

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

M-Bus driver, User information

Contents

1.1	DEVICE APPLICATION NOTES	4
1.1.1	Key features for M-Bus driver	4
1.1.2	What's new in Version 2	4
1.1.3	Principles of Operation.....	6
1.1.4	This chapter is a part of the official "The M-Bus: A Documentation" Version 4.8.....	7
1.1.5	Specifications for Bus Installations	7
1.1.6	Reference: Communications forms	8
1.1.7	Reference: Data types	12
1.1.8	Examples data types.....	13
1.1.9	Hints, Tips, and Frequently asked questions.....	24
1.2	DRIVER REFERENCE	25
1.2.1	Driver generated error codes	25
1.2.2	Standard parameters	25
1.2.3	Driver Specific parameters which can be port/group/device specific	26
1.2.4	Driver Specific Parameters which is driver specific (only maingroup).....	29
1.2.5	M-Bus Port/Group/Device specific parameters.....	29
1.2.6	Driver generated statistics	30
1.2.7	Debug messages via TraceFileLevel.....	31
1.2.8	Debug messages via TraceTx and TraceRx	31
2.	ANALYSIS	33
2.1	M-BUS OVERVIEW	33
2.1.1	Transmission Parameters.....	33
2.1.2	Telegram Format	34
2.1.3	Meaning of the Fields.....	34
2.1.4	Communication Process.....	36
2.1.5	Selection and Secondary Addressing.....	36
2.1.6	Applications of the FCB-mechanism.....	37
2.1.7	Variable Data Structure.....	37
2.1.8	Fixed Data Structure	43
2.2	DBFs	44
2.2.1	Help.dbf.....	44
2.2.2	Mbuscit.dbf Entries	44
2.2.3	Protdir.dbf.....	45
2.3	DEVELOPMENT RESOURCES	45
2.3.1	Contacts.....	45
2.3.2	Documents.....	45
2.3.3	Driver Version History	46
3.	APPENDIX 1 - INSTALLATION	47
3.1	BEGINNING OF INSTALLATION	47
3.1.1	Installing before version 2.02.01.001	47
3.1.2	Installing from version 2.02.01.001.....	47
3.2	FILES TO COPY	48
3.3	SETUP COMPLETE	49
3.4	EXPRESS COMMUNICATION WIZARD.....	50
4.	APPENDIX 2 - SOFTWARE PROTECTION	51
4.1	UNREGISTERED DRIVER	51
4.2	CITECT DONGLE NOT FOUND.....	51
4.3	INCORRECT REGISTRATION	52
4.4	REGISTRATION WIZARD.....	52
4.5	DEMO MODE	54
4.6	DISABLE THE DRIVER.....	54
4.7	CHANGING LICENSE SIZE.....	54
5.	APPENDIX 3 - M-BUS DEMO	55
5.1	DEMO PROJECTS	55
5.2	HEADER INFORMATION.....	55

5.3	RECORD AND VALUE INFORMATION.....	56
5.4	RECORD AND VALUE DEBUG INFORMATION	57

1.1 Device Application notes

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

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 heat meters, water meters, electricity meters and more.

1.1.1 Key features for M-Bus driver.

- Generic M-Bus driver.
- Supports Variable Data Structure.
- Mixes different manufacturer on the same M-Bus network.
- Supporting object tag type Value, Vib, Unit, Tariff, Storage, Function, Datatype, Record etc.
- Supports header tag type Idnumber, Manufacturer, Version, Medium, Access number, Status, Signature etc.
- All data types automatically converted to strings if needed.
- Vib reads standard and manufacturer specific text automatically from M-Bus tables
- Support for manufacturer specific data
- M-Bus telegrams is cached in the driver for better performance
- Primary and secondary addressing
- Port/Group/Device specific parameters
- Multi telegram support

1.1.2 What's new in Version 2

1.1.2.1 What's new in version 2.00.00.001

Version 2.00.00.001 has many new features. Most of them are by request from users and several of the new features are adaptations for effective handlings of bigger projects.

- Increased license sizes to 20, 250, 1000 or unlimited number of meters.
- Simplified addressing for secondary addressing.
- New parameter InitTimeOut.
- New parameter MainTimeOut.
- New parameter IgnoreDeviceStatusStartup.
- Parameter InitRead completed with Application Reset.
- New parameter AppSubCode.
- Strengthen grouphandling of meters.
- Separately debug and trace possibilities from the whole project to single meters.
- Supporting plain-text VIF and data.
- New tracking telegram types Identity, Idvalue and Idobject
- The driver is updated to follow the new M-Bus standard EN13757

1.1.2.2 What's new in version 2.00.00.002

- Error handling of secondary address is increased.
- Primary address 251 is now accepted.

1.1.2.3 What's new in version 2.01.00.001

- The driver now supports the old Fixed Data Structure.
- Number of channels is increased to 512.

1.1.2.4 What's new in version 2.01.01.001

- Adjustments for record with no data.
- Set Always FCB=1 after SND_NKE.

1.1.2.5 What's new in version 2.02.00.001

- The driver is now OS stamped for Windows 2000(5.0), Windows XP(5.1), Windows Server 2003(5.2), Windows Vista(6.0), Windows Server 2008(6.0), Windows 7(6.1) and Windows Server 2008 R2(6.1).
- Support for the old Sentinel licensing as well as the new FLEXERA software licensing system.
- Wrong value when the M-Bus data type int24 converted to Citect LONG data type is fixed.
- The Idle Filler 0x2F didn't work in a correct way. Fixed.

1.1.2.6 What's new in version 2.02.00.002

- Added internal function GetChannel

1.1.2.7 What's new in version 2.02.01.001

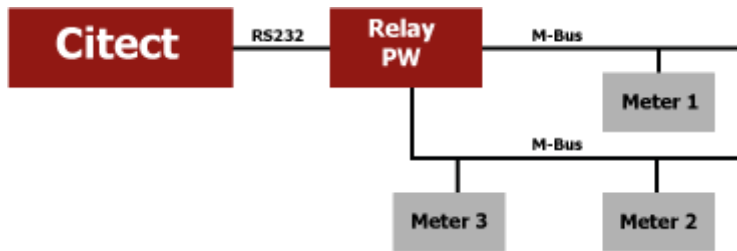
- The driver is now OS stamped for Windows 8 (6.2), Windows 8.1 (6.2), Windows Server 2012 (6.2), Windows Server 2012R2 (6.2).
- The Idle Filler 0x2F didn't work in a correct way if 0x2F is just before the CRC. Fixed. The standard says "Idle filler (no to be interpreted), following byte = DIF of next record".
- Support for new InitRead command Slave Select + SND_NKE.
- The installation doesn't use winzip self-extractor anymore due to problems with installations in 64 bit environment.

1.1.2.8 What's new in version 2.02.02.001

- The driver OS stamp is changed to Windows 8 (6.2), Windows Server 2012 (6.2), Windows 8.1 (6.3), Windows Server 2012R2 (6.3).
- BugFix: Changes in the cache solving some problem with for example slow network start.

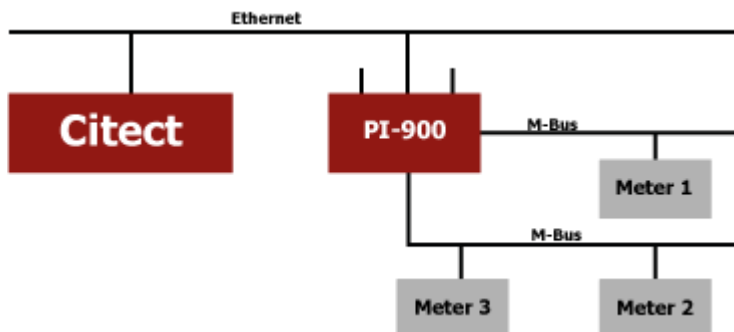
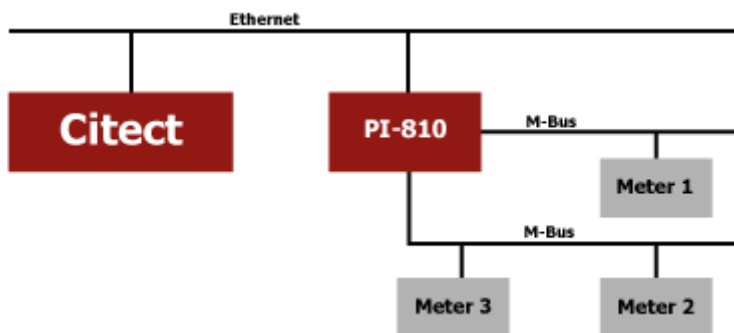
1.1.3 Principles of Operation

1.1.3.1 Serial communication



Block diagram showing principle of the M-Bus System

1.1.3.2 Ethernet communication



Block diagram showing principle of the M-Bus System

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

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.

