# INTERNATIONAL STANDARD

**ISO 10368**

First edition
1992-07-01

# Freight thermal containers — Remote condition monitoring

*Conteneurs à caractéristiques thermiques — Système de pilotage à distance des groupes frigorifiques*

Reference number
ISO 10368:1992(E)

## Contents

## Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO 10368 was prepared by Technical Committee ISO/TC 104, *Freight containers*, Sub-Committee SC 2, *Specific purpose containers*.

Annex A of this International Standard is for information only.

*blank*

# Section 1. General



## Introduction

During the preparation of this International Standard, information was gathered on patents upon which application of this standard might depend.

For the low data rate system of **3.5**, the relevant patents were identified as belonging to

Thermo King Corporation
314 W. 90th Street
Minneapolis, Minnesota 55420
USA

For the high data rate system of **3.6**, the relevant patents were identified as belonging to

Adaptive Networks Incorporated
1505 Commonwealth Ave.
Suite 30
Brighton, Massachusetts 02135
USA

ISO cannot give authoritative or comprehensive information about the evidence, validity or scope of patent and like rights. The patent holders have stated that licences will be granted under reasonable terms and conditions. Communications on this subject should be addressed to either Thermo King Corporation or Adaptive Networks Incorporated.

## 1.1 Scope

This International Standard establishes the information and interfaces required to permit complying central monitoring and control systems employed by one carrier or terminal to interface and communicate with complying remote communication devices of differing manufacture and configuration used by other carriers and terminals.

The data-logging formats and message protocols outlined in this International Standard apply to all currently available data rate transmission techniques. These formats and protocols also apply to all future techniques designed to be an ISO standard compatible system.

The performance requirements for the monitoring, communication and control system are given in section 2. The system compatibility requirements are given in section 3. All sections of this International Standard apply to all implementations, except where specified.

## 1.2 Normative references

The following standards contain provisions which, through reference in this text constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 1496-2:1988, *Series 1 freight containers — Specification and testing — Part 2: Thermal containers.*

ISO 9711-2:1990, *Freight containers — Information related to containers on board vessels — Part 2: Telex data transmission.*

## 1.3 Definitions

For the purposes of this International Standard, the following definitions apply.

**1.3.1**
**remote communications device (RCD)**

device which is physically a part of the refrigeration machinery and which communicates with any complying central monitoring and control systems (CMCSs) using the refrigeration machinery power distribution system as, the data transmission medium. See Figure 1 and Figure 2

there are two distinct types of RCD:

— a stand-alone remote communications device (sRCD) (see **1.3.9**);

— an integrated remote communications device (iRCD) (see **1.3.10**).

an sRCD is defined as a device which cannot implement the hardware write messages defined in **3.4.6.1**

**1.3.2**
**central monitoring and control system (CMCS)**

system consisting of hardware and software which monitors and controls one or more remote communications devices (RCDs). A typical system consists of at least

a) operator interface devices,

b) a master monitoring unit (MMU), and

c) power line data link equipment, such as a multiple data rate central control unit (MDCU).

**1.3.3**
**master monitoring unit (MMU)**

central processing unit such as a computer which contains specific hardware and software to control the entire remote condition monitoring system. It is the interface between the human operator and the network

**1.3.4**
**multiple data rate central control unit (MDCU)**

device which forms the link between the master monitoring unit (MMU) and the three-phase power line bus which contains the individual remote communications devices (RCDs). An MDCU consists of two components as follows:

a) a central control unit capable of receiving and transmitting at the data rates which meet the requirements of this International Standard;

b) a central control interface.

**1.3.5**
**high data rate remote communications device (HRCD)**

remote communications device (RCD) which transmits data at a high data rate, e.g. 19 200 baud

**1.3.6**
**low data rate remote communications device (LRCD)**

remote communications device (RCD) which communicates data at a low data rate, e.g. 1 200 baud

**1.3.7**
**high data rate central control unit (HDCU)**

device which links the master monitoring unit (MMU) and the power line network, communicating with the high data rate remote communications devices (HRCDs)

**1.3.8**
**low data rate central control unit (LDCU)**

device which links the master monitoring unit (MMU) and the power line network, communicating with the low data rate remote communications devices (LRCDs)

**1.3.9**
**stand-alone remote communications device (sRCD)**

slave remote communications device (RCD) which, with limited capabilities, merely monitors a container refrigeration unit. An sRCD can be either high or low data rate

**1.3.10**
**integrated remote communications device (iRCD)**

slave remote communications device (RCD) which interfaces to a refrigeration unit controller via an EIA RS232-C serial interface and can control the refrigeration machinery. An iRCD can be either high or low data rate

**1.3.11**
**controller**

device that monitors and controls the refrigeration machinery

# Section 2. Performance requirements



## 2.1 Scope

This section specifies the performance requirements of central monitoring and control systems (CMCSs) necessary for them to interface and communicate with complying remote communications devices (RCDs).

## 2.2 Requirements

### 2.2.1 System components

#### 2.2.1.1 *Remote condition monitoring system components*

A single remote condition monitoring system consists of a maximum of one master monitoring unit (MMU) and one multiple data rate central control unit (MDCU). See Figure 1 a).

#### 2.2.1.2 *Multiple data rate central control unit (MDCU)*

An MDCU may include one high data rate central control unit (HDCU) and one low data rate central control unit (LDCU). If an HDCU and an LDCU are both present, the HDCU and LDCU are joined together by a central control unit (CCU) interface, and the three components together form the MDCU.

#### 2.2.1.3 *MMU/MDCU interface*

The preferred method of connecting the MMU to the MDCU complex is through a single port as shown in Figure 1 a). However, certain expansion paths may require multiple connections as shown in Figure 1 b).

#### 2.2.1.4 *High and low data rate remote communications devices (HRCDs and LRCDs)*

HRCDs and LRCDs shall be able to coexist on the same power line network and not interfere with simultaneous communications with either the HDCU or the LDCU.

#### 2.2.1.5 *MDCU components*

An MDCU may consist of either a single HDCU (to communicate with the HRCDs on the network) or a single LDCU (to communicate with LRCDs on the network). However, all signalling protocols, data-logging formats, power levels, insertion rates and other physical requirements shall be identical to that which would be used for a combined system and therefore must be compatible. Refer to **3.2** and **3.3** for the required protocol and data-logging formats.

### 2.2.2 Performance function

#### 2.2.2.1 *Standard message*

All RCDs shall respond to a minimum list of standardized enquiries (see **2.2.2.4**) and commands with a standardized reply or acknowledgement.

#### 2.2.2.2 *Acknowledgement message*

The RCD shall send an acknowledgement message for all commands and enquiries that are received and understood.

#### 2.2.2.3 *"Not able" message*

If the RCD is not capable of executing a command received or of responding to an enquiry because of the configuration of the RCD and the thermal control machinery, it shall respond with a "Not able" message.

#### 2.2.2.4 *Required enquiries*

All RCDs shall respond to the following required enquiries.

**2.2.2.4.1 Identification number:** For an integrally refrigerated or thermal container this will be the container ISO number comprising a 4-letter alphabetical prefix and a 7-digit suffix (including the check digit). Where a demountable marine clip-on unit (MCOU) is used, the identification number will be the MCOU number in ISO format.

**2.2.2.4.2 Porthole container number:** This response will be in addition to the identification number for MCOU systems.

**2.2.2.4.3 Porthole number change:** This is recorded in the RCD memory in alphanumerical format together with the time of the change.

**2.2.2.4.4 Return air temperature:** In the form of a positive or negative value, expressed in degrees Celsius to one decimal place, within the range – 30,0 °C to + 38,0 °C.

**2.2.2.4.5 Supply air temperature:** Expressed in the same format as **2.2.2.4.4**.

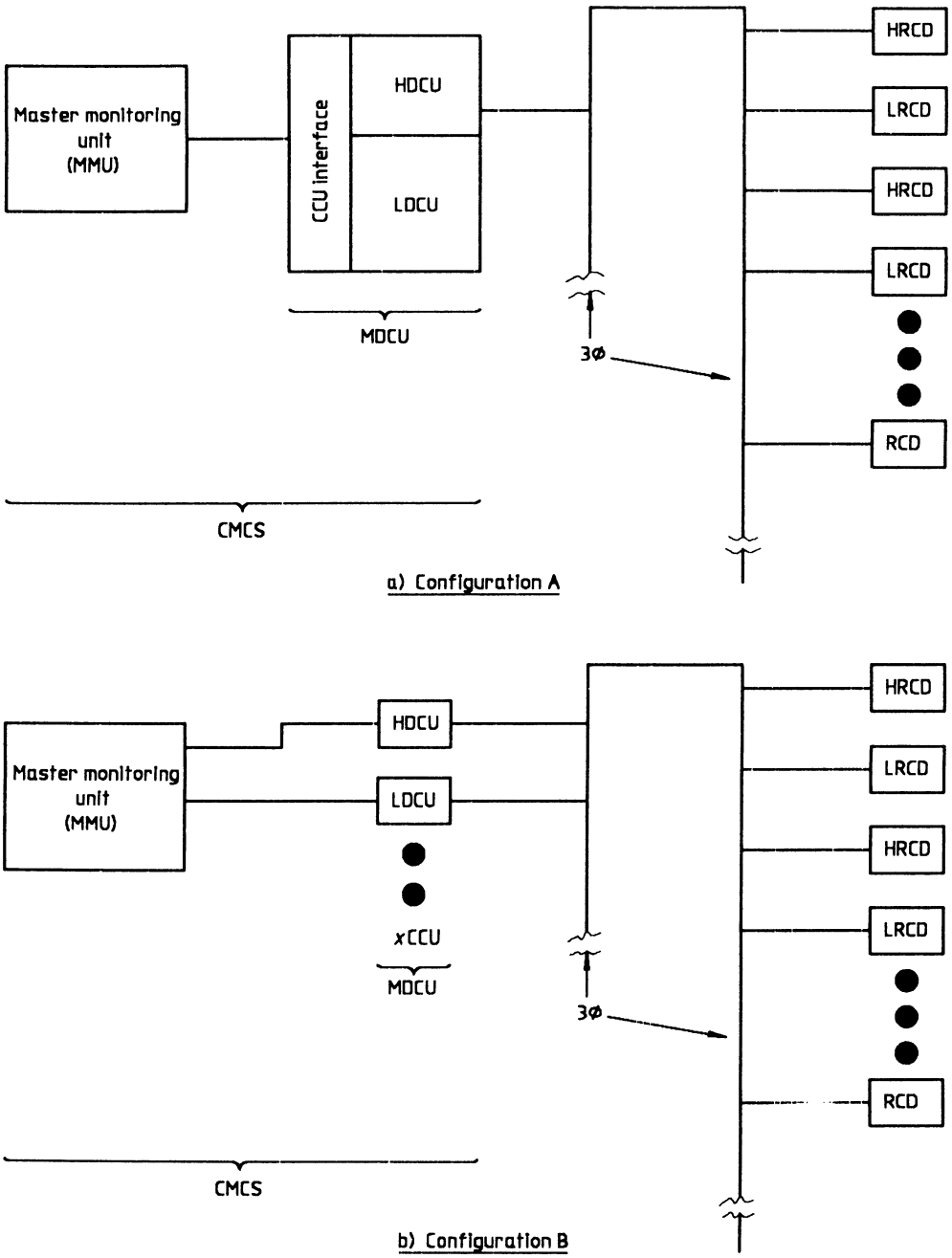



**Figure 1 — Remote condition monitoring system components layout**

**2.2.2.4.6 RCD manufacturer and type:** Consisting of a unique identification number registered and controlled by ISO, and for which ISO/TC 104 is the registration authority.

### 2.2.2.5 *Optional standard enquiries*

Other optional enquiries are standardized. RCDs arid refrigeration machinery so equipped shall respond to the following enquiries. RCDs not so equipped shall respond "Not able" (see **2.2.2.3**).

**2.2.2.5.1 Operating mode:** Full cool, Partial or Lower capacity cool, Modulated cool, Fans only or Null mode, Defrost, Heat, Off.

**2.2.2.5.2 Set-point temperature:** Expressed in the same format as **2.2.2.4.4**.

**2.2.2.5.3 Alarms:** High refrigeration pressure, Temperature out of range, Low compressor oil pressure, Defrost/Heat/Overheat, Compressor overload, Controller failure, Sensor failure — Return air, Sensor failure — Supply air, Power off, Amperage draw too high, Amperage draw too low, Defrost (out of time).

(Capacity for future development, e.g. controlled atmosphere.)

**2.2.2.5.4 All current alarms:** In sequence of occurrence.

**2.2.2.5.5 Product temperatures:** For example, tank, poultry.

**2.2.2.5.6 Data-logger interval:** One digit in half-hour intervals up to a maximum of 12 h.

**2.2.2.5.7 Amperage:** 0 to 63,75 A in 0,25 A intervals.

**2.2.2.5.8 Destination:** Three alphanumerical digits. If the destination changes, both the old and the current destination may be declared.

**2.2.2.5.9 Port of discharge:** Three alphanumerical digits.

**2.2.2.5.10 Origin:** Three alphanumerical digits.

**2.2.2.5.11 Report results of self-check level 1:** One digit, 0 = Fail. 1 = Pass.

**2.2.2.5.12 Report results of self-check level *n*:** In the format of up to 256 ASCII characters, where *n* is a single character between two and nine.

**2.2.2.5.13 Vessel and voyage designation:** (See ISO 9711-2.)

**2.2.2.6 *Commands***

RCDs and refrigeration machinery if so equipped shall respond to the following commands. RCDs not so equipped shall respond "Not able" (see **2.2.2.3**).

