Citect for Windows

Driver Specification

Beckhoff ADS

Driver version history

Version	Date	Modified By	Details
1.0.0.0.B1	2013-07-30	Tomas Rook	Main development
1.0.0.0.B2	2013-09-16	Tomas Rook	Debug/Trace implemented
1.0.0.0.B3	2013-09-16	Tomas Rook	VT_U1 added
1.0.0.0.B4	2014-04-30	Tomas Rook	iTotInclude 64->256
1.0.0.0.B5	2014-08-01	Tomas Rook	TCP request sequence update to increase speed
1.0.0.0	2014-10-01	Tomas Rook	Array stuff fixup - release
1.0.0.1	2015-03-09	Tomas Rook	LocalAMSAddress sanity check and Software-Protection implemented
1.0.0.2	2015-03-11	Tomas Rook	Failed symbol lookup trace file and parameter TraceFailedSymbols implemented
1.0.0.3	2015-03-27	Tomas Rook	Redundancy issues fixup (same unit name), changed OID lookup to also include project number to prevent variable mixup (needs new ADS.DBF and recompile). Parameter TraceOptions added.
1.0.0.4	2015-11-03	Tomas Rook	BYTE support, MAX/MIN fail for some datatypes caused zero value fixup (AssignValueToDcb function)
1.0.0.5	2015-11-10	Tomas Rook	StoreAdsSymbolTable added, Array of struct in program header also supported
1.0.0.6	2016-05-11	Tomas Rook	Failed connect attempt (TCP-mode) caused socket to not be deleted, increasing system resource allocation. Fixed answer compare issue for pOIDvsSymbol
1.0.0.7	2016-09-22	Tomas Rook	Write in invalid buffer could occur during GetAdsSymbolTableAndDataTypes when server was heavy loded causing exception error
1.0.0.8	2016-10-11	Tomas Rook	FORCESYMBOLRELOAD added
1.0.0.9	2016-10-12	Tomas Rook	TraceOptions 0x0004 and 0x0008 added, TraceFileLimit parameter implemented, TraceLimit parameter changed to TraceByteLimit
1.0.0.10	2016-10-12	Tomas Rook	IOServerName fail check, more debug and trace
1.0.0.11	2016-10-18	Tomas Rook	LocalNetworkAddress parameter was added
1.0.0.12	2016-10-26	Tomas Rook	More trace options, minor changes of texts, changed back to DBA_MODE_SHARED, since databases could not be opened if ex the Tag form was visible
1.0.0.13	2016-11-07	Tomas Rook	UseSymbolTableFileIfExisting parameter was added
1.0.0.14	2016-11-13	Tomas Rook	StoreSymbolTableFile parameter was added, parameter IOServerName removed, datatype String implemented.
1.0.0.15	2016-11-15	Tomas Rook	Multiplre reads using ADS_SUMUP implemented
1.0.0.16	2016-12-16	Tomas Rook	IgnoreStartupErrors also ignores licenseproblems and also indicates status in logfile
1.0.0.17	2017-01-18	Tomas Rook	UseAMSAddressAsDefaultIP parameter added
1.0.0.18	2017-02-10	Tomas Rook	Uses C:\ as SystemId disk, shows SystemId and DongleId in ADS.LOG

Driver documentation version history

Date	Modified By	Details
2013-08-13	Tomas Rook	Preliminary
2013-08-16	Tomas Rook	Minor updates
2013-08-19	Tomas Rook	Minor updates
2013-08-26	Tomas Rook	Minor updates
2013-09-16	Tomas Rook	Parameter updates
2014-10-01	Tomas Rook	Minor updates
2015-03-11	Tomas Rook	Minor updates
2015-03-27	Tomas Rook	Minor updates
2015-11-10	Tomas Rook	Parameter updates
2016-10-11	Tomas Rook	Minor updates
2016-10-12	Tomas Rook	Parameter updates, comm fail section
2016-10-18	Tomas Rook	Parameter updates
2016-11-01	Tomas Rook	TwinCAT3 info added
2016-11-02	Tomas Rook	Global addressing added
2016-11-08	Tomas Rook	Parameter updates
2016-11-13	Tomas Rook	Parameter updates
2016-11-15	Tomas Rook	Parameter updates
2017-01-18	Tomas Rook	UseAMSAddressAsDefaultIP parameter added

Contents

1. QA

1.1 Introduction

This document follows the development of the new driver. It serves as a functional specification, design specification and test specification.

1.2 Procedure for generating a new driver

The following check list defines the QA steps for generating a new driver. This procedure must be followed for drivers to be integrated into Citect.

	Description	**Person**	**Date**
1	This specification document is written.	TR	2013-08-13
2	Specification reviewed and accepted by R&D department.		n/a
3	Driver coded.	TR	2013-08-07
4	Code and specification reviewed and accepted by R&D department.		n/a
5	Testing with connection project, and performance test.	TR	
6	Driver integrated into Citect source and built.		n/a
7	Documentation is written (HLP or MVB files)		n/a
	At this checkpoint coding is done and the driver is available as a beta.		
8a	Full testing is carried out.	TR	
8b	Performance testing is carried out.	TR	
8c	Specification and documentation updated from testing/performance tests	TR	
	At this checkpoint the testing is complete.		
9a	Review for completeness by developer, tester, documentor and R&D staff		n/a
9b	Add driver to install disks		n/a
9c	Add driver to protocols database		n/a
9d	Support notified of new driver for training purposes		n/a
10	Sales notified of new driver		n/a
	The driver is now finished.		

The hand over of a driver requires that all the above steps are completed and checked off.

2. Target Device(s) and Protocol

2.1 Introduction

This section defines the types of I/O Devices that are targeted by this driver.

2.2 Device Manufacturer

Beckhoff Automation Sverige AB
Stenåldersgatan 2A 15
SE - 213 76 Malmö
Sweden

2.3 Device Definition

All Beckhoff PLC equipment capable of routed ADS communication (TwinCat2 & TwinCAT3 SoftPLC devices)

2.4 Communications Methods

The driver support both direct TCP/IP using ADS and TwinCat ADS DLL.

The ADS protocol (*Automation Device Specification*) is a transport layer within the Beckhoff TwinCAT system. It was developed for data exchange between the different software modules, for instance the communication between the NC and the PLC. The ADS protocol is used on top of TCP/IP. This means that in a networked system, all the data is accessible from any desired point.

