

**Citect for Windows, Version 5.xx, 6.xx**

**M-Bus driver, User information**

## Contents

1.1	DEVICE APPLICATION NOTES .....	3
1.1.1	Key features for M-Bus driver.....	3
1.1.2	Principles of Operation.....	3
1.1.3	Specifications for Bus Installations .....	4
1.1.4	Reference: Communications forms .....	5
1.1.5	Reference: Data types .....	7
1.1.6	Examples data types.....	8
1.1.7	Hints, Tips, and Frequently asked questions .....	14
1.2	DRIVER REFERENCE.....	15
1.2.1	Driver generated error codes .....	15
1.2.2	Parameters, options, and settings .....	15
1.2.3	Advanced .....	17
<b>2.</b>	<b>ANALYSIS.....</b>	<b>19</b>
2.1	M-BUS OVERVIEW.....	19
2.1.1	Transmission Parameters .....	19
2.1.2	Telegram Format .....	20
2.1.3	Meaning of the Fields.....	20
2.1.4	Communication Process .....	22
2.1.5	Selection and Secondary Addressing.....	22
2.1.6	Applications of the FCB-mechanism.....	23
2.1.7	Variable Data Structure.....	24
2.1.8	Fixed Data Header.....	24
2.1.9	Variable Data Blocks.....	26
2.1.10	Data Information Block.....	26
2.1.11	Value Information Block (VIB) .....	28
2.1.12	Manufacturer Specific Data Block .....	29
2.2	DBFs.....	30
2.2.1	Help.dbf.....	30
2.2.2	Mbuscit.dbf Entries .....	30
2.2.3	Protdir.dbf.....	30
2.3	DEVELOPMENT RESOURCES .....	31
2.3.1	Contacts .....	31
2.3.2	Documents .....	31
2.3.3	Driver Version History.....	31
<b>3.</b>	<b>APPENDIX 1 - INSTALLATION .....</b>	<b>32</b>
3.1	BEGINNING OF INSTALLATION.....	32
3.2	FILES TO COPY .....	33
3.3	FILE WITH MORE RECENT DATA EXISTS.....	33
3.4	INSTALLATION IN CITECT PROJECT .....	34
<b>4.</b>	<b>APPENDIX 2 - SOFTWARE PROTECTION .....</b>	<b>35</b>
4.1	UNREGISTERED DRIVER .....	35
4.2	CITECT DONGLE NOT FOUND .....	35
4.3	INCORRECT REGISTRATION .....	36
4.4	REGISTRATION WIZARD.....	36
4.5	DEMO MODE .....	38
4.6	DISABLE THE DRIVER .....	38
4.7	CHANGING LICENSE SIZE.....	38
<b>5.</b>	<b>APPENDIX 3 - M-BUS DEMO.....</b>	<b>39</b>
5.1	HEADER INFORMATION .....	39
5.2	RECORD AND VALUE INFORMATION .....	40
5.3	RECORD AND VALUE DEBUG INFORMATION .....	41
5.4	METER INFORMATION .....	42
5.5	CITECT.INI FOR DEMOPROJECT .....	42

## User information

## 1.1 Device Application notes

	Detail
Manufacturer	M-Bus Usergroup
Device name	Generic
Communications method	Serial

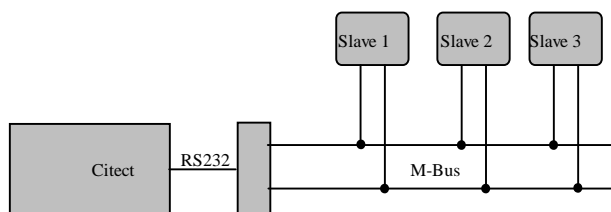
The **M-Bus** (Meter Bus) was developed, by Prof. Dr. Horst Ziegler at the University of Paderborn, to fill the need for a system for the networking and remote reading of utility meters, for example heatmeters, watermeters, electricitymeters and more.

### 1.1.1 Key features for M-Bus driver.

- Generic M-Bus driver
- Supports Variable Data Structure
- Mix different manufacturer at the same bus
- Object orientated tag addressing e.g. Drx.Value, Drx.Tariff, Drx.Vib ....
- All data types automatically converted to strings
- Drx.Vib reads standard and manufacturer specific text automatically from M-Bus tables
- Support for manufacturer specific data
- M-Bus strings is cached in the driver for better performance
- Primary and secondary addressing
- Device/Group-specific parameters in citect.ini
- Multi telegram support

### 1.1.2 Principles of Operation

This chapter is a part of the official "The M-Bus: A Documentation" Version 4.8



*Block diagram showing principle of the M-Bus System*

In order to realize an extensive bus network with low cost for the transmission medium, a two-wire cable was used together with serial data transfer. In order to allow remote powering of the slaves, the bits on the bus are represented as follows:

The transfer of bits from master to slave is accomplished by means of voltage level shifts. A logical "1" (Mark) corresponds to a nominal voltage of +36 V at the output of the bus driver (repeater/converter), when a logical "0" (Space) is sent, the repeater/converter reduces the bus voltage by 12 V to a nominal +24 V at its output.

Bits sent in the direction from slave to master are coded by modulating the current consumption of the slave. A logical "1" is represented by a constant (versus voltage, temperature and time) current of up to 1.5 mA, and a logical "0" (Space) by an increased

current drain requirement by the slave of additional 11-20 mA. The mark state current can be used to power the interface and possibly the meter or sensor itself.

The quiescent state on the bus is a logical "1" (Mark), i.e. the bus voltage is 36 V at the repeater, and the slaves require a maximum constant quiescent current of 1.5 mA each. When no slave is sending a space, a constant current will be drained from the repeater which is driving the bus. As a result of this, and also the resistance of the cable, the actual Mark voltage at the slaves will be less than +36 V, depending on the distance between the slave and the repeater and on the total quiescent current of the slaves. The slave must therefore not detect absolute voltage levels, but instead for a space detect a voltage reduction of 12 V. The repeater must adjust itself to the quiescent current level (Mark), and interpret an increase of the bus current of 11-20 mA as representing a space. This can be realized with acceptable complexity only when the mark state is defined as 36 V. This means that at any instant, transmission is possible in only one direction - either from master to slave, or slave to master (Half Duplex).

As a result of transmission in the master-slave direction with a voltage change of 12 V, and in the answering direction with at least 11 mA, besides remote powering of slaves a high degree of insensitivity to external interference has been achieved.

### 1.1.3 Specifications for Bus Installations

This capital is a part of the official "The M-Bus: A Documentation" Version 4.8

#### 1.1.3.1 Segmentation

An M-Bus system can consist of several so-called zones, each having its own group address, and interconnected via zone controllers and higher level networks. Each zone consists of segments, which in turn are connected by remote repeaters. Normally however, an M-Bus system consists of only a single segment, which is connected via a local repeater to a Personal Computer (PC) acting as master. Such local repeaters convert the M-Bus signals into signals for the RS232 interface.

#### 1.1.3.2 Cable

A two-wire standard telephone cable (JYStY N\*2\*0.8 mm) is used as the transmission medium for the M-Bus. The maximum distance between a slave and the repeater is 350 m; this length corresponds to a cable resistance of up to 29 $\Omega$ . This distance applies for the standard configuration having Baud rates between 300 and 9600 Baud, and a maximum of 250 slaves. The maximum distance can be increased by limiting the Baud rate and using fewer slaves, but the bus voltage in the Space state must at no point in a segment fall below 12 V, because of the remote powering of the slaves. In the standard configuration the total cable length should not exceed 1000 m, in order to meet the requirement of a maximum cable capacitance of 180 nF.

## 1.1.4 Reference: Communications forms

### 1.1.4.1 Boards form

#### 1.1.4.1.1 Serial communication

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

#### 1.1.4.1.2 Communication over Tcpip/Udp

Field	Default	Allowable values
Board Name	This field is user defined.	
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.	

### 1.1.4.2 Ports form

#### 1.1.4.2.1 Serial communication

Field	Default	Allowable values
Port Name	This field is user defined.	
Port number		
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	2400	Refer to the Mbus standard.
Data bits	8	
Stop bits	1	
Parity	EVEN_P	
Special Opt	BLANK	
Comment	This field is user defined and is not used by the driver.	

#### 1.1.4.2.2 Communication over Tcpip

Field	Default	Allowable values
Port Name	This field is user defined.	
Port number		
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	2400	Not used by the driver
Data bits	8	Not used by the driver
Stop bits	1	Not used by the driver
Parity	EVEN_P	Not used by the driver
Special Opt	-i192.168.100.136 -p8801 -t	
Comment	This field is user defined and is not used by the driver.	

## 1.1.4.2.3 Communication over Udp

Field	Default	Allowable values
Port Name	This field is user defined.	
Port number		
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	2400	Not used by the driver
Data bits	8	Not used by the driver
Stop bits	1	Not used by the driver
Parity	EVEN_P	Not used by the driver
Special Opt	-i192.168.100.136 -p8801 -u	
Comment	This field is user defined and is not used by the driver.	

## 1.1.4.3 I/O Devices form for primary addressing

Field	Default	Allowable values
Name	This field is user defined, and is not used by the driver.	
Number	Must be unique.	
Address	1	1 to 250
Protocol	MBUSCIT	
Port name	Refers to the port previously defined in 'ports' form.	
Comment	This field is user defined and is not used by the driver.	

## 1.1.4.4 I/O Devices form for secondary addressing

Field	Default	Allowable values
Name	This field is user defined, and is not used by the driver.	
Number	Must be unique.	
Address	FFFFFFFF.FFFF.FF.FF (Identificationnumber.Manufacturer.Version.Medium)	
Protocol	MBUSCIT	
Port name	Refers to the port previously defined in 'ports' form.	
Comment	This field is user defined and is not used by the driver.	

During selection individual positions of the secondary addresses can be occupied with wildcards (Fh). Such a wildcard means that this position will not be taken account of during selection, and that the selection will be limited to specific positions, in order to address complete groups of slaves (Multicasting). In the identification number each individual digit can be wildcarded by a wildcard nibble Fh while the fields for manufacturer, version and medium can be wildcarded by a wildcard byte FFh.

