CitectSCADA
CitectHMI

ABCLX Driver Help

Citect Pty. Limited

3 Fitzsimons Lane
PO Box 174
Pymble NSW 2073
Australia

Telephone: 61 2 9496 7300
Fax: 61 2 9496 7399

July 2004 edition for CitectSCADA Version 6.0

Manual Revision Version 6.0.

Printed in Australia.

Contents

ABCLX driver

Introduction

Citect has developed the ABCLX driver to enable Ethernet-based communication between CitectSCADA and ControlLogix system controllers.

This Help file provides the information required to set up and configure communication with a ControlLogix system. This involves a four-step process.

1 "Installing the ABCLX driver pack"

2 "Preparing the ControlLogix system"

3 "Setting up communications"

4 "Configuring your project"

There is also additional information on "Advanced configuration and maintenance" issues and a "Troubleshooting" section to help resolve communication or configuration errors.

About ABCLX

CitectSCADA connects to the ControlLogix system PLCs using Ethernet/IP with the ABCLX driver. It can be installed in parallel with a functional ControlLogix system, allowing engineering and testing to be conducted without any impact on operation.

No additional hardware (other than an Ethernet module) is required for the ControlLogix system. This allows you to replace ControlView operator stations on a step-by-step basis.

Note: Prior to version 3, the ABCLX driver (then known as the ABLogix driver) was installed as a TCP/IP board driver. Since version 3, the driver has been installed as an ABCLX-type board driver.

If you are upgrading an existing project which used a version 2.x ABLogix driver, you will need to alter the board type in your existing project to ABCLX type. Additionally, the format used for the port definition in the Special Options field of the CitectSCADA Ports Form has changed. See "ABCLX recommended settings".

Warning! The ABCLX is a tag-based driver, which means you need to be aware of the way OIDs (object identifiers) are implemented to avoid potential mismatches occurring. This is a particular concern with network distributed systems when attempting activities such as editing variable databases or restoring a project on a different computer. It is highly recommended that you

review the topic "Tag-based driver considerations" to ensure you are aware of any potential problems.

Installing the ABCLX driver pack

The ABCLX Driver Pack should be installed on the computer that will act as the connection point to the ControlLogix system. This computer must be configured as a CitectSCADA I/O server, and must contain an Ethernet card. It should also be installed on any machines where the CitectSCADA project is compiled.

Any computer you install the driver pack on must have CitectSCADA installed. Although the ABCLX driver is known to function reliably on CitectSCADA Version 5.41 and above, it is recommended you use Version 6.0.

Note: You need to close CitectSCADA Runtime and Citect Explorer before installing a driver pack. You will also require administrator permissions for the I/O server PC.

To install the ABCLX driver pack

1. Save the ABCLX Driver Pack to an appropriate location on the I/O Server PC.
2. Double click the EXE file to launch the installation.
3. Follow the instructions provided by the installation Wizard.

   Note: You will be prompted to select the CitectSCADA installation folder. By default, this is "C:\Program Files\Citect\CitectSCADA". If you installed CitectSCADA in a different folder, you should browse to the appropriate location. An error message will warn you if you select the wrong folder.
4. Click **Finish** to complete the installation.

Preparing the ControlLogix system

This section of the Help explains how to prepare your ControlLogix devices for communication with CitectSCADA. It covers:

- "Hardware requirements"
- "Software requirements"

Hardware requirements

Typically, the ControlLogix CPU module you want to connect to will be mounted on the same PLC backplane as Ethernet module your I/O Server communicates with. If this is not the case, you will have to set up a route path to indicate the location of the CPU you want to use.

See Also "Using a route path to locate a remote CPU"

Software requirements

You will need to have access to Rockwell's RSLogix 5000 Enterprise Series controller configuration software (obtainable from your ControlLogix system supplier) and its associated tools. This software provides access to controller configuration, project code, tag definitions and data values. Use it to:

1. Configure each Ethernet Module IP address
2. Create tags for each controller
3. Download the projects to the controllers.

Hint: When it comes time to configure CitectSCADA to communicate with ControlLogix system PLCs, you can use RSLogix 5000 to export PLC tag definitions to comma separated value (CSV) files, and use Windows Clipboard to copy them directly into the Citect Project Editor.

Note: CitectSCADA tags can only support arrays of 256 bytes. If your ControlLogix system has arrays that are greater than this limitation, you may have to break them up into segments using the offset and array size in the tag address. See "Mapping to more than an individual element of an array" for more information.

Setting up communications

Before configuring your CitectSCADA project, it is recommended that you first establish and confirm communication between CitectSCADA and the devices in your ControlLogix system. This allows you to test the communication path in isolation, and ensures that CitectSCADA can bring your devices online.

The best way to achieve this is by "Creating a test project". If you want to communicate with the a Logix 5000 PLC using an exclusive serial connection, see "Setting up an Exclusive Connection".

Note: The ABCLX will not perform on a very slow network, or a network that has an inconsistent connection to an I/O device in the field (for example, a wireless or dial-up connection). This is due to the fact the driver downloads all the tags from the PLC and subscribes them before it comes online. Attempting to do this over a slow network can take a long time and will cause the driver to time out on startup.

Creating a test project

CitectSCADA will not communicate with an I/O device if there is no reason to read or write data. Therefore, you will need to define and implement some variable tags within a CitectSCADA project to initiate a read request and verify that the communication channel is functioning correctly. This requires the creation of a test project.

The main goal with a test project is to keep things as simple as possible, so that any communication errors can be easily identified. For example, your test

project could do nothing more than use an integer variable to display a number on a graphics page.

Once you have used your test project to confirm communications, it can be set aside for further testing at a later date.

Note: You should build your test project on the CitectSCADA I/O server that will connect to the ControlLogix system.

To create a communications test project:

1. In **Citect Explorer**, create a new project and give it an appropriate name.
2. Add the device you would like to communicate with to the project. Typically, the information required about a device can be gathered by running the **Express Communications Wizard**, which implements a set of recommended settings for the type of device you select.

   To launch the Wizard, go to the **Citect Project Editor** and select **Express Wizard** from the Communications menu.
3. Progress through the Wizard, accepting all defaults for an I/O server and an I/O device. This will create an I/O server named "IOServer", and an I/O Device named "IODev".

   Note: If you require additional information about a particular step, use the **Help** button.
4. When you reach the **I/O Device Selection** page, locate the device you would like to communicate with; for example, Allen-Bradley | Controllogix5550 | Ethernet (TCP/IP).
5. Continue through the Wizard, until your reach the last page. Click **Finish** to implement the Wizard's settings.
6. Run the **Computer Setup Wizard**, accepting the defaults to make the computer a standalone CitectSCADA system configured to run your test project.
7. Go to the Citect Project Editor and open the **Ports** form from the **Comunications** menu. In the **Special Options** field, type in the IP address of the Ethernet module on the ControlLogix PLC, appended with a forward slash " / " and the slot number (zero-based) of the backplance containing the controller CPU module. The format should be:

   ```
   x.x.x.x/n
   ```

   for example:

   ```
   192.168.0.1/1
   ```

   If the CPU is on slot zero of the backplane, you do not need to include "/0".

8 Compile the project to confirm that there are no errors. If you receive any error messages, go back over the procedure to confirm your settings. See the "Troubleshooting" section to help resolve any persistent errors.

See Also "Setting up an Exclusive Connection"

Setting up an Exclusive Connection

CitectSCADA is able to communicate with the serial port on a Logix 5000 series processor via the 1761-NET_ENI module. This is referred to as an "exclusive connection".

As with any method of communication to a device, you should create a test project for an exclusive connection so that any communication errors can be easily identified.

To setup a serial communications test project:

1 Configure the 1761-NET_ENI module. Refer to MicroLogix Ethernet Interface (Cat. No. 1761-NET-ENI) User Manual for details on how to configure the module.

2 In Citect Explorer, create a new project and give it an appropriate name.

3 Add the device you would like to communicate with to the project. Typically, the information required about a device can be gathered by running the Express Communications Wizard, which implements a set of recommended settings for the type of device you select.

To launch the Wizard, go to the Citect Project Editor and select Express Wizard from the Communications menu.

4 Progress through the Wizard, accepting all defaults for an I/O server and an I/O device. This will create an I/O server named "IOServer", and an I/O Device named "IODev".

Note: If you require additional information about a particular step, use the Help button.

5 When you reach the I/O Device Selection page, locate the device you would like to communicate with; for example, Allen-Bradley | Controllogix5550 | Ethernet (TCP/IP).

6 Continue through the Wizard, until your reach the last page. Click Finish to implement the Wizard's settings.

7 Open the Ports Form from the Communications menu in Citect Editor. Type in the IP address of 1761-NET_ENI module in Special Opt field.

8 Set up ExclusiveConnection=1 and Route Path in Citect.ini :

```
[ABCLX]
ExclusiveConnection=1
IOServerName.IOPortName.RoutePath=/P:3/A:1
```

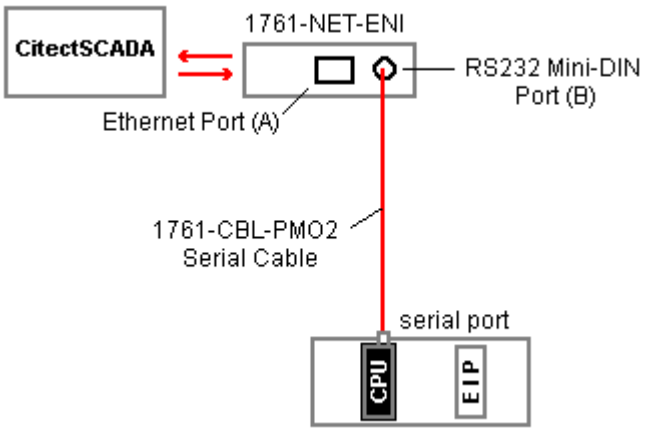
where:

/P:3 indicates the communication goes through the serial port (Port B) on 1761-NET-ENI module.

/A:1 indicates the module's unique address of serial port on cpu is 1.

9 Run the Computer Setup Wizard, accepting the defaults to make the computer a standalone CitectSCADA system configured to run your test project.

10 Compile the project to confirm that there are no errors. If you receive any error messages, go back over the procedure to confirm your settings. See the "Troubleshooting" section to help resolve any persistent errors.



Configuring your project

Once you have confirmed that communications between your CitectSCADA system and any connected devices are stable, you can start to build your project. This process includes:

- "Adding devices to your CitectSCADA project",
- "Adding tags to your CitectSCADA project",
- Consideration of the "Data types" used,
- The use of "Individual tag addressing" and/or "Array tag addressing",
- "Configuring Quality and Timestamp tags".

Adding devices to your CitectSCADA project

Adding a device to your CitectSCADA project provides information about the device's location, its communication method, the port it is connected to, and so on. This information is stored in the CitectSCADA project database.

There are two methods available for adding a device:

- "Adding a device using the Express Communications Wizard"
- "Manually configuring a device connection"

In most cases, completing the Express Communications Wizard is sufficient to set up your communications, as the settings it implements for each device have been pre-configured for preferred operation. If, however, your device connection needs are complex, you can reconfigure communications by manually configuring the CitectSCADA communication forms.

Adding a device using the Express Communications Wizard

1. To launch the Wizard, go to the **Citect Project Editor** and select **Express Wizard** from the **Communications** menu.
2. Progress through the Wizard, filling in the appropriate information for the I/O server and I/O device.

   Note: If you require additional information about a particular step, use the **Help** button.
3. When you reach the **I/O Device Selection** page, locate the device you would like to communicate with; for example, Allen-Bradley | Controllogix5550 | Ethernet (TCP/IP).
4. Continue through the Wizard, until your reach the last page. Click **Finish** to implement the Wizard's settings.

Manually configuring a device connection

In most cases, the Express Communication Wizard will be sufficient to set up communication with a device. However, if the configuration of a device entails unusual circumstances, you can manually input information into the required forms in Project Editor.

