new project, as in this tutorial, no information is presented in the windows.
Creating a Bus Configuration Database

7. Start the Bus Configuration Builder (BCB) from the **Tools** menu in the Application Builder.

8. A start-up dialog box called Select Environment is displayed. Select the environment to work in, by choosing the Bus Configuration Builder and click **OK**.

![Select Environment](image)

*Figure 31. Select Environment*

9. As this is a new project there is no Bus Configuration Database created yet a new one will be created.

10. The BCB application window is displayed, and an empty Bus Configuration Database (BCD) has been created. Leave the BCB by choosing **Exit** from the **File** menu.
Creating a new controller node

11. Return to the Application Builder and select the New > Node item from the File menu.

12. A dialog box is displayed (all fields are empty by default).

13. In the New Node dialog box, fill in the following information.
   - Set Node Name as appropriate (AC 160_2).
   - Set Node type to AC 160 with the version number of the controller.
   - Set Net, Node to 0,0.
   - Set Bus, Station as appropriate (1,2).

   The setting of the bus number must be the same as the setting used for the AC 100 OPC Server with which the controller is to communicate. The station number must correspond with the station number setting in the controller.

14. Select all needed options that are supported by the target system. Click OK.
15. The Node List dialog is displayed, updated with the newly created controller node listed among the other items in the list box.
16. Select the new controller node by marking it in the list box and choose the Function Chart Builder item from the Tools menu in the Application Builder.

![Application Builder - MYPROJ](image)

**Figure 33. Nodes Created in the Project**

17. The Function Chart Builder application is thereby started. During the start-up sequence, you are asked to define the page layout for your diagrams (for example A4, landscape English).

18. Choose an appropriate page size, and click OK.
Building the controller application

19. When the start-up sequence is complete, a dialog with the program sections is displayed.

![Figure 34. Choose Program Section](image)

20. Select **DB** and click **New**, or choose **New Section > DB** from the **File** menu in the FCB menubar.

21. The Create Element dialog box is displayed.

22. In the list box, mark the database element describing the communication interface e.g. **CI630** depending on the hardware in the controller. Click the **DB Terminal Values** button.
Figure 35. Create DB Element dialog
The Edit Db terminal Values - CI1 dialog is displayed.

![Edit DB Terminal Values - CI1 dialog box](image)

**Figure 36. Edit DB Terminal Values dialog box**

23. Set the bus and station numbers and Click **OK**.

24. When you return to the Create Element dialog box click **Apply**.
25. In the list box in the Create Element dialog box, mark the database element describing the processor module of the controller, PM646 (or other, depending on the hardware). Click the **DB Terminal Values** button.

26. Check cross-reference of POSTITION (default value=3), Click **OK**.

![Figure 37. Edit DB Terminal Values of PM](image)

27. When you return to the Create Element dialog box click the **Apply** button.
28. Create a test signal of the Analog Input type. In the list box, mark (click on) the AIS item and--this is important--click the **DB Terminal Value** button.

29. Set the Name item to an appropriate object name (TEST1).

![Edit DB Terminal Values dialog box, Editing AIS - Base Part]

30. Choose the Communication item from the Analog Input drop-down list.

   The parameter listing is updated with communication specific items.

The DSP_STAT and DSP_IDNT parameters explicitly identify a certain DataSet Peripheral. It is imperative that two DSPs with identical DSP_STAT and DSP_IDNT settings do not exist in the system. No checks are made for identical DSPs in other controllers when a new one is created. It is recommended that you run BCB (**Action > Check Database**) to avoid identical DSPs in your system.
31. Set the DS_COM item to SEND,
Set the DSP_BUS item to the actual bus number (1),
Set the DSP_STAT item to the actual station number (2).

![Edit DB Terminal Values dialog box, Editing AIS -- Communication](image)

Figure 39. Edit DB Terminal Values dialog box, Editing AIS -- Communication

32. Choose the MMI Part item from the Analog Input drop-down list. The parameter listing is updated with MMI specific items.

33. Set the MMI_USE to YES,
Set LIM_1_TR to 4, LIM_2_TR to 4,
ERR_TR to 2.
If the treat is set to zero, no OPC-events will be reported to the client.

*Treat*=0- The event is never reported to the OPC client.

*Treat*=1- No Alarms, only events are reported, i.e. no acknowledge required.

For more information on how to set the treats for other object types see the *Data Base Elements Reference manual* for the Advant Controller 100 Series.

See *Alarm and Event Texts* on page 70 for information how to change Alarm and Events descriptions.

The OPC Standard uses severity on alarms and events. For information on how to change the default values, see *Alarm and Event Severity* on page 72.

34. Click the **OK** button.

*Figure 40. Editing AIS - MMI Part*
You return to the Create Element dialog.

Click the **OK** button in the Create Element dialog box.

During the last four steps, the DB Section dialog box is updated continuously according to your selections.

![DB Section dialog box](image)

**Figure 41. DB Section dialog box**

35. Choose **New Section > PC** from the **File** menu in the Function Chart Builder menubar, or select **PC** and click **New** in the window named AC 160_2.ODB as shown in **Figure 42**.
36. The New PC Program dialog box is displayed. Fill in the name of your new PC program (PC1) as shown in the figure and click **OK**.

![New PC Program dialog box](image)

*Figure 43. New PC Program dialog box*
37. The Create Element dialog box is displayed.
38. Click on the PCPGM item in the list box.
39. Click the **Call Parameter** button.

![Create Element dialog box](image)

*Figure 44. Create Element dialog box*
40. The Edit Call Parameters dialog box is displayed.

![Edit Call Parameters dialog box](image)

*Figure 45. Edit Call Parameters*

41. Fill in the cyclicity time, for example, 100 ms, in the displayed dialog and click **OK**.

You return to the Create Elements dialog box.

42. Click the **Apply** button.

A chart showing the items created as a result of the steps above is displayed.

![Chart displaying items created](image)

*Figure 46. Example of the Chart Displayed after Inserting the PCPGM Element*
The Create Element dialog is updated with a new set of parameters in the list box.

43. Mark the CONTRM item in the list box.

44. Click the **Call Parameter** button.

![Create Element dialog box, Second part](image)

**Figure 47. Create Element dialog box, Second part**

45. Fill in the cyclicity time, for example, 500 ms, in the displayed Edit Call Parameters dialog box and click **OK**.

   You return to the Create Elements dialog box.

46. Click the **Apply** button.
The chart showing the configured program elements is updated with the CONTRM item.

47. Click the **Close** button in the Create Elements dialog box.

48. In the Function Chart Builder application, activate the Chart window (PC Section - PC1) and click on the terminal 1 (ON), belonging to the PCPGM item, to mark it.

49. Choose the **Connect** item from the **PC-Terminal** menu in the Function Chart Builder menubar.

![Connect dialog box](image)

**Figure 48. Connect dialog box**

50. Fill in the following in the displayed dialog:

   - in the To section **D=1**
   - and click the **Apply/Next** button.
   - Fill in **D=0**
   - click the **Apply/Next** button
   - click the **Next** button
   - Fill in **D=1**
   - click the **Apply/Next** button
   - Fill in **D=0**
   - click the **Apply/Next** button
Fill in \( D=0 \) click the \textbf{OK} button.

After this procedure, the chart should look like this:

![Chart after connection procedure](image)

\textit{Figure 49. Chart after connection procedure}

51. Choose the \textbf{Save} item from the \textbf{File} menu in Function Chart Builder menubar to save the configuration.

**Compiling the controller application and filling the Bus Configuration Database**

52. You are still in the Function Chart Builder application. Choose \textbf{Generate Target Code} from the \textbf{File} menu.
53. Answer the message box “Do you want to save ...” with yes.

![Generate Target Code dialog box]

*Figure 50. Generate Target Code dialog box*

Select check boxes in the Generate Target Code dialog box, according to Figure 50.

Your application is compiled and data is written to the Bus Configuration Database. Brief information about the compiling procedure is displayed.

54. Open the Bus Configuration Builder for the project and run Action - Check Database to validate that no warning or errors exist. An inconsistent application may lead to wrong data being presented by the OPC server.

AC 100 OPC Server does not check the MDB-file for duplicate defined EVS and DSP’s.
Section 2  Configuration  Creating a Small Test Configuration Using Extended DB Elements

Figure 51. Example of Information Displayed after Target Code Generation

**Downloading to controller**

55. Choose the **Connect** item from the **Target** menu in the Function Chart Builder application.

A dialog is displayed with various connection alternatives. In the case of a totally new AF100 Interface installation and a totally new controller, use a serial communication line. Select the appropriate COMx alternative
(depending on which port you are using) and click **OK**. You can use the AF100 alternative in all future application program download operations.

After connection you get a report about the status of the connected controller in the Status Report window.

56. Choose **Load Application** from the **Target** menu.

The Function Chart Builder displays some messages about the program loading procedure’s progress.

When updating an application that has previously been loaded in the controller you must first **BLOCK PROGRAM**.

A dialog box is displayed and you are asked to confirm the downloading. Press **Y** on the keyboard for Yes, or use the arrow keys to mark the Yes alternative and then press the Enter key.

The Function Chart Builder displays some messages about the program loading procedure’s progress.

Answer the question “Do you want to proceed in on-line mode now?“ with **No**.

---

**Figure 52. Download of Application**
57. Choose the **Deblock Program** item from the **Target** menu.

58. Exit the Function Chart Builder application by choosing **File > Exit**.

   A dialog is shown which asks if you want to disconnect from the target system. Enter Y for Yes.

59. A dialog is displayed which states that the program is not saved in the PROM. Saving in the PROM is not necessary for this tutorial, so enter Y for Yes.

![Function Chart Builder #12461 dialog](function_chart_builder.png)

*Figure 53. It is not necessary to save - Enter Y for Yes*

60. Ensure that the configuration of AC 100 OPC server is done as described in **Initial Configuration** on page 33.

**Testing the communication 1**

61. Select **AC 100 Diagnostics** from the ABB Start Menu.

   A dialog box as shown in **Figure 54** is displayed, showing the network nodes connected to the AF100 bus and the node status.

   By double clicking on the status indicator of the newly configured station you can get more detailed information.

   For an explanation of the different dialog texts, see **How to View the Diagnostics** on page 63.
Testing the communication

You can try to access an object using an OPC client.

This example will show how to use the ProcessX Explorer, which is an OPC client from Matrikon Systems. At Matrikon’s web page, www.matrikon.com, you will find more information about this application.

62. Start the ProcessX Explorer.
63. Connect to the AC 100 OPC Server by selecting ABB.AC100.1 from the drop menu and then select Connect > Local from the Server menu. Verify that status for Connected changes from No to Yes.
64. Create a group by choosing **Server > Add Group**. A dialog box is displayed where you can fill in the common properties for all items in this group. Enter a name, for example, ’TestGroup’, an UpdateRate and accept the default values.

![Add Group dialog box](image)

*Figure 56. Dialog box for OPC group*

65. Create items by choosing **Group > Add Items**. A new window called TagStudio is displayed. There are several ways to define the tags you want to use as items.

- You can write the object name and attribute in the Item ID input line, following the item syntax described in *Working With AC 100 OPC Data* on page 62.

- At the bottom of the window you can see available objects in the OPC Server. Click on the object and you can see the available attributes. Double click to choose an item.