## 1.1.4.5 Exempel secondary addresses

a) 00102747.0442.02.02      b) 00102747.FFFF.FF.FF      c) FFFFFFFF47.FFFF.FF.FF

## 1.1.5 Reference: Data types

IO Device Type	Citect data format	Citect data types	Description/Special Usage/Limitations/ Valid Ranges
Value	Drx.Value	STRING	Read (Record range 0 – 500)
Value with decimal point	Drx./y.Value	STRING	Read (Record range 0 – 500)
Value	Drx.Value	INT	Read (Record range 0 – 500)
Value	Drx.Value	BYTE	Read (Record range 0 – 500)
Value	Drx.Value	LONG	Read (Record range 0 – 500)
Value	Drx.Value	BCD	Read (Record range 0 – 500)
Value	Drx.Value	LONGBCD	Read (Record range 0 – 500)
Value	Drx.Value	REAL	Read (Record range 0 – 500)
Value Information Block	Drx.Vib	STRING	Read (Record range 0 – 500)
Unit	Drx.Unit	STRING	Read (Record range 0 – 500)
Unit	Drx.Unit	INT	Read (Record range 0 – 500)
Tariff	Drx.Tariff	STRING	Read (Record range 0 – 500)
Tariff	Drx.Tariff	LONG	Read (Record range 0 – 500)
Storage	Drx.Storage	STRING	Read (Record range 0 – 500)
Storage	Drx.Storage	LONG	Read (Record range 0 – 500)
Function	Drx.Function	STRING	Read (Record range 0 – 500)
Function	Drx.Function	BYTE	Read (Record range 0 – 500)
Data type	Drx.Datatype	STRING	Read (Record range 0 – 500)
Data type	Drx.Datatype	BYTE	Read (Record range 0 – 500)
Data Record	Drx.Record	STRING	Read (Record range 0 – 500)
Manufacture specific data	Drx.a.b.Mspec	STRING	Read (Record range 0 – 500)
Manufacture specific data	Drx.a.Mspec	BYTE	Read (Record range 0 – 500)
Manufacture specific data	Drx.a.Mspec	INT	Read (Record range 0 – 500)
Manufacture specific data	Drx.a.Mspec	LONG	Read (Record range 0 – 500)
Header	Header	STRING	Read
Identification number	Idnumber	STRING	Read
Manufacturer	Manufacturer	STRING	Read
Manufacturer	Manufacturer	INT	Read
Version	Version	STRING	Read
Version	Version	BYTE	Read
Medium	Medium	STRING	Read
Medium	Medium	BYTE	Read
Access number	Accessnr	STRING	Read
Access number	Accessnr	BYTE	Read
Status	Status	STRING	Read
Status	Status	BYTE	Read
Status (bit)	Status.z	DIGITAL	Read
Signature	Signature	STRING	Read
Signature	Signature	INT	Read

Where:

- x* Record number
- y* Place for decimal point
- z* Bit number, 0 – 7
- a* Offset (The first character)
- b* Number of characters from offset character

## 1.1.6 Examples data types

### 1.1.6.1 Values

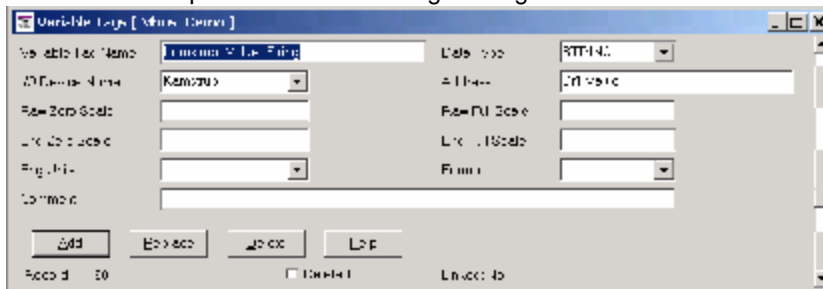
The driver can convert all M-Bus datatypes automatically to strings for you. This is especially good if you don't know the datatype the manufacturer want you shall use.

If you want to put in a decimal comma in a stringvalue you have normally to use Cicode or Vba. If you write eg Dr9/2.Value, the driver will put in a decimal comma for you in second position from right. Be aware if you want to use a disk- or memory driver Citect uses only the Mbuscit.dbf file to convert data types. These drivers can't know what to do with Dr9/2 and therefore you will not have any decimal comma. For this situation you have to use Cicode or Vba.

If you want to make some extra calculations from the values you have read out you have to first convert them via Cicode or Vba from string to the data type you want. M-Bus supports datatypes like INT64, BCD12 and this are not supported in Citect so there you have to use the Citect datatype STRING.

With the object Dr9.Datatype, explained below, you will have the information about which data type the manufacturer wants you shall use. With this information you can manually choose correct data type for Citect.

Here is an example of the Variable Tags dialog box.



Data Type      STRING  
Address         Dr9.Value  
Comment        The string value for datarecord 9  
Example         12345

Data Type      STRING  
Address         Dr9/2.Value  
Comment        The string value for datarecord 9 with decimalpoint  
Example         123,45

Data Type      BYTE  
Address         Dr9.Value  
Comment        The byte value for datarecord 9  
Example         23

If it is an int16 together with VIF6C (Timepoint [Date]) this datatype has to be a STRING. An INT will show the raw value not the date it self.

Data Type      INT  
Address         Dr9.Value  
Comment        The integer value for datarecord 9  
Example         12345

If it is an int32 together with VIF6D (Timepoint [Date & Time]) this datatype has to be STRING. A LONG will show the raw value not the date and time it self.

Data Type      LONG  
Address         Dr9.Value  
Comment        The integer value for datarecord 9  
Example         123445



Data Type      BCD  
 Address        Dr9.Value  
 Comment        The BCD2 or BCD4 value for datarecord 9  
 Example        99 or 9999

Data Type      LONGBCD  
 Address        Dr9.Value  
 Comment        The BCD6 or BCD8 value for datarecord 9  
 Example        999999 or 99999999

Data Type      REAL  
 Address        Dr9.Value  
 Comment        The real value for datarecord 9  
 Example        123,45

#### 1.1.6.2 Value Information Block

The driver is looking in the file mbuscit.ini, via the vifcode in the object, for the M-Bus standard texts. This also supports manufacturer specific texts via a manufacturer specific .ini file. If the driver will find that the vibcode is manufacturer specific it will try to find a file mbuscitXXX.ini there XXX is the manufacturer ID. In the file it will look after a section name build up from version.medium.ini text. As an example file the mbuscitABB.ini from ABB electricity meters are installed in Citect\Bin during the installation process.

If you want to control if the texts shown is correct you can set the driver specific parameter VibDebug=1. In this case you will see the M-Bus vifcodes and from them you can follow the text via mbuscit.ini and compare them via the M-Bus original documentation or the manufacture specific original documentation. The system with .ini files makes it also possible to change language for the text. If you do this be sure to take a backup copy of your own mbuscit.ini. The next installation will overwrite your personal mbuscit.ini if it still saved in Citect\Bin. If you want you can change your path for your .ini files via the driver specific parameter IniPath.

Data Type      STRING  
 Address        Dr9.Vib  
 Comment        The valueinformationblock for datarecord 9  
 Example        With driver specific parameter VibDebug=0; Current (mA\*10) L1  
                   With driver specific parameter VibDebug=1: FD5a FF01

#### 1.1.6.3 Unit

Datatype Unit has the range 0 – 1023

Data Type      STRING  
 Address        Dr9.Unit  
 Comment        Unit number for data record 9  
 Example        2

Data Type      INT  
 Address        Dr9.Unit  
 Comment        Unit number for data record 9  
 Example        2

#### 1.1.6.4 Tariff

Datatype Tariff has the range 0 – 1048575

Data Type	STRING
Address	Dr9.Tariff
Comment	Tariff for data record 9
Example	2

Data Type	LONG
Address	Dr9.Tariff
Comment	Tariff for data record 9
Example	2

#### 1.1.6.5 Storage

Datatype Tariff has the range 0 – 2199023255551

Data Type	STRING
Address	Dr9.Storage
Comment	Storage for data record 9
Example	2

Data Type	LONG
Address	Dr9.Storage
Comment	Storage for data record 9
Example	2

#### 1.1.6.6 Function

Data Type	STRING
Address	Dr9.Function
Comment	Function for data record 9
Example	Inst

Data Type	BYTE
Address	Dr9.Function
Comment	Function for data record 9
Example	0

#### 1.1.6.7 Data type

Data Type	STRING
Address	Dr9.Datatype
Comment	Data type for data record 9
Example	Int32

Data Type	BYTE
Address	Dr9.Datatype
Comment	Data type for data record 9
Example	4

**1.1.6.8 Data Record**

All the separate string types Unit, Tariff, Storage Datatype, Function, Value and Vib are added together to one string. With this type you have all information from each datarecord object you probably should need. Datarecord 0 doesn't exist therefore the driver use this to write the header for the object information. It's important to use a font type eg Courier New with the same width for each character.

Data Type      STRING  
 Address        Dr0.Record  
 Comment        Data record 0  
 Example  
 Record Unit    Tariff           Storage Datatype   Function                    Value Vib

Data Type      STRING  
 Address        Dr1.Record  
 Comment        Data record 1  
 Example  
 DR1           0           0                    0 BCD12    Inst                    237268 Energy (Wh\*10)

**1.1.6.9 Manufacturer specific data**

Data Type      STRING  
 Address        Dr9.4.8.Mspec  
 Comment        Manufacturer specific data in record 9  
 Example        8 byte from position 4

Data Type      BYTE  
 Address        Dr9.4.Mspec  
 Comment        Manufacturer specific data in record record 9  
 Example        1 byte or 8 bits

Data Type      INT  
 Address        Dr9.4.Mspec  
 Comment        Manufacturer specific data in record record 9  
 Example        2 bytes or 16 bits

Data Type      LONG  
 Address        Dr9.4.Mspec  
 Comment        Manufacturer specific data in record record 9  
 Example        4 bytes or 32 bits

**1.1.6.10 Header**

Data Type      STRING  
 Address        Header  
 Comment        Total Header  
 Example        00102886 SVM 09 Heat (inlet) 74 40 0000

**1.1.6.11 Identification number**

Data Type      STRING  
 Address        Idnumber  
 Comment        Identification number  
 Example        00102886

**1.1.6.12 Manufacturer**

Data Type     STRING  
Address        Manufacturer  
Comment       Short name for manufacturer  
Example        ABB

Data Type     INT  
Address        Manufacturer  
Comment        Value for manufacturer  
Example        1090 (0x0442)

**1.1.6.13 Version**

Data Type     STRING  
Address        Version  
Comment        Version  
Example        2

Data Type     BYTE  
Address        Version  
Comment        Version  
Example        2

**1.1.6.14 Medium**

Data Type     STRING  
Address        Medium  
Comment        Medium  
Example        Electricity

Data Type     BYTE  
Address        Medium  
Comment        Medium  
Example        2

**1.1.6.15 Access number**

Data Type     STRING  
Address        Accessnr  
Comment        Access number  
Example        2

Data Type     BYTE  
Address        Accessnr  
Comment        Access number  
Example        2

**1.1.6.16 Status**

Data Type     STRING  
Address        Status  
Comment       Status  
Example        2

Data Type     BYTE  
Address        Status  
Comment       Status  
Example        2

Data Type     DIGITAL  
Address        Status.0  
Comment       The first bit in status byte.  
Example

**1.1.6.17 Signature**

Data Type     STRING  
Address        Signature  
Comment       Signature  
Example        2

Data Type     INT  
Address        Signature  
Comment       Signature  
Example        2

### 1.1.7 Hints, Tips, and Frequently asked questions

- If the power disappears shortly from some RS232/M-Bus converters during the time a telegram is reading line faults can cause the COMX read thread to shutdown. This meant the driver would not recover after the fault. With the `-nt` option (no terminate), the thread is NOT shutdown. This allows the system to recover.
- For a low baudrate and many telegrams from one meter it can be shown some `#COM` at the screen. The reason for this `#COM` depends very often of that the IOserver get a request and it takes to long time to answer the question. Increase the Citect [LAN] timeout parameter to e.g. 20000 (20 sec); default is 8000 (8 sec).