Note: You should consider the "ABCLX recommended settings" when manually configuring a connection to a ControlLogix device.

To manually configure CitectSCADA communications:

1. Define an I/O server using the **I/O Server** form. This sets the name of the CitectSCADA server that the I/O device will communicate with.
2. Complete the **Boards** form. This form defines which board (on your CitectSCADA computer) to use for communication, such as the mother board, network card, serial board or a PLC communication card.
3. Complete the **Ports** form**.** Often boards have multiple communication ports, and you must specify the port to use. Some modern equipment can have a number of logical (virtual) ports assigned to the one physical port.

4 Complete the **I/O Devices** form. This is where you define which I/O device CitectSCADA is talking to, by specifying the address. The protocol is also defined at this level.

5 Run the **Computer Setup Wizard** to complete the configuration. This allows you to define your CitectSCADA computer as the I/O server defined above. This is usually done after compilation of the project.

ABCLX recommended settings

The following tables show the recommended settings for communication with a ControlLogix device. These are the settings implemented by the Express Communications Wizard on the Boards form, Ports form and Devices form.

Boards form The Boards form lists all boards used in the CitectSCADA project. Each board record defines a separate board within the project.

Field	Value
Board Name	A unique name (up to 16 alphanumeric character) per server for the computer board being used to connect with the device. This is used by CitectSCADA in the Ports form.
Board Type	The type of board. Select ABCLX from the drop-down list (see Note below).
Address	This field must be "0" (zero).
I/O Port	Leave this field blank.
Interrupt	Leave this field blank.
Special Options	Leave this field blank.
Comment	Any useful comment. This is viewed in the Boards records database in Citect Project Editor.

Note: Prior to version 3, the ABCLX driver (then known as the ABLogix driver) was installed as a TCP/IP board driver. Since version 3, the driver has been installed as an ABCLX-type board driver. If you are upgrading an existing project which used a version 2.x ABLogix driver, you will need to alter the board type in your existing project to ABCLX type.

Ports Form The Ports form lists all ports used in the CitectSCADA project. Each port record defines a separate port within the project.

Field	Value
Port Name	A unique name (up to 16 alphanumeric character) for the computer port being used to connect with the device. This is used by CitectSCADA in the Devices form.
Port Number	An integer value, unique to the board as defined in the Board Name field. The Ethernet port needs no identification; however, the CitectSCADA communication database requires a value in this field.
Board Name	The name of the board this port is attached to as defined on the Boards form.
Baud Rate	Leave this field blank.

Field	Value
Data Bits	Leave this field blank.
Stop Bits	Leave this field blank.
Parity	Leave this field blank.
Special Options	The IP address of the Ethernet module on the ControlLogix system PLC, appended with a forward slash and the slot number (zero based) of the backplane containing the controller CPU module, in the format: x.x.x.x[/n]. For example: "192.168.0.1/0" (See Note below.)
Comment	Any useful comment. This is viewed in the Boards records database in Citect Project Editor.

Note: With the release of Version 3 of the ABCLX driver (formerly known as the ABLogix driver), the format used for the port definition in the Special Options field of the Ports Form has changed. If you are upgrading from Version 2.x (ABLogix), you will have to adjust the content of this field.

Devices form

Field	Value
Name	A unique name (up to 16 alphanumeric character) for the I/O Device being identified. Each I/O Device must have a unique name in the CitectSCADA system.
Number	A unique number for the I/O Device (0–16383). Each I/O Device must have a unique number in the CitectSCADA system, (unless redundancy of the I/O Device is being used—See Device Note 1).
Address	Leave this field blank. Note: See "Session Control" for additional configuration information if you are using the Session Disable feature.
Protocol	This field must be "ABCLX". Default is "ABCLX" when the CitectSCADA Express I/O Device Wizard is used to complete these forms.
Port Name	This field must contain one of the names previously defined in the Port Names field of the Ports form. There must be only one I/O device per port.
Comment	Any useful comment. This is viewed in the Devices records database in CitectSCADA Project Editor.

Adding tags to your CitectSCADA project

A separate variable tag must be added to your CitectSCADA project for each data-point (memory register) on the I/O devices you want to communicate with.

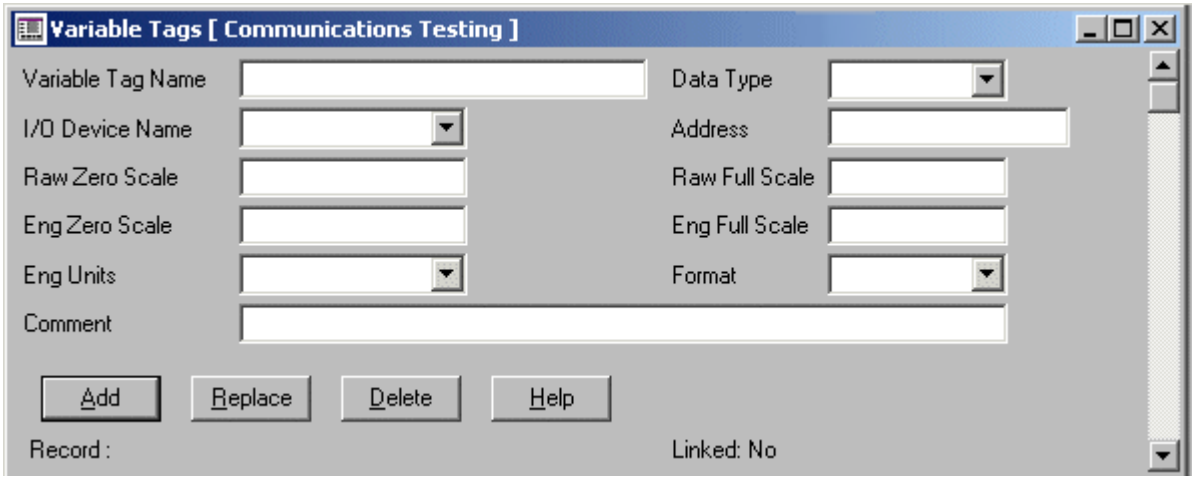
ControlLogix system PLC controllers use a tag-based addressing system. The tags associated with a controller are each assigned a unique name using the RSLogix 5000 Enterprise Series controller configuration software. See "Preparing the ControlLogix system".

You use the ControlLogix system PLC tag name when you specify the address for a tag in your CitectSCADA project. However, any square braces "[ ]" should

be replaced with "{ }", and forward slashes "/" should be used to access bits within words, rather than a period ".".

Note: The ABCLX driver supports "Individual tag addressing" and "Array tag addressing". Program tag support is also available for both these types of addressing. Local tag adressing is only supported via aliasing through the RSLogix 5000 software.

Your CitectSCADA variable tags are defined using the **Variable Tags** form:



For CitectSCADA to compile correctly, you must include appropriate values for the following fields:.

Field	Value
Variable Tag Name	A unique name per project for the variable tag (up to 79 alphanumeric characters in Version 6, 32 characters for earlier versions). This can be used by CitectSCADA graphic pages and in Cicode where defined.
Data Type	The type of variable. Select from the drop-down menu. It must be one of the predefined types used in CitectSCADA.
I/O Device Name	The name of an I/O Device defined in the I/O Devices form which is used to identify the ControlLogix PLC controller (CPU module).
Address	The name of a predefined ControlLogix system PLC tag, either individually addressed, array addressed or addressed as part of a program.

Note: CitectSCADA does not indicate any compile errors if an invalid address is entered in the Variable Tag form. Any variable tag values with an invalid address will display as "#COM" at run time on the graphic pages where they have been positioned.

Data types

When configuring your variable tags to communicate with the ControlLogix system, you must select a CitectSCADA data type that is compatible with the data type used by the ControlLogix system controller.

Ideally, you should match the data type of the CitectSCADA variable tag with the data type of the mapped ControlLogix system PLC tag as per the following table:

ControlLogix Data Description	Citect Address Format	Access rights	CitectSCADA Data Type
BOOL	<TagName>	R/W	DIGITAL
SINT	<TagName>	R/W	BYTE
INT	<TagName>	R/W	INTEGER
DINT	<TagName>	R/W	LONG
REAL	<TagName>	R/W	REAL
STRING	<TagName>	R/W	STRING

For example, if the ControlLogix tag you are addressing uses a “float” data type, you should select “real” in the Data Type field of the Variable Tags form.

Note: CitectSCADA reads a string data type from the ControlLogix PLC as a null terminated string. Therefore, if you need to read a 12 character string, you will have to configure the string length on the PLC to at least 13 characters to allow for the null character.

Data Coercion

The ABCLX driver also supports a number of data conversions from a Citect data type to a native PLC data type. However, as it is possible to coerce from a large data type to a smaller one in both directions, care must be taken to ensure that the range of the smaller type is always respected, otherwise the driver will return BAD_DATA_TYPE back to Citect. It is recommended you only rely on these conversions where circumstances demand it.

The following matrix table details the conversions supported by the driver. The plus sign (+) represents those conversions that are impacted by range differences.

	CITECT DATA TYPES										
PLC DATA TYPES	DIGITAL	BYTE	INTEGER	LONG	REAL	DIGITAL ARRAY	BYTE ARRAY	INTEGER ARRAY	LONG ARRAY	REAL ARRAY	STRING
BOOLEAN	YES										
SINT		YES	+YES	+YES	+YES						
INT		+YES	YES	+YES	+YES						
DINT		+YES	+YES	YES	+YES						
REAL		+YES	+YES	+YES	YES						
BOOLEAN ARRAY						YES					
SINT ARRAY						+YES	YES	+YES	+YES	+YES	
INT ARRAY						+YES	+YES	YES	+YES	+YES	
DINT ARRAY						+YES	+YES	+YES	YES	+YES	
FLOAT ARRAY							+YES	+YES	+YES	YES	
STRING											YES

Individual tag addressing

When declaring a CitectSCADA variable tag using individual tag addressing, type in the ControlLogix tag name, as defined on the ControlLogix PLC, into the **Address** field of the Variable Tag form.

Examples

Single Byte within a structure

PLC Tag Address	SingleStructure.SingleByte
PLC Tag Type	SINT
Citect Tag Name	SingleStructure_SingleByte
Citect Tag Address	SingleStructure.SingleByte
Citect Tag Type	BYTE

Single Digital

PLC Tag Address	SingleBoolean
PLC Tag Type	BOOLEAN
Citect Tag Name	SingleBoolean
Citect Tag Address	SingleBoolean
Citect Tag Type	DIGITAL

Single 82nd bit from a digital array

PLC Tag Address	DigitalArray
PLC Tag Type	BOOL[128]
Citect Tag Name	SingleDigital_81
Citect Tag Address	SingleDigital{81}
Citect Tag Type	DIGITAL

Single 7th bit from fourth item in a long array

PLC Tag Address	LongArray
PLC Tag Type	DINT[256]
Citect Tag Name	SingleBit_3_7
Citect Tag Address	LongArray{3}/7
Citect Tag Type	DIGITAL

Using bit access notation vs aliasing booleans

You can gain dramatic performance benefits by using ABCLX bit access syntax instead of aliasing bits in the PLC. For more information on why, see article **Q4064** the CitectSCADA Knowledge Base. The Knowledge Base can be installed from the CitectSCADA CD, or downloaded from the Citect web site (www.citect.com).

See Also "Array tag addressing"
"Addressing program tags"

Array tag addressing

Arrays can exist in the ControlLogix system PLC and/or the CitectSCADA tag database.

Within the ControlLogix system PLC, they are defined using the RSLogix 5000 Enterprise Series configuration software. See "Preparing the ControlLogix system".

CitectSCADA accesses the tag arrays within the ControlLogix system PLCs via the **Address** field of the Variable Tags form for each tag or tag array. Equally, the way to define CitectSCADA tag arrays is also to use the Address field of the CitectSCADA Variable Tags form for each tag array required in CitectSCADA.

Note: CitectSCADA tag arrays are simple, not complex, arrays. This means they can only contain elements of the same data type; for example, a digital tag array in CitectSCADA will only include digital type variables.

