Citect for Windows, Version 6.xx, 7.xx

TREND driver, User information

begcomm Communication AB
Brunnehagen 109
S-417 47 Göteborg
Phone +46 70 258 50 01
Email info@begcomm.com
www.begcomm.com

Driver version history

Version	Modified By	Details
2.00.00.000	Bertil Göransson	Tcpip implementation.
2.00.00.001	Bertil Göransson	Crasch fixed. Different size receive and packetbuffer.
2.00.00.002	Bertil Göransson	IODevice address 61 always presented 0 as value.
2.00.00.003	Bertil Göransson	New GUID.
2.01.00.001	Bertil Göransson	Init of a_map moved from InitChannel to OpenChannel. NULL implemented when delete in CloseChannel.
2.01.00.002	Bertil Göransson	Cleaning up.
2.01.01.001	Bertil Göransson	Switch and new parameter for portserver implemented.
2.01.02.001	Bertil Göransson	Support for IQ3 with new Protdir and all around this.
2.01.03.001	Bertil Göransson	Adjustments in Transmit changed DrvFirstQue to DrvRdQue.
2.01.04.001	Bertil Göransson	Compiled for V6.x. Extra trace for DRIVER_CHANNEL_OFFLINE
2.01.05.001	Bertil Göransson	Fixed a memory leak which arise when Ethernet connect and disconnect very often.
2.01.06.001	Bertil Göransson	Changed MAX_CHANNELS from 32 to 256
3.00.00.001	Bertil Göransson	Internal information changed to begcomm. New GUID.
3.01.00.001	Bertil Göransson	Ported to VS10

Contents

1. INTRODUCTION 6
1.1 Scope 6
1.2 Outline 6
2. QA 7
2.1 Developers Guidelines 7
2.1.1 Accredited Drivers 7
2.1.2 Independent Drivers 7
2.2 Accreditation process 7
3. TARGET DEVICE(S) AND PROTOCOL 9
3.1 Introduction 9
3.2 Device Manufacturer 9
3.3 Device Definition 9
3.4 Communications Method 9
3.5 Communications/Hardware Configuration 9
3.5.1 Wiring Diagrams 9
3.5.2 I/O Device Settings 10
3.5.3 Software Setup 10
3.6 Special Requirements 10
3.7 Maximum Request Length 10
4. PROTOCOL REQUIREMENTS 11
4.1 Introduction 11
4.2 Initialising the Board 11
4.3 Initialising the Port 11
4.4 Initialising the IO Device 11
4.5 IO Device Online Test 11
4.5.1 Devices on the local network 11
4.5.2 Devices on remote networks 12
4.5.3 Devices on dialled up connections 13
4.6 State Flow Description 13
4.7 Message Structure 13
4.7.1 Message information 13
4.7.2 Message type 13
4.7.3 Extended address information 14
4.7.4 Data 14
4.7.5 Virtual networking 14
4.7.6 Data Format 17

4.8 Check Sum 17
4.9 Error Handling 17
5. USER INTERFACE 18
5.1 Introduction 18
5.2 Driver Name 18
5.3 Boards Form 18
5.3.1 Board Type 18
5.3.2 Address 18
5.3.3 IO Port 18
5.3.4 Interrupt 18
5.3.5 Special Opt 18
5.4 Ports Form 18
5.4.1 Baud Rate 18
5.4.2 Data Bits 18
5.4.3 Stop Bits 18
5.4.4 Parity 18
5.4.5 Special Opt 18
5.5 IO Devices Form 19
5.5.1 Protocol 19
5.5.2 Address 19
5.6 Pulldown lists Help 19
5.7 IO Device Variable Types 20
5.7.1 Trendcs.dbf Formats and types 20
5.7.2 Trendcs.dbf Entries 21
5.8 PROTDIR.DBF 22
5.9 Parameters and INI options 22
5.9.1 Standard Parameters 22
5.9.2 Driver Specific Parameters 22
5.10 Remapping 23
5.11 Driver Specific Errors 23
5.12 Driver Error Help 23
5.13 Debug Messages 24
5.14 Stats Special Counters 24
5.15 Hints and Tips 25
6. BASIC TESTING 26
6.1 Introduction 26
6.2 Procedure 26
7. PERFORMANCE TESTING 27
7.1 Introduction 27

7.2 Calculating the Blocking Constant 27

8. REFERENCES AND CONTACTS 28

8.1 References 28

8.2 Contacts 28

1. Introduction

1.1 Scope

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

1.2 Outline

The specification is broken down into the following sections:

Section 1 - Introduction

This section defines the scope of a driver specification and outlines the items addressed by the specification.

Section 2 - Quality Assurance

The QA section defines the requirements and procedures for Quality Assurance Accreditation. It is important you read this if you want your driver integrated into Citect.

Section 3 – Target Device(s) and Protocol

The Physical Communication Method section defines the physical communication method supported, hardware/software suppliers, how the method is setup, any wiring diagrams involved etc.

Section 4 - Protocol Requirements

The Protocol Requirements section details the technical considerations required or incorporated by the driver.