1.1.5 Specifications for Bus Installations

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

1.1.5.1 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.

1.1.5.2 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 29Ω. 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.

1.1.6 Reference: Communications forms

1.1.6.1 Serial communication

1.1.6.1.1 Boards form

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.	

1.1.6.1.2 Ports form

Field	Default	Allowable values
Port Name	This field is user defined.	
Port number	If you are using your computer's COM port you should enter the port number here. The port number must be -1 for dialable remote meters.	
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	2400	Refer to the M-Bus 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.	

1.1.6.2 Ethernet communication TCP

1.1.6.2.1 Boards form

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.	

1.1.6.2.2 Ports form

Field	Default	Allowable values
Port Name	This field is user defined.	
Port number	The port number that the I/O device is connected to.	
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	BLANK	Not used by the driver
Data bits	BLANK	Not used by the driver
Stop bits	BLANK	Not used by the driver
Parity	BLANK	Not used by the driver
Special Opt	-i192.168.100.136 -p10001 -t	
Comment	This field is user defined and is not used by the driver.	

1.1.6.3 Ethernet communication UDP

1.1.6.3.1 Boards form

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.	

1.1.6.3.2 Ports form

Field	Default	Allowable values
Port Name	This field is user defined.	
Port number	The port number that the I/O device is connected to.	
Board name	Refers to the board previously defined in 'boards' form.	
Baud rate	BLANK	Not used by the driver
Data bits	BLANK	Not used by the driver
Stop bits	BLANK	Not used by the driver
Parity	BLANK	Not used by the driver
Special Opt	-i192.168.100.136 -p10001 -u	
Comment	This field is user defined and is not used by the driver.	

1.1.6.4 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	0 is reserved for unconfigured slaves. 1 to 250 is used for primary addressing of slaves. 251 is reserved for management communication with the primary master. 254 is the address for test and diagnosis.
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.	

1.1.6.5 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 accounted for 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 wild carded by a wildcard nibble Fh while the fields for manufacturer, version and medium can be wild carded by a wildcard byte FFh.

1.1.6.6 Example Primary addresses

I/O Device Address	Description
1	Standard primary address.
1/P	Standard primary address. Similar notation as for secondary addressing.
xxxx: 1	Group plus primary address
xxxx: 1/P	Group plus primary address

Where xxxx: is the group name

1.1.6.7 Example Secondary addresses

I/O Device Address	Description
747/S	Simplified secondary addressing
00102747.0442.02.02	Complete secondary addressing
00102747.FFFF.FF.FF	Identification number and wildcards
FFFFFF747.FFFF.FF.FF	Identification number and wildcards
xxxx: 1/S	Group plus secondary address
xxxx: 00102747.0442.02.02	Group plus secondary address
xxxx: 00102747.FFFF.FF.FF	Group plus secondary address
xxxx: FFFF747.FFFF.FF.FF	Group plus secondary address

Where xxxx: is the group name

The most common way to address meters via secondary addressing is to use only the meter number or parts of the number. The Citect M-Bus driver Version 2 has enhanced possibilities for secondary addressing so if you are writing e.g. 157/S it has the same meaning as FFFF747.FFFF.FF.FF. The driver itself is putting in the wildcard characters in the correct place. The very best way is to use the complete address e.g. 00102747.0442.02.02 as secondary addressing. The driver will check the first real answer after slave select so the address information in the header is equal to the slave select address. This will prevent errors if any meter should not deselect itself in a correct way after a new slave select has been made.

For more information about group addressing etc have a look in chapter 1.2.5 M-Bus Device/Group/Port specific parameters.

1.1.7 Reference: Data types

IO Device Type	Citect data format	Citect data types	Fixed Data Structure	Description/Special Usage/Limitations/ Valid Ranges
Value	Drx.Value	STRING	ok	Read (Record range 0 – 500)
Value with decimal point	Drx/y.Value	STRING	ok	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	ok	Read (Record range 0 – 500)
Value	Drx.Value	BCD	-	Read (Record range 0 – 500)
Value	Drx.Value	LONGBCD	ok	Read (Record range 0 – 500)
Value	Drx.Value	REAL	-	Read (Record range 0 – 500)
Value Information Block	Drx.Vib	STRING	ok	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	ok	Read (Record range 0 – 500)
Identity Value	Drx.IdValue	STRING	ok	Read (Record range 0 – 500)
Identity Object	Drx.IdObject	STRING	ok	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	ok	Read
Identification number	Idnumber	STRING	ok	Read
Manufacturer	Manufacturer	STRING	-	Read
Manufacturer	Manufacturer	INT	-	Read
Version	Version	STRING	-	Read
Version	Version	BYTE	-	Read
Medium	Medium	STRING	ok	Read
Medium	Medium	BYTE	ok	Read
Access number	Accessnr	STRING	ok	Read
Access number	Accessnr	BYTE	ok	Read
Status	Status	STRING	ok	Read
Status	Status	BYTE	ok	Read
Status (bit)	Status.z	DIGITAL	ok	Read
Signature	Signature	STRING	-	Read
Signature	Signature	INT	-	Read
Identity	Identity	STRING	ok	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

1.1.8 Examples data types

1.1.8.1 Values

The driver can convert all M-Bus data types automatically to strings for you. This is especially good if you don't know the data type the manufacturer wants you to use.

If you want to put in a decimal comma in a string value you normally have to use Cicode or Vba. If you write e.g. Dr9/2.Value, the driver will put in a decimal comma for you in the second position from the 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 a 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 data types like INT64, BCD12 and these are not supported in Citect so in this case 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 to use. With this information you can manually choose the correct data type for Citect.

The fixed data structure, supported from version 2.00.01.001, is limited to the transmission of only two counter states of a predetermined length, which have binary or BCD coding.

The two counters can have datatype STRING, LONG or LONGBCD. The driver is looking in the statusbit 0 so it knows how to convert the counter value. If you use string the driver will convert it in a correct way automatically. If you want to use LONG or LONGBCD you have to know the bit 0 status (See 1.1.7.18) to configure correct datatype, otherwise you will have a hardware error depending on the wrong conversion.

Here is an example of the Variable Tags dialog box in Citect V6.1

Here is an example of the Variable Tags dialog box in Citect V7.0

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

1.1.8.2 Value Information Block

The driver is looking through the file mbuscit.ini, via the objects vifcode, for the M-Bus standard texts. The driver also supports manufacturers specific texts via a specific .ini file.

If the driver finds that the vibcode is manufacturer specific it will try to find the file mbuscitXXX.ini where XXX is the manufacturer ID. It will look through the file for a section name made up by a manufacturer ID, version and medium. As an example, the file mbuscitABB.ini from ABB electricity meters is installed in the Citect\Bin during the installation process.

To control if the text shown is correct you can set the driver specific parameter VibDebug=1. In this case you will see the M-Bus vifcode 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 also makes it possible to change language for the text. If you do this be sure to take a backup copy of your own mbuscit.ini before reinstallation of the driver. If you want you can change your path for your .ini file via the driver specific parameter IniPath.

