Exhibit C

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

| Claim.Element	Analysis
16	
16. A motion system, comprising:	
16.1 an application program	**The term "application program" has been construed to mean "a software program designed to handle specific tasks."**

On the content for element 16.1:

On information and belief, systems based on ABB's Industrial System 800xA comprise one or more application programs that use the Connectivity Server.
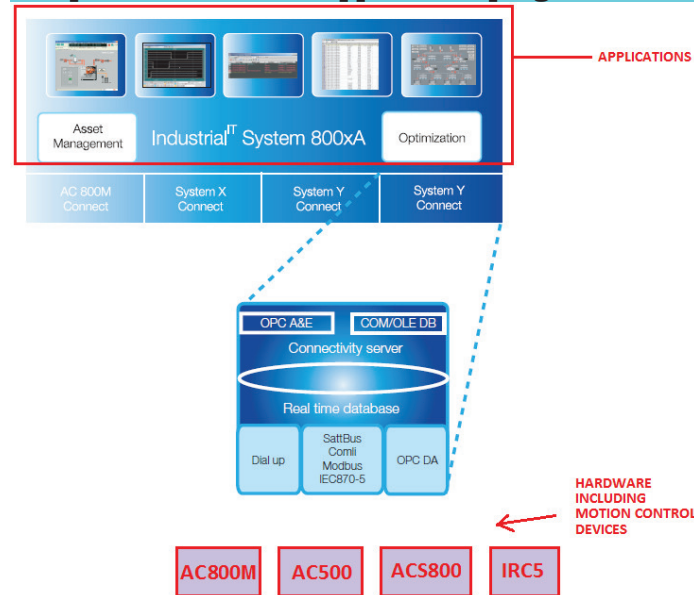
PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

There are various applications within the IndustrialIT System 800xA that use the Connectivity Server (PLC Connect is an exemplary Connectivity Server), such as the Aspect Server and Compact HMI 800.
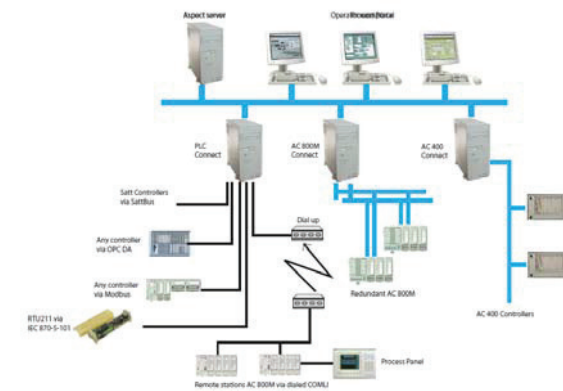
PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 1.

Compact HMI 800 is an example of one such application ...

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.
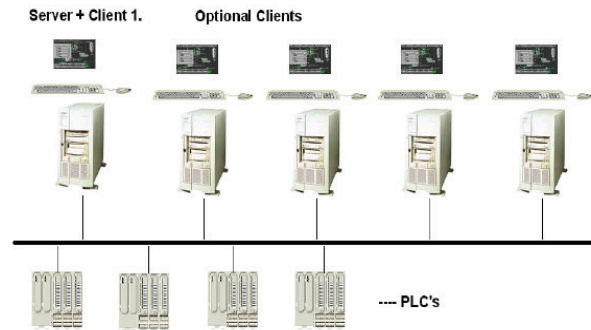

Figure 1. System Overview

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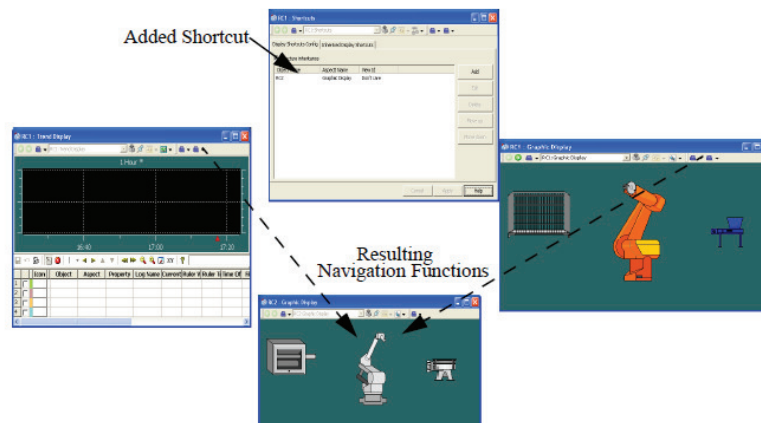
By adding a shortcut to the RC1 object, pointing to the RC2 Graphic Display, the RC2 Graphic Display can be accessed both from the RC1 Graphics Display and the RC1 Trend Display.

Industrial IT Compact HMI 800, System Version 4.1 – Getting Started, ABB, 2005, pp. 78.
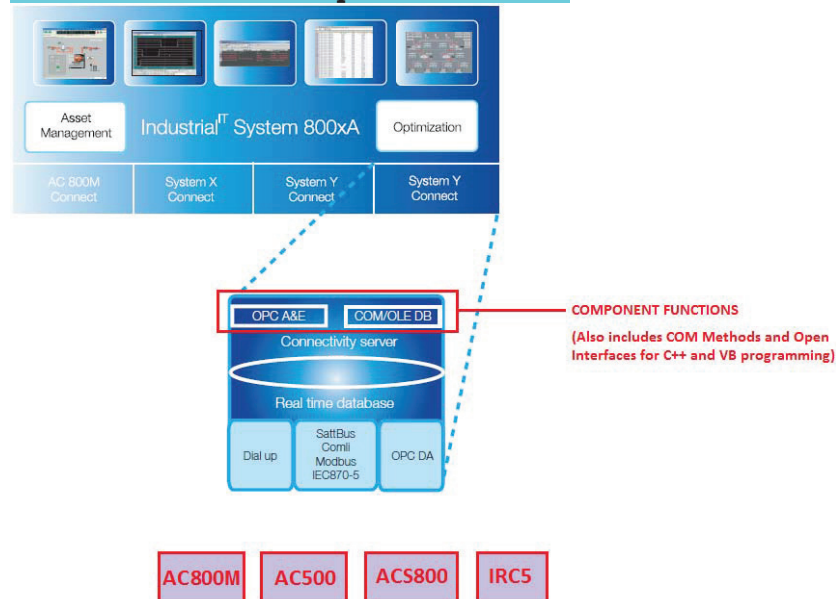
comprising at least one call to at least one component function;	**On information and belief, additional (but not limited to) application program(s) included in each of the products below provide similar functionality to that described for the products herein:**
	Product
	Panel 800
	Industrial IT System 800xA Operations
	System 800xA Smart Client
	System 800xA Engineering Tools
	System 800xA

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 the COM Interfaces (such as OPC interfaces), PLC Connect interfaces and other application programming interfaces. On further information and belief, the application programs comprise at least one call to at least one component function.