Section 5 - User Interface

The User Interface section defines how the user will see and setup the driver in Citect.

Section 6 - Basic Testing

The Basic Testing section defines the items which should be addressed in Basic testing by the developer.

Section 7 - Performance Testing

The Performance Testing section is used in full testing of the driver by the Citect Testing Department of CiT. Once complete, this will provide details on the reliability and stability of the driver, and point out where the driver needs to be improved.

Section 8 - References and Contacts

The References and Contacts section should be used as a record of reference materials and contacts used in developing this driver.

2. QA

2.1 Developers Guidelines

These guidelines are meant as a rough indication of what options there are for developing Citect drivers and the advantages of these options. It is not a technical discussion of options, rather a marketing guideline.

Drivers fall into two categories, Accredited and Independent.

2.1.1 Accredited Drivers

Accredited drivers are those drivers that have been put through the CiT Driver QA Scheme and have passed all stages of this accreditation process. It is a precondition to becoming accredited that these drivers will be included with Citect in a normal release.

Accreditation has the following advantages:
1. The driver will be included in the product and a certificate stating this driver has achieved Accreditation will be sent to the developer.
2. Accredited drivers will be honoured as part of the product in terms of Citect Support and receive full cooperation between Citect Support personnel and the developer. On the other hand, independent driver problems will immediately be referred on to the original developer.
3. Help documentation and Express Wizards are provided, free of charge, for all Accredited drivers. Help documentation for Independent drivers is the responsibility of the developer.
4. Accreditation is included in the cost of the DDK. A high level of quality is expected and if this is not met the driver will not be Accredited.
5. Citect Customers see value in Accredited drivers as there is some assurance that the driver will operate as documented. Some customers may only accept Accredited drivers.

2.1.2 Independent Drivers

Independent drivers are those that have not completed or are not intended to complete the Accreditation process. These drivers will not be included in Citect, nor will they be given any support by Citect Support personnel. We would request all drivers be sent to CiT regardless, even if they are not to be included in the product. If this is done, we can try to ensure compatibility with future versions of Citect.

Independent Drivers have the following advantages:
1. Drivers may be written by or for an end user giving them an edge over their opposition by using Citect.
2. Drivers may be developed as part of a package offered by System Integrators or including pre-configured packages etc., thereby maintaining the intellectual and financial investment. This would be similar to value added or OEM style marketing.

2.2 Accreditation process

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. It is advisable to ensure that items before each checkpoint are complete before proceeding to avoid rework if changes are required.

	Description	Person	Date
1	This specification document is written.		

2	Specification reviewed and accepted by CiT Driver Development.		
	At this checkpoint coding is ready to be commenced.		
3	Driver coded.		
4	Code and specification reviewed and accepted by CiT Driver Development.		
5	Testing with connection project, and performance test.		
6	Driver integrated into Citect source and built.		
7	Documentation is written.		
8	Documentation reviewed.		
	At this checkpoint coding is done and the driver is available as a beta.		
9a	Full testing is carried out.		
9b	Performance testing is carried out.		
9c	Specification and documentation updated from testing/performance tests		
	At this checkpoint the testing is complete.		
10a	Review for completeness by developer, tester, documenter and CiT Driver Development		
10b	Add driver to install disks		
10c	Add driver to drivers database		
10d	Support notified of new driver for training purposes		
11	Sales notified of new driver		
	The driver is now finished.		

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

3. Target Device(s) and Protocol

3.1 Introduction

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

3.2 Device Manufacturer

Trend Control Systems Limited

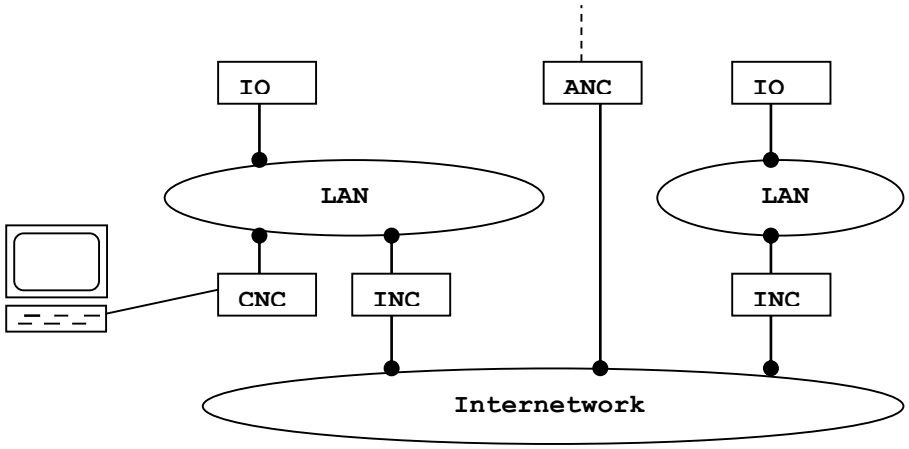
3.3 Device Definition

The devices attached to the Trend network can be of different kinds, although they all support the same communication interface. Three types of devices are supported: outstations (the actual I/0- devices; IQ:s), internetwork devices (INC:s) and autodialling devices (ANC:s). The devices must have firmware 4.7 or later installed.