- You can use the Tag Generator which is started with the button to the right of the Item ID input line.

66. Write TEST1.DESCRIPTION in the Item ID line.

67. Click on the right arrow button to put the item in the tag list.
68. Define some more items if you like.

Figure 57. ProcessX TagStudio

69. Return to the ProcessX Explorer window by selecting **Update and Return** in the **File** menu.

70. Save your configuration by choosing **File > Save As** and enter a file name, (Myproj).
71. The items will now be updated with data from the OPC Server.

![Figure 58. OPCX Explorer with Active Items](image)

**Creating a Small Test Configuration Using DAT Based Objects**

By following the steps in this tutorial, you will create a DAT Based Analog Input Object, imposing limit checking on its value and signaling limit transgression events.
Figure 59 gives an overview of the involved objects, of the relations between them and the operations on them.

To carry out this tutorial you must have the following applications and hardware installed and working:

- AC 100 OPC Server
- CBA version 1.2/0 or higher, including Application Builder, Function Chart Builder and Bus Configuration Builder.
- An Advant controller 100 Series.

You may also find it useful to consult the Function Chart Builders User's Guide (3BDS100595*) and the applicable Data Base Element Reference Manuals (3BDS005556*, 3BDS100594*, or 3BDS100593*).
The tutorial assumes an existing installation and prior knowledge on generating target code, downloading to the controller and using an OPC client.

During the tutorial you can see where in the procedure you are by looking at the “procedure bar”, seen below, that is placed at strategic positions.

![Procedure Bar Used in the Tutorial](image)

**Starting the Function Chart Builder**

1. Choose **Open > Project** from the **File** menu in the Application builder if you want to work with another project than the one automatically presented upon starting the Application Builder.
2. Select the appropriate controller by marking it in the list box and choose the **Function Chart Builder** from the **Tools** menu in the Application Builder menubar.

3. Select **DB** and click **Open**, or choose **Open Section > DB** from the **File** menu in the FCB menubar.

### Creating a DSP with corresponding DAT objects

4. The DB Section dialog box is displayed. Click **Create** to enter the Create Element dialog box.

5. Create a Data Set Peripheral. In the list box, select the **DSP** item and click the **DB Terminal Value** button.
6. Fill in the following information (see Figure 61):
   
   Set the NAME item to an appropriate object name (DSP2).
   
   Set the SOURCE item to SEND.
   
   Set the BUS item to the actual bus number (1).
   
   Set the STATION item to a suitable number according to your scheme for grouping items (2).
   
   Set the IDENT item to the lowest free number within the station (2).
The STAT ION and IDENT parameters explicitly identify a certain DataSet Peripheral. It is imperative that two DSPs with identical STAT ION and IDENT settings do not exist in the system. No checks are made for identical DSPs in other controllers when a new one is created. It is recommended that you run the BCB application to avoid identical DSPs in your system.

7. A DSP consists of 8 32-bit DATs which each may represent a different data type. In this example we choose to handle five boolean, one long integer and two real values:
   
   NO_BREC = 5, NO_INTL = 1, NO_REAL = 2.

8. Click the OK button. When you return to the Create Element dialog click the OK button again. This will take you back to the DB Section dialog box.

9. When back in the DB Section dialog box click Edit to view the newly created element.

10. Choose the Value references item from the DataSet Peripheral’s drop-down list - and you will find that five boolean, one long integer and two real DAT elements have been created in the eight longwords of the current DSP (Figure 62).

Figure 62. Five boolean, one long integer and two real DAT elements are created
11. You can navigate to the created DATs by clicking on References, selecting the desired element and then choosing Goto (Figure 63).

![References of DSP2](image)

*Figure 63. Selecting the Desired Element*

![Edit DB Terminal Values - DAT9/DSP2.R1](image)

*Figure 64. Editing one of the Real Value Elements*

The automatically generated name can be changed by changing the NAME item of the DAT object.
Creating an Event Set to be used for signaling events and alarms

12. Click **Create** in the DB Section dialog box to enter the Create Element dialog box.

13. Create a sending Event Set. In the list box, select the **EVS(S)** item and click the **DB Terminal Value** button.

![Figure 65. Editing the Event Set](image)

14. Fill in following information:

   Set the **NAME** item to an appropriate object name (**EVS2**).
   
   Set the **IDENT** item to the lowest free EVS ident number (**1**).
   
   Click on **OK** to return to the Create Element dialog box. Click on **Apply** to effectuate the object creation as we will remain in the Create Element dialog to create some more objects.

The **IDENT** parameter explicitly identifies a certain Event Set within a controller CPU. If two EVSs with identical **IDENT** settings exist in the controller CPU **Generate Target Code** will generate an error.
Creating an MMI object

15. Create a MMI-object for Analog Input. In the list box, mark the DAT_AI item and click on the DB Terminal Value button.

![ MMI Object Configuration Image ]

Figure 66. Editing the MMI Object

16. Fill in following information, see Figure 66:

NAME = AI_TEST1 (instance name shown in DB section),
OBJ_NAME = AI_TEST1 (name shown in OPC client object browsers),
UNIT = cm,
RANGEMIN = 0,
DAT_OBJ = the NAME of the corresponding DAT object (DSP2.R1),
EVS_COM = SEND,
EVS_IDNT = the IDENT of the EVS to use (1),
EVS_REF = the EVS channel to use (1).

17. Choose the MMI1 item from the Analog Input drop-down list. Change the default values to a suitable severity, see Figure 67 below.

Figure 67. Set the Severity

You should have a plan for severity in general before you start changing the severity from the defaults (recommended by OPC foundation). It is rare that only one single object of a type has other severities than the others, normally it is a group of objects for example fire indicators.
Create a calculation object for value checking

18. Click on **Create** in the DB Section dialog box to enter the Create Element dialog box.

19. Create an Analog Input Calculated element. In the list box, mark the **AIC** item and click on the **DB Terminal Value** button.

20. Set the following items in the Base part dialogue, see Figure 68:
   - NAME = an appropriate object name (AIC_AI_TEST1).
   - NORM_TR = 1 to enable event detection.

   ![Figure 68. Editing the Analog Input Calculated (AIC) Element, Base Part](image)

21. Choose the Limit Check item from the AI Calculated drop-down list. Change the lower limits and disregard hysteresis effects (HYST = 0).
   - LO_LIM1 = 20
   - LO_LIM2 = 10

   You will need to enable event generation for each limit that you want to supervise:
EN_HI2 = 1
EN_HI1 = 1
EN_LO1 = 1
EN_LO2 = 1

Figure 69. Editing AIC Limit Check

22. Link the calculation object to an event set.

23. Edit the EVS previously created and choose the Event Chan. 1-16 item from the Event Set drop-down list (Figure 70). Enter the name of AIC (AIC_AI_TEST1) as the REF1 item.
There are 31 more slots in this event set which may be used by other objects to signal events.
Connect the DAT object to the calculated AIC object in the Function Chart

24. Select PC and click Open, or choose Open Section > PC from the File menu in the FCB menubar.

25. Create a PCPGM & CONTRM, see Step 36 to Step 50 in the tutorial in Creating a Small Test Configuration Using Extended DB Elements on page 74.

26. Select the object after which you wish to insert the Move operation by clicking on it. Choose Create from the Edit menu in the Function Chart Builder menubar.

27. Select the MOVE operation in the Create Elements dialog box (Figure 71).
Figure 71. MOVE is selected in the Create Element dialog box

28. Click the **Call Parameters** button to display the Edit Call Parameters dialog. Set the Call parameter Value to R. Click **OK** twice to return to the Function Chart.
29. Click on the connection belonging to the newly created MOVE element to mark it. Choose the Connect item from the **PC-Terminal** menu in the Function Chart Builder menubar.

![Connect dialog box](image)

*Figure 72. Connect dialog box*

30. Click the **DB** button in the Connect dialog box (*Figure 72*) to select from a list of objects. Select the DAT object DSP2.R1 (*Figure 72*) and click on **OK**. When back in the Connect dialog box click on the **Apply/Next** button which will position you to the output terminal. Select the AIC object AIC_AI_TEST1 from the DB and click **OK** to exit the Connect dialog.
Figure 73. DAT Object is Selected
Generating target code and downloading to controller

31. Generate target code for the modified application check for warnings in BCB and download it to the controller. See Step 52 to Step 60 in Creating a Small Test Configuration Using Extended DB Elements on page 74.

Testing the DAT_AI object

32. Verify that the DAT_AI object has been correctly configured by using an OPC client to change its value so that it transgresses one or more limits. Use an alarms and events client, e.g. SampleClient from OPC Foundation, to observe that the limit transgression events are reported. See Step 62 to Step 71 in Creating a Small Test Configuration Using Extended DB Elements on page 74.
Section 3  Maintenance

Preventive Maintenance

For a detailed description of CI board hardware installation, see Appendix D, Installing the PCI Board CI527A.

We recommend that you follow a backup routine to secure configuration data regularly and whenever you reconfigure the system or change object definitions.

For a general description of maintenance of the Advant Fieldbus 100 network, see Advant Fieldbus 100 User’s Guide (3BSE000506*).

Error Messages

Error messages from the AC 100 OPC Server are reported in two ways.

Software Errors:
All software errors are reported to the Windows Event Viewer.

Hardware Errors:
Error information from the CI board is listed in the Diagnostics window. The Errors information from the CI board helps determine the source of errors.

• The Permanent sender identification shows the address of the permanent sender on the bus.
• The Multiple source item identifies the address of the station where two stations answer.
• The Invalid telegram item identifies the address of the station attempting to send non-identifiable information.
• The Invalid telegram data item shows the data content of the invalid telegram identified above.
Fault Finding and User Repair

General

If your system is installed and started but not properly running, you can follow the product verification sequence of actions described in Appendix E, Product Verification.

You further can follow the steps listed below to determine the cause and the actions to take:

1. View the Windows Event Viewer to determine the source of the problem.
   - If the AC 100 OPC Server software is the source, go on to the next step. If the problem originates from one of the other software units in your system, consult the corresponding user’s guide.

2. Start the Diagnostics application and check for critical status flags and error messages.

3. Check the status of the ABB Services and CI Device driver. Check also if the Windows Event Viewer reports any problems. This is described in Check ABB Services, CI Device Driver and Event Viewer on page 124.

4. Start a suitable OPC viewer and view the System Status objects for information.

5. If the AC 100 OPC Server is not responding, restart the PC. If this has no effect, power off and power on your PC, simultaneously resetting the CI board.

6. If the above actions are not successful, contact your local ABB support representative.

Check ABB Services, CI Device Driver and Event Viewer

Make sure that the four ABB Services are indicated started in the Windows Services application and that correct service account is used. You start the Services application by selecting Computer Management.

The following services must be started:

- ABB AC 100 VFI Server
• ABB AC 100 CI52x Manager
• ABB AC 100 OPC Server

![Image of Services window with ABB AC 100 CI52x Manager, ABB AC 100 OPC Server, and ABB AC 100 VFD Server listed as Started with Manual Startup Type]

Figure 75. Active Services

It is also possible to stop these services, but as long as any OPC client is running, the AC 100 OPC server will restart immediately. It is for this reason recommended to stop all OPC clients (including System 800xA EventCollector and OPC-DA service providers) before restarting AC 100 services.