Data Type: STRING
Address: Dr9.Vib
Comment: The value information block for record 9. VibDebug=0
Example: Current (mA*10) L1

Data Type: STRING
Address: Dr9.Vib
Comment: The value information block for record 9. VibDebug=0
Example: Cust. ID

Data Type: STRING
Address: Dr9.Vib
Comment: The value information block for record 9. VibDebug=1
Example: FD5a FF01

1.1.8.3 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

1.1.8.4 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

1.1.8.5 Storage

Datatype Tariff has the range 0 – 219902325551

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

1.1.8.6 Function

The function field gives the type of data as follows:

Code	Description	Function
0	Instantaneous value	Inst
1	Maximum value	Max
2	Minimum value	Min
3	Value during error state	Error

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

1.1.8.7 Data type

The data field shows how the data from the master must be interpreted. The following table contains the possible coding of the data field:

Code	Meaning	Drx.Datatype	Citect Data type
0	No data	No data	-
1	8 Bit Integer	Int8	STRING/BYTE
2	16 Bit Integer	Int16	STRING/INT
3	24 Bit Integer	Int24	STRING/LONG
3	32 Bit Integer	Int32	STRING/LONG
5	32 Bit Real	Real32	STRING/REAL
6	48 Bit Integer	Int48	STRING
7	64 Bit Integer	Int64	STRING
8	Selection for Readout	Selection	-
9	2 digit BCD	BCD2	STRING/BCD
10	4 digit BCD	BCD4	STRING/BCD
11	6 digit BCD	BCD6	STRING/LONGBCD
12	8 digit BCD	BCD8	STRING/LONGBCD
13	Variable length	Variable	STRING ¹ N/A (0x0D)
14	12 digit BCD	BCD12	STRING
15	Special Functions	Special	-

¹The M-Bus driver is supporting variable length 8-bit text string according to ISO 8859-1. The driver is responding with "N/A (0x0D)" for variable length positive/negative BCD number, binary number and floating point number according to IEEE 754 .

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

1.1.8.8 Data Record

The record type is adding all separate string types Unit, Tariff, Storage Datatype, Function, Value and Vib to one long string. With this type you have all information from the data record object you should need. Data record zero doesn't exist therefore the driver uses this to write the header for the object information. It's important to use a font type e.g. 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)
```

1.1.8.9 Identity Value

Identity Value is the identity of the meter itself and value merged together with a space between. With this telegram type you can make a track to the meter identity the value is read out from. It's especially applicable when saving the values in a database. For more information see 1.1.7.10 Identity Object and 1.1.7.20 Identity.

Data Type: STRING
 Address: Idvalue
 Comment: Identity Value
 Example: 1548290042040502 76

1.1.8.10 Identity Object

Identity Object is the identity of the meter itself and the M-Bus object in raw binary format merged together. With this telegram type you can make a track to the meter identity and to the full object, the value or other data is read out from. It's especially applicable when saving the values in a database to have a possibility to track the shown value how it's decoded depended to datatype etc. For more information see Identity and Identity Value.

Data Type: STRING
 Address: Idobject
 Comment: Identity Object
 Example: 154829004204050207fd174c00000000000000

1.1.8.11 Manufacturer specific data

The Citect M-Bus driver is supporting manufacturer specific data. Normally you need the documentation from the manufacturer to find out the correct position in the area. The manufacturer specific area itself is an own object or data record.

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

1.1.8.12 Header

Variable Data structure

Header is a telegram type which has a content of the main information from the header part of the telegram. It's merged together of Identification number, Manufacturer, Version, Medium, Access number, Status and Signature.

Data Type: STRING
Address: Header
Comment: Total Header
Example: 00102886 SVM 09 Heat 74 40 0000

Fixed Data structure

Header is a telegram type which has a content of the main information from the header part of the telegram. It's merged together of Identification number, (Manufacturer), (Version), Medium, Access number, Status and (Signature).

Data Type: STRING
Address: Header
Comment: Total Header
Example: 74350410 N/A N/A Heat 5a 10 N/A

1.1.8.13 Identification number

Identification Number is a customer number, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00000000 to 99999999

Data Type: STRING
Address: Idnumber
Comment: Identification number
Example: 00102886

1.1.8.14 Manufacturer

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:

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

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

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

1.1.8.15 Version

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

Data Type: STRING
Address: Version
Comment: Version
Example: 2

Data Type: BYTE
Address: Version
Comment: Version
Example: 2

1.1.8.16 Medium

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	04
Steam	05
Warm 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 14
Hot Water	15
Cold Water	16
Dual Water	17
Pressure	18
A/D Converter	19
Reserved	1A to 20
Reserved for valve	21
Reserved	22 to FF

Data Type: STRING
 Address: Medium
 Comment: Medium
 Example: Electricity

Data Type: BYTE
 Address: Medium
 Comment: Medium
 Example: 2

1.1.8.17 Access number

Access Number has unsigned binary coding, and is increased by one after each RSP_UD from the slave.

Data Type: STRING
 Address: Accessnr
 Comment: Access number
 Example: 2

Data Type: BYTE
 Address: Accessnr
 Comment: Access number
 Example: 2

1.1.8.18 Status

Status field are used to indicate application errors.

(Variable Data structure)

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

(Fixed Data structure)

Bit	Meaning with Bit set	Significance with Bit not set
0	Counter 1 and 2 coded signed binary	Counter 1 and 2 coded BCD
1	Counter 1 and 2 are stored at fixed date	Counter 1 and 2 are actual values
2	Power low	Not power low
3	Permanent error	No permanent error
4	Temporary error	No temporary error
5	Specific to manufacturer	Specific to manufacturer
6	Specific to manufacturer	Specific to manufacturer
7	Specific to manufacturer	Specific to manufacturer

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: true/false

1.1.8.19 Signature

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

Data Type: STRING
Address: Signature
Comment: Signature
Example: 2

Data Type: INT
Address: Signature
Comment: Signature
Example: 2

1.1.8.20 Identity

Identity is the identity of the meter itself. It's the raw value of identification number, manufacturer, version and medium merged together. This is similar to a MAC address used by computers. With this telegram type you can make an alarm if a user has changed the meter without admittance. The new meter will normally have a different identity.

Data Type: STRING
Address: Identity
Comment: Identity
Example: 1548290042040502 (16 digits)

1.1.9 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 means that 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` on the screen. The reason for this `#COM` depends very often on that the IOserver get a request and it takes too long time to answer the question. Increase the Citect [LAN] timeout parameter to e.g. 20000 (20 sec); default is 8000 (8 sec).
- If you have a multi telegram meter and want to always read only the first telegram you should set the parameter `NrOfTelegrams` to 1 and `InitRead` shall be at the lowest value 1.
- If you are using Kamstrup meters you should normally set parameter `InitRead` to 0 which means no initializing.

1.2 Driver reference

	Detail
Driver name	MBUSCIT
Maximum array size ¹	256

1.2.1 Driver generated error codes

N/A

1.2.2 Standard parameters

Parameter	Default	Allowable values	Description
Block (bytes)	256		
Delay (mS)	50		
MaxPending	2		
PollTime (mS)	0		
Timeout (mS)	3000		
Retry	2		
WatchTime (Sec)	30		

1.2.2.1 Block

The optimum size of data for a read request in bytes.

1.2.2.2 Delay

The period, in milliseconds, to wait between receiving a response and sending the next command.

1.2.2.3 MaxPending

The maximum number of pending commands that the driver holds ready for immediate execution on each channel.

1.2.2.4 Polltime

The polling time (in milliseconds). If the PollTime is set to 0 then the driver will work in interrupt mode.

1.2.2.5 Timeout

Specifies how many milliseconds to wait for a response before regarding a request as having failed

1.2.2.6 Retry

The number of times to retry a request which has timed out after no response.

1.2.2.7 WatchTime

The frequency that the driver uses to check the communications link to the I/O Device.

¹ Equivalent to 'Maximum Request Length'

1.2.3 Driver Specific parameters which can be port/group/device 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	0 to 3	Reset command before REQ_UD2 0 = No specific init packet 1 = SND_NKE 2 = Application Reset 3 = SND_NKE + Application Reset 4 = N/A 5 = N/A 6 = SlaveSelect + SND_NKE
AppSubCode	-1	0 to 15	Subgroups for application reset
InitTimeOut	Timeout		
MainTimeOut	Timeout		
IgnoreDeviceStatusStartup	0	0 or 1	If flag is set, the driver will go online without any attempt to communicate to the device.
TraceFileLevel	0	0 or 1	

1.2.3.1 CacheLiveTime (mS)

Time for the cached value to live before reread.

The slave response telegrams are saved inside the driver during the CacheLiveTime. If the client makes a new request during this time, the response will come from the cache. If the time has elapsed a new request will be made.

1.2.3.2 NrOfTelegrams

The number of telegrams the driver will request from any device.

Normally the driver is looking for MDH = 1Fh the slave signals to the master that it wants to be readout once again (multi telegram readouts). The master must readout the data until there is no MDH = 1Fh in the respond telegram.

You have possibilities to set NrOfTelegrams to a free value so you can choose how many telegrams the driver shall readout. If the device still has unread telegrams, they will be discarded. This is commonly used in heat meters which have many telegrams and you only want to read out e.g. the first telegram.

If you have set NrOfTelegram=1 remember to reset the meter before the next readout. If not the driver will continue to readout the following telegram during the next request. InitRead has to be configured at least with level 1, SND_NKE.

1.2.3.3 InitRead

InitRead or meter reset determines the behavior before any request is sent to a device.

It's most common to use the SND_NKE to switch to the first telegram, although this is not defined in the standard or in the documentation of the M-Bus Usergroup. Per definition the SND_NKE should just reset the link layer (FCB) but not the application layer. But it is now usually to reset to the first frame with SND_NKE. Some meters use correctly the Application reset and not a SND_NKE. The user group defines that a selection (secondary address) shall act like a SND_NKE (reset link layer). The newer meters, often needs an additional application reset to switch to the first frame. This has to be done after the secondary selection. Some manufacturer use the application reset subcode to select a special telegram, e.g. standard values or enhanced billing.

No specific init packet (0)

This selection is typical for elder Kamstrup meters.

SND_NKE (default (1))

If you have a multitelegram meter and have chosen to use only e.g. the first telegram via NrOfTelegrams you need to reset the telegramcounter, with the command SND_NKE or Application reset, in the meter before the next readout.

Application reset (2)

With application reset command the server can release a reset of the application layer in the slaves. Each slave himself decides which parameters to change - e.g. which data output is default - after it has received such an application reset.

SND_NKE + Application Reset (3)

This selection should work in most cases but will be quite slow depending of a lot of traffic.

N/A (4)

N/A (5)

Slave Select + SND_NKE (6)

Some meters don't support de facto standard that the telegram counter in the meter shall be reset when the client send a slave select command. This enhanced command will solve this problem. Be carefully because other meters on the M-Bus network might not like this command.

1.2.3.4 AppSubCode

Determines the behaviour if an optional application reset subcode is needed. This function is always working only together with InitRead mode 2 or 3.

It is possible to use optional parameters after application reset. The use of the value zero for the number of the subtelegram means that all telegrams are requested.

Slaves with only one type of telegram may ignore application reset and the added parameters but have to confirm it.

Coding	Description	Examples
0	All	
1	User data	consumption
2	Simple billing	actual and fixed date values + dates
3	Enhanced billing	historical values
4	Multi tariff billing	
5	Instaneous billing	for regulation
6	Load management values for management	
7	Reserved	
8	Installation and startup	bus address, fixed dates
9	Testing	high resolution values
10	Calibration	
11	Manufacturing	
12	Development	
13	Selftest	
14	Reserved	
15	Reserved	

1.2.3.5 InitTimeout

The response from the init request is only one byte 0xE5. Comparing to a normal response the init response can have a much smaller timeout than a normal timeout. It's possible to set InitTimeout to as small as 500mS instead for normal default timeout at 3000mS.

1.2.3.6 MainTimeout

MainTimeout has as default the normal Timeout settings. The main difference is that you can use different MainTimeout for different ports, groups or devices. In some cases you maybe have one port at 2400Baud and this needs a timeout at 3000mS and you have a second port at 300Baud. This port needs a timeout around 7000mS. So with MainTimeout you can have different timeouts for different ports which is impossible with standard Timeout.

1.2.3.7 IgnoreDeviceStatusStartup

This parameter is specially made to use during the time you are working and testing the project. If you have many meters configured in the project but they are not yet connected or correct configured you will have a lot of timeouts and retries. With this parameter you can prevent the init request to be sent out and instead force the meters to be online. You can do this for ports, groups or for single meters. Depending of the characteristics of this parameter you should never have this parameter working when you are leaving the project in running mode.

1.2.3.8 TraceFileLevel

Tracefilelevel is a debug complement to the standard debugger and syslog.dat. With TraceFilelevel you can debug separate meters or chose a port or a group of meters.

The maximum size for MBusCite.log is 10240kbyte. If you want to write more data you have to delete the file or rename it.

1.2.4 Driver Specific Parameters which is driver specific (only maingroup)

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.
TraceFilePath	Citect\Bin		Path for were MBusCitect.log should be saved.

1.2.5 M-Bus Port/Group/Device specific parameters

The M-Bus 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:

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

This feature can be implemented in the Citect.INI file for the following M-Bus parameters:

CacheLiveTime
 NrOfTelegrams
 InitRead
 AppSubCode
 InitTimeOut
 MainTimeOut
 IgnoreDeviceStatusStartup
 TraceFileLevel

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.

1.2.6 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)

1.2.7 Debug messages via TraceFileLevel

This debug message is from TraceFileLevel 1. The main difference here is that you will see port and iodevice name just before the message itself. You can choose to debug ports, groups or individuell meters/IOdevices.