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 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 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.

ℹ️ 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.

```
⊟ ⊞ TankController, PLC Controller Type
    ⊟ ⬚ Tank1, TankType
           ⬚ Tank1 TankTemp, TemperatureType
           ⊙ Tank1 TankVolume, VolumeType
       ⊞ ⬚ TankPump, PumpType
       ⊞ ⬚ TankValve, ValveType
    ⊞ ⬚ Tank2, TankType
```
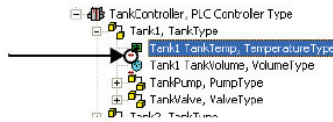
Figure 25. Icon Indicating a Not Connected Process Signal

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)

16.2 a plurality of motion control devices,	**Supported motion control devices in ABB's Industrial System 800xA systems include, but are not limited to, those based on ABB Drives, PLC's and Robot Controllers including ABB AC500 PLCs (presumably with ACS500), ABB ACS800 Multi Drives, and ABB IRC5 Robot Controllers**

ABB AC500 controller

Main catalogue

Automation products
AC500, AC31, CP400, WISA

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

ABB AC500 Motion Control Support

Automation products
Scalable PLC AC500
Motion control PS551-MC

The 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 interpolar
- 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

PLCopen
motion control

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

ABB's ACS800 controller

Low voltage AC drives

ABB industrial drives
ACS800, multidrives
1.1 to 5600 kW
Catalog

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

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 controllabi eliminate the need for an external motion controller, even in the most demanding applications, such as materials handlir 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's IRC5 Robot controller operation

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

ABB IRC5 Motion Control Support

Motion control

Based on advanced dynamic modeling, the IRC5 optimizes the performance of the robot for the physically shortest possible cycle time (QuickMove) and precise path accuracy (TrueMove). Together with a speed-independent path, predictable and high-performance behavior is delivered automatically, with no tuning required by the programmer.

"IRC5 – Industrial Robot Controller" data sheet, 2009, ABB, p. 1.

16.3	"Control command(s)" has been construed to mean "command codes in hardware language, which instruct a motion control device to perform motion control operations."
where a plurality of unique controller languages are associated with the plurality of motion control devices,	**On information and belief, a plurality of unique controller languages are associated with the plurality of motion control devices.**

The ACS800 controller shows support for various OPC tags that are associated with motion control operations.

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
1	Monitored in channel 1
2	Monitored in channel 2
3	Monitored in channel 3
4	Monitored in channel 4
5	Monitored in channel 5
6	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
1	01.03: FREQUENCY [Hz]	0	{0}{1}Par.1.3
2	01.04: CURRENT [A]	0	{0}{1}Par.1.4
3	01.07: DC BUS VOLTAGE V [V]	0	{0}{1}Par.1.7
4	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)

On information and belief, the following are associated with commands of a controller language.

Function Block Type	TC	SIL	PPA Face plate	CB Face plate	Description
DriveCommandSend	N	N	N	N	This function block type sends command data to the connected drive. It can be used as a base for control of ABB Drives ACS800, ACS600 and ACS400 and their corresponding DC drives. The function block can easily be used together with other function blocks to create more complex objects that handle functionality such as modes and HSI. See also the UniCore and BiCore function block descriptions.

ABB, IndustrialIT, 800xA – Control and I/O, Version 5.0 SP2A, Extended Control Software, p. 559.

Table 93. ProcessObjBasicLib function block types (Continued)

Function Block Type	TC	SIL	PPA Face plate	CB Face plate	Description
DriveStatusReceive	N	N	N	N	This function block type can be used as a base for control of ABB Drives ACS600 and ACS400, and their corresponding DC drives. The function block can easily be used together with other function blocks to create more complex objects that handle functionality such as modes and HSI. This function block provides the user with the ability to start and stop a drive with a chosen setpoint according to the local state matching in the drive.
Jog	Y	1-2	N	N	This function block handles the Jog functionality implemented in motor objects. Jog is a functionality that starts the motor object in a specified direction during a specified period of time. The Jog function only is applicable in manual mode of the motor object.

ABB, IndustrialIT, 800xA – Control and I/O, Version 5.0 SP2A, Extended Control Software, p. 560.

Table 95. ProcessObjExtLib function block types

Function Block Type	TC	SIL	PPA Face plate	CB Face plate	Description
Bi	N	1-2	Y	Y	Bi-directional object with alarm and graphics: This function block type is based on the BiCore object. Extensions include alarm handling and a faceplate.
LevelDetection	N	1-2	N	N	Supervises the level of an input signal to an object.
MotorBi	N	1-2	Y	Y	Motor bi-directional object with alarm and graphics: This function block type is based on the BiCore object. Extensions include alarm handling and a faceplate. This function block has additional interlocks, safety commands, and output delay timers.
MotorUni	N	1-2	Y	Y	Motor uni-directional object with alarm and graphics: This function block type is based on the UniCore object. Extensions include alarm handling and a faceplate. This function block has additional interlocks, safety commands, and output delay timers.

ABB, IndustrialIT, 800xA – Control and I/O, Version 5.0 SP2A, Extended Control Software, p. 564.