**2.2.2.6.1 Change set-point temperature:** Expressed in the same format as **2.2.2.4.4**.

**2.2.2.6.2** *Initiate self-check level 1*

**2.2.2.6.3 Initiate self-check level *n*:** In the same format as **2.2.2.5.12**, where *n* is in the range two to nine.

**2.2.2.6.4 Change identification number:** Expressed in the same format as **2.2.2.4.1**.

**2.2.2.6.5 Change data-logger interval:** Expressed in the same format as **2.2.2.5.6**.

**2.2.2.6.6 Set data-logger time and date:** With the date expressed in the format year/month/day.

**2.2.2.6.7 Change operating mode:** Expressed in the same format as **2.2.2.5.1**.

**2.2.2.6.8** *Download data-logger record to central monitor*

**2.2.2.6.9 Change porthole container number:** Expressed in the same format as **2.2.2.4.3**.

**2.2.2.6.10 Change destination:** Expressed in the same format as **2.2.2.5.8**.

**2.2.2.7 *Indecipherable or unserviceable messages***

Indecipherable or unserviceable messages shall not cause the RCD or CMCS to "crash" or "hang up". Also, failures of an electronic device in any RCD shall not cause the system to "crash" or "hang up".

**2.2.2.8 *Verification of container identification number***

The CMCS, if so equipped, shall verify the container identification number, using the check digit (the seventh digit of the numerical suffix) and an algorithm selected.

**2.2.3 Performance constraints**

**2.2.3.1 *Power interference***

RCDs and CMCSs shall not interfere with the proper functioning of power supply regulating or control devices, such as voltage regulators or protective relaying equipment.

**2.2.3.2 *Marine device interference***

CMCSs and RCDs, individually or as a system, shall not interfere with standard marine navigation and communication devices.

**2.2.3.3** ***System size***

All CMCSs shall be suitable to coordinate and report on a system of 1 024 RCDs active at the same time on one CMCS.

**2.2.3.4** ***Status update***

The MMU/MDCU system shall generate RCD updated status per **2.2.2.4** at least once per hour per container for a system of up to 1 024 containers active at the same time on one CMCS.

**2.2.3.5** ***Automatic RCD system list***

The population or database of RCDs on the CMCS shall be self-generating. No input to the MMU, whether from an operator or from another computer, shall be necessary to determine the RCDs connected to that system.

**2.2.3.6** ***Identification of new RCDs***

The MMU/MDCU system shall be designed to identify an average of at least one new container every 10 s, or 6 per minute.

**2.2.3.7** ***Voltage and frequency requirements***

RCDs shall be suitable for operating on the voltage systems specified in ISO 1496-2.

**2.2.3.8** ***Dual voltage requirements***

No special handling shall be required for RCDs to operate on the three voltage types of refrigeration machinery. (See ISO 1496-2:1988, subclause **7.2**.)

**2.2.3.9** ***RCD connection***

The RCD shall be connected on the line side of the refrigeration machinery disconnect Or circuit breaker, if any, so that communication is possible when the refrigeration machinery is switched off. The RCD may have its own disconnect switch for servicing.

**2.2.3.10** ***Error rates***

**2.2.3.10.1** All CMCSs and RCDs shall be designed to meet the following error rate criteria.

The RCD/MDCU communication system may have two different types of “undetected and uncorrected” communication errors. An “undetected and uncorrected” communication error is one which is not detected and corrected within 5 min after occurrence.

**2.2.3.10.2** An error whereby an RCD executes a command which was not commanded by the MMU shall not occur more often than one time in $25 \times 10^6$ messages (i.e. any power line disturbance which the receiver interprets as a message), or more often than once in 10 years for each CMCS, whichever is greater.

**2.2.3.10.3** An error whereby a CMCS misinterprets a message (i.e. any power line disturbance which the receiver interprets as a message) shall not occur more often than one time in $25 \times 10^5$ messages.

# Section 3. System compatibility requirements

## 3.1 Scope

This section specifies the interface requirements for communications protocol, data-logging formats, message definitions, and physical requirements for low data rate and high data rate (CU and CD).

## 3.2 Communications protocol

Each remote condition monitoring system has three interface areas as follows (see Figure 2):

— MMU to MDCU interface;

— MDCU to RCD interface;

— RCD to refrigeration machinery controller interface.

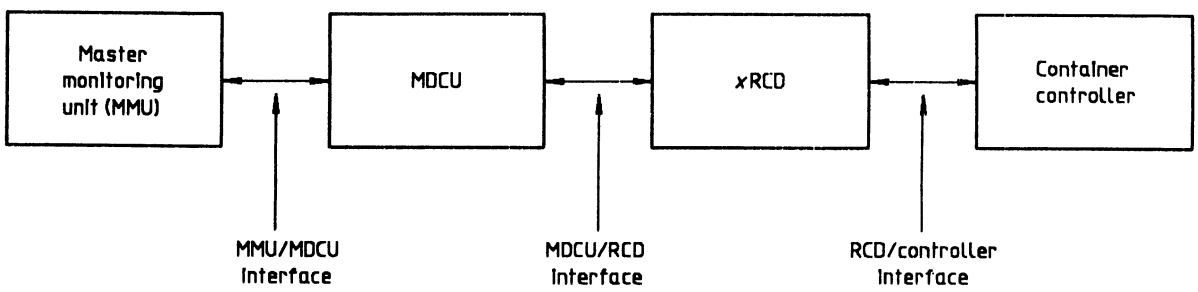



**Figure 2 — Remote condition monitoring — Communications interfaces**

### 3.2.1 MMU to MDCU communications

This subclause, in pad, defines the communications protocol to be used when the MDCU is implemented as a discrete system component which is separate from the MMU architecture. The requirements given in this subclause do not preclude the use of bus-based open architecture MDCU applications where the EIA RS232-C is not appropriate.

The MMU communicates with the MDCU via a full duplex EIA RS232-C serial interface. The baud rate shall be at least two times the baud rate of the fastest RCD in the system. A typical communications baud rate is 4 800 baud. Each character transferred requires 1 start bit (low logic level), 8 data bits, and 1 stop bit (high logic level). The-minimum time delay required between packets is one character time. This, therefore, restricts deadtime between any 2 bytes in a packet to less than one character delay.

The messages for succeeding interfaces are embedded in the formats of earlier stages (see Figure 3). These messages may be intended for action by the MDCU only. These messages are described fully in **3.2.1.1**. If the message contains embedded data intended for an RCD, the format is as described in **3.2.1.3**. Similarly, embedded data intended for the refrigeration machinery is described in **3.2.1.5** and **3.2.4**. Note that the field length, in bytes, is also defined in Figure 3.

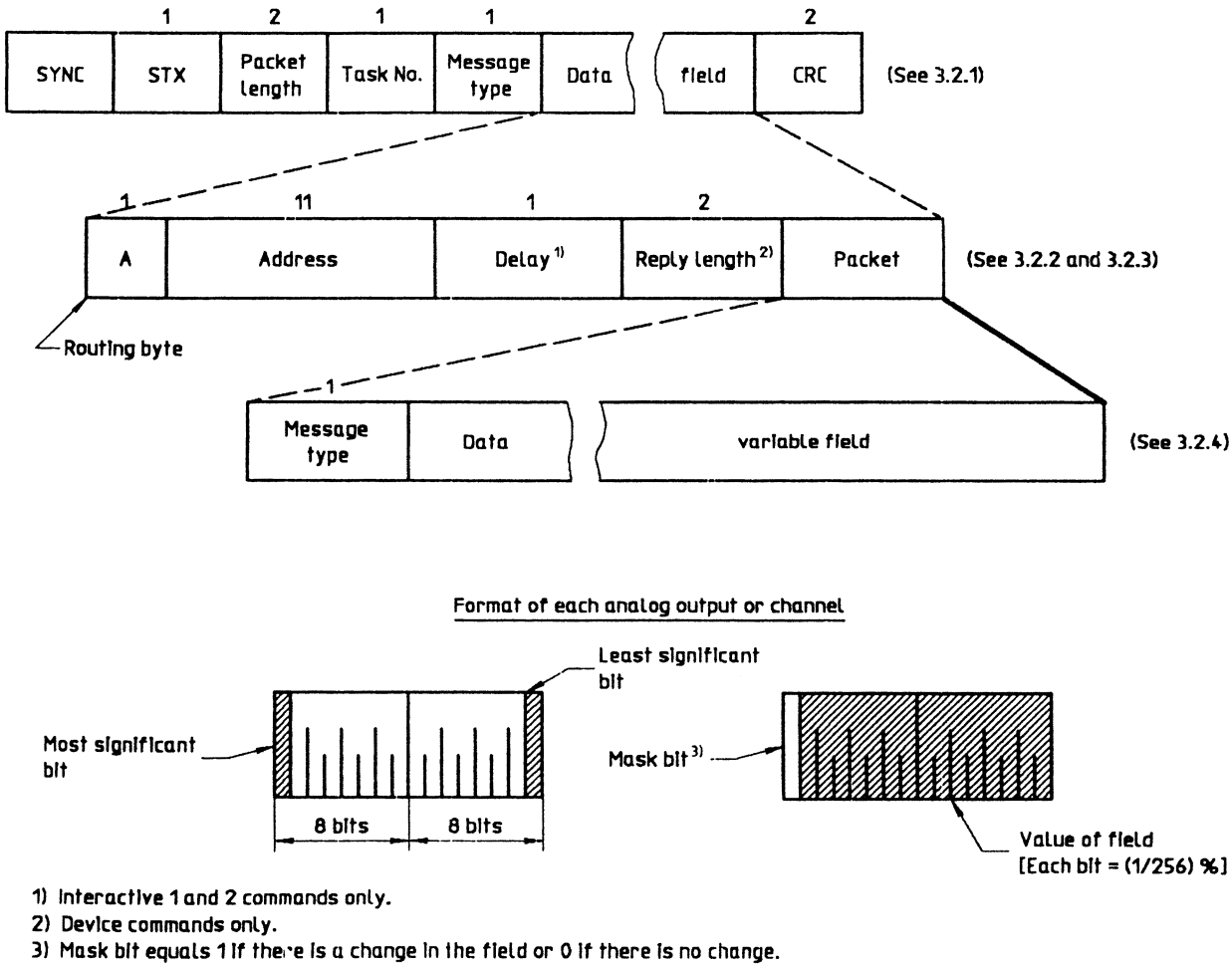



**Figure 3 — Message format overview**

The information transfer format from the MMU to the MDCU is as shown in Figure 4.

| | SYNC (optional) | STX | Packet length | Task No. | Xmit type | Data | CRC |
|---|---|---|---|---|---|---|---|
| No. of bytes | | 1 | 2 | 1 | 1 | | 2 |

**Figure 4 — Information transfer format from the MMU to the MDCU**

The information transfer format returned to the MMU from the MDCU is as shown in Figure 5.

| | SYNC (optional) | STX | Packet length | Task No. | Reply type | Data | CRC |
|---|---|---|---|---|---|---|---|
| No. of bytes | | 1 | 2 | 1 | 1 | | 2 |

**Figure 5 — Information transfer format returned to the MMU from the MDCU**

The fields in Figure 4 and Figure 5 are defined as follows.

| | |
|---|---|
| SYNC | An optional synchronization field. It can be any number of bytes, the contents of which is 16H. The MDCU strips all SYNC characters from the start of a message. |
| STX | Delimits the start of a valid message. It indicates that the next 2 bytes are the packet length. |
| Packet length | A 2 byte unsigned integer which gives the packet length in bytes not including the SYNC, STX or CRC fields. It is transmitted high byte followed by low byte. |
| Task No. | A 1 byte number assigned to the job before transmission to the MDCU. It is not used by the MDCU and is defined by the application. The MDCU reply will contain the same value in this field.<br>Task numbers shall be assigned sequentially by the MMU from 0 to 254 and then resume at 0. Task number 255 is reserved for the MDCU reset command. |
| Xmit type | A 1 byte command directing the MDCU as to the type of processing which is to be performed on the data. The assignments are discussed in **3.2.1.1**. |
| Reply type | A 1 byte command response from the MDCU indicating the success or failure status of the corresponding job received by the MDCU. The assignments are discussed in **3.2.1.2**. |
| Data | Variable length data field in which the contents are dependent on the type of message to be processed. The contents are defined in **3.2.1.1** and **3.2.1.2** for each appropriate command. |
| CRC | A 16 bit error check field generated by using the CRC-16 polynomial:<br>$x^{16} + x^{15} + x^2 + 1$<br>The optional SYNC characters are not included in the CRC calculation. |

#### 3.2.1.1 *MMU to MDCU transmit command types*

There are two groups of commands which are sent to the MDCU: those directed to the MDCU for internal processing only and those to be transferred over the power line. The former group is given in Table 1, while the latter group is given in Table 2. An explanation of each command follows.

##### 3.2.1.1.1 *MDCU directed command types*

This group of commands requires processing within the MDCU, i.e. the power line communications network is not accessed. The commands currently supported by the MDCU are given in Table 1 and explained in **3.2.1.1.1.1** to **3.2.1.1.1.3**.

**Table 1 — MDCU interrogation directed command types**

| Command | Value |
|---|---|
| MDCU reset | 20H |
| MDCU parameters | 22H |
| MDCU status | 23H |

###### 3.2.1.1.1.1 *MDCU reset command*

Upon request, the MDCU will perform its internal diagnostics and initialize with default parameters. Any data associated with this command will be ignored. Since the MDCU is reset, the initial status condition with default parameters will be returned to the MMU as defined in **3.2.1.1.1.2**. Task number 255 will be used in the reply.

