



Citect Pty Ltd ABN 88 001 158 854
3 Fitzsimons Lane Gordon 2072 NSW Australia
PO Box 174 Pymble 2073 NSW Australia
Tel 61 2 9496 7300 Fax 61 2 9496 7399 www.citect.com



v5.4x

ABLOGIX driver, User Information and Design

Driver Spec Version History

Version	Date	Modified By	Details
1.0	-	Su Kunzhe	Draft
1.2	27/6/02	Graeme Sutton	Updated for driver v1.07.03b1
1.3	11/7/02	Graeme Sutton	Minor updates after review
1.4	24/9/02	ACME	Update for new supported data formats and bug fixes. Document accurate for v1.02.08.x .
1.5	16/10/02	ACME	Adjust spec for v2.0.0.x changes for performance enhancements
1.6	6/01/03	CitectSCADA	Update Get/SetDcbDWord to Get/SetDcbWord
1.7	14/01/03	Greg Roberts	Updated to driver version 2.0.3.x. Status checking has been changed. INI Maxtimeouts has been removed.
1.8	24/01/03	Greg Roberts	Updated to driver version 2.0.4.x
1.9	13/02/03	DP/LG	Removed parameter OnErrorChannelOffline.
1.10	04/06/03	LG	Bug20411 [ABLogix]UseIncludeProjects currently defaults to 0, but 1 would be better.

Contents

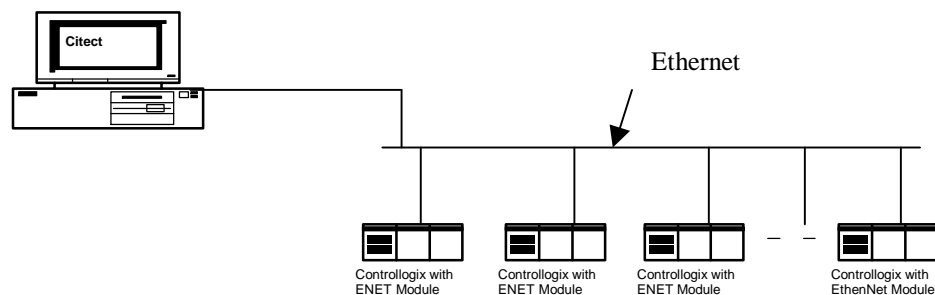
1.	USER INFORMATION	4
1.1	APPLICATION NOTES FOR DEVICE ABLOGIX	4
1.1.1.2	Overview	4
1.1.1.3	Setup guide.....	4
1.1.1.4	Hints, Tips, and Frequently asked questions	7
1.1.1.5	Reference: Required components.....	7
1.1.1.6	Reference: Communications forms.....	7
1.1.1.7	Variable Tags Form.....	9
1.1.1.8	Reference: Wiring diagrams	16
1.1.1.9	Include Functionality support	16
1.1.1.10	16
1.1.1.11	Citect Hot_Standby.....	16
1.1.12.12	Citect Redundancy and driver initial parameter.....	17
1.1.12.13	Reference: Data types.....	17
1.1.12.14	Performance Considerations	17
2.	DRIVER REFERENCE	21
2.1.12.1	Description.....	21
2.1.12.2	Driver generated error codes.....	21
2.1.12.3	Parameters, options, and settings	22
2.1.12.4	Advanced.....	23

1. User information

1.1 Application notes for Device ABLOGIX

	Detail
Manufacturer	Rockwell Automation
Device name	Controllogix5550
Communications method	Ethernet via 1756-ENET / ENBT series Modules

1.1.1.21.1.12.2 Overview



The ENET is the 10Megabit TCP/IP module, while the ENBT is the 100Megabit TCP/IP module. Where ENET appears in this document, ENBT can be substituted.

1.1.1.31.1.12.3 Setup guide

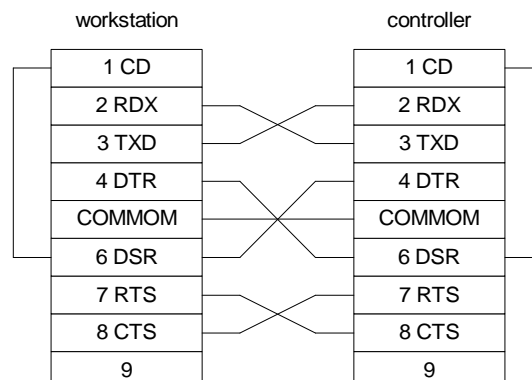
Manually configuring the 1756-ENET/ENBT Module with an IP address.