Table 96. ProcessObjExtLib control module types

Control Module Type	TC	SIL	PPA Face plate	CB Face plate	Description
BiM	N	1-2	Y	Y	Bi-directional object with alarm and graphics: This control module type is based on the BiCore alarm handling and a faceplate. See Table 44 on page 415. *This object is a member of the voting logic concept (a sending and receiving object). See also Signal and Vote Loop Concept on page 405.*
MotorBiM	N	1-2	Y	Y	Motor bi-directional object with alarm and graphics: This control module is based on the BiCore alarm handling and a faceplate. This control module has additional interlocks, safety commands, and output delay timers. See Table 44 on page 415. *This object is a member of the voting logic concept (a sending and receiving object). See also Signal and Vote Loop Concept on page 405.*
MotorUniM	N	1-2	Y	Y	Motor uni-directional object with alarm and graphics: This control module is based on the UniCore alarm handling and a faceplate. This control module has additional interlocks, safety commands, and output delay timers. See Table 44 on page 415. *This object is a member of the voting logic concept (a sending and receiving object). See also Signal and Vote Loop Concept on page 405.*

ABB, IndustrialIT, 800xA – Control and I/O, Version 5.0 SP2A, Extended Control Software, p. 566.

ABB IRC5 Robot controllers support the set of control commands making up the RAPID programming language

RAPID programming language

It provides the perfect combination of simplicity, flexibility and powerfulness. RAPID is a truly unlimited language with support for well-structured programs, shop floor language and advanced features. It also incorporates powerful support for many process applications.

"IRC5 – Industrial Robot Controller" data sheet, 2009, ABB, p. 1.

IRC5

Specification	
Control hardware:	Multi-processor system
	PCI bus
	Flash disk for mass memory
	Energy back-up power failure handling
	USB memory interface
Control software:	Well proven real-time OS
	High-level RAPID programming language
	PC-DOS file format
	Preloaded software, also available on DVD
	Extensive functionality set, see separate
	RobotWare data sheet

"IRC5 – Industrial Robot Controller" data sheet, 2009, ABB, p. 2. (annotated)

`16.3.i`
 	
each controller language comprises at least some control commands for processing information associated with motion control devices, and each of the motion control devices comprises a controller capable of generating electrical signals based on at least one control command of the controller language associated with the motion control device, and a mechanical system capable of causing a motion control operation based on electrical signals generated by the controller ;
	"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, each controller language comprises at least some control commands for processing information associated with motion control devices, and each of the motion control devices comprises a controller capable of generating electrical signals based on at least one control command of the controller language associated with the motion control device, and a mechanical system capable of causing a motion control operation based on electrical signals generated by the controller.
	
	You can change the width of a column by dragging the column separator in the title.
	
	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
	1 Monitored in channel 1
	2 Monitored in channel 2
	3 Monitored in channel 3
	4 Monitored in channel 4
	5 Monitored in channel 5
	6 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.
	
	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)
	
	On information and belief, each of the above described exemplary controllers is designed to operate a mechanical system capable of causing a motion control operation based on electrical signals generated by the controller.

Mobile

ABB   Power and productivity
for a better world™

| Home | About ABB | Products & services | News center | Careers | Investor relations |

Offerings A-Z   ABB Product Guide   Industries and utilities   Service Guide   Contact Directory

Product Guide > Robotics > RobotStudio Community > Developer Tools

Developer Tools

Developer Tools are SDKs for software developers, to enable development of PC software applications that communicate with IRC5 robot controllers or applications that run on the FlexPendant.

The section Knowledge and Content Sharing is where to discuss and share your knowledge with us and other users. Get or give help on specific coding issues on the User Forum . Upload and download complete sample applications or small code snippets at Content Sharing. We appreciate your contribution, as we believe that working together is the key to success. The usage of all Developer Tools is absolutely free.
Download, install and start exploring them today!

Develop FlexPendant applications

FlexPendant SDK Getting Started

Develop PC applications

PC SDK Getting Started

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.

OPC is a series of standards specification widely used in the automation industry. OPC is organized through the OPC Foundation and is vendor neutral. Learn all about OPC at: http://www.opcfoundation.org. A recent interview with Manny Mandrusiak, Vice President of Marketing for the OPC Foundation, is available at http://www.automatedbuildings.com/....

After installation ABB IRC5 OPC Server Help is available from Windows Start Menu at Programs\ABB Industrial IT\Robotics IT\ IRC5 OPC Server. See also ReadMe in the installation folder.

"ABB Web-Site: Product Guide > Robotics > RobotStudio Community > Developer Tools" (annotated)
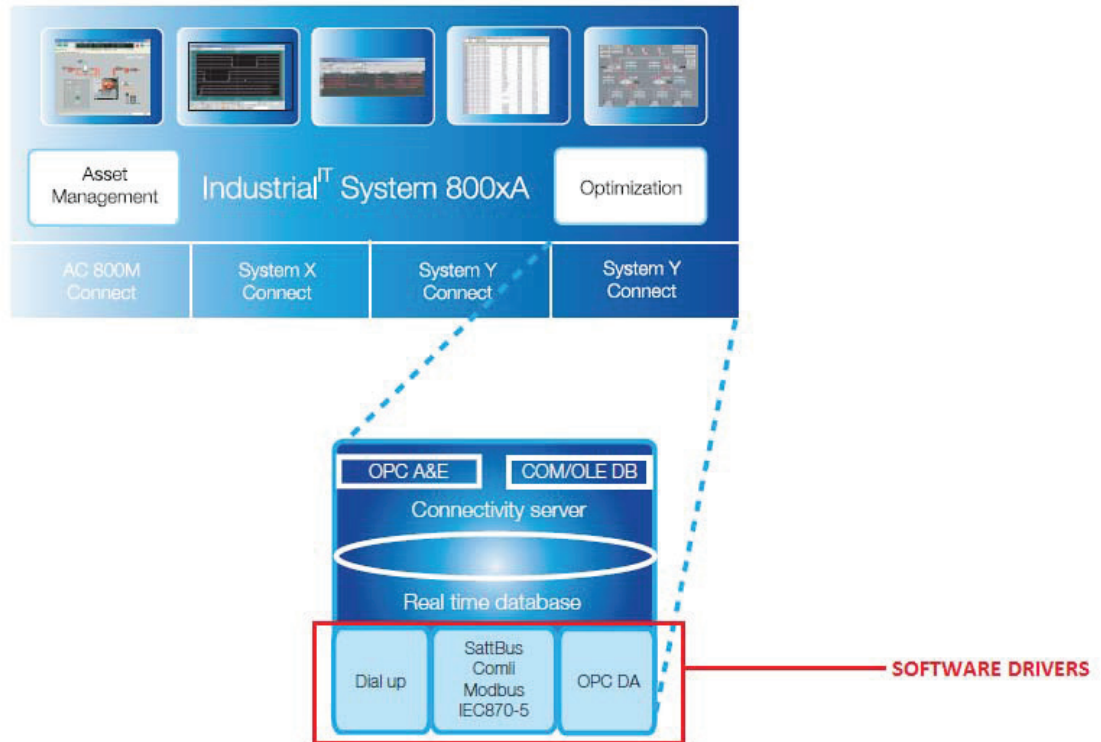
RAPID programming language

