

Exhibit B

ABB Inc.

EXHIBIT 1027

The diagrams and annotations included herein are for illustration purposes only and are not intended to necessarily represent precise software architectures.

Claim.Element	Analysis
<p>1. A system for generating a sequence of control commands for controlling a selected motion control device selected from a group of supported motion control devices, comprising:</p>	<p>“Control command(s)” has been construed to mean “command codes in hardware language, which instruct a motion control device to perform motion control operations.”</p> <p>On information and belief, systems based upon ABB’s Industrial System 800xA are systems for generating a sequence of control commands for controlling a selected motion control device (including, but not limited to, an AC500, ACS800, and IRC5) selected from a group of supported motion control devices, as depicted in the exemplary illustrations below.</p> <p>PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)</p> <p>Systems based upon ABB’s Industrial System 800xA include a System Connect, such as PLC Connect, which is used to communicate with controllers via OPC DA.</p> <p>PLC Connect is an option and provides integrated connectivity to System 800xA for ABB and third-party PLC-type controllers.</p> <p>PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2.</p> <p>On information and belief, other System Connect products operate similar to PLC Connect.</p>

1.1
 a set of motion control operations, where each motion control operation is either a primitive operation the implementation of which is required to operate motion control devices and cannot be simulated using other motion control operations or a non-primitive operation that does not meet the definition of a primitive operation;

The term “motion control operation(s)” has been construed to mean “abstract operations (such as GET POSITION, MOVE RELATIVE, or CONTOUR MOVE) used to perform motion control.”

The term “primitive operation(s)” has been construed to mean “motion control operation(s), such as GET POSITION and MOVE RELATIVE, necessary for motion control, which cannot be simulated using a combination of other motion control operations.”

ABB Systems, including systems based upon ABB’s Industrial System 800xA, comprise a set of motion control operations where each motion control operation is either a primitive operation the implementation of which is required to operate motion control devices and cannot be simulated using other motion control operations or a non-primitive operation that does not meet the definition of a primitive operation. These include system implementations that support PLC Open, such as those shown below.

Automation products
 Scalable PLC AC500
 Motion control PS551-MC

The PS551-MC is a new type of application program based on PLC open standard specifically intended for OEM machine builders and systems integrators looking for a reliable and easy-to-use high performance motion control drive module in their demanding applications for example in the field of material handling, packaging, plastics, printing and textile industry. It provides accurate positioning in one package without the need of an external motion controller.

Main features of Motion Control:

- Speed control
- Position control
- Position interpolator
- Positioning speed
- Acceleration
- Deceleration
- Standard sequential homing
- Selectable physical units for position values (mm, inch, increment, degree, revolution)
- Complete package of function blocks to work together with ABB Drives

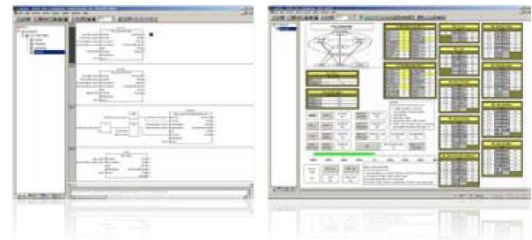


ABB Automation Products – AC500, AC31, CP400, WISA Catalog, 2009, ABB Automation Products, p. 8.

For example, ABB’s PLCopen compliance statement shows support for many motion control operations including ‘MOVE RELATIVE’ and ‘GET POSITION’.

Single Axis Function Blocks	Supported Yes / No	Comments (<= 48 char.)
MC_MoveAbsolute	Yes	
MC_MoveRelative	Yes	
MC_MoveAdditive	Yes	
MC_MoveSuperimposed	No	
MC_MoveVelocity	Yes	
MC_Home	Yes	
MC_Stop	Yes	
MC_Power	Yes	
MC_ReadStatus	Yes	
MC_ReadAxisError	Yes	
MC_Reset	Yes	
MC_ReadParameter	Yes	
MC_ReadBoolParameter	Yes	
MC_WriteParameter	Yes	
MC_WriteBoolParameter	Yes	
MC_ReadActualPosition	Yes	
MC_PositionProfile	No	
MC_VelocityProfile	No	
MC_AccelerationProfile	No	

ABB [PLCopen] Compliance – Function blocks for motion control – Version 1.1. (annotated)

ABB Documentation shows support for many non-primitive motion control operations as implemented, for example, in their support of PLCopen and also for example with what are called 'calculated softpoints' – which are softpoints created by combining two or more 'other' softpoints.

Easy object configuration

A configuration aspect for configuring the PLC connectivity with configuration data for mapping and structuring of process signals onto PLC objects in the Aspect Directory. Object orientation is supported with PLC object types and PLC object instances. Signal properties are inherited from the object type or set individually on an object instance. Complex structures can be created due to the composite object types (a type can consist of one to several sub-types).

PLC Connect Web Site, ABB Automation Technologies, *Product Guide > Control Systems > 800xA > System > 800xA PLC Connect > Tools.*

Real time data base

PLC Connect handles all dynamic process data collected by a communication server from connected controllers/PLCS/RTUs and keeps them in a Real Time Data Base (RTDB). Also internally calculated SoftPoint values are stored in RTDB.

PLC Connect Web Site, ABB Automation Technologies, *Product Guide > Control Systems > 800xA > System > 800xA PLC Connect > Tools.* (annotated)

ABB supports motion control operations in their PLC's (example: AC500) and Drives (example: ACS350).

ABB machinery drives provide high performance speed, torque and motion control for demanding machines. They can control induction, synchronous and asynchronous servo and high torque motors with various feedback devices. The compact hardware and programming flexibility ensure the optimum solution. The innovative memory unit concept enables flexible drive configuration.

High performance machinery drive, ABB, *Product Guide > Drives > Low Voltage AC Drives > Machinery Drives > High Performance Machinery Drives.*

DriveOPC features

- DriveOPC supports OPC's data access 1.0A.
- Read access to:
 - Drive status: local, running, direction, fault, warning, reference
 - Signals and parameters
 - Fault logger contents
 - Event logger contents
 - General drive information
 - Data logger settings, status and contents

Write access to:

- Drive control: local, start, stop, forward, reverse, coast stop, reset fault, home, teach-in, contactor on/off, reference
- Parameters
- Fault logger clear
- Data logger init, start, trig, clear
- RUSB-02 and Windows Vista support

ABB industrial drives – ACS800, multidrives 1.1 to 5600 kW Catalog, 2010, ABB Automation Products, p. 43.

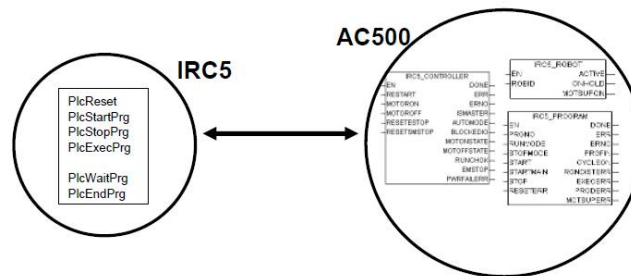
ABB also supports motion control operations in their PLC's and Robots (IRC5).

Develop OPC solutions

Monitor IRC5 from an OPC client on a PC. The ABB IRC5 OPC Server provides means to read and write IO signals and RAPID data and to be notified of events in the robot controller.

ABB Product Guide > Robotics > RobotStudio Community > Developer Tools web site, 2010, ABB.

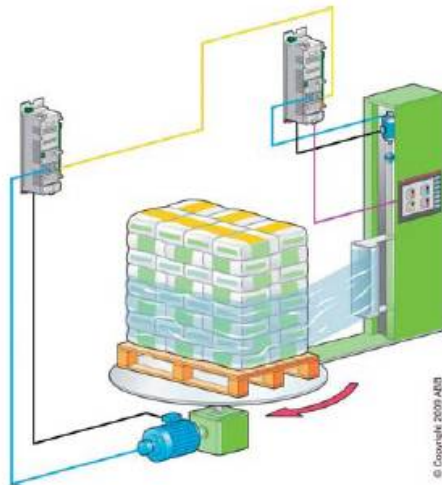
Both installed from robot factory



- IRC5
 - RAPID instructions for PLC interaction
 - Background task for handshake and error handling
 - System modules for application programming
- PLC
 - Basic PLC functionality implemented on OS level
 - Common function blocks for IRC5 control

ABB Integrated PLC AC500 in IRC5, Slide 14.

Several example ABB PLC/Drive implementations that use motion control operations are shown below.



Application 1: Wrapper

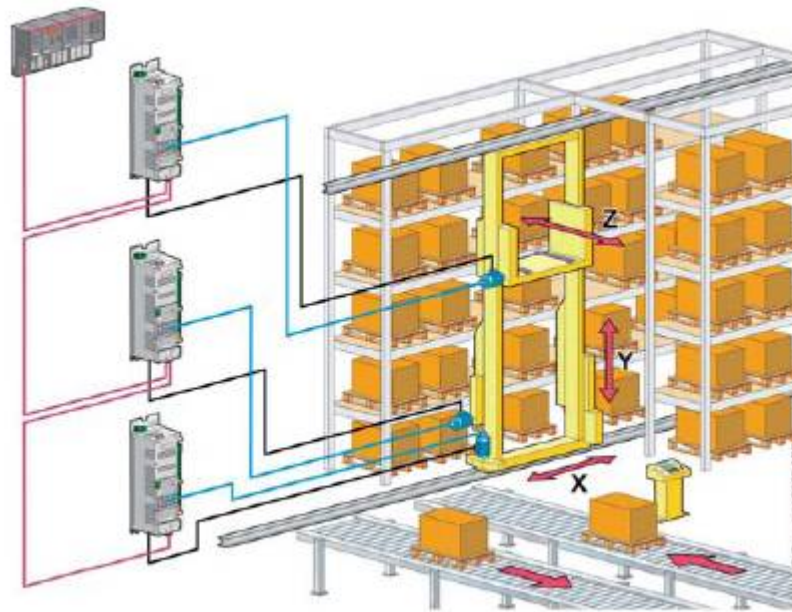
Integration

Being able to integrate the PLC, drives and electric motors provides motion control applications with a seamless, well-engineered solution.

The integrated package provides for centralized and decentralized motion control covering speed, torque, positioning and synchronization.

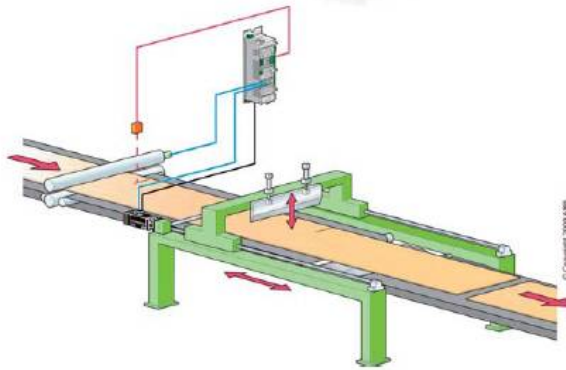
Drives and PLCs can be rapidly integrated using the Motion Control Software Library based on PLCopen function blocks. Ready-made cabling provides an easy and proven way to connect power, mechanical brake control and feedback signals from the motor to the drive in a secure and tested way.

Flexible factory automation from a global supplier ABB offers a broad motion control system with PLCs, drives, motors and accessories, ABB, 2009, p. 2.

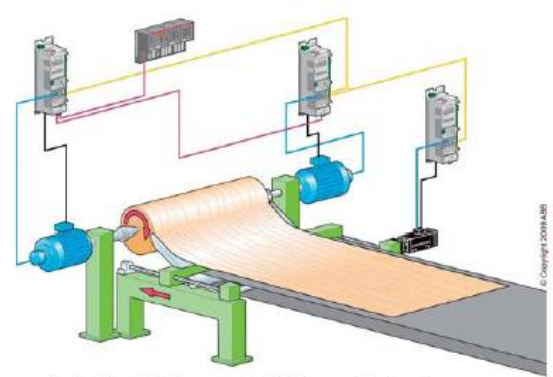


Application 2: Automatic warehouse

Flexible factory automation from a global supplier ABB offers a broad motion control system with PLCs, drives, motors and accessories, ABB, 2009, p. 2.



Application 3: Flying shear

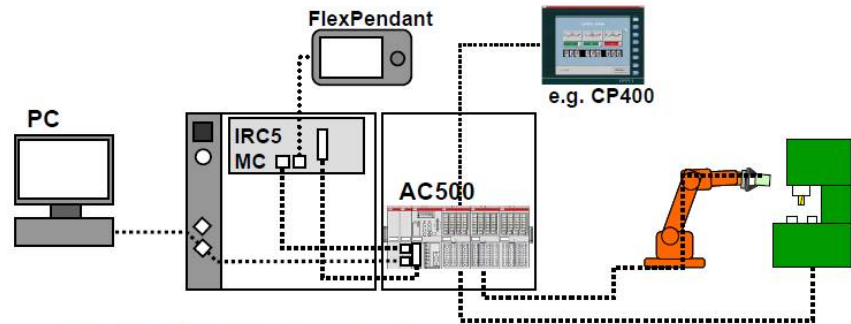


Application 4: Lathe, constant thickness adjustment

Flexible factory automation from a global supplier ABB offers a broad motion control system with PLCs, drives, motors and accessories, ABB, 2009, p. 3.

Several example ABB PLC/Robot implementations that use motion control operations are shown below.

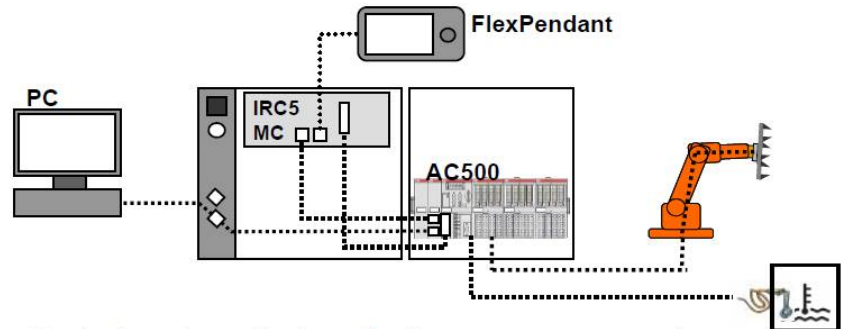
Application Example 2: Cell Control



- Control of the complete cell
 - Peripheral equipment, like clamping, machines, in/out feeders, turntables etc
 - Robot program selection and starting (PLC takes master role)
 - Operator communication
 - Master bus coupler an alternative to discrete signals

ABB Integrated PLC AC500 in IRC5, Slide 24.