###### 3.2.1.1.1.2 *MDCU parameters command*

The MDCU parameters command loads the MDCU with information passed in the data field portion of the command. A data field containing more than 15 bytes will cause the MDCU to ignore the command. The parameters to be loaded are defined below. Note that the parameters used for the low data rate system only are prefixed with an L while the parameters used for the high data rate system only are prefixed with an H.

First byte transmitted:

Parameter0 — Test status byte is a read-only location and, therefore, this parameter is ignored by the MDCU.

Parameter1-2 — MDCU software version/subversion is a read-only location and, therefore, this parameter is ignored by the MDCU.

Parameter3 — Change the MMU handshaking flag byte between the MDCU and MMU. Each bit is discussed below.

Bit7 — MDCU external UART CTS status (MMU communications). This bit will always be received by the MMU in the active, or logic one, state since the UART is in the CTS mode (i.e. communicating with the MMU). This bit is a read-only bit; the MMU cannot set/reset it.

Bit6-0 — Although accessible to the MMU, at present not used by the MDCU.

Parameter4-5 — Change available buffer size high (4) and low (5) bytes. Decreases/expands power line buffers. If the new size is below/above the MDCU's minimum/maximum value, its minimum/maximum value is used. A value of zero, or a value received which is equivalent to the previous setting, will retain the previous setting. If a change in the buffer size is performed, all jobs within queue will be lost.

LParameter6 — Change the maximum size of the packets (or blocks) to be transferred on the power line. If received larger than the available buffer size setting (described above), that value will be substituted. A value of zero will retain the previous setting (unless that setting is greater than the new available buffer size). It is recommended that the packet size not be set greater than its default value (128 or 80H) unless the power line message timeout settings within the MDCU are changed accordingly.

LParameter7 — Change the number of retransmissions to be attempted for a given job on the power line.

LParameter8 — Change the counter indicating the number of retries which have been performed on the power line.

LParameter9 — Change the counter indicating the number of successful normal polls which required one retry on the power line.

LParameter10 — Change the counter indicating the number of successful normal polls which required two retries on the power line.

Parameter11 — Change the time delay base value used within an interactive command (**3.2.1.1.2.3** and **3.2.1.1.2.8**) that is required between the MDCU transmission of data to an RCD and the interactive poll used to obtain a response from the RCD. Each count provides a 0,1 s incremental delay.

Parameter12-14 — At present undefined. The information will be copied into the status buffer as received.

Since the first 3 bytes of the status buffer are read-only locations, any data field containing less than 3 bytes will not change any parameters in the MDCU status buffer. Also, Parameter4-5 is 2 bytes in length and a byte count which transfers only one of these bytes will not update this parameter. Regardless of whether any parameters were changed, the MDCU reply will return the current status value to the MMU in a general valid type reply, as defined in **3.2.1.2.1**.

**3.2.1.1.1.3** *MDCU status command*

The MDCU status command simply requests the MDCU to reply with its present status value. Any data associated with this command will be ignored. The definition of this status buffer corresponds to the assignments discussed in **3.2.1.1.1.2** (MDCU parameters command), or as defined in **3.2.1.2.1**.

**3.2.1.1.2** *MDCU power line command types*

This group of commands requires communication with RCDs over the power line communications network. The commands currently supported by the MDCU are given in Table 2. The format of the power line data field for the RCD directed commands to be deciphered by the RCD (RCD message) is given in **3.2.1.2**, while the format of the power line data field for the device-related commands to be deciphered by the controller device for an iRCD, or a device input/output processing for an sRCD (device message), is given in **3.2.4**.

In all power line communication commands, the first byte of the data field is the routing byte. The routing byte's identification is derived from the original RCD map response (see **3.2.1.1.2.6**) and is used by the MDCU to route the message to the proper data rate modem. Bits 7-6 define routing information for the low data rate RCDs and high data rate RCDs as described in Table 3. The remaining bits (5-0) are defined in the appropriate low or high data rate subclauses, **3.2.2** and **3.2.3** respectively.

**Table 2 — MDCU power line directed command types**

| Command | Value |
|---|---|
| RCD poll | 30H |
| RCD xmit1 | 31H |
| RCD xmit2 | 32H |
| RCD interactive1 | 33H |
| RCD interactive2 | 34H |
| RCD map | 35H |
| RCD interactive poll | 36H |
| Device poll | 40H |
| Device xmit1 | 41H |
| Device xmit2 | 42H |
| Device interactive1 | 43H |
| Device interactive2 | 44H |
| Device map | 45H |
| Device interactive poll | 46H |

**Table 3 — Routing byte data rate bit assignments**

| Bits 7-6 | Definition |
|---|---|
| 0 0 | Illegal |
| 0 1 | ISO LRCD |
| 1 0 | ISO HRCD |
| 1 1 | Reserved |

The routing byte is always followed by an 11 byte ASCII RCD network address. Two address fields are reserved for special use in the RCDs. A universal address consisting of all ASCII 0s, or all 30H, is recognized by all RCDs (i.e. a broadcast address) while a field containing nine ASCII 0s followed by two ASCII 1s or nine 30Hs plus two 31Hs, is an address substituted in an RCD containing an invalid personal address.

The command or message type, routing byte and RCD network address are called the power line prefix and shall comprise ASCII alphanumeric characters. This provides certainty for distinguishing these prefix characters from control characters.

**3.2.1.1.2.1** *RCD poll command*

The RCD poll command requests the status buffer from the selected RCD. The data field sent to the MDCU from the MMU contains the routing byte followed by the 11 byte ASCII RCD network address, i.e.

Data field to MDCU = Routing byte/address

A valid reply from the MDCU contains the routing byte/address and the RCD status buffer in the data field of a general valid type reply (see **3.2.1.2.1**), i.e.

Data field from MDCU valid = Routing byte/address + RCD status values

The RCD status buffer contains the information defined below. Note that the status bytes used for the low data rate system only are prefixed with an L while the status bytes used for the high data rate system only are prefixed with an H.

First byte transmitted:

Status0 — RCD status is written during initialization of the RCD.

A bit which is set indicates that an error condition occurred during the particular test, as shown in Table 4. This is a read-only location.

**Table 4 — RCD status state bit definitions**

| Bit | RCD state |
|---|---|
| 7 | RCD type status:<br>set — Integrated RCD<br>reset — Stand-alone RCD |
| 6 | Datalog initialization incomplete |
| 5 | Reserved |
| 4<br>3<br>2<br>1<br>0 | RCD tests had an error:<br>Internal RAM error<br>External RAM error<br>Non-volatile memory error<br>Program check sum error |

Status1-2 — RCD software version (1) and subversion (2) status is a read-only location.

Status3 — Handshake byte, used by the host to reset internal RCD buffers and also to prevent data tearing when multiple transfers of a particular buffer are obtained. Each bit is discussed below.

Bit 7 — RCD external UART CTS status (controller communications for the iRCD and external portable data collection computer for the sRCD). Set active to a logic one when the UART is in the CTS mode, i.e. currently capable of transfers. This bit is a read-only bit, the MMU cannot set/reset it.

Bit 6 — RCD device status buffer status. Set active indicates that the buffer contains information available to the MMU. The MMU may reset this status by transmitting logic zero in this bit position, while a logic one will not change the buffer's state.

Bit 5 — RCD device response buffer status. Set active indicates that the buffer contains information available to the MMU. The MMU may reset this status by transmitting logic zero in this bit position, while a logic one will not change the buffer's state. Note that a buffer in the process of being loaded by the controller (iRCD) will not abort the transmission.

Bit 4 — RCD device datalog buffer status. Set active indicates that the buffer contains information available to the MMU. The MMU may reset this status by transmitting logic zero in this bit position, while a logic one will not change the buffer's state. Note that a buffer in the process of being loaded by the controller (iRCD) will not abort the transmission.

Bit 3 — RCD response buffer status. Set active indicates that the buffer contains information available to the MMU.

Bit 2 — For an sRCD only, RCD external portable data collection computer buffer status, if applicable. Set active to a logic one when either the external serial channel UART receive or the transmit buffer contains valid information. This information is not exchanged on the power line communications network. This bit can be reset by the MMU, but the serial channel communications are not affected.

Bit 1 — At present undefined.

Bit 0 — RCD data-logger buffer update handshake bit. The RCD sets this bit whenever the RCD has updated its internal data-logger buffer. The MMU resets the bit prior to transfer and checks it after the transfer is complete, thus providing a means to check possible data tearing of the buffer.

Status4-5 — Available power line buffer size high (4) and low (5) bytes.

LStatus6 — Indicates the maximum size for any data packet (or block) transmitted by the RCD on the power line network. This number will never exceed the available buffer size. Its default is set at 128 or 80H.

LStatus7 — Indicates the number of attempts the RCD will perform for a particular job in which it received an invalid response on the power line from the MDCU. The default is set at 02.

LStatus8 — Indicates the number of retries required on the power line. Rollover 0FFH to 00H will not occur. It can be reset using the RCD parameters command, as described in **3.2.1.3.2**.

Status9 — Indicates the current baud rate the iRCD is communicating with the controller. For the sRCD, this indicates the current baud rate the device is communicating with an external device, if applicable. The corresponding baud rates are given in Table 29.

Status10 — Indicates the maximum baud rate the iRCD can communicate with the controller. For the sRCD, this indicates the maximum baud rate the device can communicate with an external device, if applicable. The corresponding baud rates are given in Table 29.

Status11-14 — At present undefined.

**3.2.1.1.2.2** *RCD xmit1 and xmit2 commands*

The RCD xmit1 and xmit2 command transfers text data to a selected RCD slave. The xmit1 versus xmit2 selection defines the method of transfer on the power line and is further described for the appropriate low data rate or high data rate modem (**3.2.2** and **3.2.3** respectively). From the point of view of the MMU, these process differences are invisible, except for the time of response. The data field sent to the MDCU from the MMU contains the routing byte followed by the 11 byte ASCII RCD network address. The information following this address is the text data, or message, to be given to the selected RCD, i.e.

Data field to MDCU = Routing byte/address + RCD message

The RCD message field is discussed in **3.2.1.3**. Inscribed in it is a subcommand which the RCD evaluates and processes accordingly. If this subcommand requires a response from the RCD, the RCD buffers it in an RCD response buffer. An RCD interactive poll command (as described in **3.2.1.1.2.5**) is then used to obtain this buffered response. These xmit commands, however, only return a general valid type reply (see **3.2.1.2.1**) from the MDCU to the MMU after the MDCU receives the proper acknowledgement of transfer from the slave RCD. The routing byte/address are returned in the data field of this reply, i.e.

Data field from MDCU valid = Routing byte/address

**3.2.1.1.2.3** *RCD interactive1 and interactive2 commands*

The RCD interactive1 and interactive2 commands transfer text data to a selected RCD slave in exactly the same way as the RCD xmit commands described in **3.2.1.1.2.2**. As with the xmit commands, the interactive1 versus interactive2 selection defines the method of transfer on the power line and is further described for the appropriate low data rate or high data rate modem (**3.2.2** and **3.2.3** respectively). From the point of view of the MMU, these process differences are invisible, except for the time of response. The interactive commands provide the RCD with a subcommand inscribed in the text data, which the RCD evaluates and processes accordingly. (The RCD processes the interactive commands in the same way as the xmit commands.) If this subcommand requires a response from the RCD, the RCD buffers it in an RCD response buffer. The difference between the RCD xmit and interactive commands occurs here in the MDCU processing. Instead of returning a valid reply to the MMU when the MDCU receives a proper acknowledgement from the slave RCD for the transfer of text data, the MDCU queues an RCD interactive poll command, as described in **3.2.1.1.2.5**, without intervention from the MMU. A time delay later, as specified by a byte in the interactive command's data field, this interactive poll is sent to the RCD. Thus the data field sent to the MDCU is

Data field to MDCU = Routing byte/address + Response delay + RCD message

where the response delay is a 1 byte value. Each count of this value causes an approximate incremental delay, calculated as this number times the base value established in the MDCU Parameter11 byte described in **3.2.1.1.1.2**, between the completion of the interactive command and the transmission of the interactive poll. The response of the RCD to this poll is then returned to the MMU as the reply for the interactive command and, therefore, its format is as given in **3.2.1.1.2.5**. This reply will be dependent on the amount of RCD processing required for the RCD message subcommand. Note that subcommands not requiring RCD responses should not use an interactive command since the interactive poll would receive a negative acknowledgement from the RCD (i.e. no response buffer will be available).

**3.2.1.1.2.4** *RCD map command*

The RCD map command broadcasts a polling message from the MDCU requesting, with varying probabilities, an RCD device to return its network address. The purpose of this command is to log in, or map, any newly acquired or unpolled RCDs on the power line network. Since the mapping technique uses a polling-type process on the network, the format of the data field from the MMU to the MDCU is similar to the poll commands described in **3.2.1.1.2.1** and **3.2.1.1.2.6**, i.e. the routing byte followed by the 11 byte ASCII RCD network address. However, being a broadcast type of message transfer, it cannot simply select a particular RCD by using a unique individual network address. Instead, the 11 byte ASCII field is broken into ASCII subfields supplying all RCDs on the power line network with a mapping strategy. The format of this field is shown in Figure 6.

| | Alpha network address | Type repeat | Map subcommand | Unused HRCD destination address | Shift variable | Destination address individual |
|---|---|---|---|---|---|---|
| No. of bytes | 4 | 3 | 1 | 2 | 1 | 2 |

**Figure 6 — Format of RCD map command**

The fields in Figure 6 are defined as follows.

Alpha network address — 4 byte ASCII characters defining an RCD alpha portion (usually denoting the container operator) of the RCD network address on the power line network. All RCDs recognize the map command by using the universal address, or four ASCII 0s, or 30H 30H 30H 30H.