2.5 Communications Configuration

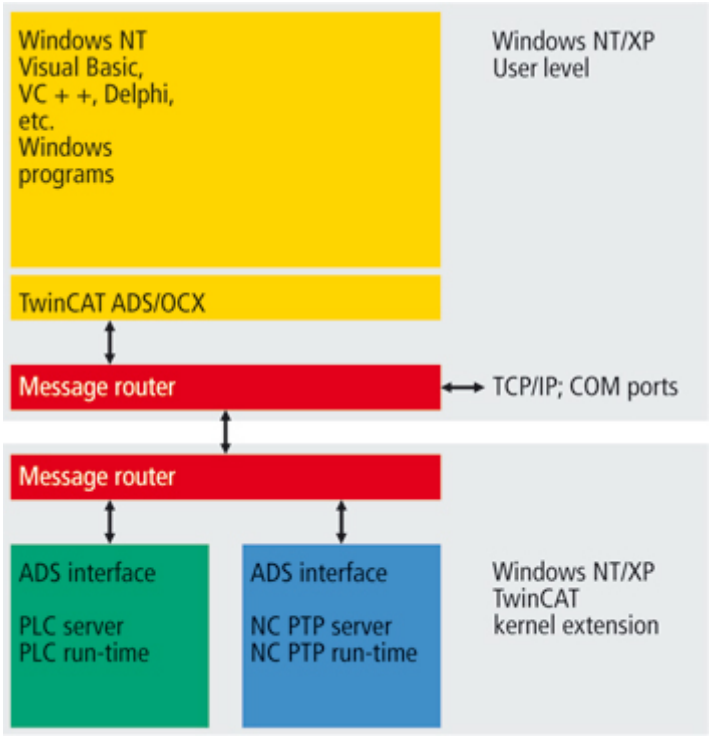
The driver support two diffent communication configurations:

DLL: The driver use *TwinCAT ADS API* (function calls to *TcAdsDll.dll*). The ADS API takes care of all communication to the *Message Router(s).*

When using ADS API the TwinCAT software has to be installed (just like the *TwinCAT Citect Driver*) and *TcAdsDll.dll* has to be accessible by the driver, either by placing it in the *System32* folder (if not already there) or point to the correct directory using the *System -> Environment -> Path variable*.

TCP: The driver connects directly to the *Message Router* using ADS packets wrapped in TCP/IP.

This method does not require any *TwinCAT* software whatsoever to be installed in the PC. The *Message Router* is built into the driver.



2.6 Contacts

Driver Development	TroSoft AB, Tomas Rook	+46 (0)8 532 57262
Beckhoff Automation	Christer Wik	+46 (0)40 680 81 76
Beckhoff Automation	Roger Grönvall	+46 (0)40 680 81 66

3. Protocol Requirements

3.1 Introduction

The driver is a semi-cached request/response driver using OID for variable lookup.

The driver was developed using DDK 5.5 and Citect 6.10 SpA, and has been tested with Citect up to version 7.30. Later OS's will report driver is old and maybe not compatible. The driver will be recompiled for VS2010 when Citect release their new DDK.

3.2 Symbol table information

When the driver DLL has loaded and communication begins it will first of all read the symbol table. During communication, the downloaded symbol table version will be cyclically read. If the version changes or the device is offline the communications is hold until the updated symbol table is retrieved from the device and *Groups/Offsets* are recalculated.

3.3 OID based adressing

The driver support Symbol name addressing only.

The Citect Driver Development Kit (DDK) only supports 2x32-bit addressing methods (*Unit-Type/UnitAddress*), which is very well suited for Beckhoff devices *Group/Offset* based adressing. From the engineer point of view this is however not very suitable, mainly because the *Group/Offset* values are changed if the symbol table and/or program is updated.

Citect compiler is not capable of handling text based addressing methods, hence the driver make use of direct variable database (*Variable.DBF*) address lookups. When the driver DLL is loaded it will scan through all variable tag databases in both the start project and all include projects. The unique OID key for all ADS variables will be stored in memory together with the address field (symbol name). When Citect wants to read some variable, it sends the OID to the driver which look in its variable lists and hopefully finds the variable address (symbol name). This name is used for lookup in the retireved symbol table to find *Group/Offset.*

3.4 Blackbox projects

In some cases this is a method that will not function properly, ex if the variable tag databases does not exist locally (*BlackBox project*). To allow for these configurations the driver need access to the symbol name using some other method using an index. To enable index based addressing, the parameter *AllowBlackBox* should be set to 1. When enabled, the driver DLL will also create a text based cross-reference index (this function is not yet implemented – aug 2013)

3.5 ADS.DBF

Template	*Unit type*	*Raw type*	*Bit width*	*Read/ Write*	*Comment*
DEVICEVERSION	0x10000000	7	1024	R	TwinCAT I/O version
DEVICENAME	0x11000000	7	1024	R	Device name
SYMBOLCOUNT	0x12000000	4	32	R	Number of symbols
SYMBOLSIZE	0x13000000	4	32	R	Symbol table size
DATATYPECOUNT	0x14000000	4	32	R	Number of datatypes
DATATYPESIZE	0x15000000	4	32	R	Datatype table size

FORCESYMBOLRELOAD	0x16000000	0	1	R	Force reload of symbols
%J%R%!	0x37000000	7	1024	R/W	String
%J%R%!	0x30000000	4	32	R/W	Long
%J%R %!	0x31000000	1	16	R/W	Integer
%J%R %!	0x32000000	8	8	R/W	Byte
%J%R %!	0x34000000	0	1	R/W	Digital
%J%R %!	0x38000000	2	32	R/W	Real

The *DEVICE.FORCESYMBOLRELOAD* template could be used to force the driver re-reading the pointlist. If set to 1 it will begin reading, when finished the driver will automatically reset it to 0.

Point description texts are not writable. If changed in device, a new pointlist must be retrieved for the texts to be updated.

3.6 Supported variables/parameters

All variables of type digital, integer (16-bit), real (IEEE 32-bit), long (32-bit), string and byte (8-bit) accessible using ADS (*BIT8 SINT, USINT, INT, UINT, DINT, UDINT, FLOAT, STRING*).

