1 Introduction

This document, which is based on references [11] and [12], gives detailed and actual information about the **M-Bus,** which is a low cost home electronic system (HES).
This documentation about the M-Bus is published by the M-Bus Usergroup, which is an international organization of users and producers of M-Bus devices. The usergroup meets several times a year to discuss problems and developments in hardware and software. Recommendations, agreements, examples and explanations are included in this paper as well as parts of the standards itself. Among other things the actual version of this document in Winword™ format can be downloaded from the M-Bus Mailbox.

M-Bus Usergroup:
Prof. Dr. Horst Ziegler
Fachbereich Physik
Universität-GH Paderborn
Warburgerstr. 100
Germany 33098 Paderborn
Phone: +49 5251 / 602735

WWW:
http://fb6www.uni-paderborn.de/M-Bus/

M-Bus Mailbox:
Phone: +49 5251 / 603830
Parameter: 300..14400 bps, 8N1
Sysop: Carsten Bories, Phone 602750

The **M-Bus** (Meter Bus) was developed to fill the need for a system for the networking and remote reading of utility meters, for example to measure the consumption of gas or water in the home. This bus fulfills the special requirements of remotely powered or battery driven systems, including consumer utility meters. When interrogated, the meters deliver the data they have collected to a common master, which can, for example, be a hand-held computer, connected at periodic intervals to read all utility meters of a building. An alternative method of collecting data centrally is to transmit meter readings via a modem.
Other possible applications in home electronic systems for the M-Bus are alarm systems, flexible illumination installations and heating controlling.

REMARKS:

- ♣ Text parts or topics marked with this symbol are new or changed information since last version 4.7.1 of this document.

2 The Basics of Serial Bus Systems

2.1 Applications and Definitions

The methods by which data processing systems communicate with each other are classified according to the distances involved. With world-wide networks the term used is Global Area Networks (GAN), whereas networks covering continents or large land masses are known as Wide Area Networks (WAN); Local Area Networks (LAN) are concerned with distances up to a few kilometers, and are limited to specific geographical areas, such as laboratories, office buildings and company premises. Such local networks are used, for example, to link terminals, computers, measuring equipment and process automation modules with one another.

In the majority of local networks, one or other of the following methods (topologies) are used to link the components in a system:

- Star Topology
  Each component is linked to a central processor unit with an individual transmission line. The equipment can transmit to the central unit either sequentially or simultaneously. One disadvantage of this arrangement is the increased requirement for cabling.
- Ring Topology
  In this case, the components are connected to one another in a ring, and the data are transferred from point to point. This topology has the disadvantage that, should a single equipment fail, the complete network will be out of action.
- Bus Topology
  The components are connected together with a common transmission line, with the result that at one instant only one equipment can transmit data. This topology is very cost-effective, it will not be disturbed if one of the components fails, and it allows the transmission of data to all components (Broadcasting) or to specific groups in the system (Multicasting).

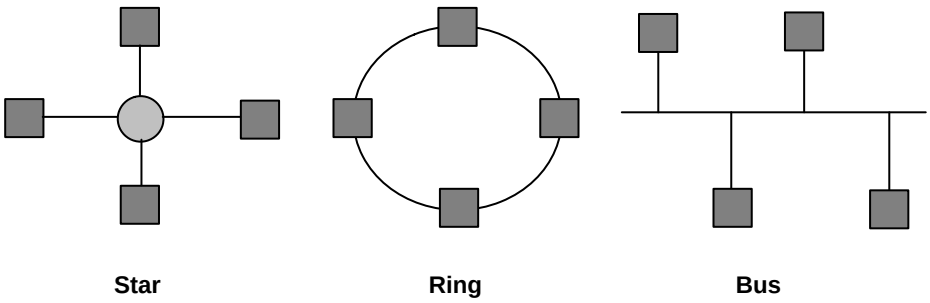

Fig. 1 *Network Topologies*

A serial bus can be defined as a transmission path over which the participants transmit their data serially (i.e. bit after bit), sequentially in time and using a common medium. In contrast, in parallel bus systems the individual bits which form a character are transmitted simultaneously by a certain number of data lines. This results in increased costs for cable and connectors; the transmission time is shorter than with a serial bus.

2.2 Basic Functions of Bus Systems

The following diagram is intended to provide an overall view of the various forms of serial bus systems:

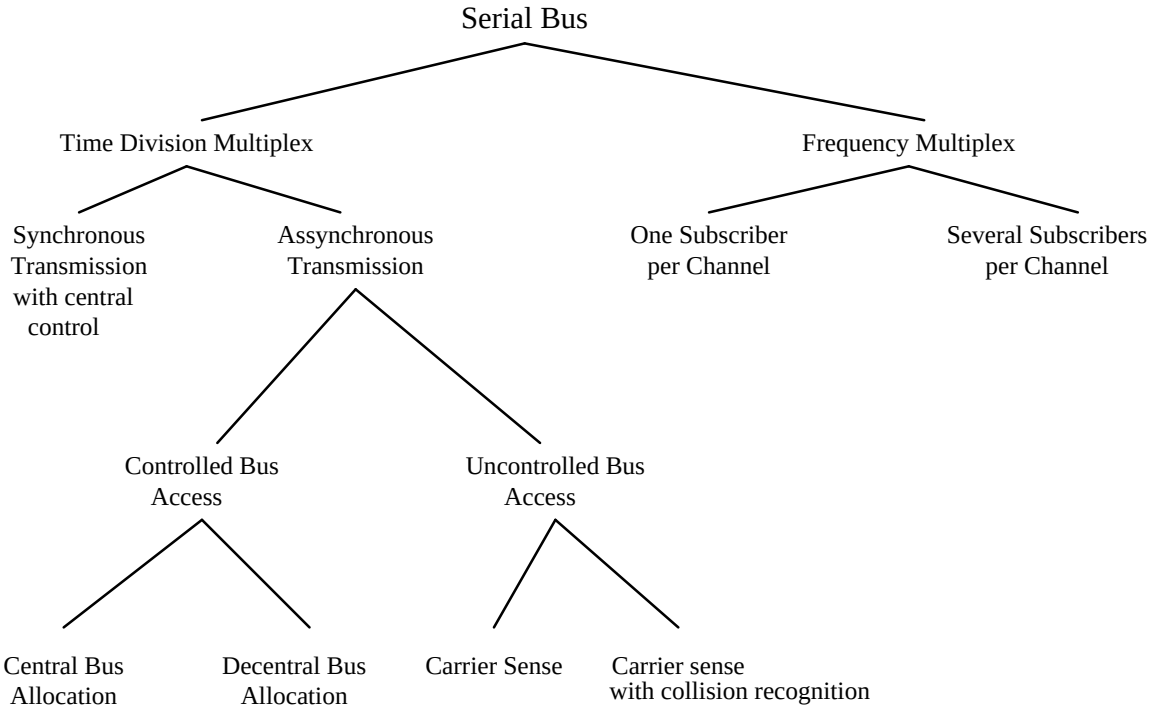

Fig. 2 *Classification of Serial Bus Systems According to Transmission and Access Techniques* [1]

The first subdivision can be made according to the multiplex technique which is used. With frequency multiplex, the frequency spectrum of the transmission medium is divided into frequency bands, each representing a channel. Each participant is then allocated a channel. In the next section, the kind of synchronization and access techniques which are used will be described in order to classify serial bus systems using time division multiplex.

2.2.1 Access Techniques

Since in bus systems the transmission medium is used by all participants together, account must be taken of their various transmission requirements. The methods used by participants who want to transmit over the bus are known as access techniques. These techniques must ensure that several stations do not transmit simultaneously, and so cause bus conflicts or collisions, and that each participant can transmit for at least a certain minimum time. The sharing of the bus among stations who want to transmit is implemented with an allocation logic system.

With central allocation logic, the central bus controller receives a request to use the bus and then takes the decision as to whether, and if so when, the user can occupy the bus. For this purpose various methods are used to register the bus occupation request:

- direct registration by means of an individual branch line to each equipment
- periodical interrogation (polling) of the participants as to their transmission needs
- requests sent on a common line with identification of the sender
- allocation of the bus according to a predetermined time frame without taking account of individual requirements

The advantage of central allocation logic is the reduced complexity which is required at individual stations.

With decentralized allocation logic, each participant is provided with functions which allow him to recognize whether the bus is already in use. There are various methods which can be used to determine whether the bus is occupied:

- mutual interrogation of stations by means of a request line for each station
- periodical bus allocation, by passing "ownership" of the bus from station to station
- CSMA (Carrier Sense Multiple Access): The participants have the ability to check whether the bus is transmitting data, and to transmit their own data if it is found to be free. To avoid collisions, which could arise as a result of signal transit times, with certain bus systems (e.g. Ethernet) the stations are able to use their own data on the bus to determine whether there is a bus conflict. In such a case, the transmission will be interrupted and repeated after an appropriate time interval.

A higher degree of logic complexity at each station is needed to implement decentralized allocation logic, but this system also has the advantage that a fault in the central bus controller will not result in a complete breakdown of the bus.

2.2.2 Synchronization of Participants

Synchronization is to be understood as the coordination in time of the communicating participants, with regard to signal transmission and reception. The various methods of synchronization can be classified into data transmission which is synchronous and that which is asynchronous (see Figure 3).

With synchronous transmission, a stable clock signal is supplied either by the central station or one of the communicating partners, which serves to measure transmission times. With asynchronous transmission, a distinction must be made between techniques with and without signal acknowledgment. Where there is signal acknowledgment (handshake), the sender shows with a specific signal on the line that he has data to send, and waits for an acknowledgment from the receiver. Techniques without acknowledgment use a transfer clock on a special line for parallel bit transmission, or start and stop bits to frame a character for bit-serial transmission.

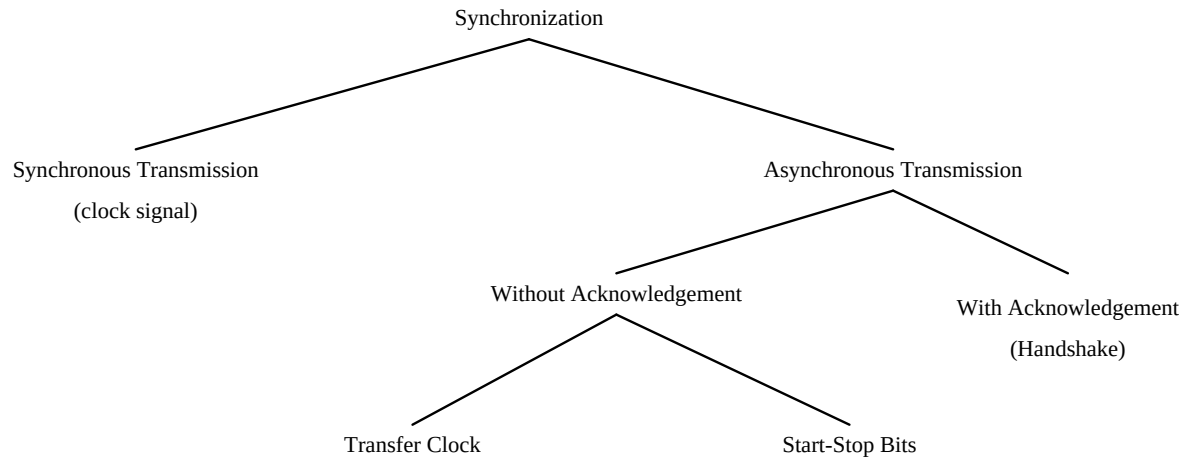

Fig. 3 *Classification of Synchronization Techniques*

2.2.3 Error Processing

The reasons for transmission errors in bus systems are widely known. These include in particular electromagnetic interference from outside, for example: inductive coupling at mains frequencies; high-frequency interference as a result of sparking at the brushes of motors or arcs of discharge lamps; capacitive coupling to other lines; or directly coupled currents from ground loops as a result of multiple grounds.

A bus system must ensure that transmission errors are recognized and corrected. For this reason additional information is supplied with the data to be transmitted, which allows the data to be checked on reception.
Particularly with asynchronous transmission, an additional parity bit is often transmitted with each character. This parity bit is constructed so that the parity conditions (an even number of ones, or an odd number of ones) are fulfilled. Another method is the creation of a block check character from specific mathematical operations e.g. addition without carry (Check Sum), which is derived from all the data. The receiving station can detect whether there have been transmission errors by comparing the check character it has received with one which it has calculated itself. The parity bit allows only the recognition of an odd number of faulty bits.
In order to correct errors the recipient sends an acknowledgment, which indicates that the transmission has been either error free, or that there have been transmission errors. For the same purpose the transmitter checks that the receiver acknowledges the reception of data in a certain period of time. If the time limit is exceeded (Timeout), or if a transmission error has been reported, then the sender repeats the transmission a predetermined number of times.
The Hamming Distance is used in order to specify the security of a character code; this is the number of errors (minus one) which can be recognized for all cases.

2.3 The OSI Reference Model

The ISO-OSI reference model provides a basis for the development of standards for **O**pen **S**ystems **I**nterconnection (**OSI**). This model devised by the "**I**nternational **O**rganization for **S**tandardization" (**ISO**) is intended to ensure that information from systems made by various manufacturers, and having different architecture, can be exchanged and interpreted in accordance with standardized procedures.
This model arranges the communications functions in seven layers, each of which has a virtual connection to the appropriate layer of the communicating partner. Only on the lowest layer (Layer 1) is there a physical connection for exchanging signals. Each layer, with the exception of Layer 1, obtains the necessary service from the layer below it. The OSI model merely defines the servicing and functions of the layers, but not the technical realization (the protocols) within the layers.

Two user programs can exchange information on Layer 7, if there is agreement between them (i.e. there are protocols) on the following points [2]:

- the representation of information in Layer 6
- the flow of communications (contents and form) in Layer 5
- the completeness of the information and the security of transport in Layer 4
- the way information should be transferred through the network in Layer 3

- the security of transmission in Layer 2
- the physical medium in Layer 1

7	Application Layer	Application Oriented Layers
6	Presentation Layer	
5	Session Layer	
4	Transport Layer	Transport Oriented Layers
3	Network Layer	
2	Data Link Layer	
1	Physical Layer	

Fig. 4 *The Seven Layers of the OSI Model*

The functions of the individual layers shown in Figure 4 will now be explained in more detail:

Physical Layer

The basic physical connection between the communicating partners takes place in this lowest layer. The mechanical and electrical coupling to the transmission medium is determined here, by specifying (among other things) the cable, the distances involved, the pinning of connectors, and the way the bits are represented.

Data Link Layer

This layer is responsible for assuring that a reliably operating connection is made between two participants. For this purpose the protocol of this layer determines the methods for protecting transmissions, the telegram structure, methods of accessing the transmission medium and for the synchronization and addressing of participants. By making use of the procedures described in Section 2.2.3 it should be possible to identify and correct faults in the Data Link Layer.

Network Layer

The network layer undertakes the choice and implementation of the best transmission route in a network between the communicating parties, and provides this service (Routing) to the Transport Layer. This function is of particular significance when different networks need to be connected by means of Gateways.

Transport Layer

The transport layer represents the boundary between the application oriented layers 5 to 7, and the transport oriented layers 1 to 4. Its job includes guiding the information through the network, controlling the flow of information and the grouping into individual packets.

Session Layer

The session layer provides procedures for the opening, the orderly progressing, and the termination of a communication "session". In this is included also the control of the dialogue between systems: that is, the determination of their respective transmission prerogatives.

Presentation Layer

The data of the application are converted in the presentation layer into a data format which the receiving application can interpret. This layer thus implements the matching of data formats and the conversion of codes.

Application Layer

This top layer represents the interface between the open system and the user. It offers the user or his program a service allowing him to work easily with the system. Application programs which need to be developed can thus access the functions of the open system via the protocol of the application layer.

In the following diagram the route to be followed by data from the transmitting to the receiving application can be seen, indicated by the continuous arrows. At the transmitting side information (Overhead) which is necessary for transmission and processing is added to the actual data in each layer; at the receiving side this information is removed again in the reverse order after processing.