Type repeat — A 3 byte check field filled with a map command (045H). The device map command value is used instead of the RCD map command to allow an alphanumeric value to reside in the normally numeric polling field.

Map subcommand — A 1 byte ASCII character which provides an RCD with a processing procedure in the mapping strategy. These procedures, and their associated values, are given below.

Unused HRCD destination address — A 2 byte hexadecimal field which provides an HRCD with a network address. Only the HRCD(s) responding to the map command use this address. This field is used only when the alpha network address specifies the universal address, i.e. four ASCII 0s, or 30H 30H 30H 30H.

Destination address individual — A 2 byte hexadecimal field which provides an HRCD with a network address. Only the HRCD responding to the map command uses this address. This field is present only when the map address specifies an individual 11 byte address.

Shift variable — The upper nibble of the byte and shall be 1111 B. The lower nibble contains a number from 0 to 15 which indicates the number of bits to mask from an RCD's 16 bit random number (0 will provide a 1111111111111111 B bit mask, while 15 will provide 0000000000000001 B). An RCD which obtains a value of zero by performing the logical AND of this mask with the 16 bit random number will attempt to return its network address provided that its internal respond to map flag is set (see below). Thus, this shift variable provides the mapping strategy with the capabilities of adjusting the range, or size, of the expected number of RCD responding.

Any field described above which does not contain a value as required will ignore the command. Thus, the data field sent to the MDCU from the MMU is

Data field to MDCU = Network address tag/map information

An RCD will not accept a map command unless the alpha network address characters match those of the particular RCD or the universal address. This provides for selection of different classes of RCDs using the same data rate and thus the same routing byte. When accepted, the RCD decodes the map subcommand to perform different strategies of the mapping sequence. These subcommands are listed in Table 5. Any other value received by the RCD in the subcommand will force the RCD to ignore the command. Each RCD addressed will perform the map function received regardless of the respond to map state. Only after execution of the map function does the RCD check the respond to map state. If a map response is indicated, the RCD performs the logical AND of the shift variable mask with its 16 bit random number. A value of zero will allow the RCD to return its network address.

**Table 5 — Map subcommand definitions**

| Subcommand | ASCII value | HEX value | RCD process |
|---|---|---|---|
| Initiate new map | I | 49 | Sets its respond to map flag, generate a new 16 bit random, and set up a new shift variable. |
| General new map | G | 47 | Generate a new 16 bit random and set up a new shift variable. |
| Change shift map | S | 53 | Set up a new shift variable. |
| Decrement map | D | 44 | Decrement its random (variable rollover will occur) and set up a new shift. |

An RCD will set its internal respond to map flag in one of three ways:

a) the RCD starts a power-up sequence, either by initially being turned on, or by an internal deadman reset occurrence by an RCD directed reset command from the MMU device (see **3.2.1.3**);

b) the RCD has not received an individually addressed command within 1 h;

c) the RCD receives an initialize map subcommand (defined above).

The flag is reset each time the RCD receives an individually addressed command. At this time the RCD initiates the 1 h timer discussed in b) above.

When the MDCU receives a network address from the RCD it returns a general valid type reply (see **3.2.1.2.1**) to the MMU with the data field containing this RCD network address following the map command message it used, i.e.

Data field from MDCU value = Routing byte/map information + RCD network address

**3.2.1.1.2.5** *RCD interactive poll command*

The RCD interactive poll command is used to retrieve information from an RCD after the RCD has been sent an RCD xmit1 or xmit2 command (see **3.2.1.1.2.2**) in which its subcommand in the RCD message field caused the RCD to buffer a response (see **3.2.2.1**). The format of the data field from the MMU to the MDCU for the interactive poll is exactly the same as that of a normal poll command, i.e.

Data field to MDCU = Routing byte/address

The time between the xmit and interactive poll is the responsibility of the MMU, and shall be enough to allow the RCD to process and buffer its response. If an RCD response buffer does not contain information, a negative acknowledge (see **3.2.1.2.4**) is returned from the RCD to the MDCU, and consequently to the MMU. If, however, the RCD has buffered information, this information is returned to the MMU via a general valid type reply (see **3.2.1.2.1**) with this information preceded in the data field by the network tag and the RCD's address, i.e.

Data field from MDCU valid = Routing byte/address + RCD message

Once the RCD transfers this buffered data, it resets (and thus empties) its RCD response buffer, therefore allowing another RCD xmit or interactive command to be accepted by the RCD.

**3.2.1.1.2.6** *Device poll command*

The device poll command requests the quick status buffer of the controller which is attached to the selected iRCD or the internal quick status of the sRCD. The data field sent to the MDCU from the MMU contains the routing byte followed by the 11 byte ASCII RCD network address and the expected reply length, i.e.

Data field to MDCU = Routing byte/address

A valid reply from the MDCU contains the routing byte/address and the controller (iRCD) or internal (sRCD) quick status buffer in the data field of a general valid type reply (see **3.2.1.2.1**), i.e.

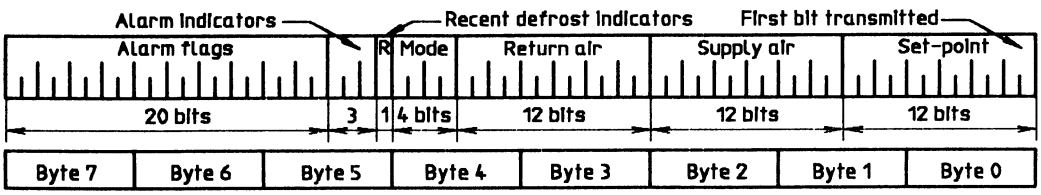
Data field to MDCU valid = Routing byte/address + Quick status

For an iRCD, an image of the controller quick status buffer is contained within the iRCD and updated approximately every 30 s. For both the iRCD and the sRCD, the reply format of the data field of this message is presented in Figure 7.

**3.2.1.1.2.7** *Device xmit1 and xmit2 commands*

The device xmit1 and xmit2 commands transfer text data to a selected RCD slave. The xmit1 versus xmit2 selection defines the method of transfer on the power line and is further described for the appropriate low data rate or high data rate modem (**3.2.2** and **3.2.3** respectively). From the point of view of the MMU, these process differences are invisible, except for the time of response. The data field sent to the MDCU from the MMU contains the routing byte followed by the 11 byte ASCII RCD network address. The information following this address is the text data, or message, to be given to the selected iRCD for transfer to the controller, or the selected sRCD for device input/output processing, i.e.

Data field to MDCU = Routing byte/address + Device message




| Contents | No. of bits | Resolution per bit | Range | Offset | Notes |
|---|---|---|---|---|---|
| Set-point<br>Supply air<br>Return air | 12 | 0,05 °C | −125 °C<br>to<br>75 °C | −125 °C | 0 ≤ −125 °C; FFF ≥ 79,75 °C |

| Contents | No. of bits | Indicated condition | Bit values |
|---|---|---|---|
| Mode | 4 (16 possible states) | Cool max. | 1 1 1 1 |
| | | Cool reduced | 1 1 1 0 |
| | | Modulation | 1 1 0 1 |
| | | Fan reduction | 1 1 0 0 |
| | | Heat reduced | 1 0 1 1 |
| | | Heat max. | 1 0 1 0 |
| | | Alarm shutdown | 1 0 0 0 |
| | | Power off | 0 0 0 0 |
| | | Humidity | |
| | | 7 states undefined | |
| R | 1 | Recent defrost | 1 |
| Alarm indicators | 3 | Change in primary alarms | 1 0 0 |
| | | Change in extended alarms | 0 1 0 |
| Alarm flags | 20 | Power up | 1 |
| | | Sensor failure | 1 |
| | | High evaporator temperature | 1 |
| | | High head | 1 |
| | | Evaporator fan motor | 1 |
| | | Condenser fan motor | 1 |
| | | Compressor | 1 |
| | | Defrost termin. on time limit | 1 |
| | | Temperature out of range | 1 |
| | | Low pressure | 1 |
| | | Low compressor oil pressure | 1 |
| | | Pump down termin. on time limit | 1 |
| | | Charge | 1 |
| | | Microprocessor faulty | 1 |
| | | Alarms reset | 1 |
| | | Fuse (control) | 1 |

First bit transmitted

NOTE All multiple byte fields shall be transmitted low-order byte first.

**Figure 7 — Device poll (total 8 bytes or 64 bits)**

This device message field is discussed in **3.2.1.5**. Inscribed in it is a subcommand which the RCD evaluates and processes accordingly. If this subcommand requires a response from the controller (iRCD) or the input/output function (sRCD), the RCD buffers it in a device response buffer. A device interactive poll command, as described in **3.2.1.1.2.10**, is then used to obtain this buffered response. These xmit commands, however, only return a general valid type reply (see **3.2.1.2.1**) from the MDCU to the MMU after the MDCU receives the proper acknowledgement of transfer from the RCD (this acknowledgement is an indication of the MDCU/RCD transfer and has no bearing on, for instance, the iRCD/controller transfer status). The routing byte/address are returned in the data field of this reply, i.e.

Data field to MDCU = Routing byte/address

**3.2.1.1.2.8** *Device interactive1 and interactive2 commands*

The device interactive1 and interactive2 commands transfer text data to a selected RCD slave in exactly the same way as the device xmit commands described in **3.2.1.1.2.7**. As with the xmit commands, the interactive1 versus interactive2 selection defines the method of transfer on the power line and is further described for the appropriate low data rate or high data rate modem (**3.2.2** and **3.2.3** respectively). From the point of view of the MMU, these process differences are invisible, except for the time of response. The interactive commands provide the RCD with a subcommand inscribed in the text data, which the RCD evaluates and processes accordingly. (The RCD processes the interactive commands in the same way as the xmit commands.) If this subcommand requires a response from the controller (iRCD) or the input/output function (sRCD), the RCD buffers it in an RCD response buffer. The difference between the device xmit and interactive commands, like the difference between the RCD commands described in **3.2.1.1.2.2** and **3.2.1.1.2.3**, occurs here in the MDCU processing. Instead of returning a valid reply to the MMU when the MDCU receives a proper acknowledgement from the RCD for the transfer of text data, the MDCU queues a device interactive poll command, as described in **3.2.1.1.2.10**, without intervention from the MMU. A time delay later, as specified by a byte in the interactive command's data field, this interactive poll is sent to the RCD. Thus the data field sent to the MDCU is

Data field from MDCU valid = Routing byte/address + Response delay + Device message

where the response delay is a 1 byte value. Each count of this value causes an approximate incremental delay, calculated as this number times the base value established in the MDCU Parameter11 byte described in **3.2.1.1.1.2**, between the completion of the interactive command and the transmission of the interactive poll. The response of the RCD to this poll is then returned to the MMU as the reply for the interactive command and, therefore, its format is as given in **3.2.1.1.2.5**. This reply will be dependent on the RCD processing required for the device message subcommand. Note that subcommands not requiring device responses should not use an interactive command since the interactive poll would receive a negative acknowledgement from the RCD (i.e. no response buffer will be available).

**3.2.1.1.2.9** *Device map command*

The device map command is indistinguishable from the RCD map command (see **3.2.1.1.2.4**).

**3.2.1.1.2.10** *Device interactive poll command*

The device interactive poll command is used to retrieve information from an RCD after the RCD has been sent a device xmit1 or xmit2 command (see **3.2.1.1.2.7**) in which its subcommand in the data field required the iRCD to buffer a controller response or the sRCD to buffer a device input/output process (see **3.2.1.5**). The format of the data field from the MMU to the MDCU for the interactive poll is exactly the same as that of a normal poll command, i.e.

Data field to MDCU = Routing byte/address

The time between the xmit and interactive poll is the responsibility of the MMU, and shall be enough to allow the RCD to process and buffer its response. If the RCD's device response buffer does not contain information, a negative acknowledge (see **3.2.1.2.4**) is returned from the RCD to the MDCU and consequently to the MMU. If, however, the RCD has buffered information, this information is returned to the MMU via a general valid type reply (see **3.2.1.2.1**) with this information preceded in the data field by the routing byte/address, i.e.

Data field to MDCU valid = Routing byte/address + Message

Once the RCD transfers this buffered data, it resets (and thus empties) its device response buffer, therefore allowing another device xmit or interactive to be accepted by the slave RCD. (Note that the RCD's device response buffer is independent of the RCD's RCD response buffer discussed in **3.2.1.1.2.5**.)

#### 3.2.1.2 *MDCU to MMU reply command types*

The reply types from the MDCU to the MMU convey the success/failure status of the jobs processed by the MDCU, and are presented in Table 6 and explained in **3.2.1.2.1** to **3.2.1.2.5**.

**Table 6 — MDCU reply types**

| MDCU reply | Value |
|---|---|
| General valid | 20H |
| MDCU reset | 27H |
| General invalid | 28H |
| Invalid NAK on power line | 29H |
| Invalid no response on power line | 2AH |

**3.2.1.2.1** *General valid reply*

The general valid reply from the MDCU to the MMU is used to indicate that the job, or command requested, as specified by the task number, was executed without error. Commands directed for MDCU interrogation will reply only with the MDCU status buffer. (Note that the format in which the MDCU reset command will return the status buffer is similar to that for the other MDCU interrogation commands with the exception that it will contain the default parameters, and that the reply type and task number will be as described in **3.2.1.2.2**.) The definition of this status buffer is given below. Note that the status bytes used for the low data rate system only are prefixed with an L while the status bytes used for the high data rate system only are prefixed with an H.