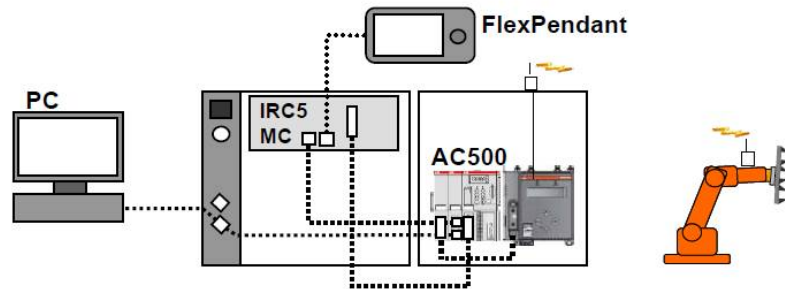
Application Example 3: Peripheral Equipment Control



- Control and monitoring of grippers, process equipment etc
 - RAPID starts PLC programs (PLC takes slave role)
 - Might include closed loop PID control
 - Master bus coupler an alternative to discrete signals
 - To be acquired from ABB Low Voltage Drives

ABB Integrated PLC AC500 in IRC5, Slide 25.

Application Example 5: Wireless Peripheral Equipment Control



- Wireless IP67 I/O units
- Very robust communication, even with arc welding
- Wireless proximity sensors
- Optional wireless power supply
- WISA equipment + master bus coupler to be acquired from ABB Low Voltage Drives

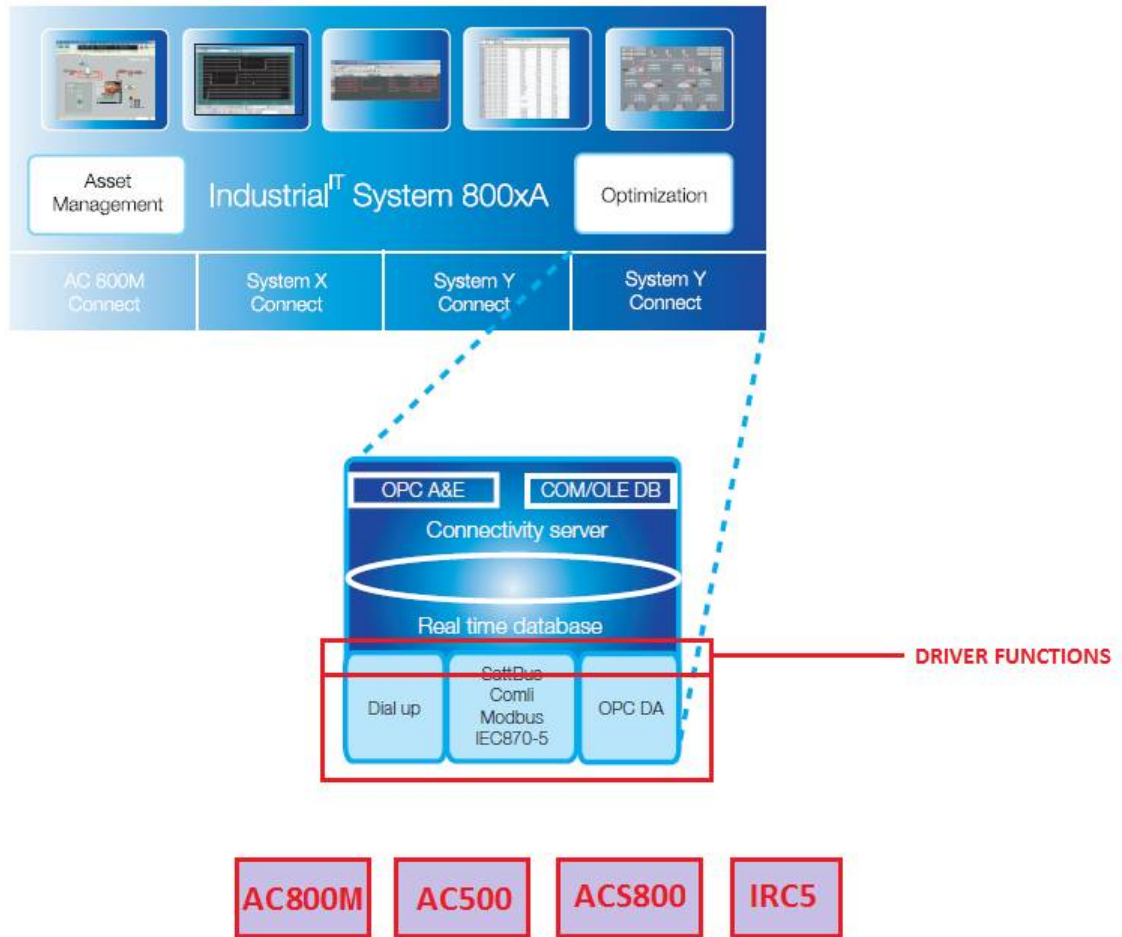
ABB Integrated PLC AC500 in IRC5, Slide 27.

1.2 a core set of core driver functions, where each core driver function is associated with one of the primitive operations;

The term "driver function(s)" has been construed to mean "abstract functions that are associated with primitive or non-primitive motion control operations and may define parameters necessary to implement such operations."

The term "core driver functions" has been construed to mean "driver functions associated with primitive motion control operations."

On information and belief, systems based upon ABB's Industrial System 800xA comprise a core set of core driver functions, where each core driver function is associated with one of the primitive operations. These driver functions include, but are not necessarily limited to, certain OPC functions including IOPCSyncIO read and write functions. First, on information and belief, the ABB Connectivity Server in Industrial System 800xA communicates with drivers using for example OPC DA. See also the OPC Alarms and Event standard.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

Second, the OPC DA specification defines functions that are associated with primitive motion control operations.

3.1.2 OPCGroup Object

IOPCGroupStateMgt

HRESULT GetState(pUpdateRate, pActive, ppName, pTimeBias, pPercentDeadband, pLCID, phClientGroup, phServerGroup)
 HRESULT SetState(pRequestedUpdateRate, pRevisedUpdateRate, pActive, pTimeBias, pPercentDeadband, pLCID, phClientGroup)
 HRESULT SetName(szName);
 HRESULT CloneGroup(szName, riid, ppUnk);

IOPCPublicGroupStateMgt (optional)

HRESULT GetState(pPublic);
 HRESULT MoveToPublic(void);

IOPCSyncIO

HRESULT Read(dwSource, dwNumItems, phServer, ppItemValues, ppErrors)
 HRESULT Write(dwNumItems, phServer, pItemValues, ppErrors)

IOPCAsyncIO

HRESULT Read(dwConnection, dwSource, dwNumItems, phServer, pTransactionID, ppErrors,)
 HRESULT Write(dwConnection, dwNumItems, phServer, pItemValues, pTransactionID, ppErrors);
 HRESULT Cancel (dwTransactionID);
 HRESULT Refresh(dwConnection, dwSource, pTransactionID);

IOPCItemMgt

HRESULT AddItems(dwNumItems, pItemArray, ppAddResults, ppErrors)
 HRESULT ValidateItems(dwNumItems, pItemArray, bBlobUpdate, ppValidationResults, ppErrors)
 HRESULT RemoveItems(dwNumItems, phServer, ppErrors)
 HRESULT SetActiveState(dwNumItems, phServer, bActive, ppErrors)
 HRESULT SetClientHandles(dwNumItems, phServer, phClient, ppErrors)
 HRESULT SetDatatypes(dwNumItems, phServer, pRequestedDatatypes, ppErrors)
 HRESULT CreateEnumerator(riid, ppUnk)

IDataObject

HRESULT DAdvise(pFmt, adv, pSnk, pConnection);
 HRESULT DUnadvise(Connection);
 Note: all other functions can be stubs which return E_NOTIMPL.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 18. (annotated)

See example OPC tags shown below.

You can change the width of a column by dragging the column separator in the title.

Name	Value	OPC Address
01.01: MOTOR SPEED [rpm]	0	Par.1.1
01.05: TORQUE [%]	0	{0}{1}Par.1.5
20.01: MINIMUM SPEED [rpm]	-30	{0}{1}Par.20.1
20.07: MINIMUM FREQ [Hz]	<Read-protected>	{0}{1}Par.20.7
Running	<Bad>	{1}{1}Status.Running

The image displayed in front of a descriptive name shows the status of the item.

Image Status

- Off-line and not locked
- On-line and not locked, background of the value is also yellow
- Off-line and locked
- On-line and locked, background of the value is also yellow
- Monitored in channel 1
- Monitored in channel 2
- Monitored in channel 3
- Monitored in channel 4
- Monitored in channel 5
- Monitored in channel 6

Note that monitored items are always locked, but the locking done by the user is separated from this lock. It means that when an item is removed from the monitor, it shows the locking status set by the user (either before or during monitoring).

Monitored items can be put on-line, too. Background of the value is yellow, as all other types of on-line items.

Name	Value	OPC Address
01.03: FREQUENCY [Hz]	0	{0}{1}Par.1.3
01.04: CURRENT [A]	0	{0}{1}Par.1.4
01.07: DC BUS VOLTAGE V [V]	0	{0}{1}Par.1.7
01.10: ACS600 TEMP [C]	50	{0}{1}Par.1.10

To select an item, just click its descriptive name. Several items can be selected. You can do multiple selection with the mouse as follows:

- To select a range of items, first click the descriptive name of the item at the one end and then, with the Shift key down, click the descriptive name at the other end.
- To change selection status of a single item at a time, keep the Ctrl key down when clicking the descriptive name of the item.

DriveWare User's Manual – Drive Window 2, 1/7/2004, ABB, p. 2-41. (annotated)

1.3
 an extended
 set of
 extended
 driver
 functions,
 where each
 extended
 driver
 function is
 associated
 with one of
 the non-
 primitive
 operations;

The term "driver function(s)" has been construed to mean "abstract functions that are associated with primitive or non-primitive motion control operations and may define parameters necessary to implement such operations."

The term "primitive operation(s)" has been construed to mean "motion control operation(s), such as GET POSITION and MOVE RELATIVE, necessary for motion control, which cannot be simulated using a combination of other motion control operations."

On information and belief, systems based upon ABB's Industrial System 800xA comprise an extended set of extended driver functions, where each extended driver function is associated with one of the non-primitive operations. These driver functions include, but are not limited to, certain OPC functions including IOPCAsyncIO read and write functions. Again, on information and belief, the ABB Connectivity Server in Industrial System 800xA communicates with drivers using for example OPC DA. And the OPC Specification supports many extended driver functions that are not essential for motion control, yet can easily be simulated.

4.5.1.8 Reading and Writing Data

There are basically three ways to get data into a client (ignoring the 'old' IDataObject/IAdviseSink).

- IOPCSyncIO::Read (from cache or device)
- IOPCAsyncIO2::Read (from device)
- IOPCallback::OnDataChange() (exception based) which can also be triggered by IOPCAsyncIO2::Refresh.

In general the three methods operate independently without 'side effects' on each other.

There are two ways to write data out:

- IOPCSyncIO::Write
- IOPCAsyncIO2::AsyncWrite

OLE for Process Control Data Access Custom Interface Standard, Version 2.05A, page 74. (annotated)

4.5.7 IConnectionPointContainer (on OPCGroup)

This interface provides functionality similar to the IDataObject but is easier to implement and to understand and also provides some functionality which was missing from the IDataObject Interface. The client must use the new IOPCAsyncIO2 interface to communicate via connections established with this interface. IOPCAsyncIO2 is described elsewhere. The 'old' IOPCAsnyc will continue to communicate via IDataObject connections as in the past.

OLE for Process Control Data Access Custom Interface Standard, Version 2.05A, page 111. (annotated)

7. Caching

When a value is displayed on the screen, it is usually not fetched directly from a drive. We will explain here the principles of moving data around in DriveWindow and DriveOPC.

We use here the following OPC terminology:

- ConnectionPts* Device is same as drive (control board)
- &
- IDataObject
- &
- AsyncIO
 - A group can be active or inactive. For each active group, there is a cyclically running thread within DriveOPC. The thread reads the active item values from a device and calls a so called advise sink within DriveWindow. This call-back mechanism is used by all on-line activities within DriveWindow.
- Emulation
 - Always, when a value is read from or written to a device, it is also cached within DriveOPC.
- SyncIO
 - DriveWindow also reads and writes values directly without the call-back. Each read/write operation can contain several items.
 - The initial quality of values in the DriveOPC cache is bad. Some items (such as datalogger channel buffers) are not automatically read from a device by DriveWindow, and their quality stays bad, until the user requests updating of them.

ABB DriveWare DriveWindow 2 User's Manual, page 10-20. (annotated)

"The Value and Quality are the values that the [OPC] server obtains from the device at a periodic rate sufficient to accommodate the specified UpdateRate. If the Quality has changed from the Quality last sent to the client [e.g. the Motion Component], then the new value and quality will be sent to the client through the IOPCDataCallback::OnChange method, and the cache of the server should be updated with the acquired value and quality. If the Quality has NOT changed from the Quality last sent to the client, the server should compare the acquired value for a change that exceeds the Deadband criteria. If the change in value exceeds the deadband criteria, then the new value and new quality will be sent to the client through the IOPCDataCallback::OnChange method, and the cache of the server should be updated with the acquired value and quality." OLE for Process Control, Data Access Custom Interface Standard, Version 2.05A, Page 26 ([web link](#))

For example, pursuant to an asynchronous read function (IOPCAsyncIO read), when the quality has not changed from the previous quality, and the data value (e.g. the motor position) has not changed outside the deadband specified, no data is sent to the client. Unlike a primitive motion operation that would for example read the position value from a single axis, the extended functionality offered provides a 'more advanced' and more efficient manner for which to read the position value of a single axis whereby only changed position values are sent to the client thus freeing the client from receiving redundant data that has not changed.

As another example, ABB offers calculated softpoints that are 'calculated' using one or more other signals (either emulated or as collected from the controller).

PLC Connect provides the following features:

- Basic object types for PLC type signals and softpoint signals.
- Configuration tools for creating and editing PLC type objects.
- A full set of faceplates for the PLC type objects.
- Integrated Real Time Database (RTDB) to keep an updated image of connected process points as well as calculated softpoints.
- Communication drivers.
- Dial Manager for remote communication.