## 1.2 Driver reference

	Detail
Driver name	MBUSCIT
Maximum array size <sup>1</sup>	256

### 1.2.1 Driver generated error codes

N/A

### 1.2.2 Parameters, options, and settings

#### 1.2.2.1 Standard Parameters

Parameter	Default	Allowable values	Description
Block (bytes)	256		
Delay (mS)	50		
MaxPending	2		
Polltime (mS)	0	0	
Timeout (mS)	2000	2000 to 30000	
Retry	3		
WatchTime (Sec)	30		

#### 1.2.2.2 Driver Specific Parameters which can be device/group specific

Parameter	Default	Allowable values	Description
CacheLiveTime (mS)	30000		How often the driver shall make a new request to the meter when Citect needs the value.
NrOfTelegrams	100		How many telegrams that will be requested in multitelegram. This parameter shall be used if the driver shall finish before M-Bus end 0x0F appear.
InitRead	1	1 or 0	Set to 0 and no SND_NKE will be sent before a REQ_UD2

#### 1.2.2.3 Driver Specific Parameters which is driver specific

Parameter	Default	Allowable values	Description
LeadingZeros	0	0 or 1	Set to 1 will show the leading zeros for datatype BCD
VibDebug	0	0 or 1	Set to 1 will show the VIB code for datatype Drx.Vib or Drx.Record is used
IniPath	Citect\Bin		Path for were Mbuscit.ini and manufacturer specific.ini should be saved.

<sup>1</sup> Equivalent to 'Maximum Request Length'

#### 1.2.2.4 MBUS Device/Group-specific Parameters

The MBUS driver supports the capability to apply different initialisation parameter values to specific I/O devices or groups of I/O devices. This means the user can specify:

- a) Global parameters that apply to all devices
- b) Channel (port) level parameters that apply to all devices on the specified port
- c) Group level parameters that apply to all devices in a specified group
- d) Device level parameters that apply only to the specified device

This feature can be implemented in the Citect.INI file for the following MBUS parameters:

CacheLiveTime  
NrOfTelegrams  
InitRead

To set parameter values for a particular group or device, you put a full stop (period) immediately after the name of the driver where it appears in the Citect.INI file, followed by the name of the particular port or group you want to specify a parameter setting for. For example:

[MBUSCIT.<Port\_Name>] applies the parameter settings to the specified port  
[MBUSCIT.<Group\_Name>] applies the parameter settings to the specified group  
[MBUSCIT.<Port\_Name>.<IODevice\_Name>] applies to the specified device  
Any parameters you then define in the following section of the Citect.INI file will relate only to the specified device or device group.

#### Example

The following Citect INI file format is an example of how the 'NrOfTelegrams' parameter could be specified differently for different I/O devices communicating using the MBUSCIT Protocol. Assume that two ports are used: PORT1 and PORT2.

PORT1 has three I/O devices attached: DEV1A DEV1B DEV1C  
PORT2 also has three devices:DEV2A DEV2B DEV2C

Assume that the user has specified that DEV1C and DEV2C belong to GROUPZ.  
The Citect INI file contains the following entries:

```
[MBUSCIT]
NrOfTelegrams =1
[MBUSCIT.PORT1]
NrOfTelegrams =2
[MBUSCIT.PORT2]
NrOfTelegrams =2
[MBUSCIT.GROUPZ]
NrOfTelegrams =3
[MBUSCIT.PORT1.DEV1A]
NrOfTelegrams =1
[MBUSCIT.PORT2.DEV2B]
NrOfTelegrams =4
```

The resultant NrOfTelegrams for the IO Devices will be as follows:

```
DEV1A:1      as a result of [MBUSCIT.PORT1.DEV1A]
DEV1B:2      as a result of [MBUSCIT.PORT1]
DEV1C:3      as a result of [MBUSCIT.GROUPZ]
DEV2A:2      as a result of [MBUSCIT.PORT2]
DEV2B:4      as a result of [MBUSCIT.PORT2.DEV2B]
DEV2C:3      as a result of [MBUSCIT.GROUPZ]
```



NOTE: As the above example shows, there is a hierarchy that determines the outcome of such settings. In simple terms, specific parameter settings overwrite general level settings. Therefore, parameters written in the scope of I/O devices will overwrite those set for groups; parameters set for groups will overwrite global settings, etc.

## 1.2.3 Advanced

### 1.2.3.1 Driver generated statistics

Number	Label	Description
0	Frames Transmitted	Frames Transmitted
1	Frames Received	Normal frames received
2	SingleFrames Received	Frames received with 0xE5
3	Write Requests	Write Requests
4	Read Request	Read Request
5	Frames Accepted	Frames Accepted
6	Frames With Error	Frames With Error
7	Bad Check Sum	Bad Check Sum
8	Communication Error	Errors as parity error, break etc
9		
10		
11		
12		
13		
14		
15		
16		
17		
18	I/O Devices in license	Size of installed I/O Devices
19	I/O Devices at this port	I/O Devices configured at this port (Memory and disk drives are not counted)

### 1.2.3.2 Debug messages

The debug messages are from the standard driver TraceTX and TraceRx functions and can be read out from syslog.dat.

#### 1.2.3.2.1 Initialisation of primary address 90

```
Sat Sep 28 16:14:40 2002 04:29:46.808 Transmit Length 5
10 40 5A 9A 16 .@Z..
Sat Sep 28 16:14:40 2002 04:29:46.860 Received Length 1
E5 .
```

#### 1.2.3.2.2 Initialisation of secondary address 00102747.0442.02.02

```
Sat Sep 28 16:14:40 2002 04:29:46.900 Transmit Length 17
68 0B 0B 68 53 FD 52 47 27 10 00 42 04 02 02 6A h..hS.RG'..B...j
16 .
Sat Sep 28 16:14:41 2002 04:29:47.031 Received Length 1
E5 .
```

## 1.2.3.2.3 Reading single telegram with primary address 90

```

Sat Sep 28 16:23:11 2002 04:38:17.841 Transmit Length 5
10 7B 5A D5 16 .{Z..
Sat Sep 28 16:23:12 2002 04:38:18.591 Received Length 152
68 92 92 68 08 5A 72 90 02 25 00 2D 2C 01 04 15 h..h.Zr..%.-,...
00 00 00 0C 05 73 33 00 00 0C 13 90 78 00 00 0C .....s3.....x...
22 81 10 00 00 0C 59 98 16 00 00 0C 5D 68 19 00 ". ....Y....]h..
00 0C 61 00 00 00 00 0C 2D 00 00 00 00 00 0C 3B 00 ..a.....-.....;
00 00 00 4C 05 00 00 00 00 4C 13 00 00 00 00 42 ...L.....L.....B
6C 5F 05 0F 90 02 25 00 00 00 00 00 00 00 00 00 l_.....%.....
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 07 31 04 00 01 01 .....1....
06 02 28 09 02 00 53 51 00 00 37 22 00 00 00 00 ..(...SQ..7"....
00 00 00 00 00 00 84 16 .....

```

## 1.2.3.2.4 Reading multi telegram with secondary address 00102747.0442.02.02

```

Sat Sep 28 16:23:52 2002 04:38:57.989 Transmit Length 17
68 0B 0B 68 53 FD 52 47 27 10 00 42 04 02 02 6A h..hS.RG'..B...j
16 .
Sat Sep 28 16:23:52 2002 04:38:58.117 Received Length 1
E5 .
Sat Sep 28 16:23:52 2002 04:38:58.117 Transmit Length 5
10 5B FD 58 16 .[.X.
Sat Sep 28 16:23:52 2002 04:38:58.568 Received Length 81
68 4B 4B 68 08 02 72 47 27 10 00 42 04 02 02 48 hKKh..rG'..B...H
20 00 00 0E 04 54 71 23 00 00 00 00 8E 10 04 54 71 ....Tq#.....Tq
23 00 00 00 8E 20 04 00 00 00 00 00 00 01 FF 13 #....
01 0C FF 12 01 00 00 00 07 FD 17 40 00 00 28 00 .....@..(
00 00 00 01 FF 18 66 1F 00 00 00 00 00 00 00 34 .....f.....4
16 .
Sat Sep 28 16:23:52 2002 04:38:58.568 Transmit Length 5
10 7B FD 78 16 .{.x.
Sat Sep 28 16:23:53 2002 04:38:59.149 Received Length 110
68 68 68 68 08 02 72 47 27 10 00 42 04 02 02 49 hhhh..rG'..B...I
20 00 00 04 29 00 00 00 00 04 A9 FF 01 00 00 00 .....
00 04 A9 FF 02 00 00 00 00 04 A9 FF 03 00 00 00 .....
00 02 FD C8 FF 01 B8 08 02 FD C8 FF 02 B5 08 02 .....
FD C8 FF 03 BD 08 0A FD DA FF 01 00 00 0A FD DA .....
FF 02 00 00 0A FD DA FF 03 00 00 0A FF 59 00 50 .....Y.P
1F 00 00 00 00 00 00 00 00 00 00 00 00 F5 16 .....
Sat Sep 28 16:23:53 2002 04:38:59.149 Transmit Length 5
10 5B FD 58 16 .[.X.
Sat Sep 28 16:23:53 2002 04:38:59.680 Received Length 100
68 5E 5E 68 08 02 72 47 27 10 00 42 04 02 02 4A h^^h..rG'..B...J
20 00 00 02 FF 60 00 00 02 FF C2 FF 02 F6 FF 02 .....`.....
FF C2 FF 03 F4 FF 02 FF CA FF 01 00 00 02 FF CA .....
FF 02 00 00 02 FF CA FF 03 00 00 81 80 40 FD 1B .....@..
00 C1 80 40 FD 1B 00 8E 80 40 FD 61 00 00 00 00 .....@.....a....
00 00 81 40 FD 1A 00 0F 00 00 00 00 00 00 00 00 .....@.....
00 00 69 16 ...i.