First byte transmitted:

Status0 — MDCU status is written during initialization of the MDCU. A bit which is set indicates that an error condition occurred during the particular test, as shown in Table 7. Since this is a read-only location, the error status cannot be reset by the MMU.

**Table 7 — MDCU status state bit definitions**

| Bit | MDCU state |
|---|---|
| 7-5 | Reserved |
| 4 | RCD tests had an error: |
| 3 | Internal RAM error |
| 2 | External RAM error |
| 1 | EEPROM (NOVRAM) error |
| 0 | Program check sum error |

Status1-2 — MDCU software version (1) and subversion (2) status is a read-only location.

Status3 — Handshake byte, used by the MMU to reset internal MDCU buffers and also to prevent data tearing when multiple transfers of a particular buffer are obtained. Each bit is discussed below.

Bit7 — MDCU UART CTS status (MMU communications). This bit will always be received by the MMU in the active, or logic one, state since the UART is in the CTS mode (i.e. communicating with the MMU). This bit is a read-only bit, the MMU cannot set/reset it.

Bit6-0 — At present undefined.

Status4-5 — Available power line buffer size high (4) and low (5) bytes.

LStatus6 — Indicates the maximum size for any data packet (or block) transmitted by the MDCU on the power line network. This number will never exceed the available buffer size. Its default is set at 128 or 80H.

LStatus7 — Indicates the number of attempts the MDCU will perform for a particular job in which it received an invalid response on the power line from the RCD. The default is set at 02.

LStatus8 — Indicates the number of retries required on the power line. Rollover 0FFH to 00H will not occur. It can be reset using the MDCU parameters command, as described in **3.2.1.1.1.2**.

LStatus9 — Indicates the number of successful normal polls which required one retry on the power line. Rollover 0FFH to 00H will not occur. It can be reset using the MDCU parameters command, as described in **3.2.1.1.1.2**.

LStatus10 — Indicates the number of successful normal polls which required two retries on the power line. Rollover 0FFH to 00H will not occur. It can be reset using the MDCU parameters command, as described in **3.2.1.1.1.2**.

Status11 — Indicates the time delay base value used within an interactive command (**3.2.1.1.2.3** and **3.2.1.1.2.8**) that is required between the MDCU transmission of data to an RCD and the interactive poll used to obtain a response from the RCD. Each count provides a 0,1 s incremental delay.

Status12-14 — At present undefined.

Commands directed for power line communications reply with data fields which are dependent on the specific command and, therefore, are discussed within the subclause relevant to each specific command.

**3.2.1.2.2** *MDCU reset reply*

The MDCU reset reply is a special case of the general valid reply for MDCU interrogated commands since it also returns the MDCU status buffer as described in **3.2.1.2.1**. The status buffer, however, will contain the MDCU default parameters since the MDCU has undergone a hardware reset. The task number returned will also be the default reset task number, or 255 (0FFH). (Note, therefore, that since the MDCU may be reset without MMU intervention via its deadman feature, this task number should be reserved for this function only, and not used by MMU/MDCU interactions.)

**3.2.1.2.3** *General invalid reply*

The general invalid reply from the MDCU to the MMU is used to indicate that the job, or command requested, as specified by the task number, could not be completed. Unlike the special negative replies used to indicate incomplete jobs on the power line (as described in **3.2.1.2.4** and **3.2.1.2.5**), this reply is used for a non-specific general purpose negative reply. Examples of its use are as follows:

a) unrecognized command type received by the MDCU;

b) invalid packet length received by the MDCU;

c) no response from a mapping request. This is used instead of a no response reply to distinguish a map from other processing. The map procedure has a much higher probability of collisions on the power line and since a collision is not distinguished from a no response, the no response is not used.

As with all negative replies, no data field is associated with the reply.

**3.2.1.2.4** *Invalid NAK on power line reply*

The invalid NAK reply indicates the unsuccessful completion of a job, or command requested, as specified by the task number. This job was intended for power line interactions. The attempts to complete the job failed such that, although the RCD responded to the MDCU at least once (retries included), invalid information was received by the MDCU. In most cases the MDCU received negative acknowledges indicating that the RCD did not have valid information to return to the MDCU or that an RCD buffer was not currently available to receive the information from the MDCU. As with all negative replies, no data field is associated with the reply.

**3.2.1.2.5** *Invalid no response on power line reply*

The invalid no response reply indicates the unsuccessful completion of a job, or command requested, as specified by the task number. This job was intended for power line interactions. The attempts to complete the job failed such that no response from the RCD was received by the MDCU even though all retries were attempted. The timeout period for each command type is dependent on which data rate modem is being accessed on the power line and is, therefore, described for the appropriate low data rate or high data rate modem (**3.2.2** and **3.2.3** respectively). In most cases this indicates the absence of the addressed RCD on the power line. As with all negative replies, no data field is associated with the reply.

**3.2.1.3** ***RCD directed power line commands***

This group of commands requires communications with RCD devices over the power line communications network. These commands are buried in the data field of an MMU/MDCU message and are deciphered by the RCD. The format of this portion of the MMU/MDCU data field is referred to as the RCD message. The first byte of the RCD message field is the RCD command. Those commands at present supported by the RCD are given in Table 8, and each is described below in its respective subclause. Table 8 also indicates whether or not the RCD will buffer data in its RCD response buffer on acceptance of each command. The bytes which follow the command in the message, if required, are discussed for each command in the respective subclause. Both types of RCD, the iRCD and the sRCD, process these RCD directed commands similarly.

**Table 8 — RCD directed command types**

| RCD command | Value | Buffering |
|---|---|---|
| RCD reset | 20H | No |
| RCD parameters write | 22H | Yes |
| RCD address/set-point read | 24H | Yes |
| RCD address write | 26H | Yes |
| RCD set-point write | 28H | Yes |
| RCD RAM absolute address read | 30H | Yes |
| RCD RAM volatile offset address read | 32H | Yes |
| RCD RAM battery offset address read | 34H | Yes |
| RCD RAM datalog offset address read | 36H | Yes |
| RCD RAM absolute address write | 40H | Yes |
| RCD RAM offset address write | 42H | Yes |
| RCD RAM battery offset address write | 44H | Yes |
| RCD RAM datalog offset address write | 46H | Yes |

Since processing of the RCD message occurs after accepting the information from the MDCU over the power line, the RCD acknowledge, or acceptance, of the network command occurs at the network level and not at the RCD message level. For example, an RCD may accept a command from the MDCU with an invalid RCD message type for the RCD. Any processing of a message after such acceptance from the MDCU, even for a valid message type which the RCD finds illegal, will be ignored by the RCD. An RCD will not accept a valid RCD network command only if its RCD response buffer already contains data waiting to be returned to the MDCU from a previous RCD network command.

**3.2.1.3.1** *RCD reset command*

The RCD reset command shall cause the RCD to initialize itself in the same way as if a power-up sequence was performed. The RCD will perform its internal diagnostics and initialize with default parameters. Any data associated with this command will be ignored. Since the RCD is reset, no data response is buffered in the RCD response buffer.

**3.2.1.3.2** *RCD parameters command*

The RCD parameters read command loads the RCD status buffer with information passed in the message following this parameter's command byte. Since the RCD status buffer is only 15 bytes in length, a message containing more than 15 bytes of information will cause the RCD to ignore the command. The parameters to be loaded are defined below. Note that the parameters used for the low data rate system only are prefixed with an L while the parameters used for the high data rate system only are prefixed with an H.

First byte transmitted:

Parameter0 — Test status byte is a read-only location and, therefore, this parameter is ignored by the RCD.

Parameter1-2 — RCD software version/subversion is a read-only location and, therefore, this parameter is ignored by the RCD.

Parameter3 — Change the MMU handshaking flag byte. This byte is used by the MMU to reset internal RCD buffers and also to prevent data tearing when multiple transfers of a particular buffer are obtained. Each bit is discussed below.

Bit7 — For an iRCD only, the UART CTS status (controller communications). Set active to a logic one when the UART is in the CTS mode that is currently capable of transfers. This bit is a read-only bit, the MMU cannot set/reset it.

Bit6 — RCD device status buffer status. Set active indicates that the buffer contains information available to the MMU. The MMU may reset this status by transmitting logic zero in this bit position, while a logic one will not change the buffer's state.

Bit5 — RCD device response buffer status. Set active indicates that the buffer contains information available to the MMU. The MMU may reset this status by transmitting logic zero in this bit position, while a logic one will not change the buffer's state. Note that for an iRCD, a buffer in the process of being loaded by the controller will not abort the transmission.

Bit4 — RCD device datalog buffer status. Set active indicates that the buffer contains information available to the MMU. The MMU may reset this status by transmitting logic zero in this bit position, while a logic one will not change the buffer's state. Note that for an iRCD, a buffer in the process of being loaded by the controller will not abort the transmission.

Bit3 — RCD response buffer status. Set active indicates that the buffer contains information available to the MMU. This bit cannot be reset by this command since this command will actually buffer this status here.

Bit2 — For an sRCD only, RCD external portable data collection computer buffer status, if applicable. Set active to a logic one when either the external serial channel UART receive or the transmit buffer contains valid information. This information is not exchanged on the power line communications network. This bit can be reset by the MMU, but the serial channel communications are not affected.

Bit1 — At present undefined.

Bit0 — RCD data-logger buffer update handshake bit. The RCD sets this bit whenever the RCD has updated its internal data-logger buffet. The MMU resets the bit prior to transfer and checks it after the transfer is complete, thus providing a means to check possible data tearing of the buffer.

Parameter4-5 — Change available buffer size high (4) and low (5) bytes. Decreases/expands power line buffers. Zero indicates no change. If the new size is below/above the RCD's minimum/maximum value, its minimum/maximum value is used.

LParameter6 — Change the maximum size of the data packets (or blocks) to be transferred on the power line. If received larger than the available buffer size setting (described above), that value will be substituted. A value of zero will retain the previous setting (unless that setting is greater than the new available buffer size). It is recommended that the packet size not be set greater than its default value (128 or 80H) unless the message timeout settings within the RCD are changed accordingly.

LParameter7 — Change the number of retransmissions to be attempted for a given job on the power line.

LParameter8 — Change the counter indicating the number of retries which have been performed on the power line.

LParameter9 — Change the current baud rate at which the iRCD is communicating with the controller. If this requested rate is greater than the iRCD maximum rate given in LParameter10, this maximum rate is substituted. The corresponding baud rates are given in Table 29. For an sRCD communicating with an external device (if applicable), this is a read-only location.

LParameter10 — Maximum baud rate the iRCD can communicate with the controller or the sRCD can communicate with an external device, if applicable. The corresponding baud rates are given in Table 29. This is a read-only location.

Parameter11-14 — At present undefined. The information will be copied into the status buffer as received.

Since the first 3 bytes of the status buffer are read-only locations, any data field containing less than 3 bytes will not change any parameters in the RCD status buffer. Also, Parameter4-5 is 2 bytes in length and a byte count which transfers only one of these bytes will not update this parameter. Regardless of whether any parameters were changed, once the RCD has accepted to process the information, the RCD will place the new status buffer into the RCD response buffer following the RCD status change reply byte (see **3.2.1.4.1**).

**3.2.1.3.3** *RCD address/set-point read command*

The RCD address/set-point read command loads the RCD response buffer with the RCD address/set-point read reply byte (see **3.2.1.4.2**) followed by the present contents of the RCD's non-volatile memory device, which contains the 11 byte RCD network address plus the 4 byte RCD set-point (**3.2.1.3.5**). Note that if an invalid address is resident in the RCD's non-volatile memory device (i.e. this information's check sum is found to be erroneous), the universal error address of nine ASCII 0s followed by two ASCII 1s, or nine 30Hs plus two 31Hs, will be buffered. Since information is not required for this read command, any data received by the RCD with this command will be ignored.

**3.2.1.3.4** *RCD address change command*

The RCD address change command changes the assigned address of the particular RCD with information passed in the message following these address change command bytes. Since an RCD network address is 11 ASCII bytes in length, a message containing more than 11 bytes of information, or information received that is not ASCII numerics/alphanumerics, will cause the RCD to ignore the command. The RCD will also ignore the command if it has been addressed with the universal address and the initial address enable device jumper is removed.

Regardless of whether any bytes within the address field were changed, once the RCD has accepted to process the information, the RCD will place the new 11 byte network address into the RCD response buffer following the RCD address change reply byte (see **3.2.1.4.3**). Note that the interactive commands should not be used when changing an RCD address while addressing that RCD with its individual address since the interactive poll command queued by the MDCU to obtain the RCD response buffer will contain the old or newly invalid RCD address.

**3.2.1.3.5** *RCD set-point change command*

The RCD set-point change command changes the assigned set-point values of the particular RCD with information passed in the message following this set-point change command byte. This set-point information consists of 4 bytes which are stored in a non-volatile memory device. At present this information is not defined and, therefore, not used by the RCD. Since there are 4 bytes, only an information field of between 1 byte and 4 bytes will be processed by the RCD. Any other number of bytes received will cause the RCD to ignore the command.

Regardless of whether any bytes within the set-point field were changed, once the RCD has accepted to process the information, the RCD will place the number of set-point bytes transferred into the RCD response buffer following the RCD set-point change reply byte (see **3.2.1.4.4**).

**3.2.1.3.6** *RCD RAM read commands*

The RCD read commands place a specified number of bytes in the RCD response buffer. Either an absolute or an offset starting address of these data is also specified. Starting addresses are calculated for each of these commands specifically as an offset from the first location of its designated buffer. Absolute addressing may be considered as a calculated offset address from a 0000H base location. The assignment of addresses is hardware specific and is outside the scope of this International Standard.