It provides the perfect combination of simplicity, flexibility and powerfulness. RAPID is a truly unlimited language with support for well-structured programs, shop floor language and advanced features. It also incorporates powerful support for many process applications.

"IRC5 – Industrial Robot Controller" data sheet, 2009, ABB, p. 1.

16.4 a set of software drivers each comprising driver code, where each software driver is associated with at least one of the plurality of controller languages,	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 comprising driver code, where each software driver is associated with at least one of the plurality of controller languages. For example, on information and belief PLC Connect comprises drivers that comprise software driver code, implement driver functions and are associated with one or of the controllers.
	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)
	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)

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."

The ACS800 controller is an example of a hardware driver that has OPC support and software that comprises software driver code.

ACS800 Controller

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.

```
⊞ P  File: ACN
   ⚠ ACS 600 0025_3SG {0}{1}
   ■ Device name {1}{1}
   ■ POSITIONING DRIVE {1}{10}
```

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)

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

See 16.3 above.

On information and belief ABB Drives, PLC's 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 software comprising software driver code that on information and belief associates the driver with the set of control commands associated with the AC500 controller.

Main catalogue
Automation products
AC500, AC31, CP400, WISA

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 AC800 has OPC support and software comprising software driver code that on information and belief associates the driver with the set of control commands associated with the AC800 controller.

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 IRC5 has OPC support and software comprising software driver code that on information and belief associates the driver with the set of control commands associated with the IRC5 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)

16.4.i	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."
and each software driver exposes a service provider interface defining a set of driver functions, where	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."

On information and belief, the software drivers in systems based upon ABB's Industrial System 800xA exposes a service provider interface defining a set of driver functions.
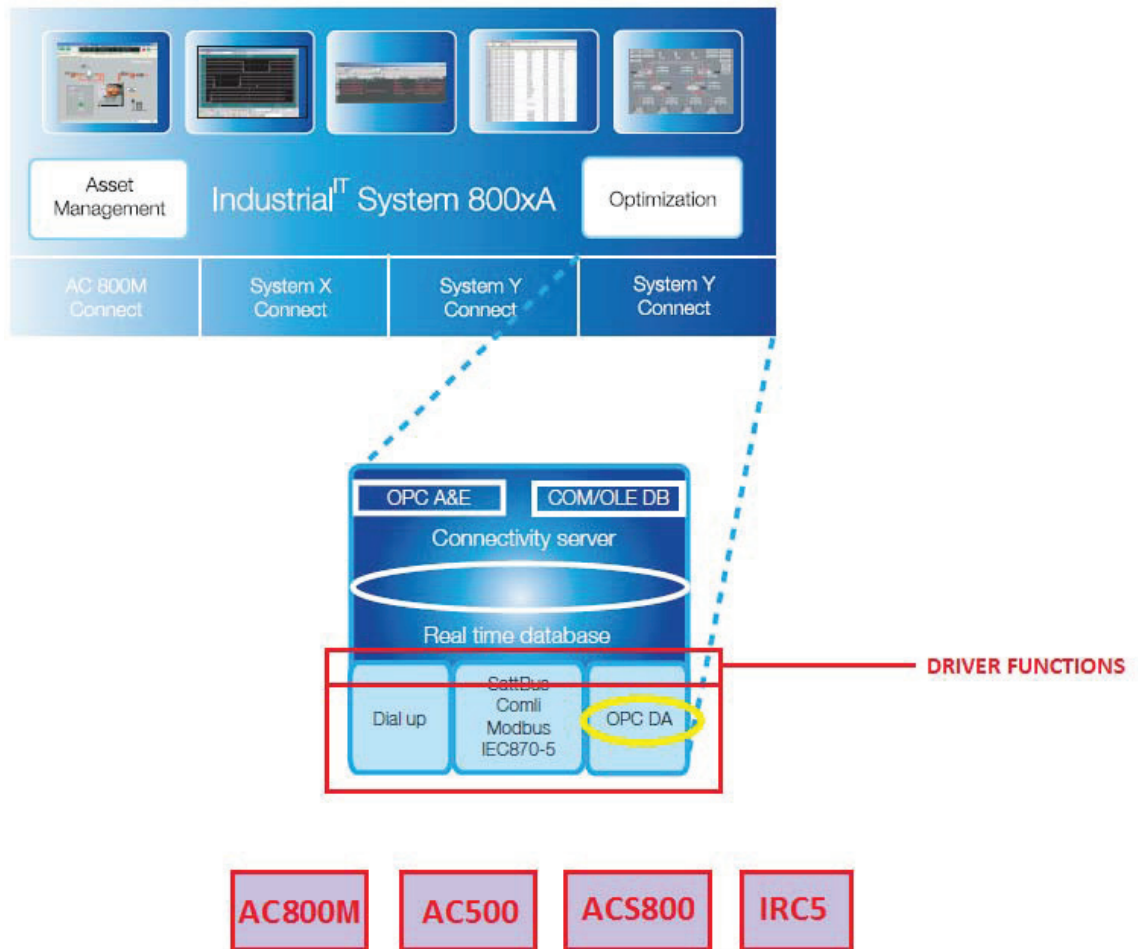
PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

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."

For example, the OPC DA specification defines driver functions that are associated with primitive or non-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.