```

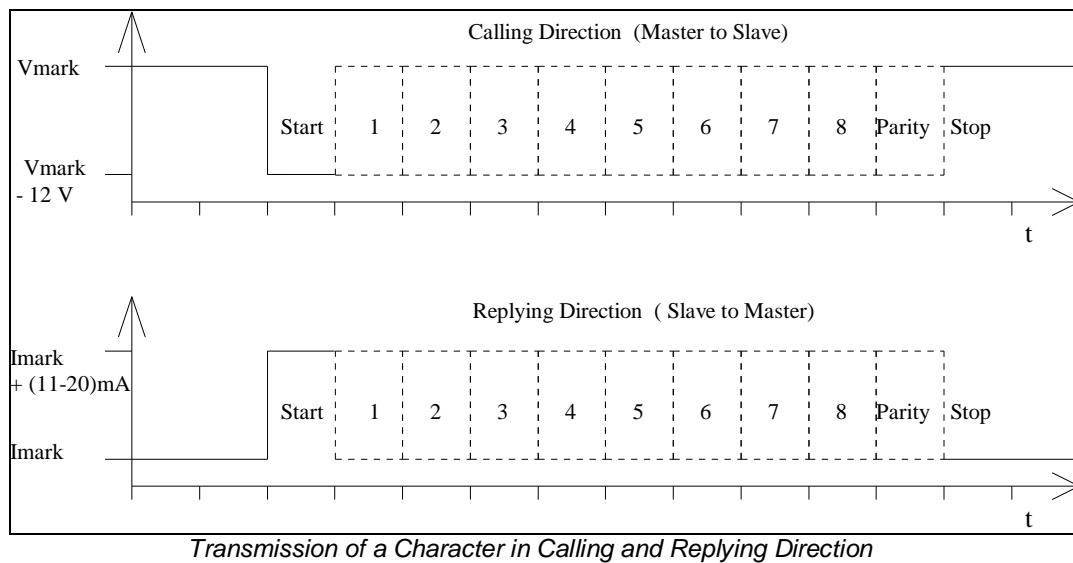
## 2. Analysis

### 2.1 M-Bus overview

This chapter is a part of the official "The M-Bus: A Documentation" Version 4.8

#### 2.1.1 Transmission Parameters

The M-Bus protocol uses asynchronous serial bit transmission, in which the synchronization is implemented with start and stop bits for each character. There must be no pauses within a telegram, not even after a stop bit. Since quiescence on the line corresponds to a 1 (Mark), the start bit must be a Space, and the stop bit a Mark. In between the eight data bits and the even parity bit are transmitted, ensuring that at least every eleventh bit is a Mark. The bits of data are transmitted in ascending order, i.e. the bit with the lowest value (LSB = least significant bit) is the first one to be found on the line. The transmission takes place in half duplex and with a data rate of at least 300 Baud. In the figure the transmission of a byte in calling and replying direction is represented.



## 2.1.2 Telegram Format

According to IEC 870-5, three different data integrity classes (I1, I2 and I3) are envisaged for the transmission of remote control data. For the M-Bus protocol of the data link layer the format class FT 1.2 is used, which is contained in the data integrity class I2, which specifies a Hamming Distance of four.

The format class FT 1.2 specifies three different telegram formats, which can be recognized by means of special start characters. Below the telegram formats used for the M-Bus will now be explained.

Single Character	Short Frame	Long Frame
E5h	Start 10h	Start 68h
	C Field	L Field
	A Field	L Field
	Check Sum	Start 68h
	Stop 16h	C Field
		A Field
		CI Field
		User Data (0-252 Byte)
		Check Sum
		Stop 16h

*Telegram Formats*

- **Single Character**  
This format consists of a single character E5h, and serves to acknowledge receipt of transmissions.
- **Short Frame**  
This format with a fixed length of five characters begins with the start character 10h, and besides the C and A fields includes the check sum and the stop character 16h.
- **Long Frame**  
With the long frame, after the start character 68h, the length field (L field) is first transmitted twice, followed by the start character once again. After this, there follow the function field (C field), the address field (A field) and the control information field (CI field). After the user data inputs, the check sum are transmitted, which is built up over the same area as the length field, and in conclusion the stop character 16h is transmitted.

## 2.1.3 Meaning of the Fields

In this section, the fields used for telegram formats will be explained. These all have a length of 1 Byte, corresponding to 8 bits.

### 2.1.3.1 L field (Length Field)

The L field gives the quantity of the user data inputs plus 3 (for C,A,CI). It is transmitted twice in Long Frame format

### 2.1.3.2 C Field (Control Field, Function Field)

Besides labeling the functions and the actions caused by them, the function field specifies the direction of data flow, and is responsible for various additional tasks in both the calling and replying directions.

Bit Number	7	6	5	4	3	2	1	0
Calling Direction	0	1	FCB	FCV	F3	F2	F1	F0
Reply Direction	0	0	ACD	DFC	F3	F2	F1	F0

*Coding of the Control Field*

The highest value (most significant) bit is reserved for future functions, and at present is allocated the value 0; bit number 6 is used to specify the direction of data flow. The frame count bit FCB indicates successful transmission procedures (i.e. those that have been replied to or acknowledged), in order to avoid transmission loss or multiplication. If the expected reply is missing or reception is faulty, the master sends again the same telegram with an identical FCB, and the slave replies with the same telegram as previously. The master indicates with a 1 in the FCV bit (frame count bit valid), that the FCB is used. When the FCV contains a "0", the slave should ignore the FCB. (The driver has always 1 in the FCV bit)

The bits 0 to 3 of the control field code the true function or action of the message. The table shows the function codes used in the calling and the replying directions.

Name	C-field (binary)	C-field (hex)	Telegram	Description
SND_NKE	0100 0000	40	Short Frame	Initialization of Slave
SND_UD	01F1 0011	53/73	Long Frame	Send User Data to Slave
REQ_UD2	01F1 1011	5B/7B	Short Frame	Request for Class 2 Data
REQ_UD1	01F1 1010	5A/7A	Short Frame	Request for Class 1 Data*
RSP_UD	00AD 1000	08	Long Frame	Data Transfer from Slave to Master after Request

*Control Codes of the M-Bus Protocol (F = FCB-Bit, A = ACD-Bit, D = DFC-Bit)*

*\*The driver does not support Class 1 Data*

### 2.1.3.3 A Field (Address Field)

The address field serves to address the recipient in the calling direction, and to identify the sender of information in the receiving direction. The size of this field is one Byte, and can therefore take values from 0 to 255. The addresses 1 to 250 can be allocated to the individual slaves, up to a maximum of 250. Unconfigured slaves are given the address 0 at manufacture, and as a rule are allocated one of these addresses when connected to the M-Bus. The addresses 254 (FEh) and 255 (FFh) are used to transmit information to all participants (Broadcast). With address 255 none of the slaves reply, and with address 254 all slaves reply with their own addresses. The latter case naturally results in collisions when two or more slaves are connected, and should only be used for test purposes. The address 253 (FDh) indicates that the secondary addressing has been performed in the Network Layer instead of Data Link Layer. The remaining addresses 251 and 252 have been kept for future applications.

### 2.1.3.4 CI Field (control information field)

The CI-field codes the type and sequence of application data to be transmitted in the frame. The EN1434-3 defines two possible data sequences in multibyte records. Bit two (counting begins with bit 0, value 4), which is called M bit or Mode bit, in the CI-field gives an information about the used byte sequence in multi-byte data structures. If the Mode bit is not set (Mode 1), the least significant byte of multi-byte record is transmitted first, otherwise (Mode 2) the most significant byte. The M-Bus user group recommends using only the Mode 1 in future applications. The driver is supporting only Mode 1.

### 2.1.3.5 Check Sum

The Check Sum serves to recognize transmission and synchronization faults, and is configured from specific parts of the telegram. The Check Sum is calculated from the arithmetical sum of the data, without taking carry digits into account.

## 2.1.4 Communication Process

The Data Link Layer uses two kinds of transmission services:

- Send/Confirm: SND/CON
- Request/Respond: REQ/RSP

### 2.1.4.1 Send/Confirm Procedures

#### SND\_NKE

This procedure serves to start up after the interruption or beginning of communication. The value of the frame count bit FCB is adjusted in master and slave, i.e. the first master telegram with FCB=1 after SND\_NKE contains a FCB=1. The slave responds to a correctly received SND\_NKE with an acknowledgment consisting of a single character (E5h).

#### SND\_UD

With this procedure the master transfers user data to the slave. The slave can either confirm the correct receipt of data with a single character acknowledge (E5h), or by omitting a confirmation signal that it did not receive the telegram correctly.

### 2.1.4.2 Request/Respond Procedures

#### REQ\_UD2 → RSP\_UD

The master requests data from the slave using the REQ\_UD2 telegram. The slave can either transfer its data with RSP\_UD, or give no response indicating that the REQ\_UD2 telegram has not been received correctly or that the address contained in the telegram does not match.

## 2.1.5 Selection and Secondary Addressing

The driver supports secondary addressing. The secondary addressing starts with a slave selection command:

68h	0Bh	0Bh	68h	53h	FDh	52h	ID1-4	Man 1-2	Gen	Med	CS	16h
-----	-----	-----	-----	-----	-----	-----	-------	---------	-----	-----	----	-----

*Structure of a telegram for selecting a slave (mode 1)*

The master sends a SND\_UD with the control information 52h to the address 253 (FDh) and fills the specific meter secondary address (identification number, manufacturer, version and medium) with the values of the slave which is to be addressed. The address FDh and the control information 52h are the indication for the slaves to compare the following secondary addresses with their own, and to change into the selected state should they agree. In this case the slave must answer the selection with an acknowledgement (E5h), otherwise the slave doesn't send an answer. "Selected state" means that this slave will be addressed with the bus address 253 (FDh). In other words the network layer has associated this slave with the address 253 (FDh).

During selection individual positions of the secondary addresses can be occupied with wildcards (Fh). Such a wildcard means that this position will not be taken account of during selection, and that the selection will be limited to specific positions, in order to address complete groups of slaves (Multicasting). In the identification number each individual digit can be wildcarded by a wildcard nibble Fh while the fields for manufacturer, version and medium can be wildcarded by a wildcard byte FFh.

The state of the selection remains unchanged until the slave is deselected with a selection command with non-matching secondary addresses, or a SND\_NKE to address 253. The slave, who uses mode 1 for multibyte records, will be selected by a telegram with the CI-Field 52h and the right secondary address, but it will be deselected by a telegram with the CI-Field 56h and any secondary address.

## 2.1.6 Applications of the FCB-mechanism

### 2.1.6.1 Multi-telegram answers (RSP\_UD) from slave to master

If a total answer sequence from a slave will not fit into a single RSP\_UD (respond user data) telegram from the slave to the master, the master signals by a toggled FCB-Bit together with a set FCV-Bit in the next REQ\_UD (Request user data) telegram that its link layer has properly received the last RSP\_UD-telegram from the slave. The slave answers to a REQ\_UD-request with toggled FCB-Bit with a set FCV-bit from the master with a RSP\_UD containing the next link layer telegram section of a multi-telegram answer, otherwise it will repeat the last telegram. Note that a slave with a single RSP\_UD-telegram may simply ignore the FCB in the REQ\_UD2-telegram and send always the same (single) telegram. Note also that a slave with exactly two (sequential) RSP\_UD-answer telegrams may simply use the FCB of the REQ\_UD2 to decide which of both telegrams should be transmitted. Thus a slave with one or two (sequential) RSP\_UD answer-telegrams does not require an internal "Last-REQ\_UD2-FCB"-image bit. A slave with three or more (sequential) RSP\_UD answer telegrams requires such an internal memory bit. Note that such an internal memory bit for the RSP\_UD-direction must be independent of an possible additional internal memory bit for the SND\_UD direction.