Industrial IT Compact HMI 800 System Version 4.1 – Product Guide, 2005, ABB Automation Technologies, p. 33. (annotated)

Softpoints

Softpoints are user-defined data points that do not directly connect to physical I/O. Softpoints include various data types such as Boolean, integer, single and double precision, floating point and string.

Softpoint data is fully integrated and treated in the same manner as any other value in the system. Engineering unit definitions, descriptor text, and alarm limits are part of softpoint configuration. Softpoint alarms are fully integrated with system wide alarms and events. Softpoints are accessible everywhere in the system including displays, historical recording and reports. Desktop Trends and Excel Data Access can read from and write to Softpoints.

The Softpoint Services software may run on any number of servers within a system. Each Softpoint Server can have up to 2500 softpoint objects. Each softpoint object can have up to 100 signals; however, the total number of signals cannot exceed 25,000.

Calculations

With Calculations, a calculation script is created with user-defined inputs and outputs.

Inputs can be any aspect object property (i.e. mode, measured value, etc.) of an actual process tag or a Softpoint, and outputs can be any updateable point in the system.

Calculations can be triggered by input changes, scheduled to execute cyclically or scheduled at a given date and time. Data quality information is maintained within all calculations. Therefore, if the input to a calculation has bad data quality, the resulting calculation will be marked as bad quality.

Uses for Softpoint and Calculation functions include:

- Calculations required for regulatory reporting.
- Preventative maintenance monitoring.
- Model predictive control.

Industrial IT Extended Automation System 800xA – System Version 3.1, System Guide, 2004, ABB Automation Technologies, p. 139.

The following is an example of a calculated softpoint.

- Process analysis.
- Integration of external data into the system.

The following is an example of a calculation that calculates the average value for up to four input variables, [Figure 47](#).

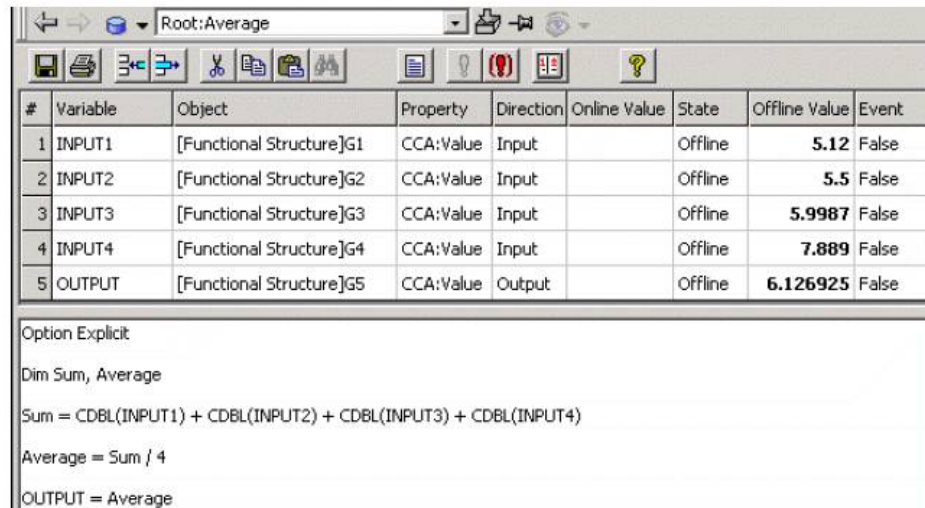


Figure 47. Example, Calculation for Average

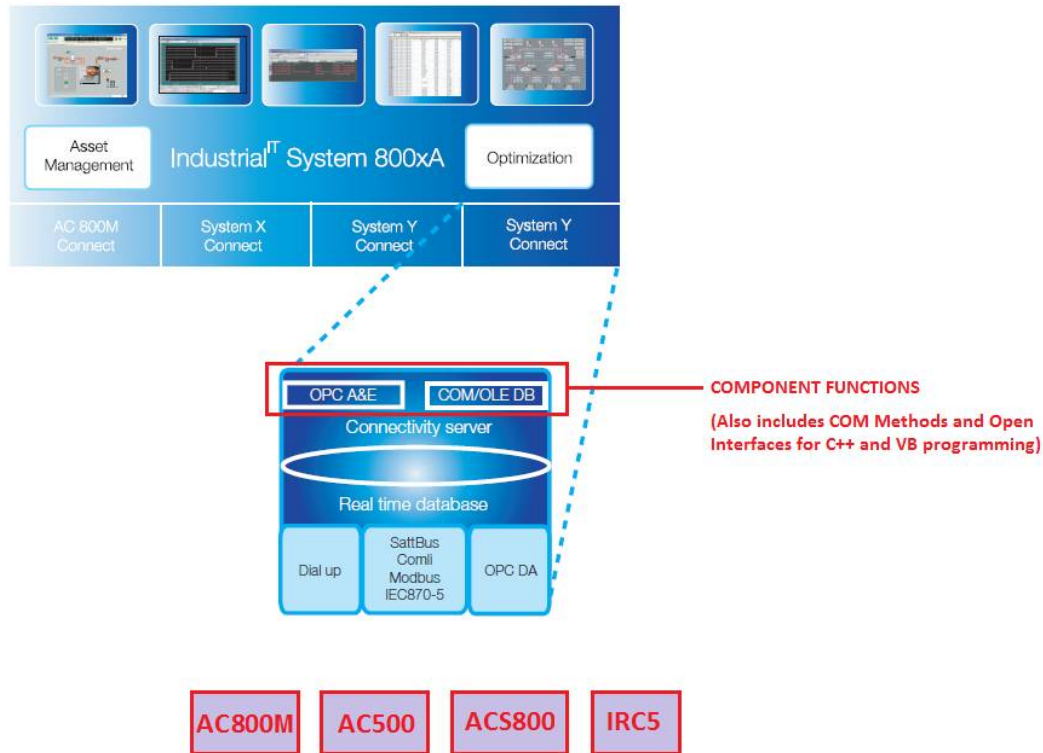
The Calculations function has a built-in scheduling tool for scheduling calculations. This tool supports cyclical scheduling or scheduling on a given date and time. As an alternative you can use the 800xA Scheduler for a wider range of scheduling options.

Industrial IT Extended Automation System 800xA – System Version 3.1, System Guide, 2004, ABB Automation Technologies, p. 140.

1.4
a set of
component
functions;

The term "component function" has been construed to mean "a hardware independent instruction that corresponds to an operation performed on or by a motion control device."

On information and belief, systems based upon ABB's Industrial System 800xA comprise a set of component functions, including but not limited to the component functions available through, for example, the COM Interfaces, PLC Connect interfaces and other application programming interfaces that communicate with the Connectivity Server.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

Value Pre Treatment and Open Interface

COM interfaces are provided to give possibility to access (read and write) process values and SoftPoint values from, for example, a VB program. A program can run in any node and access multiple PLC Connect Connectivity Servers. Application specific pre-treatment calculations can be added for process values and alarms and events.

PLC Connect Web Site, ABB Automation Technologies, *Product Guide > Control Systems > 800xA > System > 800xA PLC Connect > Tools.* (annotated)

Uniform integration of different PLCs

PLC Connect integrates individual signals in any connected controller, PLC or RTU with the 800xA system.

The operator receives process data in the same graphics regardless of the type of controller or the communication protocol used.

Real time database (RTDB)

All dynamic process data from connected controllers, PLCs and RTUs is stored in a real time database. Current values and status are always available and constantly updated, so there is no need to wait for the OPC server to set up subscriptions – values are available directly. A browser interface in third-party OPC servers is not required.

Open Interfaces

A number of open interfaces are available in PLC Connect for access by external applications. Real time access for reading and writing process values is available through COM Methods. Application-specific pre-treatment calculations can be added for received process values as well as detected alarms and events.

PLC Connect includes a COM interface for integrating an application that is to be executed on an event, an OLE DB provider for accessing logged events and alarms, and a COM interface for initiating and disconnecting calls handled by the dial manager for dialed communication with PLCs.

PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

PLC Connect adds traditional PLC type functionality as an integrated part of the Industrial IT concept. This means that traditional system capabilities, typically requiring a large number of process I/O:s to be connected through a range of controllers from different manufacturers, can be realized with an Industrial IT Compact HMI 800.

PLC Connect provides the following features:

- Basic object types for PLC type signals and softpoint signals.
- Configuration tools for creating and editing PLC type objects.
- A full set of faceplates for the PLC type objects.
- Integrated Real Time Database (RTDB) to keep an updated image of connected process points as well as calculated softpoints.
- Communication drivers.
- Dial Manager for remote communication.
- Alarms detection and OPC Alarms and Events generation for PLC binary signals.
- Alarm limit detection and OPC Alarms and Events generation for PLC integer and real signals.
- Open interface to PLC signals and softpoints from application programs in VB and C++.

PLC Connect is typically used in the following cases:

- For integration of AC800M/C Industrial IT Baseline 2 controllers when full DCS controller integration is not required.
- When remote connection of PLCs and RTUs are required.

ABB, Industrial IT Compact HMI 800 System Version 4.1, Product Guide, PLC Connect, p. 33-34, (annotated)

On information and belief, Variable Access functions are provided through the Variable Access Interface.

Variable Access

The variable access is performed by one or more executable files independently from the RTDB.

In the RTDB two classes are implemented: **Variables** and **Variable**.

Client means a certain instance of a Variables object, see [Variable Access Interface](#) on page 75. A program (.EXE file) can have several instances of Variables, although in most cases there is only one instance.

ABB IndustrialIT 800xA – System PLC Connect System Version 5.0 – Configuration, page 73. (annotated)

On information and belief, Variable Access functions operate on PLC Connect signals.

Item

Syntax: Item(sVarName As String) As Variable

Gets a reference to a named variable. If the variable is not found, "Nothing" is returned.

sVarName(in): name of the variable.

ReadValue

Syntax: ReadValue(sVarName As String, vntValue As Variant, OPCQuality As Integer) As Boolean

Reads the value of a named variable. FALSE if the variable is not found.

sVarName(in): name of the variable.

vntValue(out): returns the value of the signal.

OPCQuality(out): indicates the OPC quality of the variable.



There are also variants of the ReadValue method, see [More on Reading and Writing](#) on page 88.

WriteValue



If you write data to a signal connected to an external IO then that data will be written to the controller.

Syntax: WriteValue(sVarName As String, vntValue As Variant, OPCQuality As Integer) As Long

Writes to a named variable, but not if vntValue is "vtEmpty". This is useful if you only want to write into the error bits. A variable is "vtEmpty" if it is declared but not assigned. If the variable is fetched from a controller, the vntValue is written to the controller if the variable is controllable. Otherwise, the functions connected to the signal are performed, for example the alarm function.

Returns status codes according to [Table 4](#) on page 86.

sVarName(in): name of the variable.

vntValue (in): value to the signal.

ABB IndustrialIT 800xA – System PLC Connect System Version 5.0 – Configuration, page 80.

The Variable Access Interface provides read, write, and event support (among others) for various hardware independent PLC Connect signals. The PLC Connect signals are defined within PLC Connect and map to actual OPC items.

Signal Types Used for OPC Server Data Uploads

The PLC_XXX signal types, for example PLC_Binary, are pre-defined in PLC Connect. Do not change any configuration with respect to these signal types. They are used for mapping OPC server items to PLC Connect signals during an OPC server data upload. The PLC Uploader aspect is used to retrieve and filter the configuration from a connected OPC server, and automatically create PLC Connect process objects connected to the OPC server items.

ABB IndustrialIT 800xA – System PLC Connect System Version 5.0 – Configuration, page 20. (annotated)

Connect Process Signal

An extended signal represents a signal in the real world, or an internal state. The former is connected externally to a hardware address in a controller, while the latter is not.

Configuration Example (Continued)


- 25. Select a process signal. In this example, the ‘Tank1 TankTemp’ signal part of the ‘Tank1’ object is selected.
- 26. The  icon indicates that the process signal is not connected externally, see also Figure 25.



Figure 25. Icon Indicating a Not Connected Process Signal

- 27. Select the Signal Configuration aspect, see also Figure 26. Under the ID tab, you can connect the signal to a hardware address in the controller.

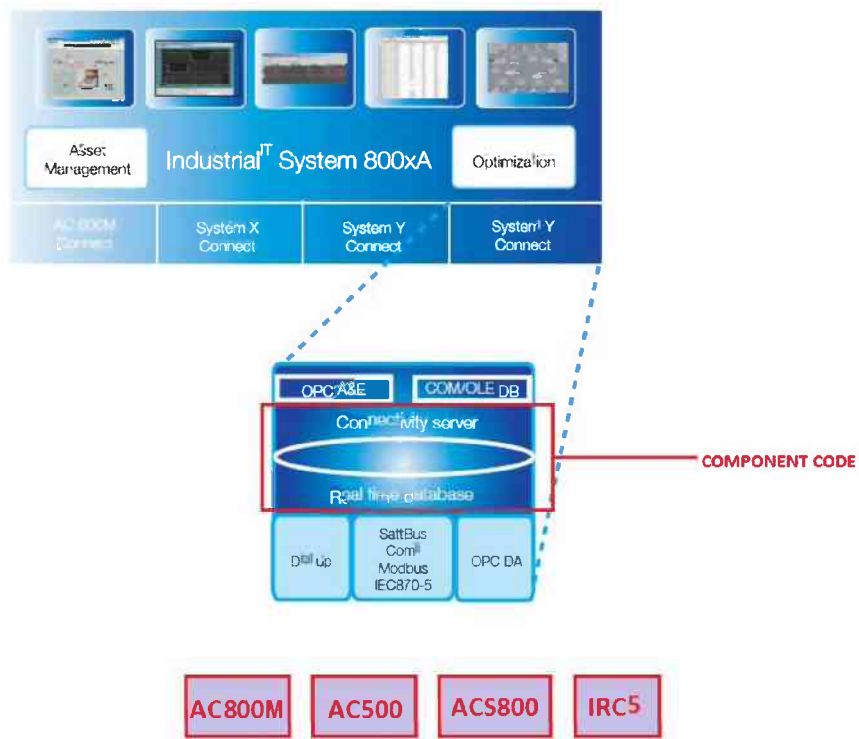
ABB IndustrialIT 800xA – System PLC Connect System Version 5.0 – Configuration, page 30. (annotated)

1.5 component code associated with each of the component functions, where the component code associates at least some of the component functions with at least some of the driver functions;

The term "component code" has been construed to mean "software code in the motion control component that associates at least some of the component functions with at least some of the driver functions."