1. Slot the module in a chassis with a controller.
2. Obtain a 1756-CP3 serial cable.

If you make your own serial cable:

?? Limit the length to 15.2m

?? With the connectors as follows:

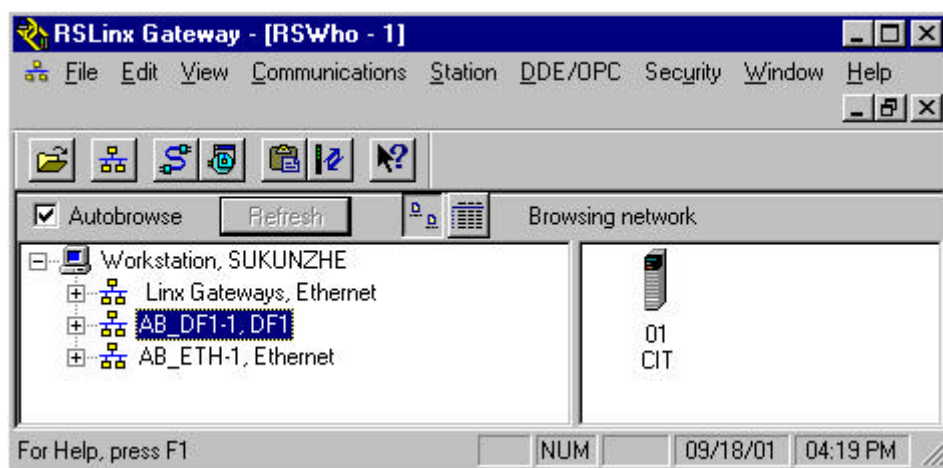


?? Attach the shield to both connectors.

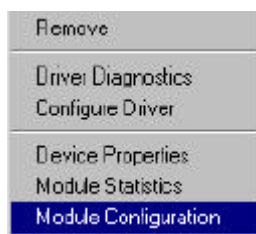
3. Connect the cable to the controller and to your workstation.
4. Start the RSLinxTM software.
5. From the *Communications* menu, select *Configure Drivers*.
6. Click *Add New*.
7. Click OK to accept the default name for the driver.
8. From the *Com Port* drop-down list, select the serial port (on the workstation) that the cable is connected to.
9. From the *Device* drop-down list, select *logix 5550-Serial Port*.
10. Click *Auto-Configure*.



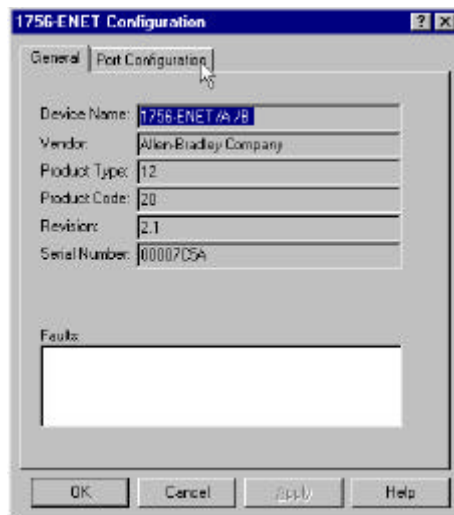
11. If the dialog box display the following message: *Auto Configuration Successful!*
12. Click OK, otherwise go to step 8, and verify that you selected the correct Com Port.
13. Click *Close*.
14. The **RSWho** window will now be open.



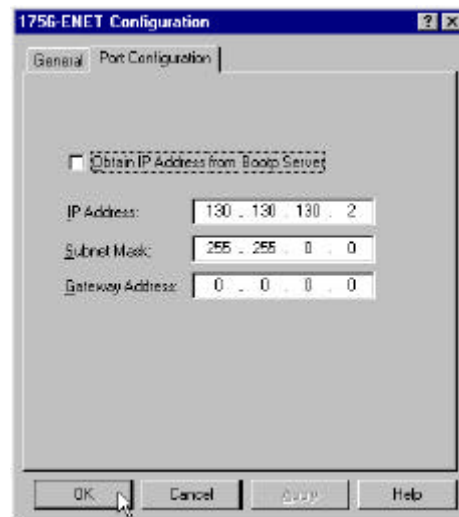
15. Select the appropriate driver (e.g., AB_KT-1 for Data Highway Plus, AB_KTC-1 for ControlNet, or AB_DF1-1 for the Logix 5550's serial port).
16. Expand the driver tree through the backplane of the chassis containing the 1756-ENET/B module.
17. Right click on the module. The following pop-up menu will appear:



18. Select **Module Configuration**.
19. The **1756-ENET Configuration** window will open.



20. Select the **Port Configuration** tab.



21. Uncheck the **Obtain IP Address from BootP Server** box.
22. Enter the desired **IP Address**, **Subnet Mask**, and **Gateway Address**. The values we used for one of the ENET/B modules in the example applications are shown above.
23. Click on **OK**.