3.4 Communications Method

The driver communicates with the Trend network via a CNC device through the computer's COM-port using serial communication or via an EINC device through the computer's Ethernet port. The protocol uses 7 bits, one odd-parity bit and one stop bit.

3.5 Communications/Hardware Configuration



The CNC device is connected to a local Trend network (LAN) to which a number of devices (max 116) can be attached. Among those can be an internetwork device (INC) that links the local network to other Trend networks via an internetwork. To this network other networks can be linked via INC:s or ANC:s. The maximum number of devices on the internetwork is also 116.

For more detailed information about the hardware configuration in the Trend network, please refer to the *CNC+ Interface Development Technical Manual*.

3.5.1 Wiring Diagrams

The CNC to device link is implemented as an RS232 communications link. If the CNC is housed in a NETB box, the device is connected via the 25 way socket on the NETB. Alternatively, it can be connected directly to the CNC via the 10 way 'Stocko connector' on the CNC PCB as follows:

10 way Stocko No	25 way NETB Connector	Signal No

1		(not connected)
2		(tied to signal ground)
3		(not connected)
4	7	GND – signal ground
5		(not connected)
6->	5	DTR – TRUE if CNC can accept data
7->		RTS – TRUE when CNC has power
8->	3	TX – data from CNC to device
9<-	2	RX – data from device to CNC
10<-	20	CTS – TRUE if device can accept data

where <- implies signal into CNC and -> implies signal from CNC.

3.5.2 I/O Device Settings

See the *Trend IQ Configuration Manual* for details.

3.5.3 Software Setup

None.

3.6 Special Requirements

None.

3.7 Maximum Request Length

The maximum request length is 100 bytes, including start and stop characters.

4. Protocol Requirements

4.1 Introduction

This section documents all the requirements of the protocol itself.

4.2 Initialising the Board

No board initialisation is needed.

4.3 Initialising the Port

The CNC device needs to be put in supervisor mode to enable the reception of global messages. This is done by replying to the CNC request:

STX NUL NUL w 0 0 *ENQ* 0 0 *ETX*

This request is sent by the CNC device regularly to obtain information about the attached device.
The reply in our case should be:

STX NUL NUL w 0 0 *FF* 0 0 ! 0 0 0 0 *v1 v2 v3* 0 0 1 B @ *p NUL NUL ETX*

where the parts in italics mean:

STX	Start character (ASCII 2).
NUL	NUL-character (ASCII 0)
ENQ	Code character meaning the command is a READ request (ASCII 5).
FF	Code character meaning the message is a REPLY to a preceding READ (ASCII 12).
v1, v2 ,v3	Version number
p	Polltime that should be used for subsequent similar requests from the CNC device. 0 => Do not poll 1 => Poll with 15 seconds interval 2 => Poll with 1 minute interval
ETX	End character (ASCII 3).

4.4 Initialising the IO Device

No initialisation is needed for the I/O devices.

4.5 IO Device Online Test

To test the status of a device, a read command is sent to the device and the response message is examined. The value returned by the read command is then discarded. The command asks the device for the value of sensor number one, and since all devices have at least one sensor, every device that is online should be able to respond correctly to this command.

4.5.1 Devices on the local network

For devices on the local network, the read command looks like the following:

STX DEST SRC v *n1 n2 ENQ* 0 0 S 1 (V) *ETX*

and the response message should look like this if the device is responding correctly:

STX DEST SRC v *n1 n2 FF* 0 0 S 1 (V = *val*) *ETX*

If the device cannot process the request for some reason, it returns a message like:

STX DEST SRC v *n1 n2 NAK* 0 0 *ETX*

If no response is received from the device at all, the network returns the following message:

STX DEST SRC v *n1 n2 SYN* 0 0 *ETX*

The parts in italics above have the following meanings:

STX	Start character (ASCII 2).
DEST	Destination address; in the read request, it's a number containing the address of the I/O device we're checking, while in the responses it's the address of ourselves, that is the CNC device we're connected to.
SRC	Source address; in the read request, it's our address and in the responses it's the answering device's address.
n1 n2	Message number. Used to associate incoming responses to the right requests.
ENQ	Code character meaning the command is a READ request (ASCII 5).
ETX	End character (ASCII 3).
FF	Code character meaning the message is a REPLY to a preceding READ (ASCII 12).
val	The value returned from the READ-operation.
NAK	Code character meaning the command wasn't understood/performed by the device (ASCII 21).
SYN	Code character meaning the command wasn't delivered by the network (ASCII 22).

To summarize, the thing to look for in the response to the READ command is the seventh byte. If it is *FF*, the device is online and responding correctly, but if it is *NAK* or *SYN*, the online test failed.
If no response at all is received from the network within the timeout period, the online test also fails.