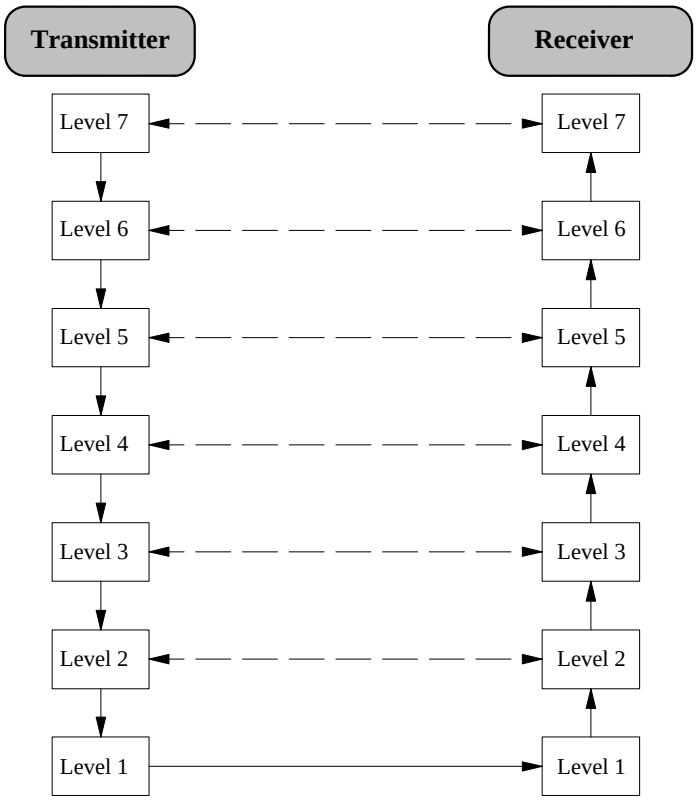

Fig. 5 *Data Transmission in Accordance with the OSI Model*

3 Overview of the M-Bus

3.1 Requirements of a Bus System for Consumer Utility Meters

Of the various possible topologies which might be considered for reliable and cost-effective networking consumer utility meters, only the bus topology (see Section 2.1) is in fact suitable. The requirements which are made by meters on such a bus system will now be explained.

The most important requirement is the interconnection of many devices (several hundred) over long distances - up to several kilometers. Since the data sent by the meters are used for end user billing, a high degree of transmission integrity is required for the bus. On the other hand it is possible to dispense with high speed of transmission, since usually only a relatively small amount of information must be transferred. To ensure this high degree of transmission integrity, the bus must be exceptionally insensitive to external interference, as a result of capacitive or inductive coupling. In order to avoid ground loops, the slave should be electrically isolated.
A further requirement for the bus is low cost for the complete system. The transmission medium which is used should therefore not require shielding, and the cost of individual meters can be minimized by using as few components as possible and by remotely powering the slaves from the bus. In addition the costs of installing and servicing the system must be taken into account: These can be reduced by incorporating protection against reversed polarity, and making provision for the connection of additional facilities (Life Insert) during operation of the bus system.

3.2 The M-Bus in the OSI Model

Since no bus system was available which met the requirements detailed in Section 3.1[1], the Meter-Bus (M-Bus) was developed by Professor Dr. Horst Ziegler of the University of Paderborn in cooperation with Texas Instruments Deutschland GmbH and Techem GmbH. The concept was based on the ISO-OSI Reference Model, in order to realize an open system which could utilize almost any desired protocol.

Since the M-Bus is not a network, and therefore does not - among other things - need a transport or session layer, the levels four to six of the OSI model are empty. Therefore only the physical, the data link , the network and the application layer are provided with functions.

[1] see Reference [3], Chapter 2.1, Field Bus Systems

Layer	Functions	Standard	Chapter
Application	Data structures, data types, actions	EN1434-3	6
Presentation	empty		
Session	empty		
Transport	empty		
Network	extended addressing (optional)	-	7
Data Link	transmission parameters, telegram formats, addressing, data integrity	IEC 870	5
Physical	cable, bit representation, bus extensions, topology, electrical specifications.	M-Bus	4

Fig. 6 *The M-Bus layers in the OSI-Model*

Because changing of parameters like baudrate and address by higher layers is not allowed in the ISO-OSI-Model, a **Management Layer** beside and above the seven OSI-Layers is defined:

MANAGEMENT LAYER	
Application Layer	
Presentation Layer	
Session Layer	
Transport Layer	
Network Layer (if address = 253)	Address 253 / Enable Disable CI=$52/$56
Data Link Layer	
Physical Layer	Address 254 (255)

Fig. 7 *Management-Layer of the M-Bus*

So the address 254 and perhaps 255 are reserved for managing the physical layer and the address 253 (selection) for network layer (see chapter 7), which is only used in certain cases. With this new layer we can directly manage each OSI-layer to implement features, which do not conform to the OSI-Model.

4 Physical Layer

More and detailed informations about the specifications of the physical layer are listed in the document 'WG4N85R2.DOC' ♣.

4.1 Principles of Operation

The M-Bus is a hierarchical system, with communication controlled by a master (Central Allocation Logic). The M-Bus consists of the master, a number of slaves (end-equipment meters) and a two-wire connecting cable: see Figure 8. The slaves are connected in parallel to the transmission medium - the connecting cable.

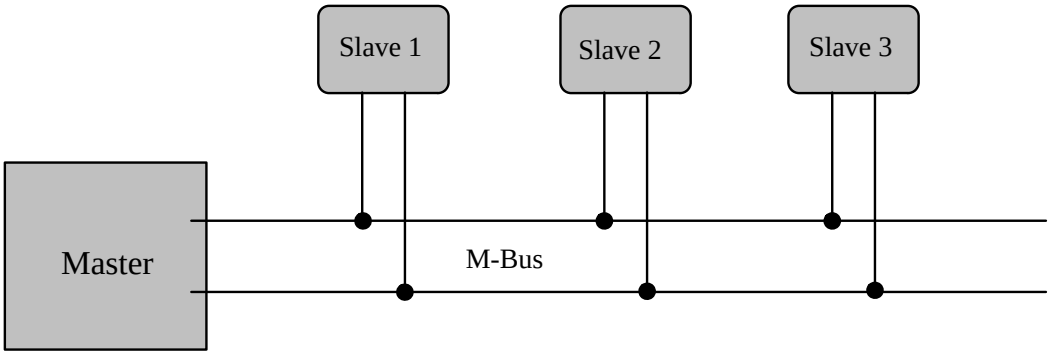

Fig. 8 *Block diagram showing principle of the M-Bus System*

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), which is a part of the master; when a logical "0" (Space) is sent, the repeater 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.

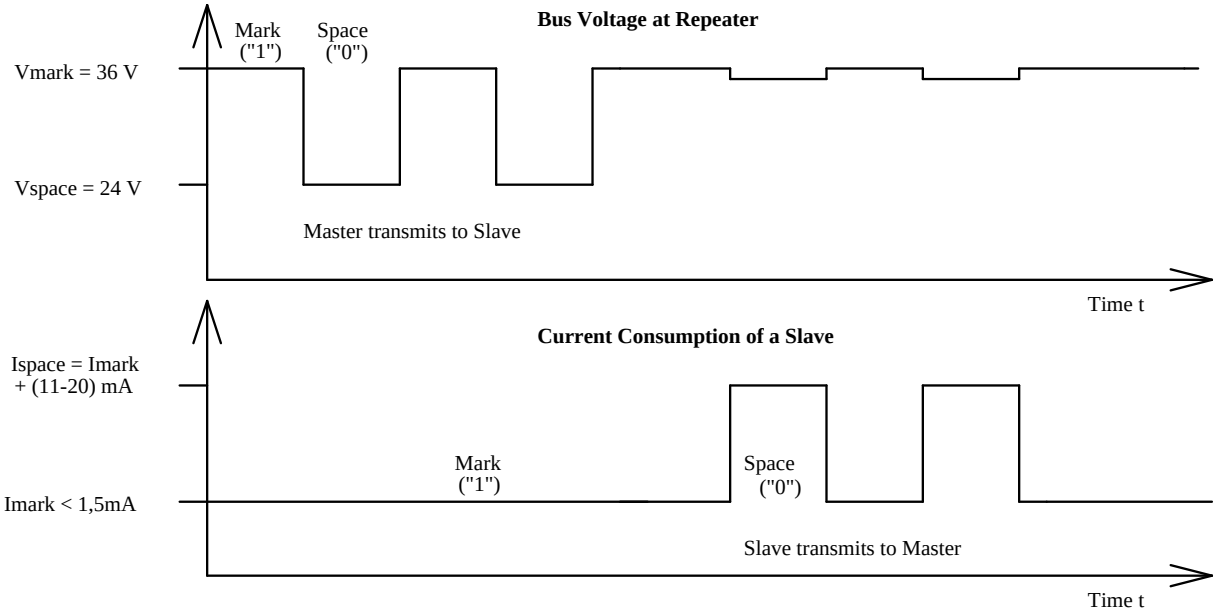

Fig. 9 *Representation of bits on the M-Bus*

The transmission of a space by a slave results in a slight reduction in the bus voltage at the repeater due to output impedance, as can be seen in Figure 9.

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.

4.2 Specifications for Bus Installations

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. From now on, the local repeater will simply be termed the "repeater", and the combination of PC and local repeater termed the "master".

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 Ω. 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.

Plug

There is so far no standard or recommendation for a M-Bus plug to connect the meters to the bus system, but the Usergroup investigates in defining a proper connector. Three different plugs have to be defined for the connector at a) the installation mode b) meter to fixed installation and c) meter to handheld connection.

4.3 Specifications of the Repeaters

See chapter 'Electrical Requirements Master' in the document 'WG4N85R2.DOC' ♣.

4.4 Slave Design

The requirements for slaves are listed in the paper 'WG4N85R2.DOC' ♣. The following characteristics are part of it:

- **Transmission Characteristics**

  The slaves are designed to be constant current sinks with two different currents, whereby the current which is "sunk" must not vary by more than 0.2 % for 1 V voltage change on the bus. In order to transmit a Mark, a so-called Unit Load consisting of a constant current of 1.5 mA maximum is specified. If the slave needs more current, an appropriate number of additional Unit Loads must be used. When sending a Space, the slave increases its current consumption by 11-20 mA. In order to receive data, the slave detects the maximum value Vmax of the bus voltage, which can be between 21 V and 42 V. With a bus voltage of more than Vmax - 5.5 V, a Mark should be registered, and with a voltage of less than Vmax - 8,2 V, a Space should be registered.

- **Remote Powering**

  The bus interface - that is, the interface between the slave and the bus system - must take the current it needs from the bus system. If possible, the complete slave should be fed from the bus; in this case, should the bus fail, it must automatically switch over to battery operation, or the significant data must be saved. If the slaves are operated only from batteries, it is necessary that a battery life of several years should be attained, in order to reduce maintenance costs.

- **Protective Measures**

  The bus interfaces of the slaves are polarity independent: that is, the two bus lines can be interchanged without affecting the operation of the slaves. Besides protection aspects, this also results in simplified installation of the bus system. In order to maintain correct operation of the bus in case of a short circuit of one of the slaves as metioned before these must have a protection resistor with a nominal value of (430±10)Ω in their bus lines. This limits the current in the case of a short circuit to a maximum of 100 mA (42 V / 420 Ω), and reduces the energy converted into heat in the bus interface.

M-Bus Transceiver TSS721

In order to meet the requirements for the slaves mentioned above, an IC was developed by Texas Instruments Deutschland GmbH, namely the Transceiver (i.e. Transmitter and Receiver) TSS721. The use of the TSS721 in M-Bus slaves as the interface to the bus reduces the number of components needed, and therefore the cost of slaves. Apart from the transmission and reception of data in accordance with the M-Bus specification, this IC also provides translation from and to the operating voltage of the microprocessor to which it is connected, in order to be able to communicate with it. The communication can take place at baudrates from 300 to 9600 Baud. Additional features include integrated protection against reversed polarity, a constant 3.3V power supply for the microprocessor, and the prompt indication of failure of the bus voltage.

be dispensed with, the bus voltage may also be connected directly between the VB and GND pins.

- **Reception**
  The comparator circuit TC3 is provided to detect signals from the master; it adjusts itself to the Mark voltage level with the help of the capacitor SC. This capacitor is charged up to 8.6 V under the Mark voltage when in the Mark state, and discharged during the Space state. The ratio of charge to discharge current is more than 30 to make any kind of UART protocol work indepedently of the data contents. The voltage across the capacitor SC results in dynamic matching of the comparator to the Mark level. From the relationship between the charging and discharging current results the requirement in the transmission protocol that at least every eleventh bit (with adequate certainty) must be a logical 1 - that is, a Mark. This guarantees that SC is not discharged too much, and that matching to the Mark voltage level is always effective. With a voltage of 7.9 V under the Mark level, the TSS721 gives a logical 0 to the TX pin (0 V) and to the inverted TXI pin (Supply Voltage).

- **Transmission**
  The signal from the microprocessor applied to the RX Pin or RXI Pin (inverted) is converted into a current by TC4 and the constant current source CS3. When there is a Mark at the inputs (RX or RXI), the quiescent current is taken from the bus with the help of the constant current source. If however the processor transmits a Space, then TC4 switches on the constant current source CS3, and consequently the additional pulse current. The quiescent current can be adjusted over a certain range with the resistor Ridd, and the pulse current adjusted with Ris. In order to allow the processor to recognize collisions, the signal on the RX(I) pins is echoed on the TX(I) pins.

- **Powering of the Processor**
  The TSS721 provides a nominal voltage of 3.3 V at its VDD Pin, in order to supply power to a microprocessor. When limited to a standard load, according to the data sheet this processor may however consume an average current of about 600 µA. For pulse current requirements, use is made of the reservoir capacitor STC. When connection is made to the bus, this capacitor will be charged at up to 7 V, and the power supply at the VDD pin is activated at V_{STC} = 6 V. The TSS721 signals the failure of the bus voltage at the PF-Pin (power fail), so that the processor has time to store its data in e.g. an EEPROM, whilst powered by the reservoir capacitor. In addition, the transceiver permits the connection of a battery to the VDD Pin should the bus fail, by means of FET at the VS Pin (voltage switch). In such a case, and when the microprocessor is powered solely with a battery, the voltage must also be taken to the BAT Pin in order to match into the TSS721.

Figure 11a) shows three alternative operating modes for the TSS721 which can be used to power a microprocessor. It shows that the processor can be supplied exclusively by the transceiver (remote supply), normally from the TSS721 and with bus failure from a battery (remote supply/battery support), or only by the battery. Few external components are needed to build a complete slave with the TSS721, apart from the microprocessor or microcontroller and the components specifically required for the sensing elements. Besides Fig. 11b) shows a basic optocoupler application.