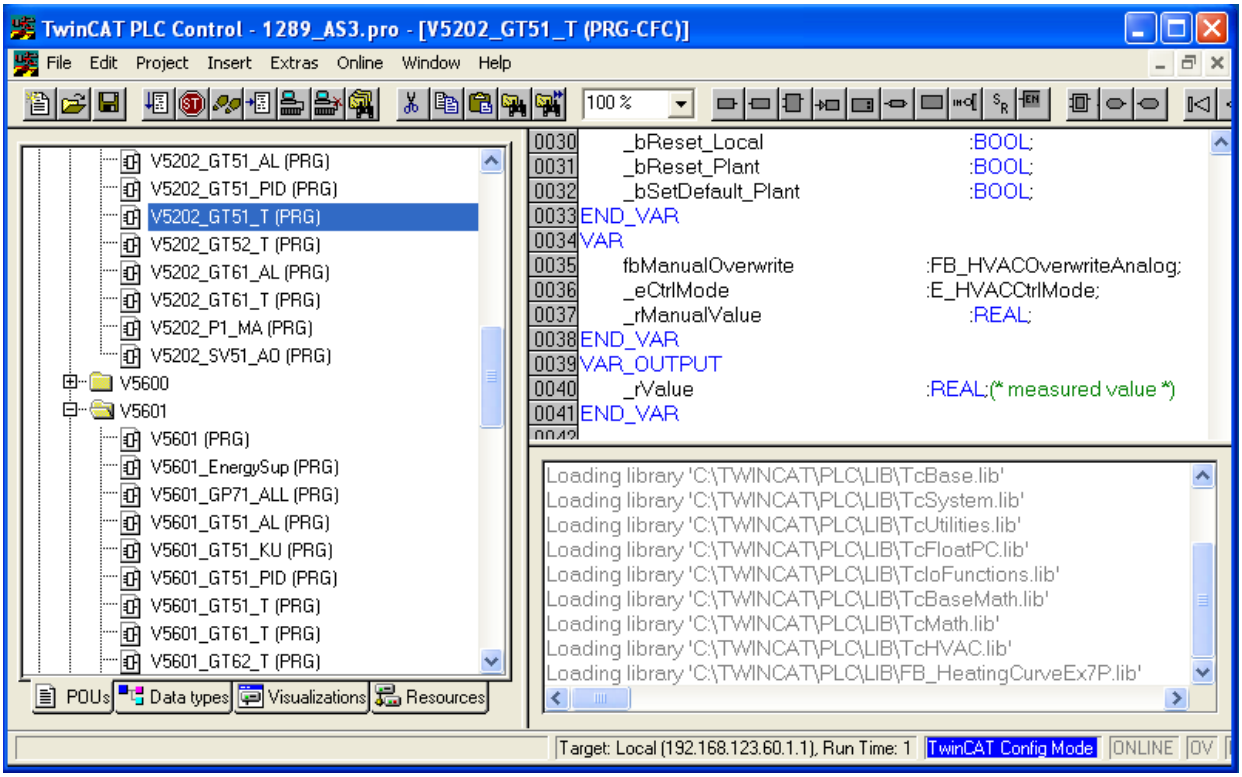
The driver will try to cast variables if possible, ex if *V5202_GT51_T._iRawValue* is defined as a *Digital* even though it is a *Long* in ADS it will return *FALSE* when the value is zero (0) and *TRUE* otherwise.

The driver will also limit variable values, ex if you define read an *UDINT* with the value *75123* from the PLC using a Citect variable defined as *INT* it will be read as *32767* (max for signed int's)

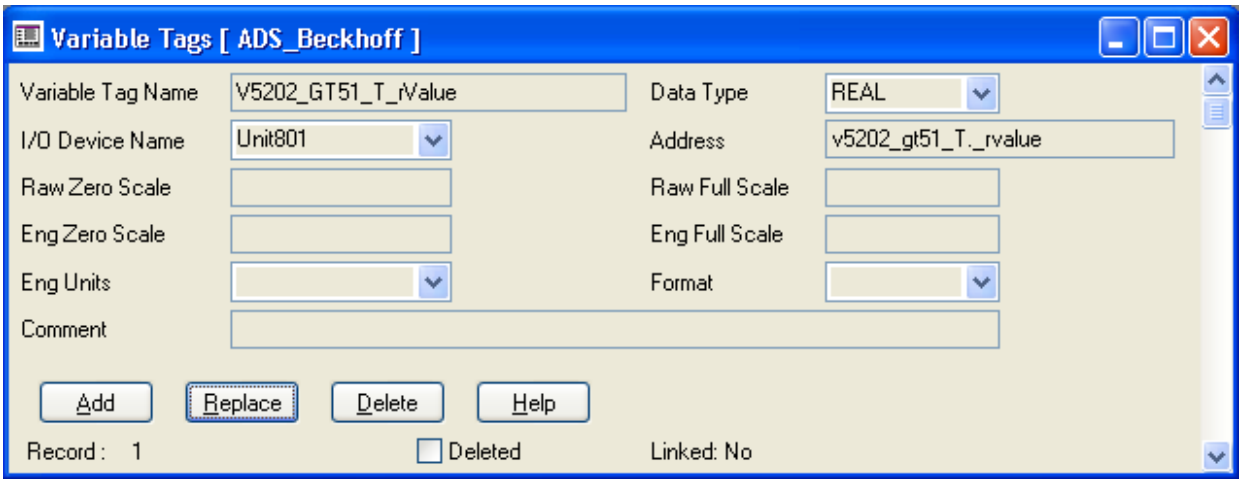
There is no support for other variable type, ex *TIME*.

3.7 Address examples (Symbol name (OID) lookup) TwinCAT 2

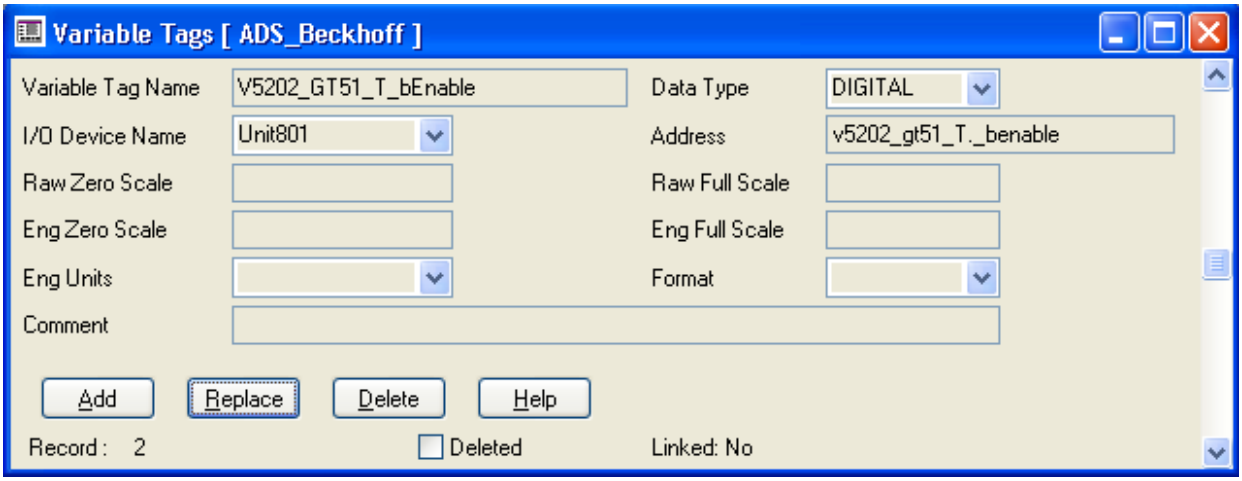
Example from TwinCat 2:



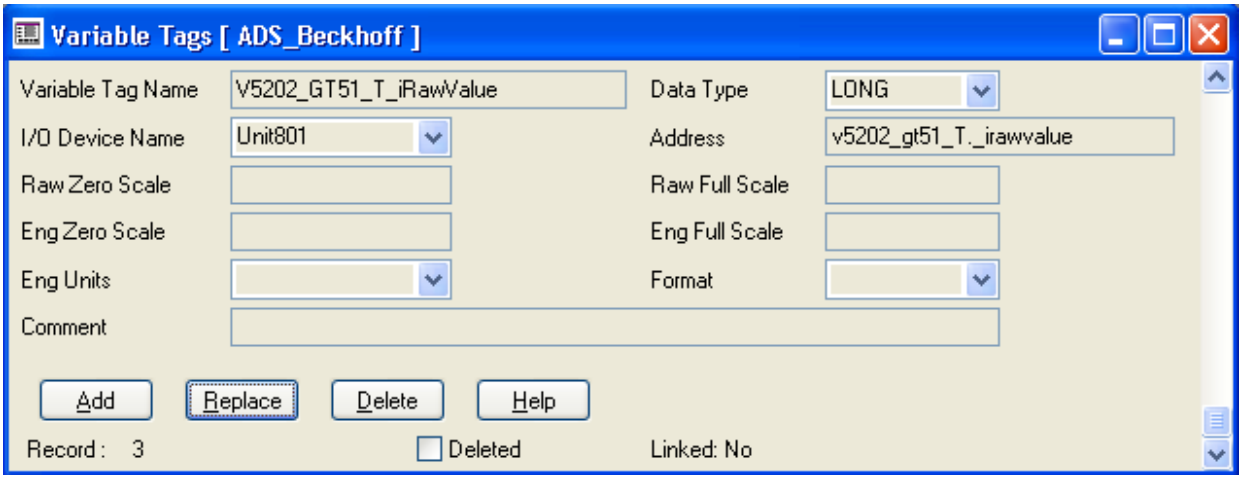
If *rValue* is to be accesed it would have the following Citect addressing



The tag is of *Data Type Real and* using OID lookup. The *Address* field contains the symbol address.



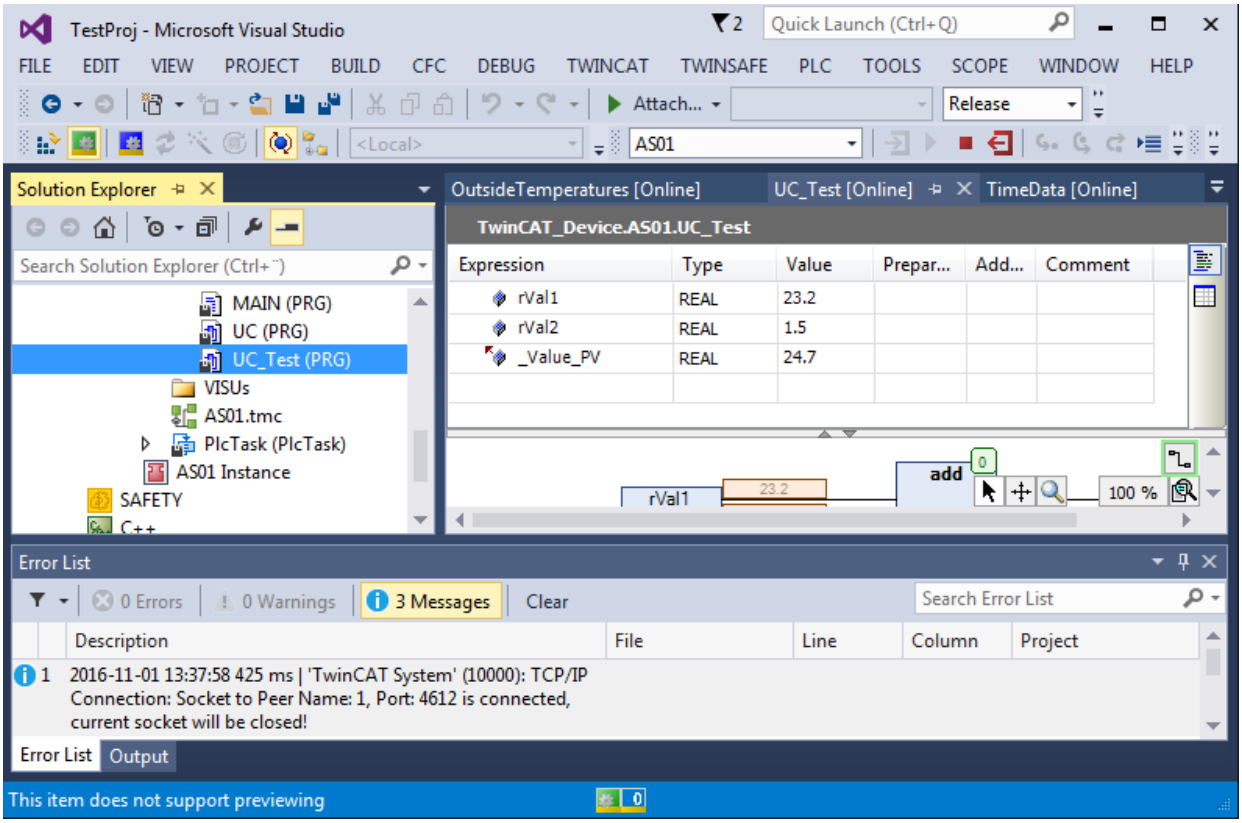
Typical tag of *Data Type Digital* using OID lookup. The *Address* field contains the symbol address.



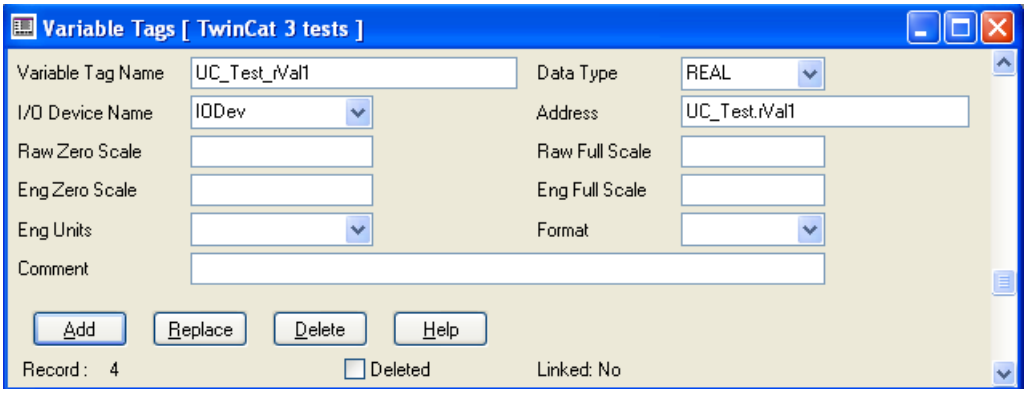
Typical tag of *Data Type Long* using OID lookup. The *Address* field contains the symbol address.