1.1.1.41.1.12.4 Hints, Tips, and Frequently asked questions

None

1.1.1.51.1.12.5 Reference: Required components

Software:

- ?? The RSLogix application for configuring/obtaining the Ethernet address of the PLC
- ?? The RSLogix5000 application for programming the PLC
- ?? If using WindowsNT, admin privileges will be required
- ?? CitectSCADA/HMI v5.40 or later required (Note: Included projects not supported in v5.40)

Hardware:

- ?? 1756-ENET ControlLogix Gateway Ethernet module (ENBT module currently not supported)

1.1.1.61.1.12.6 Reference: Communications forms

Boards form

Field	Default	Allowable values
Board Name	This field is user defined, and is not used by the driver.	
Board Type	TCPIP	
Address	0	
I/O Port	BLANK	
Interrupt	BLANK	
Special Opt	BLANK	
Comment	This field is user defined and is not used by the driver.	

Ports form

Field	Default	Allowable values
Port Name	This field is user defined and is not used by the driver.	
Port number	Must be unique, but is not used by the driver	
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	BLANK	
Data bits	BLANK	
Stop bits	BLANK	
Parity	BLANK	
Special Opt	PLC's IP address*	
Comment	This field is user defined and is not used by the driver.	

* The IP address must use the following format: **-Ia**

Where: **a** is the destination IP address in standard Internet dot format

Example: **-I280.9.21.60**

Note: The IP destination Port number for this protocol defaults to 44818. This is the TCP port number reserved with the Internet Assigned Numbers Authority (IANA) for use by the encapsulation protocol and should not be changed.

Note: If you have two PLCs in the same rack, you must create two ports with the same IP address and Port Number and two I/O Device records, one for each processor.

I/O Devices form

Field	Default	Allowable values
Name	This field is user defined, and is not used by the driver.	
Number	Must be unique, but is not used by the driver.	
Address	The slot number of the Logix5550 processor*	
Protocol	ABLOGIX**	
Port name	Refers to the port previously defined in 'ports' form.	
Comment	This field is user defined and is not used by the driver.	

* This field is the slot number of the backplane where the Logix processor is placed.
eg. If the logix5550 processor lies in slot number 2, use 2 as the value.

** For: CitectSCADA v5.41 (or later) – ABLOGIX
CitectSCADA v5.40 – ABLOGIX1

Note: This driver is One-Channel-One-Unit, so you should define one port and one I/O Device for every ControlLogix processor. Multiple Units per Port (i.e. via a gateway) are not supported in this version of the driver.

The IODevice name used should be the same for the standby and active units.

1.1.1.71.1.12.7 Variable Tags Form

1.1.6.1 Using single tag to access the single element of the variable

?? *Single variables*

ABLOGIX driver is tag based, so the name of the variable in the ControlLogix processor is the only way to access the data.

Note: The maximum address length in Citect is 64 characters, so tag names in RSLogix must be configured to be shorter than this.

Example 1 -

ControlLogix using integer tag name “parts”, the address would be: parts

Example 2 -

ControlLogix using DINT tag name”LONG1”, the address would be: LONG1

?? *One dimensional array*

The address is same as for a single tag, but with [] to reference the element number.

Example -

To read the 5th element of an array of INTs named “IntArray1” (see following picture) the address would be: IntArray1[4]

Note: All arrays start at element 0. So [4] refers to element number 5.

The screenshot shows a software window titled "Variable Tags [ABLOGIX]". It contains several input fields and buttons. The "Variable Tag Name" field is filled with "IntArray4". The "Data Type" dropdown is set to "INT". The "I/O Device Name" dropdown is set to "ABLOGIX". The "Address" field is filled with "IntArray1[4]". There are also fields for "Raw Zero Scale", "Raw Full Scale", "Eng Zero Scale", "Eng Full Scale", "Eng Units", and "Format", all of which are currently empty. A "Comment" field is at the bottom. Below the fields are four buttons: "Add", "Replace", "Delete", and "Help". At the very bottom, it says "Record: 6" and "Linked: No".

?? *Two dimensional array*

Example –

Two dimensional array named “counts” of size 10 x 10, one valid element address could be: counts[5,0] (Not: counts[5][0])

?? *Three dimensional array*

Example –

Single integer of a three dimensional array named “profile”, its address could be: profile[2,5,256]

?? *Element of the structure (use the full stop symbol “.”)*

Between Element and the structure, use “.” The Citect data type is the same as the element being read.

Example 1 –

An accumulated value of a timer named “dwell3”, its address is: dwell3.ACC

Example 2 –

For the present value of a one element in a two dimensional array of counters named “counts”, its address would be: counts[5,0].PRE

Example 3 –

For the set point value of one element in loop1 of a customer user data type named “flowmeter6”, its address would be: flowmeter6.loop1.SP

?? *To access the PLC’s SINT type tag*

When access the PLC’s SINT tag, you must add a “\$” symbol to the beginning of the address. The variable tag DATA TYPE should be INT.

Example 1 –

If you read or write the PLC’s SINT tag named “sint1”, your address should be: **\$sint1**

?? *To access a single bit of a variable (use the symbol “/”)*

The CIP protocol does not directly support reading bits in words, therefore to access these, you must define a tag with the same data type as the tag that you are reading the work from and use a special syntax.