Access to the data-logger buffer while it is being updated is not allowed. The RCD will not transfer the datalog information to its internal RCD response buffer. Therefore, a NAK will be returned to the MDCU when trying to retrieve this information. Since the MMU communication and data-logging updates are independent one or more blocks of data may be affected during an update. The transfer lockout ensures that if the MMU obtains a contiguous block of the data-logger this will not result in data tearing. However, if the MMU must obtain various portions of the data-logger, another lockout is required. A bit in the RCD status buffer (see **3.2.1.3.2** and **3.2.1.4.1**) is used, in conjunction with the MMU, to provide an interlock. The MMU should reset this bit prior to obtaining multiple block area transfers and, when the transfers are completed, should check that it has remained unchanged. The RCD will set this bit when it updates any portion, or multiple portions, of the data-logger. An update which requires data in distinctly different areas of the data-logger (i.e. block 0 pointers and the block affected by those pointers) can be accessed with confidence by the MMU.

The RCD does not perform address checking on any given location and, therefore, it is the responsibility of the MMU to provide an existing area of memory. (Note that support peripherals on the RCD, such as an external UART, which is memory mapped, are also accessible.) The starting address and byte count are 16 bit parameters, passed in that order, following the RCD read commands, i.e.

Message field to RCD = Command + Start address + Byte count

where the 16 bit parameters are passed high byte followed by low byte. The RCD will not accept the command if the number of bytes in the messages does not equal five or the total length requested by the byte count is greater than the RCD response buffer.

Once the RCD has accepted to process the information, the RCD will place the number of bytes requested into the RCD response buffer following the RCD RAM read reply byte. the starting address parameter and the byte count (see **3.2.1.4.5**).

**3.2.1.3.7** *RCD RAM write commands*

The RCD write commands place a specified number of bytes in RCD memory. Either the absolute or the offset starting location for placing these data is also specified. Starting addresses are calculated for each of these commands specifically as an offset from the first location of its designated buffer. Residences of these buffers are hardware specific and, therefore, outside the scope of this International Standard. All of these write commands except, possibly, some defined non-volatile memory, have the potential of destroying internal RCD processing procedures and, therefore, are not allowed unless the initial address enable device jumper is inserted. Since most of these writes are performed with the initial address enable device jumper inserted, the RCD does not perform address checking on any of these given locations and, therefore, it is the responsibility of the MMU to provide an existing area of memory. (Note that support peripherals on the RCD, such as external UART, which may be memory mapped, are also accessible.) The non-volatile memory write which may be reserved to store input data directly from the MMU, however, may be performed during system execution and thus its range of address to be written must be checked. If any byte to be transferred is outside this usable external range of memory, the command is aborted.

The starting address and byte count are 16 bit parameters, passed in that order, following the RAM write command. The data to be written follows this information, i.e.

Message field to RCD = Command + Start address + Byte count + Data

where the 16 bit parameters are passed high byte followed by low byte. The RCD will not accept the command if the number of bytes in the message is not greater than five. Also, the RCD will not accept the command if the byte count does not match the number of data bytes received. Since the power line buffers are variable, dependent on their setting in their status buffers (for example, see **3.2.1.3.2**), the size of the message is limited.

Once the RCD has accepted to process information, the RCD will place the RCD RAM write reply byte, along with the starting address parameter and the byte count received, in the RCD response buffer to inform the MMU that the transfer has been accepted (see **3.2.1.4.6**),

**3.2.1.4** ***RCD directed power line slave replies***

The RCD message reply types from the RCD to the MDCU convey the successful status of a previously accepted RCD message command received by the RCD. This reply has been buffered in the RCD response buffer and must be retrieved before the RCD can accept another RCD message. The buffer is retrieved using an RCD interactive poll command (see **3.2.1.1.2.5**), either exclusively from the MMU, or inclusively via an RCD interactive1 or RCD interactive2 command (see **3.2.1.1.2.3**). The RCD returns the buffered data in text block packets over the power line communications network to the MDCU. The first byte of this RCD message field is the RCD reply type. The reply types currently supported by the RCD are given in Table 9, and each is described below in its respective subclause. Note that the reply has the same value as its corresponding command presented in **3.2.1.3** with the least significant bit set. The bytes which follow the reply type in the message are discussed for each reply in the respective subclause.

**Table 9 — RCD directed slave reply types**

| RCD reply | Value |
|---|---|
| RCD status change | 23H |
| RCD address/set-point read | 25H |
| RCD address change | 27H |
| RCD set-point change | 29H |
| RCD RAM absolute address read | 31H |
| RCD RAM offset address read | 33H |
| RCD RAM battery offset address read | 35H |
| RCD RAM datalog offset address read | 37H |
| RCD RAM absolute address write | 41H |
| RCD RAM offset address write | 43H |
| RCD RAM battery offset address write | 45H |
| RCD RAM datalog offset address write | 47H |

On receipt of the RCD text message from the RCD, the MDCU will forward the information to the MMU. This is returned in the data field of a general valid reply (see **3.2.1.2.1**), preceded by the RCD routing byte/address of the RCD interactive poll command. The full data field to the MMU is, therefore,

Data field to MMU = Routing byte/address + Reply command + RCD message data

The individual replies discussed below, however, will be in terms of the RCD message field, which is in relation to the RCD/MDCU transfer. The information received at the MMU will have the routing byte/address added.

**3.2.1.4.1** *RCD status change reply*

The information returned to the MDCU from the ROD following an ROD parameters command (see **3.2.1.3.2**) contains the RCD status. This information is transferred from the RCD status buffer to the RCD response buffer immediately after the parameters are changed in the status buffer. Therefore, the delay in retrieving this reply is proportional to the age of the valid data (but obviously this relates only to those bytes being updated in RCD processing). Except for this aging of buffered data, the information returned is the same as that returned by the RCD for a network RCD poll command (see **3.2.1.1.2.1**), i.e. the RCD status buffer definitions. One other slight difference between the two responses is that this reply precedes the status buffer information with the reply type byte, i.e.

Message field to MDCU = Reply command + RCD status

**3.2.1.4.2** *RCD address/set-point read reply*

The information returned to the MDCU from the RCD following an RCD address/set-point read command (see **3.2.1.3.3**) contains the 11 byte RCD network address followed by the RCD set-point bytes, all of which are stored in an RCD non-volatile memory device. This information is transferred to the RCD response buffer immediately after the command is accepted. This 15 byte ASCII information is returned to the MDCU preceded by the reply type byte, i.e.

Message field to MDCU = Reply command + RCD network address + RCD set-point

**3.2.1.4.3** *RCD address change reply*

The information returned to the MDCU from the RCD following an RCD address change command (see **3.2.1.3.4**) contains the newly accepted RCD network address. This information is transferred from the RCD non-volatile memory device to the RCD response buffer immediately after the non-volatile memory device has been updated. This 11 byte ASCII information is returned to the MDCU preceded by the reply type byte, i.e.

Message field to MDCU = Reply command + RCD's new network address

**3.2.1.4.4** *RCD set-point change reply*

The information returned to the MDCU from the RCD following an RCD set-point change command (see **3.2.1.3.5**) contains the newly accepted set-point byte, or bytes. This information is transferred from the RCD non-volatile memory device to the RCD response buffer immediately after the non-volatile memory device has been updated. The 1 byte to 4 bytes of information are returned to the MDCU preceded by the reply type byte, i.e.

Message field to MDCU = Reply command + RCD's new set-point data

**3.2.1.4.5** *RCD RAM read replies*

The information returned to the MDCU from the RCD following an RCD RAM read command (see **3.2.1.3.6**) contains the reply type byte, the starting or offset address and byte count received in the request, and the corresponding number of bytes of text data requested. The starting address and byte count are both 16 bit parameters and are returned as verification that the text data requested is that returned. Thus, the format of the RCD reply is

Message Field to MDCU = Reply command + Starting address + Byte count + Data

where the starting or offset address and byte count are transferred high byte followed by low byte, and the number of data bytes is given by the byte count.

**3.2.1.4.6** *RCD RAM write replies*

The information returned to the MDCU from the RCD following an RCD RAM write command (see **3.2.1.3.7**) contains the reply type byte and the starting or offset address and byte count received in the request. The starting address and byte count are both 16 bit parameters, transferred high byte followed by low byte. Although the write command does not require the RCD to return actual text data, this reply is nevertheless used as a verification of the next data written into RCD memory. Thus, the format of the RCD reply is

Message field to MDCU = Reply command + Starting address + Byte count

**3.2.1.5** ***Device directed power line commands***

This group of commands, like the RCD directed commands discussed in **3.2.1.3**, requires communications with RCD devices over the power line communications network. Similarly, these commands are buried in the date field of an MDCU/MMU message and are deciphered by the RCD. Processing of the commands differs slightly, however, since this set of commands is used to interact directly with an iRCD's controller or the sRCD's input/output functions. Therefore, the format of this portion of the MDCU/MMU data field is referred to as a device message. The device message structure is given as

Device message to MDCU = Return byte count + Device command + Data field

where the return byte count is a 2 byte value (high byte first) which provides the RCD with the number of bytes the MMU will expect in the reply. This value is equivalent to the reply type byte count (which is one) plus the number of data bytes in the reply message.

The return byte count is passed in the device message so that an iRCD does not have to decipher fully the contents of the device message, i.e. this message is intended specifically for the controller. By providing this count, the iRCD interacts correctly with the controller with only a general knowledge of the overall RCD to controller protocol (**3.2.4**). The iRCD provides the SYNC characters preceding the Device command + Data field and follows this with the proper CRC characters of the protocol. It also checks, with simple masking, that the proper reply type is received for a given transmitted device command. The particular device command and its associated data field need not be deciphered.

To imitate the iRCD/controller's format, the message structure of the sRCD will be the same. Therefore, the device message will also contain the same information. Since the sRCD may only support a limited number of controller functions, those functions not supported by an sRCD are simply not utilized (an sRCD will ignore any received field which is not supported and return zeros in any field in a return message which is not utilized). Since processing is internal to the sRCD and does not require a communications medium to a controller, the sRCD also checks that the return byte count is the expected number of bytes to be returned in the reply message. If not, the sRCD should ignore the device command. The sRCD will process the device commands which it supports as does the controller (see **3.4**).

Since processing of the device message occurs after accepting the information from the MDCU over the power line, the RCD acknowledge, or acceptance, of the network command occurs at the network level and not at the device message level. For example, an RCD may accept a command from the MDCU with an invalid device message type for the RCD. Any processing of a message after such acceptance from the MDCU, even for a valid message type which the RCD finds illegal, will be ignored by the RCD.

An RCD will not accept a valid device network command only if its device response buffer already contains data waiting to be returned to the MDCU from a previous network command.

**3.2.1.6** ***Device directed power line slave replies***

The device message reply types from the RCD to the MDCU convey the successful status of a previously accepted device message command received by the RCD. This reply has been buffered in the device response buffer and must be retrieved before the RCD can accept another device message. (Note that this buffer can be reset using the RCD parameters command, see **3.2.1.3.2**.) The buffer is retrieved using a device interactive poll command (see **3.2.1.1.2.10**), either exclusively from the MMU, or inclusively via a device interactive1 or device interactive2 command (see **3.2.1.1.2.8**). The RCD returns the buffered data in text block packets over the power line communications network to the MDCU.

The first byte of this device message field is the device reply type. It should be noted that the reply has the same value as its corresponding command with the most significant bit (7) set. The bytes which follow this reply type constitute the data field of the reply message and are defined in **3.4**.

On receipt of the device message text from the RCD, the MDCU will forward the information to the MMU. This is returned in the data field of a general valid reply (see **3.2.1.2.1**), preceded by the RCD routing byte/address of the device interactive poll command. The full data field to the MMU is, therefore,

Data field to MMU = Routing byte/address + Reply command + Data field

### 3.2.2 MDCU to LRCD communications

The physical requirements of LDCU communications with the slave LRCDs are described in **3.5**. This serial communications system consists of a single control station, called an LDCU, which initiates all transactions (initial master device) to interrogate a multiple tributary stations environment (slave devices). The tributary stations are called LRCDs. Control of mastership is transferred to a selected tributary device when the control station requests information. The information is returned to the control station only, i.e. no tributary-to-tributary communications are allowed. Completion of the transfer returns control of mastership back to the LDCU control station.

#### 3.2.2.1 *LDCU power line communications components*

There are four basic components available to the transmission function on the power line. These include the preamble, the communications control characters, the prefix and the data. All transmissions shall be initiated by the preamble and shall contain some type of control character(s). A prefix or data field may or may not be present in a transmission.

##### 3.2.2.1.1 *Preamble*

The preamble delimits the start of a valid message. It consists of a variable number (minimum of three) of transitions followed by 10 or more logic one bits and is required to synchronize communications between devices on the power line. This provides a unique sequence of bits which is illegal for normal information transfers since each byte transferred contains a start bit (low logic level) followed by a maximum of nine logic one bits. The occurrence of this sequence must force resynchronization (at any instance within a transfer) of receiving slave devices on the power line. Thus, the deadtime between any two bytes in a transfer is restricted to less than one-half bit length.

##### 3.2.2.1.2 *Communications control characters*

Ten ASCII communications control characters are reserved to provide control functions during communications. These characters are given in Table 10 and described in **3.2.2.1.2.1** to **3.2.2.1.2.10**. Those characters indicated in Table 10 by an asterisk (*) provide a control function only when preceded by a DLE control character (termed a control character sequence).