3.8 Address examples (Symbol name (OID) lookup) TwinCAT 3

Example from TwinCat 3:



If *rVal1* is to be accesed it would have the following Citect addressing



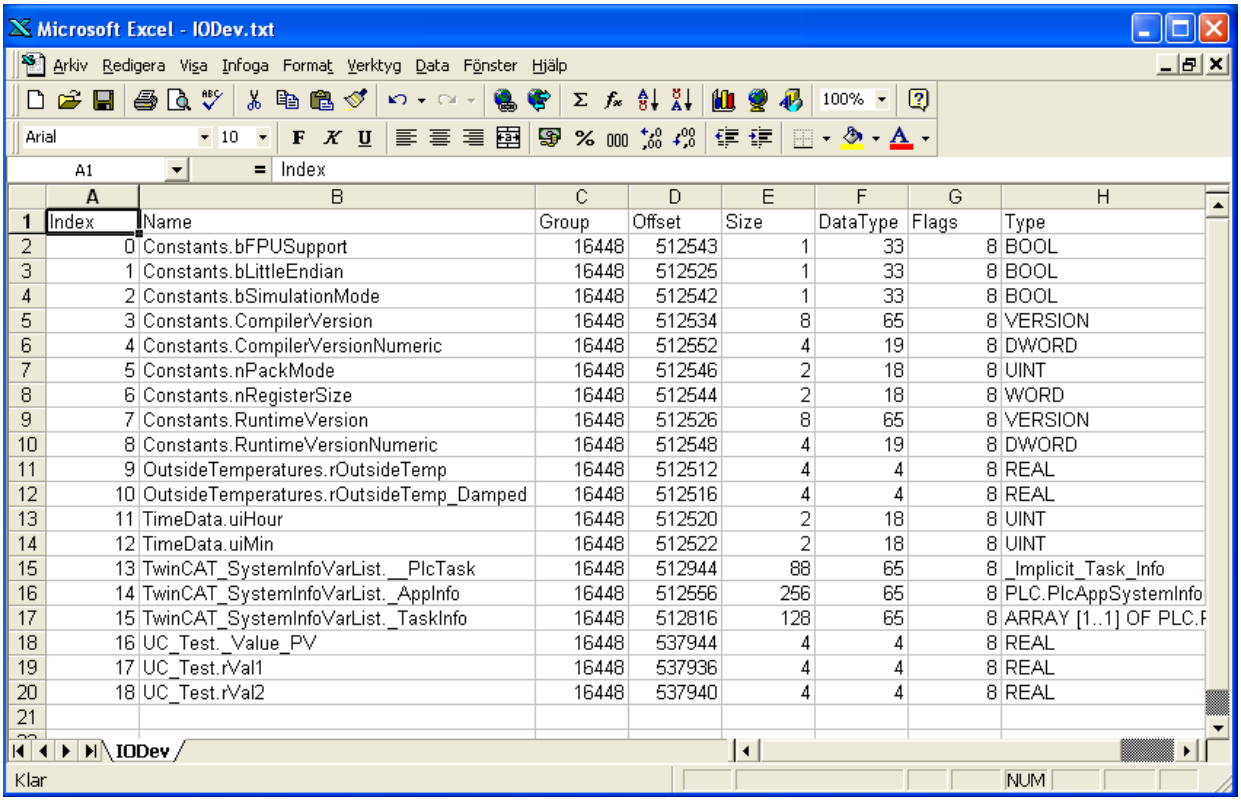
3.9 Use Symbol Table text file to find TwinCAT 2 and TwinCAT 3 addresses

If it is difficult to find addresses, use the Citect.ini parameter *[ADS]StoreAdsSymbolTable* (see *Parameters and INI-options*).

The driver will store the symbol table to a text file, ex above I/O device symbol table would be stored *to [Data]:\ADS\IODev.txt*.

The [ADS]StoreAdsSymbolTable should not be used in production, because it will create the file each time the symbol table is updated.

Example of contents of a symbol table text file



Microsoft Excel - IODev.txt

Arkiv Redigera Visa Infoga Format Verktyg Data Fönster Hjälp

Arial 10 100%

A1 = Index

	A	B	C	D	E	F	G	H
1	Index	Name	Group	Offset	Size	DataType	Flags	Type
2	0	Constants.bFPUSupport	16448	512543	1	33	8	BOOL
3	1	Constants.bLittleEndian	16448	512525	1	33	8	BOOL
4	2	Constants.bSimulationMode	16448	512542	1	33	8	BOOL
5	3	Constants.CompilerVersion	16448	512534	8	65	8	VERSION
6	4	Constants.CompilerVersionNumeric	16448	512552	4	19	8	DWORD
7	5	Constants.nPackMode	16448	512546	2	18	8	UINT
8	6	Constants.nRegisterSize	16448	512544	2	18	8	WORD
9	7	Constants.RuntimeVersion	16448	512526	8	65	8	VERSION
10	8	Constants.RuntimeVersionNumeric	16448	512548	4	19	8	DWORD
11	9	OutsideTemperatures.rOutsideTemp	16448	512512	4	4	8	REAL
12	10	OutsideTemperatures.rOutsideTemp_Damped	16448	512516	4	4	8	REAL
13	11	TimeData.uiHour	16448	512520	2	18	8	UINT
14	12	TimeData.uiMin	16448	512522	2	18	8	UINT
15	13	TwinCAT_SystemInfoVarList.__PlcTask	16448	512944	88	65	8	_Implicit_Task_Info
16	14	TwinCAT_SystemInfoVarList._AppInfo	16448	512556	256	65	8	PLC.PlcAppSystemInfo
17	15	TwinCAT_SystemInfoVarList._TaskInfo	16448	512816	128	65	8	ARRAY [1..1] OF PLC.F
18	16	UC_Test._Value_PV	16448	537944	4	4	8	REAL
19	17	UC_Test.rVal1	16448	537936	4	4	8	REAL
20	18	UC_Test.rVal2	16448	537940	4	4	8	REAL
21								

IODev

Klar NUM

Here the *'Name'* field is the complete address to be used in the Citect tag form.