Notes:

- ?? Bit number is in the range 0 – 31. As bits start from 0, bit /N is in fact the N+1th bit.
- ?? You cannot access the bit of the type REAL.
- ?? Exception: If you want to read or write bits from a LONG, these should be defined in Citect as DIGITAL

Examples:

1. A single bit of a variable

Two methods are available to read and write bits for variables.

A. If you want to read or write a single bit of a variable, such as type LONG, INT or SINT. You should form its address in this way: **tagname/bit number**

Eg. To read the 5th bit of the LONG (DINT) variable named “long1”, your address is: long1/4, Data Type is BYTE, INT or LONG.

Eg. To write the 16th bit of the INT variable named “INT1”, you should form the address: INT1/15. Data Type is INT.

Eg. To access the 14th bit of the INT array named “intarray[4]”, you should form the address: intarray[4]/13

Eg. To access the 27th bit of one DINT element in loop1 of a custom user data type named “flowmeter6”, its address would be: flowmeter6.loop1.SP/26
Data type is LONG.

B. If the Data Type is DIGITAL, then the underlying variable type needs to be known by the driver. To do this the “!B16” and “!B32” suffixes must be used. 8 Bit is the default. E.g.

To access bit 5 of int0 as a DIGITAL, “int0!B16/5”. This tells the driver that the variable is in fact an INT

The default data type assumed by the bit operator for NON array tags is BYTE, i.e. 8 bits.

Note: It is possible to reference one element of an array, as a single item, e.g. Address: int_array2{4}/13 This uses the 5th word of the array and accesses bit 13.

2. A single bit of a BOOL (bit) array

As with other tags we need to flag the internal BOOL structure. The default bit size is 32.

In general, if you want to access the single element of a BOOL array, you should form the address in this way: **tagname[bit number]**

The variable tag entry would have :

Tag Name: boolarray2 Address : boolArrayPID!A[1024] Type: DIGITAL
In the PLC “boolArrayPID” is known as the array ID.

Eg. To access bit 3 of BOOL array named “boolarray2”, its tag name is: boolarray2[3]

Eg. To access the bit 890 of BOOL array named “boolarray2”, its tag name is is: boolarray2[890]

To access bool array structures, the format needs to be :
Address: itemArray!BxA[512]

Where x can be 8 , 16, 32 for a BYTE, INTEGER or LONG array.

For consistency, !B32A[x] can be used to flag a 32 bit array (the default type), e.g.

Address: longArray!A[64] or longArray!B32A[64] are equivalent.

3. A single bit of the SINT tag

Eg. To access the 6th bit of the sint1 the address would be: **\$sint1/6**. The variable tag DATA TYPE should be INT

1.1.6.2 Using Arrays

Because this driver is tag based, arrays must be defined differently from most other Citect drivers. In order to specify a variable tag as an array, the start address must be defined and if you want to use Cicode functions TagRead() or TagWrite() then the address should be embraced with the “{}”, not the “[]” and then followed by a delimiter “!A”¹ and the length. If you use the “[]” to embrace the first address, Citect will consider the first “[]” as the array bounds, so the Cicode function TagRead() and TagWrite() will not work.

Example –

In this example, 10 register addresses are referred to by the variable tag Turbine_Speed, starting at address 5 in “IntArray”:

Variable Tag Name:	Turbine_Speed
Variable Tag Type:	INT
Address:	IntArray1{5}!A[10]

When used in Citect, each element of an array is then referred to by an index. Individual variables (from the array) can be extracted by specifying the tag name and index (starting at 0).

Example –

To use the third variable of the Citect array in the above example (Turbine_Speed) on a graphics page, use the following syntax:

Variable Tag Name: Turbine_Speed[2]

Note1: BOOL (bit) arrays in the PLC are stored as 32bit data. Normally the start address {start} would be on an address boundary, eg. 0, 32, 64, 96, 128. However it is allowed to have any value. Thus a bool array like “bool{5}!A[4]” would return 4 bits from the 6th bit.

Note2: When using arrays to write to the PLC’s of two or more dimensions, you should ensure that the element number does not exceed the dimension of the array in the PLC.

Example –

For a two dimension integer array named “testarray” in the PLC, which is 4 x 5 dimension, if you use an array named “citect_array” to write to this array and you start from testarray[1,0] you can refer to five consecutive addresses. If you start from testarray[1,1] you can refer to only four consecutive address. So your array’s address may be like this: testarray{1,1}!A[4].

Note3: All the address above should not contain white space character.

¹ For all BOOL array cases where !A[x] appears, !BnA[x] can be used where n=8, 16 or 32.