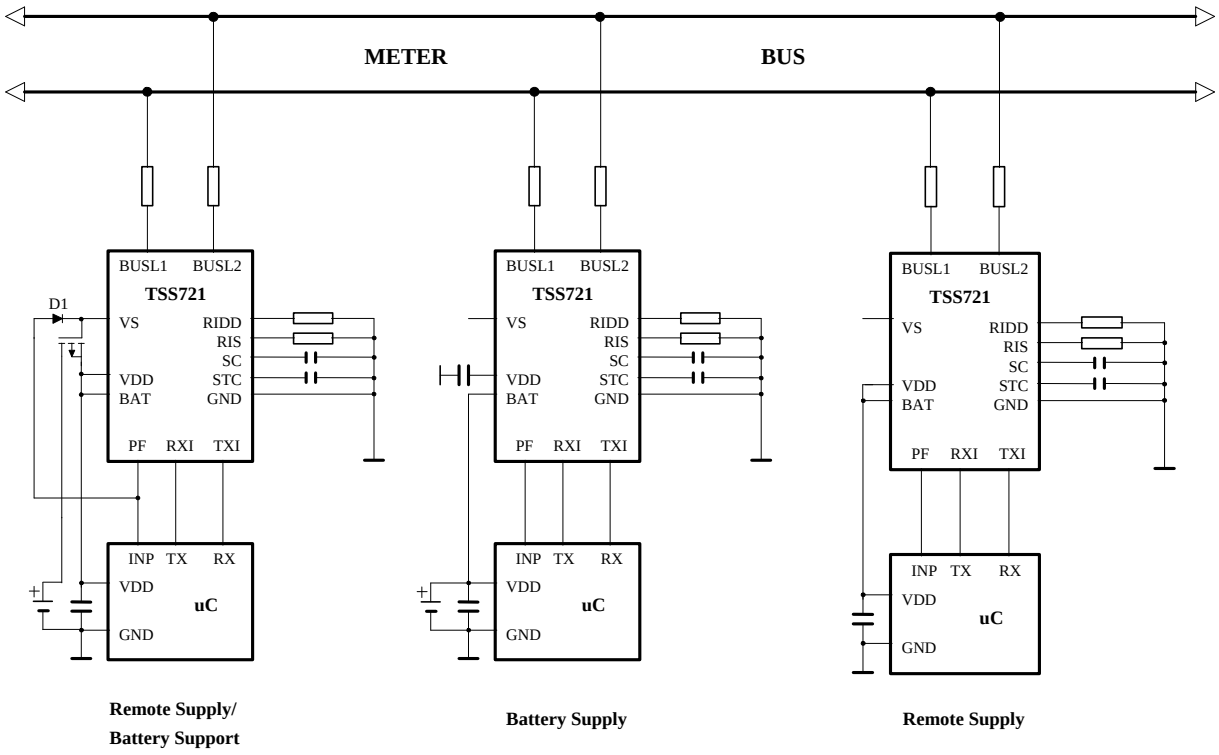

Fig. 11a) *Operating Modes of the TSS721 for Powering a Microcontroller [4]*

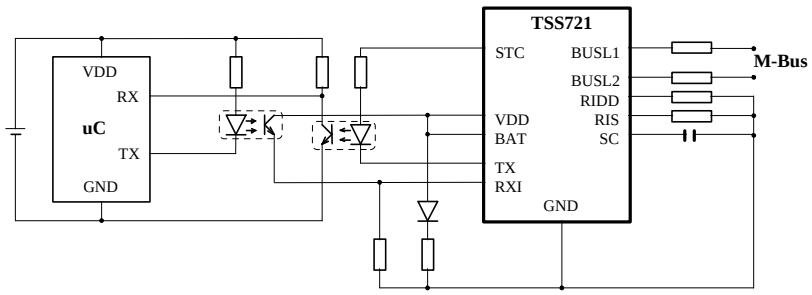

Fig. 11b) *Basic optocoupler application [4]*

5 Data Link Layer

The physical layer makes certain demands on the data link layer. Besides half-duplex asynchronous serial transmission with data rates between 300 and 9600 Baud, these include the requirement that at least every eleventh bit should be a logical 1, and also that there should be a Master-Slave structure, since the slaves can not communicate with each other.
The protocol of the data link layer is based on the international standard IEC 870-5, which defines the transmission protocols for telecontrol equipment and systems. The M-Bus protocol described below derives from the above standard, but doesn´t use all the IEC functions.
The signal quality requirements for master and slaves are listed in the document 'WG4N86R2.DOC' ♣ (available in the mailbox and via internet). It is based on the international standard IEC 7480.

5.1 Transmission Parameters

This 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 Figure 12, the transmission of a byte in calling and replying direction is represented.

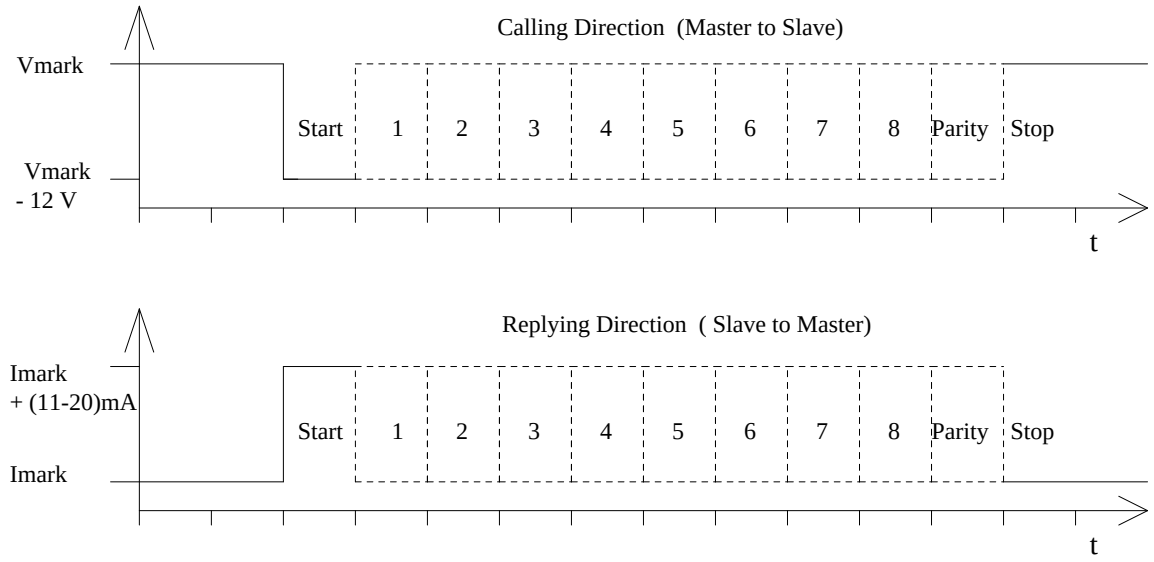

Fig. 12 *Transmission of a Character in Calling and Replying Direction*

5.2 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. The data integrity class is a measure of the quotient between the rate of undetected false messages and the probability of faulty bits during transmission. For the data integrity classes mentioned above, various format classes have been identified, in which measures to recognize transmission faults are defined. 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, and in figure 13, the telegram formats used for the M-Bus will now be explained.

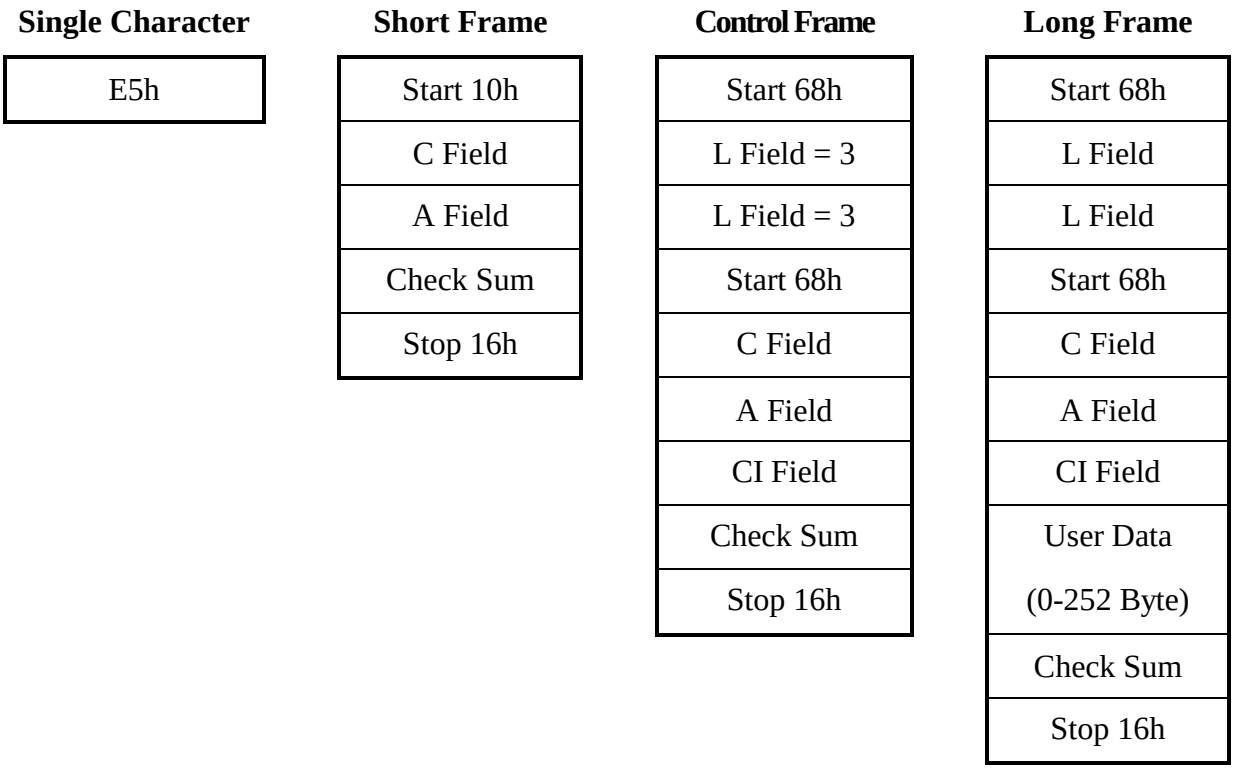

Fig. 13 *Telegram Formats of the M-Bus Protocol*

- **Single Character**
  This format consists of a single character, namely the E5h (decimal 229), and serves to acknowledge receipt of transmissions.
- **Short Frame**
  This format with a fixed length begins with the start character 10h, and besides the C and A fields includes the check sum (this is made up from the two last mentioned characters), 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). The L field gives the quantity of the user data inputs plus 3 (for C,A,CI). After the user data inputs, the check sum is transmitted, which is built up over the same area as the length field, and in conclusion the stop character 16h is transmitted.
- **Control Frame**
  The control sentence conforms to the long sentence without user data, with an L field from the contents of 3. The check sum is calculated at this point from the fields C, A and CI.

5.3 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.

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. Figure 14 shows the coding of the individual bits of the C field:

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

Fig. 14 *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 - see 5.4), 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. For more about the FCB see chapter 5.5.

In the replying direction, both these bits can undertake other tasks. The DFC (data flow control) serves to control the flow of data, in that the slave with a DFC=1 indicates that it can accept no further data. With an ACD bit (access demand) with a value of 1, the slave shows

that it wants to transmit Class 1 data. The master should then send it a command to request Class 1 data. Such Class 1 data is of higher priority, which (in contrast to Class 2 data) should be transmitted as soon as possible. The support of Class 1 data and the bits DFC and ADC is not required by the standard.

The bits 0 to 3 of the control field code the true function or action of the message. Table 1 shows the function codes used in the calling and the replying directions. The functions shown in this table will be explained in more detail in the next section. All additional function codes defined in IEC 870-5-2 can also be used.

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/Control 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 Class1 Data (see 8.1: Alarm Protocol)
RSP_UD	00AD 1000	08/18/28/38	Long/Control Frame	Data Transfer from Slave to Master after Request

Table 1 *Control Codes of the M-Bus Protocol (F : FCB-Bit, A : ACD-Bit, D : DFC-Bit)*

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 adressing has been performed in the Network Layer (see chapter 7) instead of Data Link Layer. The remaining addresses 251 and 252 have been kept for future applications.

CI Field (control information field)

The control information field is already a part of the Application Layer, and is described in more detail in section 6.1. It was included in the telegram format used, in order to distinguish between the formats of the long and the control frames. The control information allows the implementation of a variety of actions in the master or the slaves.

Check Sum

The Check Sum serves to recognize transmission and synchronization faults, and is configured from specific parts of the telegram. These parts are mentioned when presenting the individual telegram formats in 5.2. The Check Sum is calculated from the arithmetical sum of the data mentioned above, without taking carry digits into account.

5.4 Communication Process

The Data Link Layer uses two kinds of transmission services:

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

♣ After the reception off a valid telegram the slave has to wait between 11 bit times and (330 bit times + 50ms) before answering (see also EN1434-3).

Send/Confirm Procedures

- SND_NKE → Single control character
  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 FCV=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 → Single control character
  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 ("$E5"), or by omitting a confirmation signal that it did not receive the telegram correctly.

Request/Respond Procedures

- REQ_UD2 → RSP_UD
  The master requests data from the slave according to Class 2. 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 REQ_UD2 telegram does not match.

Minimum Communication

According to the European standard EN1434-3, as a minimum for communication the procedures REQ_UD2 / RSP_UD and ♣ SND_NKE / $E5 are needed. All other functions are optional.

Transmission Procedures in case of faults

A fault in a received telegram can be detected by the receiver (master or slave), by checking the following points:

- Start /Parity /Stop bits per character
- Start /Check Sum /Stop characters per telegram format
- the second Start character, the parity of the two field lengths, and the number of additional characters received (= L Field + 6) with a long or control frame

When a fault has been detected as a result of the above checks, the transmission will not be accepted, and the reply or acknowledgement will not be sent.
After a time limit of (330 bit periods + 50 ms) the master interprets the lack of a reply as a fault and repeats the same telegram up to two times. If a valid telegram has not been received at that time a so called "idle time" of at least 33 bit periods is introduced. When slaves send faulty or corrupt replies, three attempts are also made, and if there is a fault during the last attempt then the 33 bit periods "idle time" is introduced.
The master may try a SND_NKE. If this fails also it will continue with the next slave address.

5.5 FCB- and FCV-Bits and Addressing

(♣ whole chapter reworked)

The FCB (Frame Count-Bit) in the REQ_UD2 can be considered as the LSB of a telegram counter of transmitted telegrams in the slave to master direction. On the other hand, the FCB in the SND_UD can be considered as the LSB of a (separate) telegram counter for the transmitted telegrams in the master to slave direction. A set FCV (Frame Count Valid)-Bit signals whether this frame count mechanism is active.

5.5.1 Applications of the FCB-mechanism

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 (see master to slave section).

2.) Frozen answer telegrams from slave to master

In same 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. In meter applications this frozen-telegram aproach will result in possibly very old meter data if the last REQ_UD2 with toggled FCB-bit occured a long time ago. Thus for meter readout this frozen telegram technique is not recommended.

3.) Multi-telegram data (SND_UD) from master to slave

If the master sends a large (sequential) data block to a slave (e.g. RAM/EEPROM-initialize, code upload) which must be divided into multiple telegrams a similar situation like in the slave to master direction might occur. If the slave receives a telegram correctly and answers with a positive acknowledge (usually by a $E5 single byte answer) but the master does not receive this positive answer correctly, the master will repeat the last telegram with the identical FCB-Bit as in the first attempt. From this the slave can recognize that this next telegram does not contain the next data block but repeats the last data block which has been received correctly. So the slave may either ignore this telegram repetition completely or may accept it thus overwriting the last telegrams data with the second identical data. In both cases an internal telegram sequence counter is not incremented. Note that a slave which will accept only single telegram master to slave communication may simply ignore the FCB in the SND_UD. Note also that a master which can accept exactly two (sequential) SND_UD-telegrams may simply use the FCB of the SND_UD to decide which of both telegrams has been sent. Thus a slave which can accept one or two (sequential) SND_UD answer-telegrams does not require an internal "Last-SND_UD-FCB"-image bit. A slave which can accept three or more (sequential) SND_UD telegrams requires such an internal memory bit. Note that such an internal memory bit for the SND_UD-direction must be independent of an possible additional internal memory bit for the RSP_UD direction.

4.) Incremental actions in slave initiated by master

If single telegram SND_UD will initiate some incremental action in a slave (like toggling a relais or counting something) in contrast to sending some "absolute" data or parameters the FCB-mechanism allows as in the multi-telegram SND_UD situation a distinction between a repetition of the last telegram due to missed acklowledge reception and the next action. In this case the action is only taken if the FCB of the current SND_UD-telegram is different from the FCB in the previous SND_UD-telegram.

5.5.2 Implementation aspects for primary addressing

1.) Master

The master must always contain a "Next REQ_UD2-FCB-image bit" and also a "Next SND_UD-FCB image bit" for each primary slave address used by its application layer. After sending a SND_NKE-request to a slave adress both themust_U rg/F0 18d (-) Tj3.96 0.00 Td102FCB

required to send a SND_NKE to all affected addresses. All subsequent RSP_UD2-telegrams must contain the "Next REQ_UD2- FCB-image bit" of the appropriate primary address as a FCB. This "Next REQ_UD2 FCB-image bit" is toggeled after an error free link layer RSP_UD telegram has been received. All subsequent SND_UD-telegrams must contain the "Next SND_UD- FCB-image bit of the appropriate primary address as a FCB. If a SND_UD has been successfully transmitted to a slave (reception of a valid acknowledge byte $E5 or a valid RSP_ACK telegram) this "Next SND_UD-FCB-image bit" associated with this address is toggled.

2.) Slave

If a slave wants to utilize the FCB-Bit mechanism for the REQ_UD2-type (slave to master) transfers for more than two sequential telegrams it must provide a "Last REQ_UD2-FCB"-memory bit. If a valid REQ_UD2 telegram with a set FCV-Bit is received its FCB-Bit is compared to this "Last REQ_UD2-FCB-Bit". If they differ or the FCV-bit is clear, the next actual telegram data are used for the RSP_UD answer otherwise the last (stored) telegram is repeated.
If a slave wants to utilize the FCB-Bit mechanism for the SND_UD-type (master to slave) transfers for more than two sequential telegrams it must provide a "Last SND_UD-FCB"-memory bit. If a valid SND_UD telegram with a set FCV-Bit is received, its FCB-Bit is compared to this "Last SND_UD-FCB-memory Bit". If they differ or the FCV-bit is clear, the next actual telegram data are used for the RSP_UD answer otherwise the last (stored) telegram is repeated.
Note that after a valid reception of a SND_NKE to the primary address of the device or to the test adress 254 ($FE) or the broadcast adress 255 ($255) these internal "Last FCB" memory bits must be cleared.