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
1	Monitored in channel 1
2	Monitored in channel 2
3	Monitored in channel 3
4	Monitored in channel 4
5	Monitored in channel 5
6	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)

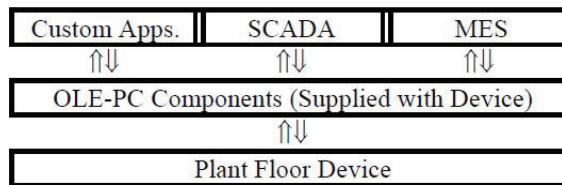|	

`16.4.i.a`
`the driver`
`functions`
`are`
`independent`
`of the`
`plurality of`
`controller`
`languages,`
	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."
	
	"Control command(s)" has been construed to mean "command codes in hardware language, which instruct a motion control device to perform motion control operations."
	
	By definition of the OPC standard itself, the OPC functions are independent of the plurality of control commands.
	
	"OLE for Process Control (OPC) draws a line between hardware providers and software developers. It provides a mechanism to provide data from a data source and communicate the data to any client application in a standard way. A vendor can now develop a reusable, highly optimized server to communicate to the data source, and maintain a mechanism to access data from the data source/device efficiently. Providing the server with an OPC interface allows **any** client to access their devices."
	OLE for Process Control Overview Version 1.0, 10/27/1998, OPC Foundation, p. 3.
	
	### 1.2 Purpose
	What is needed is a common way for applications to access data from any device on the plant floor, allowing compliant applications to seemlessly access data in a manufacturing environment.
	
	Custom Apps.　SCADA　MES
	OLE-PC Components (Supplied with Device)
	Plant Floor Device
	
	Figure 1-2. Manufacturing Environment
	OLE for Process Control Standard Version 1.0, 12/22/1995, OPC Foundation, p. 2. (annotated)

16.4.i.b	
at least one driver function is an extended driver function that is associated with a non-primitive motion operation that can be performed using at least one primitive motion operation, where the at least one primitive motion operation cannot be performed using a combination of primitive or non-primitive motion 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 "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."

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. 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)
- IOPCCallback::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)
&
- Group is a collection of items within DriveOPC. It is not the same as a parameter group in a drive.
IDataObject
- Items within a group can be active or inactive.
&
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::OnDataChange 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::OnDataChange 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.

In another example, ABB offers calculated softpoint that are 'calculated' using one or more other signals (either emulated as a or 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 a further 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.

#	Variable	Object	Property	Direction	Online Value	State	Offline Value	Event
1	INPUT1	[Functional Structure]G1	CCA:Value	Input		Offline	5.12	False
2	INPUT2	[Functional Structure]G2	CCA:Value	Input		Offline	5.5	False
3	INPUT3	[Functional Structure]G3	CCA:Value	Input		Offline	5.9987	False
4	INPUT4	[Functional Structure]G4	CCA:Value	Input		Offline	7.889	False
5	OUTPUT	[Functional Structure]G5	CCA:Value	Output		Offline	6.126925	False

```
Option Explicit

Dim Sum, Average

Sum = CDBL(INPUT1) + CDBL(INPUT2) + CDBL(INPUT3) + CDBL(INPUT4)

Average = Sum / 4

OUTPUT = Average
```

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.


`16.4.ii.c`
`at least one`
`driver`
`function is`
`a core`
`driver`
`function`
`that is`
`associated`
`with a`
`primitive`
`motion`
`operation,`
	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 "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."
	
	As stated previously, on information and belief, the OPC-compliant drivers in ABB's Industrial System 800xA expose a service provider interface comprising driver functions.
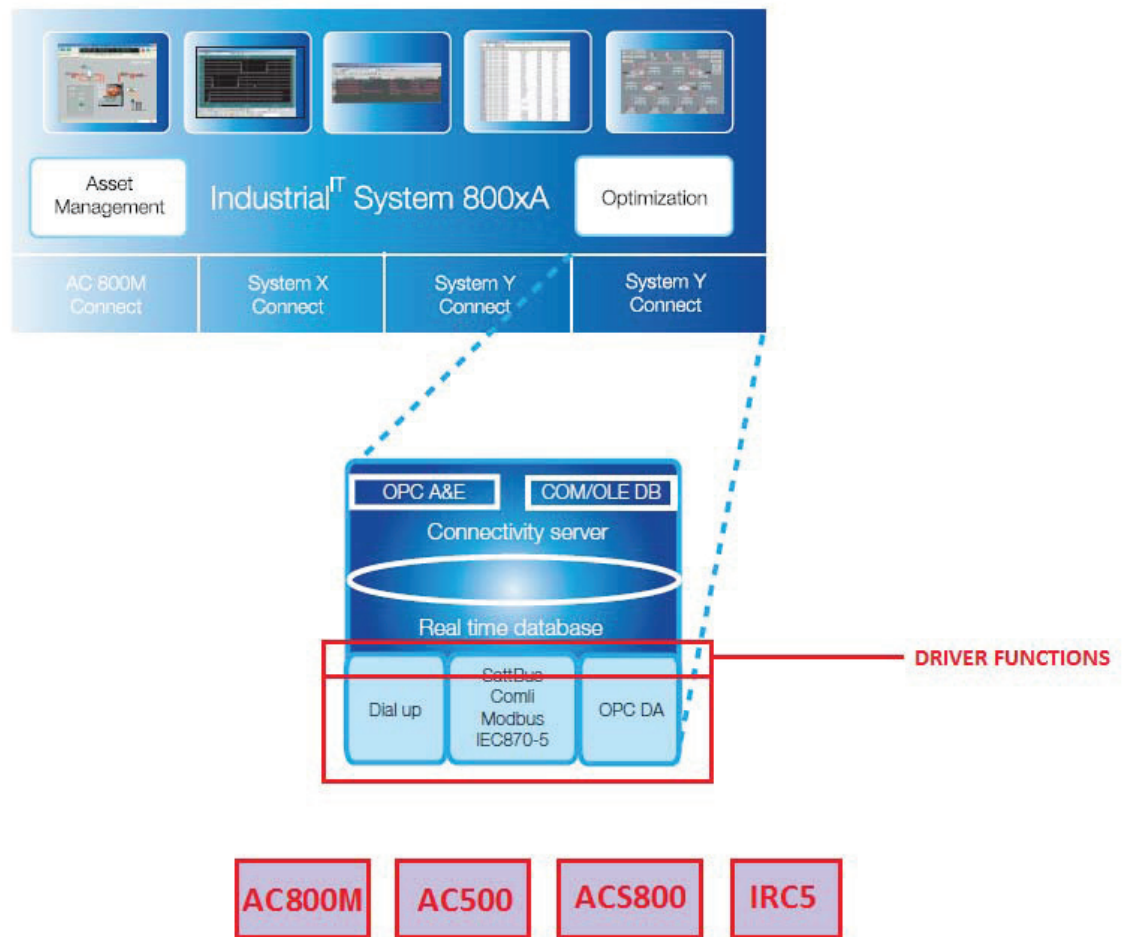	
	PLC Connect integrates ABB and third-party controllers with IndustrialIT System 800xA, 2004, ABB Automation Technologies, p. 2. (annotated)