Make sure that the CI Device driver is indicated in the Windows Device Manager.
Figure 76. Windows Devices

If the AC 100 OPC Server Services do not start, the reason for this can be found in the Windows Event Viewer, group application.
In this example the problem occurred when the AC 100 OPC Server software was installed on an older version without rebooting the personal computer (PC) after the installation.

The problem was solved by rebooting the personal computer (PC)
Hints on Known Failure Situations

The following known failure cases may show during test or commissioning:

- If you connect in FCB a DAT_DO element to a DIC element (and this one to an EVS element) there is periodically an entry in the windows event log file:

  Event item does not exist for EVS from Station: x, CPU: y EVS: Inst=n, Pos=m. File: WiaEventSink.cpp

  This is a case that results from a not supported connection.

- Events on AF100 (at least from Event Sets used in the OPC Server) which are not configured at MMI DB Elements generate an entry in the Windows event log file every time the event is fired.

- After on-line changes of the application program it can happen that the MDB file gets corrupt and objects are no longer exposed. The solution is to compact the database in BCB (Bus Configuration Builder).

- If “Dynamic AIS AlarmLimits” is selected the performance of the OPC server has to be considered. It is not recommended to subscribe for more than 10 AIS objects simultaneously, if the limit properties (HI_LIM1, HI_LIM2, LO_LIM1, LO_LIM2) are included in the subscription. These properties are typically used in bargraphs. Use bargraphs without limits instead, or turn off the Dynamic AIS AlarmLimits check box in the Configuration Wizard.

- At high system load Windows Event Viewer shows errors about EVS although Event Set configurations are correct. The message “Failed to read station version” can also be reported. This could happen if service data traffic is heavy loaded. Consider to lower SDP traffic if these events are reported frequently.

- After an on-line change of DIS/DAT_DI terminal VALVE_TR from 2 to 3, an existing Alarm List entry can not be acknowledged. The entry needs to be deleted.

- Station status reported from old controllers are not supported. This affects controller versions lower than AC 160 2.1, AC 110 2.3 and AC 70 1.2.

- Ensure that the limits defined for AIS elements are within RANGRMIN/RANGEMAX, else wrong limits are presented by the OPC server.
Appendix A  Object Attributes and Events

General

The tables Table 10 to Table 41 in this appendix list for all object types:

- Attributes that can be accessed by the client application.
  Attributes are the accessible data items according to the specific AC 100 OPC Server item syntax defined as: <objectname>(.<attribute>).
  In many cases an attribute represents the value (or extracted parts of it) of a terminal of a signal or hardware DB element in the controller.
- Status bits in the attribute STATUS
- Events

Status bits additionally are exposed as own attributes.

Strings describing Event Conditions and Categories are not supported to be configured in national languages. Currently they are provided in English language only.
Attribute Name Mapping

Note that in the tables listing the attributes some object attribute names are not equal to the corresponding DB Element Terminal names but are mapped to them according to Table 8.

In all other cases, provided the object represents a DB Element and the DB Element Terminal is exposed via OPC, the attribute name equals the DB Element Terminal name.

Table 8. Mapping OPC Object Attribute Names and DB Element Terminal Names

<table>
<thead>
<tr>
<th>OPC Object Attribute Name</th>
<th>DB Element Terminal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALARM_PERIOD_BLK</td>
<td>AL_PERIOD_BLK</td>
</tr>
<tr>
<td>CPU</td>
<td>POSITION</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>DESCR, DESCR1</td>
</tr>
<tr>
<td>DESCRIPTION2</td>
<td>DESCR2</td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>ERR_TR</td>
</tr>
<tr>
<td>HWVALMAX</td>
<td>SCAL_MAX</td>
</tr>
<tr>
<td>HWVALMIN</td>
<td>SCAL_MIN</td>
</tr>
<tr>
<td>LIM1_TREAT</td>
<td>LIM_1_TR</td>
</tr>
<tr>
<td>LIM2_TREAT</td>
<td>LIM_2_TR</td>
</tr>
<tr>
<td>MAX_LIM</td>
<td>DISPMAX</td>
</tr>
<tr>
<td>MIN_LIM</td>
<td>DISPMIN</td>
</tr>
<tr>
<td>NAME</td>
<td>OBJ_NAME</td>
</tr>
<tr>
<td>NETW</td>
<td>BUSNO</td>
</tr>
<tr>
<td>NO_OF_DEC</td>
<td>DEC</td>
</tr>
<tr>
<td>NODE</td>
<td>STNNO</td>
</tr>
</tbody>
</table>
Table 8. Mapping OPC Object Attribute Names and DB Element Terminal Names (Continued)

<table>
<thead>
<tr>
<th>OPC Object Attribute Name</th>
<th>DB Element Terminal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM_POSN</td>
<td>NORM_POS</td>
</tr>
<tr>
<td>RANGE_MAX</td>
<td>RANGEMAX</td>
</tr>
<tr>
<td>RANGE_MIN</td>
<td>RANGEMIN</td>
</tr>
<tr>
<td>START_VALUE</td>
<td>INITVAL</td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>PROC_SEC</td>
</tr>
<tr>
<td>UPD_BLK</td>
<td>UPDATE_BLK</td>
</tr>
<tr>
<td>USE_HI_LIM1</td>
<td>INITENH1</td>
</tr>
<tr>
<td>USE_HI_LIM2</td>
<td>INITENH2</td>
</tr>
<tr>
<td>USE_LO_LIM1</td>
<td>INITENL1</td>
</tr>
<tr>
<td>USE_LO_LIM2</td>
<td>INITENL2</td>
</tr>
<tr>
<td>VALUE_TREAT</td>
<td>VALUE_TR</td>
</tr>
</tbody>
</table>

**Event Quality**

When the AC 100 OPC Server reports an event back to an OPC client, each event will be marked with an OPC quality. This quality of an event can be one of the four listed in Table 9 below:

Table 9. OPC Quality of Events

<table>
<thead>
<tr>
<th>OPC_Quality</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOD</td>
<td>OK</td>
</tr>
<tr>
<td>BAD</td>
<td>Queue overflow</td>
</tr>
</tbody>
</table>
AC 100 Event Categories

- **AC 100 Process Condition Event** is used for all full alarms of the listed process object types. These alarms have to be acknowledged. The unack bit is set if the alarm gets active and reset if the alarm is acknowledged.
  - AIS and DAT_AI with Limit Treat 4
  - AOS with Error Treat 2
  - DIS, DOS, DAT_DI and MBS with Value Treat 2

- **AC 100 Process Event** is used for process events (no acknowledge) of following object types.
  - AIS and DAT_AI with Limit Treat 1, 2, 3, 5, or 6
  - AOS with Error Treat 1
  - DIS, DOS, DAT_DI and MBS with Value Treat 1, 3 or 4

- **AC 100 System Condition Event** is used for all system alarms, i.e. for all alarms of the system objects. They have to be acknowledged.
  - PM, CI, IO and OPC

- **AC 100 System Simple Event** is used for all simple system events like "Alarm blocked", "Update blocked", Manual/Automatic changes etc. They don't have to be acknowledged

---

Table 9. OPC Quality of Events (Continued)

<table>
<thead>
<tr>
<th>OPC_Quality</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSOR_FAILURE</td>
<td>Hardware error</td>
</tr>
<tr>
<td>UNCERTAIN</td>
<td>Uncertain time from controller or timestamp locally created in the PC. System Messages will inform if the time stamp is locally created. Note that all DAT-based acknowledgements are locally time stamped in the PC</td>
</tr>
</tbody>
</table>

---
## Analog Input Object Type (AIS and DAT_AI)

### Attributes

The Table 10 below shows all available attributes for an analog input object.

**Table 10. Analog Input Object Type Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not available for DAT_AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION2</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = AIS or DAT_AI</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object(^{(1)})</td>
<td>REAL</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>NO_OF_DEC</td>
<td>Number of decimals</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>UNIT</td>
<td>The unit of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>Class</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>Presents the status of the object</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>RANGE_MIN</td>
<td>The smallest allowed object value</td>
<td>REAL</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>RANGE_MAX</td>
<td>The highest allowed object value</td>
<td>REAL</td>
<td>Read</td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) For analog input objects, the current value is the most recent measurement.
### Table 10. Analog Input Object Type Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not available for DAT_AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_LIM</td>
<td>Maximum value displayed on screen</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>MIN_LIM</td>
<td>Minimum value displayed on screen</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>START_VALUE</td>
<td>Initial value at controller startup</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HI_LIM2(2)</td>
<td>Alarm limit value HI_LIM2</td>
<td>REAL</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>HI_LIM1(2)</td>
<td>Alarm limit value HI_LIM1</td>
<td>REAL</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>LO_LIM1(2)</td>
<td>Alarm limit value LO_LIM1</td>
<td>REAL</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>LO_LIM2(2)</td>
<td>Alarm limit value LO_LIM2</td>
<td>REAL</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>USE_HI_LIM2</td>
<td>Alarm limit value HI_LIM2 used</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>USE_HI_LIM1</td>
<td>Alarm limit value HI_LIM1 used</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>USE_LO_LIM1</td>
<td>Alarm limit value LO_LIM1 used</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>USE_LO_LIM2</td>
<td>Alarm limit value LO_LIM2 used</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>LIM1_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>LIM2_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HWVALMAX</td>
<td>Highest value the controller hardware can process</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HWVALMIN</td>
<td>Lowest value the controller hardware can process</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>T0VAL_H1</td>
<td>String to use when high limit 1 is no longer transgressed.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>T0VAL_H2</td>
<td>String to use when high limit 2 is no longer transgressed.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
</tbody>
</table>
### Table 10. Analog Input Object Type Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not available for DAT_AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0VAL_L1</td>
<td>String to use when low limit 1 is no longer transgressed.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>T0VAL_L2</td>
<td>String to use when low limit 2 is no longer transgressed.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>T1VAL_H1</td>
<td>String to use for high limit 1 transgression.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>T1VAL_H2</td>
<td>String to use for high limit 2 transgression.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>T1VAL_L1</td>
<td>String to use for low limit 1 transgression.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>T1VAL_L2</td>
<td>String to use for low limit 2 transgression.</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SEVAL_H2</td>
<td>Severity value for high limit 2 transgressions</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SEVAL_H1</td>
<td>Severity value for high limit 1 transgressions</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SEVAL_L1</td>
<td>Severity value for low limit 1 transgressions</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SEVAL_L2</td>
<td>Severity value for low limit 2 transgressions</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>HYST</td>
<td>Hysteresis</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SEERR</td>
<td>Severity value for HW error</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>(Not Supported)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scaling of VALUE attribute

The full range defined by the engineering unit is scaled to the I/O board value range.

\[
\text{VALUE} = \left( \frac{\text{RANGEMAX} - \text{RANGE_MIN}}{\text{HWVALMAX} - \text{HWVALMIN}} \right) \times (\text{Input Value} - \text{HWVALMIN}) + \text{RANGEMIN}
\]

This yields
- \( \text{VALUE} = \text{RANGE_MIN} \) if Input value = HWVALMIN
- \( \text{VALUE} = \text{RANGEMAX} \) if Input value = HWVALMAX

The scaling is reciprocal when writing to the VALUE attribute.