```

27/03/08 11:56:19 -> PORT1_BOARD1.IODev2 (Transmit)
-> 10 40 09 49 16
27/03/08 11:56:19 <- PORT1_BOARD1.IODev2 (Response)
<- e5
27/03/08 11:56:56 -> PORT1_BOARD1.IODev2 (Transmit)
-> 10 40 09 49 16
27/03/08 11:56:56 <- PORT1_BOARD1.IODev2 (Response)
<- e5
27/03/08 11:56:56 -> PORT1_BOARD1.IODev2 (Transmit)
-> 10 5b 09 64 16
27/03/08 11:56:57 <- PORT1_BOARD1.IODev2 (Response)
<- 68 46 46 68 08 09 72 15 48 29 00 42 04 05 02 69
<- 12 00 00 0e 04 00 00 00 00 00 00 0d fd 0e 04 36
<- 31 33 44 07 fd 17 4c 00 00 00 00 00 00 00 01 ff
<- 18 0a 0e ff 6c 00 00 00 00 00 00 0e ed 18 00 00
<- 00 00 00 00 1f 00 00 00 00 00 11 16
27/03/08 11:56:57 -> PORT1_BOARD1.IODev2 (Transmit)
-> 10 7b 09 84 16
27/03/08 11:56:57 <- PORT1_BOARD1.IODev2 (Response)
<- 68 74 74 68 08 09 72 15 48 29 00 42 04 05 02 6a
<- 12 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 0a fd c8 ff 01 79 22 0a fd c8 ff 02 83 22 0a
<- fd c8 ff 03 82 22 0a fd da ff 01 00 00 0a fd da
<- ff 02 00 00 0a fd da ff 03 00 00 0a ff 59 98 49
<- 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
<- 00 00 00 00 00 00 00 00 65 16
27/03/08 11:56:57 -> PORT1_BOARD1.IODev2 (Transmit)
-> 10 5b 09 64 16
27/03/08 11:56:57 <- PORT1_BOARD1.IODev2 (Response)
<- 68 16 16 68 08 09 72 15 48 29 00 42 04 05 02 6b
<- 12 00 00 02 ff 60 00 00 1f 00 53 16
27/03/08 11:56:58 -> PORT1_BOARD1.IODev2 (Transmit)
-> 10 7b 09 84 16
27/03/08 11:56:58 <- PORT1_BOARD1.IODev2 (Response)
<- 68 25 25 68 08 09 72 15 48 29 00 42 04 05 02 6c
<- 12 00 00 ce 00 ed eb 15 00 00 00 00 00 00 ce 00
<- 84 15 00 00 00 00 00 00 0f 05 16

```

1.2.8 Debug messages via TraceTx and TraceRx

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

1.2.8.1 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 .