4.5.2 Devices on remote networks

For devices on remote networks connected to the internetwork, the read command differs slightly from above:

STX 126 *SRC* v *n1 n2 ENQ DEST LAN* S 1 (V) *ETX*

The successful response message also looks a bit different:

STX DEST 126 v *n1 n2 FF SRC LAN* S 1 (V = *val*) *ETX*

The two messages telling us that something went wrong are modified in the same way:

STX DEST 126 v *n1 n2 NAK SRC LAN ETX* and

STX DEST 126 v *n1 n2 SYN SRC LAN ETX*

The italic parts that are new or differs from above are:

SRC	Source address; in the read request, it's the address of our CNC device, while in the responses it's the address of the I/O device on the remote network.
DEST	Destination address; in the read request, it's the address of the I/O device on the remote network, while in the responses it's our CNC address.
LAN	A number containing the address of the remote network.

4.5.3 Devices on dialled up connections

When communicating with devices dialled up from an ANC device, first of all the connection has to be established. This can be very time consuming, wherefore the online test is not carried out in the same way for these devices. Instead, all devices that are to be dialled up are initially considered online. Because of this, it is not until a device is used for the first time that the required connection is made and the actual state of the device can be established.

4.6 State Flow Description

The request for a read or write is sent through the CNC device to the network. The request contains a source, a destination and a message number. The source is the address of the CNC device. After processing an incoming request, the destination device sends a response message to the network containing the same message number as in the request and with the source and destination switched. The CNC device picks up the response (because the destination in the message now is the address of that device) and the message can then be associated with the right request by comparing the message numbers.

4.7 Message Structure

The message structure looks as follows, where each item represents a byte:

```
STX DEST SRC T1 T2 T3 CLASS X1 X2 D1 D2 ... Dn ETX
```

STX	Start character (ASCII 2)
DEST	Destination address
SRC	Source address
T1, T2, T3	Message information
CLASS	Message type
X1, X2	Extended address information
D1, D2 ... Dn	Data
ETX	End character (ASCII 3)

4.7.1 Message information

T1 is a character that codes the type of communication that is to be used. Because the driver uses attribute communication, T1 is always 'v'. T2 and T3 is used to code the message number.

4.7.2 Message type

CLASS can be one of the following:

EOT (ASCII 4)	WRITE command

ACK (ASCII 6)	Acknowledge WRITE
ENQ (ASCII 5)	READ command
FF (ASCII 12)	Reply to READ
NAK (ASCII 21)	READ/WRITE not understood/performed by device
SYN (ASCII 22)	READ/WRITE not delivered by network

4.7.3 Extended address information

When sending requests to a device on the local network, the DEST-byte contains the address of the device, while X1 and X2 are '0'. When sending requests to a device on a remote network connected to the internetwork via an INC device on the other hand, the DEST-byte contains the address of the INC (always 126), X1 contains the address of the device on the remote network and X2 contains the number of the network. When receiving messages from the remote network, the SRC-byte contains the INC address (126), X1 the device address and X2 the network number.

4.7.4 Data

The data part of the message can be 0 to 90 bytes long. The driver uses attribute communication, meaning that the data part consists of a string of ASCII characters. For a read request, the string is built up in the following way:

```
MTYPE NBR ( PARAM )
```

where MTYPE is the module type, NBR the module number and PARAM the requested parameter. A reply message is built up like:

```
MTYPE NBR ( PARAM = VALUE )
```

where MTYPE, NBR and PARAM are the same as above, while VALUE is the requested value. A write message is built up in the same way as the reply message, with VALUE as the value to write. The ACK-message that comes as a response to a WRITE doesn't contain any data.

Some examples:

Read request for the value of sensor number three: `S3(V)`
Reply returning the value of 19 for sensor three: `S3(V=19)`
Write request setting the state of switch five to I: `W5(S=I)`

4.7.5 Virtual networking

When communicating with devices residing on a remote network using an autodialling device (ANC), first of all a connection has to be established. This is done by first sending the message:

STX 127 *SRC* X 1 1 *BEL* 0 0 @ A *D1 D2 ... D20 RLAN ETX*

to the Trend network. It is a request to find any ANC that already may have a connection established to the desired device.

If the response is:

STX DEST 126 X 2 3 *BEL SRC ALAN D1 D2 ... D20 RLAN VCHAR1 VCHAR2 ETX*

a connection exists, and we can use it. Otherwise, we have to dial up a new connection by sending the request:

STX 127 *SRC* X 1 2 *BEL* 0 0 @ A *D1 D2 ... D20 RLAN ETX*

If we then get the response (same as above):

STX DEST 126 X 2 3 *BEL SRC ALAN D1 D2 ... D20 RLAN VCHAR1 VCHAR2 ETX*

we have got a valid connection. Otherwise, we may receive the response:

STX DEST 126 X 2 1 *BEL SRC ALAN D1 D2 ... D20 REASON ETX*

which tells us what went wrong.