**Table 10 — Network control characters**

| Control character | HEX value | Definition |
|---|---|---|
| SOH* | 01 | Start of heading |
| STX* | 02 | Start of text |
| ETX* | 03 | End of text |
| EOT | 04 | End of transmission |
| ETB* | 17 | End of transmission block (packet) in a multiple data block transfer |
| ENQ | 05 | Enquiry |
| ACK* | 30 or 31 | Acknowledgements:<br>30H = ACK0<br>31H = ACK1 |
| NAK | 15 | Negative acknowledge |
| SYNC | 16 | Synchronous idle |
| DLE | 10 | Data link escape |

**3.2.2.1.2.1** *SOH (Start of heading)*

The DLE SOH control character sequence is transmitted by the master device at the beginning of the data portion of a transfer to delimit the start of a message heading. This heading consists of addressing or routing information. The heading may or may not be present for a data portion of a transfer.

**3.2.2.1.2.2** *STX (Start of text)*

There are two ways in which the DLE STX control character sequence is used. The first condition is when the optional header and its corresponding DLE SOH control character sequence are used. In this case the function of the DLE STX control character sequence is used as a delimiter which terminates the header information field and signals the beginning of the test information portion of the data transfer. (See **3.2.2.1.2.1**.)

The second condition occurs when the optional header is not present. In this case the DLE STX simply signals the beginning of the text information portion of the data transfer. In both cases the DLE STX control character sequence is always used to indicate the start of the transfer of text information in either a single or multiple block transfer.

The text portion of a transfer is a sequence of characters which is to be treated as an entity and is transmitted in its entirety to the ultimate destination. The DLE STX must always be present in the data portion of a text transfer, although it may not appear in the first block(s) when header information is included in a multiple block transfer.

**3.2.2.1.2.3** *ETX (End of text)*

The DLE ETX control character sequence is transmitted by the master device at the end of the data portion of a transfer to terminate the transfer of information which contains one or more blocks of heading, if present, and text. A block of information which transfers a single heading and text block, or a single text block (i.e. all of the information is transferred in a single block), is terminated by this control sequence.

**3.2.2.1.2.4** *EOT (End of transmission)*

The EOT control character is transmitted by the master device (or by the LDCU control station to re-establish control of the power line network) to terminate a transmission that may have contained one or more blocks of texts and any associated heading. The EOT must immediately follow a preamble, i.e. no prefix or other control may precede it. Detection of the EOT control requires all tributary stations to return to their receive mode, thus allowing the LDCU control station to become the master, or only device capable of initiating a transmission.

**3.2.2.1.2.5** *ETB (End of transmission block)*

The DLE ETB control character sequence is transmitted by the master device at the end of the data portion of a transfer to terminate a sequence of characters for any block except the last of a multiple block transfer. This block transfer started with the DLE SOH or DLE STX control character sequence.

**3.2.2.1.2.6** *ENQ (Enquiry)*

The ENQ control character is transmittted by the LDCU control station to initiate a polling session to a tributary station. It is also transmitted by a master device to request a slave device receiving a data transfer to retransmit its previous ACK/NAK sequence. In these cases the ENQ control character is preceded by the prefix. A DLE ENQ control character sequence within a data block portion of a transfer signifies to the receiving slave device that the transmitting master device is aborting the data heading/text transfer. In this instance, the receiving slave device should discard the block and return a NAK response to the aborted data block transfer.

**3.2.2.1.2.7** *ACK (Acknowledgement)*

The DLE ACK control character sequence is transmittted by a slave device to acknowledge the acceptance of the tributary station addressed by the prefix/ENQ control character sequence within a selection with a response handshaking procedure (see **3.2.2.2**). The tributary station will acknowledge with an ACK0 (DLE followed by a 30H). A DLE ACK control character sequence is also transmitted by a slave device to acknowledge the receipt of a block of information in the data portion of a transfer. The first block is acknowledged with an ACK1 (DLE followed by a 31H), while successive blocks of heading/text data are acknowledged with subsequently alternating ACK0 and ACK1. In all cases, the DLE ACK control character sequence is preceded by the prefix.

**3.2.2.1.2.8** *NAK (Negative acknowledge)*

The NAK control character is transmitted by a slave device to acknowledge negatively the receipt of a block of information in the data portion of a transfer. The transmitting device of the block transfer (and thus the receiving device of this NAK control sequence) may either retransmit the previous block of information or terminate the transfer with an EOT control sequence. The NAK control character is always preceded by the prefix.

**3.2.2.1.2.9** *SYNC (Synchronous idle)*

The SYNC control character is reserved as a "time-fill" character during periods in a synchronous communications system when no other characters are available for transmission. Since the power line communications medium is asynchronous, this control character is not required and should cause the receiving slave device to abort reception of the present information packet.

**3.2.2.1.2.10** *DLE (Data link escape)*

The DLE control character is used to distinguish a small number of characters as either control or data. A DLE character must precede an SOH, STX, ETX, ETB and ACK character to allow these characters to perform as control characters. A DLE character preceding another DLE character indicates a single DLE data character. Thus, a single DLE character in a sequence of characters indicates that the character to follow provides a control function, while two DLE characters in sequence requires the removal of one in the data block and the use of the second as a data character. The DLE DLE sequence will only be found in the data portion of a transfer and is termed "DLE stuffing". Thus, a single DLE character followed by an indistinguishable control character in a data portion of a transfer must cause the receiving device to abort the reception of the data.

**3.2.2.1.3** *Prefix*

The prefix is a string of 13 ASCII alphanumeric characters (this ensures that these prefix characters can be distinguished from the communications control characters specified in **3.2.2.1.2**). The prefix is arranged in the following format:

| Message type | Routing byte | Network address |

The prefix is always followed by a control character or a control character sequence. The prefix fields are defined as follows.

Message type — ASCII select control byte. Directs which type of transfer is being requested by the LDCU control station (see Table 12).

Routing byte — ASCII routing byte. Selects a particular LDCU requested modification on the communications medium which the responding LRCD will use. The definition of the bits for the low data rate network is given in Table 11.

**Table 11 — Low data rate routing byte definitions**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Modem type | | Standard specific | | | | | |
| 01 | | 0 = ISO | 00 = 1 200 baud<br>01 = Reserved<br>10 = Reserved<br>11 = Maximum burst | | Burst rate capability of the LRCD taken from Table 29 with offset of + 4 | | |
| 01 | | 1 = MFG | Manufacturer specific modifications | | | | |

Network address — 11 byte ASCII characters defining the address of (a) tributary station(s) on the power line network. The bytes are categorized as the container ISO number with a 4 byte alpha field followed by two 3 byte numeric fields (numeric1 and numeric2) and a 1 byte check digit. A universal address consisting of all ASCII 0s, or all 30H, is recognized by all LRCDs (i.e. a broadcast address) while a field containing nine ASCII 0s followed by an ASCII 1 and another ASCII 0, or nine 30Hs plus 31H plus 30H, is an address substituted in an LRCD containing an invalid personal address.

Response to the initial transfer establishment requiring a prefix (such as ACK or NAK) will return the same information.

**3.2.2.1.4** *Data*

The data portion of a transfer consists of the information to be exchanged between two devices (master to slave) on the power line network. Control character sequences are used to initiate and terminate various portions of the information transfer. A transfer of information may be subdivided into multiple blocks, or packets, and transferred. The structure of a block is as shown in Figure 8.

| Start of header | Header | Start of text | Text | End of data | CRC |
|---|---|---|---|---|---|

**Figure 8 — Block structure**

The data fields in Figure 8 are defined as follows.

Start of header — A DLE SOH control character sequence which is present if a header field is to be included in the data block.

Header — Variable length field consisting of addressing or routing information. This information is transferred in raw data form and requires DLE stuffing. A heading may or may not be present. A header must be followed by text data in a transfer. Note, however, that (a) block(s) in a multiple block transfer may contain only header information, but must be followed by text data information.

Start of text — A DLE STX control character sequence which is present if a text data field is to be included in the data block. If only a header is transferred in a particular block of a multiple block transfer, this field will not be present.

Text — Variable length field consisting of the information required to be transferred from the master device to a slave device on the power line network. This information is transferred in raw data form and requires DLE stuffing. The contents are dependent on the type of message to be processed. The contents of this field are described in **3.2.1**.

End of data — A DLE ETX or DLE ETB control character sequence used to terminate the information portion of a data transfer. The DLE ETB is used to terminate a data sequence of characters for any block except the last of a multiple block transfer. The DLE ETX is used to terminate the transfer of information which contains one or more blocks of heading, if present, and text. This includes a block of information which transfers a single heading and text block, or a single text block (i.e. all the information is transferred in a single block).

CRC — A 16 bit error check field generated by using the CRC-16 polynomial:

$$x^{16} + x^{15} + x^{2+} + 1$$

The CRC generation is started by the DLE SOH control sequence or, if not present, the DLE STX sequence. It is terminated by either the DLE ETX or the DLE ETB control sequence. The DLE included in any control sequence, or one of the DLE characters used for DLE stuffing, is not included in the generation. The SOH character is not included in the generation either. The STX character is included only if a header, and, therefore, an SOH control character, is present. The ETX or ETB character is also included in the calculation.

**3.2.2.2** ***Power line communications sessions***

Communications between the LDCU control station and LRCD tributary stations are always initiated by the LDCU. A session begins with the LDCU having a master status and none of the tributary stations having slave status. The LDCU master initiates power line communication by various methods as determined by the message type field. The message types available at present for power line communications are given in Table 12.

**Table 12 — Prefix message types**

| Network command | Transaction |
|---|---|
| Poll | The control station polls a particular tributary station to receive basic information. |
| Xmit1 | Xmit data from the control station to a particular tributary station using fast select. |
| Xmit2 | Xmit data from the control station to a particular tributary station using selection with response. |
| Interactive1 | Xmit1 then, after a delay, interactive poll the same tributary station to receive response. |
| Interactive2 | Xmit2 then, after a delay, interactive poll the same tributary station to receive response. |
| Map | Map tributary stations on the network. |
| Interactive poll | The control station polls a particular tributary station to receive information from a previous xmit. |

The LDCU control station only relinquishes mastership to a tributary station, and thus can obtain information, following the transmission of a polling sequence (poll, map or interactive poll, although the interactive1 and interactive2 use an embedded interactive poll sequence). The xmit sequences (including those embedded in the interactive1 and interactive2 sequences) only transfer information to a tributary station and do not, therefore, relinquish mastership.

The 1 or 2 in xmit and interactive indicates the type of LDCU/LRCD handshaking select sequence used to transmit the data. The 1 indicates a fast select sequence, while the 2 indicates a selection with response sequence. The fast select method of transfer simply sends the data to the addressed LRCD tributary station along with the addressing information (i.e. a prefix preceding the first block of data without the ENQ control character). The selection with response initially selects a tributary station using a polling process (i.e. a prefix followed by an ENQ character). Once the tributary station acknowledges its selection, the (first) block of data is transferred. From the point of view of the MMU, the differences in these processes are invisible, except for the time of response.

The type of power line command processing, e.g. select with response versus fast select or xmit versus interactive, should be carefully selected to coincide with the handling of the LRCD/device message. For instance, a command not required to buffer, and, therefore, not to return any data to the LDCU, should use the xmit power line command so that the LDCU may respond with the LRCD's acknowledge/nacknowledge reply to the MMU. If an interactive power line command that did not require the LRCD to buffer a response was sent, the LRCD would ACK or NAK. Then when the LDCU requested the buffered response, the buffer would be empty and the LRCD would reply with a NAK. The MMU would not know whether the NAK came from the command or the request for the buffered response. Passing this status onto the MMU would not determine whether the LRCD had accepted the command or not, i.e. whether the negative acknowledge came from the first (interactive) or second (interactive poll) command.

The data portion of the transfer session, which comprises the information discussed in **3.2.2.1.4**, is accomplished similarly in all message types. A block of information is transferred from the master device to the slave device. An unsuccessful reception of the information will cause the slave device to return a NAK prefix/control character sequence thus allowing the master device either to retransmit or to abort with an EOT control sequence. A successful reception by the slave device will return the proper ACK prefix/control character sequence thus allowing the master device to transmit the next data block of information, if required. If all data have been transmitted the master device terminates the interaction with an EOT control character sequence, returning mastership to the LDCU control station. Note also that a master device may abort the data transfer at any instance by transmitting a DLE ENQ control character sequence within a data block portion of a transfer.

The LDCU must provide a timeout or waiting period for each expected response from the LRCD. These time periods are used for each retry, i.e. the timeout shall occur for each transaction on the power line; it is not accumulative over all the retries of a single command. A timeout will require the LRCD to transmit a preamble/EOT control sequence onto the power line network to abort all network processes and to allow the LDCU to regain mastership. The timeout period for each command type, and its expected response type, are given in Table 13.

**Table 13 — Power line timeout variables**

| Command type | Timeout period | LDCU's expected LRCD response |
|---|---|---|
| Poll | 0,68 s | Poll data |
| Xmit | 0,68 s | Acknowledge |
| Interactive | 0,68 s | Acknowledge |
| Interactive poll | 4,50 s | Message |
| Map | 0,60 s | Map data |

**3.2.2.2.1** *Poll*

The poll message session allows the LDCU control station to obtain basic information from an RCD tributary station. The LDCU will address the LRCD by transmitting the prefix immediately followed by an ENQ control character. Mastership is assumed by the addressed LRCD and text data are transmitted to the LDCU. (Note that if a broadcast network address is used to select a tributary station, all stations will attempt to respond.) An example showing the flow of information is presented in Figure 9. This example assumes no DLE SOH/header fields (which, if present, would occur between the preamble and DLE STX in line 2). Also a single block contains all the text data to be transferred. An invalid reception by the LDCU from line 2 would have resulted in a preamble/prefix/NAK transfer by the LDCU in line 3. The LRCD would then retry the transmission of line 2 or terminate the session with line 4.