### 2.1.6.2 Frozen answer telegrams from slave to master

In some instances a slave will freeze the data of its last RSP\_UD answer telegram into an additional temporary storage and will repeat these previously frozen RSP\_UD answer, if the FCB has not been toggled. After the reception of a toggled FCB-Bit with a set FCV-Bit or after the reception of a REQ\_UD2 with the FCV-Bit cleared, the slave will generate a next answer telegram reflecting the current state of all its data instead of repeating the data values frozen at the first REQ\_UD2 attempt with toggled FCB.

## 2.1.7 Variable Data Structure

The CI-Field codes 72h are used to indicate the variable data structure in long frames (RSP\_UD).

Fixed Data Header	Variable Data Blocks (Records)	MDH	Manufacturer specific data
12 Byte	Variable number of bytes	1 Byte	Variable number of bytes

*Variable Data Structure in Reply Direction*

## 2.1.8 Fixed Data Header

The first twelve bytes of the user data consist of a block with a fixed length and structure

Identification No	Manufacturer	Version	Medium	Access No	Status	Signature
4 Byte	2 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Byte

*Fixed Data Block*

- **Identification Number** is a customer number, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00000000 to 99999999.
- **Manufacturer** is coded unsigned binary with 2 bytes. This manufacturer ID is calculated from the ASCII code of EN 61107 manufacturer ID (three uppercase letters) with the following formula:

$\begin{aligned} \text{IEC 870 Man. ID} &= && [\text{ASCII}(1\text{st letter}) - 64] \cdot 32 \cdot 32 \\ &+ && [\text{ASCII}(2\text{nd letter}) - 64] \cdot 32 \\ &+ && [\text{ASCII}(3\text{rd letter}) - 64] \end{aligned}$
--

- **Version** specifies the generation or version of this counter and depends on the manufacturer.



- **Medium** is coded with a whole byte. The text is readout from Mbuscit.ini

Medium	Code hex.
Other	00
Oil	01
Electricity	02
Gas	03
Heat (Volume measured at return temperature: outlet)	04
Steam	05
Hot Water	06
Water	07
Heat Cost Allocator.	08
Compressed Air	09
Cooling load meter (Volume measured at return temperature: outlet)	0A
Cooling load meter (Volume measured at flow temperature: inlet)	0B
Heat (Volume measured at flow temperature: inlet)	0C
Heat / Cooling load meter	0D
Bus / System	0E
Unknown Medium	0F
Reserved	10 to 15
Cold Water	16
Dual Water	17
Pressure	18
A/D Converter	19
Reserved	20 to FF

- **Access Number** has unsigned binary coding, and is increased by one after each RSP\_UD from the slave.
- **Status** field are used to indicate application errors.

Bit	Meaning
0	Application busy
1	Any application error
2	Power low
3	Permanent error
4	Temporary error
5	Specific to manufacturer
6	Specific to manufacturer
7	Specific to manufacturer

- **Signature** remains reserved for future encryption applications, and until then is allocated the value 00 00 h.

### 2.1.9 Variable Data Blocks

The data, together with information regarding coding, length and the type of data is transmitted in data records. As many blocks of data can be transferred as there is room for, within the maximum data length of 255 Bytes, and taking account of the C, A, and CI fields, the fixed data block. The upper limit for characters in the variable data blocks is thus 240 byte. The Usergroup recommends a maximum total telegram length of 255 bytes (234 bytes for variable data blocks) to avoid problems in modem communication. The manufacturer data header (MDH) is made up by the character 0Fh or 1Fh and indicates the beginning of the manufacturer specific part of the user data and should be omitted, if there is no manufacturer specific data.

Data Record Header (DRH)				Data
Data Information Block (DIB)		Value Information Block (VIB)		
DIF	DIFE	VIF	VIFE	
1 Byte	0 - 10 Bytes	1 Byte	0 - 10 Bytes	0 - n Bytes

*Structure of a Data Record (transmitted from left to right)*

Each data record consists of a data record header (DRH) and the actual data. The DRH in turn consists of the DIB (data information block) to describe the length, type and coding of the data, and the VIB (value information block) to give the value of the unit and the multiplier.

### 2.1.10 Data Information Block

The DIB contains at least of one byte (DIF, data information field), and can be extended by a maximum of ten DIFE's (data information field extensions). The following information is contained in a DIF:

Bit 7	6	5	4	3	2	1	0
Extension Bit	LSB of storage number	Function Field		Data Field : Length and coding of data			

*Coding of the Data Information Field (DIF)*

The **function field** gives the type of data as follows:

Code	Description	Drx.Function
00	Instantaneous value	Inst
01	Maximum value	Max
10	Minimum value	Min
11	Value during error state	Error

The **data field** shows how the data from the master must be interpreted in respect of length and coding. The following table contains the possible coding of the data field:

Length in Byte	Code	Meaning	Drx.Datatype	Citect Data type
0	0000	No data	No data	-
1	0001	8 Bit Integer	Int8	STRING/BYTE
2	0010	16 Bit Integer	Int16	STRING/INT
3	0011	24 Bit Integer	Int24	STRING/LONG
4	0100	32 Bit Integer	Int32	STRING/LONG
4	0101	32 Bit Real	Real32	STRING/REAL
6	0110	48 Bit Integer	Int48	STRING
8	0111	64 Bit Integer	Int64	STRING
0	1000	Selection for Readout	Selection	-
1	1001	2 digit BCD	BCD2	STRING/BCD
2	1010	4 digit BCD	BCD4	STRING/BCD
3	1011	6 digit BCD	BCD6	STRING/LONGBCD
4	1100	8 digit BCD	BCD8	STRING/LONGBCD
n	1101	Variable length*	Variable	-
6	1110	12 digit BCD	BCD12	STRING
n	1111	Special Functions	Special	-

*Coding of the data field*

#### Special Functions (data field = 1111b):

DIF	Function
0Fh	Start of manufacturer specific data structures to end of user data
1Fh	Same meaning as DIF = 0Fh + More records follow in next telegram
2Fh	Idle Filler (not to be interpreted), following byte = DIF
3Fh..6Fh	Reserved
7Fh	Global readout request (all storage#, units, tariffs, function fields)

If data follows after DIF=0Fh or 1Fh these are manufacturer specific data records. The number of bytes in these manufacturer specific data can be calculated with the L-Field. The DIF 1Fh signals a request from the slave to the master to readout the slave once again. The master must readout the slave until there is no DIF=1Fh inside the respond telegram (multi telegram readout).

The Bit 6 of the DIF serves to give the **storage number** of the data concerned, and the slave can in this way indicate and transmit various stored counter states or historical values, in the order in which they occur. This bit is the least significant bit of the storage number and allows therefore the storage numbers 0 and 1 to be given without further DIFE's. In this way the storage number 0 stands for the actual value. If higher storage numbers are needed, the slave allows a DIFE to follow the DIF, and indicates this by setting the extension bit.

Each DIFE (maximum ten) contains again an extension bit to show that a further DIFE is being sent. Besides giving the next most significant bit of the storage number, this DIFE allows the transmission of informations about the **tariff** and the **subunit** of the device from which the data come. In this way, exactly as with the storage number, the next most significant bit or bits will be transmitted. The figure shows the structure of a DIFE:

Bit 7	6	5	4	3	2	1	0
Extension Bit	(Device) Unit	Tariff		Storage Number			

*Coding of the Data Information Field Extension (DIFE)*

With the maximum of ten DIFE's that are provided, there are 41 bits for the storage number, 20 bits for the tariff, and 10 bits for the subunit of the meter. There is no application conceivable in which this immense number of bits could all be used.

No	DIB	Bit 7	6	5	4	3	2	1	0
0	DIF	E	S	F	F	D	D	D	D
1	DIFE	E	U	T	T	S	S	S	S
2	DIFE	E	U	T	T	S	S	S	S
3	DIFE	E	U	T	T	S	S	S	S
4	DIFE	E	U	T	T	S	S	S	S
5	DIFE	E	U	T	T	S	S	S	S
6	DIFE	E	U	T	T	S	S	S	S
7	DIFE	E	U	T	T	S	S	S	S
8	DIFE	E	U	T	T	S	S	S	S
9	DIFE	E	U	T	T	S	S	S	S
10	DIFE	E	U	T	T	S	S	S	S

*Detailed Coding of the Data Information Field Extension (DIFE)*

Where: E = Extension Bit  
 S = Storage Number (0 – 2199023255551)  
 F = Function Field  
 D = Data Field: Length and coding of data  
 U = Unit (Device) (0 – 1023)  
 T = Tariff (0 – 1048575)

### 2.1.11 Value Information Block (VIB)

After a DIF or DIFE without a set extension bit there follows the VIB (value information block). This consists at least of the VIF (value information field) and can be expanded with a maximum of 10 extensions (VIFE). The VIF and also the VIFE's show with a set MSB that a VIFE will follow. In the value information field VIF the other seven bits give the unit and the multiplier of the transmitted value.

Bit 7	6	5	4	3	2	1	0
Extension Bit	Unit and multiplier (value)						

*Coding of the Value Information Field (VIF)*

**There are five types of coding depending on the VIF:**

#### 1) Primary VIF: 00h .. 7Ah

The unit and multiplier is taken from the table for primary VIF.

#### 2) Plain-text VIF: 7Ch (Not supported in the driver)

In case of VIF = 7Ch / FCh the true VIF is represented by the following ASCII string with the length given in the first byte. Please note that the byte order of the characters after the length byte depends on the used byte sequence. This plain text VIF allows the user to code units that are not included in the VIF tables.

**3) Linear VIF-Extension: FDh or FBh**

In case of VIF = FDh or VIF = FBh the true VIF is given by the next byte and the coding is taken from the table for secondary VIF. This extends the available VIF's by another 256 codes.

**4) Any VIF: 7Eh / FEh (Not supported in the driver)**

This VIF-Code can be used in direction master to slave for readout selection of all VIF's. See chapter 6.4.3.

**5) Manufacturer specific: 7Fh / FFh**

In this case the remainder of this data record including VIFE's has manufacturer specific coding.

The **VIFE** can be used for actions which shall be done with the data (master to slave), for reports of application errors (slave to master) and for an enhancement of the VIF (orthogonal VIF). The last feature allows setting VIF's into relation to the base physical units (e.g. VIF=10 liter, VIFE= per hour) or coding indirect units, pulse increments and change speeds.

In case of **VIFE** = FFh the next **VIFE's** and the data of this block are manufacturer specific, but the VIF is coded as normal.

After a VIF or VIFE with an extension bit of "0", the value information block is closed, and therefore also the data record header, and the actual data follow in the previously given length and coding.

**2.1.12 Manufacturer Specific Data Block**

The MDH consists of the character 0Fh or 1Fh (DIF = 0Fh or 1Fh) and indicates that all following data are manufacturer specific. When the number of bytes given in the length field of the connection protocol has not yet been used up, then manufacturer specific data follow this character, whose coding is left to the manufacturer. The length of this data is calculated from the L-Field minus the length of the so-called standard data (C-Field, A-Field, CI-Field and the data up to and including the data block 0Fh).

In case of MDH = 1Fh the slave signals to the master that it wants to be readout once again (multitelegram readouts). The master must readout the data until there is no MDH = 1Fh in the respond telegram.