The symbols used above but not explained are:

D1...D20	The telephone number as ASCII-characters.
RLAN	The address of the network at the remote end.
BEL	Code character meaning that the command is an unacknowledged write. (ASCII 7)
ALAN	The address of the network containing the ANC device.
VCHAR1, VCHAR2	Characters that are to be used later on when using the established connection.
REASON	Character coding the reason for disconnection. Possible values: '!' : Number did not answer '"' : No dial tone '#' : Number busy '$' : Modem fault detected '0' : Local ANC had to dial elsewhere '@': Modem dropped link 'A' : Link failed (perhaps poor line) 'P' : Remote ANC had to dial elsewhere 'p' : Normal link termination

The result of not being able to establish a connection is that the device has to be considered offline. When a valid connection is established on the other hand, messages can be sent in the format:

STX VCHAR1 SRC T1 T2 T3 CLASS DEST VCHAR2 D1 D2 ... Dn ETX

where VCHAR1 and VCHAR2 are the special characters received as described above.

4.7.5.1 Text alarms

Device alarms can be sent to the driver in text format. When an incoming message has the CLASS-byte set to 'F', it is an alarm message and can be either in text format or code format. Only text format alarms can be recognised as item alarms by the driver. The data part in a text alarm consists of:

Pos	**Contains**
1-15	Device label
16	ASCII 32
17-36	Item label
37	ASCII 32
38-57	Alarm description
58	ASCII 32
59-60	Hour that alarm occurred
61	ASCII 58
62-63	Minute that alarm occurred
64	ASCII 32
65-66	Day of month that alarm occurred

67	ASCII 47
68-69	Month of year that alarm occurred
70	ASCII 47
71-72	Year since 1900 (?)
73	ASCII 13
74	‘O’
75-77	Device address
78	Item type of point in alarm
79-81	Item number
82	ASCII 32
83-86	Alarm code
87	ASCII 13

All alarms that are received by the driver are acknowledged by responding with an ACK-message. Item alarms are stored in a cache in the driver, whereas other types of alarms that are not recognized as item alarms (i.e. general alarms, network alarms, non-text alarms etc.) are written to an output file if one is specified. (See 5.9.2)

4.7.6 Data Format

Values are sent in one of these formats:

NUMBER: -16.345
STRING: “High temperature”
BINARY: O
TIME: 17:13:24
INVALID: “inv”

Numbers are recognised as being characters ‘0’-‘9’,’-‘,’+’ or ‘.’ only.
Strings are recognised as being enclosed in double-quotes.
Binaries are recognised as being made of characters ‘I’ or ‘O’ only (letters).
Times are recognised by being made of characters ‘0’-‘9’ or ‘:’ only, in the form h:mm, hh:mm, h:mm:ss or hh:mm:ss.

4.8 Check Sum

No check sum is used.

4.9 Error Handling

When receiving a NAK or a SYN-message when the online test is made (see above!) the unit is put offline.

5. User Interface

5.1 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.

5.2 Driver Name

TRENDCS

5.3 Boards Form

5.3.1 Board Type

For serial communication choose **COMX** as the board driver.

For communication over tcp/ip choose **TCPIP** as the board driver.

5.3.2 Address

0

5.3.3 IO Port

None. (The number of the COM-port is specified in the Ports form.)

5.3.4 Interrupt

None.

5.3.5 Special Opt

None.

5.4 Ports Form

5.4.1 Baud Rate

Maximum baud rate is 19200. Other possible rates are 9600, 4800 and 1200.
Note that the CNC device has to be set accordingly.

5.4.2 Data Bits

7 data bits.

5.4.3 Stop Bits

1 stop bit.

5.4.4 Parity

ODD_P

5.4.5 Special Opt

None for serial mode.

For tcp/ip mode the IP address and port number has to be filled in.
Eg. –i192.168.0.32 –p10005

5.5 IO Devices Form

5.5.1 Protocol

Choose TRENDCS for series IQ200
Choose TRENDCS3 for series IQ3

5.5.2 Address

On a Trend network the address of an I/O device is in the range [1,4-9, 11-119]. To address devices residing on remote networks, a LAN number also has to be specified. The LAN number has the range [1, 4-9, 11-119]. Finally, when using dialled up connections, a telephone number has to be specified. The phone number can be up to 20 digits long.

The I/O device address can then be entered in the three following ways:

1) Device address only. This is used when the device is connected to the local network.
Example: `114` (device 114 on the local network).

2) The LAN number followed by the device address on that LAN, separated with a dot (.).
Example: `12.18` (device 18 on LAN 12)

3) The LAN number and device address followed by a telephone number after an octothorp (#).
Example: `20.111#55512345` (device 111 on LAN 20 on the dialled up connection using telephone number 55512345)

The telephone number can consist of the following characters:

'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'	The corresponding digits
'A', 'B', 'C', 'D', '*', '#'	Passed to modem; not used by the ANC device
'<' or 'L'	Force following digits to be PULSE dialled
'=' or 'M'	Force following digits to be TONE dialled
':' or 'J'	Wait 2 seconds (or until dial tone) before dialling next digit
';' or 'K'	Wait 10 seconds (or until dial tone) before dialling next digit

The CNC address to be used can also be specified explicitly for each I/O device. This is done by entering the CNC address first of all followed by a colon (:). Example: `5:13.22` specifies that when communicating with device 13 on LAN 22, the CNC residing on address 5 should be used. This value overrides the one (if any) entered in Citect.ini. (See section 5.9.2!)