Tag arrays can be extremely flexible and quite complicated. It is possible to declare a tag array in CitectSCADA which maps to a tag array in a ControlLogix system PLC, or which maps to a contiguous sequence of tags (of the same data type) which may or may not be defined as a tag array in the ControlLogix system PLC.

Mapping to individual elements of an array

To map to an individual tag in a ControlLogix PLC tag array, use { } braces to define the array element as a zero based index. Use the following syntax:

```
<TagName>{<IndexNumber>}
```

Where:

<TagName> =	the ControlLogix tag name as defined on the ControlLogix system PLC;
{ } =	the required braces to define the array element index;
<IndexNumber> =	the zero-based index number of the array element.

Note: With versions of CitectSCADA prior to version 6, curly brackets { } were required when addressing arrays. This is no longer the case, as Version 6 supports the use of both curly brackets and square brackets [ ].

Individual tag address type mapping

You should match the data type of the CitectSCADA variable tag with the data type of the mapped ControlLogix system PLC tag as per the following table:

ControlLogix Data Description	Citect Address Format	Access rights	CitectSCADA Data Type
BOOL	<TagName>{<IndexNumber>}	R/W	DIGITAL
SINT	<TagName>{<IndexNumber>}	R/W	BYTE
INT	<TagName>{<IndexNumber>}	R/W	INTEGER
DINT	<TagName>{<IndexNumber>}	R/W	LONG
REAL	<TagName>{<IndexNumber>}	R/W	REAL
String structure (see Note below) or STRING (supplied by RSLinx)	<TagName>{<IndexNumber>}	R/W	STRING

Note: The string structure must follow the format:

```
{
DINT LEN
SINT[ ] DATA
}
```

Example **Single integer within an array of integers**

PLC Tag Address	IntegerArray[5]
PLC Tag Type	INT
Citect Tag Name	IntegerArray_5
Citect Tag Address	IntegerArray{5}
Citect Tag Type	INTEGER

Single float within an array of structures

PLC Tag Address	StructureArray[5].SingleFloat
PLC Tag Type	FLOAT
Citect Tag Name	StructureArray_5_SingleFloat
Citect Tag Address	StructureArray{5}.SingleFloat
Citect Tag Type	REAL

Mapping to more than an individual element of an array

CitectSCADA can only handle simple (same-type) arrays of up to 256 bytes in length, so when declaring variable arrays to access ControlLogix system PLC arrays (or data structures larger than 256 bytes), you will need to split them into smaller structures of less than 256 bytes. For example, you'll need to declare multiple variable arrays each accessing a separate 256 byte portion of the larger structure.

To map to a portion of an array greater than a single tag, you identify the starting position for the array, and the size of the array to map using the syntax:

```
<TagName>/<ArrayOffset>!A[<ArraySize>]
```

Where:

<TagName> =	the ControlLogix tag name as defined on the PLC
/<ArrayOffset> =	the zero based offset of the item index used within the PLC (not on bit widths or other sizes) representing the starting point for the tag array value (optional)
!A<ArraySize> =	the zero based array size to return using the item count within the PLC, (not on bit widths or any other sizes)

Note: The array offset value is a zero-based item count used within the PLC to define the start of the array structure which is being mapped to the CitectSCADA variable tag array being declared. The array size value is a continuation of the same count and is used to define the size of the array structure being mapped to the CitectSCADA variable tag array being declared. They are used together to determine the start and end points of the variable tag array data in the ControlLogix system PLC which will be mapped to for the CitectSCADA variable tag array being declared in the CitectSCADA Variable Tags form.

CitectSCADA ignores all characters following an exclamation mark "!" in the tag address field, so that tag-based drivers like the ABCLX driver can use square brackets in the tag address for device array addressing. Therefore, you must precede an array address schema with an exclamation mark when defining an array in a CitectSCADA variable tag address.

Mapping data types for array tag addressing

You should match the data type of the CitectSCADA variable tag array with the mapped ControlLogix PLC tag array as per the following table:

ControlLogix Data Description	Citect Address Format	Access rights	Citect Data Type
Array of BOOL	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	DIGITAL array
Array of SINT	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	BYTE array
Array of INT	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	INTEGER array
Array of DINT	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	LONG array
Array of REAL	<TagName>/<ArrayOffset>!A[<ArraySize>]	R/W	REAL array

Examples

Configuration of an array of 128 integers

PLC Tag Address	IntegerArray
PLC Tag Type	INT[128]
Citect Tag Name	IntegerArray_128
Citect Tag Address	IntegerArray!A[128]
Citect Tag Type	INT

Configuration of an array of 64 floats offset by 32 items

PLC Tag Address	LargeFloatArray
PLC Tag Type	FLOAT[256]
Citect Tag Name	LargeFloatArray_32_64
Citect Tag Address	LargeFloatArray/32!A[64]
Citect Tag Type	FLOAT

Configuration of an array of digitals within a structure

PLC Tag Address	SingleStructure.BooleanArray
PLC Tag Type	BOOLEAN[2048]
Citect Tag Name	SingleStructureBooleanArray
Citect Tag Address	SingleStructure.BooleanArray!A[2048]
Citect Tag Type	DIGITAL

Configuration of an array of 256 bytes, offset by 256 items within an array of structures at element 32

PLC Tag Address	StructureArray.LargeByteArray
PLC Tag Type	SINT[1024]
Citect Tag Name	StructureArray_32_LargeByteArray_256_256
Citect Tag Address	StructureArray{32}.LargeByteArray/256!A[256]
Citect Tag Type	BYTE

See Also "Addressing program tags"

Addressing program tags

ControlLogix program tags contain data that is used exclusively by the routines within an individual program. These tags can be viewed as local variables.

Individual tag addressing and array tag addressing are available for program tags by using `PROGRAM:<TaskName>` as a prefix.

Examples

Single Byte within a structure in program MyProgram.

PLC Tag Address	MyProgram.SingleStructure.SingleByte
PLC Tag Type	SINT
Citect Tag Name	MyProgram_SingleStructure_SingleByte
Citect Tag Address	PROGRAM:MyProgram.SingleStructure.SingleByte
Citect Tag Type	BYTE

Single Digital in program MyProgram

PLC Tag Address	MyProgram.SingleBoolean
PLC Tag Type	BOOLEAN
Citect Tag Name	MyProgram_SingleBoolean
Citect Tag Address	PROGRAM:MyProgram.SingleBoolean
Citect Tag Type	DIGITAL

Single integer within an array of integers in program MyProgram

PLC Tag Address	MyProgram.IntegerArray[5]
PLC Tag Type	INT
Citect Tag Name	MyProgram_IntegerArray_5
Citect Tag Address	PROGRAM:MyProgram.IntegerArray{5}
Citect Tag Type	INTEGER

Configuring Quality and Timestamp tags

The CitectSCADA ABCLX driver can use additional variable tags for quality and timestamp values. Quality or timestamp tags are duplicates of a variable tag, with an appended "!Q" or "!T" or "!M" in the address field to define their purpose. Their data type needs to be Long to enable them to store a quality or timestamp value.

Hint: The quickest way to create a Quality or Timestamp tag in CitectSCADA is to locate the relevant variable tag using the Variable Tags form, edit it as required to create a quality or timestamp tag, and save it. This creates a copy of the original variable tag with the required changes saved.

To create a Quality tag:

1. Open the **Variable Tags** form in **Project Editor** (go to the **Tags** menu and select **Variable Tags**).
2. Create a new variable tag, or locate and display the tag that you want to associate the quality tag with.

3 In the **Variable Tag Name** field, enter an appropriate and unique name for the quality tag. Common practice suggests using a name such as "<*Tagname*>_Quality".

4 In the **Data Type** field, select **Long** from the drop-down menu (if not already selected).

5 In the **I/O Device Name** field, select the appropriate I/O Device for the tag (if not already displayed).

6 In the **Address** field, enter the name of the tag you want to associate with the quality value (if not already displayed), and append "!Q" directly to the end of the address (without the quotes and without any spaces).

7 Click **Add**.

Note: If the exclamation mark character "!" is already defined in the tag address field (for example, you have already used it to declare an array size "!A[n]"), don't include another exclamation mark; instead, just append the "Q" to the tag address after the closing square bracket of the array size declaration.

The quality value generated by the ABCLX driver is a mask of the quality level and quality status field values as per the following table:

Quality level	Quality status	Value (Hex)
QLEVEL_UNCERTAIN	QSTATUS_NONE	0x00
QLEVEL_GOOD	QSTATUS _NOTDEFINED	0x01
QLEVEL_BAD	QSTATUS _INITIAL	0x02
N/A	QSTATUS _STALE	0x03
N/A	QSTATUS _ERRORDATA	0x04
N/A	QSTATUS_WRITE	0x05

Hint: The quality is calculated by shifting up the quality level by 8 bits and OR masking in the quality status.

Examples Quality tag settings for a single long in root of PLC:

Citect Tag Name	SingleLong_Quality
Citect Tag Address	SingleLong!Q
Citect Tag Type	LONG

Quality tag settings for an array of 256 bytes, offset by 256 items within an array of structures at element 32:

Citect Tag Name	StructureArray_32_.LargeByteArray_256_256_Quality
Citect Tag Address	StructureArray{32}.LargeByteArray/256!A[256]Q
Citect Tag Type	LONG

To create a Timestamp tag:

1. Open the **Variable Tags** form in **Project Editor** (go to the **Tags** menu and select **Variable Tags**).
2. Create a new variable tag, or locate and display the tag that you want to associate the timestamp tag with.
3. In the **Variable Tag Name** field, enter an appropriate and unique name for the timestamp tag. Common practice suggests using a name such as "<*Tagname*>_sTimestamp" (for seconds) or "<*Tagname*>_msTimestamp" (for milliseconds) .
4. In the **Data Type** field, select **Long** from the drop-down menu (if not already selected).
5. In the **I/O Device Name** field, select the appropriate I/O device for the tag (if not already displayed).
6. In the **Address** field, enter the name of the tag you want to associate with a timestamp value (if not already displayed), and append "!T" or "!M" (as appropriate).

   Note: If there is already an exclamation mark with other declarations included in the tag address, you do not need to include them for timestamping. For example, a timestamp tag monitoring TagName!A[64] can be addressed as TagName!T.
7. Click **Add**.

Note: Timestamps store the number of seconds since 01/01/1970 when using the "!T" element, or the number of milliseconds since midnight using the "!M" element. If you want to store a complete time in millisecond accuracy, you must create two separate timestamp tags, one for seconds and one for milliseconds, each addressed appropriately.

Examples Timestamp tag settings for a single long in root of PLC:

Citect Tag Name	SingleLong_msTimestamp
Citect Tag Address	SingleLong!M
Citect Tag Type	LONG

Timestamp setting for an array of 256 Bytes offset by 256 items within an array of structures at element 32:

Citect Tag Name	StructureArray_32_.LargeByteArray_256_256_sTimestamp
Citect Tag Address	StructureArray{32}.LargeByteArray/256!T
Citect Tag Type	LONG

Note: A timestamp value is based on the time a value change is detected by the ABCLX driver in UTC time, so the timestamp is only as accurate as the polling

cycle of the I/O device. If, for example, the data value changed while the unit was not polling, the timestamp value will not be accurate.

Advanced configuration and maintenance

This section of the help provides advanced configuration instructions for following topics.

- "Customizing a project using Citect.ini parameters"
- "Configuring Redundancy"
- "Using a route path to locate a remote CPU"
- "Performance tuning"
- "Session Control"
- "Mode detection"
- "Status tags"
- "Identifying bad tags using FailOnBadData"
- "Scan rates"

Customizing a project using Citect.ini parameters

Citect.INI parameters are used to tune the performance of the CitectSCADA ABCLX driver and perform runtime maintenance diagnostics.

You can customize the way CitectSCADA communicates with the ControlLogix system, and even individual PLCs, by creating or editing the [ABCLX] section of the Citect.INI file for your project.