```

1.2.8.2 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 .

```

1.2.8.3 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 0C 2D 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 00 07 31 04 00 01 01  .....1....
06 02 28 09 02 00 53 51 00 00 37 22 00 00 00 00 00  ..(...SQ..7"....
00 00 00 00 00 00 84 16                .....

```

1.2.8.4 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 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.

```

1.2.8.5 Reading a Fixed Data Structure telegram with primary address 3

```

2008/08/28-20:20:48.177 Transmit Length 5
10 7B 03 7E 16                .{.~.
2008/08/28-20:20:48.349 Received Length 25
68 13 13 68 08 03 73 10 04 35 74 CE 10 06 6A 17      h..h..s..5t...j.
91 01 00 14 71 42 04 FD 16                .....qB...

```

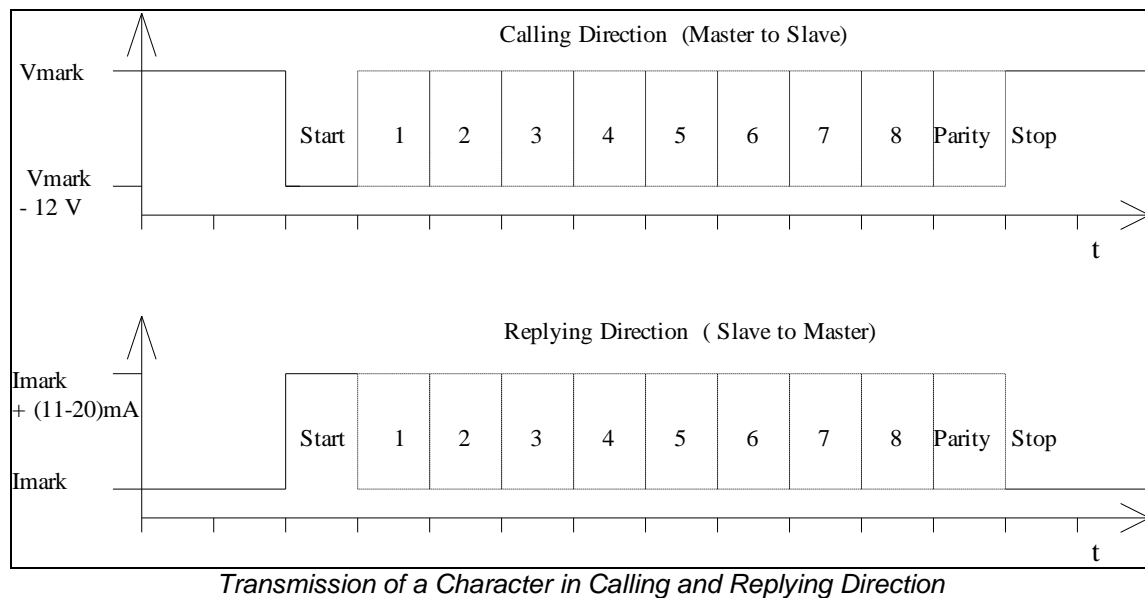

2. Analysis

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

2.1.7.1 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:

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

- **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.

2.1.7.2 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.

2.1.7.3 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)

2.1.7.4 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.

2.1.7.5 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.

2.1.8 Fixed Data Structure

The fixed data structure, besides a fixed length, is limited to the transmission of only two counter states of a predetermined length, which have binary or BCD coding. In contrast the variable data structure allows the transmission of more counter states in various codes and it also allows for further useful information about the data. The number of bytes of the transmitted counter states is also variable with this data structure. Contrary to the fixed structure, the variable structure can also be used in calling direction. For this reason the fixed data structure is not recommended for future developments.

To identify the fixed data structure, the number 73h for the control information field are used.

Identification No	Access No	Status	Medium/Unit	Counter 1	Counter 2
4 Byte	1 Byte	1 Byte	2 Byte	4 Byte	4 Byte

Fixed Data Structure in Reply Direction (transmit sequence from left to right)

The **Identification Number** is a serial number allocated during manufacture, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00000000 to 99999999.

The **Access Number** has unsigned binary coding, and is increased by one after each RSP_UD from the slave. With the field **Status** various information about the status of counters, and faults which have occurred, can be communicated - see the table below.

Bit	Meaning with Bit set	Significance with Bit not set
0	Counter 1 and 2 coded signed binary	Counter 1 and 2 coded BCD
1	Counter 1 and 2 are stored at fixed date	Counter 1 and 2 are actual values
2	Power low	Not power low
3	Permanent error	No permanent error
4	Temporary error	No temporary error
5	Specific to manufacturer	Specific to manufacturer
6	Specific to manufacturer	Specific to manufacturer
7	Specific to manufacturer	Specific to manufacturer

Coding of the Status Field

The field **Medium/Unit** is **always** transmitted with least significant byte first and gives the medium measured for both counter states, and the units for each of the two counter states.

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.IDVALUE	0x00000009	7	2040	0	500	String