3.) Implementation for multiple address slaves

A slave might be configured to respond to more than one primary address. This could be useful for slaves which internally consist of more than one independent functional blocks. If this slave wants to utilize FCB-funcionalities they must implement the appropriate number of internal memory bits (0, 1 or 2) for each of these adresses.

4.) Implementation for the primary (broadcast) address 255

All transfers to the primary broadcast address 255 ($FF) are not answered and should hence be implemented by the master with the FCV-Bit cleared. Note that a SND_NKE to primary address 255 will clear the internal "Last received FCB"-Bits of all slaves with primary addresses 0-250 and with FCB-Bit implementation simultaneously.

5.) Implementation for the primary (test) address 254 ($FE)

A slave should answer to all requests to the primary address 254 ($FE=test address) irrespective of its own primary address. The answer must contain its own primary address and not the address 254 ($FE). This test address is used by readout- or test equipment in point-to-point mode. Although this is a second primary address for each slave separate "Last received FCB"- Bit(s) are not required for this special case, since any test equipment or master is required to issue a SND_NKE after each reconnect or power fail thus clearing the "Last received FCB"-Bit(s) and thus preparing for a virgin transaction irrespective of the previous communication history.

6.) Implementation for secondary addressing

For the usage of the FCB-Bit in secondary addressing see chapter 7.2.

6 Application Layer

The standardized application protocol in the standard EN1434-3 for data exchange with heat meters will be the basis for the following discussion. This standard is also suitable for other consumer utility meters, e.g. for gas and water. However, EN1434-3 only covers the data structure in the reply direction, the data structure generally used in the direction master to slave will be presented here.

6.1 CI-Field

The CI-Field codes the type and sequence of application data to be transmitted in this frame.
The EN1434-3 defines two possible data sequences in multibyte records. The 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 multibyte data structures. If the Mode bit is not set (Mode 1), the least significant byte of a multibyte record is transmitted first, otherwise (Mode 2) the most significant byte. The Usergroup recommends to use only the Mode 1 in future applications.

Mode 1	Mode 2	Application	Definition in
51h	55h	data send	EN1434-3
52h	56h	selection of slaves	Usergroup July ´93
50h		application reset	Usergroup March ´94
54h		synronize action	suggestion
B8h		set baudrate to 300 baud	Usergroup July ´93
B9h		set baudrate to 600 baud	Usergroup July ´93
BAh		set baudrate to 1200 baud	Usergroup July ´93
BBh		set baudrate to 2400 baud	Usergroup July ´93
BCh		set baudrate to 4800 baud	Usergroup July ´93
BDh		set baudrate to 9600 baud	Usergroup July ´93
BEh		set baudrate to 19200 baud	suggestion
BFh		set baudrate to 38400 baud	suggestion
B1h		request readout of complete RAM content	Techem suggestion
B2h		send user data (not standardized RAM write)	Techem suggestion
B3h		initialize test calibration mode	Usergroup July ´93
B4h		EEPROM read	Techem suggestion
B6h		start software test	Techem suggestion
90h to 97h		codes used for hashing	obsolete and no longer recommended

Table 2 *CI-Field codes used by the master*

Application reset (CI = $50)

With the CI-Code $50 the master 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. This application reset by a SND_UD with CI=$50 is the counterpart to the reset of the data link layer by a SND_NKE.

Application reset subcode ♣

It is allowed to use optional parameters after CI = $50. The first parameter (the application reset subcode) defines which telegram function and which subtelegram is requested by the master. The datatype of this parameter is 8 bit binary. The upper 4 bits define the telegram type or telegram application and the lower 4 bits define the number of the subtelegram. The use of the value zero for the number of the subtelegram means that all telegrams are requested.
Slaves with only one type of telegram may ignore application reset and the added parameters but have to confirm it ($E5).

The following codes can be used for the upper 4 bits of the first parameter:

Coding	Description	Examples
0000b	All	
0001b	User data	consumption
0010b	Simple billing	actual and fixed date values+dates
0011b	Enhanced billing	historic values
0100b	Multi tariff billing	
0101b	Instaneous values	for regulation
0110b	Load management values for management	
0111b	Reserved	
1000b	Installation and startup	bus adress, fixed dates
1001b	Testing	high resolution values
1010b	Calibration	
1011b	Manufacturing	
1100b	Development	
1101b	Selftest	
1110b	Reserved	
1111b	Reserved	

Table 3 *Coding of the upper four bits of the first parameter after CI = $50*

Example:
The master releases an enhanced application reset to all slaves. All telegrams of the user data type are requested.

Master to Slave: 68 04 04 68 | 53 FE 50 | 10 | B1 16
Slave to Master: E5

Syncronize action (CI = $54) ♣

This CI-code can be used for syncronizing functions in slaves and masters (e.g. clock syncronization).

The use of the other control information codes is described in the chapters 6.4.1 (set baudrate to 300 .. 38400 Bd), 6.4.2 (data send) and 7 (selection of slaves).

The following codes can be used for the direction slave to master:

CI M=0	CI M=1	Application	Defined in
70h		report of general application errors	Usergroup March ´94
71h		report of alarm status	Usergroup March ´94
72h	76h	variable data respond	EN1434-3
73h	77h	fixed data respond	EN1434-3

Table 4 *CI-Field codes used by the slave*

The use of these control information codes is described in the chapters 6.1 (fixed data respond), 6.2 (variable data respond), 6.6 (report of general application errors) and 8.1 (report of alarm status).

6.2 Fixed Data Structure

In the reply direction with a long frame two different data structures are used. 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 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 reasons the fixed data structure is not recommended for future developments.

To identify the fixed data structure, the numbers 73h/77h for the control information field are used. In this way the master software can see how it must interpret the data.

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

Fig. 15 *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 Figure 16:

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

Fig. 16 *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. The units of counter 1 are coded with the first 6 bits of the first byte, and the units of counter 2 with the first 6 bits of the second byte. The coding of the medium is made up of the two highest bits of these bytes, and can therefore have 16 different values (4 bits). Tables to represent the physical units and the coding of the medium are in the appendix.

Byte	**Byte No. 8 (byte 2 of medium/unit)**								**Byte No. 7 (byte 1 of medium/unit)**							
Bit	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
	Medium		**physical unit of counter 2**						**Medium**		**physical unit of counter 1**					
	MSB		**MSB**					**LSB**		**LSB**	**MSB**					**LSB**

Fig. 17 *Coding of physical unit and medium in fixed data structure (data type E)*

To allow transmission of one historic value with one of the two counters the special unit (111110b or hex code share of 3Eh) has been defined. This unit declares that this historic counter has the same unit as the other actual counter.

Example for a RSP_UD with fixed data structure (mode 1):

The slave with address 5 and identification number 12345678 responds with the following data (all values hex.):

68 13 13 68	header of RSP_UD telegram (L-Field = 13h = 19d)
08 05 73	C field = 08h (RSP_UD), address 5, CI field = 73h (fixed, LSByte first)
78 56 34 12	identification number = 12345678
0A	transmission counter = 0Ah = 10d
00	status 00h: counters coded BCD, actual values, no errors
E9 7E	Type&Unit: medium water, unit1 = 1l, unit2 = 1l (same, but historic)
01 00 00 00	counter 1 = 1l (actual value)
35 01 00 00	counter 2 = 135 l (historic value)
3C 16	checksum and stop sign

6.3 Variable Data Structure

The CI-Field codes 72h/76h are used to indicate the variable data structure in long frames (RSP_UD). Figure 18 shows the way this data is represented:

Fixed Data Header	Variable Data Blocks (Records)	MDH	Mfg.specific data
12 Byte	variable number	1 Byte	variable number

Fig. 18 *Variable Data Structure in Reply Direction*

6.3.1 Fixed Data Header

The first twelve bytes of the user data consist of a block with a fixed length and structure (see fig. 19).

Ident. Nr.	Manufr.	Version	Medium	Access No.	Status	Signature
4 Byte	2 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Byte

Fig. 19 *Fixed Data Block*

♣ In contrast to the fixed data structure here the **Identification Number** is a customer number, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00000000 to 99999999. It can be preset at fabrication time with a unique number, but could be changeable afterwards, especially if in addition an unique and not changeable fabrication number (DIF = $0C, VIF = $78, see chapter 6.7.3) is provided.
The **access number** is described above in the fixed data structure (see chapter 6.2).
The field **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:

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

The field **version** specifies the generation or version of this counter and depends on the manufacturer. In contrast to the fixed data structure, the **Medium** is coded with a whole byte instead of four bits and the lowest two bits of the **Status** field are used to indicate application

errors (see chapter 6.6). Apart from this, the significance of the individual bits of the Status field is the same as that of the fixed data structure. The **Signature** remains reserved for future encryptation applications, and until then is allocated the value 00 00 h.

6.3.2 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 Information Block

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

Fig. 20 *Structure of a Data Record (transmitted from left to right)*

Each data record contains one value with its description as shown in figure 20, a data record, which 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.

The DIB contains at least 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			

Fig. 21 *Coding of the Data Information Field (DIF)*

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

Code	Description	Code	Description
00b	Instantaneous value	01b	Maximum value
10b	Minimum value	11b	Value during error state

The **data field** shows how the data from the master must be interpreted in respect of length

Like all multibyte fields in mode 1 the last character and in mode 2 the first character is transmitted first.

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=$0F or $1F 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 22 which follows shows the structure of a DIFE:

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

Fig. 22 *Coding of the Data Information Field Extension (DIFE)*

With the maximum of ten DIFE´s which 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.

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)						

Fig. 23 *Coding of the Value Information Field (VIF)*

There are five types of coding depending on the VIF:

1. **Primary VIF: E000 0000b .. E111 1011b**
   The unit and multiplier is taken from the table for primary VIF (chapter 8.4.3).
2. **Plain-text VIF: E111 1100b**
   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 and FBh**
   In case of VIF = FDh and VIF = FBh the true VIF is given by the next byte and the coding is taken from the table for secondary VIF (chapter 8.4.4). This extends the available VIF´s by another 256 codes.
4. **Any VIF: 7Eh / FEh**
   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, chapter 6.5), for reports of application errors (slave to master, chapter 6.6) and for an enhancement of the VIF (orthogonal VIF, chapter 8.4.5). 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.

6.3.3 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.

Example for a RSP_UD with variable data structure answer (mode 1):
(all values are hex.)

68 1F 1F 68	header of RSP_UD telegram (length 1Fh=31d bytes)
08 02 72	C field = 08 (RSP), address 2, CI field 72H (var.,LSByte first)
78 56 34 12	identification number = 12345678
24 40 01 07	manufacturer ID = 4024h (PAD in EN 61107), generation 1, water
55 00 00 00	TC = 55h = 85d, Status = 00h, Signature = 0000h
03 13 15 31 00	**Data block 1:** unit 0, storage No 0, no tariff, instantaneous volume, 12565 l (24 bit integer)
DA 02 3B 13 01	**Data block 2:** unit 0, storage No 5, no tariff, maximum volume flow, 113 l/h (4 digit BCD)
8B 60 04 37 18 02	**Data block 3:** unit 1, storage No 0, tariff 2, instantaneous energy,

	218,37 kWh (6 digit BCD)
18 16	checksum and stopsign

6.4 Configuring Slaves

The means for configuring slaves, for example set primary address or secondary address, set baudrate or set other configuration data inside the slave are explained in this section.

6.4.1 Switching Baudrate

All slaves must be able to communicate with the master using the minimum transmission speed of 300 baud. Splitted baudrates between transmit and receive are not allowed, but there can be devices with different baudrates on the bus.
In point to point connections the slave is set to another baudrate by a Control Frame (SND_UD with L-Field = 3) with address FEh and one of the following CI-Field codes:

CI-Field	B8h	B9h	BAh	BBh	BCh	BDh	BEh	BFh
Baud	300	600	1200	2400	4800	9600	19200	38400
Note		1	1		1		1,2	2

Fig. 24 *CI-Field-Codes for Baudrate Switching*

Notes:
1) These baudrates are not recommended.
2) These baudrates will be available in future with new repeater hardware. CI-Field codes are suggestions by the Usergroup.

The slave confirms the correctly received telegram by transmitting an E5h with the old baudrate and uses the new baudrate from now on, if he is capable of this.
The master must know the highest available baudrate on the bus to forbid the user switching to a transmission speed, which is not available on the bus. Otherwise the slave would never answer again.

Example:
The master switches the slave (in point to point connection) from now 2400 baud to 9600 baud.

Master to slave:	68 03 03 68 | 53 FE BD | 0E 16	with 2400 baud
Slave to master:	E5	with 2400 baud

From that time on the slave communicates with the transmission speed 9600 baud.

6.4.2 Writing Data to a Slave

The master can send data to a slave using a SND_UD with CI-Field 51h for mode 1 or 55h for mode 2. Note that the data structure in such a write telegram has been changed in contrast to previous definitions by means of leaving out the fixed data header of 12 byte. The following figure shows the data structure for a write telegram. The order of the first three blocks in the following figure can be turned round, but the write only data record must be at the end of the telegram. All records are optional.

Primary Address Record	Enhanced Identification Record	Normal Data Records	Write Only Data Records

Fig. 25 *Data Structure for Writing Data*

- Primary Address Record:
  The primary address record is optional and consists of three bytes:

DIF = 01h	VIF = 7Ah	Data = Address (1 byte binary)

  With this data record a primary address can be assigned to a slave in point to point connections. The master must know all the used addresses on the bus and forbid setting the address of a slave to an already used address. Otherwise both slaves with the same address couldn´t be read out anymore.

- Enhanced Identification Record:
  With this optional data record the identification (secondary address) can be changed. There are two cases to be distinguished:

  1) Data is only the identification number

DIF = 0Ch	VIF = 79h	Data = Identification No. (8 digit BCD)

  2) Data is the complete identification

DIF = 07h	VIF = 79h	Data = complete ID (64 bit integer)

  The data is packed exactly as in the readout header of a $72/$76 variable protocol with low byte first for mode 1 and high byte first for mode 2:

Identification No.	Manufacturer ID	Generation	Medium
4 byte	2 byte	1 byte	1 byte

- Normal Data Records:
  The data records, which can be read out with a REQ_UD2, are sent back to the slave with the received DIF and VIF and the new data contents. Additional features can be implemented using the generalized object layer (see chapter 6.5 ♣).
- Write-Only Data:
  Data, which cannot be read out of the slave with a normal data block, can be transmitted using the VIF = 7Fh for manufacturer specific coding. The DIF must have a value corresponding to the type and length of data.

After receiving the SND_UD correctly without any error in data link layer the slave must answer with an acknowledgement (E5h). The slave decides whether to change variables or not after a data write from the master. In case of errors in executing parts of or whole write instructions the slave can decide whether to change no variables or single correct variables. The slave can report the this errors to the master in the next RSP_UD telegram using some of the methods which are described in chapter 6.6.
There are some methods for implementing write protect, for example allowing only one write after a hardware reset of the processor or enabling write if a protect disable jumper is set.

Examples:

1. Set the slave to primary address 8 without changing anything else:
   68 06 06 68 | 53 FE 51 | 01 7A 08 | 25 16

2. Set the complete identification of the slave (ID=01020304, Man=4024h (PAD), Gen=1, Med=4 (Heat):
   68 0D 0D 68 | 53 FE 51 | 07 79 04 03 02 01 24 40 01 04 | 95 16 ♣

3. Set identification number of the slave to "12345678" and the 8 digit BCD-Counter (unit 1 kWh) to 107 kWh.
   68 0F 0F 68 | 53 FE 51| 0C 79 78 56 34 12 | 0C 06 07 01 00 00 | 55 16

6.4.3 Configuring Data Output

For default the slave transmits all his data with a RSP_UD. It could be useful for some applications to read only selected data records out of one or more devices. There are two ways to select data records:

Selection without specified data field

The selection of the wanted data records can be performed with a SND_UD (CI-Field = 51h/55h) and data records containing the data field 1000b, which means "selection for readout request". The following VIF defines the selected data as listed in EN1434-3 and no data are transmitted. The answer data field is determined by the slave. The master can select several variables by sending more data blocks with this data field in the same telegram.