## 2.2 DBFs

### 2.2.1 Help.dbf

TYPE	DATA	FILTER
PROTOCOL	MBUSCIT	

### 2.2.2 Mbuscit.dbf Entries

TEMPLATE	UNIT_TYPE	RAW_TYPE	BIT_WIDTH	LOW	HIGH	COMMENT
DR%<16[/%+U%*256] .VALUE	0x00000001	7	512	0	500	String (12BCD)
DR%<16.VALUE	0x00000001	1	16	0	500	Integer
DR%<16.VALUE	0x00000001	8	8	0	500	Byte
DR%<16.VALUE	0x00000001	4	32	0	500	Long
DR%<16.VALUE	0x00000001	3	16	0	500	BCD (2BCD, 4BCD)
DR%<16.VALUE	0x00000001	5	32	0	500	LongBCD (6BCD, 8BCD)
DR%<16.VALUE	0x00000001	2	32	0	500	Real
DR%<16.VIB	0x00000002	7	1024	0	500	String
DR%<16.UNIT	0x00000003	7	128	0	500	String
DR%<16.UNIT	0x00000003	1	16	0	500	Integer
DR%<16.TARIFF	0x00000004	7	128	0	500	String
DR%<16.TARIFF	0x00000004	4	32	0	500	Long
DR%<16.STORAGE	0x00000005	7	128	0	500	String
DR%<16.STORAGE	0x00000005	4	32	0	500	Long
DR%<16.FUNCTION	0x00000006	7	128	0	500	String
DR%<16.FUNCTION	0x00000006	8	8	0	500	Byte
DR%<16.DATATYPE	0x00000007	7	128	0	500	String
DR%<16.DATATYPE	0x00000007	8	8	0	500	Byte
DR%<16.RECORD	0x00000008	7	2040	0	500	String
DR%<16.%+U%*256.%+U%*256.MSPEC	0x00000009	7	2040	0	500	String
DR%<16.%+U%*256.MSPEC	0x00000009	8	8	0	500	Byte
DR%<16.%+U%*256.MSPEC	0x00000009	1	16	0	500	Integer
DR%<16.%+U%*256.MSPEC	0x00000009	4	32	0	500	Long
HEADER	0x00000010	7	1024	0	0	String
IDNUMBER	0x00000011	7	128	0	0	String
MANUFACTURER	0x00000012	7	32	0	0	String
MANUFACTURER	0x00000012	1	16	0	0	Integer
VERSION	0x00000013	7	128	0	0	String
VERSION	0x00000013	8	8	0	0	Byte
MEDIUM	0x00000014	7	128	0	0	String
MEDIUM	0x00000014	8	8	0	0	Byte
ACCESSNR	0x00000015	7	128	0	0	String
ACCESSNR	0x00000015	8	8	0	0	Byte
STATUS	0x00000016	7	128	0	0	String
STATUS[.%u]	0x00000016	8	8	0	0	Byte
SIGNATURE	0x00000017	7	128	0	0	String
SIGNATURE	0x00000017	1	16	0	0	Integer

### 2.2.3 Protidir.dbf

TAG	FILE	BIT_BLOCK	MAX_LENGTH	OPTIONS
MBUSCIT	MBUSCIT	16	256	0x0ff

## 2.3 Development resources

### 2.3.1 Contacts

#### 2.3.1.1 Technical support for Sweden, Norway, Denmark, Finland

[www.beijer.se](http://www.beijer.se)  
[support@beijer.se](mailto:support@beijer.se)

#### 2.3.1.2 Technical support for countries outside Sweden, Norway, Denmark, Finland

PiiGAB, Processinformation i Göteborg AB  
 Anders Carlssons gata 14  
 S-417 55 Göteborg  
[info@piigab.se](mailto:info@piigab.se)  
[www.piigab.se](http://www.piigab.se)

#### 2.3.1.3 General questions

PiiGAB Processinformation i Göteborg AB  
 Anders Carlssons gata 14  
 S-417 55 Göteborg  
[info@piigab.se](mailto:info@piigab.se)  
[www.piigab.se](http://www.piigab.se)

### 2.3.2 Documents

The M-Bus: A Documentation  
 Version 4.8 November 11, 1997

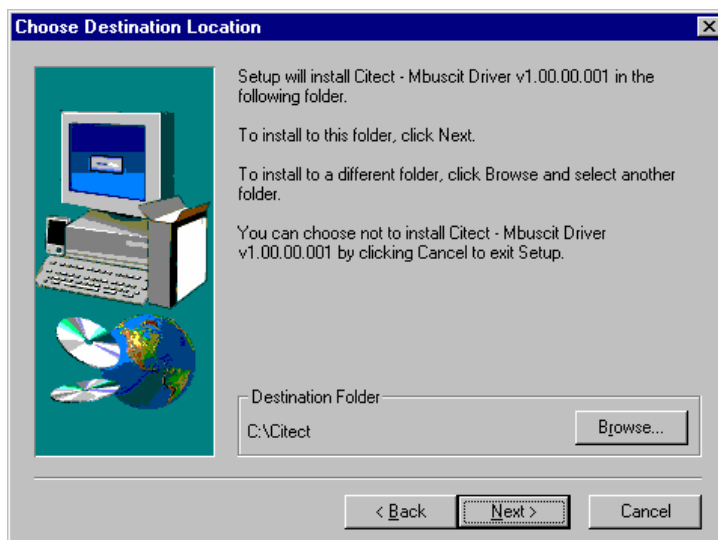
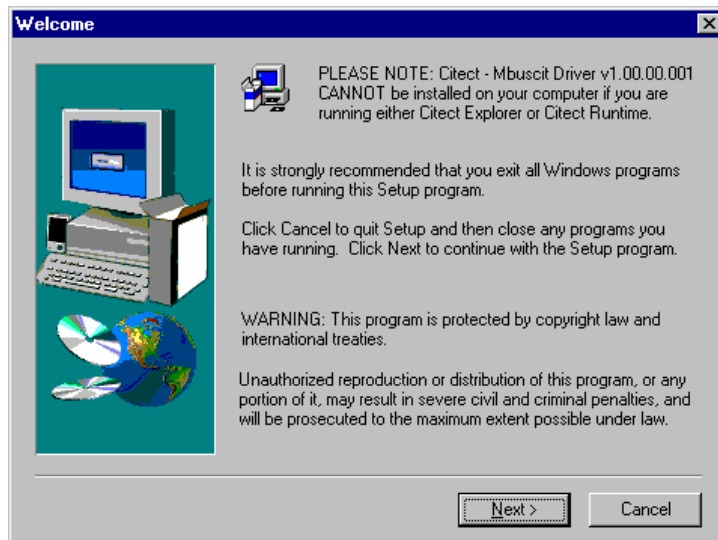
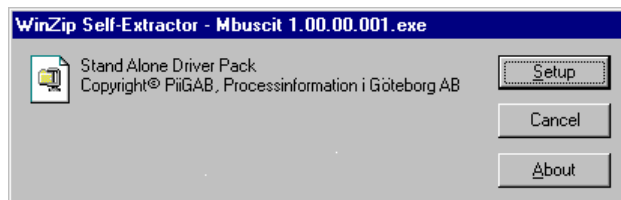
### 2.3.3 Driver Version History

Driver Version	Modified by	Comments
1.00.00.001	Bertil Göransson	Initial release
1.00.00.002	Bertil Göransson	Reverse of LSB and MSB in BCD6 to LONGBCD
1.00.00.003	Bertil Göransson	Filling up the two high bytes with 0x00 in int48 to string
1.00.00.004	Bertil Göransson	Changing in cache for Dial
1.01.00.001	Bertil Göransson	New license system udp support, [MBUSCIT.PORT] in Citect.ini didn't work correct.
1.01.01.001	Bertil Göransson	Stop Conntimer in Cpu. Extra Trace in ConnTimeout and Transmit for Channel Offline
1.01.02.001	Bertil Göransson	When BCD negative or outside 0-9 = 0. Previous error DRIVER_BAD_DATA.
1.01.03.001	Bertil Göransson	Build for Citect V6

## 3. Appendix 1 - Installation

### 3.1 Beginning of installation

You are starting the installation by activating the self-extracting .exe file Mbuscit1.00.00.001.exe. The information should be self-explaining during the installation process. But anyway here is little short information about it.





## 3.2 Files to copy

Before the files starting to be copied you will have a dialog box with similar information as below.

Destination Folder:  
d:\cit540  
Required disk space:  
417 Kb

The information below is required by advanced users only.

The following files will be added:

Ctregion.dll will be updated in Bin and backed up to DrvBack

Db3utils.dll will be updated in Bin

EXPRBRD.DBF will be updated in Bin and backed up to DrvBack

EXPRWIZ.DBF will be updated in Bin and backed up to DrvBack

HELP.DBF will be updated in Bin and backed up to DrvBack

MBUSCIT.DBF will be updated in Bin

Mbuscit.dll will be updated in Bin

Mbuscit.ini will be updated in Bin

MbuscitABB.ini will be updated in Bin

PROTDIR.DBF will be merged in Bin, backed up to DrvBack and copied to User\Include

PROTERR.DBF will be updated in Bin and backed up to DrvBack

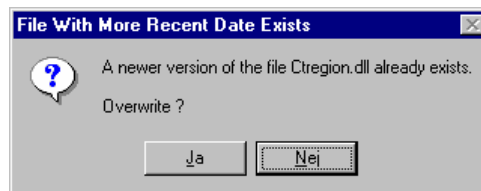
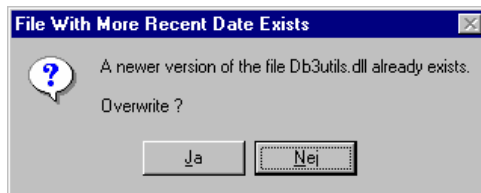
Readme.wri will be added to DrvDoc

The following sample project directories will be created:

Mbus

## 3.3 File With More Recent Data Exists

If these two dialog boxes appear at the screen you should normally press the Nej or No button.



### 3.4 Installation in Citect project

When the driver is installed in this way you can choose M-Bus from the standard Express Communication Wizard when you shall make a new project in citect.