DR%<16.IDOBJECT	0x0000000a	7	2040	0	500	String
DR%<16.%+U%*256.%+U%*256.MSPEC	0x0000000b	7	2040	0	500	String
DR%<16.%+U%*256.MSPEC	0x0000000b	8	8	0	500	Byte
DR%<16.%+U%*256.MSPEC	0x0000000b	1	16	0	500	Integer
DR%<16.%+U%*256.MSPEC	0x0000000b	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
IDENTITY	0x00000020	7	256	0	0	String

2.2.3 Protdir.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 For licenses bought from Beijer please contact:

Phone: +46(0)40 - 358603

E-mail: support@beijer.se

Website: www.beijer.se

2.3.1.2 For Norway contact our distributor:

Autic System A/S

Phone: +47 333 00 950

E-mail: support@autic.no

Website: www.autic.no

2.3.1.3 For all other countries and licenses bought from PiiGAB:

PiiGAB, Processinformation i Göteborg AB

Anders Carlssons gata 7

S-417 55 Göteborg

Phone: +46(0)31 – 55 99 77

E-mail: support@piigab.se

Website: www.piigab.se

2.3.1.4 General questions

PiiGAB Processinformation i Göteborg AB

Anders Carlssons gata 7

S-417 55 Göteborg

Phone: +46(0)31 – 55 99 77

E-mail: info@piigab.se

Website: www.piigab.se

2.3.2 Documents

The M-Bus: A Documentation

Version 4.8 November 11, 1997

Die Europäische Norm EN13757-1, -2, -3, -4

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
2.00.00.001	Bertil Göransson	Initial V2 release. For further information see chapter 1.1.2

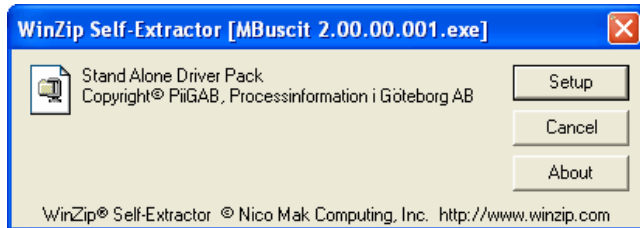
3. Appendix 1 - Installation

3.1 Beginning of installation

The information should be self-explaining during the installation process. But anyway here is little short information about it.

3.1.1 Installing before version 2.02.01.001

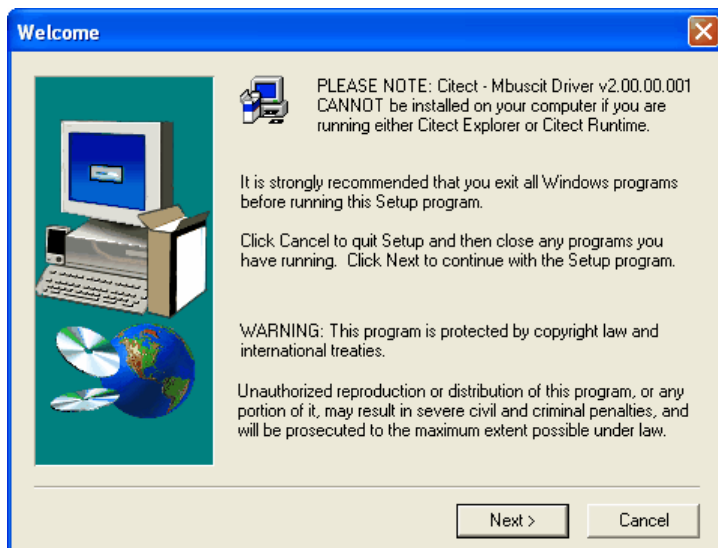
You are starting the installation by activating the self-extracting .exe file Mbuscit2.00.00.001.exe.

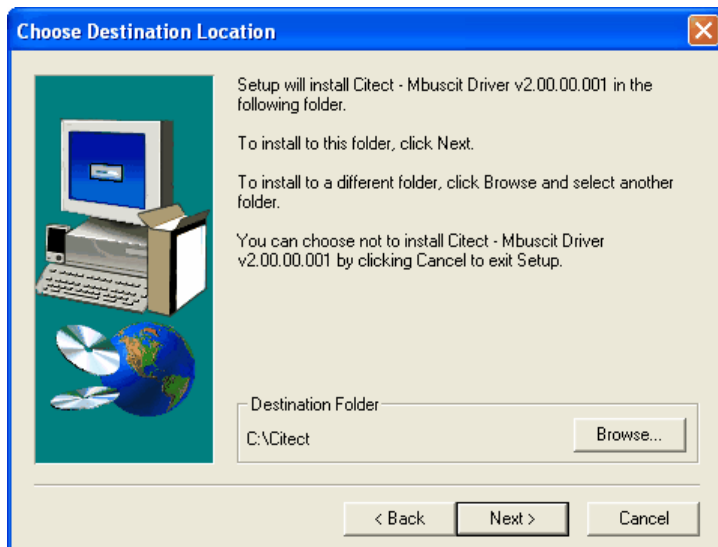


3.1.2 Installing from version 2.02.01.001

Depending of problems with WinZip self extractor in 64 bit operating system we have discarded this software.

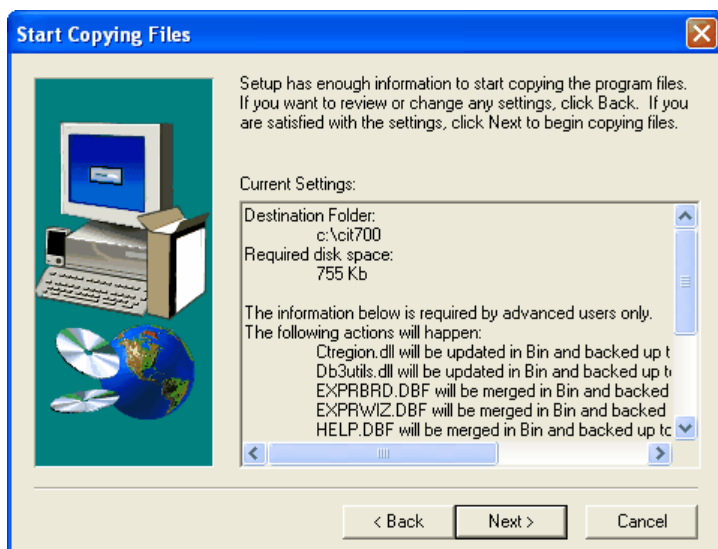
When you have unzipped the driver start the installation from the Setup.exe which you can find in the root path.





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:

c:\cit700

Required disk space:

755 Kb

The information below is required by advanced users only.

The following actions will happen:

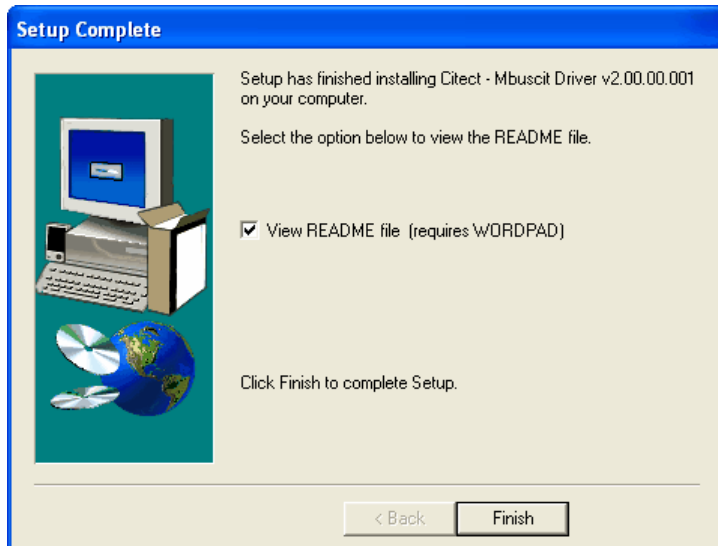
Ctregion.dll will be updated in Bin and backed up to DrvBack
 Db3utils.dll will be updated in Bin and backed up to DrvBack
 EXPRBRD.DBF will be merged in Bin and backed up to DrvBack
 EXPRWIZ.DBF will be merged in Bin and backed up to DrvBack
 HELP.DBF will be merged in Bin and backed up to DrvBack
 MBUSCIT.DBF will be updated in Bin and backed up to DrvBack
 Mbuscit.dll will be updated in Bin and backed up to DrvBack
 Mbuscit.ini will be updated in Bin and backed up to DrvBack
 MbuscitABB.ini will be updated in Bin and backed up to DrvBack

PROTDIR.DBF will be merged in Bin, backed up to DrvBack and copied to User\Include
PROTERR.DBF will be merged in Bin and backed up to DrvBack
Readme.wri will be added to DrvDoc

The following sample project directories will be created:
Mbus

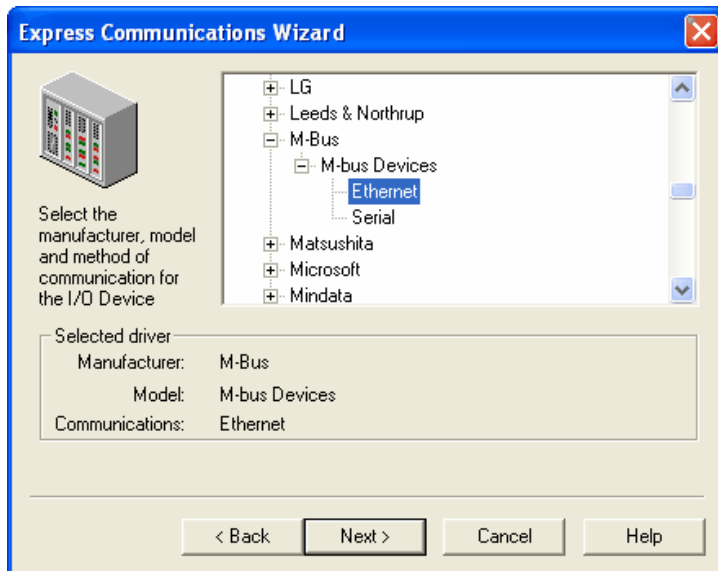
3.3 Setup Complete

When setup is complete you will see this dialog.

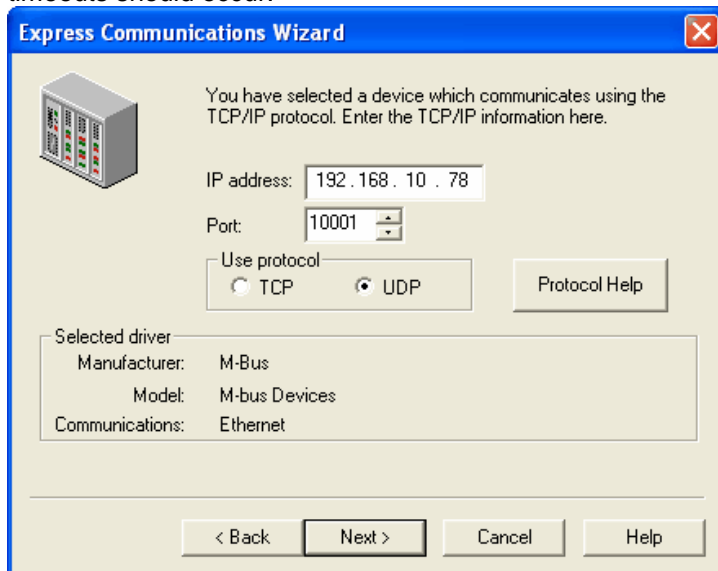


3.4 Express Communication Wizard

After the driver is installed in this you can use the standard Express Communication Wizard when you shall make a new project in Citect.