For example, the OPC DA specification defines core driver 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, for example, OPC tags shown below, which are examples of motion control operations. For example, reading the speed of a motor is a motion control operation.

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
1	Monitored in channel 1
2	Monitored in channel 2
3	Monitored in channel 3
4	Monitored in channel 4
5	Monitored in channel 5
6	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
1 01.03: FREQUENCY [Hz]	0	{0}{1}Par.1.3
2 01.04: CURRENT [A]	0	{0}{1}Par.1.4
3 01.07: DC BUS VOLTAGE V [V]	0	{0}{1}Par.1.7
4 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)

16.4.i.d the driver code of at least one software driver associates at least one driver function with at least one control command of the at least one controller language associated with at least one of the software drivers, 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."

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."

On information and belief, in systems based upon ABB's Industrial System 800xA, the driver code of at least one software driver associates at least one driver function with at least one control command of the at least one controller language associated with at least one of the software drivers as shown below by way of example for the ACS800 Controller.

ACS800 Controller

ABB ACS800 has OPC Support and therefore has driver functions as shown in 16.4 above.

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)

"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, at least one driver function of the ACS800 driver is associated with at least one control command.

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
1	Monitored in channel 1
2	Monitored in channel 2
3	Monitored in channel 3
4	Monitored in channel 4
5	Monitored in channel 5
6	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
1	01.03: FREQUENCY [Hz]	0	(0)(1)Par.1.3
2	01.04: CURRENT [A]	0	(0)(1)Par.1.4
3	01.07: DC BUS VOLTAGE V [V]	0	(0)(1)Par.1.7
4	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)
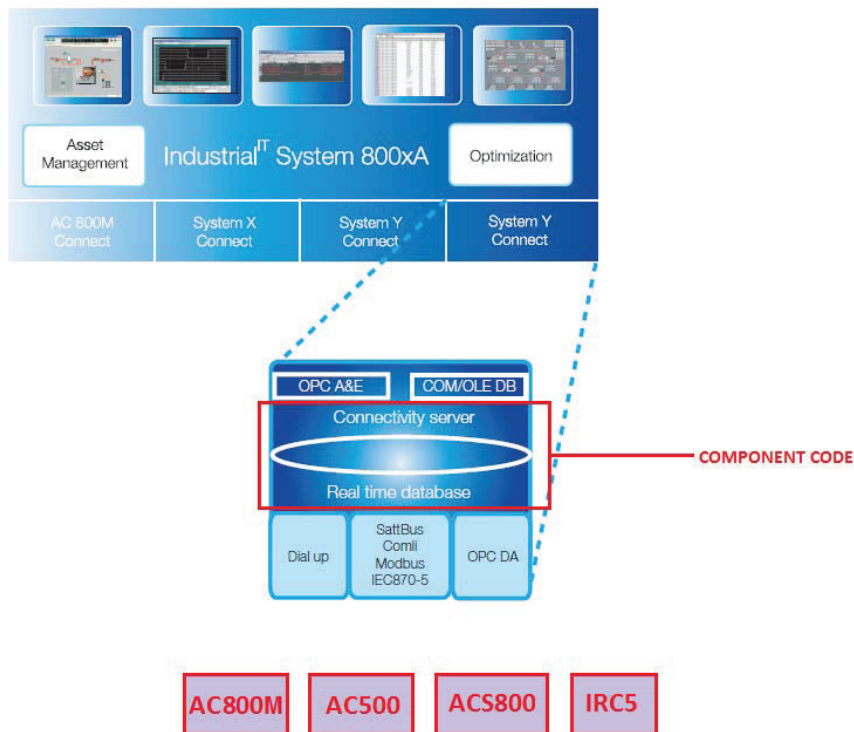
16.4.ii at least one selected software driver is associated with at least one selected motion control device;	For the reasons set forth in 16.4 above, on information and belief, at least one selected software driver is associated with at least one selected motion control device.

| 16.5 a motion component comprising component code, where | 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."

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." |
|---|---|

On information and belief, the Connectivity Server in ABB's Industrial System 800xA comprises a motion control component comprising component code. On further information and belief, examples of such component code includes the code in the software that associates the component functions with the driver functions such as the code in 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 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)

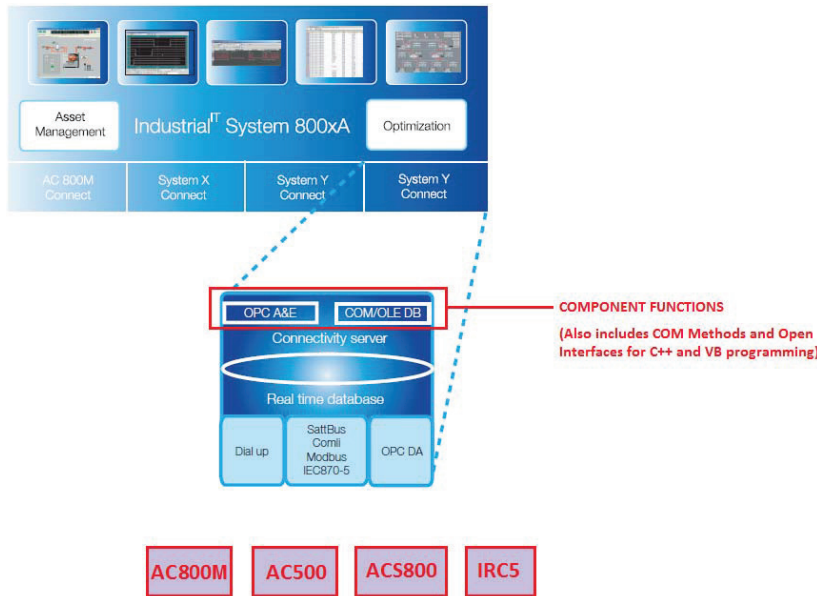PLC Connect provides the following features:

- 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)

| 16.5.i

the motion component exposes an application programming interface comprising a set of component functions, where | 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."