The term "motion control component" has been construed to mean "a binary software module that associates component functions with driver functions by calling the driver functions."

On information and belief, systems based upon ABB's Industrial System 800xA comprise component code associated with each of the component functions, where the component code associates at least some of the component functions with at least some of the driver functions. Specifically, the component code is in the connectivity server (and/or real time database software) and associates component functions (as described above) called by application programs through one or more application programming interfaces with driver functions (as described above) called by the real time database and/or connectivity server (which comprises the motion control component) on the software driver(s), which may be part of the connectivity server. On information and belief, examples of the connectivity server associating the component functions with the driver functions include the real-time database, Aspect Objects, and other data structures.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

Uniform integration of different PLCs

PLC Connect integrates individual signals in any connected controller, PLC or RTU with the 800xA system.

The operator receives process data in the same graphics regardless of the type of controller or the communication protocol used.

Real time database (RTDB)

All dynamic process data from connected controllers, PLCs and RTUs is stored in a real time database. Current values and status are always available and constantly updated, so there is no need to wait for the OPC server to set up subscriptions – values are available directly. A browser interface in third-party OPC servers is not required.

Open Interfaces

A number of open interfaces are available in PLC Connect for access by external applications. Real time access for reading and writing process values is available through COM Methods. Application-specific pre-treatment calculations can be added for received process values as well as detected alarms and events.

PLC Connect includes an COM interface for integrating an application that is to be executed on an event, an OLE DB provider for accessing logged events and alarms, and a COM interface for initiation and disconnecting calls handled by the dial manager for dialed communication with PLCs.

PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

PLC Connect provides the following features:

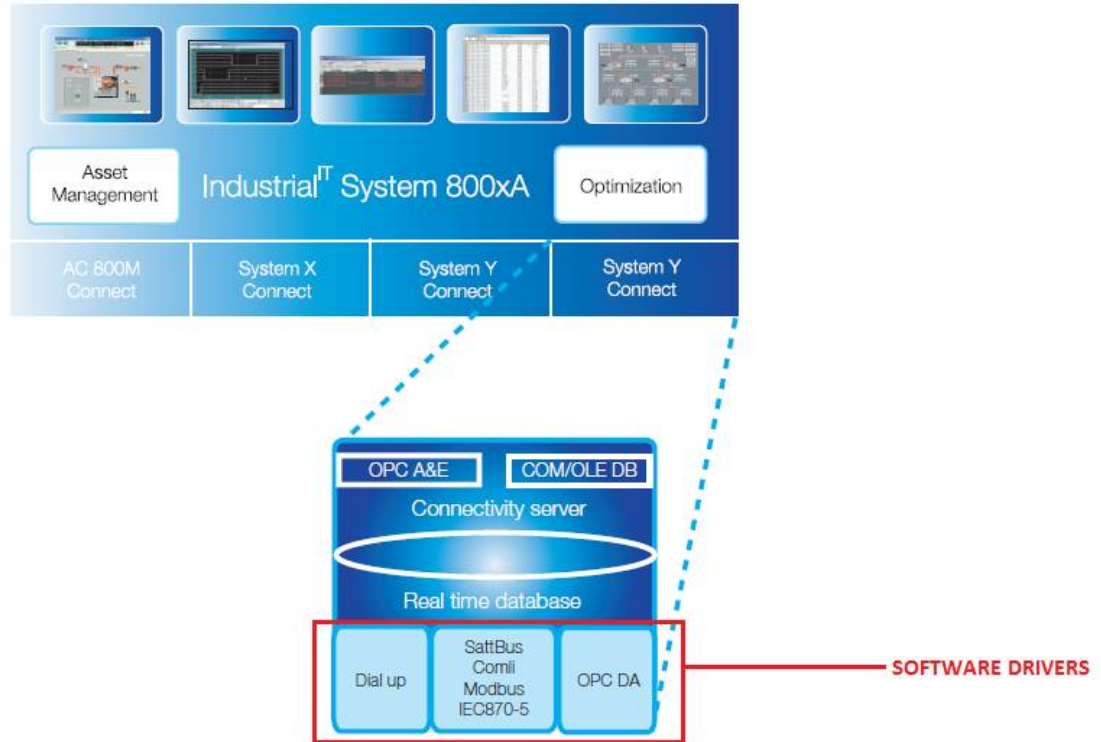
- Basic object types for PLC type signals and softpoint signals.
- Configuration tools for creating and editing PLC type objects.
- A full set of faceplates for the PLC type objects.
- Integrated Real Time Database (RTDB) to keep an updated image of connected process points as well as calculated softpoints.
- Communication drivers.
- Dial Manager for remote communication.

Industrial IT Compact HMI 800 System Version 4.1 – Product Guide, 2005, ABB Automation Technologies, p. 33. (annotated)

1.6
a set of
software
drivers,
where

The term “software driver(s)”/“drivers” has been construed to mean “a controller dependent software module that supports some core driver functions and is used to control a hardware device or group of related hardware devices.”

On information and belief, systems based upon ABB’s Industrial System 800xA comprise a set of software drivers. For example, on information and belief PLC Connect uses drivers that comprise driver code, implement driver functions and are associated with one or of the hardware devices.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2.

PLC Connect provides the following features:

- Basic object types for PLC type signals and softpoint signals.
- Configuration tools for creating and editing PLC type objects.
- A full set of faceplates for the PLC type objects.
- Integrated Real Time Database (RTDB) to keep an updated image of connected process points as well as calculated softpoints.
- Communication drivers.
- Dial Manager for remote communication.

Industrial IT Compact HMI 800 System Version 4.1 – Product Guide, 2005, ABB Automation Technologies, p. 33.

OPC support

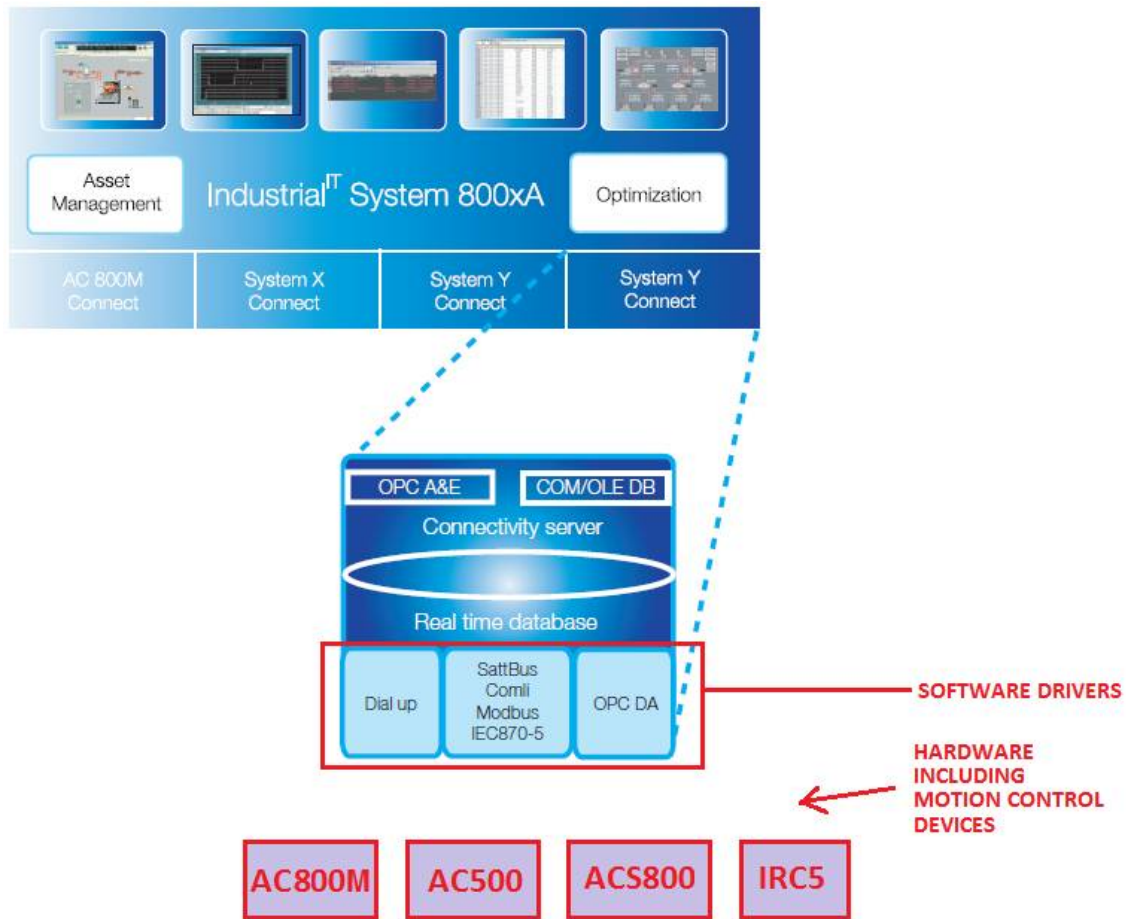
The Communication Server has, besides protocol drivers for traditional communication protocols, also a built in OPC DA client driver, which makes it possible to communicate with any OPC DA server following the guidelines of OPC Foundation. Timestamp from the connected device is supported.

PLC Connect Web Site, ABB Automation Technologies, *Product Guide > Control Systems > 800xA > System > 800xA PLC Connect > Tools.* (annotated)

1.6.i
each
software
driver is
associated
with one
motion
control
device in
the group of
supported
motion
control
devices,

The term "software driver(s)"/"drivers" has been construed to mean "a controller dependent software module that supports some core driver functions and is used to control a hardware device or group of related hardware devices."

On information and belief, systems based upon ABB's Industrial System 800xA comprise software drivers that are associated with one motion control device in the group of supported motion control devices. These drivers include all OPC-compliant drivers for motion control devices supported in Industrial System 800xA systems.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

On information and belief, all ABB Drives, PLCs, and Robot Controllers support OPC.

The information below shows OPC support for at least the following ABB Motion Control Devices:

- **ABB AC500 PLC (presumably with ACS500)**
- **ABB ACS800 Multi Drive**
- **ABB IRC5 Robot Controllers**

ABB AC500 has OPC support and Motion Control



ABB Automation Products – AC500, AC31, CP400, WISA Catalog, 2009, ABB Automation Products, p. 1.

ABB AC500 OPC Support

Open interfaces

DDE and OPC alarm and events.

ABB Automation Products – AC500, AC31, CP400, WISA Catalog, 2009, ABB Automation Products, p. 4. (annotated)

ABB AC500 Motion Control Support

Automation products

Scalable PLC AC500

Motion control PS551-MC

The PS551-MC is a new type of application program based on PLC open standard specifically intended for OEM machine builders and systems integrators looking for a reliable and easy-to-use high performance motion control drive module in their demanding applications for example in the field of material handling, packaging, plastics, printing and textile industry. It provides accurate positioning in one package without the need of an external motion controller.

Main features of Motion Control:

- Speed control
- Position control
- Position interpolator
- Positioning speed
- Acceleration
- Deceleration
- Standard sequential homing
- Selectable physical units for position values (mm, inch, increment, degree, revolution)
- Complete package of function blocks to work together with ABB Drives

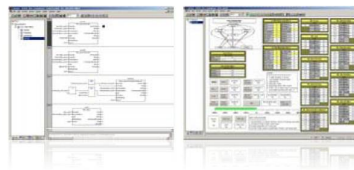


ABB Automation Products – AC500, AC31, CP400, WISA Catalog, 2009, ABB Automation Products, p. 8.

ABB ACS800 has OPC support and Motion Control



ABB industrial drives – ACS800, multidrives 1.1 to 5600 kW Catalog, 2010, ABB Automation Products, p. 1.

ABB ACS800 OPC Support

Integration tool

DriveOPC is a software package which allows OLE for Process Control (OPC) communication between Windows applications and ABB industrial drives. It allows Object Linking and Embedding (OLE) for Process Control (OPC) communication. This OPC server is an ideal tool for integrating ABB industrial drives and commercial PC software, and creating PC based control and monitoring systems.

ABB industrial drives – ACS800, multidrives 1.1 to 5600 kW Catalog, 2010, ABB Automation Products, p. 43. (annotated)

ABB ACS800 Motion Control Support

ABB provides a set of ready-made control solutions for specific industrial drive applications. Such software adds application-dedicated features and protection without an external PLC - improving productivity and reducing costs. Function blocks are easy to program using the DriveAP PC tool.

Motion control program

The motion control program is a cost-effective solution for precision positioning and synchronization. Intelligent integrated motion control functions and versatile controllability eliminate the need for an external motion controller, even in the most demanding applications, such as materials handling, packaging, printing and the plastics industry.

Motion control has four operating modes – speed, torque, positioning and synchronization – and also provides the possibility for switching online between two selected modes.

ABB industrial drives – ACS800, multidrives 1.1 to 5600 kW Catalog, 2010, ABB Automation Products, p. 37.

ABB IRC5 Robot Controller has OPC support and Motion Control

The heart
of Robotics



IRC5

Industrial Robot Controller



IRC5 Industrial Robot Controller Data Sheet, April 2007, ABB Automation Technology Products, p. 1.

ABB IRC5 Robot Controller OPC Support

PC Interface provides the communication interface between the robot and a network PC. This is useful if you want to:

- Use an OPC Server interface for SCADA integration (delivered with the RobotWare CD)
- Use RobotStudio^{Online} to interact with the controller over a network connectivity.

Note: For local connection over the service channel, PC Interface is not required.

- Communicate with other ABB Industrial Robot Software

RobotWare Industrial Robot Controller Software IRC5, 2004, ABB Automation Products, p. 3. (annotated)

1.6.ii
each software driver comprises driver code for implementing the motion control operations associated with at least some of the driver functions, and

The term "driver code" has been construed to mean "code associated with a hardware device or group of related hardware devices, which helps generate commands necessary to perform motion control operations associated with at least some driver functions."

On information and belief, the software drivers in ABB's Industrial System 800xA systems comprise driver code for implementing the motion control operations associated with at least some of the driver functions. For example, on information and belief, the ACS800 Driver is an example hardware driver that has OPC support that is implemented using software driver code.

ACS800 Driver

16.1 Browse Tree Pane

The browse tree pane is the upper left pane within the window area. You can use it for following purposes:

- To navigate within a drive.
- To navigate within an open parameter file.
- To select a drive, which is the object of some command, like taking control.
- To select a datalogger, if the drive has more than one of them.
- To change the drive of monitored items.