Note4: If you want to refer to array from the array's beginning, your array's address must still specify a start element of 0 –

Eg. arrayname{0}!A[CountNumber]

Note5: If you want to use array to access the PLC's SINT array tag, the array's format is same as the other array, but don't forget to add a "\$" to the beginning of the address, just as explained below: **\$sintarray{0}!A[10]**. Tag's DATA TPYE should be INT.

Note6: You cannot directly access bits from words in Citect array tags. You have to use Cicode bit operators to mask out the bit required.

1.1.6.3 Using string data type

String data type is new in v8.02 of the RSLogix 5000 programming software and requires RSLinx v2.30.00 or later.

The Citect tag address is just the plc string tag name

Eg. For a string tag named "TestString" in the plc, the Citect tag address would be: TestString. Data type is STRING.

String Lengths: In the PLC, the string data type is a predefined structure, with two elements: LEN and DATA[]. When you write to string, you can see, these two elements are updated with the new value at same time. This is different to that the AB's OPC Test Client utility, which needs to define two tags to refer to these two elements. The driver does some additional work, to change the tag address to stringtagname.DATA[0] and then write the string to the DATA[] element, then update the stringtagname.LEN with the new value. The user will not aware of these changes, they just see that the string have written to the plc, and the length of string is updated.

Note1: Maximum string length in ControlLogix is 82 bytes (656bits). Citect strings are 256 bytes.

Note2: You can directly use "stringname.DATA[0]" as string address, but in this case string LEN element will not updated automatically if the length changes.

1.1.6.4 The size of array

Maximum arrays sizes that can be accessed by the driver are as follows –

DIGITAL (bool) array:	2048
SINT array:	128
INT array:	128
LONG (dint) array:	64
REAL array:	64

Note: Arrays of strings are not supported.

Citect	To access		Address Format	Example
Single Tag	PLC's Single tag	Single integer tag named "parts"	Tagname	parts
		7 th element of an array of REALs named "Setpoints"		Setpoints{6}
	PLC's Single element of Array tag	Single integer[2,5,257] of a three dimensional array named "Profile"	Arrayname[dimension number1,dimension number2,]	Profile[2,5,257]
		Two dimensional array of counters[5,0] named "Counts"		Counts[5,0]
		The PLC's SINT type tag named "sint1"	\$tagname	\$sint1
	PLC's Element of the structure	Accumulated value of a timer named "Dwell3"	Structurename.elementname	Dwell3.ACC
		Preset value of a two dimensional array counters[5,0] named "Counts"		Counts[5,0].PRE
	PLC's Single bit of the tag	5 th bit of the LONG variable named "Long1"	Tagname/bit number	Long1/4
		4 th element of an array of BOOLs named "BoolArray1"	Name{bit number}	BoolArray1[3]
		7 th bit of the SINT variable named "sint1"	\$tagname/bitnumber	\$sint1/6
		5 th bit of the 32 bit based DIGITAL variable named "Long1"	Tagname/bit number	Long1!B32/4
		7 th bit of the 16 bit based DIGITAL variable named "Int1"	Tagname/bitnumber	Int1!B16/6
Array tag	PLC's one dimension array	Four integers of the PLC's integer type array "IntArray1" starting from the address 23	Arrayname!A[count]	IntArray1{23}!A[4]
	PLC's two, three dimension array	Five longs of the PLC's long type three dimension array "LongArray1" starting from the address [2,3,5]	Arrayname!A[count]	LongArray1{2,3,5}!A[5]

	PLC's BOOL array	Ten elements of the PLC's 32 bit based (the default) Boolean array "BoolArray1" starting from bit 64	Arrayname!A[count]	BoolArray1{64}!A[10] optionally BoolArray1{64}!B32A[10]
	PLC's BOOL array (16 bit)	Ten elements of the PLC's 16 bit based Boolean array. "IntBoolArray" startin g from bit 5.	Arrayname!B16A[count]	IntBoolArray{5}!B16A[10]
	PLC's BOOL array (8 bit)	Ten elements of the PLC's 8 bit based Boolean array. "ByteBoolArray" starti ng from bit 9.	Arrayname!B8A[count]	ByteBoolArray{9}!B8A[10]
	PLC's SINT array	Ten SINTs of the PLC'S SINT type array named" sintarray1" starting from the address[0]	\$Firstaddress!A[count]	\$sintarray1{0}!A[10]
String	string	PLC's string named "teststring"	stringname	teststring

1.1.1.81.1.12.8 Reference: Wiring diagrams

None.

1.1.1.91.1.12.9 Include Functionality support

Include Functionality for tag based driver is the new feature for CitectSCADA v5.41 and later, when a project is compiled, an OID (Object ID) will be generated for each tag and sorted in the Variable.dbf

1.1.1.101.1.12.10

In order to use this functionality, one of citect.ini parameter needs to be set. You should add [ABLOGIX]UseIncludeProjects=1 parameter into the citect.ini file.