Before writing the value (via SDP or DSP) it is checked against the RANGE_MIN/RANGEMAX attributes.

The limit value change generates an event which distributes the new value to all active OPC Servers. The AIS must be linked to an EVS for this to work. If you checked the Dynamic AIS Alarm Limits in the Configuration Wizard, the updated alarm limits will be visible in the OPC server. If the checkbox is not set the OPC server has to be restarted before alarm limit changes become visible.
The Table 11 below shows all available status bits for an analog input object.

*Table 11. Analog Input Object Type: Status bits in Attribute STATUS*

<table>
<thead>
<tr>
<th>Bit in Attribute STATUS</th>
<th>Bit-No. AIS</th>
<th>Bit-No. DAT_AI / Not available for DAT_AI</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>1</td>
<td>1</td>
<td>Object active, that is, ready to run</td>
<td>Read</td>
</tr>
<tr>
<td>ERROR</td>
<td>2</td>
<td>2</td>
<td>The object is in an error state</td>
<td>Read</td>
</tr>
<tr>
<td>DISTURBANCE</td>
<td>3</td>
<td>15</td>
<td>Object is in an alarm state</td>
<td>Read</td>
</tr>
<tr>
<td>UPDATE_BLK</td>
<td>4</td>
<td>x</td>
<td>The value is blocked or forced</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>5</td>
<td>13</td>
<td>Unacknowledged alarm pending</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_BLK</td>
<td>6</td>
<td>x</td>
<td>Alarms are blocked</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_PERIOD_BLK</td>
<td>7</td>
<td>x</td>
<td>Value of event set status (EV_STAT): 0: mode changes do not generate events 1: mode changes do generate events</td>
<td>Read</td>
</tr>
<tr>
<td>ERR_CTRL</td>
<td>8</td>
<td>x</td>
<td>Value of event set status (EVS_ERR): 0: HW or AMPL errors do not generate events 1: HW or AMPL errors do generate events</td>
<td>Read</td>
</tr>
<tr>
<td>ABOVE_HI_LIM2</td>
<td>17</td>
<td>9</td>
<td>Alarm limit transgressed</td>
<td>Read</td>
</tr>
<tr>
<td>ABOVE_HI_LIM1</td>
<td>18</td>
<td>10</td>
<td>Alarm limit transgressed</td>
<td>Read</td>
</tr>
<tr>
<td>BELOW_LO_LIM1</td>
<td>19</td>
<td>11</td>
<td>Alarm limit transgressed</td>
<td>Read</td>
</tr>
<tr>
<td>BELOW_LO_LIM2</td>
<td>20</td>
<td>12</td>
<td>Alarm limit transgressed</td>
<td>Read</td>
</tr>
</tbody>
</table>
Events

The Table 12 below shows all available events for an analog input object.

*Table 12. Events for the Analog Input Object Type*

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
<th>Not available for DAT_AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW_ERROR</td>
<td>Hardware error</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type</td>
<td>Category taken from the related event of the object</td>
<td></td>
</tr>
<tr>
<td>H2LIM_TRANS</td>
<td>High Limit 2 transgression (Level HH) and LIM2_TREAT = 4</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td></td>
</tr>
<tr>
<td>H2LIM_TRANS</td>
<td>High Limit 2 transgression (Level HH) and LIM2_TREAT = 1, 2, 3, 5, 6</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td></td>
</tr>
<tr>
<td>H1LIM_TRANS</td>
<td>High Limit 1 transgression (Level H) and LIM1_TREAT = 4</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td></td>
</tr>
<tr>
<td>H1LIM_TRANS</td>
<td>High Limit 1 transgression (Level H) and LIM1_TREAT = 1, 2, 3, 5, 6</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td></td>
</tr>
<tr>
<td>L1LIM_TRANS</td>
<td>Low Limit 1 transgression (Level L) and LIM1_TREAT = 4</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td></td>
</tr>
</tbody>
</table>
### Table 12. Events for the Analog Input Object Type (Continued)

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC</th>
<th>OPC Event Category</th>
<th>Not available for DAT_AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1LIM_TRANS</td>
<td>Low Limit 1 transgression (Level L) and LIM1_TREAT = 1, 2, 3, 5, 6</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td></td>
</tr>
<tr>
<td>L2LIM_TRANS</td>
<td>Low Limit 2 transgression (Level LL) and LIM2_TREAT = 4</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td></td>
</tr>
<tr>
<td>L2LIM_TRANS</td>
<td>Low Limit 2 transgression (Level LL) and LIM2_TREAT = 1, 2, 3, 5, 6</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td></td>
</tr>
<tr>
<td>H2LIM_MOD</td>
<td>High Limit 2 value changed</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>H1LIM_MOD</td>
<td>High Limit 1 value changed</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>L1LIM_MOD</td>
<td>Low Limit 1 value changed</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>L2LIM_MOD</td>
<td>Low Limit 2 value changed</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>UPDATE_BLOCK</td>
<td>Disable/enable process updates</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_BLOCK</td>
<td>Block/deblock of alarm handling</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>LOST_EVENT</td>
<td>Lost event in controller</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td></td>
</tr>
</tbody>
</table>
Analog Output Object Type (AOS and DAT_AO)

Attributes

The Table 13 below shows all available attributes for an analog output object.

Table 13. Analog Output Object Type Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not avail. for DAT_AO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION2</td>
<td>The second description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = AOS or DAT_AO</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object (1)</td>
<td>REAL</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>NO_OF_DEC</td>
<td>Number of decimals (to be presented by MMI)</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>UNIT</td>
<td>The unit of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>Class</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>Presents the status of the object</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>RANGE_MIN</td>
<td>The smallest allowed object value</td>
<td>REAL</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>RANGE_MAX</td>
<td>The highest allowed object value</td>
<td>REAL</td>
<td>Read</td>
<td></td>
</tr>
</tbody>
</table>
### Table 13. Analog Output Object Type Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not avail. for DAT_AO</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_LIM</td>
<td>The highest allowed value given as a command</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>MIN_LIM</td>
<td>The lowest allowed value given as a command</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HWVALMAX</td>
<td>Highest value the controller hardware can process</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HWVALMIN</td>
<td>Lowest value the controller hardware can process</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HI_LIM</td>
<td>Alarm limit value HI_LIM</td>
<td>REAL</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>LO_LIM</td>
<td>Alarm limit value LO_LIM</td>
<td>REAL</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>T0VAL_H1</td>
<td>String to use when high limit 1 is no longer transgressed.</td>
<td>STRING</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>T0VAL_L1</td>
<td>String to use when low limit 1 is no longer transgressed.</td>
<td>STRING</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>T1VAL_H1</td>
<td>String to use for high limit 1 transgression.</td>
<td>STRING</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>T1VAL_L1</td>
<td>String to use for low limit 1 transgression.</td>
<td>STRING</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SEVAL_H1</td>
<td>Severity value for high limit transgressions</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SEVAL_L1</td>
<td>Severity value for low limit transgressions</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>HYST</td>
<td>Hysteresis</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SEERR</td>
<td>Severity value for HW error</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
</tbody>
</table>
Table 13. Analog Output Object Type Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not avail. for DAT_AO</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITENHI</td>
<td>Initial value for EN_HI</td>
<td>BOOLEAN</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>0 = disable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = enable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITENLO</td>
<td>Initial value for EN_LO</td>
<td>BOOLEAN</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>0 = disable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = enable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>START_VALUE</td>
<td>Initial value at controller startup</td>
<td>REAL</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected.</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>PRINT_BLK</td>
<td>Block alarm printouts.</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>UPDATED</td>
<td>Indicates if the object value is updated</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>OUTP_BLK</td>
<td>Output to output connection is blocked.</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
</tbody>
</table>

(1) **Scaling of VALUE attribute**

The full range defined by the engineering unit is scaled to the I/O board value range.

\[
VALUE = \left( \frac{\text{RANGEMAX} - \text{RANGEMIN}}{\text{HWVALMAX} - \text{HWVALMIN}} \right) (Output Value - \text{HWVALMIN}) + \text{RANGEMIN}
\]

This yields
- VALUE = RANGEMIN if Output value = HWVALMIN
- VALUE = RANGEMAX if Output value = HWVALMAX

The scaling is reciprocal when writing to the VALUE attribute.

Before writing the value (via SDP or DSP) it is checked against the RANGEMIN/RANGEMAX attributes.
The Table 14 below shows all available status bits for an analog output object.

**Table 14. Analog Output Object Type: Status bits in Attribute STATUS**

<table>
<thead>
<tr>
<th>Bit in Attribute STATUS</th>
<th>Bit-No. AOS</th>
<th>Bit-No. DAT_AO / Not available for DAT_AO</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>1</td>
<td>1</td>
<td>Object active, i.e. ready to run</td>
<td>Read</td>
</tr>
<tr>
<td>ERROR</td>
<td>2</td>
<td>2</td>
<td>The object is in an error state</td>
<td>Read</td>
</tr>
<tr>
<td>DISTURBANCE</td>
<td>3</td>
<td>x</td>
<td>Object is in an alarm state</td>
<td>Read</td>
</tr>
<tr>
<td>MAN_MODE</td>
<td>4</td>
<td>x</td>
<td>Object is in manual mode</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>5</td>
<td>x</td>
<td>Unacknowledged alarm pending</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_BLK</td>
<td>6</td>
<td>x</td>
<td>Alarms are blocked</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_PERIOD_BLK</td>
<td>7</td>
<td>x</td>
<td>Value of event set status (EV_STAT): 0: mode changes do not generate events 1: mode changes do generate events</td>
<td>Read</td>
</tr>
<tr>
<td>ERR_CTRL</td>
<td>8</td>
<td>x</td>
<td>Value of event set status (EVS_ERR): 0: HW or AMPL errors do not generate events 1: HW or AMPL errors do generate events</td>
<td>Read</td>
</tr>
<tr>
<td>ON_MAX_LIM</td>
<td>9</td>
<td>x</td>
<td>Signals that output value is above high limit</td>
<td>Read</td>
</tr>
<tr>
<td>ON_MIN_LIM</td>
<td>12</td>
<td>x</td>
<td>Signals that output value is below low limit</td>
<td>Read</td>
</tr>
</tbody>
</table>
Table 14. Analog Output Object Type: Status bits in Attribute STATUS (Continued)

<table>
<thead>
<tr>
<th>Bit in Attribute STATUS</th>
<th>Bit-No. AOS</th>
<th>Bit-No. DAT_AO / Not available for DAT_AO</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVE_HI_LIM1</td>
<td>18</td>
<td>x</td>
<td>Alarm limit transgressed</td>
<td>Read</td>
</tr>
<tr>
<td>BELOW_LO_LIM1</td>
<td>19</td>
<td>x</td>
<td>Alarm limit transgressed</td>
<td>Read</td>
</tr>
</tbody>
</table>

Events

The Table 15 below shows all available events for an analog output object.