There are some common CitectSCADA driver settings, and custom CitectSCADA ABCLX driver settings.

When CitectSCADA starts at run-time, it reads configuration values from the Citect.INI file stored locally. Therefore, any ControlLogix configuration settings must be included in the Citect.INI file located on the computer acting as the I/O server to the ControlLogix system.

Note: By default, CitectSCADA looks for the INI file in the CitectSCADA project "\Bin" directory. If it can't find it there, it will search the default Windows directory.

Warning! The default values for these parameters have been determined (during driver development and testing) to be the best value for the CitectSCADA ABCLX driver. All of these parameters default to a value tested to work in most cases. You should carefully consider adjusting these default values except on the direct advice of Citect Customer Support.

Global and device-specific parameters

Parameters grouped beneath the [ABCLX] general category are global and impact all ControlLogix system PLCs. You can, however, override a global parameter with a device-specific parameter.

Device-specific parameters define the settings for a particular ControlLogix system PLC. The syntax for defining a device-specific ABCLX driver parameter is:

```
[<IOServerName>.<IOPortName>]
```

Where:

<IOServerName> =	the name of the I/O server
<IOPortName> =	the name of the I/O port (as defined on the I/O Ports form) which is connected to the ControlLogix system Ethernet/IP network

With the current release of the ABCLX driver, there are device-specific parameters you can use to override global parameters; they are:

- "ExclusiveConnection"
- "FailOnBadData"
- "ForwardOpenPoolSize"
- "InactiveScanAdjust"
- "LogTagInfo"
- "LogTagInfoPath"
- "MissedPollTolerance"
- "StatusTag"
- "ScanRate"

For more details, see "Driver specific Citect.INI parameters"

Examples

Implementation of global parameters:

```
[ABCLX]
Watchtime=5
ScanRate=1000
```

Implementation of a device-specific parameter:

```
[CLServer.CLServerPort]
ScanRate=500
```

Common Citect.INI driver parameters

[ABCLX] Parameter	Allowable Values	Default Value	Description
Block	1 to 256	256	A value (bytes) used by the CitectSCADA I/O server to determine if two or more packets can be blocked into one data request before being sent to the ABCLX driver.
Delay	0 to 1000	5	The period (in milliseconds) to wait between receiving a response and sending the next command to the I/O server.
MaxPending	1 to 256	256	The maximum number of pending commands that the CitectSCADA ABCLX driver holds ready for immediate execution.
Polltime	20 to 200	50	The polling service time (in milliseconds) at which cached data is updated. Recommended to be at least twice that of the value used in the Citect.INI [ALARM] ScanTime parameter .
Retry	0 to 5	3	The number of times to retry a command after a timeout.
Timeout	100 to 30000	5000	Specifies how many milliseconds to wait for a response before considering a write or polling request to be invalid and displaying an error message in CitectSCADA.
Watchtime	5 to 30	30	The frequency (in seconds) that the CitectSCADA ABCLX driver uses to check the communications link to the ControlLogix system and attempts to bring offline units back online. Also determines the frequency of STATUS_UNIT commands to the driver. See "Configuring Redundancy".

Driver specific Citect.INI parameters

[ABCLX] Parameter	Allowable Values	Default Value	Description
ConnTimeout	5000 to 30000 (milliseconds)	15000	Timeout for establishing TCP/IP connection.

[ABCLX] Parameter	Allowable Values	Default Value	Description
DebugLevel	WARNN ERROR TRACE ALL	-	Trace level used for log file See "Logging" for examples.
DebugCategory	TAG SOCK EIP DCB BUFF THRD ALL	-	Debug category. TAG: Tag configuration trace. SOCK: Socket trace. EIP: EIP session trace. DCB: Front end driver trace. BUFF: Dump a counted buffer to hex. THRD: Thread trace. See "Logging" for examples.
ExclusiveConnection	0 or 1	0	Used by the ABCLX driver to determine whether the device communicates with the serial port on a Logix 5000 series processor via the 1761-NET_ENI module. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".
FailOnBadData	0 or 1	1	Used by the ABCLX driver to determine whether or not to force the display of good data within a block read which contains bad tags during commissioning. When this parameter is set to1, a block read which contains a bad tag would result in the whole block returning #COM. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".
ForwardOpenPoolSize	1 - 32	4	Used by the ABCLX driver to determine the number of Forward Open services can be used on the device. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".

[ABCLX] Parameter	Allowable Values	Default Value	Description
InactiveScanAdjust	0 to 15	10	Used by the ABCLX driver to determine the factor to adjust the scan rate of devices which are not currently active. NOTE: The PLC will drop the connection if it is not polled within a 30 second period. Therefore, you should make sure InactiveScanAdjust*ScanRate is less than 30 seconds. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".
InitTimeOut	0 to 360000 (milliseconds)	180000	Timeout for the retrieval of the tag database and initialization functions.
LogSize	100 to 50000	2000	The maximum size (in kilobytes) of the ABCLX.LOG file before rolling over into the ABCLX.BAK file and starting a new ABCLX.LOG file. This is a system wide parameter and should be placed (if required) under the driver's general section.
LogTagInfo	0 or 1	0	Driver will generate three files: TAGLIST.TXT: Shows all tags downloaded from PLC. SUBSCRIBE.TXT: Shows all Citect tags subscribed. OPT_BLOCKS.TXT: Shows all optimized blocks the driver generated. File name will have [IOServerName]_[PortName]_ as prefix. For example: IOServer__p1_TagList.txt IOServer__p1_subscribe.txt IOServer__p1_opt_blocks.txt This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".

[ABCLX] Parameter	Allowable Values	Default Value	Description
LogTagInfoPath	Any valid path.	C:\	File path for the three files generated by LogTagInfo. If the directory does not exist, files will not be generated. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".
MissedPollTolerance	1 - 5	3	Number of polls that can be missed before the unit is taken offline. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".
ScanRate	100 to 30000 milliseconds	500	The time period (in milliseconds) between each poll of the root group. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters".
StatusTag	(comparison equation)	(no status tags)	Used by the ABCLX driver to determine whether to bring ControlLogix system PLC devices online or offline depending upon the result of the comparison equation operation as defined in the parameter. This can be system wide if placed in the driver's general section, or device specific if placed under a specific device's section. See "Global and device-specific parameters"

Using a route path to locate a remote CPU

A route path allows you to indicate the location of a particular CPU on a ControlLogix network in relation to the Ethernet module you're using as your initial point of communication. It maps out a communication channel across a network of PLCs by defining a series of ports and addresses that eventually lead to the destination CPU.

This information is configured in the [ABCLX] section of the Citect.ini file, using the following syntax:

```
[IOserver name].[Port name].RoutePath=/P:m/A:n,...,/P:m/A:n
```

where:

[IOServer name] =	your I/O server name
[Port name] =	port name on I/O server
m =	port type/number on the module 0 - Reserved 1 - Backplane 2 - Port A 3 - Port B 4 - Port C
n =	the slot number if the module is on the backplane, or the address of the module, which will either be a CNET address, IP address or DH address. It can also be an IP address if it is an EIP module

Example Starting from the initial Ethernet module, each point in the communication path is defined with a Port value and Address in the following format:

```
/P:m/A:n,
```

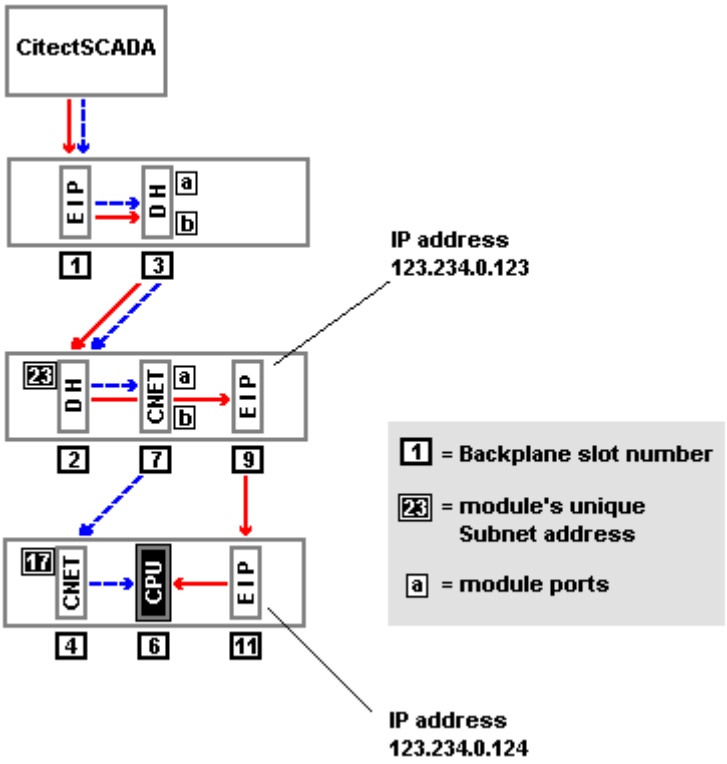
Include an appropriate numeric value for “m” to indicate if the next communication point is on the same backplane, or connected via a particular port. The value for “n” will then become a backplane slot number, or a module’s unique address.

If you are attempting to route from one Ethernet module to another, you’ll have to use “2” (Port A) for “m”, and the destination IP address for “n”. For example:

```
/P:2/A:123.234.0.124,
```

Simply recreate this syntax for each point in the communication path, separated by commas, to complete the Route Path parameter.

Consider the following diagram, showing a CitectSCADA I/O server, and a network of three ControlLogix PLC backplanes. Note there are two example route paths represented, one distinguished by a dotted line.



The syntax for the first example (solid line) would be:

```
IOServerName.PortName.RoutePath=/P:1/A:3,/P:2/A:23,/P:1/A:9,/P:2/A:123.234.0.124,/P:1/A:6
```

The syntax for the second example would be:

```
IOServerName.PortName.RoutePath=/P:1/A:3,/P:2/A:23,/P:1/A:7,/P:2/A:17,/P:1/A:6
```

Configuring Redundancy

Redundancy is handled in CitectSCADA by the CitectSCADA clients. They determine which I/O server to request I/O device data from. This is done by allowing the scan rate to be specified per I/O device, and then configuring multiple I/O devices to the same PLC.

The inactive unit under redundancy will receive a STATUS_UNIT command from Citect every configured WatchTime. The reply sent back to Citect will be based on the current state of the status tag condition. If the condition fails the standby unit will be set to UNIT_OFFLINE until the status tag condition passes. The status group polling rate remains unchanged on the inactive unit. See "Stop_Unit handling".

A status tag can be defined in the Citect.ini file to determine whether or not to set a unit offline or online depending on the condition of the status tag. If the condition is true the unit is allowed to come online, otherwise the unit is set to offline.

The status tag is polled at the same rate configured for the groups of the corresponding unit. The driver supports STATUS_UNIT commands on the inactive server, so even though a particular unit is not currently servicing Citect requests, the state of the status condition will be checked periodically allowing the unit to be taken either offline or online.

Both the primary and standby devices will be offline if both have the same status condition and the condition is failing. All points in the system for that unit would therefore be #COM.

The status tag can be defined either in the main [ABCLX] section to apply to all devices or a status can be defined specifically for a particular device within the [<IOServerName>.<IOPortName>] section.

The inactive unit under redundancy will receive a STATUS_UNIT command from CitectSCADA every configured WatchTime. The reply sent back to CitectSCADA will be based on the current state of the status tag condition. If the condition fails, the standby unit will be set to UNIT_OFFLINE until the status tag condition passes. The status group polling rate remains unchanged on the inactive unit.

Stop_Unit handling

If the driver receives a STOP_UNIT command from the I/O Server for a particular device, then the driver places the unit offline and adjusts the polling rate based on the inactive scan adjust. When the INIT_UNIT command is received again, the driver will start polling again and place the unit back online.

If the unit is currently offline, the STOP_UNIT command will not be sent down to the driver so the driver will not know to stop polling the PLC.