If you are using Ethernet communication we suggest you to use UDP depending of that M-Bus is a half duplex protocol and with UDP the driver will have full control if any faults as retries and timeouts should occur.

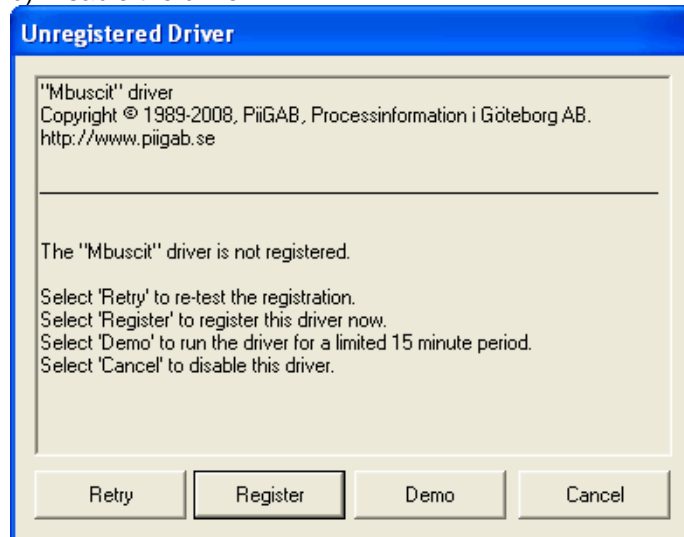


4. Appendix 2 - Software Protection

4.1 Unregistered driver

When a driver is running 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:

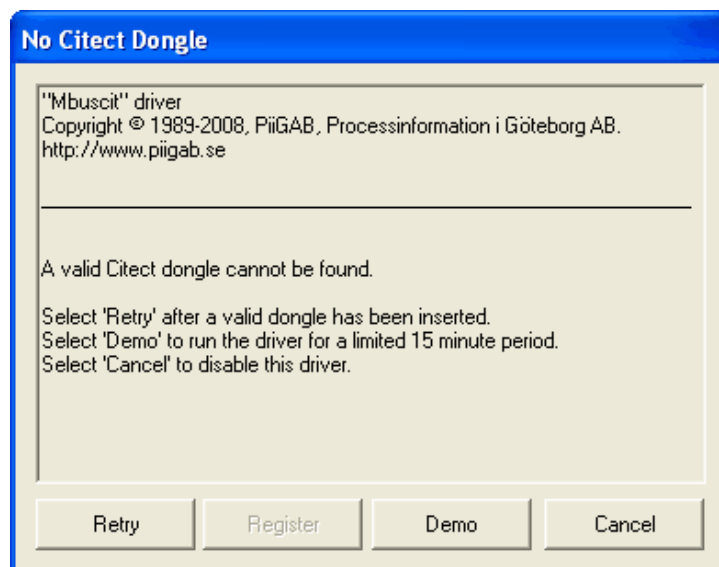
- a) Let the driver re-test the registration settings
- b) Register the driver
- c) Run in a demo mode
- d) Disable the driver



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:

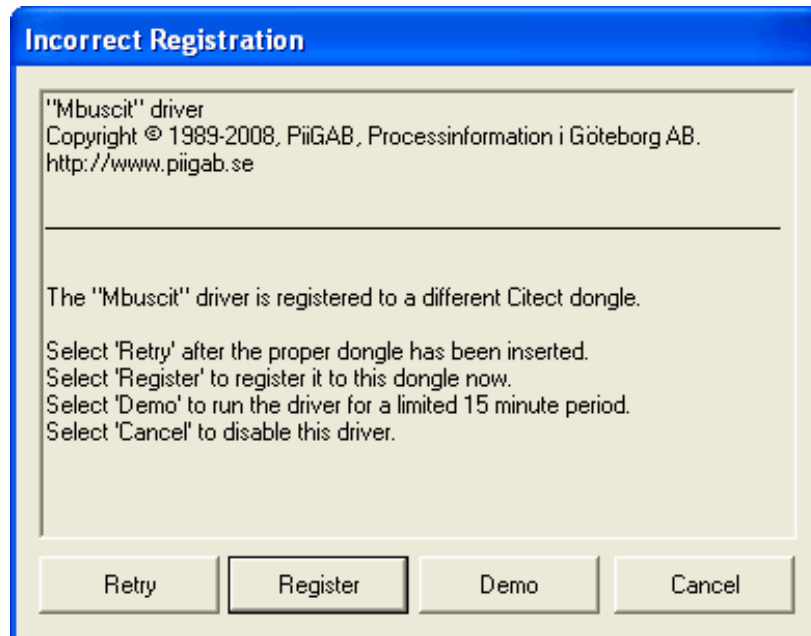
- a) Insert a valid Citect dongle and retry
- b) Run in a demo mode
- c) Disable the driver



4.3 Incorrect registration

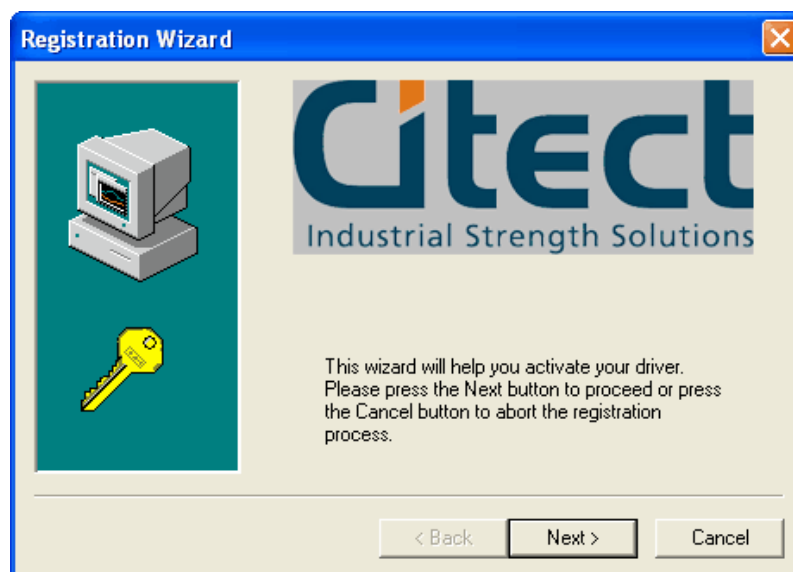
If the Citect dongle does not match the registration key, then the 'Incorrect registration' dialog box will be displayed. This dialog box will prompt you to either:


- Replace the dongle and retry
- Register the driver to the current attached dongle
- Run in a demo mode
- 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.



Registration Wizard 




To activate your driver, please contact the registration centre:


PiiGAB. Registration Centre, Sweden,
Contact telephone number:
+46 31 559977
Business hours 8:00 to 16:00 Monday - Friday
Email: info@piigab.se

The registration centre will require your Citect dongle ID.
Your Citect ID is:

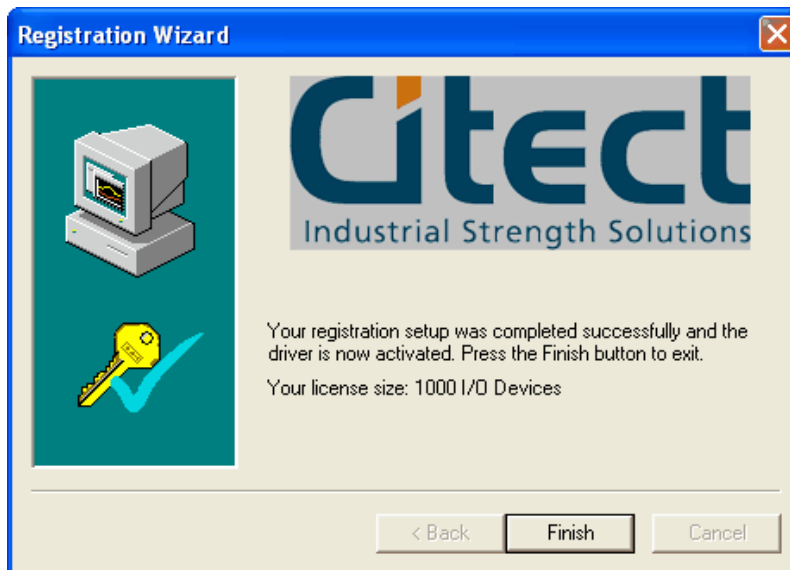
Type the registration number in the following boxes. The
Registration Centre Representative will provide this ID.

- - -

Registration Wizard 



The registration number you have entered is not valid. You
can either press the Back button to retry or press the
Cancel button to abort the registration process.



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 20, 250, 1000 or unlimited numbers of meters.

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.

5.1 Demo projects

You have four demo projects MBusDemo55_E.ctz, MBusDemo55_S.ctz, MBusDemo70_E.ctz, MBusDemo70_S.ctz installed in the MBus catalogue in the Citect path. S respectively E stands for serial respectively Ethernet communication.

5.2 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 a software window titled "Utility_Header" with a menu bar (Pages, Trends, Alarms, File, Tools) and a toolbar. The main content area displays the "Header" information for an M-Bus device. The data is presented in a table-like format with various fields and their corresponding values. A "Status (Digital)" field is visualized as a row of seven colored circles (red and green). The bottom right corner of the window shows the CitectSCADA logo, the time 14:17:07, and the date Mon Apr 07 2008.

Field	Value
Header (String)	00102747 ABB 02 Electricity aa 20 0000
Idnumber (String)	00102747
Manufacturer (String)	ABB
Manufacturer (Uint)	1090
Manufacturer (Uint/Hex)	0442
Version (String)	02
Version (Byte)	2
Version (Byte/Hex)	02
Medium (String)	Electricity
Medium (Byte)	2
Medium (Byte/Hex)	02
Accessnr (String)	aa
Accessnr (Byte)	170
Accessnr (Byte/Hex)	aa
Status (String)	20
Status (Byte)	32
Status (Byte/Hex)	20
Status (Digital)	●●●●●●●
Signature (String)	0000
Signature (Uint)	0
Signature (Uint/Hex)	0000

5.3 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 as it is at the next picture.