Special multiple values can be selected with the following methods:

- Any VIF:
  The VIF-Code $7E (any VIF) is especially for readout request of "all VIF" from the slave and can be interpreted as a selection wildcard for the value information field.
- Global readout request:
  The DIF-Code $7F is defined as "selection of all data for readout request", i.e. all storage numbers, units, tariffs and functions. If this DIF is the last byte of user data or the VIF=$7E follows, then all data is requested. So the selection of all data of one slave can de done with a SND_UD and the character $7F as the user data. If there follows a DIF unequal to $7E, then all subfields of this VIF are selected for readout.
- All Tariffs:
  The highest tariff number in the selection record is defined as selection of "all tariffs". For example the tariff 1111b (15) means selection of all tariffs in a record with two DIFE´s.
- All Storage Numbers:
  A selection of all storage numbers can be done with the maximum storage number if there is a minimum of one DIFE. For example the highest storage number is $1F (31) with one DIFE and $1FF (511) with two DIFE´s.
- All Units:
  "All units" can be selected by using a data record header with minimum two DIFE´s and the highest unit number.

- High Resolution Readout:
  The master can select the slave to answer with the maximum resolution to a given value / unit by a VIF with "nnn" = 000 (minimum exponent for range coding). The meter may then answer with a resolution of e.g. 1mWh (VIF=0000000b) or some higher decimal value if required. The unit values have been chosen so that their minimum provides sufficient resolution even for calibration. A readout request for a VIF with "nnn"=max (maximum exponent for range coding) signals a request for the standard resolution of the meter.

After the next REQ_UD2 the slave answers with the selected data in his own format, if the requested data are available. Otherwise the slave transmits his normal data and the master has to find out that the data are not the requested one. If there are more than one variables with the selected VIF, the device should send all these data records.

Selection with specified data field

The master is able to perform a readout request with a specified data field by using the object action “add to readout list“ (VIFE = E000 1100b) from VIFE-table for object actions (see chapter 6.5). The master transmits a SND_UD (CI-Field = 51h/55h) with a data record which consists of the desired DIF (data field), VIF and the VIFE = 0Ch / 8Ch. No data follows this VIFE and the slave should ignore the data field on reception. The slave should transmit this data record with the requested data field from now on, if he is capable of this. If the slave doesn´t support this data field (data coding), it can report a record error using one of the VIFE = E000 011x (data class not implemented or data size not implemented).

Deselection of data records

The master can release a reset of the application layer and especially a fallback to the slaves standard RSP_UD-Telegram by transmitting a SND_UD with the CI-Field $50.
Single data records can be deselected by transmitting a data record with DIF, VIF and the VIFE for the object action “Delete from Readout-List“ (VIFE = E000 1101b).

If the selected data is supported by the slave but too long for one RSP_UD telegram (especially for readout of all historic values), the slave transmits an additional data record consisting only of the DIF=$1F, which means that more data records follow in the next respond telegram. In this case the master must readout the slave again until the respond telegram is only an $E5 (no data) or there is no DIF=$1F in the RSP_UD.
To avoid lost of data respond telegrams the slave should in this case support the Frame Count Bit (FCB). If the master wants to premature end such a multitelegram sequential readout of the selected data, it may send an application reset with CI=$50 instead of further REQ_UD2´s.

Examples:

1. A slave with address 7 is to be configured to respond with the data records containing volume (VIF=13h: volume, unit 1l) and flow temperature (VIF=5Ah: flow temp., unit 0.1 °C).
   68 07 07 68 | 53 07 51 | 08 13 08 5A | 28 16

2. A slave with address 1 is to be configured to respond with all storage numbers, all tariffs, and all VIF´s from unit 0.
   68 06 06 68 | 53 01 51 | C8 3F 7E | 2A 16

3. A slave with address 3 is to be configured to respond with all data for a complete readout of all available. After that the master can poll the slave to get the data.
   68 04 04 68 | 53 03 51 | 7F | 26 16

6.5 Generalized Object Layer

The fundamental idea of an object is the encapsulation of data and methods or actions for the data. In case of writing data to a slave the master software can pack data and information about the action, which the slave shall do with this data, in one data record. This variable data record with actions is now called an object. If the VIF has not got the value $FD (extended VIF-Code table) the following VIFE-Codes define the action:

Action: (E: extension bit)

VIFE-Code binary	Action	Explanation
E000 0000	Write (Replace)	replace old with new data
E000 0001	Add Value	add data to old data
E000 0010	Subtract Value	subtract data from old data
E000 0011	OR (Set Bits)	data OR old data
E000 0100	AND	data AND old data
E000 0101	XOR (Toggle Bits)	data XOR old data
E000 0110	AND NOT (Clear Bits)	NOT data AND old data
E000 0111	Clear	set data to zero
E000 1000	Add Entry	create a new data record
E000 1001	Delete Entry	delete an existing data record
E000 1010	Reserved	
E000 1011	Freeze Data	freeze data to storage no.
E000 1100	Add to Readout-List	add data record to RSP_UD
E000 1101	Delete from Readout-List	delete data record from RSP_UD
E000 111x	Reserved	
E001 xxxx	Reserved	

Fig. 26 *Action Codes for the Generalized Object layer (Master to Slave)*

With these actions the master can alter the data of the slaves or configure the output data of the slaves (actions 12 and 13). The actions 0 to 6 alter the data of the slave by replacing the old data (action 0, equals to data write without VIFE) or do arithmetical or logical operations with the old and the transmitted data.
Note that this method of configuring the readout list (action 12 and 13) allows not only the adding but also the removal of elements in contrast to the method of using the DIF=1000b-type of readout request (described in chapter 6.4.3).

All these actions can be used for normal slaves and for intelligent master which are manipulated by a higher order master.
The functions "Add entry" and "Delete entry" are useful to tell an intelligent master to add e.g. a new data record like maximum or minimum values of any slave.

With the action "freeze data to storage #" the master can tell the slave to freeze the actual value corresponding to the transmitted VIF, unit, tariff and function to a certain storage number given in the DIF/DIFE´s. In this case the data field inside the VIF has got the value 0000b (no data). This action allows freeze of selected values or multiple freeze with VIF=$7E (all VIF). The date / time should also be freezed to the same storage number.

Examples:

1) Set the 8 digit BCD-Counter (instantaneous, actual value, no tariff, unit 0) with VIF=06 (1kWh) of the slave with address 1 to 107 kWh.
   68 0A 0A 68 | 53 01 51 | 0C 86 00 07 01 00 00 | 3F 16

2) Same as in example 1) but add 10 kWh to the old data.
   68 0A 0A 68 | 53 01 51 | 0C 86 01 10 00 00 00 | 48 16

3) Add an entry with an 8 digit BCD-Counter (instantaneous, actual value, no tariff, unit 0, 1kWh) with the start value of 511 kWh to the data records of the slave with address 5.
   68 0A 0A 68 | 53 05 51 | 0C 86 08 11 05 00 00 | 59 16

4) Freeze actual flow temperature (0.1 °C: VIF = 5Ah) of the slave with address 1 into the storage number 1.
   68 06 06 68 | 53 01 51 | 40 DA 0B | CA 16

6.6 Application Layer Status

There can be so far only data link layer errors reported from slave to master by means of leaving out the acknowledgement or negative acknowledgement. It is not allowed to report errors in the application layer, which can occur for example in data writing, using any of the above methods, because these are reserved for link layer errors. The slave can transmit an $E5 after a REQ_UD2 to indicate that it has received the telegram, but can´t respond with data. There are three different techniques for reporting application errors:

1. Status Field

One possible solution is to use the reserved 2 lowest bits of the Status field in the variable data structure for the application layer status:

Status bit 1 bit 0	Application status
0 0	No Error
0 1	Application Busy
1 0	Any Application Error
1 1	Reserved

Fig. 27 *Application Errors coded with the Status-Field*

2. General Application Errors

For reporting general application errors a slave can use a RSP_UD telegram with CI=$70 and zero or one byte data, which then describes the type of error:

68h	04h	04h	68h	08h	PAdr	70h	DATA	CS	16h

Fig. 28 *Telegram for reporting general application errors*

The following values for DATA are defined:

0	Unspecified error: also if data field is missing
1	Unimplemented CI-Field
2	Buffer too long, truncated
3	Too many records
4	Premature end of record
5	More than 10 DIFE´s
6	More than 10 VIFE´s
7	Reserved
8	Application too busy for handling readout request
9	Too many readouts (for slaves with limited readouts per time)
10..255	Reserved

Table 6 *Codes for general application errors*

3. Record Errors

To report errors belonging to a special record the slave can use this data record header with a VIFE containing one of the following values to code the type of application error, which has been occured.

VIFE-Code	Type of Record Error	Error Group
E000 0000	None	DIF Errors
E000 0001	Too many DIFE´s	
E000 0010	Storage number not implemented	
E000 0011	Unit number not implemented	
E000 0100	Tariff number not implemented	
E000 0101	Function not implemented	
E000 0110	Data class not implemented	
E000 0111	Data size not implemented	
E000 1000 to E000 1010	Reserved	
E000 1011	Too many VIFE´s	VIF Errors
E000 1100	Illegal VIF-Group	
E000 1101	Illegal VIF-Exponent	
E000 1110	VIF/DIF mismatch	
E000 1111	Unimplemented action	
E001 0000 to E001 0100	Reserved	
E001 0101	No data available (undefined value)	Data Errors
E001 0110	Data overflow	
E001 0111	Data underflow	
E001 1000	Data error	
E001 1001 to E001 1011	Reserved	
E001 1100	Premature end of record	Other Errors
E001 1101 to E001 1111	Reserved	

Table 7 *Codes for record errors (E = extension bit)*

In case of record errors the data maybe invalid. The slave has some options to transmit the data:

- datafield = 0000b: no data
- datafield = 0000b: no data and idle filler (DIF=$2F): fill telegram up to the normal length
- other datafield: dummy data of correct length
- other datafield: unsafe or estimated data

6.7 Special Slave Features

Some optional or recommended features of the slaves, which are not mentioned in EN1434-3, will be described in this section.

6.7.1 Auto Speed Detect

♣ This feature is implemented in several slaves. It is no longer recommended by the M-Bus Usergroup because it is difficult to guarantee a hamming distance of four with this method.

The fact that several baudrates are allowed on the bus causes the problem for the master to know the used baudrates of all connected slaves. The software must perform the search for primary and secondary addresses with all allowed baudrates. For this reasons it is useful, if the slaves would answer with just the baudrate, which the master uses for this telegram. In addition the risk to loose contact with a slave would be dismissed and commands for baudrate switching would no longer be required.

The slave detects the baudrate by scanning the start sign of the telegram, which has the value 10h for a short frame or the value 68h for a long frame.

Startsign 10h on the bus:

Start	LSB	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	MSB	Parity	Stop
0	0	0	0	0	1	0	0	0	1	1

Startsign 68h on the bus:

Start	LSB	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	MSB	Parity	Stop
0	0	0	0	1	0	1	1	0	1	1

The startsign 10h begins with five space bits and the startsign 68h with four space bits on the line. After detecting the beginning of a new telegram the device measures the duration of

space. By comparing the space duration with a stored table of ranges for five and four bits of all implemented transmission speeds the slave makes a decision about the baudrate and the number of space bits (four or five). Then it receives the remaining bits of the startsign (seven or six) with the detected transmission speed. This procedure has to set for example a flag or a variable to tell the normal receive procedure the used baudrate.
Auto speed detect of 300, 2400 and 9600 baud has been implemented with an eigth bit microcontroller and works fine. This feature can be easily programmed in microcontrollers using software polling for communication, but is more difficult with UART-based communication.

6.7.2 Slave Collision Detect

Collisions between transmitting slaves can occur during slave search activities by the master. Very light collisions of (22..33) mA, which are equivalent to 2 or 3 transmitting slaves, are electrically undetectable by master and slave. New master hardware with double current detect can detect light collisons of (20..200) mA and then transmit a break (50 ms space) on the bus. The slave can detect medium collisions of (70..500) mA, if this is a collision between a mark and a space and if the slave supports this feature. Heavy collisions of (90..5000mA) will have the effect of a break down of the bus voltage (power fail in the slave) and possibly a shortcircuit in the master.
To avoid these consequences of (heavy) collisions new master have the feature of double current detect with break signaling and switching off the bus in overcurrent states. There are some means for the slaves to detect collisions and then stop transmitting:

1. Software based UART´s can test at the end of each Mark-Send-Bit whether the input is really a mark. This guarantees a very fast detection of collisions, is simple to implement and is strongly recommended for pure software UART.
2. A variation of the preceeding method is to test whether the bus voltage is mark after each stop-bit. This is simple for a software UART, but very tricky for a hardware UART and requires a master sending a break on collision detect.
3. A simple method for unbuffered hardware UART, but tricky for buffered hardware UART, is to compare the transmitted with the received byte.
4. Another method, which requires a master with break collision detect, is a hardware UART with break detect.
5. The baudrate of the communication process after a detected break will be 300 Baud ♣.

6.7.3 Use of the fabrication Number

(♣ new chapter)

The fabrication number is a serial number allocated during manufacture. It is part of the variable data block (DIF = $0C and VIF = $78) and coded with 8 BCD packed digits (4 Byte).

Example:

68 15 15 68	header of RSP_UD telegram (length 1Fh=31d bytes)
08 02 72	C field = 08 (RSP), address 2, CI field 72H (var.,LSByte first)
78 56 34 12	identification number = 12345678
24 40 01 07	manufacturer ID = 4024h (PAD in EN 61107), generation 1, water
13 00 00 00	TC = 13h = 19d, Status = 00h, Signature = 0000h
0C 78 04 03 02 01	fabrication number = 01020304
9D 16	checksum and stopsign

The use of this number is recommended if the identification number is changeable. In this case two or more slaves can get the same secondary adress and can not be uniquely selected. The fabrication number together with manufacturer, version and medium field build an unique number instaed. Suitable masters use this number for an enhanced selection method if two or more slaves have the same secondary adress (see chapter 7.4).

6.7.4 Hex-Codes $A-$F in BCD-data fields

(♣ new chapter)

General description

1.) Standard Reference

EN1434 allows multi-digit BCD-coded datafields. The current standard does not contain information about what happens if a non-BCD hex code ($A-$F) is detected by the master software.

2.) Purpose of this proposal

a) Define the treatment of non BCD-digits in slave to master RSP_UD-telegrams

To fully define a master software including error treatment such a definition would be desirable.

b) Utilize these codes for simplified error treatment by slave

The current user group proposal contains various techniques for signalling errors or abnormal situations. Most of them are hard to implement on weak mikro-processors. Utilizing these "illegal" codes $A to $F for signalling these states to the master would simplify the software design of the slaves.

c) Anormal states of variables

- Value not available

  This happens for example, if a fixed date value is not yet available, because the first fixed date is in the future. The display at the meter or the remote PC should read "----".

- Device error

  This could happen for a temperature variable, if the sensor is malfunctioning. The display at the meter or a remote PC should signal some error code. Multiple error codes should be supported.

- Soft overflow

  Exceeding the upper count limit on integral values or the upper value limit on momentary values should be signallable. For a wrap around carry of integral variables the display should be consistent with old mechanical wrap around counters. In addition a wrap around flag should be extractable.

- Soft underflow

  Underflowing the lower count limit of 00 on integral values or a negative value on momentary values should be signallable. For a wrap around carry of integral variables the display should be consistent with old mechanical wrap around counters. In addition a wrap around flag should be extractable.

- Simple visible error signalling

  To simplify the design of slaves with integrated displays, the above mentioned non-BCD states of the variables should be both transmittable in the form of suitable (Hex) codes but also be displayable directly from the value codes of a 7-segment (usually LCD) display by extending the normal ten entry BCD to 7-segment decoding table to either a dual 16-entry or a single 32-entry decoding table where 16 entries are used for decoding the MSD (Most Significant Digit) and the other 16 entries are used for the decoding for all other for all other digits. For very weak mikroprocessors with a maximum of a single decoding table with only 16-entries a compatible solution with decreased functionality is also presented.