1.1.1.111.1.12.11 Citect Hot Standby

If enabled, using the [ABLOGIX>StatusTag<n> parameter, the driver will check at startup and every watchtime (see driver parameters) a user specified variable in the PLC. You may also specify the [IOSERVER]WatchDogPrimary parameter in Citect.ini if you want checks to the online unit.

If NO StatusTag<n> entry exists then the Status checks by the driver will always return TRUE. You must specify a StatusTag<n> setting if you wish to check that your PLC is still online.

[IOSERVER]

WatchDogPrimary=1.

[ABLOGIX] or [ABLOGIX1]

StatusTag_<PortName>=Name_of_check_tag

<PortName> is the port name defined in the Citect project Port form.

Do this for each unit. Globally set the following 2 INIs

StatusValue=x

Where x is the value you wish to check

StatusMask=0xXXXXXX

Where XXXXX is the hex mask of the value you wish to check.

To avoid having to specify the correct record, it is possible to reference the tag by name, e.g.

[ABLOGIX]

StatusTag_Port1=WatchDog_1

StatusValue=1

StatusMask=0x1

If you wish to see that this mechanism is working, add to Citect.ini

[ABLOGIX]
 ExtraTraceLevel=4
 Debugstr=* error

To see a log message of the status check.

This redundancy is independent of ControlLogix system redundancy. This driver is only for use where the CPU(s) are in the same rack as the Ethernet card. ControlLogix system redundancy requires the use of ControlNet communications to each redundant rack.

1.1.1.121.1.12.12 Citect Redundancy and driver initial parameter

You can also use the Hot-Standby parameters to define any tag in the PLC and not set [IOSERVER]WatchDogPrimary parameter (default 0).

Note: This redundancy is independent of ControlLogix system redundancy. This driver is only for use where the CPU(s) are in the same rack as the Ethernet card. ControlLogix system redundancy requires the use of ControlNet communications to each redundant rack.

1.1.1.131.1.12.13 Reference: Data types

IO Device Type	Citect Tag Address Format	Citect Data Types	Description
BOOL	Controllogix tag description	DIGITAL	R/W
SINT	Controllogix tag description	INTEGER	
INT	Controllogix tag description	INTEGER	
DINT	Controllogix tag description	LONG	
BIT ARRAY	Controllogix tag description	DIGITAL array	
REAL	Controllogix tag description	REAL	
STRING	Controllogix tag description	STRING	R/W

1.1.1.141.1.12.14 Performance Considerations

This section applies for driver versions 2.00.xx.xx on. Items here can be experimented with to improve performance. Each time the TAG database has been changed, a fresh “Pack” and “Compile” should be done in Citect Explorer.

Use of the Citect kernel probe window and other traces has a big impact on performance. When doing measurements these should be turned off and the kernel “driver stats” used as a guide.

1.1.1.14.1 Tag Blocking

Blocking means that one request maybe made to the driver for say 5 TAGs on a page instead of 5 separate requests. This greatly improves response times. The ABLOGIX driver template currently allows 32 items to be blocked together.

From version v2.00.x.x of the driver, compiler blocking of certain tags are supported. These are :

- INTs
- REALs
- LONGs

Blocking of TAGs is problematic as each TAG doesn't generally have a relationship to another tag and if one TAG has errors (from the PLC), this can affect other TAGs in the block. However if we treat the blocking based on the record number (not the TAG name) progress can be made.

Compiler blocking has rules; the unit type, IODevice and raw data type need to be the same. Then the number of blocked items has to be less than the maximum number of bits for the driver and / or 32 items. Blocked TAGs are arranged based on alarm entries, trends and display pages.

Blocking is done based on the TAG name (NOT PLC address name) and the record number of the tag. Thus no special needs exist in the TAG ADDRESS field or PLC to support blocking.


Rules change between ABLOGIX & ABLOGIX1 . The OID supported version will only block continuos items in the TAG database, the non OID version (ABLOGIX1) can block across data type gaps, e.g.

Record# DataType

1. INT
2. INT
3. REAL
4. REAL
5. INT
6. INT
7. STRING
8. INT

For ABLOGIX, integers will be blocked as 3 reads, [2@1](#) (2 items AT address 1), [2@5](#), [1@8](#), for ABLOGIX1 only one request will be done, [8@1](#).

For 5.41+ users who do NOT need the include project "mix and match" feature of 5.41, they maybe better of using "ABLOGIX1" (5.41+ can do this, however, 5.40 users cannot use "ABLOGIX") .

 Maximum blocking is achieved by ensuring alarm/trend items (if arrays are not being used) are next to each other in the TAG database. Specific TAGs for screens should be grouped based on what screen they are in. It is suggested that this work is only done if the default performance is lacking.

1.1.1.14.2 Multiple Requests

v2.00.xx.xx on supports multiple outstanding requests to the PLC. This means instead of having to wait for a reply from the PLC for one request, many requests can be asked at the same time.