Performance tuning

There are two ways you can fine tune the performance of the ABCLX driver on your CitectSCADA system:

- "Optimizing driver to PLC communications"
- "Optimizing CitectSCADA's client to server communications"

Optimizing driver to PLC communications

The ABCLX driver connects to the ControlLogix system devices using the Ethernet/IP network. To know the connection state of all configured PLCs, the driver supports the use of Status tags for the system and for each device.

To minimize network traffic, the ABCLX driver also supports the use of different scan rates for different devices, and active and inactive devices. See "Status tags" and "Scan rates".

Note: The ABCLX driver maintains an internal tag cache and services CitectSCADA DCB requests from the internal cache.

You can also adjust the System Overhead Time Slice in the PLC to the highest value possible without interfering with the running of operations on the PLC. The System Overhead Time Slice in the PLC is adjusted from the Controllogix programming software by right clicking on the controller and navigating to the "Controller Properties" then click on the "Advanced" tab.

It is also good engineering practice to arrange your variable.dbf by IO Device then DataType. This allows Citect to create requests by blocking the same data types within an IO Device together.

For further information on optimizing PLC communications, see the CitectSCADA Knowledge Base article **Q4035**. (The Knowledge Base can be installed from the CitectSCADA CD, or downloaded from **www.citect.com**.)

Optimizing CitectSCADA's client to server communications

Due to this driver being a front-end/back-end driver, there are no transport delays between CitectSCADA and the driver involved in processing a read request from the driver cache. This means that under heavy loads, the driver could use up most the CPU processing requests. The driver can tune the CPU usage and throughput to help manage the resources of the I/O server. The standard driver parameters MaxPending and Delay can be used as follows:

```
[ABCLX]
MaxPending=x
Delay=y
```

where:

x =	the maximum number of simultaneous requests the driver will receive from CitectSCADA.
y =	the average delay applied to the requests to control CPU usage.

See "Customizing a project using Citect.ini parameters".

Examples

```
[ABCLX]
MaxPending=256
Delay=50
```

The above example would mean that the driver is capable of servicing 256 requests every 50 milliseconds.

```
[ABCLX]
MaxPending=1
```

```
Delay=0
```

This example would mean the driver would reply to every request as quickly as possible, which could cause CPU issues.

Session Control

You can use session control to release any current references to tags on the ControlLogix PLCs. This is required if you want to modify or delete any of your PLC tags, as the Logix5000 software cannot access a tag if CitectSCADA is currently subscribed to it. You can also use session control to load a refreshed tag list from a PLC.

Session control is achieved by using the **SessionDisable:x** tag. This tag needs to be configured on a virtual device on a virtual channel. The virtual channel is specified by entering 0.0.0.0 as the IP address in the **Special Options** field of the Ports form. There should be a single virtual port and virtual device configured for each I/O server, and the virtual devices should not be configured for redundancy; i.e., the network numbers for the virtual units should not be the same. Otherwise it is not possible to communicate with a specific virtual unit from a CitectSCADA client.

Device Data Description	Citect Address Format	Access rights	Citect Data Type
Session Disable and Enable	SessionDisable: x	R/W	INTEGER

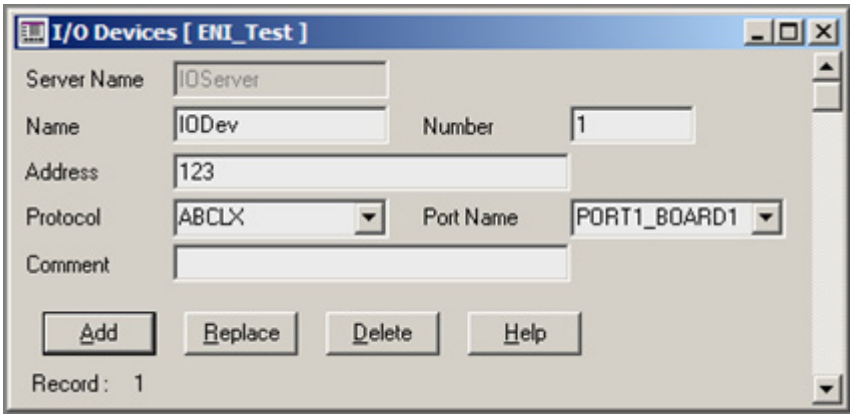
Where:

x = the address entered for the specified device in the I/O Devices form. It needs to be unique across all systems.

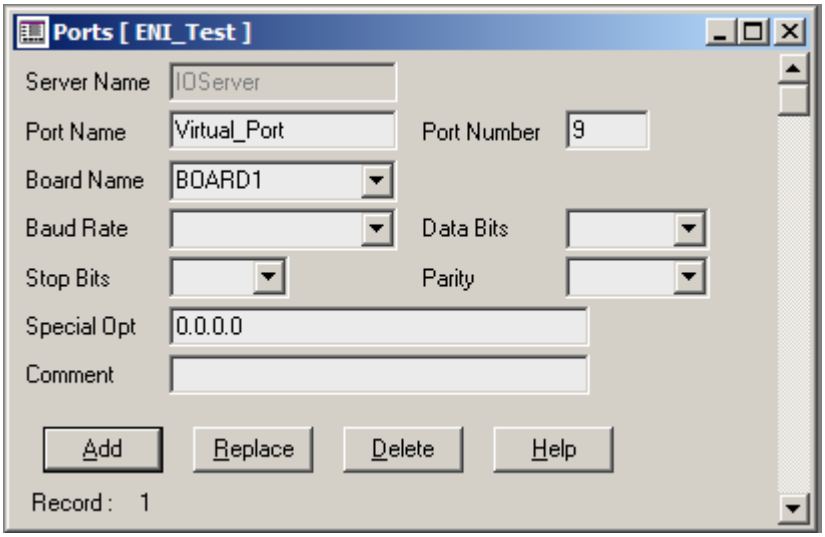
If this tag is written to with a 1, the session is disabled. Writing a 0 to this tag will re-enable the session.

Example

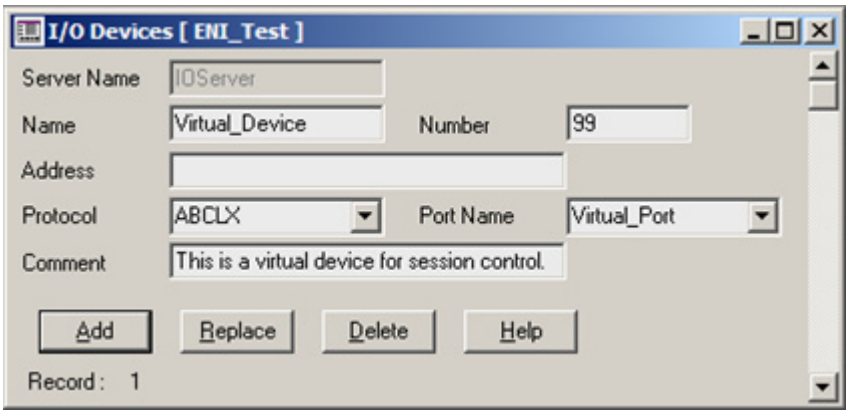
The following example will operate on the I/O Device with 123 in the address field of the I/O Device Form.



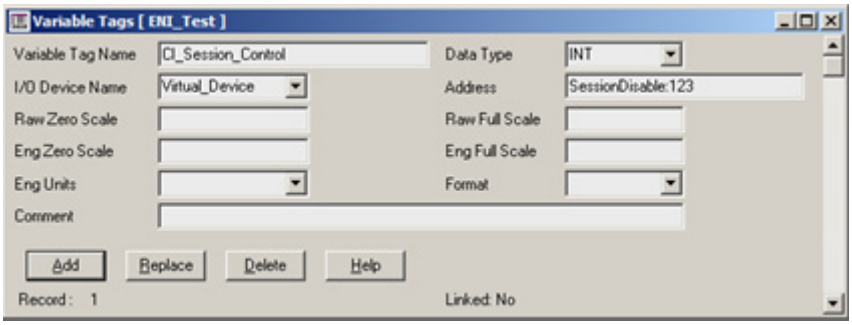
Firstly, create a virtual channel by entering 0.0.0.0 as the IP address in Special Opt field in Ports Form.



Create a virtual device.



Create a virtual tag on the virtual device.



This will allow you to control the session on IODev device by writting 0 or 1 to the virtual tag CI_Session_Control.

See Also "Status tags"

Status tags

A status tag can be defined in the Citect.INI file to determine whether or not to set all or any ControlLogix system PLC devices offline or online depending on the condition of the status tag. If the condition is true, the device is allowed to come online; otherwise the device is set to offline by the ABCLX driver.

There can be one general status tag (defined in the general [ABCLX] section of the project Citect.INI file) which determines the online status of all I/O devices using the ABCLX driver. There can also be an individual status tag set for each I/O device (using the syntax [IOServerName.IOPortName]) that will override the general section status tag. See "Customizing a project using Citect.ini parameters".

By default, no status tag is created for the driver or a device until defined in the project Citect.INI file.

The status tag is polled at the same rate configured for the groups of the corresponding device. The ABCLX driver supports STATUS_UNIT commands on the inactive server, so even though a particular device is not currently servicing CitectSCADA requests, the state of the status condition will be checked periodically allowing the device to be taken offline or online. See "Stop_Unit handling".

Note: If both the primary and standby devices have the same status condition and the condition is failing, both will be taken offline. All points in the system for that unit will display "#COM".

The syntax for a status tag is:

```
StatusTag=<TagName><Operator><Operand>
```

where:

<TagName> =	the name of the PLC tag address whose value is to be used in the comparison operation;
<Operator> =	the comparison operator (see operator list below);
<Operand> =	the value to be used in the comparison operation against the value of <TagName>.

The comparison operator can be one of the following:

Operator	Description
>	<TagName> is greater than <Operand>
>=	<TagName> is greater than or equal to <Operand>
<	<TagName> is less than <Operand>
<=	<TagName> is less than or equal to <Operand>
&	<TagName> has common asserted bit(s) with <Operand>
=	<TagName> is equal to <Operand>
!&	<TagName> has no common asserted bit(s) with <Operand>
!=	<TagName> is not equal to <Operand>

Examples

Bring all ABCLX driver devices online only when the integer tag named "TankLevel" is greater than 100:

```
[ABCLX]
StatusTag=TankLevel>100
```

Bring any device on the port named “Port1” on the I/O server named "CLServer" online only when the digital tag named "isReady" is true (where 0=false and 1=true):

```
[CLServer.Port1]
StatusTag=isReady=1
```

Bring the particular device named "Port123" on the I/O server named "CLServer" online only when the current value shares common asserted bits with “5”.

```
[CLServer.Port123]
StatusTag=TagName&5
```

If the current value for the tag called “TagName” was 3, the example above would set the status to true and bring the device online, as 3 and 5 share a common bit (3 = 001**1** in binary, and 5 = 010**1**). If the value for TagName were 2, the device would remain off line as 2 and 5 do not share common bits (2 = 0010, 5 = 0101).

With the example below, the port named "Port123" on "CLServer" will come online only when the current value for TagName does not share common asserted bits with “5”.

```
[CLServer.Port123]
```

```
StatusTag=TagName!&5
```

Mode detection

The PLC mode is determined by two things. Firstly, the physical key position in the PLC; and secondly, when in REMOTE mode, whether or not the PLC has been set to RUN mode or PROGRAM mode. This state is represented in the PLC by a numeric value that can be queried.

The following table sets out the different values and the corresponding PLC mode:

Mode Value	Key Position	Mode
0x1060	RUN	RUN
0x3060	REMOTE	RUN
0x3070	REMOTE	PROGRAM
0x2070	PROGRAM	PROGRAM

These values can be used to set up a status tag to determine what mode the PLC is currently in, as shown below.

Status Tag	Configuration Description
@Mode<0x2000	Key position in RUN
@Mode>0x3000	Key position in REMOTE
@Mode!&0x1000	Key position in PROGRAM
@Mode!&0x10	PLC in RUN mode
@Mode&0x10	PLC in PROGRAM mode

Hence the unit can be taken offline or back online depending on the key position and the current mode of the PLC to display #COM for the points on this unit.