New proposal

1.) Definition of hex code meanings

a) $A

Such a code in the MSD (Most significant digit) position signals a one digit value overflow either of a number or due to an addition or increment carry. The display at the meter or a remote PC should display a "0" at the appropriate display position. This makes the display compatible with conventional counter rollover. In addition the leading digit can be treated by the slave software simply as a hexadecimal digit instead of the BCD-coded other digits to realize this function. Processing software in the master could convert this data digit to a value of 10 in an extended length data field. In addition an appropriate application error code could be generated if desired. If this hex code appears in any other digit position than the MSD, it signals a value error of the complete data field. If an alternative display decoding table for the digits other than the MSD is possible, this hex code should be displayed in these other digit positions as the symbol "A". This would allow more flexible displayable error codes.

Example: A 4-digit BCD code of "A321" should be interpreted by the master software as "10321" with an optional overrange VIFE-error code and displayed as 0321 on a 4-digit only display.

b) $B

Such a code in the MSD digit position signals a two digit value overflow either of a number or due to an addtion or increment carry. The display at the meter or a remote PC should display a "1" at the appropriate display position. This makes the display compatible with conventional counter rollover. In addition the leading digit can be treated by the slave software simply as a hexadecimal digit instead of the BCD-coded other digits to realize this function. Processing software could convert this data digit to a value of 11 in an extended length data field. In addition an appropriate application error code could be generated if desired. If this hex code appears in any other digit position than the MSD, it signals a value or availability error of the complete data field. It should be displayed as the symbol "-".

Example: A 4-digit BCD code of "B321" should be interpreted by the master software as "11321" with an optional overrange VIFE-error code and displayed as 1321 on a 4-digit only display with digit selective decoding.

c) $C

Such a code in the MSD digit position signals a three digit value overflow either of a number or due to an addition or increment carry. The display at the meter or a remote PC should

display a "2" at the appropriate display position. This makes the display compatible with conventional counter rollover. In addition the leading digit can be treated by the slave software simply as a hexadecimal digit instead of the BCD-coded other digits to realize this function. Processing software could convert this data digit to a value of 12 in an extended length data field. In addition an appropriate application error code could be generated if desired. If this hex code appears in any other digit position than the MSD, it signals a value error of the complete data field. It should be displayed as the symbol "C". Note that the suggested interpretation of $A to $C in the MSD effectivly supports a 30% overrange guard band against an undetected rollover and flexible error codes including the letters "A", "C", "E" and "F".

Example: A 4-digit BCD code of "C321" should be interpreted by the master software as "12321" with an optional overrange VIFE-error code and displayed as 2321 on a 4-digit only display.

d) $D

Such a code in any digit position signals a general error of the complete data field. The display at the meter or a remote PC should display a blank at the appropriate display position. Since both an overflow from $C and an underflow from $E end in this out of range type error the function of an out-of-range over/underflow can be implemented by simple hex arithmetic. It is however recommended that the slave arithmetic checks this $D-code in the MSD before incrementing or decrementing the value for integral variables to make such an error irreversible if the slave does not expect such an over- or underflow.

e) $E

Such a code in the MSD digit position signals a two digit value underflow either of a number or due to a subtraction or decrement borrow. The display at the meter or a remote PC should display an "8" at the appropriate display position. This makes the display compatible with conventional counter rollunder. In addition the leading digit can be treated by the slave software simply as a hexadecimal digit instead of the BCD-coded other digits to realize this function. Processing software could convert the data in the field to a negative value using 16´s complement on the leading digit and tens complement at the other digits. In addition an appropriate application error code could be generated if desired. If this hex code appears in any other digit position than the MSD, it signals a value error of the complete data field. It should be displayed as the symbol "E".

Example: A 4-digit BCD code of "E321" should be interpreted by the master software as "- 1679" (F999-E321+1) with an optional underrange VIFE-error code and displayed as 8321 on a 4-digit only display.

f) $F

Such a code in the MSD digit position signals an one digit value underflow either of a number or due to a subtraction or decrement borrow. The display at the meter or a remote PC should display an "9" at the appropriate display position. This makes the display compatible with conventional counter rollunder. In addition the leading digit can be treated by the slave software simply as a hexadecimal digit instead of the BCD-coded other digits to realize this function. Processing software could convert the data in the field to a negative value using 16´s complement on the leading digit and tens complement at the other digits. In addition an appropriate application error code could be generated if desired. If an alternative display decoding table for the digits other than the MSD is possible, this hex code should be displayed in these other digit positions as the symbol "F". Note that the suggested interpretation of $E and $F in the MSD effectivly supports a 20% underrange guard band against an undetected rollunder for displays and flexible error displays if dual decoding tables are available. In addition it allows a simplified coding for small negative values as often required for values like temperature or flow rate.

Example: A 4-digit BCD code of "F321" should be interpreted by the master software as "-0679" (F999-F321+1) with an optional underrange VIFE-error code and displayed as 9321 on a 4-digit only display.

g) Combinations

If with the exception of the MSD all other digits are true BCD-digits ($0-$9) the value is either considered as "Overflow" for the MSD hex codes $A to $C or as "Underflow" for the MSD hex codes $E and $F or a general error for the MSD hex code $D.
The code $DB..BB in the data field is always considered as "not available". This is displayed as " ----" (with a blank in the MSD). Any other non BCD-hex codes in one or several digits other than the MSD is interpreted as an error for the complete data field. The error type is formed from the characters "A", "C", "E", "F" (all corresponding to their hex code), "-", blank and the digits 0-9. The display may show an identical error code if displaying the variable, but the MSD digit on the display can contain only blank or the digits 0..9.

h) Decoding table for 2*16 entries (or 32 entries)

	0	1	2	3	4	5	6	7	8	9	$A	$B	$C	$D	$E	$F
MSD	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	"0"	"1"	"2"	" "	"8"	"9"

other digits	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	"A"	"-"	"C"	" "	"E"	"F"

i) Subset functions

A slave may utilize either non, a single, several or all suggested special functions and their associated hex codes. A slave might utilize also a different number of hex code functions for different data fields. A slave could also use different display implementations for the various special functions and error displays but the suggested solution would simplify the operation of the system, since the master display will be identical to the slave display for the value associated with the appropriate data field.

2.) Subset for single 16-entry display decoding

a) Decoding table

0	1	2	3	4	5	6	7	8	9	$A	$B	$C	$D	$E	$F
"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	"0"	"-"	"C"	" "	"E"	"9"

b) Overrange

The overrange feature should be limited to a 10% overrange ($A in MSD). A further increment should lead directly to the MSD-error hex code $D, which should stop further increment and decrement and generate a suitable error code in the other digits

c) Underrange

The underrange feature should be limited to a 10% underrange ($F in MSD). A further decrement should lead directly to the MSD-error hex code $D, which should stop further increment and decrement and generate a suitable error code in the other digits.

d) Error codes

Error codes may contain only the letters "C" and "E", blank, "-" and the digits 0-9.

e) Compatiblity

At the master side this subset realisation is completely compatible and transparent to the full implementation.

7 Network Layer

The network layer takes care of choosing the best transmission route between the communication parties in a network. We define that the network layer in the M-Bus protocol "connects" a slave with a certain secondary address to the bus and and associates it with the primary address of 253 ($FD). So the number of 250 addresses (primary) is extended by the network layer associating a selected slave to this address 253. The network layer is only enabled by a SND_UD with CI_Field $52 / $56 to address 253.

7.1 Selection and Secondary Addressing

When addressing in the data link layer with the help of the A-Field, the problem of the address allocation arises. The addresses are normally set to a value of 0 by the manufacturer of the meters, in order to designate them as unconfigured slaves. A very laborious method of address allocation consists of setting the addresses when installing the slaves, for example with DIP switches. A further method of address allocation is to determine the bus addresses when connecting the equipments to the bus with the master software. This sends a command for address allocation (see 6.4.2) to the address 0. In this case the slaves must however all be successively connected to the bus, which very much gets in the way of a simple installation procedure.
When however addressing in the network layer these disadvantages are avoided and the address region is essentially extended beyond the number of 250 with primary addressing (A-Field). The addressing of the slaves takes place with secondary addressing with the help of the following so-called selection:

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

Fig. 29 *Structure of a telegram for selecting a slave (mode 1)*

The master sends a SND_UD with the control information 52h (Mode 1) or 56h (Mode 2) 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 / 56h 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 at the following commands (e.g. REQ_UD) with the bus address FDh and in this example will reply with RSP_UD. In other words the network layer has associated this slave with the address 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 (as described above) with non-matching secondary addresses, or a SND_NKE to address 253. The slave, which 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.

A Slave with implemented primary and secondary addressing should also answer telegrams to his primary address. A Slave with only secondary addressing should occupy the address field in the RSP_UD telegram with $FD to signal that it will not participate in primary addressing.

7.2 FCB-Bit and Selection

(♣ chapter 7.2 reworked)

FCB-Implementation slave

A slave with implemented secondary addressing and with implemented FCB-administration must have an additional set of 0, 1 or 2 separate "Last Received FCB"-memery Bit(s) for all communication via the pseudo primary address 253 ($FD). If it can communicate also alternatively over some other primary address (exept the special addresses 254 and 255) an additional set of 0, 1 or 2 "Last received FCB"-memry bit(s) for each of these primary addresses is required. A valid selection telegram will not only set the internal selection bit but will also clear all 0, 1 or 2 internal "Last received FCB"-memory bit(s) associated with secondary addressing via the pseudo primary address 253 ($FD). The master will start the communication (REQ_UD2 or SND_UD) after any selection telegram (CI=$52 or $56) with the FCV-Bit set and the FCB-Bit set. If a slave has more than one alternative secondary identification, only a single set of 0, 1 or 2 "Last received FCB"-memory bit(s) for all secondary addresses is required.

FCB-Implementation master

The master must implement a separate pair of "Next FCB image"-Bits for pseudo primary address 253 ($FD) as for each other primary address. Although these "Next FCB image"-bits might be used for many slaves, no confusion exists, since for accessing another slave a

selection telegram is required which will define the future FCB sequence both for slave and master.

7.3 Searching for Installed Slaves

Primary Addresses

To read out all installed slaves the master software must know all the slaves, which are connected to the bus. Therefore the software searches for slaves with primary adressing by sending a REQ_UD2 to all allowed adresses (1..250) with all available baudrates. The master notes used primary addresses with the respective baudrates.

Secondary Addresses

The secondary addressing described in the preceding section draws attention to the problem of determining the secondary addresses of slaves connected to the bus. The master can after this read out the slaves making use of secondary addresses with previous selection. Testing all possible identification numbers with the master software would take years, since the identification number offers millions of combinations. For this reason, a procedure was developed for the rapid and automatic determination of already installed slaves:

Wildcard searching procedure

The following wildcard searching procedure uses the occupation of individual parts of the secondary address with wildcards (Fh) for selection:
In this case with the identification number (BCD) each individual position, and by manufacturer, version and medium (binary coding), only one complete byte, can be occupied with wildcards. The master begins the selection using a SND_UD with the control information 52h (Mode 1), and occupies all positions in the identification number, except the top one, with wildcards. The top position is run through in ten selections from 0 to 9 (0FFFFFFF to 9FFFFFFF).
If after such a selection the master receives no acknowledgement, it then goes to the next selection. If the master receives an E5h, it then sends a REQ_UD2 and learns the secondary address of the slaves from the reply telegram, as long as no collision occurs. If there is a collision after the selection or the REQ_UD2, the master varies the next positions and holds the existing one. If there is a collision, for example at 5FFFFFFF, the selection is run through from 50FFFFFF to 59FFFFFF. If in this case collisions again occur, then a change is made to a variation of the next position. After running through a complete position, the next higher position is processed up to 9.

With this Wildcard searching procedure, it will be seen that at least the top position must be run through in order to reach all slaves. Running through further positions may be necessary, depending on the number of the slaves and the distribution of the identification numbers. This procedure allows a statement of the maximum number of selections in relation to the number of slaves, but as disadvantage frequent collisions, which occur, should be mentioned. The wildcard searching procedure must be performed for all used baudrates and both byte sequences (mode 1 and 2).
The search procedure can be extended with searching for manufacturer, generation and finally media to find slaves, which have the same identification number. It is also possible to search for all slaves of a certain manufacturer or all slaves of a certain media by setting the corresponding value.

The company Aquametro AG has simulated such a search to find the minimum, the average and the maximum number of selections as a function of the number of slaves. For the minimum number of attempts the optimum distribution of the identification numbers was chosen, for the maximum number the most unfavourable, and for the average number of attempts a random distribution. The following diagram shows the result of these calculations:

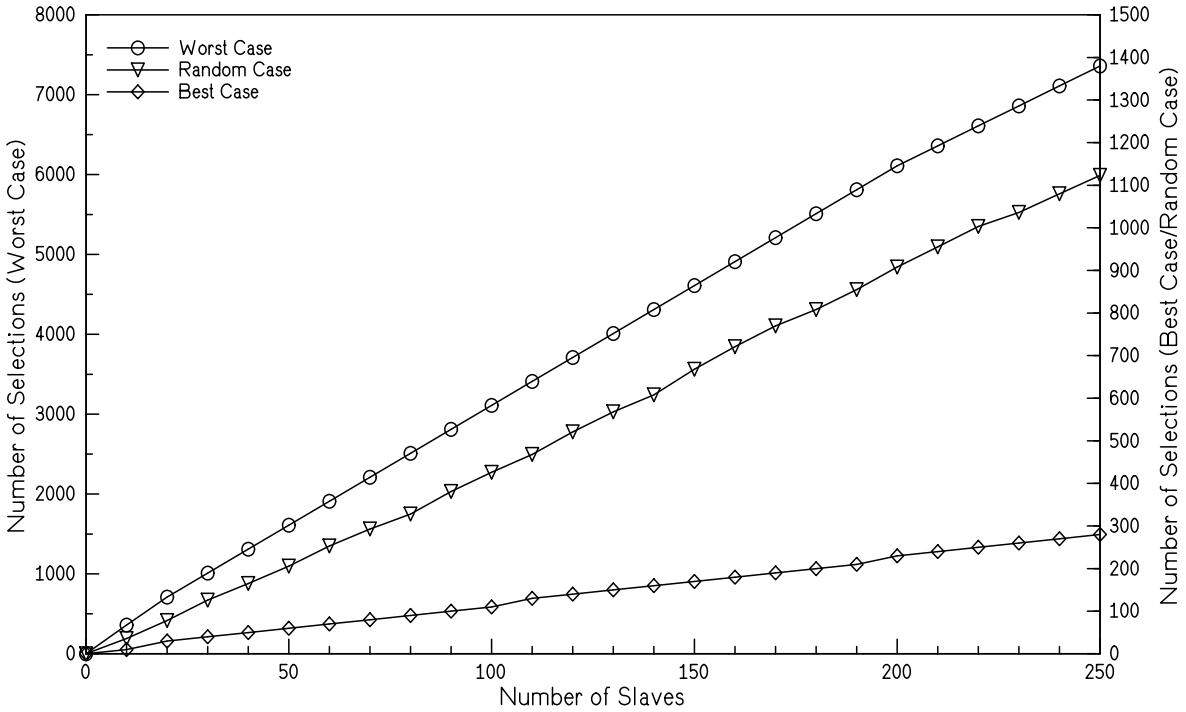

Fig. 30 *Number of Selections with Wildcard Searching Procedure*

Instructions for implementation of Wildcard Search

The following program flow diagram shows the realization of the Wildcard searching procedure, whereby the search is made only with the identification number. The codes for manufacturer, version and medium are in general specified with wildcards, but can be changed by the user in order (for example) to locate all meters from a particular manufacturer. In order to avoid the categorisation by a factor of eight of the "For-To" loops for the eight positions, the array "Value" is defined with 8 byte numbers, which are intended to define the contents of the positions. The digit number of the identification number which is presently running is noted in the variable "Pos" of type byte.