A tree consists of branches shown in the browse tree pane, and items (leaves) of a branch, shown in the item list pane.

The top level name, within which current selection resides, is also shown within parentheses in the title bar. It is also shown in the status bar with the status image of the drive.

On the top level, an open parameter file (the file name if preceded by "File:") and all drives (if OPC Server has been connected) are shown.



The image displayed in front of a drive shows the status of the drive.

Image Status

- Fault and (Forward) Direction
- Fault and not (Forward) Direction
- Not Running and Warning and (Forward) Direction
- Not Running and Warning and not (Forward) Direction
- Not Running and (Forward) Direction
- Not Running and not (Forward) Direction
- Running and Warning and (Forward) Direction
- Running and Warning and not (Forward) Direction
- Running and (Forward) Direction
- Running and not (Forward) Direction
- Otherwise (status display is off-line or status cannot be read, for example)

The status image is also displayed in the status bar in front of the name, within which the current selection resides. If control of a drive is taken, the status image of the drive is shown in the drive panel toolbar in front of the name of it.

To expand a collapsed branch or to collapse an expanded branch, double-click the branch. An expanded branch can be collapsed also by clicking the minus-sign on the left of the branch. If a plus-sign is present on the left side of a collapsed branch, you can also expand it by clicking the plus-sign.

DriveWare User's Manual – Drive Window 2, 1/7/2004, ABB, p. 2-38. (annotated)

The ACS800 supports various OPC tags that are associated with motion control operations.

16.2 Item List Pane

The item list pane is the upper right pane within the window area. You can use it for following purposes:

- To view and change item values in a drive or in an open parameter file.
- To add items to and remove items from the monitor or a datalogger.
- To change the drive of monitored items.
- To copy or cut items to the clipboard.
- To change the control item set.

Each line under the title displays one item. The display is divided into three columns:

- Name, which consists of an icon that shows the status of the item, and a descriptive name. Note that the descriptive name is normally fetched from the drive or created by the OPC Server, but by executing Add New Item or Set Variable (in Monitor and Datalogger menus) you can use a name of your own.
- Value of the item (if the item is not read-protected). If quality of the value is not good, quality is shown with or instead of the value.
- OPC Address of the item. Note that items in an open parameter file are shown with an OPC Address without channel and node.

Note! Items are sorted by increasing order of channel, node, and item ID of the OPC Address. Group and parameter IDs are ordered numerically, other IDs are ordered alphabetically.

DriveWare User’s Manual – Drive Window 2, 1/7/2004, ABB, p. 2-40.

You can change the width of a column by dragging the column separator in the title.

Name	Value	OPC Address
01.01: MOTOR SPEED [rpm]	0	Par.1.1
01.05: TORQUE [%]	0	{0}{1}Par.1.5
20.01: MINIMUM SPEED [rpm]	-30	{0}{1}Par.20.1
20.07: MINIMUM FREQ [Hz]	<Read-protected>	{0}{1}Par.20.7
Running	<Bad>	{1}{1}Status:Running

The image displayed in front of a descriptive name shows the status of the item.

Image Status

- Off-line and not locked
- On-line and not locked, background of the value is also yellow
- Off-line and locked
- On-line and locked, background of the value is also yellow
- Monitored in channel 1
- Monitored in channel 2
- Monitored in channel 3
- Monitored in channel 4
- Monitored in channel 5
- Monitored in channel 6

Note that monitored items are always locked, but the locking done by the user is separated from this lock. It means that when an item is removed from the monitor, it shows the locking status set by the user (either before or during monitoring).

Monitored items can be put on-line, too. Background of the value is yellow, as all other types of on-line items.

Name	Value	OPC Address
01.03: FREQUENCY [Hz]	0	{0}{1}Par.1.3
01.04: CURRENT [A]	0	{0}{1}Par.1.4
01.07: DC BUS VOLTAGE V [V]	0	{0}{1}Par.1.7
01.10: ACS600 TEMP [C]	50	{0}{1}Par.1.10

To select an item, just click its descriptive name. Several items can be selected. You can do multiple selection with the mouse as follows:

- To select a range of items, first click the descriptive name of the item at the one end and then, with the Shift key down, click the descriptive name at the other end.
- To change selection status of a single item at a time, keep the Ctrl key down when clicking the descriptive name of the item.

DriveWare User’s Manual – Drive Window 2, 1/7/2004, ABB, p. 2-41. (annotated)

1.6.iii
 one of the software drivers in the set of software drivers is a selected software driver, where the selected software driver is the software driver associated with the selected motion control device;

On information and belief, in ABB's Industrial System 800xA systems, one of the software drivers in the set of software drivers is a selected software driver, where the selected software driver is the software driver associated with the selected motion control device. Selection can occur during configuration and/or during system operation. For example, the instructions below reflect selection of the driver for a Compact HMI 800.

Installation and Configuration of OPC Servers for PLCs

Compact HMI 800 can interface to standard PLCs. The connection is done using a specific communication driver, or via an OPC server for the PLC. To connect one or more PLCs via an OPC server follow the description below.

Install the OPC server for the PLC. The OPC server shall be installed as a Windows service if possible. Below is a typical workflow how to install and set up an OPC server to correctly interface the Compact HMI 800 software.

1. Log in as SysAdmin.
2. Install the OPC server.
3. If the server is prepared to be a Windows service, it shall be configured to run on an account SwServiceAccount.
4. Open the Plant Explorer and go to the Control Structure.
5. Select the OPC server object. See [Engineering Workflow](#) on page 47
6. Create a new object of the PLC Controller type. Name it to recognize the OPC server. (For example ABB_OPC_server_1 etc.)
7. Select the protocol to be PLC OPC client.
8. Click Edit Driver.
9. If the OPC server is located in another node than the Compact HMI 800 server, fill in the server node name.
10. Select the OPC server in the OPC server drop-down list.
11. Press OK and then the Create button.

Now the OPC server is installed.

Connection to Third Party Alarm & Event OPC Server

To get time tagged alarms from the controller into the Compact HMI 800 alarm list, the OPC AE client function of Compact HMI 800 should be used.

To associate an alarm with a suitable Aspect Object (normally a PLC signal object) an OPC Source Name aspect, or an Aspect Name aspect, should be created on the mentioned signal object.

Note. The Name Property in the OPC Source Name should be the same as exposed by the OPC AE server. Do not check "Is an Event" and "Is an Alarm" in "Alarm Event Configuration".

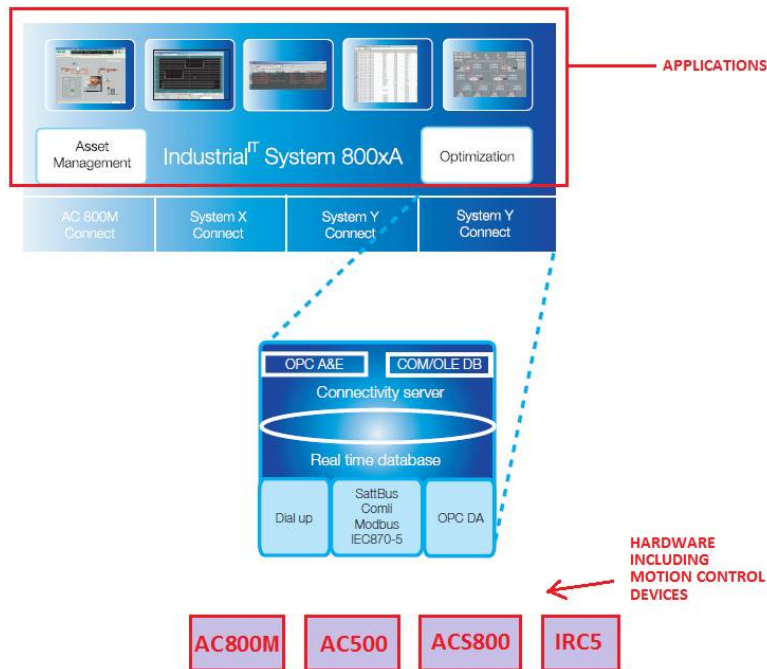
Avoid identical names of PLC Process Object and PLC Signal Object (both Aspect Objects).

Industrial IT Compact HMI 800, System Version 4.1 – Getting Started, ABB, 2005, pp. 34-35. (annotated)

1.7 an application program comprising a series of component functions, where the application program defines the steps for operating motion control devices in a desired manner; and

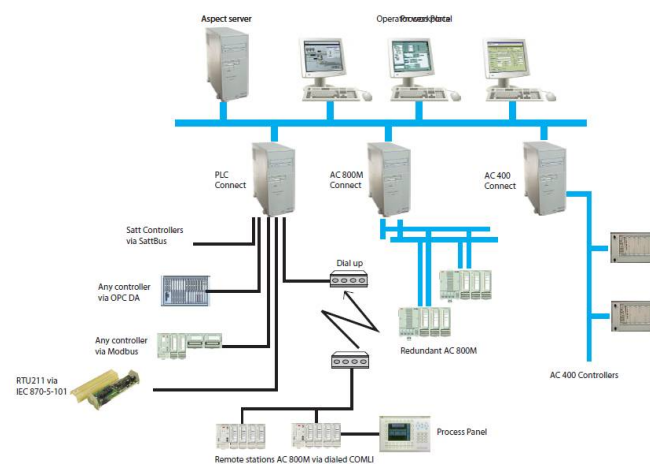
The term "application program" has been construed to mean "a software program designed to handle specific tasks."

On information and belief, systems based on ABB's Industrial System 800xA comprise one or more application programs comprising a series of component functions, where the application program defines the steps for operating motion control devices in a desired manner. Specifically, the application programs are the various software programs comprising component function calls that can be made on the Connectivity Server.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

On information and belief, applications within the IndustrialIT System 800xA include the Aspect Server and Compact HMI 800.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 1.

Compact HMI 800 is an example of one such application, as shown below.

System Overview

The Industrial^{IT} Compact HMI 800 is designed to be an HMI to any kind of automation solution. It interfaces to AC 800M and most other PLCs found on the market. It is based on the Industrial^{IT} Base standard system. To make it easy to set up and install the product, some configurations are pre-set at delivery.

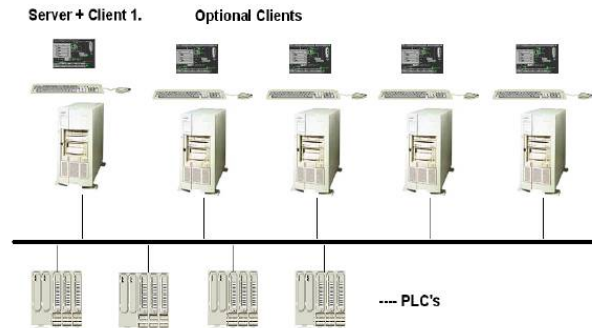


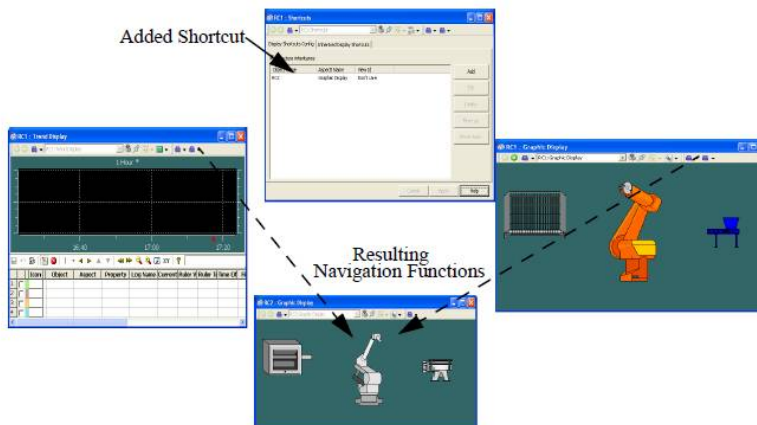
Figure 1. System Overview

Industrial IT Compact HMI 800, System Version 4.1 – Getting Started, ABB, 2005, pp. 19.

Adding Shortcuts for Navigation

In Compact HMI 800 the navigation is done in an object oriented way. When defining shortcuts for navigation, these are available from all aspects of an object by just adding the shortcut to the object itself, once as an aspect. This means that by adding a shortcut to the RC1 Robot Cell object, we can use the shortcut from all views of the cell. e.g. if we add a shortcut to RC1 that points to RC2, we will be able to navigate to RC2 aspects from RC1 Graphical Display, RC1 Trends Display, RC1 Mechanical Drawing etc.

The figure below shows how the navigation works:



By adding a shortcut to the RC1 object, pointing to the RC2 Graphic Display, the RC2 Graphic Display can be accessed both from the RC1 Graphics Display and the RC1 Trend Display.

Industrial IT Compact HMI 800, System Version 4.1 – Getting Started, ABB, 2005, pp. 78.

On information and belief, additional (but not limited to) application program(s) included in each of the products listed below provide similar functionality to that described for the products herein:

Product
Panel 800
Industrial IT System 800xA Operations
System 800xA Smart Client
System 800xA Engineering Tools
System 800xA

1.8
 a motion control component for generating the sequence of control commands for controlling the selected motion control device based on the component functions of the application program, the component code associated with the component functions, and the driver code associated with the selected software driver.

“Control command(s)” has been construed to mean “command codes in hardware language, which instruct a motion control device to perform motion control operations.”

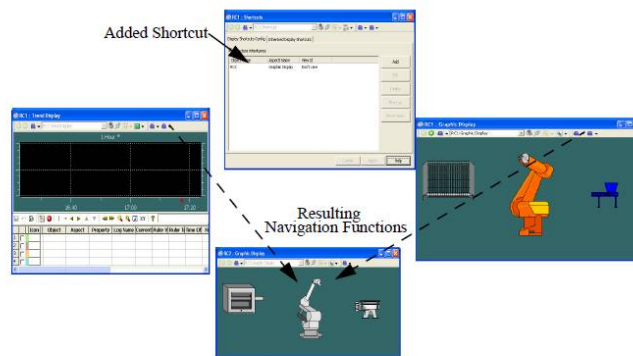
The term “motion control component” has been construed to mean “a binary software module that associates component functions with driver functions by calling the driver functions.”

On information and belief, systems based on ABB’s Industrial System 800xA comprise a motion control component for generating a sequence of control commands for controlling the selected motion control devices based on the component functions of the application program, the component code associated with the component functions, and the driver code associated with the selected software driver. On information and belief, the OPC DA server selected transmits command codes to the associated motion control device to read and write data as well as set-up device based (if supported) event handlers for events and alarms based on control exerted by the motion control component. SEE 1.4, 1.5, 1.6 and 1.7 above.