Identifying bad tags using FailOnBadData

A bad tag is defined as any tag that does not exist within the ControlLogix system, which is a problem that can occur when developing large systems.

During the commissioning phase a project, you need to clearly identify any blocks of data that include bad tags. The “FailOnBadData” parameter helps achieve this by allowing you to run your project in two different modes:

- When **FailOnBadData** is **off**, the ABCLX driver will not return "#COM" for a block of data if at least one valid value is in the read block.
- If **FailOnBadData** is **on,** if any values in a blocked read fail, the whole block is failed as "#COM". This highlights the presence of bad tags and identifies where corrective action is required. This default behavior is the preferred mode of operation when you are running a project.

A value of 1 (default) indicates that commissioning mode is ON, 0 indicates commissioning mode is OFF. See “Driver specific Citect.INI parameters”

Note: When you've completed commissioning, you should reset the Commissioning parameter to zero or delete it from the project Citect.INI file.

Example Display bad tag reads with #COM during project commissioning:

```
[ABCLX]
FailOnBadData=1
```

Ignore bad tag reads during normal operation:

```
[ABCLX]
FailOnBadData=0
```

Scan rates

The CitectSCADA ABCLX driver can define I/O devices with different scan rates. This provides you with the ability to better tune your system, by ordering the tag database into logical groups and defining different scan rates for each group. For example tags that are being read frequently can be placed in a group that is scanned frequently, and tags that are read infrequently can be placed in a group that is scanned infrequently.

By default, all tags are scanned at the default scan rate defined by the ScanRate parameter in the project Citect.INI file. See Citect.INI specific parameters for details.

Note: If you have configured I/O server redundancy in your project, it may be considered an inefficient use of network bandwidth to be continually polling devices non-critical to the currently inactive server. To help reduce unnecessary network traffic, the ABCLX driver supports the use of the InactiveScanAdjust parameter, which applies a delay factor to the scan rate of currently inactive I/O Servers. See “Driver specific Citect.INI parameters” for details.

Examples Set the project scan rate to 1 second (1000 milliseconds):

```
[ABCLX]

ScanRate=1000
```

Set the project scan rate to 1 second (1000 milliseconds), and the inactive server scan adjust rate to a factor of 5:

```
[ABCLX]

ScanRate=1000

InactiveScanAdjust=5
```

Set a faster (half second) scan rate for a specific port named “Port1” on the I/O server named "CLServer":

```
[CLServer.Port1]

ScanRate=500
```

Set a slower (5 second) scan rate for a specific port named "Port123" on the I/O port named "CLServer":

```
[CLServer.Port123]

ScanRate=5000
```

Troubleshooting

Most large projects will suffer 'bugs' in the run-time system. However, most problems have simple solutions and require only perseverance to solve them.

The following topics provide information about the CitectSCADA tools provided to help resolve problems with communication and configuration.

- "Hardware Alarms"
- "The SysLog.DAT file"
- "Driver Errors"
- "Logging"
- "Citect Kernel diagnostics"
- "Maintaining the project database"
- "Tag-based driver considerations"

For answers to common problems that relate more specifically to the ABCLX driver and its operation, see "Frequently Asked Questions".

Hardware Alarms

When a system error occurs that is a malfunction in CitectSCADA operation, CitectSCADA generates a hardware alarm. Hardware alarms are usually displayed on a dedicated Hardware Alarm page, which is available as a standard template.

The hardware alarm page is your primary indicator of what is happening in your CitectSCADA system. If a communication fault occurs, if Cicode can't execute, if a graphics page is not updating correctly, or if a server fails, this page will alert you to it. Hardware alarms consist of a unique description and error code.

The hardware alarms do not have detailed information, but point you in the direction of a problem. For example, if you have a Conflicting Animation alarm, CitectSCADA will not tell you the cause. You must observe which page causes the hardware alarm, and locate the animations yourself.

Note: Your system should have no recurring hardware alarms. If, after reviewing all documentation, you cannot rectify an alarm, contact Citect Technical Support.

The SysLog.DAT file

The SysLog.DAT is a file maintained by runtime CitectSCADA, that contains a useful log of CitectSCADA System information. The variety of information that can be logged to the SysLog.DAT is extensive: from low level driver traffic and Kernel messages, to user defined messages.

The Log Read and Log Write fields in the I/O Devices Properties control whether logs are made for each I/O Device.

Note: CitectSCADA locks the SysLog file while running. However, you can still view it by using the SysLog command in the Kernel. See "Citect Kernel diagnostics", and the CitectSCADA User Guide for details.

Driver Errors

CitectSCADA ABCLX driver errors can occur when a ControlLogix Device fails to respond, or is disconnected or offline, or returns an error itself. Error codes are listed below.

The ABCLX driver provides the ability to log combinations of trace levels across different categories. For more information, see "Logging".

Common Protocol Errors

CitectSCADA has two kinds of protocol driver errors - generic and specific. Generic errors are hardware errors 0-31, and are common to all protocols. There are a number of generic errors that are also common to all protocols. In some cases only the generic error is available, though often both the generic error and a specific error are given.

Protocol drivers also have their own specific errors, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert their specific errors into generic errors that can be identified by the I/O Server. For example, when a driver has a fault, there is often both a protocol-specific error and a corresponding generic error.

When a hardware error occurs, CitectSCADA generates an alarm, and displays the alarm on the hardware alarm page (in the alarm description of the hardware alarm). To see the error number, make sure you have Alarm Category 255 defined with a display format that includes {ErrDesc,15} {ErrPage,10}.

The following errors, listed in (hexadecimal) sequence, are common to all protocols. CitectSCADA displays the error number and description for common protocol-specific errors. Uncommon errors are not contained in the CitectSCADA error database, in which case CitectSCADA will only display the error number.

Error Value (in Hex)	Error Definition
0x20000007	Failed to read tag
0x20000006	Failed to write to tag
0x20000009	Failed to initialise channel and download PLC tag database
0x2000000B	Failed to initialise unit and start polling tags
0x2000000C	The configured status condition is currently false
0x20000010	Failed to access the Citect Variable.DBF file
0x20000011	Failed to access the Citect Units.DBF file
0x20000015	Unit initialisation is still in progress
0x20000016	The PLC session or TCP/IP connection has failed

Note: If one of these errors occurs by itself and is persistent, you should contact Citect Support.

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the I/O Device (or network). If, after reviewing all documentation, you cannot rectify an error, contact Citect Technical Support.

ABCLX Protocol Errors

The following errors, listed in (hexadecimal) order, are specific to the CitectSCADA ABCLX protocol:

Error Value (in Hex)	Error Definition	Error Description
0x13	Driver cannot cancel	Ignore, non critical internal issue
0x17	Driver unit offline	ControlLogix unit no longer responding
0x18	Bad data type	Invalid format or ABCLX.dbf
0x19	Bad unit type	Check, ABCLX.dbf, re-compile
0x1E	Address range error	Invalid format or ABCLX.dbf
0x800	Windows error	Associated with an operating system call
0x100 to 0x899	ABCLX API calls	Refer to Citect Support

You may require additional information to enable you to rectify an error. This information should be detailed in the documentation that accompanied the ControlLogix system.

Logging

Logs are a useful tool when it comes to debugging a driver.

Setting the parameter "LogTagInfo" to 1 allows you to generate three files:

TAGLIST.TXT	Shows all tags downloaded from PLC.
SUBSCRIBE.TXT	Shows all Citect tags subscribed.
OPT_BLOCKS.TXT	Shows all optimized blocks the driver generated.

The location of these files is determined by the parameter "LogTagInfoPath" which accepts any valid path. The default setting is "C:\". Note that if the destination directory does not exist, the files will not be generated.

The file names will have `[IOServerName]_[PortName]_` as a prefix.

For example:

```
IOServer__p1_TagList.txt
IOServer__p1_subscribe.txt
IOServer__p1_opt_blocks.txt
```

The ABCLX driver also provides the ability to log combinations of trace levels across different categories. This is achieved by setting the following ini parameters:

- "DebugLevel"

  Allows you to define the trace level. The options include warnings, errors traces, or all of the above.

- "DebugCategory"

  Allows you to enable logging for a particular category of trace. The options include:

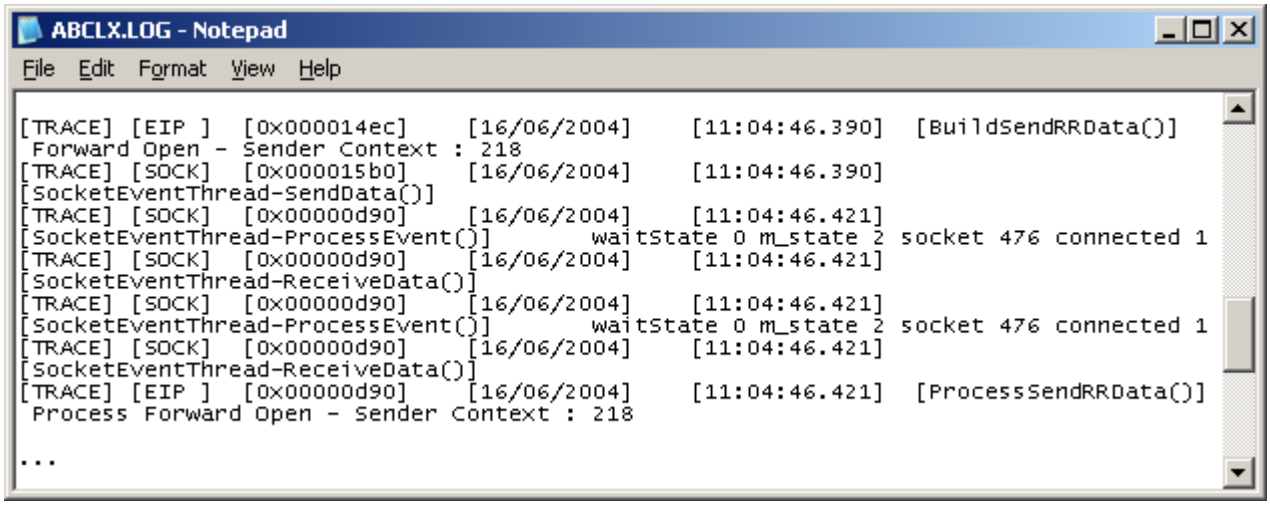
TAG	Tag configuration trace
SOCK	Socket trace
EIP	EIP session trace
DCB	Front end driver trace
BUFF	Dump a counted buffer to hex
THRD	Thread trace
ALL	all of the above

Note: The more detailed the logging, the larger the amount of log data will be generated. This could create a large, cumbersome log file, or cause multiple wraparounds that write over data.

Example The following parameter settings:

```
[ABCLX]
DebugLevel=WARNN|ERROR|TRACE
DebugCategory=ALL
```

would send the following log information to a file called **ABCLX.log**:



This file is delivered to the Windows system directory (e.g. `C:\Windows` on Windows XP). It's location is not configurable.

See the CitectSCADA Knowledge Base article Q3994 for further examples of ABCLX logging.

See Also "Driver specific Citect.INI parameters"

Citect Kernel diagnostics

The Citect Kernel window diagnostics framework (commonly referred to as the Kernel), is the primary gateway into the internal workings of CitectSCADA at runtime, and is provided for diagnostics and debugging purposes only.

The Kernel can display several different diagnostic windows each displaying an active view into the workings of the CitectSCADA runtime system. Each window is displayed when selected from the Citect Kernel View menu.

For more details of using the Citect Kernel, see the CitectSCADA User Guide.

Warning! Use the Kernel window with care, because once you are in the Kernel, you can execute any Cicode function with no privilege restrictions. You (or anyone using the Kernel) has total control of CitectSCADA (and subsequently your plant and equipment).