3.10 Global variable addressing difference TwinCAT 2 / TwinCAT 3

When setting up global addresses in TwinCAT 2, the symbol adress should be preceeded by a period sign '.', ex address for global memory variable *rOutside_T* in *OutsideTemperature:*

.rOutside_T

In TwinCAT 3 the global address must have the complete hierarchical name, ex address for global memory variable *rOutside_T* in *OutsideTemperature:*

OutsideTemperature.rOutside_T

4. User Interface

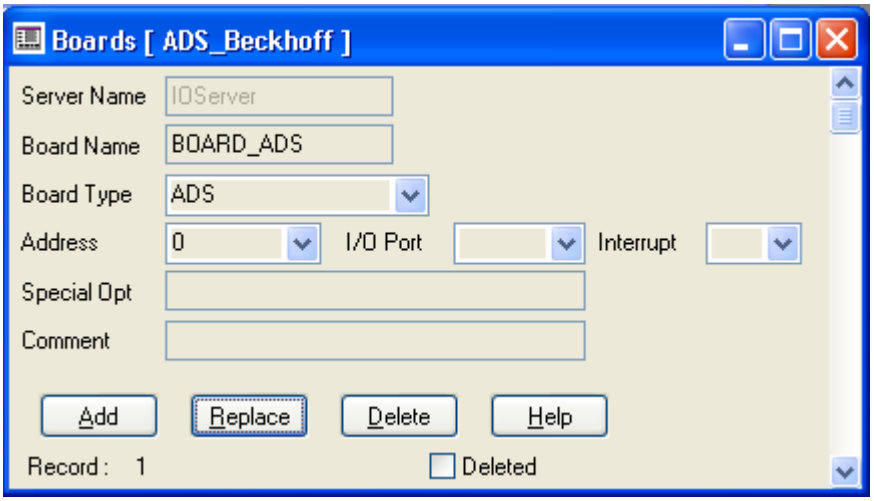
4.1 Introduction

This section defines how the user will see the driver. This relates directly to how the Citect forms need to be filled out and any special INI options. For the kernel, the debug trace messages and the *Stats.Special* counters are documented.

4.2 Driver Name

ADS

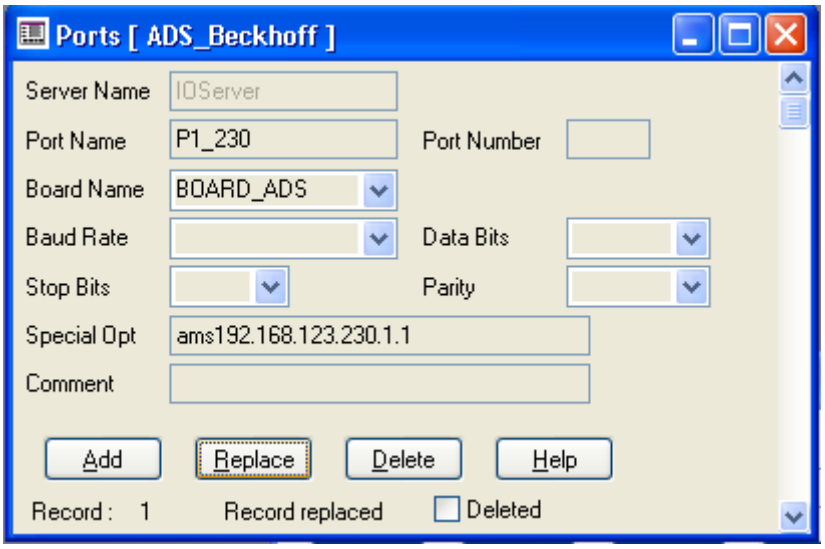
4.3 Boards Form



The driver is a board driver, *Board Type* should be set to *ADS*

Enter *0* as address.

4.4 Ports Form



Wheter to use direct driver with TCP/IP or the ADS API is determined by a *Citect.ini* parameter, se later on.

Here is where the AMS Net Id should be entered. The ID number has the following format,

x.x.x.x.y.z This number refers to the AMS Net Id specified in the TwinCAT System Properties. If the TwinCAT is on the local computer the *Local Computer AMS Net Id* should be used, for TwinCAT systems in a network the *AMS Net Id* specified in the *Remote Computers* should be used.

For easy to understand configuration in TwinCAT the first four digits x.x.x.x in the *AMS Net Id* should be the TCP/IP address of the local/remote computer, i.e. *192.168.123.230*. The two last digits *y.z* is information used in the *AMS Net Id* by the TwinCAT system.

These two digits have to be the same as configured in the *AMS Net Id* of the local/remote computer. If the *AMS Net Id* for the local/remote computer is set to 192.168.123.230.1.1, the *Special Option* field has to be set according to the above example.

x, y, z should be a decimal value between 0 – 255

-s Optional. This forces the driver to send single requests (read/write) only.

TwinCAT versions before v2.11 (build 1550) does not support multiple requests using *ADSIGRP_SUMUP_READ* and/or *ADSIGRP_SUMUP_WRITE*.

This parameter affects this specific AMS Net Id device(s) only. See also parameters *MaxReadsPerRequest* and *MaxWritesPerRequest* later on.

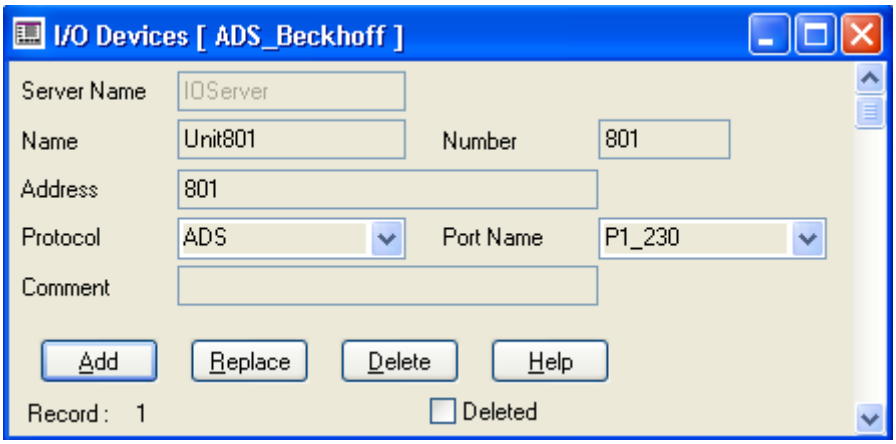
The driver automatically determine the TwinCAT PLC I/O version so this parameter should normally not be used. It will degrade performance.

Example,

-ams192.168.123.230.1.1

-AMS could be lower or upper case.

4.5 IO Devices Form



4.5.1 Address

In the each TwinCAT system you can have up to 4 PLC's, each PLC will get a consecutive *Port Number* as follows. *PLC1 -> 801, PLC2 -> 811, PLC3 -> 821* and *PLC4 -> 831*, these unique *Port Numbers* will automatically be assigned to each PLC in the TwinCAT system. The address specified in the address field has to be one of these numbers as the driver uses the address when communicating to the TwinCAT system. This means that every local/remote TwinCAT system will have 1 to 4 PLC's which Citect can read.

Which PLC is read on the local/remote TwinCAT system by Citect is decided by the *IO Device Address*. Which computer the PLC is located on is decided by the *AMS Net Id* (*Special Opt* in the *Ports* form).

The *Address* field could be any TwinCAT2 or TwinCAT3 address, ex *801, 821, 851* etc.

Example,

801

4.5.2 Protocol

ADS

4.6 Pulldown lists Help

The following entries should be included in the Citect Help.DBF spec file.

TYPE	DATA	FILTER
PROTOCOL	**ADS**	

4.7 PROTDIR.DBF

The following entries should be included in the Citect Protdir.DBF spec file.