5.6 Pulldown lists Help

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

TYPE	DATA	FILTER
PROTOCOL	TRENDCS	
PROTOCOL	TRENDCS3	

5.7 IO Device Variable Types

5.7.1 Trendcs.dbf Formats and types

IO Device Type	Citect data format	Citect data types	Description/Special Usage/Limitations/ Valid Ranges
Digital input status	I*u*(S)	DIGITAL	*u* = 1 to 96
Digital input hours run	I*u*(H)	REAL	*u* = 1 to 96
Digital input number of starts	I*u*(N)	LONG	*u* = 1 to 96
Digital input delay	I*u*(D)	REAL	*u* = 1 to 96
Digital bit status	B*u*(S*v*)	DIGITAL	Bit *v* of byte *u*; *u* = 1 to 1012, *v* = 0 to 7
Sensor value	S*u*(V)	REAL	*u* = 1 to 96
Driver hours run	D*u*(H)	REAL	*u* = 1 to 64
Driver number of starts	D*u*(N)	LONG	*u* = 1 to 64
Driver analogue value	D*u*(V)	REAL	*u* = 1 to 64
Analogue value	A*u*(V)	REAL	*u* = 1 to 510
Knob value	K*u*(V)	REAL	*u* = 1 to 60
Switch status	W*u*(S)	DIGITAL	*u* = 1 to 60
Loop gain	L*u*(G)	REAL	*u* = 1 to 64
Loop integral	L*u*(I)	REAL	*u* = 1 to 64
Loop derivate	L*u*(T)	REAL	*u* = 1 to 64
Zone starttime 1	Z*u,v*(E)	STRING	4-8 characters, in the form h:mm, hh:mm, h:mm:ss or hh:mm:ss. *u* = 1 to 5, *v* = 1 to 14. The value of *v* specifies the day: 1 – 7 equals Monday -> Sunday in current week, 8 – 14 equals Monday -> Sunday in standard week.
Zone stoptime 1	Z*u,v*(F)	STRING	Same as above
Zone starttime 2	Z*u,v*(G)	STRING	Same as above
Zone stoptime 2	Z*u,v*(H)	STRING	Same as above
Zone starttime 3	Z*u,v*(I)	STRING	Same as above
Zone stoptime 3	Z*u,v*(J)	STRING	Same as above
Time module, hours	T(H)	INTEGER	
Time module, minutes	T(N)	INTEGER	
Time module, day	T(D)	INTEGER	
Time module, month	T(M)	INTEGER	
Time module, year	T(Y)	INTEGER	
Time module, weekday	T(W)	INTEGER	
Sensor HIGH alarm	S*u*(A_HIGH)	DIGITAL	*u* = 1 to 96

Sensor LOW alarm	S*u*(A_LOW)	DIGITAL	*u* = 1 to 96
Sensor OUTL alarm	S*u*(A_OUTL)	DIGITAL	*u* = 1 to 96
Sensor READ alarm	S*u*(A_READ)	DIGITAL	*u* = 1 to 96
Digital input DI=0 alarm	I*u*(A_DI=0)	DIGITAL	*u* = 1 to 96
Digital input DI=1 alarm	I*u*(A_DI=1)	DIGITAL	*u* = 1 to 96
Driver SDGT alarm	D*u*(A_SDGT)	DIGITAL	*u* = 1 to 64
Driver MINT alarm	D*u*(A_MINT)	DIGITAL	*u* = 1 to 64
Loop SDEV alarm	L*u*(A_SDEV)	DIGITAL	*u* = 1 to 32
Loop PVFL alarm	L*u*(A_PVFL)	DIGITAL	*u* = 1 to 32

Where:

u Decimal number

v Decimal number

5.7.2 Trendcs.dbf Entries

TEMPLATE	UNIT_TYPE	RAW_TYPE	BIT_WIDTH	LOW	HIGH	COMMENT
I%U(0)(S)%*2048	0x495300	0	1	1	96	DigIn, status
I%U(0)(H)%*256	0x494800	2	32	1	96	DigIn, hrs run
I%U(0)(N)%*256	0x494E00	4	32	1	96	DigIn, no starts
I%U(0)(D)%*256	0x494400	2	32	1	96	DigIn, delay
B%U(0)(S%u(0,0,7))%*2048	0x425300	0	8	1	1012	DigBit, status (B12(S3) gives the fouth bit in byte 12)
S%U(0)(V)%*256	0x535600	2	32	1	96	Sensor, value
D%U(0)(H)%*256	0x444800	2	32	1	64	Driver, hrs run
D%U(0)(N)%*256	0x444E00	4	32	1	64	Driver, no starts
D%U(0)(V)%*256	0x445600	2	32	1	64	Driver, analog value
A%U(0)(V)%*256	0x415600	2	32	1	510	Analog, value
K%U(0)(V)%*256	0x4B5600	2	32	1	60	Knob, value
W%U(0)(S)%*2048	0x575300	0	1	1	60	Switch, status
L%U(0)(G)%*256	0x4C4700	2	32	1	64	Loop, Gain
L%U(0)(I)%*256	0x4C4900	2	32	1	64	Loop, Integral
L%U(0)(T)%*256	0x4C5400	2	32	1	64	Loop, Derivate
Z%U(0,1,5)%*15,%+U(0)(E)%*256	0x5A4501	7	72	1	14	Zone, start 1
Z%U(0,1,5)%*15,%+U(0)(F)%*256	0x5A4601	7	72	1	14	Zone, stop 1
Z%U(0,1,5)%*15,%+U(0)(G)%*256	0x5A4701	7	72	1	14	Zone, start 2
Z%U(0,1,5)%*15,%+U(0)(H)%*256	0x5A4801	7	72	1	14	Zone, stop 2
Z%U(0,1,5)%*15,%+U(0)(I)%*256	0x5A4901	7	72	1	14	Zone, start 3
Z%U(0,1,5)%*15,%+U(0)(J)%*256	0x5A4A01	7	72	1	14	Zone, stop 3
T(H)%N%*256	0x544802	1	16	1	1	Time module, hrs
T(N)%N%*256	0x544E02	1	16	1	1	Time module, min
T(D)%N%*256	0x544402	1	16	1	1	Time module, day
T(M)%N%*256	0x544D02	1	16	1	1	Time module, month
T(Y)%N%*256	0x545902	1	16	1	1	Time module, year
T(W)%N%*256	0x545702	1	16	1	1	Time module, weekday
S%U(0)(A_HIGH)%*2048	0x534101	0	1	1	96	Alarm: Sensor above specified value
S%U(0)(A_LOW)%* 2048	0x534102	0	1	1	96	Alarm: Sensor below specified value
S%U(0)(A_OUTL)%* 2048	0x534104	0	1	1	96	Alarm: Sensor outside working limits of hardware
S%U(0)(A_READ)%* 2048	0x534108	0	1	1	96	Alarm: Sensor not giving any reading

I%U(0)(A_DI=0)%* 2048	0x494101	0	1	1	96	Alarm: Digital input reads 0 (should read 1)
I%U(0)(A_DI=1)%* 2048	0x494102	0	1	1	96	Alarm: Digital input reads 1 (should read 0)
D%U(0)(A_SDGT)%* 2048	0x444101	0	1	1	64	Alarm: Driver readback fault
D%U(0)(A_MINT)%* 2048	0x444102	0	1	1	64	Alarm: Driver maintenance interval alarm
L%U(0)(A_SDEV)%* 2048	0x4C4101	0	1	1	32	Alarm: Loop value outside setpoint band
L%U(0)(A_PVFL)%* 2048	0x4C4102	0	1	1	32	Alarm: Loop value (input) has fault

Comment: The UnitType has the structure `0xaabbcc`, where *aa* is the hexadecimal ASCII-code for the module type, *bb* is the hexadecimal ASCII-code for the module variable and *cc* is used for special flags as listed below.

Bit	Meaning
1	When used in normal tags: Trend-specific TIME-format. When used in alarm tags: First alarmtype for item type
2	When used in normal tags: No module number required. (i.e. for time modules) When used in alarm tags: Second alarmtype for item type
3	When used in alarm tags: Third alarmtype for item type
4	When used in alarm tags: Fourth alarmtype for item type

5.8 PROTDIR.DBF

TAG	FILE	BIT_BLOCK	MAX_LENGTH	OPTIONS
TRENDCS	TRENDCS	2048	2048	0x4F
TRENDCS3	TRENDCS3	2048	2048	0x4F

5.9 Parameters and INI options

5.9.1 Standard Parameters

Block	256
Delay	0
MaxPending	9
Polltime	0
Timeout	2000
Retry	3
WatchTime	30

5.9.2 Driver Specific Parameters

Driver specific parameters are entered in the Citect.ini-file under the tag [TRENDCS].
The following parameters are supported:

CNC_ADDRESS — The address of the CNC-device connected to the computer. If no value is supplied for this parameter, the default of 1 is used.

MAXOUTSTANDING — The number of requests that simultaneously can be sent out on the Trend network. If no value is supplied for this parameter, the default of 8 is used.

ALARM_LOG_FILE Path and name of a logfile where unrecognized alarms are logged. If no path and name is supplied for this parameter, no logfile is generated.

PORTSERVER If this parameter is set to 1 the parity algorithm is not involved for tcpip. Default = 0 and that means communication with EINC.
This parameter allows port grouping in Citect.ini e.g.
[TRENDCS.PORT1_BOARD2]
PortServer=1

5.10 Remapping

Not implemented.