Adding Shortcuts for Navigation

In Compact HMI 800 the navigation is done in an object oriented way. When defining shortcuts for navigation, these are available from all aspects of an object by just adding the shortcut to the object itself, once as an aspect. This means that by adding a shortcut to the RC1 Robot Cell object, we can use the shortcut from all views of the cell. e.g. if we add a shortcut to RC1 that points to RC2, we will be able to navigate to RC2 aspects from RC1 Graphical Display, RC1 Trends Display, RC1 Mechanical Drawing etc.

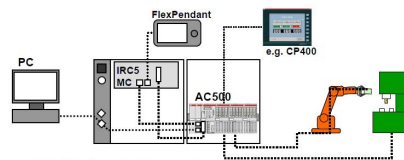
The figure below shows how the navigation works:



By adding a shortcut to the RC1 object, pointing to the RC2 Graphic Display, the RC2 Graphic Display can be accessed both from the RC1 Graphics Display and the RC1 Trend Display.

Industrial IT Compact HMI 800, System Version 4.1 – Getting Started, ABB, 2005, pp. 78.

Application Example 2: Cell Control

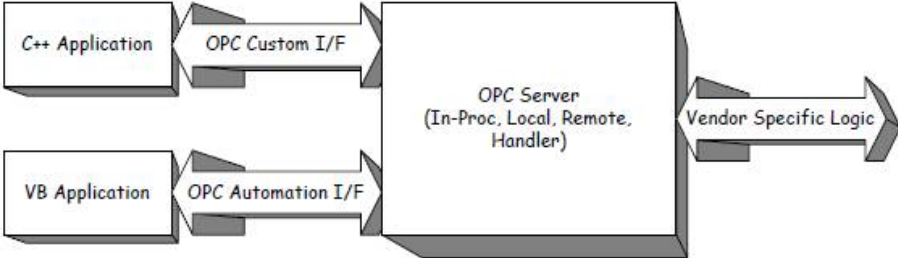


- Control of the complete cell
 - Peripheral equipment, like clamping, machines, in/out feeders, turntables etc
 - Robot program selection and starting (PLC takes master role)
 - Operator communication
 - Master bus coupler an alternative to discrete signals

ABB Integrated PLC AC500 in IRC5, Slide 24.

On information and belief, additional (but not limited to) application program(s) included in each of the products listed below provides similar functionality to that described for the products herein:

Product
Panel 800
Industrial IT System 800xA Operations
System 800xA Smart Client
System 800xA Engineering Tools
System 800xA

Claim.Element	Analysis
<p>2</p> <p>2. A system as recited in claim 1, in which the software drivers comprise driver code for implementing all of the core driver functions.</p>	<p>On information and belief, the software drivers (discussed above) comprise driver code for implementing all of the core driver functions, which are required by the OPC Specification.</p> <p>2.3 General OPC Architecture and Components</p> <p>OPC is a specification for two sets of interfaces; the OPC Custom Interfaces and the OPC Automation interfaces. A revised automation interface will be provided with release 2.0 of the OPC specification. This is shown in Figure 2-5.</p>  <p style="text-align: center;"><i>Figure 2-5 - The OPC Interfaces</i></p> <p>The OPC Specification specifies COM interfaces (what the interfaces are), not the implementation (not the how of the implementation) of those interfaces. It specifies the behavior that the interfaces are expected to provide to the client applications that use them.</p> <p>Included are descriptions of architectures and interfaces that seemed most appropriate for those architectures. Like all COM implementations, the architecture of OPC is a client-server model where the OPC Server component provides an interface to the OPC objects and manages them.</p> <p>There are several unique considerations in implementing an OPC Server. The main issue is the frequency of data transfer over non-sharable communications paths to physical devices. Thus, we expect that the OPC Server will either be a local or remote EXE which includes code that is responsible for efficient data collection from a physical device.</p> <p style="border: 1px solid red; padding: 5px;">An OPC client application communicates to an OPC server through the specified OPC custom and automation interfaces. OPC servers must implement the custom interface, and optionally may implement the automation interface.</p> <p>OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 9. (annotated)</p> <p style="border: 1px solid red; padding: 5px;">2.5 Required Interface Definition</p> <p style="border: 1px solid red; padding: 5px;">OPC server developers must implement all functionality of required interfaces. An OPC client communicates to an OPC server by calling functions from the OPC required interfaces.</p> <p>2.6 Optional Interface Definition</p> <p>OPC server developers may implement the functionality of the optional interfaces</p> <p>An optional interface is one that the server developer may elect to implement. When an OPC Server supports an optional interface, all functions within that optional interface must be implemented, even if the function just returns E_NOTIMPL. An OPC client that wishes to use the functionality of an optional interface will query the OPC server for the optional interface. The client must be designed to not require that this optional interface exist.</p> <p>OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 11. (annotated)</p>

4. OPC Custom Interface

4.1 Overview of the OPC Custom Interface

The OPC Custom Interface Objects include the following custom objects:

- OPCServer
- OPCGroup

The interfaces and behaviors of these objects are described in detail in this chapter. Developers of OPC servers are required to implement the OPC objects by providing the functionality defined in this chapter.

This chapter also references and defines expected behavior for the standard OLE interfaces. Interfaces that an OPC server and an OPC client are required to implement when building OPC compliant components.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 27. (annotated)

3.1.2 OPCGroup Object

IOPCGroupStateMgt

HRESULT GetState(pUpdateRate, pActive, ppName, pTimeBias, pPercentDeadband, pLCID,
 phClientGroup, phServerGroup)
HRESULT SetState(pRequestedUpdateRate, pRevisedUpdateRate, pActive, pTimeBias,
 pPercentDeadband, pLCID, phClientGroup)
HRESULT SetName(szName);
HRESULT CloneGroup(szName, riid, ppUnk);

IOPCPublicGroupStateMgt (optional)

HRESULT GetState(pPublic);
HRESULT MoveToPublic(void);

IOPCSyncIO

HRESULT Read(dwSource, dwNumItems, phServer, pItemValues, ppErrors)
HRESULT Write(dwNumItems, phServer, pItemValues, ppErrors)

IOPCAsyncIO

HRESULT Read(dwConnection, dwSource, dwNumItems, phServer, pTransactionID, ppErrors,)
HRESULT Write(dwConnection, dwNumItems, phServer, pItemValues, pTransactionID,
 ppErrors);
HRESULT Cancel(dwTransactionID);
HRESULT Refresh(dwConnection, dwSource, pTransactionID);

IOPCItemMgt

HRESULT AddItems(dwNumItems, pItemArray, ppAddResults, ppErrors)
HRESULT ValidateItems(dwNumItems, pItemArray, bBlobUpdate, ppValidationResults,
 ppErrors)
HRESULT RemoveItems(dwNumItems, phServer, ppErrors)
HRESULT SetActiveState(dwNumItems, phServer, bActive, ppErrors)
HRESULT SetClientHandles(dwNumItems, phServer, phClient, ppErrors)
HRESULT SetDatatypes(dwNumItems, phServer, pRequestedDatatypes, ppErrors)
HRESULT CreateEnumerator(riid, ppUnk)

IDataObject

HRESULT DAdvise(pFmt, adv, pSnk, pConnection);
HRESULT DUnadvise(Connection);
Note: all other functions can be stubs which return E_NOTIMPL.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 18. (annotated)

Claim.Element	Analysis
<p>3</p>	
<p>3. A system as recited in claim 2, in which the software drivers comprise driver code for implementing at least some of the extended driver functions.</p>	<p>On information and belief, the software drivers (discussed above) comprise driver code for implementing at least some of the extended driver functions, as required by the OPC Specification.</p> <p>4. OPC Custom Interface</p> <p>4.1 Overview of the OPC Custom Interface</p> <p>The OPC Custom Interface Objects include the following custom objects:</p> <ul style="list-style-type: none"> • OPCServer • OPCGroup <p>The interfaces and behaviors of these objects are described in detail in this chapter. Developers of OPC servers are required to implement the OPC objects by providing the functionality defined in this chapter.</p> <p>This chapter also references and defines expected behavior for the standard OLE interfaces. Interfaces that an OPC server and an OPC client are required to implement when building OPC compliant components.</p> <p>OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 27. (annotated)</p> <p>3.1.2 OPCGroup Object</p> <p>IOPCGroupStateMgt</p> <p>HRESULT GetState(pUpdateRate, pActive, ppName, pTimeBias, pPercentDeadband, pLCID, phClientGroup, phServerGroup) HRESULT SetState(pRequestedUpdateRate, pRevisedUpdateRate, pActive, pTimeBias, pPercentDeadband, pLCID, phClientGroup) HRESULT SetName(szName); HRESULT CloneGroup(szName, riid, ppUnk);</p> <p>IOPCPublicGroupStateMgt (optional)</p> <p>HRESULT GetState(pPublic); HRESULT MoveToPublic(void);</p> <p>IOPCSyncIO</p> <p>HRESULT Read(dwSource, dwNumItems, phServer, pItemValues, ppErrors) HRESULT Write(dwNumItems, phServer, pItemValues, ppErrors)</p> <p>IOPCAsyncIO</p> <p>HRESULT Read(dwConnection, dwSource, dwNumItems, phServer, pTransactionID, ppErrors.) HRESULT Write(dwConnection, dwNumItems, phServer, pItemValues, pTransactionID, ppErrors); HRESULT Cancel (dwTransactionID); HRESULT Refresh(dwConnection, dwSource, pTransactionID);</p> <p>IOPCItemMgt</p> <p>HRESULT AddItems(dwNumItems, pItemArray, ppAddResults, ppErrors) HRESULT ValidateItems(dwNumItems, pItemArray, bBlobUpdate, ppValidationResults, ppErrors) HRESULT RemoveItems(dwNumItems, phServer, ppErrors) HRESULT SetActiveState(dwNumItems, phServer, bActive, ppErrors) HRESULT SetClientHandles(dwNumItems, phServer, phClient, ppErrors) HRESULT SetDatatypes(dwNumItems, phServer, pRequestedDatatypes, ppErrors) HRESULT CreateEnumerator(riid, ppUnk)</p> <p>IDataObject</p> <p>HRESULT DAdvise(pFmt, adv, pSnk, pConnection); HRESULT DUnadvise(Connection); Note: all other functions can be stubs which return E_NOTIMPL.</p> <p>OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 18. (annotated)</p>

Claim.Element	Analysis
4	
<p>4. A system as recited in claim 3, in which:</p>	<p>See claim 3 above.</p>
<p>4.1 non-supported extended driver functions are extended driver functions having no driver code associated therewith; and</p>	<p>On information and belief, extended driver functions exist that do not have associated driver code.</p> <p>4.4.8 IDataObject</p> <p>The OPC Specification requires the IDataObject to be implemented for the OPC servers.</p> <p>IDataObject is implemented on the OPCGroup rather than on the individual items. This allows the creation of an Advise connection between the client and the group using the OPC Data Stream Formats for the efficient data transfer.</p> <p>It is required that the following methods be supported.</p> <ul style="list-style-type: none"> DAdvise DUnadvise <p>Because the IDataObject deals with a STREAM rather than individual items, the following methods do not need to be supported (they can be implemented as stubs which return E_NOTIMPL).</p> <ul style="list-style-type: none"> GetData GetDataHere GetCanonicalFormatEtc <p>The server vendor may chose to implement additional methods on the IDataObject. It is the intent of this design that data items be transferred to applications primarily via the Advise connection or via the Synchronous or Asynchronous Read methods.</p> <p>OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 98.</p>

4.2
the motion control component generates control commands based on the driver code associated with a combination of the core driver functions to emulate the motion control operations associated with at least some of the nonsupported extended driver functions.

And, on further information and belief, the motion control component (discussed above) generates control commands based on the driver code associated with a combination of core driver functions to emulate the motion control operations associated with at least some of the non-supported extended driver functions.

4.4.6 IOPCAsyncIO

IOPCAsyncIO allows a client to perform asynchronous read and write operations to a server. The operations will be 'queued' and the function will return immediately so that the client can continue to run. Each operation is treated as a 'transaction' and is associated with a transaction ID. As the operations are completed, a callback will be made to the IAdvise Sink in the client (if one has been established). The information in the callback will indicate the transaction ID and the error results. By convention, 0 is an invalid transaction id.

Also the expected behavior is that for any one transaction to Async Read, Write and Refresh, ALL of the results of that transaction will be returned in a single call to OnDataChange.

A server must be able to 'queue' at least one transaction of each type (read, write, refresh) for each group. It is acceptable for a server to return an error (CONNECT_E_ADVISELIMIT) if more than one transaction of the same type is performed on the same group by the same client. Server vendors may of course support queueing of additional transactions if they wish.

All operations are expected to complete even if they complete with an error. The concept of 'time-out' is not explicitly addressed in this specification however it is expected that where appropriate the server will internally implement any needed time-out logic.

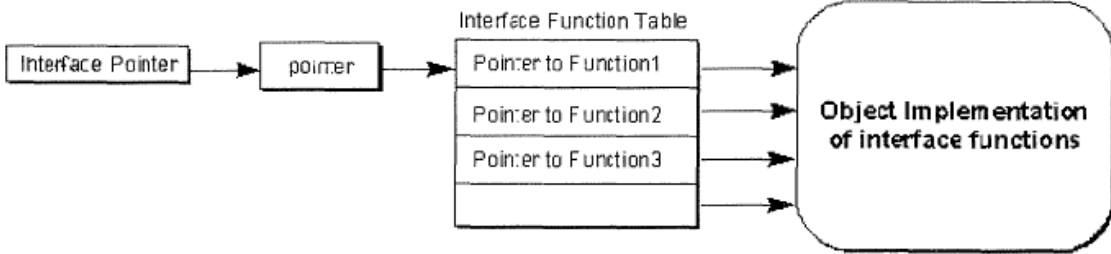
Client Implementation Note:

The Transaction ID is generated by the Server and returned to the client in the callback. Some clients may want to save the ID returned by the server in some list of 'outstanding transactions' in order to verify completion of a transaction. This could be complicated if the OnDataChange callback occurs before the client has saved the returned ID.

Note: Version 1.0 of this specification suggested an approach involving critical sections. However, depending on the mix of client and server threading models used, it has been found in practice that the OnDataChange callback can occur within the same thread as the Read or Write and in fact can occur before the Read or Write returns to the caller. Clearly, critical sections cannot resolve this case.