TAG	**FILE**	**BIT_BLOCK**	**MAX_LENGTH**	**OPTIONS**

ADS	ADS	256	2048	0x2000cf

4.8 Parameters and INI options

4.8.1 Standard Parameters

Block	256		(Citect blocking is prohibited)
Delay	10	ms	
MaxPending	50		
Polltime	0	ms	
Timeout	5000	ms	
Retry	0		(Sets the complete packet retries, should be 0)
WatchTime	30	s	

4.8.2 Driver Specific Parameters

All ADS specific parameters are located in the section 'ADS'

PARAMETER	DEFAULT	DESCRIPTION
Delay	10 ms	Each request cycle, the driver will scan through internal queues for work to do (create packets, update statistics, cleanup memory, return new values from buffers etc). Performance is linear to this time.
TimeOut	5000 ms	Response timeout for the device (for complete packet)
MaxPending	50	Number of DCBs the driver may handle simultanously. Actually the default is quite low for this driver, but Citect is not capable of handling an unlimited number of outstanding DCBs. This number must be multiplied by number of channels to get the total DCBs for Citect.
WatchTime	30 s	When Citect send DCB requests to the driver, it have to respond within this time. This is also the cycle time for init and status requests for offline devices.
DriverType	DLL	Determines if the ADS driver should use *TwinCat ADS DLL* calls or direct TCP connection to the router/devices. Allowable values *DLL* or *TCP*. Defaults to *DLL*. Warning! If *TwinCAT* is installed on the Citect server, the *TCP* mode should not be used, because PLC's won't allow two connections simultaneous from the same IP.
SymbolVersionCheckIdleInterval	10000	Determines how ofter the driver should check for updates in the symbol table when no values are requested.
SymbolVersionCheckBusyInterval	1000	Determines how ofter the driver should check for updates in the symbol table when values are requested.
MaxReadsPerRequest	30	Number of read requests handled in one packet. Packets are bundled in a per-SoftPLC fashion, ex if address *801* and *802* exist on the same *AMS Net Id*, and Citect simultanously request 5 values from each SoftPLC it will result in two requests with 5 variables in each. TwinCAT versions before v2.10 (build 1324) does not support multiple read requests. The driver automatically determine if multiple requests could be used. To force single reads either use the op-

		tional –*s* parameter in the *Ports form* of the affected SoftPLC's or set both *MaxReadsPerRequest* and *MaxWritesPerRequest to 1* to affect all SoftPLCs.
MaxWritesPerRequest	30	Number of write requests handled in one packet. Packets are bundled in a per-SoftPLC fashion, ex if address *801* and *802* exist on the same *AMS Net Id*, and Citect simultanously request 5 values from each SoftPLC it will result in two requests with 5 variables in each. TwinCAT versions before v2.11 (build 1550) does not support multiple write requests. The driver automatically determine if multiple requests could be used. To force single reads either use the optional –*s* parameter in the *Ports form* of the affected SoftPLC's or set both *MaxReadsPerRequest* and *MaxWritesPerRequest to 1* to affect all SoftPLCs. *OBS! Not yet implemented. This parameter has no effect since the driver always issues single writes.*
IgnoreStartupErrors	0	If set to 1 no error message will be displayed if the driver detecs some problem in the variable databases during startup. If Citect is running as service it will just hang the IOServer process if an error message is presented during startup. The error box is not visible to the user, hence the *Ok* button could not be pressed.
TraceToLogFile	0	If set to 1 all debug data that would appear in Kernel using the *debug <portname> all* command will be logged to file *ADS.log* in the ADS subfolder of Citect's data folder. The file could grow very big, but its size will be limited by *TraceFileLimit*.
TraceFileLimit	100	The file size limit of the *ADS.log* file. This is the maximum file size in megabytes. If the file exceed this size, the logging will be stopped, however to continue logging just change the file name or delete it and logging will automatically resume (no restart is required)
TraceByteLimit	64	Limit the number of data bytes presented in logfile and Kernel debug (per packet presented hex data)
TraceFailedSymbols	0	If the driver is not capable of finding symbol addresses in the downloaded symbol table they may be written to a text file to ease up database fixing. The file is named *ADS_FailedSymbols.log* and is placed in the ADS subfolder of Citect's data folder. *0 – No logging* *1 – Trace only missing symbols* *2 – Trace all symbol lookups*
TraceOptions	0	Options for trace: *1 – Trace found I/O devices in DBF's* *2 – Trace variables found in DBF's (will also trace I/O devices)* *4 – Trace address resolving and connection issues (TCP)* *8 – Trace device version and symbol table requests* *16 – Trace all rx loops* Values may be bit-weighted togheter, ex if *TraceOption=15* all above trace options will be used
StoreAdsSymbolTable	0	If set to 1 it will store all symbol tables (and types) as text files named using the I/O device name. The files are placed in the ADS subfolder of Citect's data folder.

AltRunPath	[CTEDIT] RUN	Alternative Run path, is used to find the top project.
AltUserPath	[CTEDIT] USER	Alternative User path. Is used by OID lookup mechanisms to find the *Master.DBF* file.
~~IOServerName~~		For later Citect versions (when clusters was implemented) it is not possible to find the current *IOServer name*, hence it must be entered here. *Parameter has been removed from driver version 1.0.0.14, driver will use the I/O device number instead.*
LocalAmsAddress		The local *AMS Net Id* (and its port). Needed when using TCP. Ex: *192.168.123.80.1.1* If using ADS API DLL the *Local AMS Address* is automtically retrieved from the TwinCAT system, but could be overriden using this parameter.
AMS…=IP		Router table. Remote *AMS Net Id's* (as specified in the *Ports form Special Opt field*)and their physical IP. See section *Setup direct TCP/IP communication without TwinCAT software.* Ex: ams*192.168.123.230.1.1=192.168.123.230* ams*192.168.123.231.1.1=192.168.123.231* ams*192.168.123.232.1.1=192.168.123.232*
UseAMSAddressAs-DefaultIP	0	If set to 1 the driver will automatically use the left part of the *Ports form Special Opt field* AMS-netid as physical IP if a corresponding router table translation is missing in *Citect.ini*, ex *in Ports form Special Opt field* AMS netid is *-ams192.168.123.230.1.1* and it is missing in *Citect.ini* then the driver will use *192.168.123.230* as physical IP instead of indicating an *error (Missing remote Message Router IP in Citect.INI*).
LocalNetworkAddress		Network address (network card) to be used (bind) if the server have more than one network card or IP-address. Generally the OS selects the most appropriate card for the local endpoint. Ex *192.168.123.80*
UseSymbolTable-FileIfExisting	0	If communication to slow devices cause the driver to timeout when trying to read symboltable and datatypes the ADSConfig tool could be used to create a binary address file the driver will use instead of trying to read it from the device. The file should be stored in the ADS subfolder of Citect's data folder. The file is automatically named by ADSConfig to the AMS id and port of the device, ex *ams192.168.123.235.1.1_801.adr*. If the file is missing or corrupt the driver will try to read symboltable and datatypes from the device as it normally would. The binary address file contains also the version number of the downloaded program. If a new program is downloaded to the controller and the versions differ, the driver will try to read symboltable and datatypes from the device. If that is the case it is important to use ADSConfig to update the binary address file. See also *StoreSymbolTableFile* for automatic updates of the file.
StoreSymbolTableFile	0	When the driver has read the symboltable and datatypes it could store it also in a binary format. Used togheter with UseSymbolTableFileIfExisting the driver will only read symboltable and

		datatypes if the version in the PLC changes (new program downloaded). The file is stored in the ADS subfolder of Citect's data folder. The file is automatically named to the AMS id and port of the device, ex *ams192.168.123.235.1.1_801.adr*.