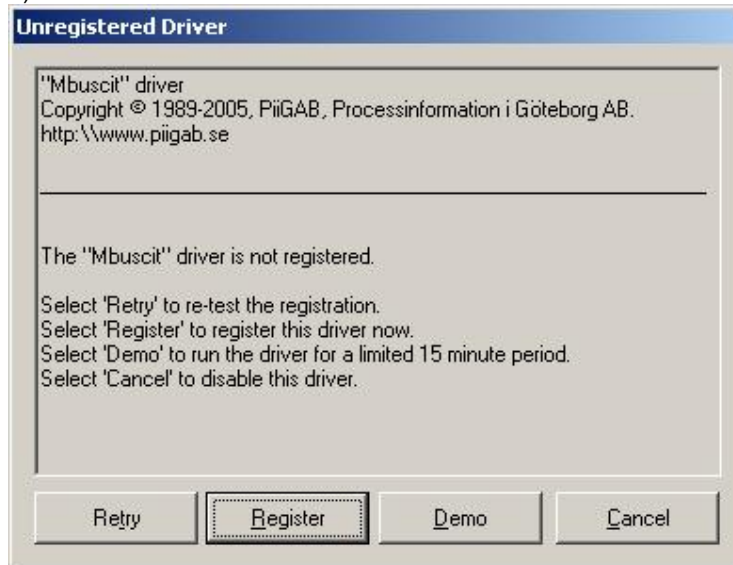


## 4. Appendix 2 - Software Protection

### 4.1 Unregistered driver

When a driver is run for the first time, or each time an unregistered driver runs, an 'Unregistered driver' dialog box will be displayed. This dialog box will prompt you to either:

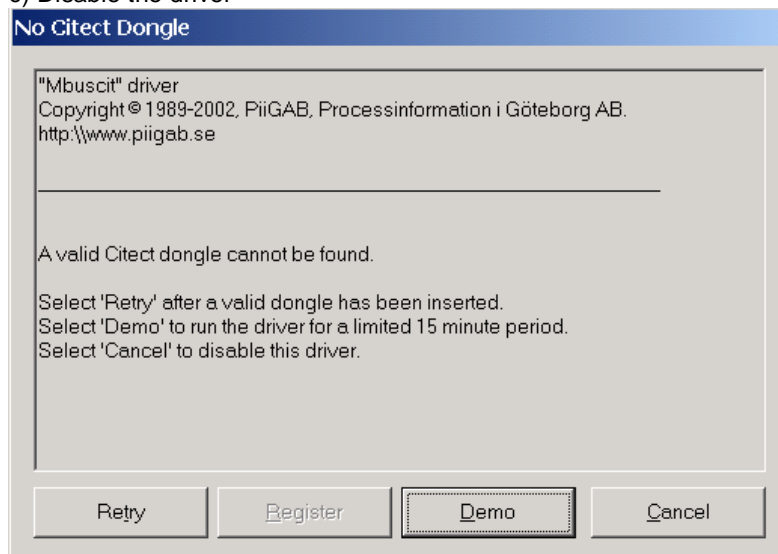
- Let the driver re-test the registration settings
- Register the driver
- Run in a demo mode
- Disable the driver



### 4.2 Citect Dongle not found

If there is no Citect dongle attached to local parallel port, or the dongle cannot be detected, then the 'No Citect Dongle' dialog box will be displayed. This dialog box will prompt you to either:

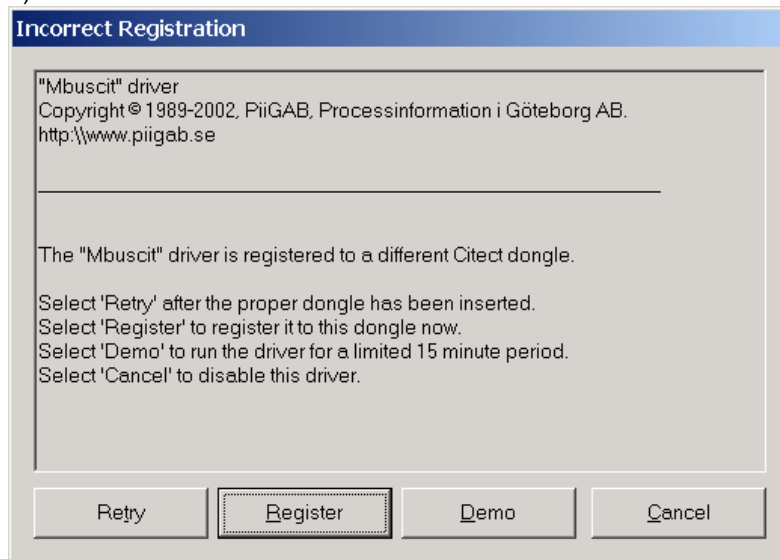
- Insert a valid Citect dongle and retry
- Run in a demo mode
- Disable the driver



### 4.3 Incorrect registration

If the Citect dongle does not match the registration key, then the 'Incorrect registration' dialog box will be displayed. This dialog box will prompt you to either:

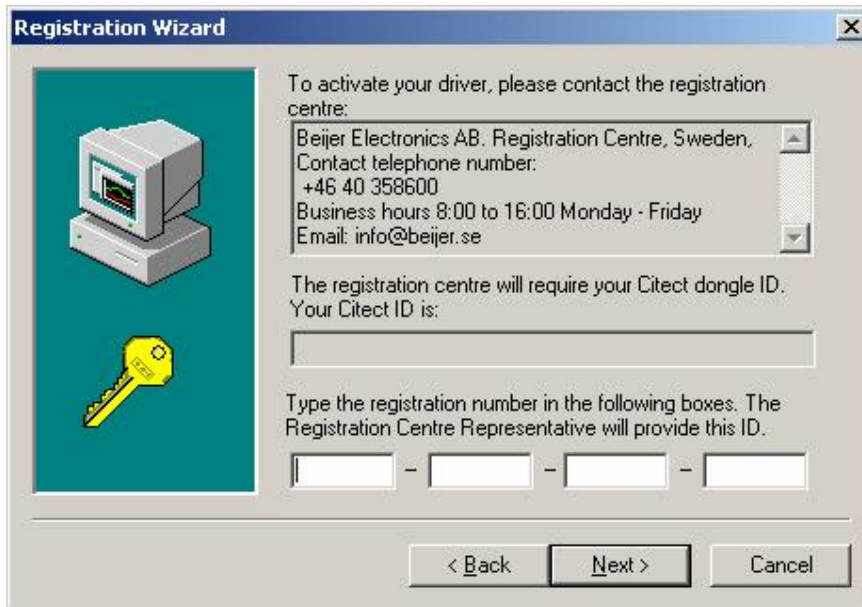
- a) Replace the dongle and retry
- b) Register the driver to the current attached dongle
- c) Run in a demo mode
- d) Disable the driver

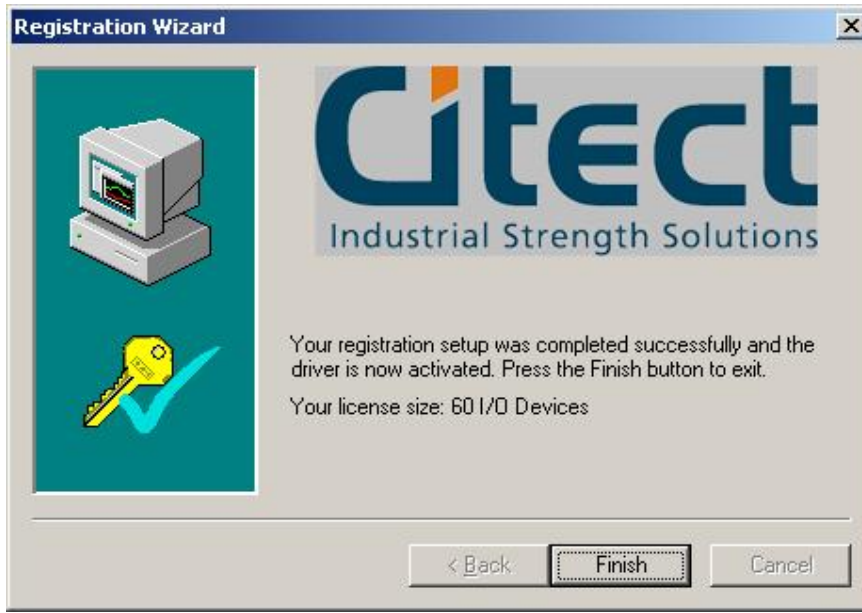


### 4.4 Registration wizard

If the user selects the 'Register' button on the dialog box, the 'Registration Wizard' will be activated which guides you through the registration process. If you have entered an invalid registration number, then a fail message will be displayed as shown, otherwise a successful message will be displayed and the registration number will be saved in the group [DRIVER\_REGISTRATION] in Citect.ini file.







If you cancel the registration process, the 'Unregistered Driver' dialog box will remain open. If the 'Finish' button is pressed (process finished successfully), the dialog box will close automatically. You will have information about the size of your license. The license sizes are 3, 60, 250 or unlimited.

#### 4.5 Demo Mode

If you choose the 'Demo' button from the dialogue, then the driver will run in demo mode for period of time.

#### 4.6 Disable the Driver

If you choose the 'Cancel' button from the dialog, the driver will be put in 'Channel Offline' mode. This means the driver will report the 'Channel Offline' hardware alarm whenever the 'WatchTime' parameter triggers, which is normally every 30 seconds.

#### 4.7 Changing license size

Go to your Citect.ini and locate the group [DRIVER\_REGISTRATION]. Put an exclamation mark just before your old license key as in the following example  
!Mbuscit=XXXXXX-XXXXXX-XXXXXX-XXXXXX. This is the same as a comment. Next time you are starting Citect the driver can't find the key and therefore it will ask you about a new registration.

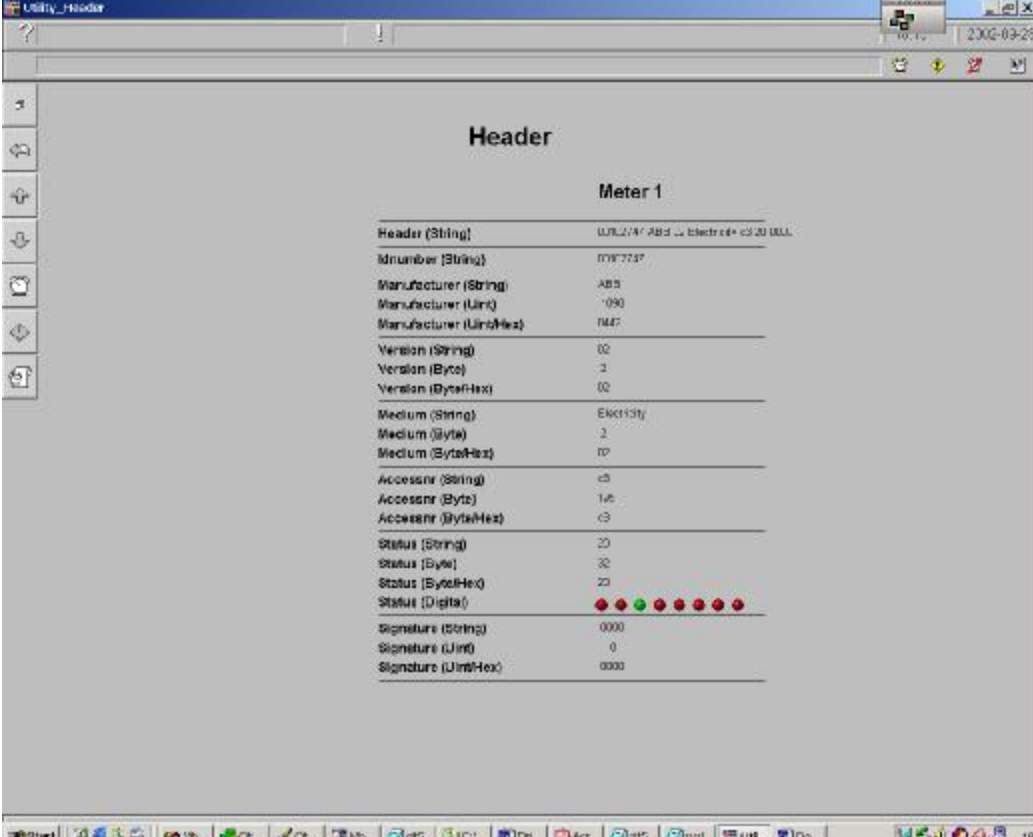
## 5. Appendix 3 - M-Bus Demo

In this appendix you can see some of the pictures from the demo project. The Mbus\_Demo.ctz project is made in Citect version 5.40 SP B R1 and can be restored in the same way as all other Citect projects.