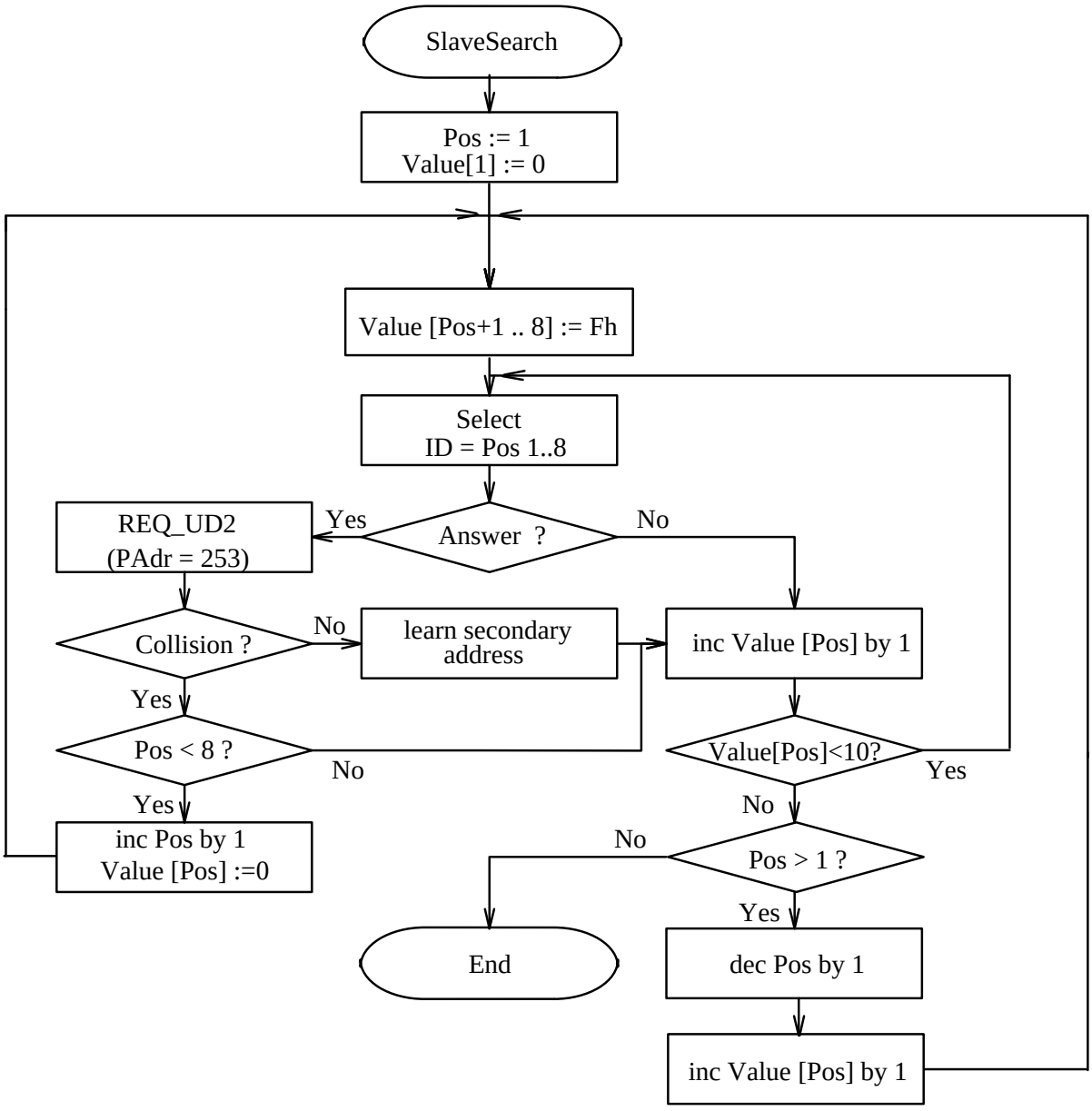

Fig. 31 *Flow Diagram for Slave Search with Wildcards*

The routine begins at the first position, and implements the following actions for the value of this position from 0 to 9:

- Selection with the ID-Nr. Pos 1, Pos 2, ..., Pos 8
- if no reply, Value [Pos] is raised by 1
- if there is a reply, a REQ_UD2 is sent to address 253, and if the telegram is correctly received the secondary address is learnt and the Value [Pos] raised by 1
- if there is a collision a jump is made to the next position (Pos increased by 1), as long as the last position has not yet been reached
- after going through a complete position from 0 to 9 the subroutine proceeds to the next lower position, or ends the search if the position Nr. 1 has already been processed

Example

The next figure shows an example for secondary addresses in order from top to bottom, as they will be found by the master software:

No.	Identification-Nr.	Manufacturer. (hex.)	Version (hex.)	Media (hex.)
1	14491001	1057	01	06
2	14491008	4567	01	06
3	32104833	2010	01	02
4	76543210	2010	01	03

Fig. 32 *Secondary Addresses found with a Wildcard Search of Four Slaves*

Search Process:

1. Start with ID = 0FFFFFFF : no reply
2. ID = 1FFFFFFF : collision between Nr.1 and Nr.2
3. ID = 10FFFFFF, 11FFFFFF, 12FFFFFF, 13FFFFFF : no reply
4. ID = 14FFFFFF : collision between Nr.1 and Nr.2
5. Repeated steps 3 to 4 up to the ID = 1449100F
6. Learn ID = 14491001 and 14491008
7. Go backwards to 19999999
8. ID = 2FFFFFFF : no reply
9. ID = 3FFFFFFF : learn ID = 32104833
10. ID = 4FFFFFFF, 5FFFFFFF, 6FFFFFFF : no reply
11. ID = 7FFFFFFF : learn ID = 76543210
12. ID = 8FFFFFFF, 9FFFFFFF : no reply
13. End of the Search

7.4 Generalized Selection Procedure

For including new or restructed identification parameters into a selection procedure an enhanced definition of the selection telegram (CI=$52/$56) is suggested:

After the 8 byte of the fixed selection header may also follow standard records with data. In this case only those meters will be selected, where in addition to the fixed header all record data agree. In most but not all cases this means that the DIF and parts of the VIF (not exponent) must match. Again wildcard rules apply to the record data (digit wildcard for BCD-coded data and byte wildcard for binary or string data).
With some new defined VIF-codes (extension) and this generalized selection it will be possible to select slaves using e.g. longer identification numbers, customer, customer location and more information. See chapter 8.4.4 for this new VIF-Codes. Two useful examples from the primary table for VIF´s are the “Fabrication No.“ and “Bus Address“.

Enhanced selection with fabrication number ♣

The identification number is a customer number and can be changed by the master. Therefore it can be possible that two slaves have the same secondary adress. For this reason the selection telegram can be extended by a **fabrication number** to devide such slaves. This number is a serial number allocated during manufacture, coded with 8 BCD packed digits (4 Byte) like the identification number, and thus runs from 00000000 to 99999999.
The following figure shows the structure of an enhanced selection telegram released by the master.

68h	11h	11h	68h	53h	FDh	52h	ID1-4	Man1-2	Gen	Med	0Ch	78h	Fab1-4	CS	16h

Fig. 33 *Structure of a telegram for enhanced selection (mode 1)*

After the field medium the new data is given in form of a structured datarecord with DIF=0Ch and VIF=78h. Parts of the fabrication number (Fab1..Fab4) can be occupied with wildcards (Fh).
If a fabrication number exists the slave should add this data to the variable data blocks in every RSP-UD telegram. If the fabrication number and enhanced selection is not implemented in a slave this device will not confirm the enhanced selection telegram and will be deselected.
Enhanced selection should be used only if the normal kind of selection is not successful.

8 Appendix

8.1 Alarm Protocol

The formerly described method for an alarm protocol (see diploma work of Andreas Steffens "Eigenschaften und Anwendungen des M-Bus") was based on time slices for each of the maximum 64 alarm devices. This alarm protocol has not been standardized.
We now suggest to return to standard alarm protocol which conforms to the standard IEC 870-2:
The master software polls the maximum 250 alarm devices by requesting time critical data (REQ_UD1 to adresses 1 .. 250). A slave can transmit either a single character acknowledgement E5h signalling no alarm or a RSP_UD with the CI-Field 71h to report an alarm state.

68h	04h	04h	68h	08h	Adr	71h	Alarm State	CS	16h

Fig. 34 *Telegram for an Alarm-Respond*

The alarm state is coded with data type D (boolean, in this case 8 bit). Set bits signal alarm bits or alarm codes. The meaning of these bits is manufacturer specific.

The timeout for time critical communication must be set to 11..33 bit periods to ensure a fast poll of all alarm devices. With a baudrate of 9600 Bd and all 250 slaves reporting an alarm just in time before a timeout occurs each slave will be polled in periods of maximum 5.5 seconds. This seems to be fast enough for alarms in building control systems and other applications. For faster alarm systems the number of alarm sensors could be limited to 63 (reducing the worst case overall signal delay to less than 1.5 sec or increase the transmission speed to 38400 Bd (with the new repeater hardware) and achieve the same speed for up to 250 devices.
The functionality of the FCB- and FCV-Bit should be fully implemented in this alarm protocol to ensure that one-time alarms are safely transmitted to the master. If the slave has reported an one-time alarm and the next REQ_UD1 has a toggled FCB (with FCV=1) the slave will answer with an E5h signalling no alarm. Otherwise it will repeat the last alarm frame to avoid that the alarm message gets lost.

This new alarm protocol has the advantages of being standardized in IEC 870-2, simple implementation in slaves and master, fast poll cycles and using almost (with the exception of shorter timeout) the normal protocol. In addition it is allowed under EN1434-3 which allows all other types of communication of IEC-870-5-2.

8.2 Coding of Data Records

The standard IEC 870-5-4 defines the following data types for usage inside the application layer:

Type A = Unsigned Integer BCD := XUI4 [1 to 4] <0 to 9 BCD>

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
digit 10				digit 1			
8	4	2	1	8	4	2	1
...	...	...	...	...	...	...	...
8	4	2	1	8	4	2	1

1UI4 [1 to 4] <0 to 9 BCD> := digit 10^0
2UI4 [5 to 8] <0 to 9 BCD> := digit 10^1
...
XUI4 [5 to 8] <0 to 9 BCD> := digit 10^{X-1}

Type B = Binary Integer := I[1..X] <-2^{X-1} to $+2^{X-1}-1$>

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...							...
S	2^{X-2}						2^{X-8}

1B1 [X] := S=Sign: S<0> := positive
S<1> := negative
negative values in two´s complement

Type C = Unsigned Integer := UI[1 to X] <0 to 2^X-1>

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...							...
2^{X-1}							2^{X-8}

UI8 [1 to 8] <0 to 255>

Type D = Boolean (1 bit binary information) := XB1 B1[i] <0 to 1>

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...							...
2^{X-1}							2^{X-8}

XB1: B1[i] <0 to 1>
B1[i] <0> := false
B1[i] <1> := true

Type E = Compound CP16 (types and units information)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8

1UI6[1 to 6] <0 to 63> := physical unit 1
1UI6[9 to 14] <0 to 63> := physical unit 2
1UI4[7,8,15,16] <0 to 15> := measured media

The following data types can only be used with the variable data structure:

Type F = Compound CP32: Date and Time

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}

min: UI6 [1 to 6] <0 to 59>
hour: UI5 [9 to13] <0 to 23>
day: UI5 [17 to 21] <1 to 31>
month: UI4 [25 to 28] <1 to 12>
year: UI7[22 to 24,29 to 32] <0 to 99>
IV: B1[8] {time invalid}: IV<0> := valid,
IV>1> := invalid
SU: B1[16] {summer time}:
SU<0> := standard time,
SU<1> := summer time
RES1: B1[7] {reserved}: <0>
RES2: B1[14] {reserved}: <0>
RES3: B1[15] {reserved}: <0>

Type G: Compound CP16: Date

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8

day: UI5 [1 to 5] <1 to 31>
month: UI4 [9 to 12] <1 to 12>
year: UI7[6 to 8,13 to 16] <0 to 99>

Type H: Floating point according to IEEE-standard

"Short floating Point Number IEEE STD 754" = R32IEEESTD754
R32IEEESTD754 := R32.23 {Fraction, Exponent, Sign}

Fraction = F := UI23 [1to 23] <0 to $1-2^{-23}$>
Exponent = E := UI8 [24 to 31] <0 to 255>
Sign = S := BS1 [32] S<0> = positive
S <1> = negative

F <0> and E <0>	:= (-1) S ∗ 0	= ± zero
F <≠0> and E <0>	:= (-1) S ∗ 2E-126(0.F)	= denormalized numbers
E <1 to 254>	:= (-1) S ∗ 2E-127(1.F)	= normalized numbers
F <0> and E <255>	:= (-1) S ∗ ∞	= ± infinite
F <≠0> and E <255>	:= NaN	= not a number, regardless of S

bits	8	7	6	5	4	3	2	1
octet 1	F = Fraction							
	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}
octet 2	F = Fraction							
	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
octet 3	E (LSB)	F = Fraction						
	2^{-0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
octet 4	Sign	E = Exponent						
	S	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}

The following ranges are specified by IEE Std 754-1985 for floating point arithmetics:

Range: $(-2^{128} + 2^{104})$ to $(+2^{128} - 2^{104})$, that is $-3.4*10^{38}$ to $+3.4*10^{38}$
smallest negative number: -2^{-149}, that is: $-1.4*10^{-45}$
smallest positive number: $+2^{-149}$, that is: $+ 1.4*10^{-45}$

8.3 Tables for Fixed Data Structure

8.3.1 Measured Medium Fixed Structure

Value hexadecimal	Field Medium/Unit				Medium
	Bit 16	Bit 15	Bit 8	Bit 7	
0	0	0	0	0	Other
1	0	0	0	1	Oil
2	0	0	1	0	Electricity
3	0	0	1	1	Gas
4	0	1	0	0	Heat
5	0	1	0	1	Steam
6	0	1	1	0	Hot Water
7	0	1	1	1	Water
8	1	0	0	0	H.C.A.
9	1	0	0	1	Reserved
A	1	0	1	0	Gas Mode 2
B	1	0	1	1	Heat Mode 2
C	1	1	0	0	Hot Water Mode 2
D	1	1	0	1	Water Mode 2
E	1	1	1	0	H.C.A. Mode 2
F	1	1	1	1	Reserved

Notes:

1. Record Medium/Unit is always least significant byte first.
2. H.C.A. = Heat Cost Allocator
3. Media from "Gas Mode2" to "H.C.A. Mode2" are defined additionally to EN1434-3 for some existing meters with CI-Field 73h (intentionally mode1), which transmit the multibyte records with high byte first in contrast to the CI-Field. The master must know that these media codes mean mode 2 or high byte first. Further use of these codes for "pseudo media" is not allowed for new developments.

8.3.2 Table of Physical Units

Unit	MSB..LSB	Hex code share Byte 7/8	Unit	MSB..LSB	Hex code share Byte 7/8
h,m,s	000000	00	MJ/h	100000	20
D,M,Y	000001	01	MJ/h * 10	100001	21
Wh	000010	02	MJ/h * 100	100010	22
Wh * 10	000011	03	GJ/h	100011	23
Wh * 100	000100	04	GJ/h * 10	100100	24
kWh	000101	05	GJ/h * 100	100101	25
kWh * 10	000110	06	ml	100110	26
kWh * 100	000111	07	ml * 10	100111	27
MWh	001000	08	ml * 100	101000	28
MWh * 10	001001	09	l	101001	29
MWh * 100	001010	0A	l * 10	101010	2A
kJ	001011	0B	l * 100	101011	2B
kJ * 10	001100	0C	m^3	101100	2C
kJ * 100	001101	0D	m^3 * 10	101101	2D
MJ	001110	0E	m^3 * 100	101110	2E
MJ * 10	001111	0F	ml/h	101111	2F
MJ * 100	010000	10	ml/h * 10	110000	30
GJ	010001	11	ml/h * 100	110001	31
GJ * 10	010010	12	l/h	110010	32
GJ * 100	010011	13	l/h * 10	110011	33
W	010100	14	l/h * 100	110100	34
W * 10	010101	15	m^3/h	110101	35
W * 100	010110	16	m^3/h * 10	110110	36
kW	010111	17	m^3/h * 100	110111	37
kW * 10	011000	18	°C * 10^{-3}	111000	38
kW * 100	011001	19	units for HCA	111001	39
MW	011010	1A	reserved	111010	3A
MW * 10	011011	1B	reserved	111011	3B
MW * 100	011100	1C	reserved	111100	3C
kJ/h	011101	1D	reserved	111101	3D
kJ/h * 10	011110	1E	same but historic	111110	3E
kJ/h * 100	011111	1F	without units	111111	3F

8.4 Tables for Variable Data Structure

8.4.1 Measured Medium Variable Structure

Medium	Code bin. Bit 7 .. 0	Code hex.
Other	0000 0000	00
Oil	0000 0001	01
Electricity	0000 0010	02
Gas	0000 0011	03
Heat (Volume measured at return temperature: outlet)	0000 0100	04
Steam	0000 0101	05
Hot Water	0000 0110	06
Water	0000 0111	07
Heat Cost Allocator.	0000 1000	08
Compressed Air	0000 1001	09
Cooling load meter (Volume measured at return temperature: outlet) ♣	0000 1010	0A
Cooling load meter (Volume measured at flow temperature: inlet) ♣	0000 1011	0B
Heat (Volume measured at flow temperature: inlet)	0000 1100	0C
Heat / Cooling load meter ♣	0000 1101	OD
Bus / System	0000 1110	0E
Unknown Medium	0000 1111	0F
Reserved	.....	10 to 15
Cold Water	0001 0110	16
Dual Water	0001 0111	17
Pressure	0001 1000	18
A/D Converter	0001 1001	19
Reserved	.....	20 to FF

8.4.2 Data Field Codes

Length in Bit	Code	Meaning	Code	Meaning
0	0000	No data	1000	Selection for Readout
8	0001	8 Bit Integer	1001	2 digit BCD
16	0010	16 Bit Integer	1010	4 digit BCD
24	0011	24 Bit Integer	1011	6 digit BCD
32	0100	32 Bit Integer	1100	8 digit BCD
32 / N	0101	32 Bit Real	1101	variable length
48	0110	48 Bit Integer	1110	12 digit BCD
64	0111	64 Bit Integer	1111	Special Functions

Variable Length:

With data field = `1101b` several data types with variable length can be used. The length of the data is given with the first byte of data, which is here called LVAR.