4.9 Debug Messages

Shows the outgoing/incoming data packets. Header in verbose format and data/value in hex.

4.10 Stats Special Counters – DLL mode

Number	Label	Purpose/Meaning of this counter
0	DCB Requests	Total number of driver requests
1	Symbol reloads	Number of symbol table updates
2	Phys read time	Physical read time in millisecs

4.11 Stats Special Counters – TCP mode

Number	Label	Purpose/Meaning of this counter
0	DCB Requests	Total number of driver requests
1	Symbol reloads	Number of symbol table updates
2	Phys read time	Physical read time in millisecs
3	Tx bytes	Transmitted bytes
4	Rx bytes	Received bytes
5	Tx packets	Transmitted packets
6	Rx packets	Received packets
7	Connections ok	Succeeded TCP connection attemps
8	Connections failed	Failed TCP connection attemps

4.12 Other necessary setup – typical Citect.INI settings

The driver default is to communicate using *TwinCAT ADS API*, it could be changed by the *Citect.ini [ADS]* parameter *DriverType*.

Typical *Citect.INI [ADS]* section when using the TwinCAT software and the ADS API:

[ads]
drivertype=dll

4.13 Setup direct TCP/IP communication without TwinCAT software

Set Citect.ini *[ADS]DriverType=TCP*

The *ADS Message Router* is built into the driver and need to be configured in the *Citect.ini [ADS]* section as:

[ads]
drivertype=tcp
localamsaddress=192.168.0.132.1.1

; May use DNS name(s) instead of IP
ams192.168.123.230.1.1=192.168.123.230
ams192.168.123.235.1.1=192.168.123.235

The driver have an own local *AMS Net Id* defined by parameter *LocalAMSAddress*. Use the Citect server's IP address followed by *1.1* as above to avoid confusion. The port defaults to 65534 (same as TwinCAT uses) but can be changed, ex:

LocalAMSAddress=192.168.0.132.1.1:65530

In the remote device's *TwinCAT Message Router* table you must also define the Citect driver's *LocalAMSAddress* (and its IP-address), even if communication takes place in same TCP connection the driver established. If you fail to do this the device will not answer.

To define your own router table you enter the text 'AMS' followed by the *remote message router AMS Net Id's* and the IP-address. To avoid confusion try to use same *AMS Net Id* as the IP (as in the above examples). The IP-address can be exchanged to *DNS adresses*, ex

ams192.168.123.230.1.1=softplc1.trosoft.se

The port is by default 48898 but may be changed, ex:

ams192.168.123.230.1.1=softplc1.trosoft.se:10025

The *AMS Net Id's* must be used in the *Ports form* to associate the *IO Devices* with the correct IP-address. It would of course been more logical to place also the IP-address in the *Special Opt's* field in the *Ports form*, but it has not enough field length.

4.14 Network Address Translation (NAT) using TCP/IP communication without TwinCAT software

Communication to NAT networks behind firewalls is possible because communication use a TCP/IP connection (not UDP). Obviously the remote devices public IP-address must be static or registered dynamically using ex *DynDNS*.

The public IP of the Citect installation must however always be static because the *Remote Message router* table only allows IP-addresses.

5. Basic Testing

5.1 Introduction

The programmer will perform a minimum level of testing which is outlined here.

A sample Project is available which can be used as a starting point for the programmers test Project. When the programmer has completed basic testing and debugging this Project should by backed up and supplied to the Citect Testing department.

5.2 Procedure

The following are points should be covered by basic testing.

- On startup the IO Device comes online without errors.

  - Ok

- The driver supports IO Devices of addresses as documented in the specification.

  - Ok

- The driver reports the IO Device offline when the IO Device is a) powered down, b) disconnected.

  - Ok

- The driver will re-establish communication with the IO Device after a) power cycle, b) disconnection/ reconnection.

  - Ok

- Confirm that retries (if supported) and error reporting operate correctly.

  - Ok (Statistics report)

- The driver reads all the device data types documented as readable in this specification.

  - Ok

- The driver writes to all the device data types documented as write-able in this specification.

  - Ok

- The driver reads and writes all data formats supported by the protocol, ie DIGITAL, INT, LONG, REAL, STRING.

  - Ok

- Test the limit of the IO Devices request size, this should be done for at least DIGITAL and an INT data formats.

  - Not applicable to this driver.

- Let the driver run over night and check that no retries or other errors have occurred.

  - Ok

- If a multidrop or network protocol and if the hardware is available then the protocol should be tested with more than one IO Device connected.

  – Ok

6. Performance Testing

6.1 Introduction

Tests which give some indication of the drivers performance. The programmer needs to perform these tests since the results feed back into the Constants structure and the PROTDIR.DBF.

6.2 Calculating the Blocking Constant – Not applicable

Because this driver is a hybrid of the 'Front-End-Back-End' type, and due to the nature of the ADS protocol, it does not support blocking of any kind

7. References

7.1 References

n/a.

8. TCP Communication fail debugging

8.1 Communication problems

If the unit will not come online, ensure following are ok (TCP mode):

- *LocalAMSAddress* is correct
- *TwinCAT System Manager* in the PLC is in Run mode
- *TwinCAT PLC Control* in the PLC is in Run mode
- The PLC has a program
- The device address (usually *801*) is ok
- The AMS router entries in Citect.ini matches the Ports form
- The router table in the PLC contains a valid entry for the Citect server's IP *and LocalAMSaddress*
- The logged in user (or the user in the Citect service) has *Full control in* the Citect's *User* and *Data* folders
- Parameters are not misspelled

Use the *TraceToLogFile* and *TraceOptions* parameters to create an *ADS.log* file.

If the driver cannot connect there is usually some problem with the router entry in the PLC and Citect.ini.

If the driver can connect there usually are some addressing issues, or the PLC is not online.

If no values appear even if the PLC is connected and online there is usually some addressing issue. Use the parameters *TraceFailedSymbols* and *StoreADSSymbolTable* to lookup failed addresses

8.2 ADSConfig

There is a simple tool that may be used to check communication and create router entries remote in the PLC. It does not require TwinCAT to be installed in the server.