Record	Unit	Tariff	Storage	Datatype	Function	Value	Vib
DR1	0	0	0	BCD12	Inst	873277	Energy (Wh*10)
DR2	0	1	0	BCD12	Inst	873277	Energy (Wh*10)
DR3	0	2	0	BCD12	Inst	0	Energy (Wh*10)
DR4	0	0	0	Int8	Inst	1	Current tariff
DR5	0	0	0	BCD8	Inst	1	Transformer ratio
DR6	0	0	0	Int64	Inst	671088704	Error flags (binary)
DR7	0	0	0	Int8	Inst	255	Power fail counter
DR8	0	0	0	Special	Inst	N/A (0x1F)	Manufacturer specific
DR9	0	0	0	Int32	Inst	0	Power (W*0.01)
DR10	0	0	0	Int32	Inst	0	Power (W*0.01) L1
DR11	0	0	0	Int32	Inst	0	Power (W*0.01) L2
DR12	0	0	0	Int32	Inst	0	Power (W*0.01) L3
DR13	0	0	0	Int16	Inst	2262	Voltage (V*0.1) L1
DR14	0	0	0	Int16	Inst	2256	Voltage (V*0.1) L2
DR15	0	0	0	Int16	Inst	2266	Voltage (V*0.1) L3
DR16	0	0	0	BCD4	Inst	0	Current (mA*10) L1
DR17	0	0	0	BCD4	Inst	0	Current (mA*10) L2
DR18	0	0	0	BCD4	Inst	0	Current (mA*10) L3
DR19	0	0	0	BCD4	Inst	5000	Frequency (Hz*0.01)
DR20	0	0	0	Special	Inst	N/A (0x1F)	Manufacturer specific
DR21	0	0	0	Int16	Inst	0	Power factor (*0.001)
DR22	0	0	0	Int16	Inst	13	Phase angle voltage (degrees*0.1) L2
DR23	0	0	0	Int16	Inst	9	Phase angle voltage (degrees*0.1) L3
DR24	0	0	0	Int16	Inst	0	Phase angle current (degrees*0.1) L1
DR25	0	0	0	Int16	Inst	0	Phase angle current (degrees*0.1) L2
DR26	0	0	0	Int16	Inst	0	Phase angle current (degrees*0.1) L3
DR27	2	0	0	Int8	Inst	0	Digital Input (binary)
DR28	2	0	1	Int8	Inst	0	Digital Input (binary)
DR29	2	0	0	BCD12	Inst	0	Cumulation counter
DR30	1	0	0	Int8	Inst	0	Digital Output (binary)
DR31	0	0	0	Special	Inst	N/A (0x0F)	Manufacturer specific
N/A (Empty)							
N/A (Empty)							
N/A (Empty)							

5.4 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.

The screenshot shows the CitectSCADA Records window. The title bar reads 'Records'. The menu bar includes 'Pages', 'Trends', 'Alarms', 'File', and 'Tools'. The toolbar contains various navigation icons. The main content area displays the record ID '00102747 ABB 02 Electricity d3 20 0000' and a table of record data. The table has columns for Record, Unit, Tariff, Storage, Datatype, Function, Value, and Vib. The data rows show various record types and their corresponding values and vibration codes. At the bottom right, the CitectSCADA logo is visible along with the time '14:20:58' and date 'Mon Apr 07 2008'.

Record	Unit	Tariff	Storage	Datatype	Function	Value	Vib
DR1	0	0	0	BCD12	Inst	873288	VIF04
DR2	0	1	0	BCD12	Inst	873288	VIF04
DR3	0	2	0	BCD12	Inst	0	VIF04
DR4	0	0	0	Int8	Inst	1	FF13
DR5	0	0	0	BCD8	Inst	1	FF12
DR6	0	0	0	Int64	Inst	671088704	FD17
DR7	0	0	0	Int8	Inst	255	FF18
DR8	0	0	0	Special	Inst	N/A (0x1F)	1F Manufacturer specific
DR9	0	0	0	Int32	Inst	0	VIF29
DR10	0	0	0	Int32	Inst	0	VIF29 FF01
DR11	0	0	0	Int32	Inst	0	VIF29 FF02
DR12	0	0	0	Int32	Inst	0	VIF29 FF03
DR13	0	0	0	Int16	Inst	2258	FD48 FF01
DR14	0	0	0	Int16	Inst	2252	FD48 FF02
DR15	0	0	0	Int16	Inst	2261	FD48 FF03
DR16	0	0	0	BCD4	Inst	0	FD5a FF01
DR17	0	0	0	BCD4	Inst	0	FD5a FF02
DR18	0	0	0	BCD4	Inst	0	FD5a FF03
DR19	0	0	0	BCD4	Inst	5000	FF59
DR20	0	0	0	Special	Inst	N/A (0x1F)	1F Manufacturer specific
DR21	0	0	0	Int16	Inst	0	FF60
DR22	0	0	0	Int16	Inst	8	FF42 FF02
DR23	0	0	0	Int16	Inst	16	FF42 FF03
DR24	0	0	0	Int16	Inst	0	FF4a FF01
DR25	0	0	0	Int16	Inst	0	FF4a FF02
DR26	0	0	0	Int16	Inst	0	FF4a FF03
DR27	2	0	0	Int8	Inst	0	FD1b
DR28	2	0	1	Int8	Inst	0	FD1b
DR29	2	0	0	BCD12	Inst	0	FD61
DR30	1	0	0	Int8	Inst	0	FD1a
DR31	0	0	0	Special	Inst	N/A (0x0F)	0F Manufacturer specific
N/A (Empty)							
N/A (Empty)							
N/A (Empty)							