The requests used is up to the MAXPENDING value for the driver. Thus the user can adjust this themselves. The default value is 15.

1.1.1.14.3 Template

The driver DBF files are shown below. The “%s32” encodes the number items blocked together. Citect testing has only been with 32. The max. value possible for REALs or LONGs is 64 and for INTs, 128.

1.1.1.14.4 Error Handling

To find the specific tag in error in a block (in a non array context), the debuglog level should be at least 2, e.g.

```
[ablogix]
extradebuglevel=2
debugstr=* all ! or error
```

A message will look like :

```
***Ablogix BadTagReport** Error 0x0201 Tag: IntA Address: IntAinPlc" and
"-- Blocked Read for address XXX and item count YY, offset ZZ"
```

This can be used during commissioning to resolve any TAG database Address name to PLC name mismatches.

For TAG based drivers, Citect recommend that the TAGReadKit project be downloaded and used to verify valid driver TAGs. This can be found at <http://www.citect.com/mycitect/downloads/toolbox?&cid=24> .

1.1.1.14.5 Super Genies

It should be noted that supergenies use runtime blocking which is done in the IOServer. Thus extensive use of supergenies can load the Display Client to IOServer connection. Supergenies should not be used for complex pages or for high scan rates.

ABLOGIX1.dbf

TEMPLATE	UNIT_TYPE	RAW_TYPE	BIT_WIDTH	LOW	HIGH	COMMENT
%K%J%!	0xA0000000	7	664	0	0	string
%K%J%!!A	0x90000000	2	32	0	0	Real array
%K%J%!!A	0x80000000	4	32	0	0	Long array
%K%J%!!A	0x70000000	1	16	0	0	Integer array
%K%J%!!A	0x60000000	0	32	0	0	Bit array, 4 bytes
%K%J%!!B32A	0x60000000	0	32	0	0	Bit array, 4 bytes
%K%J%!!B16A	0x50000000	0	16	0	0	bool array bit, 2 bytes
%K%J%!!B8A	0x40000000	0	8	0	0	bool array bit, 1 bytes
%J%s32%J%!	0x30000000	2	32	0	0	Real, 4 bytes
%J%s32%J%!	0x20000000	4	32	0	0	Long integer, 4 bytes
%J%s32%J%!	0x10000000	1	16	0	0	integer, 2 bytes
%K%J%!	0x00000000	0	8	0	0	Boolean 1byte
%K%J%!!B16	0x00000000	0	16	0	0	Integer by bit
%K%J%!!B32	0x00000000	0	32	0	0	Long by bit

ABLOGIX.DBF (OID SUPPORT, 5.41+)

TEMPLATE	UNIT_TYPE	RAW_TYPE	BIT_WIDTH	LOW	HIGH	COMMENT
%N+%*32%!	0xA0000000	7	664	0	0	string
%N+%*32%R%!!A	0x90000000	2	32	0	0	Real array
%N+%*32%R%!!A	0x80000000	4	32	0	0	Long array
%N+%*32%R%!!A	0x70000000	1	16	0	0	Integer array
%N+%*32%R%!!A	0x60000000	0	32	0	0	Bit array, 4 bytes
%N+%*32%R%!!B32A	0x60000000	0	32	0	0	Bit array, 4 bytes
%N+%*32%R%!!B16A	0x50000000	0	16	0	0	integer, 2 bytes
%N+%*32%R%!!B8A	0x40000000	0	8	0	0	BOOL array bit, 1 bytes
%N+%s32%N+%!	0x30000000	2	32	0	0	Real, 4 bytes
%N+%s32%N+%!	0x20000000	4	32	0	0	Long integer, 4 bytes
%N+%s32%N+%!	0x10000000	1	16	0	0	integer, 2 bytes
%N+%*32%R%!	0x00000000	0	8	0	0	Boolean 1byte

2. Driver reference

	Detail
Driver name	ABLOGIX
Maximum request size	2048 bits

2.1.1.12.1.12.1 Description

The ABLOGIX driver is used to communicate to Allen-Bradley ControlLogix5000 via the Ethernet.

2.1.1.22.1.12.2 Driver generated error codes

Driver Error Code (Hexadecimal)	Mapped to (Generic Error Label)	Meaning of Error Code
0x101	GENERIC_CHANNEL_OFFLINE	Received packet did not correspond to the request. Check IODevice Port "Address" value.
0x103	GENERIC_INVALID_DATA_TYPE	Bad parameter, size >12 or size greater than size of element (RMW)
0x104	GENERIC_INVALID_DATA_TYPE	The IOI could not be deciphered. Either it was not formed correctly or the match tag does not exist.
0x105	GENERIC_INVALID_DATA_TYPE	The particular item referenced (usually instance) could not be found
0x106	GENERIC_INVALID_DATA_TYPE	The amount of data requested would not fit into the response buffer. Partial data transfer has occurred.
0x10A	GENERIC_INVALID_DATA_TYPE	An error has occurred trying to process one of the attributes.
0x110	GENERIC_INVALID_DATA_TYPE	Device state Conflict: keyswitch position.(RMW)
0x113	GENERIC_INVALID_DATA_TYPE	Not enough command data/parameters were supplied in the command to execute the service requested
0x11C	GENERIC_INVALID_DATA_TYPE	An insufficient number