Setting up the Kernel

You need to configure your CitectSCADA project to provide run-time access to the Kernel diagnostics window. You can do one or both of the following:

- enable the Kernel on CitectSCADA start-up; or
- add the Kernel menu item to the runtime CitectSCADA Control menu.

To enable the Kernel diagnostics window on CitectSCADA start-up:

1 In the Citect Explorer menu, select **View | Configuration File**. This launches Windows Notepad and loads and displays the Citect.INI file for manual editing.

2 Scroll down to the "[Debug]" category, and edit the section to include the following parameter:

```
[DEBUG]
Kernel=1
```

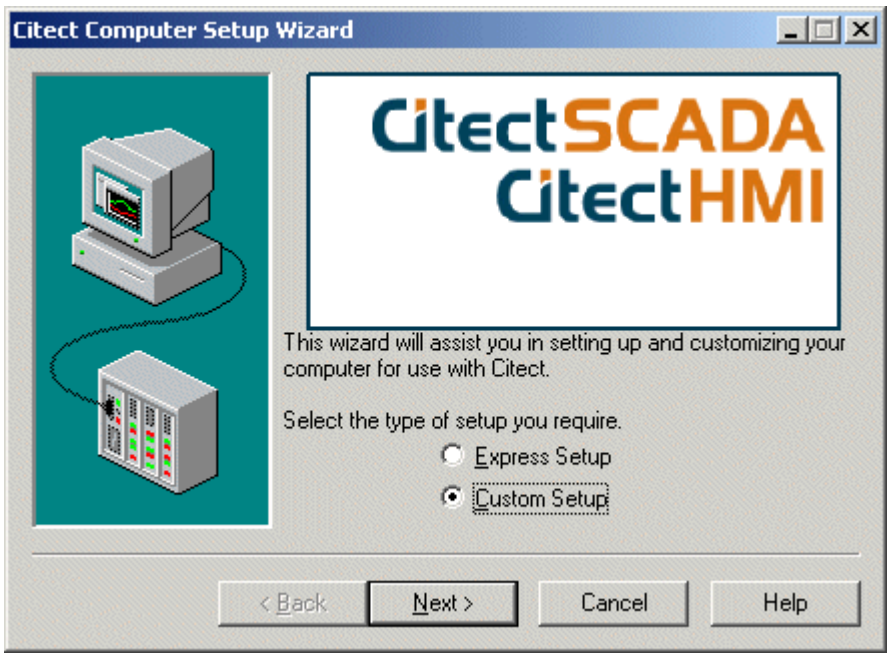
3 **Save** and **Close** Notepad.

Note: The Citect.INI file is located in the Windows folder. Typically: "C:\WINDOWS" or "C:\WINNT".

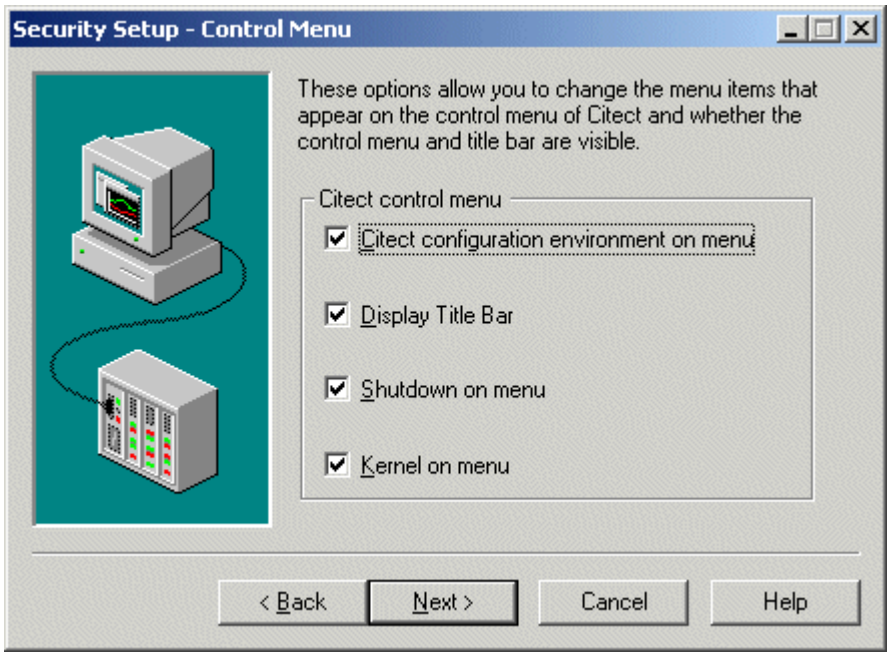
This parameter will cause the automatic display of the Citect Kernel window diagnostics framework upon start-up of CitectSCADA. Swap to it to view what's happening inside CitectSCADA at runtime.

To add the Kernel menu item to the CitectSCADA runtime Control menu:

1 In **Citect Explorer** or **Project Editor**, launch the **Computer Setup Wizard** and select **Custom** mode:



2 Click Next until you reach the **Security Setup - Control Menu** page:



3 Check the **Kernel on menu option** and click **Next** until finished.

At runtime, you can display the Kernel window by selecting Kernel from the **Control** menu (top left corner) of CitectSCADA. If you do not have a Title Bar displayed in your runtime project, you can access the Control menu by pressing **Alt + Spacebar** (if the Alt-Space enabled option is ticked on the Security Setup - Keyboard page).

Note: You should uncheck these options before handover, to prevent accidental or unauthorized use of the Kernel in the delivered system.

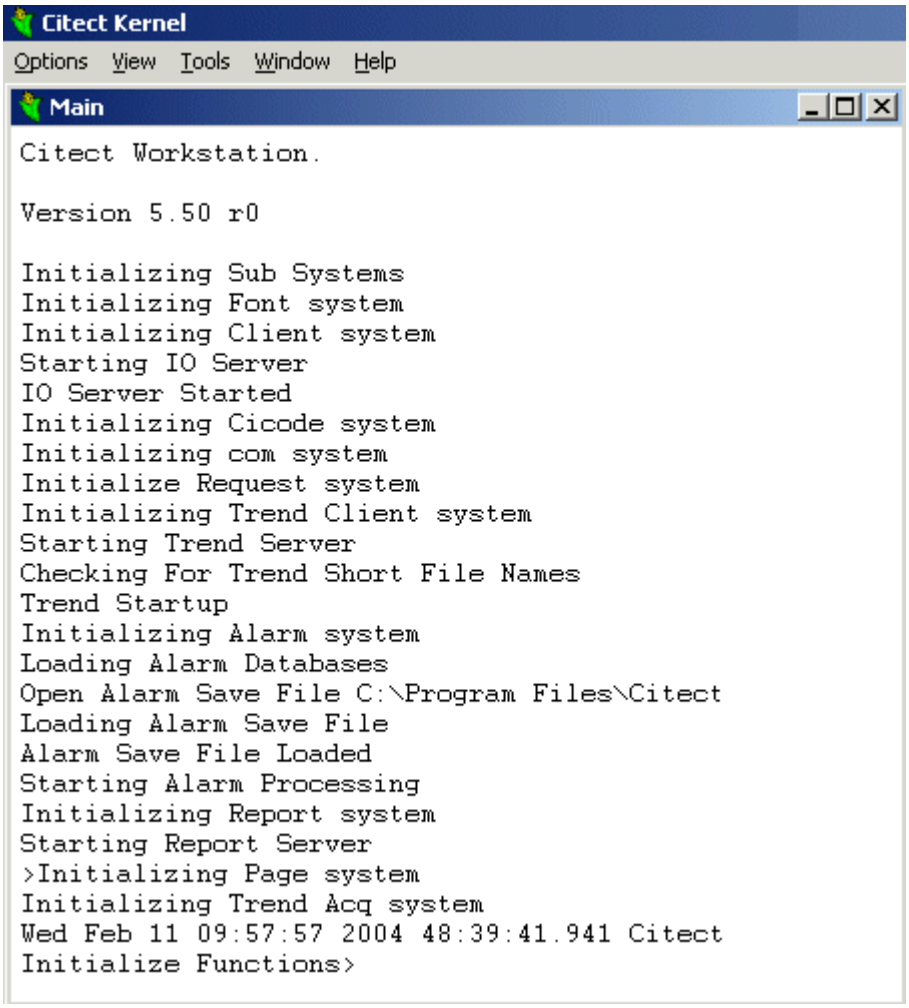
The Diagnostic Windows

The key diagnostic windows included with the Kernel for testing I/O device communications are the **Main** window, the **I/O Devices** window**,** and the **Driver Statistics** window:

- The "Citect Kernel Main window" displays the diagnostic messages a line at a time indicating the current CitectSCADA startup operation and status. When running, the Main diagnostics window continues to report changes in the status of the I/O devices.
- The "Citect Kernel Unit window" displays the current status of all devices defined in the I/O Device database.
- The "Citect Kernel Driver window"displays information about each driver in the CitectSCADA system. This window is only displayed if the computer is configured as an I/O server with physical I/O devices attached.

Citect Kernel Main window

After your I/O devices are properly configured, the Main window is particularly useful to check that all of your I/O devices come online correctly when CitectSCADA is started.



First the ports will be initialized, then the I/O Device will be brought online. If there is a problem, CitectSCADA will display a message; "PLC not responding", "I/O Device Offline" or similar.

Some I/O Devices may take two attempts to come online. If this is the case, CitectSCADA will wait (usually 30 seconds) and try again. If your I/O device does not come online after the second attempt, you should check your configuration (at both ends) and the network in between.

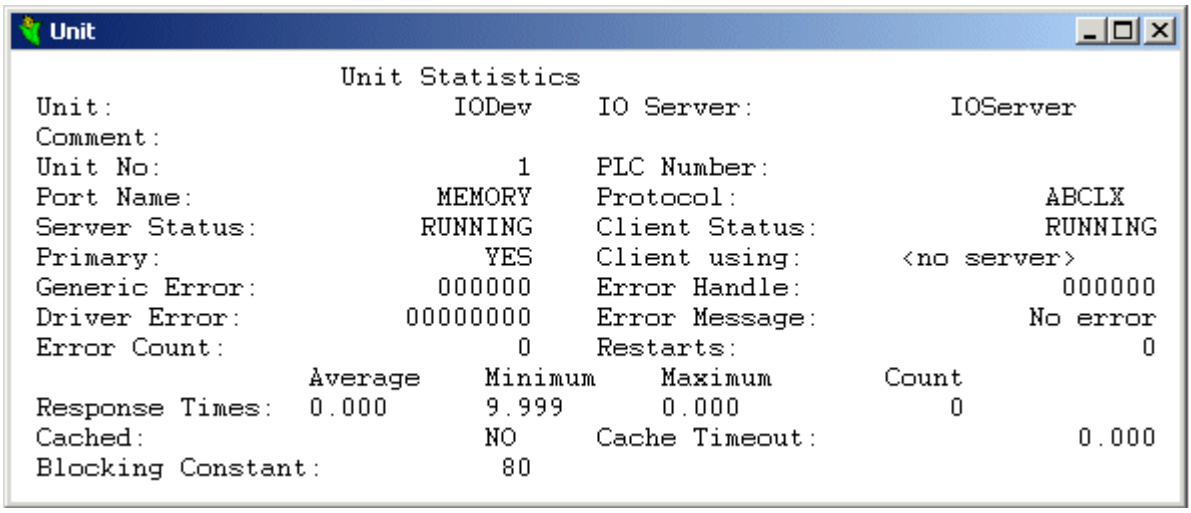
Common problems that may cause startup errors include:

- Incorrect computer setup - usually solved by the Computer Setup Wizard
- Networking faults or bad hardware - cables wrongly wired or unplugged, or equipment not powered up, or faulty equipment
- Communication faults - most often a configuration mistake like wrong protocol settings, wrong port, etc.

Note: The Kernel Main diagnostic window contains legacy behavior from the early days of Windows. It has no scroll bar, so as new messages are added a line at a time to the end of the displayed list of messages, and the window is filled, the oldest of messages (at the top of the window list) will scroll out of view and be irretrievable. However, you can view this information using "The SysLog.DAT file".