5.11 Driver Specific Errors

Driver Error Code (Hexadecimal)	Mapped to (Generic Error label)	Meaning of Error Code
61	**GENERIC_ADDRESS_RANGE_ERROR**	**An invalid I/O-device address is specified**
62	**GENERIC_INVALID_DATA**	**Invalid time format**
71	**GENERIC_INVALID_DATA**	**Command not understood/performed by I/O-device**
72	**GENERIC_ADDRESS_RANGE_ERROR**	**Command not delivered by Trend network**
81	**GENERIC_UNIT_OFFLINE**	**No ANC-node available for connection**
82	**GENERIC_UNIT_OFFLINE**	**No answer from ANC-connection**
83	**GENERIC_UNIT_OFFLINE**	**No dial tone from ANC-connection**
84	**GENERIC_UNIT_OFFLINE**	**Remote ANC-node busy**
85	**GENERIC_UNIT_OFFLINE**	**ANC modem fault**
86	**GENERIC_UNIT_OFFLINE**	**ANC did not connect. Reason unknown.**
87	**GENERIC_UNIT_OFFLINE**	**ANC recently offline**
88	**GENERIC_UNIT_OFFLINE**	**ANC normal disconnection**

5.12 Driver Error Help

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

PROTOCOL	MASK	ERROR	MESSAGE
TRENDCS	**0**	**61**	**An invalid I/O-device address is specified**
TRENDCS	**0**	**62**	**Invalid time format**
TRENDCS	**0**	**71**	**Command not understood/performed by I/O-device**
TRENDCS	**0**	**72**	**Command not delivered by Trend network**

TRENDCS	**0**	**81**	**No ANC-node available for connection**
TRENDCS	**0**	**82**	**No answer from ANC-connection**
TRENDCS	**0**	**83**	**No dial tone from ANC-connection**
TRENDCS	**0**	**84**	**Remote ANC-node busy**
TRENDCS	**0**	**85**	**ANC modem fault**
TRENDCS	**0**	**86**	**ANC did not connect. Reason unknown.**
TRENDCS	**0**	**87**	**ANC recently offline**
TRENDCS	**0**	**88**	**ANC normal disconnection**

5.13 Debug Messages

Message	Displayed when
Sending ANC request for existing connection	First step in initialising a dialled up unit. A message is sent to ANC units to see if anyone already has an existing connection.
Sending ANC dial request	If no response is received for the request above, a dial request is sent to any free ANC.
ANC control handles received	In the dialling phase, the control characters to be used later on are received and stored.
ANC connection established	When dialling has finished and the connection was valid or when a positive response is received when asking for an existing connection.
ANC dialling timed out	When no answer is received from the ANC within the dial up timeout period, the unit is considered offline.
ANC disconnection message received	When a disconnection messages is received from an ANC.
ANC connection couldn't be established	When dial up attempt failed.
Disconnecting unit	After repeated failures to read from the unit, a disconnection message is sent to the ANC.
Responding to CNC: Set supervisor mode	A request is received from the CNC device asking for the type and version of the attached device. A reply is immediately sent in which supervisor mode is set.

5.14 Stats Special Counters

Number	Label	Purpose/Meaning of this counter
0	**MSG RX**	**Number of responses received**
1	**MSG TX**	**Number of requests transmitted**
2	**Bad Requests**	**Number of bad requests**
3	**Comm errors**	**Number of communication errors**
4	**Data errors**	**Number of errors reported from target devices**
5	**Outstanding**	**Number of request currently outstanding on the Trend network**

5.15 Hints and Tips

6. Basic Testing

6.1 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.

6.2 Procedure

The following points should be covered by basic testing.

- On startup the IO Device comes online without errors.
- The driver supports IO Devices of addresses as documented in the specification.
- The driver reports the IO Device offline when the IO Device is a) powered down, b) disconnected.
- The driver will re-establish communication with the IO Device after a) power cycle, b) disconnection/ reconnection.
- Confirm that retries (if supported) and error reporting operate correctly.
- The driver reads all the device data types documented as readable in this specification.
- The driver writes to all the device data types documented as write-able in this specification.
- The driver reads and writes all data formats supported by the protocol, ie DIGITAL, INT, LONG, REAL, BCD, LONG_BCD.
- Test the limit of the IO Devices request size, this should be done for at least DIGITAL and INT data formats.
- Let the driver run over night and check that no retries or other errors have occurred.
- If a multidrop or network protocol is used and if the hardware is available then the protocol should be tested with more than one IO Device connected.

7. Performance Testing

7.1 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.

7.2 Calculating the Blocking Constant

The Performance test procedure is documented in the driver development kit in Appendix A, 'Calculating the Block Constant'. The results of the performance test are recorded here.[1]

block size to read [words]	Average response time [mS]
1	
{25% of maximum}	
{50% of maximum}	
{75% of maximum}	
{maximum}	

From these results the overhead and rate are determined and the ideal blocking constant is calculated

Overhead [mS] =

Word Rate [words / mS] =

Blocking constant [words] =

Note that the calculated blocking constant must now be set by the programmer in the Constants structure (the Block field) in bytes and in the PROTDIR.DBF (the BIT_BLOCK field) in bits.

[1] Since the driver in this case only supports one request per block, this test is not of any practical use.

8. References and Contacts

8.1 References

Trend Control Systems product code 90-1533:
Trend IQ Configuration Manual, issue 6/A 1/6/93
Trend Control Systems product code 91-1756:
Trend IQ Controller Configuration Reference Card, issue 2.0 1/7/90
Trend Control Systems, ref. TD100659A:
CNC+ Interface Development Technical Manual, issue 4/B 21/6/93

8.2 Contacts

www.begcomm.com

info@begcomm.com