Although it has also been found in practice that many clients do not actually need to record the transaction ID (the Group's ClientHandle is generally sufficient to identify the returned data), the following possible approach is suggested for those cases where this is needed.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 85.

Claim.Element	Analysis
<p>5</p>	<p>See Claim 4 above.</p>
<p>5. A system as recited in claim 4, further comprising:</p> <p>5.1 an extended function pointer table that maps the non-supported extended driver functions to the combination of core driver functions employed to emulate the non-supported extended driver functions;</p> <p>and</p>	<p>On information and belief, systems based on ABB's Industrial System 800xA for example comprise an extended function pointer table maps the non-supported extended driver functions of the combination of core driver functions employed to emulate the non-supported extended driver functions, such as the function pointer table structure used in COM and OLE, as set forth below.</p> <p>At run time, an "interface" is always seen as a pointer typed with an IID. The pointer itself points to another pointer that points to a table that holds the addresses of the implementation of each member function in the interface. This binary structure, illustrated in Figure 3, is a core standard of COM, and all of COM and OLE depend upon this standard for interoperability between software components written in arbitrary languages. As long as a compiler can reduce language structures down to this binary standard, it doesn't matter how one programs a component or a client—the point of contact is a run-time binary standard.</p>  <p>Figure 3. The binary interface structure</p> <p>It is this exact interface structure that provides the ability to marshal one of these pointers between processes and machines, as described in the section titled "Local/Remote Transparency."</p> <p>To "implement an interface" on some object means to build this exact binary structure in memory and provide the pointer to the structure. This is what we want instead of having clients do it themselves! You can do this in assembly language if you want, but higher-level languages, especially C++, build the structures automatically. In fact, the interface structure is, by design, identical to that used for C++ virtual functions—programming COM and OLE in C++ is highly convenient.</p> <p>This is also why COM calls the table portion the "vtable" and the pointer to that table "lpVtbl." A pointer to an interface is a pointer to lpVtbl, which points to the vtable. Because this is what C++ expects to see, calling an interface member given an interface pointer is just like calling a C++ object's member function. That is, if I have a pointer to ISpellChecker in the variable <i>pSC</i>, I can call a member like this:</p> <pre>pSC->LookUpWord(pszWord);</pre> <p>Because the interface definition describes all the argument types for each interface member function, the compiler will do all the type checking for you. As a client, you never have to define function prototypes for these things yourself.</p> <p>In C, the same call would look like this, with an explicit indirection through lpVtbl and passing the interface pointer as the first argument:</p> <pre>pSC->lpVtbl->LookUpWord(pSC, pszWord);</pre> <p>What OLE Is Really About, Kraig Brockschmidt, July 1996, pp. 17-18.</p>

4. OPC Custom Interface

4.1 Overview of the OPC Custom Interface

The OPC Custom Interface Objects include the following custom objects:

- OPCServer
- OPCGroup

The interfaces and behaviors of these objects are described in detail in this chapter. Developers of OPC servers are required to implement the OPC objects by providing the functionality defined in this chapter.

This chapter also references and defines expected behavior for the standard OLE interfaces. Interfaces that an OPC server and an OPC client are required to implement when building OPC compliant components.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 27. (annotated)

3.1.2 OPCGroup Object

IOPCGroupStateMgt

HRESULT GetState(pUpdateRate, pActive, ppName, pTimeBias, pPercentDeadband, pLCID, phClientGroup, phServerGroup)
 HRESULT SetState(pRequestedUpdateRate, pRevisedUpdateRate, pActive, pTimeBias, pPercentDeadband, pLCID, phClientGroup)
 HRESULT SetName(szName);
 HRESULT CloneGroup(szName, riid, ppUnk);

IOPCPublicGroupStateMgt (optional)

HRESULT GetState(pPublic);
 HRESULT MoveToPublic(void);

IOPCSyncIO

HRESULT Read(dwSource, dwNumItems, phServer, ppItemValues, ppErrors)
 HRESULT Write(dwNumItems, phServer, pItemValues, ppErrors)

IOPCAsyncIO

HRESULT Read(dwConnection, dwSource, dwNumItems, phServer, pTransactionID, ppErrors,.)
 HRESULT Write(dwConnection, dwNumItems, phServer, pItemValues, pTransactionID, ppErrors);
 HRESULT Cancel(dwTransactionID);
 HRESULT Refresh(dwConnection, dwSource, pTransactionID);

IOPCItemMgt

HRESULT AddItems(dwNumItems, pItemArray, ppAddResults, ppErrors)
 HRESULT ValidateItems(dwNumItems, pItemArray, bBlobUpdate, ppValidationResults, ppErrors)
 HRESULT RemoveItems(dwNumItems, phServer, ppErrors)
 HRESULT SetActiveState(dwNumItems, phServer, bActive, ppErrors)
 HRESULT SetClientHandles(dwNumItems, phServer, phClient, ppErrors)
 HRESULT SetDatatypes(dwNumItems, phServer, pRequestedDatatypes, ppErrors)
 HRESULT CreateEnumerator(riid, ppUnk)

IDataObject

HRESULT DAdvise(pFmt, adv, pSnk, pConnection);
 HRESULT DUnadvise(Connection);
 Note: all other functions can be stubs which return E_NOTIMPL.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 18. (annotated)

<p>5.2 the motion control component generates the control commands further based on the contents of the extended function pointer table.</p>	<p>On information and belief, the motion control component (discussed above) generates control commands further based on the contents of the extended function pointer table (discussed above in 5.1).</p> <p>4. OPC Custom Interface</p> <p>4.1 Overview of the OPC Custom Interface</p> <p>The OPC Custom Interface Objects include the following custom objects:</p> <ul style="list-style-type: none">• OPCServer• OPCGroup <p>The interfaces and behaviors of these objects are described in detail in this chapter. Developers of OPC servers are required to implement the OPC objects by providing the functionality defined in this chapter.</p> <p>This chapter also references and defines expected behavior for the standard OLE interfaces. Interfaces that an OPC server and an OPC client are required to implement when building OPC compliant components.</p> <p>OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 27. (annotated)</p>
--	--

Claim.Element	Analysis
6	

6.
 A system as recited in claim 5, in which the extended function pointer table contains pointers for both supported and nonsupported extended driver functions, where the pointers for the supported extended driver functions point to driver code for implementing the supported extended driver functions and the pointers for the non-supported extended driver functions point to the combination of core driver functions that emulate the non-supported extended functions.

On Information and belief, the extended function pointer table (discussed above) contains pointers for both supported and non-supported extended driver functions, where the pointers for the supported extended driver functions point to driver code for implementing the supported extended driver functions and the pointers for the non-supported extended driver functions point to the combination of core driver functions that emulate the non-supported extended functions. (See also OPC Alarms and Events Standard).

The following pictures show the overall flow of alarms and events, starting in the AC 800M controller. Buffering of events takes place at several levels in the system. Such buffers, or queues, are shown. Presentation and acknowledgement can be made in several ways as discussed above.

Alarm and Event texts in AC 800M applications shall be defined according to ISO 8859/ISO Latin-1 8-bit ASCII character set.

IndustrialIT Extended Automation System 800xA System Version 3.1 System Guide, January 2004, ABB, p. 30.

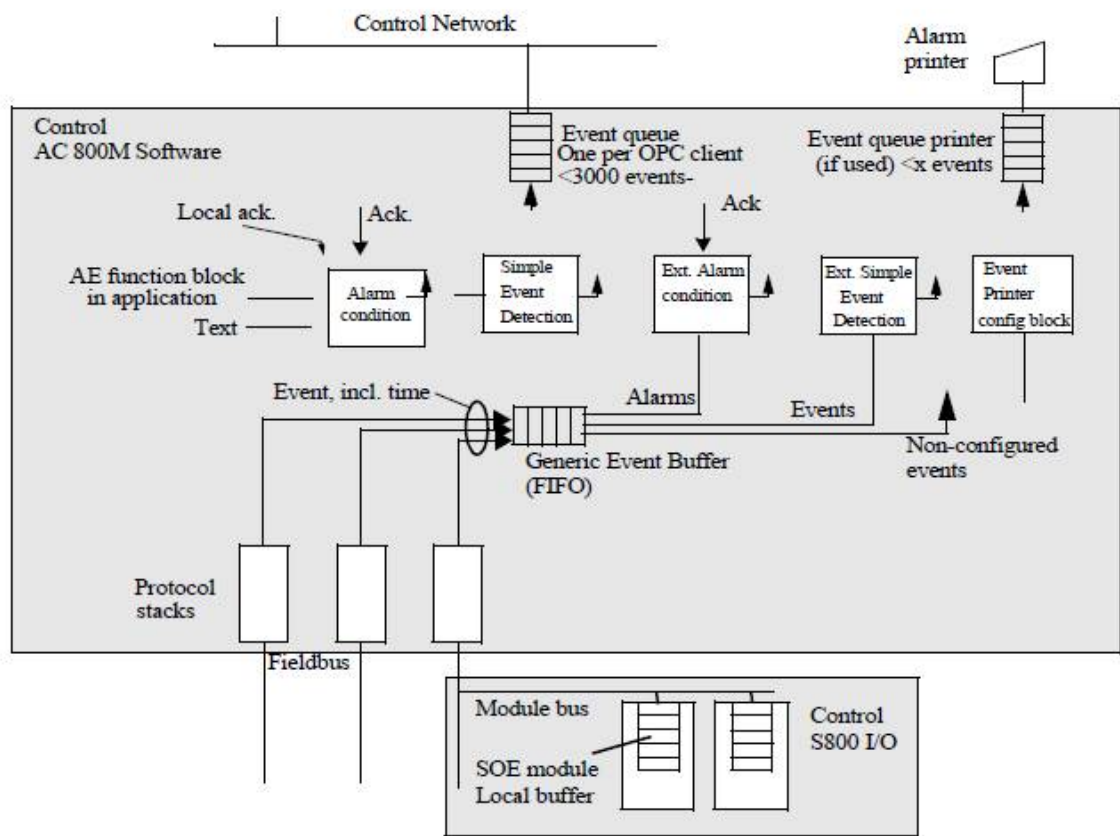


Figure 3. Alarm & Event management, controller level

IndustrialIT Extended Automation System 800xA System Version 3.1 System Guide, January 2004, ABB, p. 31.

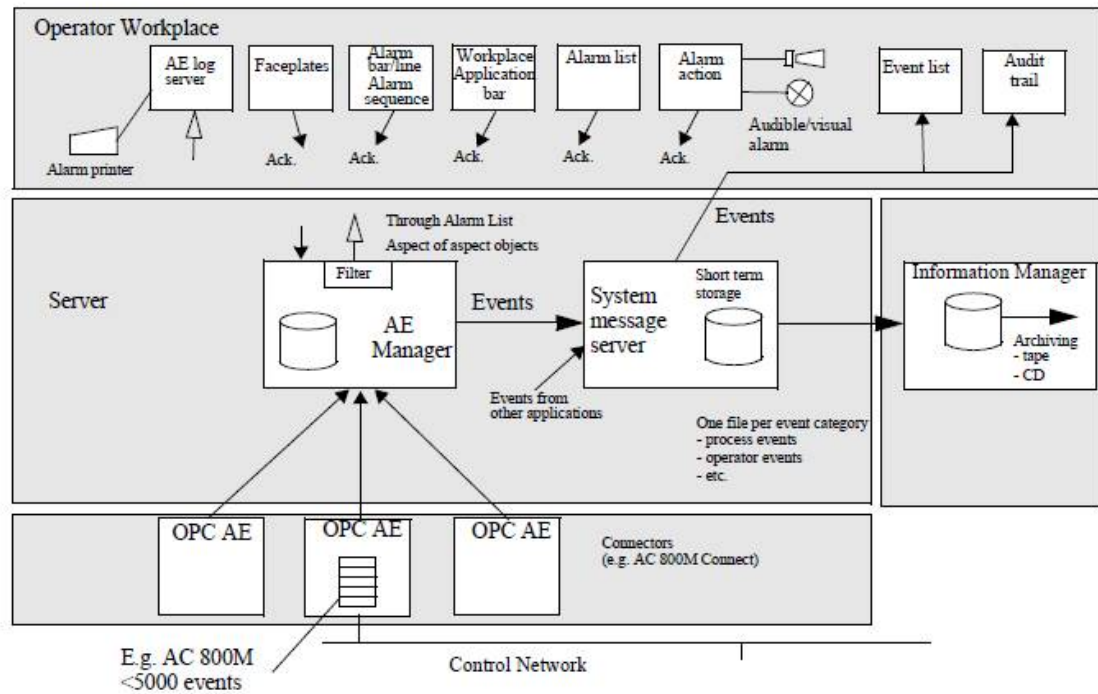


Figure 4. Alarm & Event management, workplace and server level

IndustrialIT Extended Automation System 800xA System Version 3.1 System Guide, January 2004, ABB, p. 32.

4.4.6 IOPCAsyncIO

IOPCAsyncIO allows a client to perform asynchronous read and write operations to a server. The operations will be 'queued' and the function will return immediately so that the client can continue to run. Each operation is treated as a 'transaction' and is associated with a transaction ID. As the operations are completed, a callback will be made to the IAdvise Sink in the client (if one has been established). The information in the callback will indicate the transaction ID and the error results. By convention, 0 is an invalid transaction id.

Also the expected behavior is that for any one transaction to Async Read, Write and Refresh, ALL of the results of that transaction will be returned in a single call to onDataChange.

A server must be able to 'queue' at least one transaction of each type (read, write, refresh) for each group. It is acceptable for a server to return an error (CONNECT_E_ADVISELIMIT) if more than one transaction of the same type is performed on the same group by the same client. Server vendors may of course support queueing of additional transactions if they wish.

All operations are expected to complete even if they complete with an error. The concept of 'time-out' is not explicitly addressed in this specification however it is expected that where appropriate the server will internally implement any needed time-out logic.

OLE for Process Control Data Access Standard (UPDATED) Version 1.0A, 9/11/1997, OPC Foundation, p. 85.