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, the Connectivity Server exposes one more application programming interfaces comprising a set of component functions. On further information and belief, COM Interfaces (such as OPC functions) are among several API's exposed. On further information and belief, the API's expose a set of component functions.

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 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 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)

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, Variable Access functions that are provided through the Variable Access Interface are examples of component functions.

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.

ℹ️ 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)

| 16.5.i.a each component function is implemented by component code, | 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." 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." See 16.5 and 16.5.i above. |

`16.5.i.b`
`at least the`
`component`
`code is`
`independent`
`of the`
`plurality of`
`controller`
`languages,`
`and`
	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."
	
	"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 the component code is independent of the plurality of controller languages.
	
	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 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)
	
	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)

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 and by example, calculations use a function table as shown below.

- 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.

#	Variable	Object	Property	Direction	Online Value	State	Offline Value	Event
1	INPUT1	[Functional Structure]G1	CCA:Value	Input		Offline	5.12	False
2	INPUT2	[Functional Structure]G2	CCA:Value	Input		Offline	5.5	False
3	INPUT3	[Functional Structure]G3	CCA:Value	Input		Offline	5.9987	False
4	INPUT4	[Functional Structure]G4	CCA:Value	Input		Offline	7.889	False
5	OUTPUT	[Functional Structure]G5	CCA:Value	Output		Offline	6.126925	False

Option Explicit

Dim Sum, Average

Sum = CDBL(INPUT1) + CDBL(INPUT2) + CDBL(INPUT3) + CDBL(INPUT4)

Average = Sum / 4

OUTPUT = Average

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.

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."

For example, on information and belief the calculation values are available via the component functions, such as the 'Open interface' provided by the Variable functions shown above.

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:

- 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)

16.5.ii the component code associates at least one of the component functions with at least one of the driver functions;	On information and belief the component code associates at least one of the component functions with at least one of the driver functions. See 16.5 and 16.5.i above.

16.6 wherein the at least one selected driver generates at least one control command in the controller language associated with the at least one selected motion control device based on the calls to component functions of the application program, the component code, and the driver code of the at least one selected software driver.	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." "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 "application program" has been construed to mean "a software program designed to handle specific tasks." 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 "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 at least one selected driver generates at least one control command in the controller language associated with the at least one selected motion control device based on the calls to component functions of the application program, the component code, and the driver code of the at least one selected software driver. For example, on information and belief the System Connect software (such as PLC Connect) comprises a selected driver that generates at least one control command from the set of control commands associated with the at least one selected controller based on the calls to the component functions of the application program, the component code, and the driver code. On information and belief the selected application program calls the Variable interface of PLC Connect and the selected driver transmits command codes to the associated controller to read and write data as well as set-up device based (if supported) event handlers based on control exerted by the motion control component. *See* 16.4 and 16.5 above. ABB Integrated PLC AC500 in IRC5, Slide 24. (annotated)
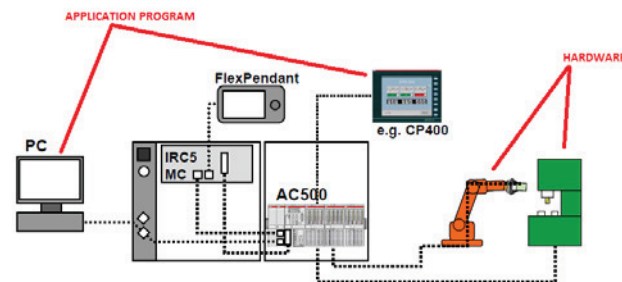
Claim.Element	Analysis
17	
17. A motion control system as recited in claim 16, in which: the component code associates at least one component function with a plurality of core driver functions; and the component code associates at least one component function with one of the core driver functions.	**On information and belief, the component code associates at least one component function with a plurality of core driver functions and the component code associates at least one component function with one of the core driver functions.** **Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

Claim.Element	Analysis
18	
18. A motion control system as recited in claim 16, in which: the component code implements at least one extended driver function by associating at least one component function with a plurality of core driver functions; and the component code implements at least one core driver function by associating at least one component function with one of the core driver functions.	**On information and belief, the component code that implements at least one extended driver function by associating at least one component function with a plurality of core driver functions; and the component code implements at least one core driver function by associating at least one component function with one of the core driver functions.** **Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

Claim.Element	Analysis
19	
19. A motion control system as recited in claim 16, in which: the component code emulates at least one extended driver function by associating at least one component function with a plurality of the core driver functions; the component code implements at least one extended driver function by associating at least one component function with one of the extended driver functions; and the component code implements at least one core driver function by associating at least one component function with one of the core driver functions.	**On information and belief, the component code emulates at least one extended driver function by associating at least one component function with a plurality of the core driver functions; the component code implements at least one extended driver function by associating at least one component function with one of the extended driver functions; and the component code implements at least one core driver function by associating at least one component function with one of the core driver functions.** **Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

Claim.Element	Analysis
20	
20. A motion control system as recited in claim 16, in which the application program further comprises at least one call to a component function comprising at least one parameter, where: the at least one parameter is associated with a data item associated with the at least one selected motion control device; and the at least one parameter is passed from the application program to the at least one selected software driver by the motion component.	On information and belief the application program further comprises at least one call to a component function comprising at least one parameter, where: the at least one parameter is associated with a data item associated with the at least one selected motion control device; and the at least one parameter is passed from the application program to the at least one selected software driver by the motion component. Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.