Citect Kernel Unit window

The Unit diagnostics window, launched via the Page Unit Kernel command, displays the current status of all devices defined in the I/O Device database. In this window, use the **Page Up** or **Page Down** keys to browse through the available devices.

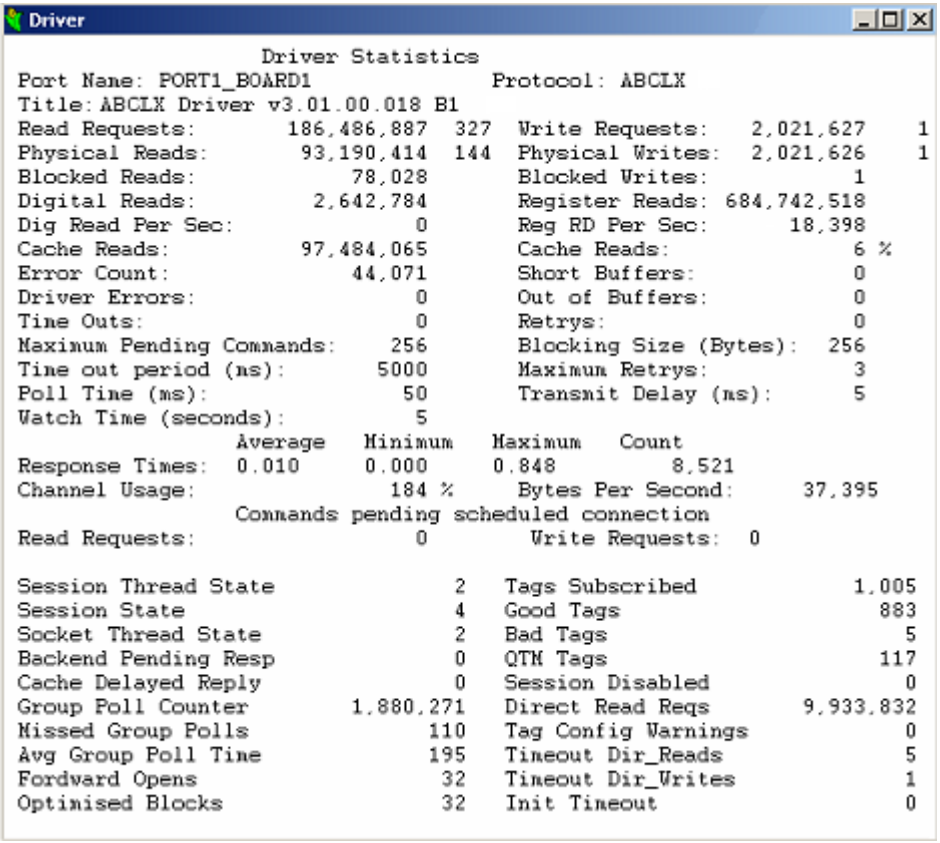


The important values to check to confirm that communications with the I/O device are enabled and running are the **Server Status** and **Client Status** fields. When the I/O device is online, these will both display "RUNNING".

Search for the Kernel Page User Command in the Citect SCADA User Guide for a description of all the fields displayed for each I/O device in the Unit diagnostics window.

Citect Kernel Driver window

The Citect Kernel Driver window, launched via the Page Driver Kernel command, displays information about each driver in the CitectSCADA system. The statistics it presents include read requests, physical reads, digital reads per second, register reads, cache reads, error count, timeouts, and so on. See the CitectSCADA User Guide for more information on the general statistics it presents.



Note that in the case of the ABCLX driver, the Kernel Driver window includes additional fields that are not normally supported. You will need to press "V" to see these additional fields. They are:

Field	Description
Session Thread State	Current state of the driver EIP session thread: 0 – Thread is stopped 1 – Thread is starting 2 – Thread is running 3 – Thread is stopping

Field	Description
Session State	Current state of the driver EIP session to the PLC: 0 – The session has not been initialized yet 1 – The session is connecting to the PLC and downloading the PLC tag database 2 – The session has connected to the PLC and downloaded the PLC tag database 3 – The session is subscribing the tags to the PLC 4 – The session is currently polling the subscribed tags 5 – The session has failed and will be restarted
Socket Thread State	Current state of the socket to the PLC (same thread states as Session Thread State above)
Backend Pending Resp	Number of requests currently awaiting response from the PLC
Cache Delayed Reply	Number of cache read requests currently being delayed by driver
Group Poll Counter	Number of polls sent out by the driver
Missed Group Polls	Number of polls that were skipped by the driver
Avg Group Poll Time	Average time in milliseconds for a complete scan of the tags
Forward Opens	Number of Forward Open sessions to the PLC
Optimized Blocks	Number of optimized blocks created by the driver. Should match number of Forward Opens.
Tags Subscribed	Number of non Quality and non Timestamp Citect tags for this unit
Good Tags	Number of tags that were successfully resolved in the PLC
Bad Tags	Number of tags that were not successfully resolved in the PLC
QTM Tags	Number of Quality and Timestamp tags subscribed to
Session Disabled	Indicates whether session is currently disabled. Value of 1 indicates session is disabled.
Tag Config Warnings	Number of tags that still are considered good but have some issue that needs to be looked at
Init Timeout	Number of connection attempts that have been timed out

Maintaining the project database

After you've edited your project for a while, the project database can become fragmented and problematic. Good practice during project development is to periodically clear out all duplicated records and any orphaned ones.

Improperly maintained databases have proven to be a source of communication problems at runtime. The majority of all communications problems come from having duplicated or orphaned records in the communications database.

Sometimes there will be 'orphaned' records from a previous I/O Server left behind in the database files. As all the forms are indexed on the I/O Server name, you can use the scroll bar to quickly navigate to the end of the current range of records for a particular I/O Server and examine the last few and next few records to validate that they should be there. If you find any extra (unwanted) records, delete them.

You should also make sure that there are no duplicated records. Use the record search facility to search for duplicate entries of devices that are causing unexplained communication errors. If you find any extra (unwanted) records, delete them.

Once you have deleted orphaned and duplicated records, pack the project. Packing the database removes deleted records, and re-indexes the database. To pack the project databases, from the File menu in Citect Project Editor, select Pack.

There is information about the way CitectSCADA handles OIDs that you should be aware of before attempting the tasks mentioned here. For more information on OIDs, see the CitectSCADA Knowledge Base article **Q3657**. (The Knowledge Base can be installed from the CitectSCADA CD, or downloaded from **www.citect.com**.)

Tag-based driver considerations

The CitectSCADA ABCLX driver uses OIDs (object identifiers) to uniquely identify tag addresses. These OIDs are generated by CitectSCADA during compilation.

You need to be aware of the way OIDs are implemented for a tag-based driver, as this will avoid potential tag mismatches occurring. This is a particular concern with network distributed systems. For example, all the CitectSCADA servers and clients must have the exactly the same variable database on them, otherwise a client could make a request for a tag with an OID that either does not exist on the I/O Server, or refers to a different tag. For this reason, care must be taken when editing a variables database.

The following information is intended to draw attention to activities that require careful attention because of the potential impact they may have on the OIDs in your system.

For more information on OIDs, see the CitectSCADA Knowledge Base article **Q3657**. (The Knowledge Base can be installed from the CitectSCADA CD, or downloaded from **www.citect.com**.)

Project IDs

OIDs are generated using a Project ID, and the position (i.e. record number) in the associated variable.dbf file. Therefore, it is imperative when using display clients, standby servers, and so on, that a project restored to a PC has a consistent Project ID across all the machines included in a system.

This should be checked whenever you restore a project to a PC, because if the Project ID is already used by another project on that particular computer, Citect will generate a new Project ID number.

A project's ID number can be determined by selecting a project in Citect Explorer and viewing its properties. The Project ID is shown on the "General" tab. It is also stored in the project backup file (.ctz).

Using Include Projects

You should ensure that each include project has the same Project ID on each computer using the same include project. Again it is imperative that this is checked after restoring a project to a computer.

Variable Databases

If you have different variable.dbf files across your plant, it is possible to have OID mismatches.

To avoid this, ensure that once you have made changes to your variable.dbf on your Primary Server, that you reset the OID (in citect.ini file set [OID] Reset=1) and do a pack and full compile (i.e. ensure that incremental compile is turned off).

Once this is done, the variable.dbf file should be copied to the Standby Server and any Display Clients using the project. This ensures that an OID on your Primary Server is the same as the OID used on your Display Client and/or Standby Server.

Hint: If you are using include projects, you should pack all your variable.dbf before resetting the OIDs and doing a full compile of your project. There is a downloadable tool available called **PackAll** that will pack all the variable.dbfs that are used in your project (even if they are include projects). This tool is located under "Database Tools" in the **Toolbox** section of **www.citect.com**.

See Also "Frequently Asked Questions"

Frequently Asked Questions

This section addresses problems that may be encountered when setting up and operating the ABCLX driver. They have been grouped into categories to help you easily determine the likely solution to the type of problem you're encountering.

The answers to these questions typically point to an article in the CitectSCADA Knowledge Base, which can be installed from the CitectSCADA CD. The most

current version of the Knowledge Base can also be downloaded from the Citect web site (www.citect.com).

Startup issues

- "Why do I get a "Channel Offline Cannot Talk" error message when the ABCLX driver is coming online?"
- "Why does it take so long for the ABCLX driver to come online?"
- "Every time I start CitectSCADA, I see a nuisance hardware alarm for my ABCLX device that becomes inactive after about 30 seconds. How do I stop this happening?"

CTAPI issues

- "Using CTAPI to read tags from an ABCLX device causes blank tags to appear."

Why do I get a "Channel Offline Cannot Talk" error message when the ABCLX driver is coming online? There can be a number of reasons why ABCLX driver cannot bring its devices online at startup. You should initially confirm that the Boards, Ports and I/O Devices forms are filled out correctly. You should also confirm that you have enough idle time in your PLC program to allow communication requests to take place.

See the Knowledge Base article **Q4009** for further information on how to diagnose this problem. It also includes information about parameter settings and adjustments that may impact this process.

Why does it take so long for the ABCLX driver to come online? When the ABCLX driver attempts to bring devices online, it initiates a process of downloading and subscribing the required tags from the associated devices. If a project has a large number of tags, this process may overlap with driver polling, causing unnecessary delays. Insufficient idle time in the PLC processor may also cause Citect driver requests to be prolonged unnecessarily.

The Knowledge Base article **Q4001** provides a description of how CitectSCADA brings the ABCLX driver online.

Every time I start CitectSCADA, I see a nuisance hardware alarm for my ABCLX device that becomes inactive after about 30 seconds. How do I stop this happening? Hardware alarms are generated by CitectSCADA whenever InitUnit fails. In the case of ABCLX driver, this can occur because the required tags have not downloaded and completed subscription.

As a result, there is no current workaround or solution for suppressing this hardware alarm. However, it should be noted that the speed at which a driver comes online can be impacted by network bandwidth, PLC CPU utilization, and the number of tags and the number of bad tags.

Please refer to Knowledge Base articles **Q4001** and **Q4009** for more information on how to diagnose and fine tune the startup process.

Using CTAPI to read tags from an ABCLX device causes blank tags to appear. Using CTAPI to read tags from a device using the ABCLX driver causes the first tag to display its value correctly, but the rest to appear blank. This is caused by a blocking issue in CTAPI.

The workaround is to change the Protdir.dbf entry for ABCLX from a Max_Length of 2048 to 2032, and Bit_Block from 2048 to 2032 on all Citect computers on the system. This will ensure that all the requested tags are read correctly. Please note, however, that this will reduce the maximum request length from 256 bytes to 254 bytes, which reduces the maximum size of arrays and strings.

It is imperative that this change be made to the protdir.dbf file in the \Citect\Bin and the \Citect\User\Include directories.

See the Knowledge Base article **Q4068** for more details.

Index