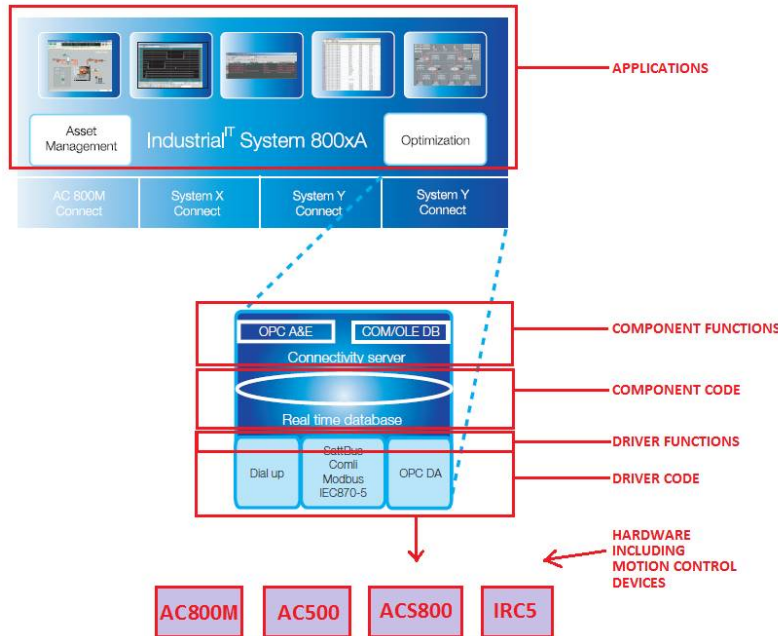
<p>7</p>	
<p>7. A system as recited in claim 1, further comprising: means for determining a driver unit system employed by the software drivers; and means for converting an application unit system employed by the application program into the driver unit system.</p>	<p>On information and belief, systems based on ABB's Industrial System 800xA further comprise means for determining a driver unit system employed by the software drivers (discussed above); and means for converting an application unit system employed by the application program (discussed above) into the driver unit system .</p>

8

8. A system as recited in claim 1, further comprising: a plurality of destinations of control commands, with one of the plurality of destinations of control commands being a selected destination of control commands;

“Control command(s)” has been construed to mean “command codes in hardware language, which instruct a motion control device to perform motion control operations.”

On information and belief, systems based on ABB’s Industrial System 800xA comprise a system for generating a sequence of control commands for controlling a selected motion control device selected from a group of supported motion control devices, as shown below.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

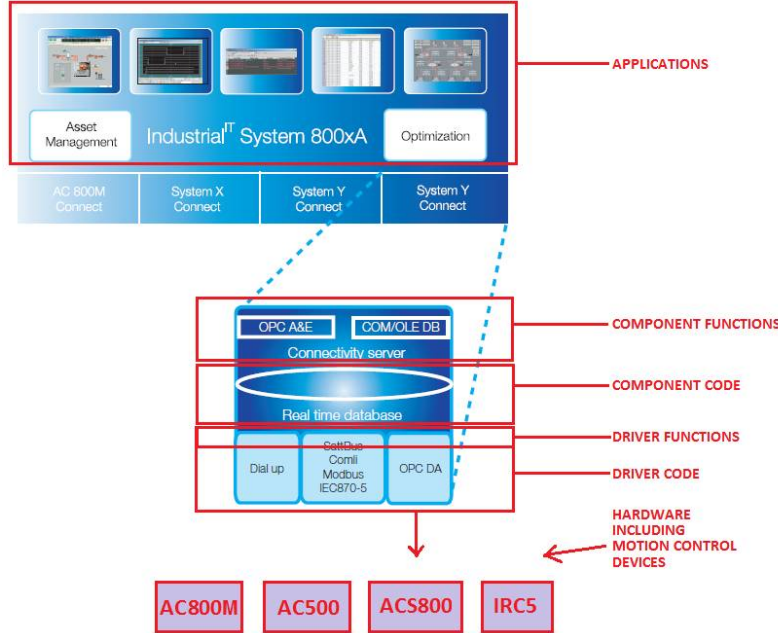
PLC Connect is an option and provides integrated connectivity to System 800xA for ABB and third-party PLC-type controllers.

PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2.

8.1.i

a plurality of streams, where each stream contains transmit stream code that determines how the control commands are to be transferred to at least one of the plurality of destinations of control commands;

Systems based on ABB's Industrial System 800xA communicate with a wide variety of motion control devices, including through PLC Connect.



PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

On information and belief, communication is enabled via the OPC connectors and other techniques that consist of streams comprising stream code, which determine how control commands are transmitted to the destination.

How OPC Works (Functional View)

The OPC "device abstraction" is realized by using two, specialized OPC components called an OPC Client and OPC Server. Each of which is described in a following section. What's important to note is that just because the Data Source and Data Sink can communicate with each other via OPC does not mean their respective native protocols are no longer necessary or have been replaced by OPC. Instead, these native protocols and/or interfaces are still present, but only communicate with one of the two OPC components. In turn, the OPC components exchange information amongst each other and the loop is closed. Data can travel from the Application to the Device without having one talk directly to the other.

Benefits of using OPC Connectivity

At first glance, trading a single Custom Driver for two OPC components (OPC Client and OPC Server) may not look like much of an improvement but as experience has shown, it is. Following are some key benefits of using OPC:

1. An OPC enabled Application can freely communicate with any OPC-enabled Data Source visible to it on the network without the need for any driver software, specific to the Data Source.
2. OPC-enabled applications can communicate with as many OPC-enabled Data Sources as they need. There is no inherent OPC limitation on the number of connections made.
3. Today OPC is so common that there's an OPC connector available for almost every modern and legacy device on the market. It's easy to get started using OPC.

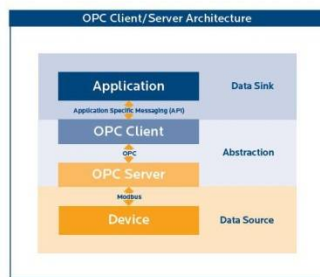


Figure 3: OPC Client/Server Architecture - A closer look at the OPC abstraction layer reveals two components; an OPC Client and an OPC Server. OPC specifications define the messaging between these two components.

OPC: The Ins and Outs to what it's About, page 4. (annotated)

8.1.ii

and stream control means for communicating the control commands to the selected destination of control commands based on the transmit stream code contained by the stream associated with the selected destination of control commands.

On information and belief, systems based on ABB's Industrial System 800xA further comprise stream control means for communicating the control commands to the selected destination of control commands based on the transmit stream code contained by the stream associated with the selected destination of control commands.

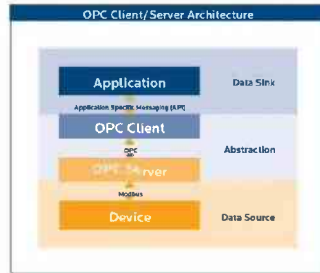


Figure 3: OPC Client/Server Architecture - A closer look at the OPC abstraction layer reveals two components: an OPC Client and an OPC Server. OPC specifications define the messaging between these two components.

How OPC Works (Functional View)

The OPC "device abstraction" is realized by using two, specialized OPC components called an OPC Client and OPC Server. Each of which is described in a following section. What's important to note is that just because the Data Source and Data Sink can communicate with each other via OPC does not mean their respective native protocols are no longer necessary or have been replaced by OPC. Instead, these native protocols and/or interfaces are still present, but only communicate with one of the two OPC components. In turn, the OPC components exchange information amongst each other and the loop is closed. Data can travel from the Application to the Device without having one talk directly to the other.

Benefits of using OPC Connectivity

At first glance, trading a single Custom Driver for two OPC components (OPC Client and OPC Server) may not look like much of an improvement but as experience has shown, it is. Following are some key benefits of using OPC:

1. An OPC enabled Application can freely communicate with any OPC-enabled Data Source visible to it on the network without the need for any driver software, specific to the Data Source.
2. OPC-enabled applications can communicate with as many OPC-enabled Data Sources as they need. There is no inherent OPC limitation on the number of connections made.
3. Today OPC is so common that there's an OPC connector available for almost every modern and legacy device on the market. It's easy to get started using OPC.

OPC: The Ins and Outs to what it's About, page 4. (annotated)

9

9. A system as recited in claim 8, in which certain of the destinations of control commands generate response data, wherein: the streams associated with the destinations of control commands that generate response data are each associated with response stream code; and the stream control means processes the response data based on the response stream code.

The term "providing response stream code" has been construed to mean "providing software code that defines the specific protocol for receiving response data."

On information and belief, in systems based on ABB's Industrial System 800xA certain of the destinations of control commands (e.g., motion control devices) generate response data, wherein: the streams associated with the destinations of control commands that generate response data are each associated with response stream code; and the stream control means processes the response data based on the response stream code. Specifically, on information and belief PLC Connect implements bi-directional communications between motion control devices and OPC servers via OPC Connectors and other techniques.

OPC SERVERS

What is an OPC Server?

An OPC Server is a software application, a "standardized" driver, specifically written to comply with one or more OPC specifications. The word "server" in "OPC Server" does not refer to the type of computer being used but instead reflects its relationship with its OPC counterpart, the OPC Client.

What Do OPC Servers Do?

OPC Servers are connectors that may be thought of as translators between the OPC world and a Data Source's native communication protocol or interface. Since OPC is bi-directional, this means OPC Servers can both read-from and write-to a Data Source. The OPC Client/OPC Server relationship is a Master/Slave one which means one OPC Server will only transfer data to/from a Data Source if an OPC Client commands it to.

What Types of Data Sources (Devices) can OPC Servers Communicate With?

OPC Servers can communicate with virtually any Data Source whose output can be read or written to via electronic means. A brief list of possible Data Sources includes: devices, PLCs, DCSs, RTUs, electronic scales, databases, historians, web-pages, and automatically updating CSV files. To communicate with any of these devices requires only the use of an OPC Server that employs the appropriate native protocol or interface. Once such an OPC Server is configured, any OPC enabled application (with permission) can begin communicating with the Data Source without concern for how the Data Source communicates natively.

"OPC Servers can communicate with virtually any Data Source."

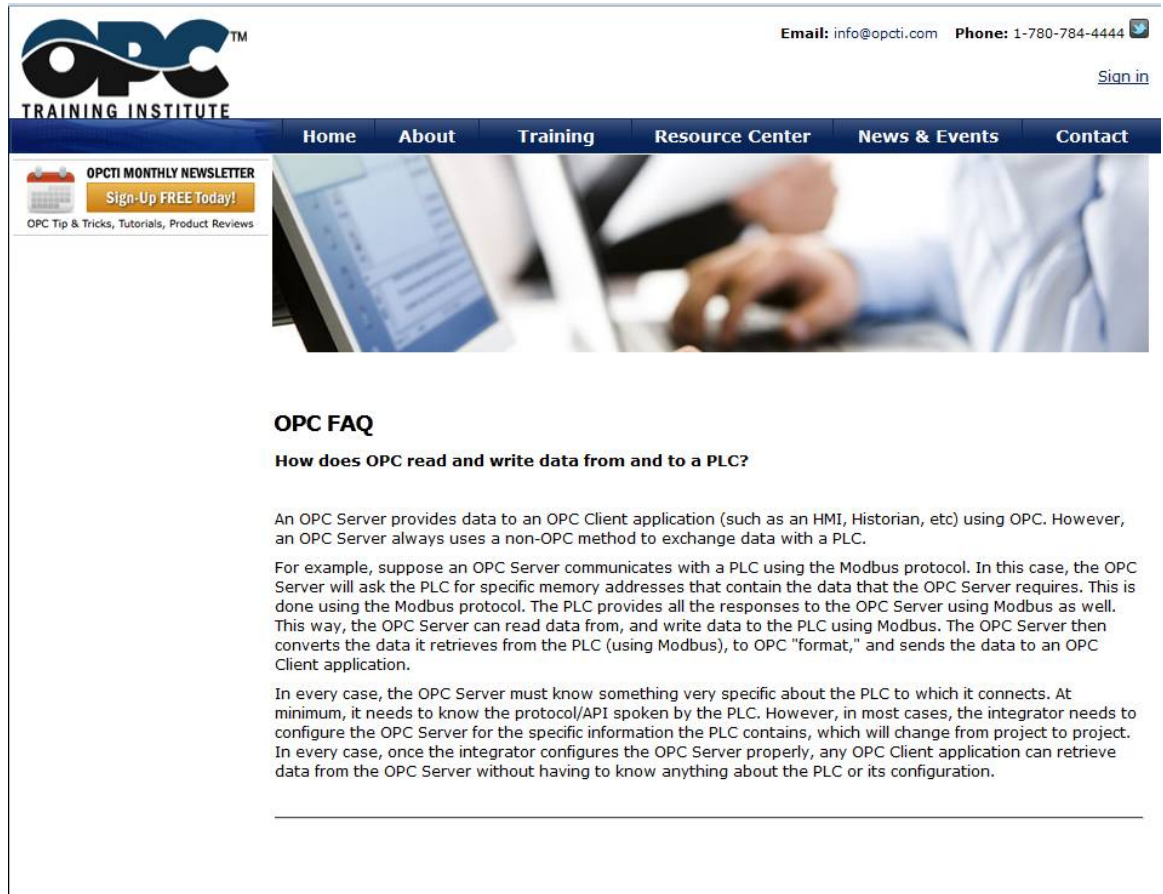
OPC: The Ins and Outs to what it's About, page 6. (annotated)

10

10.

A system as recited in claim 1, further comprising a command format template and a response format template associated with each driver function, wherein:

On information and belief, systems based on ABB's Industrial System 800xA command format template and a response format template associated with each driver function. In particular, because an OPC server reads and writes data to a motion control device in a method specific to the motion control device and presents the data to an OPC Client in OPC "format," on information and belief there exists within ABB OPC systems a command format template and a response format template associated with each driver function.



OPC Training Institute web site.

<p>10.1 the motion control component further comprises: means for generating command data strings for controlling the selected motion control device based on the command format template and the application program;</p>	<p>On information and belief, systems based on ABB's Industrial System 800xA the motion control component (discussed above) further comprises a means for generating command data strings for controlling the selected motion control device based on the command format template and the application program. Specifically, on information and belief, data is written to the motion control device and translated from OPC "format" to specific motion control device format and protocol and a means exists for generating command strings based upon the command format template and the application program (see above).</p>
---	--

<p>10.2 and means for parsing response data strings generated by the selected motion control device based on the response format template and the application program.</p>	<p>On information and belief, systems based on ABB's Industrial System 800xA the motion control component (discussed above) further comprises a means for parsing response data strings generated by the selected motion control device based on the response format template and the application program. Specifically, on information and belief, data is read from the motion control device and translated from specific motion control device format and protocol to OPC "format" and a means exists for parsing response data strings generated by the motion control device based on the response format template and the application program (see above).</p>
---	--