Table 15. Events for the Analog Output Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
<th>Not available for DAT_AO</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW_ERROR</td>
<td>Hardware error</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td>Category taken from the related event of the object</td>
<td>x</td>
</tr>
<tr>
<td>H1LIM_TRANS</td>
<td>High Limit transgression and ERROR_TREAT=2</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>H1LIM_TRANS</td>
<td>High Limit transgression and ERROR_TREAT=1</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td>x</td>
</tr>
<tr>
<td>AC 100 Event</td>
<td>Description</td>
<td>OPC Event Type</td>
<td>OPC Event Category</td>
<td>Not available for DAT_AO</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>----------------</td>
<td>-----------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>L1LIM_TRANS</td>
<td>Low Limit transgression and ERROR_TREAT=2</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>L1LIM_TRANS</td>
<td>Low Limit transgression and ERROR_TREAT=1</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td>x</td>
</tr>
<tr>
<td>MAN_AUTO</td>
<td>Manual/auto mode changes</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_BLOCK</td>
<td>Block/deblock of alarm handling</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>LOST_EVENT</td>
<td>Lost event in controller</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
</tbody>
</table>

*Table 15. Events for the Analog Output Object Type*
Digital Input Object Type (DIS and DAT_DI)

Attributes

The Table 16 below shows all available attributes for a digital input object.

Table 16. Digital Input Object Type Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
<th>Not available for DAT_DI</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION2</td>
<td>The second description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = DIS or DAT_DI</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>VALUE_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NORM_POS</td>
<td>Normal position</td>
<td>BOOLEAN</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
</tbody>
</table>
Up to 32 DAT_DI objects can be mapped to a single DAT by using the INDEX attribute in the CBA to address specific bits in the DAT.

The Table 17 below shows all available status bits for a digital input object.
<table>
<thead>
<tr>
<th>Bit in STATUS</th>
<th>Bit-No. DIS</th>
<th>Bit-No. DAT_DI / Not available for DAT_DI</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>17</td>
<td>1</td>
<td>Object active, i.e. ready to run</td>
<td>Read</td>
</tr>
<tr>
<td>ERROR</td>
<td>18</td>
<td>2</td>
<td>The object is in an error state</td>
<td>Read</td>
</tr>
<tr>
<td>DISTURBANCE</td>
<td>19</td>
<td>15</td>
<td>Object is in an alarm state</td>
<td>Read</td>
</tr>
<tr>
<td>UPDATE_BLK</td>
<td>20</td>
<td>x</td>
<td>The value is blocked by OPC Server operator or forced by CBA.</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>21</td>
<td>13</td>
<td>Unacknowledged alarm pending</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_BLK</td>
<td>22</td>
<td>x</td>
<td>Alarms are blocked</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_PERIOD _BLK</td>
<td>23</td>
<td>x</td>
<td>Value of event set status (EV_STAT): 0: mode changes do not generate events 1: mode changes do generate events</td>
<td>Read</td>
</tr>
<tr>
<td>ERR_CTRL</td>
<td>24</td>
<td>x</td>
<td>Value of event set status (EVS_ERR): 0: HW or AMPL errors do not generate events 1: HW or AMPL errors do generate events</td>
<td>Read</td>
</tr>
<tr>
<td>VALUE</td>
<td>25</td>
<td>x</td>
<td>Current value</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>
Events

The Table 18 below shows all available events for a digital input object.

Table 18. Events for the Digital Input Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
<th>Not available for DAT_DI</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW_ERROR</td>
<td>Hardware error</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>VALUE_CHANGE</td>
<td>Value changes and VALUE_TREAT=2</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td></td>
</tr>
<tr>
<td>VALUE_CHANGE</td>
<td>Value changes and VALUE_TREAT=1,3,4</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td></td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE_BLOCK</td>
<td>Disable/enable process updates</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_BLOCK</td>
<td>Block/deblock of alarm handling</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>LOST_EVENT</td>
<td>Lost event in controller</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td></td>
</tr>
</tbody>
</table>

Digital Output Object Type (DOS and DAT_DO)

Attributes

The Table 19 below shows all available attributes for a digital output object.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>DataType</th>
<th>Permissions</th>
<th>Not available for DAT_DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION2</td>
<td>The second description of the object</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = DOS or DAT_DO</td>
<td>STRING</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>VALUE_TREAT</td>
<td>Value of event treatment definition</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>START_VALUE</td>
<td>Start value of the object</td>
<td>BOOLEAN</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>NORM_POS</td>
<td>Normal position</td>
<td>BOOLEAN</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>CLASS</td>
<td>Class</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
</tbody>
</table>
Table 19. Digital Output Object Type Attributes (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>DataType</th>
<th>Permissions</th>
<th>Not available for DAT.DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>Presents the status of the object</td>
<td>INTEGER</td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LONG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0VAL</td>
<td>Text to display in event description when value equals NORM_POS</td>
<td>STRING</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>T1VAL</td>
<td>Text to display in event description when value differs from NORM_POS</td>
<td>STRING</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SEVAL</td>
<td>Severity value for deviation from normal position</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SEERR</td>
<td>Severity value for HW error</td>
<td>INTEGER</td>
<td>Read</td>
<td>x</td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected.</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>(Not Supported)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINT_BLK</td>
<td>Block alarm printouts.</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>UPDATED</td>
<td>Indicates if the object value is updated</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td>x</td>
</tr>
<tr>
<td>OUTP_BLK</td>
<td>Output to output connection is blocked.</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
<td>x</td>
</tr>
</tbody>
</table>

Up to 32 DAT.DO objects can be mapped to a single DAT by using the INDEX attribute in the CBA to address specific bits in the DAT.

The Table 20 below shows all available status bits for a digital output object.
### Table 20. Digital Output Object Type: Status bits in Attribute STATUS

<table>
<thead>
<tr>
<th>Bit in STATUS</th>
<th>Bit-No. DOS</th>
<th>Bit-No. DAT_DO / Not available for DAT_DO</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>17</td>
<td>1</td>
<td>Object active, i.e. ready to run</td>
<td>Read</td>
</tr>
<tr>
<td>ERROR</td>
<td>18</td>
<td>2</td>
<td>The object is in an error state</td>
<td>Read</td>
</tr>
<tr>
<td>DISTURBANCE</td>
<td>19</td>
<td>x</td>
<td>Object is in an alarm state</td>
<td>Read</td>
</tr>
<tr>
<td>MAN_MODE</td>
<td>20</td>
<td>x</td>
<td>Object is in manual mode</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>21</td>
<td>x</td>
<td>Unacknowledged alarm pending</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ALARM_BLK</td>
<td>22</td>
<td>x</td>
<td>Alarms are blocked</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>
| ALARM_PERIOD_BLK| 23          | x                                        | Value of event set status (EV_STAT):
0: mode changes do not generate events
1: mode changes do generate events | Read        |
| ERR_CTRL        | 24          | x                                        | Value of event set status (EVS_ERR):
0: HW or AMPL errors do not generate events
1: HW or AMPL errors do generate events | Read        |
| VALUE           | 25          | x                                        | Value                                                                       | Read/Write  |
Events

The Table 21 below shows all available events for a digital output object.

Table 21. Events for the Digital Output Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
<th>Not available for DAT.DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW_ERROR</td>
<td>Hardware error</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>VALUE_CHANGE</td>
<td>Value changes and ERROR_TREAT=2</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
<td>x</td>
</tr>
<tr>
<td>VALUE_CHANGE</td>
<td>Value changes and ERROR_TREAT=1</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td>Event category taken from the related event of the object</td>
<td>x</td>
</tr>
<tr>
<td>MAN_AUTO</td>
<td>Manual/auto mode changes</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>ALARM_BLOCK</td>
<td>Block/deblock of alarm handling</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
<tr>
<td>LOST_EVENT</td>
<td>Lost event in controller</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
<td>x</td>
</tr>
</tbody>
</table>

Simple Memory Object Types (DAT(x), MB, MI, MIL, MR)

Attributes

Attributes for DAT_DAT/DAT(x)

The Table 22 below shows all available attributes for an object DAT(X) exposed to MMI via DAT_DAT object.
### Table 22. Attributes for the DAT(x) Object Types

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CPU</td>
<td>Position</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>Object active, i.e. ready to run</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>VAL_TYPE</td>
<td>Value type</td>
<td>INTEGER LONG</td>
<td>Read</td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object</td>
<td>INTEGER INTEGER LONG REAL</td>
<td>Read/Write</td>
</tr>
<tr>
<td>B0_VAL</td>
<td>Boolean value 0</td>
<td>BOOLEAN</td>
<td>(1) Read/Write</td>
</tr>
<tr>
<td>B1_VAL</td>
<td>Boolean value 1</td>
<td>BOOLEAN</td>
<td>Read</td>
</tr>
<tr>
<td>. .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B31_VAL</td>
<td>Boolean value 31</td>
<td>BOOLEAN</td>
<td>Read</td>
</tr>
</tbody>
</table>

(1) For DAT(R) no bits are used. For DAT(B) only the first bit is used.

### Attributes for MB

The Table 23 below shows all available attributes for a simple memory object MB.
Appendix A  Object Attributes and Events

Events

Table 23. Attributes for the MB Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = MB</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>Object active, i.e. ready to run</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object</td>
<td>INTEGER LONG</td>
<td>Read/Write</td>
</tr>
<tr>
<td>B0_VAL</td>
<td>Boolean value 0</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
</tr>
<tr>
<td>B1_VAL</td>
<td>Boolean value 1</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
</tr>
<tr>
<td>B2_VAL</td>
<td>Boolean value 2</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
</tr>
<tr>
<td>B31_VAL</td>
<td>Boolean value 31</td>
<td>BOOLEAN</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>

Attributes for MI, MIL and MR

The Table 24 below shows all available attributes for a simple memory objects MI, MIL and MR.

Events

The simple memory object type support no events (except for the MBS object type)
### Table 24. Attributes for the MI, MIL and MR Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>= MI, MIL or MR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>VALUE</td>
<td>The current value of the object</td>
<td>MI: INTEGER,</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIL: INTEGER,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LONG,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MR: REAL</td>
<td></td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>Object active, i.e. ready to run</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>(Not Supported)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO_OF_DEC</td>
<td>Number of decimals (to be presented by MMI)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNIT</td>
<td>The unit of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>RANGE_MAX</td>
<td>The highest allowed value given as a command</td>
<td>REAL</td>
<td>Read</td>
</tr>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RANGE_MIN</td>
<td>The lowest allowed value given as a command</td>
<td>REAL</td>
<td>Read</td>
</tr>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Available only for MR.
Simple Memory Object Type (MBS)

Attributes

The Table 25 below shows all available attributes for an MBS object.

Table 25. Attributes for the MBS Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = MBS</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CPU</td>
<td>CPU position</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>VALUE_TREAT</td>
<td>Value of event treatment definitions</td>
<td>INTEGER LONG</td>
<td>Read</td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Process section 0-16</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CLASS</td>
<td>Class</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NORM_POS</td>
<td>Normal position</td>
<td>BOOLEAN</td>
<td>Read</td>
</tr>
<tr>
<td>T0VAL</td>
<td>Text to display in event description when value equals NORM_POS</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>T1VAL</td>
<td>Text to display in event description when value differs from NORM_POS</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>SEVAL</td>
<td>Severity value for deviation from normal position</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>
Table 25. Attributes for the MBS Object Type (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>Presents the status of the object</td>
<td>INTEGER LONG</td>
<td>Read</td>
</tr>
<tr>
<td>SELECTED</td>
<td>Object selected</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>

The Table 26 below shows all available status bits for a MBS object.