In the tag database you have a lot of address examples.

### 5.1 Header information

At this picture you can see all different ways you use the information from the M-Bus header. Look especially at the status byte there you can have each separate bit as a digital tag. You can tag them direct as alarm tags and use them direct in the alarm system.



The screenshot shows the 'Utility\_Header' window displaying the following data for 'Meter 1':

Field	Value
Header (String)	000274F ABd 00 Electrod+ c320 0000
Idnumber (String)	0707737
Manufacturer (String)	ABB
Manufacturer (UInt)	7090
Manufacturer (UIntHex)	00000000
Version (String)	00
Version (Byte)	2
Version (ByteHex)	00
Medium (String)	Elektricitet
Medium (Byte)	2
Medium (ByteHex)	00
Accessnr (String)	c3
Accessnr (Byte)	146
Accessnr (ByteHex)	c3
Status (String)	20
Status (Byte)	20
Status (ByteHex)	20
Status (Digital)	●●●●●●●
Signature (String)	0000
Signature (UInt)	0
Signature (UIntHex)	0000

## 5.2 Record and Value information

The next two pictures are rather similar. At the first one you can see the information from each data record in the unit. Here is the data type Drx.Record used. You have more information in the chapter "Examples data types". Look at the Vib Voltage (V\*0.1) L1. The first part Voltage (V\*0.1) is automatically taken from the standard M-Bus tables and the second part L1 from the manufacturer specific tables. If you should be unsure if the text is true or not you can set the driver specific parameter VibDebug=1 in citect.ini and you can see the result at the next picture.

Record	Unit	Tariff#	Storage	Datatype	Function	Value	Vib
DE1	3	0	0	BOD12	Tr4	237268	Energy (kWh*10)
DE2	3	1	0	BOD12	Inst	237260	Energy (MWh*10)
DE3	3	2	0	BOD12	Inst	0	Energy (MWh*10)
DE4	3	0	0	Int8	Inst	1	Current tariff
DE5	3	0	0	BOD8	Inst	1	Transformer ratio
DE6	3	0	0	Int84	Inst	671088704	Error flags (binary)
DE7	3	0	0	Int8	Inst	102	Power takt counter
DE8	3	0	0	Special	Inst	N/A (Unit)	Manufacturer specific
DE9	3	0	0	Int32	Inst	0	Power (W*0.01)
DE10	3	0	0	Int32	Inst	0	Power (W*0.01) L1
DE11	3	0	0	Int32	Inst	0	Power (W*0.01) L2
DE12	3	0	0	Int32	Inst	0	Power (W*0.01) L3
DE13	3	0	0	Int16	Inst	2232	Voltage (V*0.1) L1
DE14	3	0	0	Int16	Inst	2227	Voltage (V*0.1) L2
DE15	3	0	0	Int16	Inst	2236	Voltage (V*0.1) L3
DE16	3	0	0	BOD4	Inst	0	Current (mA*10) L1
DE17	3	0	0	BOD4	Inst	0	Current (mA*10) L2
DE18	3	0	0	BOD4	Inst	0	Current (mA*10) L3
DE19	3	0	0	BOD4	Inst	5000	Frequency (Hz*0.01)
DE20	3	0	0	Special	Inst	N/A (Unit)	Manufacturer specific
DE21	3	0	0	Int16	Inst	0	Power factor (*0.001)
DE22	3	0	0	Int16	Inst	2	Phase angle voltage (degrees*0.1) L2
DE23	3	0	0	Int16	Inst	6	Phase angle voltage (degrees*0.1) L3
DE24	3	0	0	Int16	Inst	0	Phase angle current (degrees*0.1) L1
DE25	3	0	0	Int16	Inst	0	Phase angle current (degrees*0.1) L2
DE26	3	0	0	Int16	Inst	0	Phase angle current (degrees*0.1) L3
DE27	2	0	0	Tr4	Tr4	0	Digital Input (binary)
DE28	2	0	1	Int8	Inst	0	Digital Input (binary)
DE29	2	0	0	BOD12	Inst	0	Computation counter
DE30	1	0	0	Tr8	Tr8	0	Digital Output (binary)
DE31	3	0	0	Special	Inst	N/A (Unit)	Manufacturer specific
N/A	(Empty)						
N/A	(Empty)						
N/A	(Empty)						



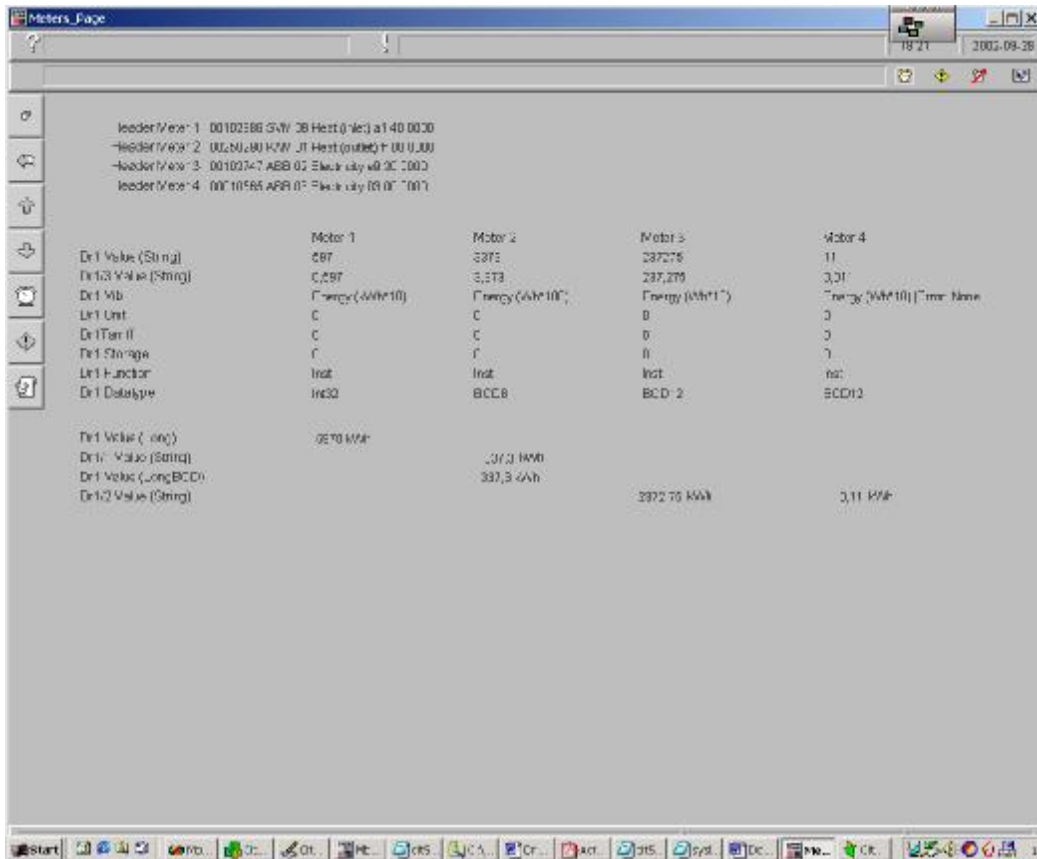
### 5.3 Record and Value debug information

At this picture you can see the basic Vib information after decoding. You can have four different groups VIF, FB, FD, VIFE. If it is FF it means manufacturer specific text. How the driver can find this information is explained in chapter "Examples data types". You can see the text from the example above have the code FD48 FF01.

Record	Unit	Tariff	Storage	Datatype	Function	Value	Vib
D01	0	0	0	ACT2	Inst	237285	VIF04
D52	0	-	0	SC12	Inst	237285	VIF05
D53	0	2	0	SC12	Inst	0	VIF05
D04	0	0	0	Int8	Inst	1	FF01
D05	0	0	0	SC18	Inst	1	FF02
D06	0	0	0	Int64	Inst	671088704	FDL7
D07	0	0	0	Int8	Inst	102	FF18
D58	0	0	0	Special	Inst	N/A (0x1F)	1F Manufacturer specific
D09	0	0	0	Int32	Inst	C	VIF29
D010	0	0	0	Int32	Inst	C	VIF29 FF01
D011	0	0	0	Int32	Inst	C	VIF29 FF02
D012	0	0	0	Int32	Inst	C	VIF29 FF03
D013	0	0	0	Int16	Inst	2221	FD48 FF01
D014	0	0	0	Int16	Inst	2227	FD48 FF02
D015	0	0	0	Int16	Inst	2231	FD48 FF03
D016	0	0	0	Int4	Inst	C	FD5a FF01
D017	0	0	0	Int4	Inst	C	FD5a FF02
D018	0	0	0	SC14	Inst	C	FD5a FF03
D019	0	0	0	SC14	Inst	4994	FF59
D520	0	0	0	Special	Inst	N/A (0x1F)	1F Manufacturer specific
D021	0	0	0	Int16	Inst	C	FF60
D022	0	0	0	Int16	Inst	E	FF42 FF02
D023	0	0	0	Int16	Inst	C	FF42 FF03
D024	0	0	0	Int16	Inst	C	FF4a FF01
D025	0	0	0	Int16	Inst	C	FF4a FF02
D026	0	0	0	Int16	Inst	C	FF4a FF03
D027	2	0	0	Int8	Inst	C	FDLb
D028	2	0	0	Int8	Inst	C	FDLb
D029	2	0	0	SC12	Inst	C	FD61
D030	1	0	0	Int8	Inst	C	FDLa
D031	0	0	0	Special	Inst	N/A (0x0F)	0F Manufacturer specific
N/A (Empty)							
N/A (Empty)							
N/A (Empty)							

## 5.4 Meter information

At this picture you have information from four different meters. They were all multidrop connected at the same M-Bus wire. Some are addressed with primary addressing and some with secondary addressing. You have also at the same time single telegram and multitelegram.



## 5.5 Citect.ini for demoproject

Here is an example how the Citect.ini can be disposed if you need to use different parameters for different IODevices.

```
[MBUSCIT]
!DebugStr=* ALL
CacheLiveTime=40000
NrOfTelegrams=5
VibDebug=0
```

```
[MBUSCIT.PORT1_BOARD1.Kamstrup]
InitRead=0
CacheLiveTime=10000
```

```
[MBUSCIT.PORT1_BOARD1.Svm]
NrOfTelegrams=1
```