	PE	of attributes were provided compared to the attribute count.
0x126	GENERIC_INVALID_DATA_TY PE	The IOI word length did not match the amount of IOI which was processed.
0x108	GENERIC_INVALID_DATA_TY PE	Route path error, this may cause by the wrong propocessor slot number
0x200	GENERIC_INVALID_DATA	Tag's OID can't be find in the database
0x210	GENERIC_INVALID_DATA	Data length that are to be copyed is negative.
0x220	GENERIC_INVALID_DATA	Tag's bit number defined has overrun.
0x11e	GENERIC_INVALID_DATA	Embedded service error. An embedded service resulted in an error

Popup Boxes

"Ablogix : Error 0xXX on opening 'PortYYY'" – The OpenChannel() function could not create a connection on the TCP/IP port given.

"Can't find user project directory" – Citect.ini's [CTEDIT]RUN= was not set != was not set !!

"Cannot open" – Unable to open the variable.dbf file for the project .

"Can't allocate memory for variable table" – Very unlikely.

"Can't re-allocate memory for variable table" – Very unlikely.

"No records, please check if Citect Editor has TAG table open" – The variable.dbf table was empty of someone else has the file open.

2.1.1.32.1.12.3 Parameters, options, and settings

Standard Parameters

Parameter	Default	Allowable values
Block (bytes)	256	1 to 256
Delay (mS)	0	0 to 50
MaxPending	15	1 to 32 (do not exceed 32) NB: This value is per Port
Polltime (mS)	0	0 to 300 – Do not change
Timeout (mS)	2000	0 to 32000 – Do not set below the max. physical time to send and get a data request back from a unit.
Retry	3	0 to 8 – Set higher if errors can occur in the COMs
WatchTime (Sec)	30	1 to 128 -

Driver Specific Parameters

Parameter	Default	Allowable values	Description
UseIncludeProjects	1	0 or 1	Enable use of variable tags in included projects (see section 1.1.8)
StatusValue	none	(see section 1.1.9)	Enable hot standby functionality (see section 1.1.9)
StatusMask	0xffffffff	(see section 1.1.9)	Enable hot standby functionality (see section 1.1.9)
StatusTag_<PortName>	None	(see section 1.1.9)	Enable hot standby functionality (see section 1.1.9)
ExtraDebugLevel	0	1,2,4,8	Debug output must be enabled, e.g. “debugstr=* error”, normal setting will be 2. Citect support may ask for a 8 for some problems. NB: This is read each unit startup or status check. Thus it is possible to turn traces on and off while the driver is running. A 4 will trace reads of Status tags.
ConnTimeout	15000		TCP connection timeout period

IODevice Form – “Name” – The name used for redundant IODevices MUST be the same as the active unit as driver uses the VARIABLE.DBF file to obtain the TAG information.

The projects TAG database cannot be opened by Citect when the driver runs else a popup box will tell you to close it.

2.1.1.42.1.12.4 Advanced***Driver generated statistics***

Number	Label	Description
0	NoTX	The number of packet sent
1	NoRX	The number of packet receives.
2	NoWER	The number of write error.
3	NoWOK	The number of write success.
4	NoRER	The number of read error.
5	NoROK	The number of read success.
6	NoRBP	The number of received bad packet
7	NoBDC	The number of invalid DCB.

Debug messages explained

The debug messages are from the TraceTx and TraceRx functions.

Request:

Sat Sep 22 12:42:45 2001 01:28:20.365 Transmit A whole Packet Length 110

6F	00	56	00	00	01	02	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	02
00	00	00	00	B2	00	46	00	52	02	20	06	24	01	04	05
38	00	0A	02	20	02	24	01	03	00	08	00	1A	00	26	00
4C	07	91	09	49	6E	74	41	72	72	61	79	31	00	28	04
01	00	4C	04	91	05	4C	4F	4E	47	31	00	01	00	4C	04
91	05	52	45	41	4C	31	00	01	00	01	00	01	00		

Encapsulation
HeaderCIP Multi-
Request Service

Route_Path

Reply:

Sat Sep 22 12:42:45 2001 01:28:20.430 Receive A Whole Packet Length 80

6F	00	38	00	00	01	02	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	02	00
00	00	00	00	B2	00	28	00	8A	00	00	00	03	00	08	00
10	00	1A	00	CC	00	00	00	C3	00	1E	00	CC	00	00	00
C4	00	04	00	00	00	CC	00	00	00	CA	00	7D	FF	F7	42

Encapsulation
Header