Table 26. MBS Object Type: Status bits in Attribute STATUS

<table>
<thead>
<tr>
<th>Bit in STATUS</th>
<th>Bit-No.</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>1</td>
<td>Object active, i.e. ready to run</td>
<td>Read</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>13</td>
<td>Unacknowledged alarm pending</td>
<td>Read/Write</td>
</tr>
<tr>
<td>DISTURBANCE</td>
<td>15</td>
<td>Object is in an alarm state</td>
<td>Read/Write</td>
</tr>
<tr>
<td>VALUE</td>
<td>0</td>
<td>Current value</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>

Events

The Table 27 below shows all available events for an MBS object.

Table 27. Events for the MBS Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE_CHANGE</td>
<td>Value changes and VALUE_TREAT=2</td>
<td>Condition</td>
<td>AC 100 Process Condition Event</td>
</tr>
<tr>
<td>VALUE_CHANGE</td>
<td>Value changes and VALUE_TREAT=1,3,4</td>
<td>Condition</td>
<td>AC 100 Process Event</td>
</tr>
</tbody>
</table>
### Table 27. Events for the MBS Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOST_EVENT</td>
<td>Lost event in controller</td>
<td>Simple</td>
<td>AC 100 System Simple Event</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td>Event category taken from the related event of the object</td>
</tr>
</tbody>
</table>
System Status Object Types (PM, PM_SUM, CI, CI_SUM, I/O, OPC)

Processor Module (PM) Attributes

The Table 28 below shows all available attributes for an PM object.

Table 28. Attributes for the PM Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The description of the object</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = PM</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CPU</td>
<td>Position of processor module (static) of this PM</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>DEVICE_ID</td>
<td>Controller type</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>WITH_BAT</td>
<td>Power Backup</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>RPOS1</td>
<td>Position of redundancy processor module 1</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>RPOS2</td>
<td>Position of redundancy processor module 2</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>REDUNDAN</td>
<td>Redundancy (NO/YES)</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>STATUS</td>
<td>Extended Status of this PM</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>STATUS_X</td>
<td>Status of this PM as hex string</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>APP_ID</td>
<td>Controller base software, including version</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>O_SUPPRESS_ERRORS</td>
<td>If set no status information is to be reported for the object</td>
<td>INTEGER</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>
Table 28. Attributes for the PM Object Type (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_LOAD</td>
<td>System load (in %) retrieved from user specified DAT_OBJ.</td>
<td>REAL</td>
<td>Read</td>
</tr>
<tr>
<td>APP_TR</td>
<td>Treat for application version event.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SEAPP</td>
<td>Severity for application version event.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Error Treat</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SEERR</td>
<td>Severity for errors</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_STATUS</td>
<td>Status code (ok=0, error=1, warning=2)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_TIME</td>
<td>Time when current status was set. Given as UTC time.</td>
<td>DATE</td>
<td>Read</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Unacknowledged alarm pending</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>APP_VERSION_AC</td>
<td>Version (=date) of application as actually loaded in this PM</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>APP_VERSION_BCD</td>
<td>Version of application for this PM as actually present in BCD</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>APP_VERSION_OPC</td>
<td>Version of application for this PM as currently loaded in OPC Server</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>AVC_AC(^{(1)})</td>
<td>Application version change. =1, if APP_VERSION_BCD != APP_VERSION_AC</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>AVC_BCD(^{(1)})</td>
<td>Application version change. =1, if APP_VERSION_OPC != APP_VERSION_BCD</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>
The attribute AVC from AF 100 OPC Server 3.0 is no longer provided. It is substituted by AVC_AC and AVC_BCD. The Application Version Change (AVC) module of the OPC Server monitors the time stamps of the BCD and the controller (AC). AVC compares them to the own actually loaded timestamp (last load from BCD). These time stamps are available at any time in the attributes APP_VERSION_BCD, APP_VERSION_AC and APP_VERSION_OPC.

If OPC Server recognizes any differences, it sets the corresponding bits in the AVC attributes:

- If APP_VERSION_BCD and APP_VERSION_OPC differ, AVC_BCD is set.
- If APP_VERSION_BCD and APP_VERSION_AC differ, AVC_AC is set.

In addition, the events are generated as described in Table 30.

**Table 29. PM Object Type: Status bits in Attribute STATUS**

<table>
<thead>
<tr>
<th>Bit in STATUS</th>
<th>Bit-No.</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>0</td>
<td>Object active, i.e. ready to run</td>
<td>Read</td>
</tr>
<tr>
<td>SE</td>
<td>1</td>
<td>PM error prevents application from executing, e.g. Address error, RAM check error, Process instruction set error, CRC error in system flash EPROM.</td>
<td>Read</td>
</tr>
<tr>
<td>HWE</td>
<td>2</td>
<td>Minor PM hardware error which does not prevent application from execution, e.g. Battery missing or battery power down.</td>
<td>Read</td>
</tr>
<tr>
<td>SWE</td>
<td>3</td>
<td>Software error which prevents part of the application from executing, e.g. CRC error in application flash EPROM.</td>
<td>Read</td>
</tr>
<tr>
<td>GE</td>
<td>14</td>
<td>General station error. (Unspecified)</td>
<td>Read</td>
</tr>
<tr>
<td>GW</td>
<td>15</td>
<td>General station warning. (Unspecified)</td>
<td>Read</td>
</tr>
</tbody>
</table>
### Processor Module (PM) Events

The Table 30 below shows all available events for an PM object.

**Table 30. Events for the PM Object Type**

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMEJITTER</td>
<td>Problems with wavering time synchronization on bus</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>LOST_EVENT</td>
<td>Lost event in controller</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>AVC_BCD</td>
<td>Application Version Change Event generation see Table 31 below</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>AVC_AC</td>
<td>Application Version Change Event generation see Table 31 below</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>ACT</td>
<td>Application is stopped/executing normally again in the controller</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>SE</td>
<td>PM error prevents application from executing/PM error resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>HW_ERROR</td>
<td>Nonfatal minor PM hardware error occurred/resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>SW_ERROR</td>
<td>Software error occurred/resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>GE</td>
<td>(Unspecified) General station error occurred/resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>GW</td>
<td>(Unspecified) General station warning occurred/resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td>Category taken from the related event of the object</td>
</tr>
</tbody>
</table>
### Table 31. Application Change Event Generation

<table>
<thead>
<tr>
<th>Case</th>
<th>Action</th>
<th>Attribute Settings</th>
<th>Generated PM Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_VERSION_OPC != APP_VERSION_BCD</td>
<td>Starts updating internal OPC cache</td>
<td>AVC_BCD=1</td>
<td>AVC_BCD</td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td>AVC_AC=1</td>
<td>AVC_AC</td>
</tr>
<tr>
<td>'AVC: OPC unequal BCD'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'AVC: BCD unequal AC'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP_VERSION_OPC == (again)</td>
<td>Finishes updating OPC cache</td>
<td>AVC_BCD=0</td>
<td>AVC_BCD</td>
</tr>
<tr>
<td>APP_VERSION_BCD</td>
<td></td>
<td>AVC_AC=1</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'AVC: OPC matches BCD'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP_VERSION_OPC == (after download)</td>
<td>Verifies all APPs are equal</td>
<td>AVC_BCD=0</td>
<td>AVC_AC</td>
</tr>
<tr>
<td>APP_VERSION_AC</td>
<td></td>
<td>AVC_AC=0</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'AVC: BCD matches AC'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Event message texts may be customized using the “Alarms & Events Texts for AF100 OPC” tool in the Bus Configuration Builder of CBA similar as shown in Figure 78.

![Figure 78. Setting Alarm & Event Texts for the PM Object Type](image-url)
AC 100 Processor Module Summary Status (PM_SUM) Attributes

The PM_SUM object is an aggregate object representing all Processor Modules in an AF100 Station. Its status attributes are therefore aggregated from the status of the PM modules present.

The Table 32 below shows all available attributes for an PM_SUM object.

Table 32. Attributes for the PM_SUM Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Object name.</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Generated by OPC Server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;&lt;NETW&gt; &lt;NODE&gt; PM_SUM&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Object description:</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Generated by OPC Server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;PM summary status node &lt;NODE&gt;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = PM_SUM</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>STATUS</td>
<td>Extended Status of this PM_SUM</td>
<td>INTEGER LONG</td>
<td>Read</td>
</tr>
<tr>
<td>STATUS_X</td>
<td>Status of this PM_SUM as hex string</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>S_STATUS</td>
<td>Status code (ok=0, error=1, warning=2)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_TIME</td>
<td>Time when current status was set. Given as UTC time.</td>
<td>DATE</td>
<td>Read</td>
</tr>
<tr>
<td>O_SUPPRESS_ERRORS</td>
<td>If set no status information is to be reported for the object</td>
<td>INTEGER</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>
Table 33. PM_SUM Object Type: Status bits in Attribute STATUS

<table>
<thead>
<tr>
<th>Bit in STATUS</th>
<th>Bit-No.</th>
<th>Description</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>0</td>
<td>Set if all represented PMs have ACT=1 (Application is executing normally in the controller (deblocked)).</td>
<td>Read</td>
</tr>
</tbody>
</table>
Communication Module (CI) Attributes

The Table 34 below shows all available attributes for a CI object.

*Table 34. Attributes for the CI Object Type*

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Object name. Set by OPC server for OPC server nodes.</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Object description. Set by OPC Server for OPC server nodes.</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = CI</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CPU</td>
<td>Position</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>REDUND</td>
<td>Redundancy (0 = disabled / 1 = enabled)</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DEVICE_ID</td>
<td>CI Board type. Not available for remote OPC server nodes</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>BUSLEN</td>
<td>Bus Length in meters.(PCI board on local node only).</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>O_SUPRESS_ERRORS</td>
<td>If set no status information is to be reported for the object.</td>
<td>INTEGER</td>
<td>Read/Write</td>
</tr>
<tr>
<td>BUSADMIN</td>
<td>Bus Master. Retrieved from BCD for controllers, in runtime using status CDP for OPC server nodes.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>TIMESYNC</td>
<td>Time synchronization mode. Retrieved from the BCD for controllers, in runtime using status CDP for OPC server nodes. Valid values are {“None”, ”Master”, ”Backup”, ”Slave”}</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>UP_TR1)</td>
<td>Treat for availability events. Set to 1 for OPC server nodes.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>
Table 34. Attributes for the CI Object Type (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEUP</td>
<td>Severity for availability events.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>
| ERROR_TREAT     | Error Treat.  
1 = Event  
2 = Alarm  
Default: 1  
Set to 1 (=event) for OPC Server nodes \(^{(1)}\). | INTEGER   | Read        |
| SEERR           | Severity for errors.  
Default: 102\(^{(1)}\)                                                       | INTEGER   | Read        |
| S_STATUS        | Status code (ok=0, error=1, warning=2)                                       | INTEGER   | Read        |
| S_TIME          | Time when current status was set. Given as UTC time.                         | DATE      | Read        |
| ALARM_UNACK     | Unacknowledged alarm pending                                                  | INTEGER   | Read        |
| ACTIVE          | CI board (& node) is available.                                              | INTEGER   | Read        |
| BOARD_STATE     | State of CI board. Valid values for local node are {"Operational", "Ready", "Not ready", "Malfunctioning"}.  
Valid values for remote nodes are {"Operational", "Malfunctioning"} | STRING    | Read        |
| CRA\(^{(2)}\)   | Cable redundancy active.                                                      | BOOLEAN   | Read        |
| RC1\(^{(2)}\)   | Redundant cable 1 failed.                                                     | BOOLEAN   | Read        |
| RC2\(^{(2)}\)   | Redundant cable 2 failed                                                      | BOOLEAN   | Read        |
(1) For OPC Server objects, this Attribute reside in Windows registry key: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ABB\Advant\SystemModules\AF100 OPC Server\Private

(2) For CI object no STATUS attribute is available, the status bits CRA, RC1, RC2 are only exposed as separate attributes
### Communication Module (CI) Events

The Table 35 below shows all available events for an CI object.