Claim.Element	Analysis
21	
21. A motion control system as recited in claim 16, in which the software drivers are binary modules.	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 the software drivers are implemented as binary modules. Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

Claim.Element	Analysis
22	
22. A motion control system as recited in claim 16, in which the software drivers conform to the service provider interface such that the software drivers may be interchanged without changing the component code.	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, the software drivers conform to the service provider interface such that they may be replaced without changing the component code. See also 16.2 above. Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

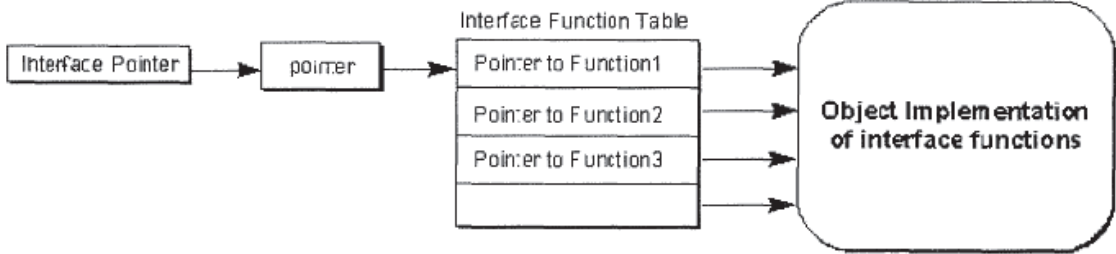| Claim.Element	Analysis
23	
23. A motion control system as recited in claim 16, further comprising a plurality of streams, where each stream contains transmit stream code that determines how control commands are transmitted to a destination of control commands.	**On information and belief, systems based on ABB's Industrial System 800xA comprise a plurality of streams where each stream contains transmit stream code that determines how control commands are transmitted to a destination of control commands.** PLC Connect integrates ABB and third-party controllers with Industrial IT System 800xA, 2004, ABB Automation Technologies, p. 1.

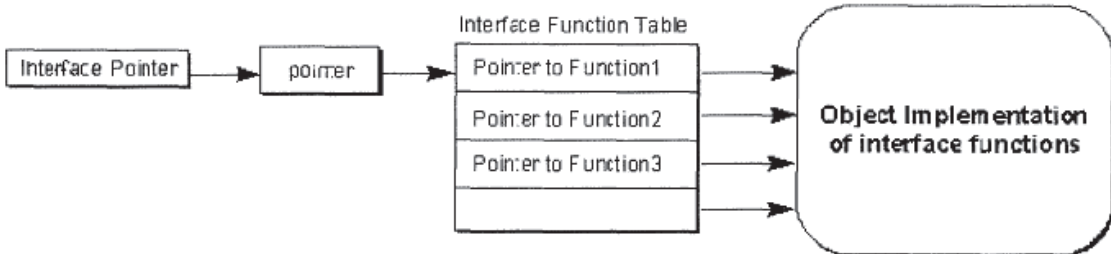| Claim.Element	Analysis
24	
24. A motion control system as recited in claim 23, in which the destination of the control commands is the at least one selected motion control device.	**On information and belief inspection the destination of the control commands is at least one selected motion control device.** *See* **16.1 and 16.2 above.**

Claim.Element	Analysis
25	
25. A motion control system as recited in claim 23, where in at least one stream is a stream binary module that can be interchanged with other stream binary modules without changing the software driver that uses the stream.	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, at least one stream is a stream binary module that can be interchanged with other stream binary modules without changing the software driver that uses the stream. Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.

Claim.Element	Analysis
26	
26. A motion control system as recited in claim 23, in which the destination of the control commands is a file.	**On information and belief inspection, at least one destination of the control commands is a file.** **Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

Claim.Element	Analysis
27	
27. A motion control system as recited in claim 16, in which: the motion component is a component binary module; the software drivers are driver binary modules; and the driver binary modules may be interchanged without changing the component binary module.	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 the motion control component is a binary module.** 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 the software drivers are driver binary modules and the driver binary modules may be interchanged without changing the component binary module.** **Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.**

Claim.Element	Analysis
28	
28. A motion control system as recited in claim 16, wherein the control commands of each of the plurality of controller languages comprise binary codes.	The term "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, the control commands of each of the plurality of controller languages comprise binary codes. Despite diligent efforts, we were unable to locate specific evidence of this feature in the public literature. Upon information and belief, we expect such evidence to be uncovered during source code review.

Claim.Element	Analysis
29	

29.
 A motion control system as recited in claim 16, in which the component code uses a function table that associates at least one of the component functions with at least one of the driver functions.
	On information and belief, the component code uses a function table that associates at least one of the component functions with at least one of the driver functions. Specifically, on information and belief, for example at least some of the component functions discussed in section 16.1 above are exposed by the COM interfaces referenced below.
	
	 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)
	
	 For example, COM Interfaces are implemented as function pointer tables.
	
	 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.
	
	 Figure 3. The binary interface structure
	
	 It is this exact interface structure that provides the ability to marshall one of these pointers between processes and machines, as described in the section titled "Local/Remote Transparency."
	
	 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.
	
	 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 *pSC*, I can call a member like this:
	
	 `pSC->LookUpWord(pszWord);`
	
	 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.
	
	 In C, the same call would look like this, with an explicit indirection through **lpVtbl** and passing the interface pointer as the first argument:
	
	 `pSC->lpVtbl->LookUpWord(pSC, pszWord);`
	
	 What OLE Is Really About, Kraig Brockschmidt, July 1996, pp. 17-18.

Claim.Element	Analysis
30	

30.
A motion control system as recited in claim 29, in which the function table is a function pointer table.
	On information and belief, the function table is a function pointer table.
	
	For example, the COM Interface Function Table is an example of such a function table.
	
	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.
	
	Figure 3. The binary interface structure
	
	It is this exact interface structure that provides the ability to marshall one of these pointers between processes and machines, as described in the section titled "Local/Remote Transparency."
	
	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.
	
	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 pSC, I can call a member like this:
	
	`pSC->LookUpWord(pszWord);`
	
	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.
	
	In C, the same call would look like this, with an explicit indirection through **lpVtbl** and passing the interface pointer as the first argument:
	
	`pSC->lpVtbl->LookUpWord(pSC, pszWord);`
	
	What OLE Is Really About, Kraig Brockschmidt, July 1996, pp. 17-18.