LVAR = 00h .. BFh : ASCII string with LVAR characters
LVAR = C0h .. CFh : positive BCD number with (LVAR - C0h) • 2 digits
LVAR = D0h .. DFH : negative BCD number with (LVAR - D0h) • 2 digits
LVAR = E0h .. EFh : binary number with (LVAR - E0h) bytes
LVAR = F0h .. FAh : floating point number with (LVAR - F0h) bytes [to be defined]
LVAR = FBh .. FFh : Reserved

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)

8.4.3 Codes for Value Information Field (VIF)

The first block of the table contains integral values, the second typically averaged values, the third typically instantaneous values and the fourth block contains parameters (E: extension bit).

Coding	Description	Range Coding	Range
E000 0nnn	Energy	$10^{(nnn-3)}$ Wh	0.001Wh to 10000Wh
E000 1nnn	Energy	$10^{(nnn)}$ J	0.001kJ to 10000kJ
E001 0nnn	Volume	$10^{(nnn-6)}$ m^3	0.001l to 10000l
E001 1nnn	Mass	$10^{(nnn-3)}$ kg	0.001kg to 10000kg
E010 00nn	On Time	nn = 00 seconds nn = 01 minutes nn = 10 hours nn = 11 days	
E010 01nn	Operating Time	coded like OnTime	
E010 1nnn	Power	$10^{(nnn-3)}$ W	0.001W to 10000W
E011 0nnn	Power	$10^{(nnn)}$ J/h	0.001kJ/h to 10000kJ/h
E011 1nnn	Volume Flow	$10^{(nnn-6)}$ m^3/h	0.001l/h to 10000l/h
E100 0nnn	Volume Flow ext.	$10^{(nnn-7)}$ m^3/min	0.0001l/min to 1000l/min
E100 1nnn	Volume Flow ext.	$10^{(nnn-9)}$ m^3/s	0.001ml/s to 10000ml/s
E101 0nnn	Mass flow	$10^{(nnn-3)}$ kg/h	0.001kg/h to 10000kg/h
E101 10nn	Flow Temperature	$10^{(nn-3)}$ °C	0.001°C to 1°C
E101 11nn	Return Temperature	$10^{(nn-3)}$ °C	0.001°C to 1°C
E110 00nn	Temperature Difference	$10^{(nn-3)}$ K	1mK to 1000mK
E110 01nn	External Temperature	$10^{(nn-3)}$ °C	0.001°C to 1°C
E110 10nn	Pressure	$10^{(nn-3)}$ bar	1mbar to 1000mbar
E110 110n	Time Point	n = 0 date n = 1 time & date	data type G data type F
E110 1110	Units for H.C.A.		dimensionless
E110 1111	Reserved		
E111 00nn	Averaging Duration	coded like OnTime	
E111 01nn	Actuality Duration	coded like OnTime	
E111 1000	Fabrication No		
E111 1001	(Enhanced)		see chapter 6.4.2 ♣
E111 1010	Bus Address		data type C (x=8)

VIF-Codes for special purposes:

Coding	Description	Purpose
1111 1011	Extension of VIF-codes	true VIF is given in the first VIFE and is coded using table 8.4.4 b) (128 new VIF-Codes)
E111 1100	VIF in following string (length in first byte)	allows user definable VIF´s (in plain ASCII-String) *
1111 1101	Extension of VIF-codes	true VIF is given in the first VIFE and is coded using table 8.4.4 a) (128 new VIF-Codes)
E111 1110	Any VIF	used for readout selection of all VIF´s (see chapter 6.4.3)
E111 1111	Manufacturer Specific	VIFE´s and data of this block are manufacturer specific

Note:

* Coding the VIF in an ASCII-String in combination with the data in an ASCII-String (datafield in DIF = 1101 b) allows the representation of data in a free user defined form.

8.4.4 Extension of primary VIF-Codes

If the VIF contains an extension indicator (VIF = $FD or $FB) the true VIF is contained in the first VIFE.

a) Codes used with extension indicator $FD

Coding	Description	Group
E000 00nn	Credit of 10^{nn-3} of the nominal local legal currency units	Currency Units
E000 01nn	Debit of 10^{nn-3} of the nominal local legal currency units	
E000 1000	Access Number (transmission count)	Enhanced Identification
E000 1001	Medium (as in fixed header)	
E000 1010	Manufacturer (as in fixed header)	
E000 1011	Parameter set identification	
E000 1100	Model / Version	
E000 1101	Hardware version #	
E000 1110	Firmware version #	
E000 1111	Software version #	
E001 0000	Customer location	Implementation of all TC294 WG1 requirements (improved selection ..)
E001 0001	Customer	
E001 0010	Access Code User	
E001 0011	Access Code Operator	
E001 0100	Access Code System Operator	
E001 0101	Access Code Developer	
E001 0110	Password	
E001 0111	Error flags (binary)	
E001 1000	Error mask	
E001 1001	Reserved	
E001 1010	Digital Output (binary)	
E001 1011	Digital Input (binary)	
E001 1100	Baudrate [Baud]	
E001 1101	response delay time [bittimes]	
E001 1110	Retry	
E001 1111	Reserved	

Coding	Description	Group
E010 0000	First storage # for cyclic storage	Enhanced storage management
E010 0001	Last storage # for cyclic storage	
E010 0010	Size of storage block	
E010 0011	Reserved	
E010 01nn	Storage interval [sec(s)..day(s)]	
E010 1000	Storage interval month(s)	
E010 1001	Storage interval year(s)	
E010 1010	Reserved	
E010 1011	Reserved	
E010 11nn	Duration since last readout [sec(s)..day(s)]	
E011 0000	Start (date/time) of tariff	Enhanced tariff management
E011 00nn	Duration of tariff (nn=01 ..11: min to days)	
E011 01nn	Period of tariff [sec(s) to day(s)]	
E011 1000	Period of tariff months(s)	
E011 1001	Period of tariff year(s)	
E011 1010	dimensionless / no VIF	
E011 1011	Reserved	
E011 11xx	Reserved	
E100 nnnn	10^{nnnn-9} Volts	electrical units
E101 nnnn	$10^{nnnn-12}$ A	

Coding	Description	Group
E110 0000	Reset counter	
E110 0001	Cumulation counter	
E110 0010	Control signal	
E110 0011	Day of week	
E110 0100	Week number	
E110 0101	Time point of day change	
E110 0110	State of parameter activation	
E110 0111	Special supplier information	
E110 10pp	Duration since last cumulation [hour(s)..years(s)]	
E110 11pp	Operating time battery [hour(s)..years(s)]	
E111 0000	Date and time of battery change	
E111 0001 to E111 1111	Reserved	

Notes:

nn = 00 second(s)
01 minute(s)
10 hour(s)
11 day(s)

The information about usage of data type F (date and time) or data type G (date) can be derived from the datafield (0010b: type G / 0100: type F).

pp = 00 hour(s)
01 day(s)
10 month(s)
11 year(s)

b) Codes used with extension indicator $FB

Coding	Description	Range Coding	Range
E000 000n	Energy	$10^{(n-1)}$ MWh	0.1MWh to 1MWh
E000 001n	Reserved		
E000 01nn	Reserved		
E000 100n	Energy	$10^{(n-1)}$ GJ	0.1GJ to 1GJ
E000 101n	Reserved		
E000 11nn	Reserved		
E001 000n	Volume	$10^{(n+2)}$ m^3	100m3 to 1000m3
E001 001n	Reserved		
E001 01nn	Reserved		
E001 100n	Mass	$10^{(n+2)}$ t	100t to 1000t
E001 1010 to E010 0000	Reserved		
E010 0001	Volume ♣	0,1 feet^3	
E010 0010	Volume ♣	0,1 american gallon	
E010 0011	Volume	1 american gallon	
E010 0100	Volume flow ♣	0,001 american gallon/min	
E010 0101	Volume flow	1 american gallon/min	
E010 0110	Volume flow	1 american gallon/h	
E010 0111	Reserved		
E010 100n	Power	$10^{(n-1)}$ MW	0.1MW to 1MW
E010 101n	Reserved		
E010 11nn	Reserved		
E011 000n	Power	$10^{(n-1)}$ GJ/h	0.1GJ/h to 1GJ/h
E011 0010 to E101 0111	Reserved		
E101 10nn	Flow Temperature	$10^{(nn-3)}$ °F	0.001°F to 1°F
E101 11nn	Return Temperature	$10^{(nn-3)}$ °F	0.001°F to 1°F
E110 00nn	Temperature Difference	$10^{(nn-3)}$ °F	0.001°F to 1°F
E110 01nn	External Temperature	$10^{(nn-3)}$ °F	0.001°F to 1°F
E110 1nnn	Reserved		
E111 00nn	Cold / Warm Temperature Limit	$10^{(nn-3)}$ °F	0.001°F to 1°F
E111 01nn	Cold / Warm Temperature Limit	$10^{(nn-3)}$ °C	0.001°C to 1°C
E111 1nnn	cumul. count max power ♣	$10^{(nnn-3)}$ W	0.001W to 10000W

8.4.5 Codes for Value Information Field Extension (VIFE)

The following values for VIFE´s are defined for an enhancement of VIF´s other than $FD and $FB:

VIFE-Code	Description
E00x xxxx	Reserved for object actions (master to slave): see table on page 75 or for error codes (slave to master): see table on page 74
E010 0000	per second
E010 0001	per minute
E010 0010	per hour
E010 0011	per day
E010 0100	per week
E010 0101	per month
E010 0110	per year
E010 0111	per revolution / measurement
E010 100p	increment per input pulse on input channel #p
E010 101p	increment per output pulse on output channel #p
E010 1100	per liter
E010 1101	per m^3
E010 1110	per kg
E010 1111	per K (Kelvin)
E011 0000	per kWh
E011 0001	per GJ
E011 0010	per kW
E011 0011	per (K*l) (Kelvin*liter)
E011 0100	per V (Volt)
E011 0101	per A (Ampere)
E011 0110	multiplied by sek
E011 0111	multiplied by sek / V
E011 1000	multiplied by sek / A
E011 1001	start date(/time) of
E011 1010	VIF contains uncorrected unit instead of corrected unit
E011 1011	Accumulation only if positive contributions
E011 1100	Accumulation of abs value only if negative contributions
E011 1101 to E011 1111	Reserved

VIFE-Code	Description
E100 u000	u=1: upper, u=0: lower limit value
E100 u001	# of exceeds of lower u=0) / upper (U=1) limit
E100 uf1b	Date (/time) of: b=0: begin, b=1: end of, f=0: first, f=1: last, u=0: lower, u=1: upper limit exceed
E101 ufnn	Duration of limit exceed (u,f: as above, nn=duration)
E110 0fnn	Duration of (f: as above, nn=duration)
E110 1x0x	Reserved
E110 1f1b	Date (/time) of (f,b: as above)
E111 0nnn	Multiplicative correction factor: 10^{nnn-6}
E111 10nn	Additive correction constant: 10^{nn-3} • unit of VIF (offset)
E111 1100	Reserved
E111 1101	Multiplicative correction factor: 10^{3}
E111 1110	future value
E111 1111	next VIFE's and data of this block are maufacturer specific

Notes:

"Date(/time) of" or "Duration of" relates to the information which the whole data record header contains.

The information about usage of data type F (date and time) or data type G (date) can be derived from the datafield (0010b: type G / 0100: type F).

VIFE-Codes for reports of record errors (slave to master):

VIFE-Code	Type of Record Error	Error Group
E000 0000	None	DIF Errors
E000 0001	Too many DIFE´s	
E000 0010	Storage number not implemented	
E000 0011	Unit number not implemented	
E000 0100	Tariff number not implemented	
E000 0101	Function not implemented	
E000 0110	Data class not implemented	
E000 0111	Data size not implemented	
E000 1000 to E000 1010	Reserved	
E000 1011	Too many VIFE´s	VIF Errors
E000 1100	Illegal VIF-Group	
E000 1101	Illegal VIF-Exponent	
E000 1110	VIF/DIF mismatch	
E000 1111	Unimplemented action	
E001 0000 to E001 0100	Reserved	
E001 0101	No data available (undefined value)	Data Errors
E001 0110	Data overflow	
E001 0111	Data underflow	
E001 1000	Data error	
E001 1001 to E001 1011	Reserved	
E001 1100	Premature end of record	Other Errors
E001 1101 to E001 1111	Reserved	

VIFE-Codes for object actions (master to slave):

VIFE-Code	Action	Explanation
E000 0000	Write (Replace)	replace old with new data
E000 0001	Add Value	add data to old data
E000 0010	Subtract Value	subtract data from old data
E000 0011	OR (Set Bits)	data OR old data
E000 0100	AND	data AND old data
E000 0101	XOR (Toggle Bits)	data XOR old data
E000 0110	AND NOT (Clear Bits)	NOT data AND old data
E000 0111	Clear	set data to zero
E000 1000	Add Entry	create a new data record
E000 1001	Delete Entry	delete an existing data record
E000 1010	Reserved	
E000 1011	Freeze Data	freeze data to storage no.
E000 1100	Add to Readout-List	add data record to RSP_UD
E000 1101	Delete from Readout-List	delete data record from RSP_UD
E000 111x	Reserved	
E001 xxxx	Reserved	

Note:
The object action "write / replace" (VIFE = E000 0000) is the default and is assumed if there is no VIFE with an object action for this record.

8.5 References

[1] Färber, G. : Bussysteme, R.Oldenbourg Verlag München Wien, 1987

[2] Gabele, E., Kroll, M., Kreft, W. : Kommunikation in Rechnernetzen, Springer Verlag Heidelberg, 1991

[3] Steffens, Andreas : Diplomarbeit " Der M-Bus - Eigenschaften und Anwendungen", University of Paderborn, Department of Physics, 1992

[4] Texas Instruments Deutschland GmbH : Data Sheet TSS 721, 1993

[5] Texas Instruments Deutschland GmbH : Seminar Material, M-Bus Workshop, 1992

[6] Ziegler, Horst : Seminar Material, M-Bus Workshop, 1992

[7] IEC 870-5-1 : Telecontrol Equipment and Systems, Part 5 Transmission Protocols, Section One - Transmission Frame Formats, 1990

[8] IEC 870-5-2 : Telecontrol Equipment and Systems, Part 5 Transmission Protocols, Section Two - Link Transmission Procedures, 1992

[9] EN1434-3: Heat Meters, Part 3 Data Exchange and Interface, 1997 ♣

[10] Aquametro AG Therwil : M-Bus Automatic Slave Recognition with Wildcard Algorithm, 1992

[11] Papenheim, Andreas: Diplomarbeit " Anwendungsbeispiele für den M-Bus", University of Paderborn, Department of Physics, 1993

[12] Texas Instruments Deutschland GmbH: Applications Report "Designing Applications for the Meter-Bus", 1994 (translation of reference [11])

[13] Ziegler, Horst; Froschermeier Günther: "M-Bus: Die Meßbus-Alternative", Elektronik 16/1993