*Table 35. Events for the CI Object Type*

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODEUP</td>
<td>Station is down/up</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>VFI_ERROR</td>
<td>Communication failure occurred</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>RC1</td>
<td>Redundant cable 1 failed/working again</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>RC2</td>
<td>Redundant cable 2 failed/working again</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td>Category taken from the related event of the object</td>
</tr>
</tbody>
</table>
Event message texts may be customized using the “Alarms & Events Texts for AF100 OPC” tool in the Bus Configuration Builder of CBA similar as shown in Figure 79 below.

![Alarm & Event Texts for AF100 OPC](image)

*Figure 79. Setting Alarm & Event Texts for the CI Object Type*
AC 100 AF100 CI Module Summary Status (CI_SUM) Attributes

The CI_SUM object is an aggregate object representing all AF 100 Bus Couplers (CI626, CI627, CI630, CI631, PM810,...) in an AF 100 station. Its status attributes are therefore aggregated from the status of the CI / PM modules present.

The Table 34 below shows all available attributes for a CI_SUM object.

Table 36. Attributes for the CI_SUM Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Object name. Generated by OPC Server: &quot;&lt;NETW&gt; &lt;NODE&gt; CI_SUM&quot;</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Object description. Generated by OPC server: &quot;CI summary status node &lt;NODE&gt;&quot;</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = CI_SUM</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>CI board (&amp; node) is available.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_STATUS</td>
<td>Status code (ok=0, error=1, warning=2)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_TIME</td>
<td>Time when current status was set. Given as UTC time.</td>
<td>DATE</td>
<td>Read</td>
</tr>
<tr>
<td>O_SUPPRESS_ERRORS</td>
<td>If set no status information is to be reported for the object.</td>
<td>INTEGER</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>
AC 100 AF100 CI Module Summary Status (CI_SUM) Events

The Table 35 below shows all available events for a CI_SUM object.

Table 37. Events for the CI_SUM Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODEUP</td>
<td>Station is down/up</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>VFI_ERROR</td>
<td>Communication failure occurred</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>RC1</td>
<td>Redundant cable 1 failed/working again</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>RC2</td>
<td>Redundant cable 2 failed/working again</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Text taken from the related event of the object</td>
<td>Type taken from the related event of the object</td>
<td>Category taken from the related event of the object</td>
</tr>
</tbody>
</table>

Event message texts may be customized using the “Alarms & Events Texts for AF100 OPC” tool in the Bus Configuration Builder of CBA similar as shown in Figure 79.
AC 100 IO CI Summary Status (IO) Attributes

The AC 100 IO CI Summary (IO) object is an aggregate object representing all IO and CI modules of an AF 100 station. It exists even if the node does not actually have any IO modules. Its status attributes are therefore aggregated from the status of the IO and CI modules present. The table below shows all available attributes for an IO object.

Table 38. Attributes for the I/O Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Object name. Generated by OPC Server: “&lt;NETW&gt; &lt;NODE&gt; IO”</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Object description. Generated by OPC Server: “IO subsystem node &lt;NODE&gt;”</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Object type = IO</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>O_SUPRESS_ERRORS</td>
<td>If set no status information is to be reported for the object.</td>
<td>INTEGER</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Error Treat. Set to 1 = event</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SEERR</td>
<td>Severity for errors. Set to 102</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_STATUS</td>
<td>Status code (ok=0, error=1, warning=2)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_TIME</td>
<td>Time when current status was set. Given as UTC time.</td>
<td>DATE</td>
<td>Read</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Unacknowledged alarm pending</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>
The Table 39 below shows all available events for an I/O object.

Table 39. Events for the I/O Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEE</td>
<td>IO device error occurred/resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
<tr>
<td>PRE</td>
<td>I/O process error occurred/resolved</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
</tbody>
</table>
Event message texts may be customized using the “Alarms & Events Texts for AF100 OPC” tool in the Bus Configuration Builder of CBA similar as shown in Figure 80 below.

![Alarm & Event Texts for AF100 OPC](image)

Figure 80. Setting Alarm & Event Texts for the I/O Object Type
## OPC Station (OPC) Attributes

The Table 40 below shows all available attributes for an OPC object.

### Table 40. Attributes for the OPC Object Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Object name. Set by OPC Server</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Object description. Set by OPC Server</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>TYPE</td>
<td>Status object type = OPC</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>NETW</td>
<td>Bus number</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>NODE</td>
<td>Station</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>APP_ID</td>
<td>OPC Server software, including version</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>OPC_DAC_VERS</td>
<td>Version of the OPC standard supported for Data Access using the Custom interface. Available for local node only</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>OPC_DAA_VERS</td>
<td>Version of the OPC standard supported for Data Access using the Automation interface. Available for local node only</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>OPC_AEC_VERS</td>
<td>Version of the OPC standard supported for Alarm &amp; Events using the Custom interface. Available for local node only</td>
<td>STRING</td>
<td>Read</td>
</tr>
</tbody>
</table>
Table 40. Attributes for the OPC Object Type (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC_AEAVERS</td>
<td>Version of the OPC standard supported for Alarm &amp; Events using the Automation interface. Available for local node only</td>
<td>STRING</td>
<td>Read</td>
</tr>
<tr>
<td>O_SUPRESS_ERRORS</td>
<td>If set no status information is to be reported for the object.</td>
<td>INTEGER</td>
<td>Read/Write</td>
</tr>
<tr>
<td>ERROR_TREAT</td>
<td>Error Treat. 1 = Event 2 = Alarm Default: 1 Set to 1 (=event) for OPC Server nodes (1).</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>SEERR</td>
<td>Severity for errors. Set to 102 (1)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_STATUS</td>
<td>Status code (ok=0, error=1, warning=2)</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>S_TIME</td>
<td>Time when current status was set. Given as UTC time.</td>
<td>DATE</td>
<td>Read</td>
</tr>
<tr>
<td>ALARM_UNACK</td>
<td>Unacknowledged alarm pending</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>ACT</td>
<td>AC 100 OPC Server is up-and-running.</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_TRUE</td>
<td>Constant provided for programming support</td>
<td>BOOLEAN</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_FALSE</td>
<td>Constant provided for programming support</td>
<td>BOOLEAN</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ONE_I</td>
<td>Constant provided for programming support</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
</tbody>
</table>
Table 40. Attributes for the OPC Object Type (Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Data Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONST_ONE_IL</td>
<td>Constant provided for programming support</td>
<td>INTEGER LONG</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ZERO_I</td>
<td>Constant provided for programming support</td>
<td>INTEGER</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ZERO_IL</td>
<td>Constant provided for programming support</td>
<td>INTEGER LONG</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ONE_F</td>
<td>Constant provided for programming support</td>
<td>REAL (FLOAT)</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ZERO_F</td>
<td>Constant provided for programming support</td>
<td>REAL (FLOAT)</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ONE_D</td>
<td>Constant provided for programming support</td>
<td>DOUBLE</td>
<td>Read</td>
</tr>
<tr>
<td>CONST_ZERO_D</td>
<td>Constant provided for programming support</td>
<td>DOUBLE</td>
<td>Read</td>
</tr>
</tbody>
</table>

(1) For OPC Server objects, this Attribute reside in Windows registry key:
    HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ABB\Advant\SystemModules\AF100
    OPC Server\Private

**OPC Station (OPC) Events**

The Table 41 below shows all available events for an OPC object.

Table 41. Events for the OPC Object Type

<table>
<thead>
<tr>
<th>AC 100 Event</th>
<th>Description</th>
<th>OPC Event Type</th>
<th>OPC Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC</td>
<td>AC 100 OPC Server stopped /restarted</td>
<td>Condition</td>
<td>AC 100 System Condition Event</td>
</tr>
</tbody>
</table>
Event message texts may be customized using the “Alarms & Events Texts for AF100 OPC” tool in the Bus Configuration Builder of CBA similar as shown in Figure 81 below.

![Figure 81. Setting Alarm & Event Texts for the OPC Object Type](image-url)
To send commands via DSPs as outlined in subsection **DSPs Sent by the AC 100 OPC Server** on page 45 the AC 100 OPC Server must be set up to send DSPs.

Based on the configuration data provided in the BCD the AC 100 OPC Server can send DSPs for receiving DSPs configured in the controllers.

To achieve this the following configuration activities in FCB DB Element Editor and Bus Configuration Builder have to be performed:

1. Create and configure the receiving DSP and setting the DSP Station Number using FCB DB Element Editor, see Figure 82
2. Create and configure the DATs and further DB and PC elements to deal with the received data using FCB.
3. Create the AF 100 OPC station in BCB
4. Configure the DSP Station Number for the AF 100 OPC station in BCB

Note that an AF 100 OPC station can send DSPs under more than one station number, see Figure 83 and Figure 84.

Redundant OPC Servers cannot send the same DSP from both OPC Servers, as it is not allowed to have 2 sending DSPs with the same identification (STATION, IDENT). The DSP would only be sent by one of the OPC Servers.

**Sending DSPs** are by no means compatible with redundancy. If you work with redundant OPC Servers you **must not** have sending DSPs in one of them. If you have sending DSPs (on a non redundant OPC Server) these objects will not be visible on any other OPC Server.
Configure DSP DB Element

To be able to send DSPs from the AC 100 OPC Server Configure a DSP with SOURCE = RECEIVE and Set STATION to a station number that is only used for
DSPs sent by this OPC Server. (46 in this example). Configure the DAT and MMI objects as usual.

Refer to AMPL Configuration Advant Controller 100 Series Ref. Manual (3BSE009626*), Function Chart Builder User's Guide (3BDS100595*) and the applicable Data Base Elements Reference Manuals (3BDS005556*, 3BDS100594*, or 3BDS100593*) and PC Elements Reference Manuals (3BDS005557*, 3BSE000504*, or 3BSE009177*) for more details.

Configure AF 100 OPC Station

![Figure 83. Bus Configuration Builder - Station Properties](image)
Run BCB and add a unique station number of another station to the AF 100 station representing your AC 100 OPC Server as in the picture above.

![Station Properties]

**Figure 84. Station Properties**

Note that only entering Station numbers of other stations make sense here.

Refer to *Bus Configuration Builder User’s Guide (3BDS100312*) for more details.
Appendix C Extended Database Elements

Overview

This appendix will give you a general view of the Extended Database Element structure and functionality.

Extended DB elements are used to simplify the communication between the controller and client as well as the communication between different controllers.

The extended DB element is a concept that facilitates the engineering of AC 100 Series controllers. It addresses the following issues:

- Definition of signals independently of the I/O hardware.
- Definition of signals and texts for MMI purposes. Every aspect of the signal, from DSP communication up to event generation can be seen at a single DB element.
- Distribution of signals on the AF100. There is only one point in the system where the signal is defined. The definition includes also the conversion of raw values to engineering units.

The intention behind extended DB elements is the single point data entry in order to configure the controller as well as the MMI.

Extended DB elements carry information concerning the following items:

- Controller configuration
- MMI configuration
- Communication
- Signal description

The following extended DB elements are available for AC 110 / AC 160:

- Input elements
  - Analog input (AIS)
Overview

Appendix C  Extended Database Elements

– Digital input (DIS)

• Output elements
  – Analog output (AOS)
  – Digital output (DOS)

• Memory elements (MB, MI, MIL, MR)

To work with extended DB elements with all their advantages, you need the following three systems:

• Controller with base software or options supporting extended DB elements, such as AC 110 (Version 2.1 or higher) or AC 160 (Version 1.0 or higher).

• A generic OPC client together with the AC 100 OPC Server.

• Engineering environment supporting extended DB elements, such as CBA 1.0 or higher.

If you create, modify or delete extended DB elements, the FCB automatically creates, modifies or deletes other DB elements, as required by the extended DB element.

Configuration of extended DB elements is done in two ways, depending on whether the configuration parameter is changeable at run-time by the MMI:

• Some terminals have fixed values that can only be changed by FCB.

• Some terminal values can be changed by the MMI.
  For example, the MMI can initialize them at run-time by writing initial values.

We describe the usage of extended DB elements from the following points of view:

• Selection of signal sources, connection to signal sources.

• Internal processing in extended DB elements.

• Connection to signal sinks.

Figure 85 shows the main system components participating in the realization of extended DB element concept.
Extended DB Element Communication Between Stations

Extended DB elements are very versatile because:

- Extended DB elements allow transfer of I/O signals or signals generated by PC programs (for example, output by PC elements) to the MMI system and AC 100 Series controllers.
- They also allow transfer of commands and calculated data from the MMI system to AC 100 Series controllers.
- In case of AC 160, extended DB elements also support data exchange between redundant CPUs within the AC 160 stations via MDAT elements.

The following figure shows how the AIS extended DB element can be used to access and propagate an analog signal in different AC 100 Series stations:

*Figure 85. Extended DB element referencing HW element*
The AIS in Station 1 is configured to fetch the value from a hardware input channel and send it via DS/DAT communication. Here, the extended DB element is referenced also by an EVS(S) element that provides the event queue. DSP, DAT(R), DAT(IL), and EVS are created automatically at the creation of AIS. AIS6xx, for example, AIS630, has to be created explicitly.

The AIS in Station 2 is configured to receive the value from Station 1 via DS/DAT communication. DSP, DAT(R), and DAT(IL) are created automatically at the creation of AIS.

**Event Handling with Extended DB Elements**

To allow an extended DB element to generate events, set its EV_COM terminal to SEND. Fill also the EVS_IDNT and EVS_REF terminals appropriately. Don't forget to set the terminals LIM_TR and ERR_TR to appropriate values.

Then FCB creates the necessary EVS(S) element automatically.

The extended DB elements can generate events in the following cases:

- Operational changes.
- Error conditions.

For more information on DB elements in general see

*Data Base Elements Advant Controller 70 Reference Manual,*
*Data Base Elements Advant Controller 110 Reference Manual* and
Follow the mounting instructions below and the instructions in your PC manual regarding how to add cards on the PCI bus.

Cut main power supply and read the instructions supplied with your PC’s documentation before you start to disassemble.

Install the PCI board as follows:
1. Disconnect the PC’s main power supply.
2. Open up the PC’s chassis and locate an empty PCI bus slot.
3. Remove the cover bracket.
4. Insert the board firmly into place and secure the board bracket.
5. Close the PC’s chassis and Connect cables and terminator according to Figure 86.

6. Reconnect the main power supply and reboot your PC.
7. Wait until Windows has detected the CI527A card and install the driver.

Figure 86. CI527A Connections
8. The AC 100 OPC Server and the CI527A card and driver are successfully installed and you are now ready for the initial configuration.

9. Reboot the PC.

If the CI527A driver is not successfully installed open the Windows device manager and perform a manual installation of the CI527A driver.
Appendix E  Product Verification

After you have performed the initial configuration of the AC 100 OPC Server, you are recommended to verify that the software has been properly configured and that the system has contact with the Advant Fieldbus 100 network. To accomplish this, do the following:

It is not supported to run AC 100 OPC Server without the CI527A board properly installed.

1. Verify the About OPC Server, see Figure 87.

![Figure 87. Example of About AC 100 OPC Server](image)

2. Verify that the board is operational and all stations are reachable by viewing the Diagnostics.

Select Diagnostics from the ABB Start Menu, see Figure 88. For more information about the Diagnostics function, refer to How to View the Diagnostics on page 63.
3. Check the Windows Event Viewer to ensure that no fatal errors are reported from the ABB AC 100 OPC Server.

If the recommended verifications show problems, you can check the hardware of the CI527A by doing the following:

a. Check that the green P (power) is lit.
b. Check that the yellow RX (receive) LED is lit (if other stations on the Advant Fieldbus 100 network are up and running).

c. When the Advant Fieldbus 100 Interface is started, check that the yellow TX (transmit) LED starts to blink.
Numerics

800xA for AC 100  20

A

ABB Master Programming Language  12
AC  12, 53
AC 100 OPC Server  19, 27, 42, 62
AC 100 Series  12
ACC  12
Advant Controller  12
Advant Fieldbus 100  12, 19, 22
Advant OCS  12
AF100  12, 19, 22
AF100 Interface  32
AIS  133
Alarms and Events Interface  31
AMPL  12
AMPL Control Configuration  12
Analog Input  104
  object  104
Analog Input Object Type  133
Analog Output Object Type  140
AOS  140
APB  12
APC  12
Application Builder  12, 75
Application Performance Controller  12
Attributes
  AIS  133
  AOS  140
  CI  168
  CI_SUM  173
  DAT_AI  133
  DAT_AO  140
  DAT_DI  146
  DAT_DO  149
  DATx  153
  DIS  146
  DOS  149
  IO  175
  MB  154
  MBS  157
  MI  155
  MIL  155
  MR  155
  OPC  178
  PM  160
  PM_SUM  166

B

BCB  12, 78
BCD  12, 19
BCI  13
Board State  64
Bus Bandwidth  49
Bus Cable Disconnect  26
Bus Configuration Builder  12
Bus Configuration Database  12, 19, 27, 78, 95
Bus Configuration Interface  13
Bus Length  34
Bus Load  49
Bus Master  13, 64
Bus Number  64

C

Capacity  48
Index

CBA 13, 19, 27, 32
CI Board 13
CI Board Errors
  Invalid telegram 123
  Invalid telegram data 123
  Multiple source 123
  Permanent sender 123
CI Board Start-up 24
CI527A
  Connection drawing 191
Client Application 13, 47
Clock Master 13
COM 13
Commands
  AI 138
  AOS 144
  DAT_AI 138
  DAT_AO 144
  DAT_DO 153
  DATx 155
  DOS 153
  MB 155
  MI 155
  MIL 155
  MR 155
Communication Mechanism
  DataSet Peripheral 22
  Service Data Protocol 22
Compatibility 53
Component Object Model 13
Conceptual Overview 21
Configuration Wizard 33
  Application Version Supervision 38
  BCD path 37
  Bus Length 34
  Bus Master 34
  Bus Number 34
  Dynamic AIS Alarm Limits 38
  Keep server running 35
  OPC Server Filtering 39
  OPC Server Path 36
  Redundant Line Support 35
  Station Number 34
  Time Synchronization 35
Control Builder A 13, 19, 27, 32
Cyclic Subscriptions 24, 43

D

DAT 41
DAT Based Objects 13, 27 to 28
DAT_AI 133
DAT_AO 140
DAT_DI 146
DAT_DO 149
DAT(x) 153
Data Rate 48
Data Subscription 27
DataSet Peripheral (DSP) 14, 22, 24, 27, 48
DB Element 13
DCOM 14
Device Errors 65
Device Id 64
Diagnostics 63
  Board State 64
  Bus Master 64
  Bus Number 64
  Device Errors 65
  Device Id 64
  Line 1 Active 64
  Line 2 Active 64
  Reachable Stations 65
  Reconfiguration 64
  Redundant Line Support 64
  Signal Configuration 64
  Station Number 64
  Time Synchronization 65
Digital Input Object Type 146
Digital Output Object Type 149
DIS 146
Distributed COM 14
Index

DOS 149
DSP 14, 22
DSP Station Number 184

E
Electrostatic Discharge 14
Error messages
   Hardware Errors 123
   Software Errors 123
ESD 14
Event enabling cycle time 38
Event Handling 29
Event Sets 14, 22
Events
   AI 138
   AOS 144
   CI 171
   CL_SUM 174
   DAT_AI 138
   DAT_AO 144
   DAT_DI 149
   DAT_DO 153
   DATx 155
   DIS 149
   DOS 153
   IO 176
   MB 155
   MBS 158
   MI 155
   MIL 155
   MR 155
   OPC 180
   PM 163
EVS 14
Extended Automation System 800xA 20
Extended DB Elements 14 to 15, 27, 41, 74

F
FCB 14
For Extended DB Elements 28

Function Chart 14
Function Chart Builder 14, 81

H
Hardware Configuration and Diagnostics 31
Hardware Errors Messages 123
HMI 14
HSI 14
Human Machine Interface (HMI) 14
Human System Interface 14

L
LED 14
Light Emitting Diode 14
Line 1 Active 64
Line 2 Active 64

M
Man Machine Interface 14
MB 154
MBS 157
MI 155
MIL 155
MMI 14
MR 155
Multiple OPC Servers 32, 53
Mutual Lifesign Supervision 26, 31

O
Object (Process) 14
Object Handling 27
   Data Subscription 27
   Event Handling 29
   Service Requests 28
Object Linking and Embedding 14
OLE 14
Online Changes 32
OPC 14 to 15, 22
OPC Client Applications 22
OPC Custom Interface 62
Index

OPC Server 62
  Group 62
  Item 62
Operator Station 15
OS 15

PC 15
PC Element 15
PC Program 15
PCPGM 15
Performance 47
Personal Computer 15
Programmatic Identifier (ProgID) 62

Reachable Stations 65
Reconfiguration 64
Redundant Line Support 64
Redundant OPC Servers 32, 53, 60
Runtime 24

SDP 15, 22
Service Data Protocol (SDP) 15, 22
Service Requests 28
Shut-down 25
Signal Configuration 64
Simple Memory Object Type 153
Software Errors Messages 123
Station Number 64
Status Bits
  Analog input object 137
  Analog output object 143
  Digital input object 147
  Digital output object 151, 158
  PM 162
    PM_SUM 167
System Messages 32, 67
System Status 31, 65

system status objects 31

Time Synchronization 31, 65
Time Tagged Events 22
tutorial 104

XDB 15

X
Contact us

www.abb.com/800xA
www.abb.com/controlsystems

Copyright© 2003-2014 ABB.
All rights reserved.