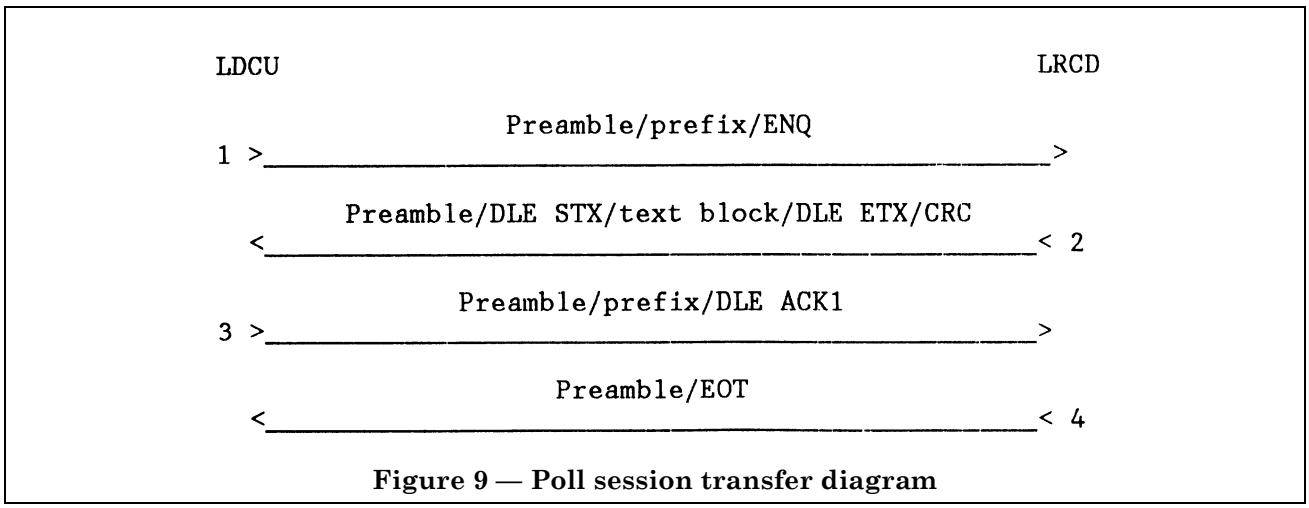
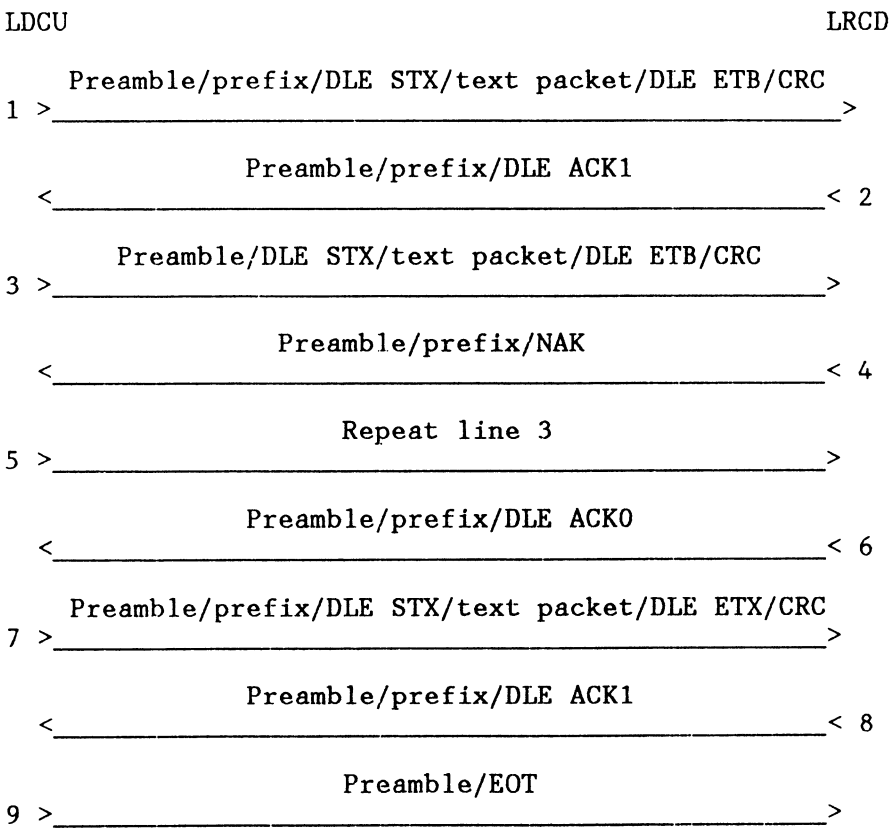



**Figure 9 — Poll session transfer diagram**

**3.2.2.2.2** *Xmit1*

The xmit1 message session transfers text data from the LDCU control station to a selected LRCD tributary station using the fast select process. This fast select method of transfer simply sends the text data to the addressed LRCD tributary station along with the addressing information, i.e. a prefix is transmitted (and not followed by an ENQ control character) immediately preceding the first block of data. Since the LRCD only receives data it never obtains mastership of the power line. An example showing the flow of information is presented in Figure 10. Note that the negative acknowledgement by the LRCD in line 4 requires the LDCU to repeat line 3 in line 5. The acknowledgement for line 5 remains the same (ACK0) as would have been sent in line 3.




**3.2.2.2.3** *Xmit2*

The xmit2 message session transfers text data from the LDCU control station to a selected LRCD tributary device using the selection with response process. This selection with response method of transfer initially selects an LRCD using a polling process (prefix followed by an ENQ character). Once the tributary station acknowledges its selection, the block(s) of data is(are) transferred. Since the LRCD only receives data it never obtains mastership of the power line. An example showing the flow of information is presented in Figure 11.

**3.2.2.2.4** *Interactive1*

The interactive1 message transfer initiates an xmit1 message transfer session (see **3.2.2.2.2**). The LDCU control station then provides a delay before initiating an interactive poll session (see **3.2.2.2.7**) to the same tributary station to obtain information requested and buffered from the xmit1 session. The LDCU control station is free to communicate with other tributary stations during the delay period between the xmit1 session and interactive poll session to this tributary station. The flow of information is thus a dual function and can be exemplified by Figure 10 followed by Figure 14.

**3.2.2.2.5** *Interactive2*

The interactive2 message transfer initiates an xmit2 message transfer session (see **3.2.2.2.3**). The LDCU control station then provides a delay before initiating an interactive poll session (see **3.2.2.2.7**) to the same tributary station to obtain information requested and buffered from the xmit2 session. The LDCU control station is free to communicate with other tributary stations during the delay period between the xmit2 session and interactive poll session to this tributary station. The flow of information is thus a dual function and can be exemplified by Figure 11 followed by Figure 14.

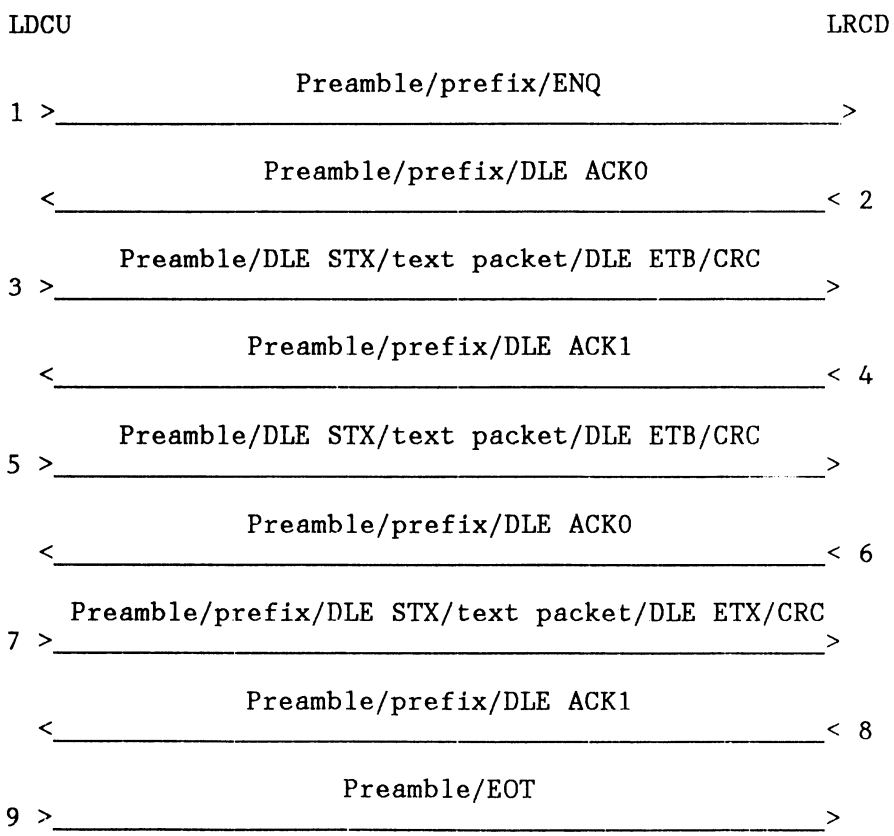



**Figure 11 — Xmit2 session transfer diagram**

**3.2.2.2.6** *Map*

The map message session broadcasts a polling session from the LDCU control station requesting, with varying probabilities, an LRCD tributary station to return its routing byte/network address. The purpose of this command is to log in, or map, any newly acquired or unpolled LRCDs on the power line network.

Since this mapping technique uses a polling type session on the network, the format of the information transmittted from the LDCU is similar to the poll message session described in **3.2.2.2.1**, i.e. a prefix followed by the ENQ control character. However, since it is a broadcast type of message, it cannot simply select a particular LRCD tributary slave by using a unique individual network address. Instead, the 11 byte ASCII network address field described in **3.2.2.1.3** is broken into ASCII subfields supplying all LRCDs on the power line network with a mapping strategy (as defined in **3.2.1.1.2.4**). An individual LRCD selected by this mapping strategy obtains mastership of the power line network and transmits a text data block containing its unique routing byte/network address to the LDCU. Since, however, this type of broadcast technique inherently suggests the possibility of selecting more than one tributary station, multiple masters may temporarily drive the power line communications medium. This results in an indistinguishable reception in the LDCU slave unit. In this instance, the LDCU slave transmits an EOT control character sequence to regain master status rather than returning the NAK prefix/control character sequence. A distinguishable transmission reception indicates a single tributary station has mapped onto the system and thus the LDCU slave returns an ACK prefix/control character sequence. Examples of the flow of information for an identified and an unidentified map are presented in Figure 12 and Figure 13 respectively. Note that the information in the text data block will contain the individual LRCD 12 byte ASCII routing byte/network address.

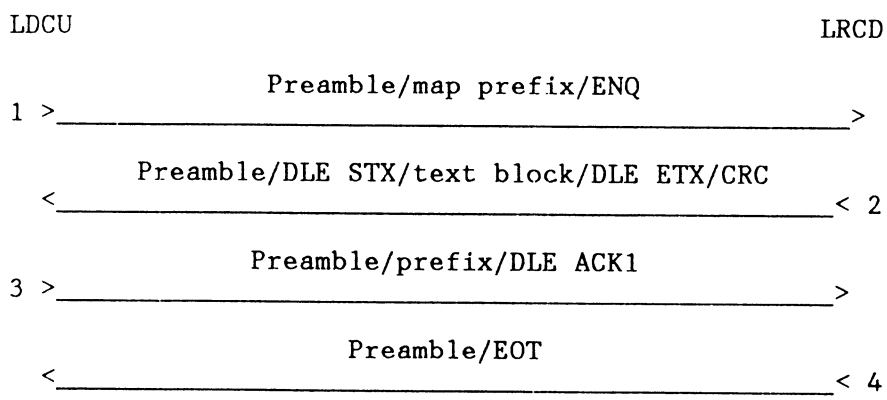



**Figure 12 — Identified map session transfer diagram**

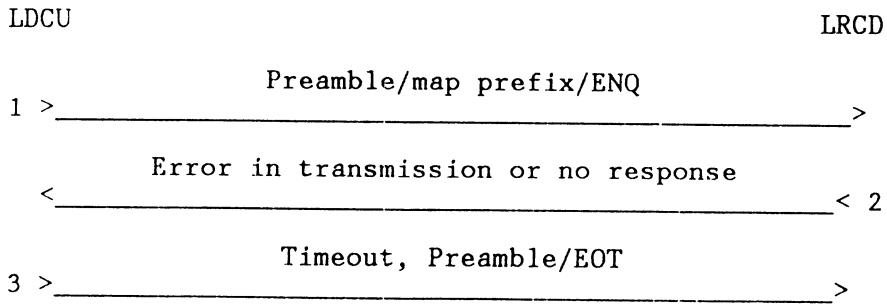



**Figure 13 — Unidentified map session transfer diagram**

**3.2.2.2.7** *Interactive poll*

The interactive poll message session is used by the LDCU control station to retrieve information from an LRCD tributary station after the LRCD has been sent an xmit1 or xmit2 message session (see **3.2.2.2.2** and **3.2.2.2.3** respectively) containing information which required the LRCD to buffer response information. This process is executed in the same manner as the poll message session of **3.2.2.2.1**. The LDCU will address the LRCD by transmitting the prefix immediately followed by an ENQ control character. Mastership is assumed by the addressed LRCD and the buffered text data are transmitted to the LDCU. An example of the flow of information is presented in Figure 14.

### 3.2.3 MDCU to HRCD communications

The physical requirements and low level data link protocol for HDCU communications with the slave HRCDs is described in **3.6**. Two basic types of communication are possible between an HDCU and an HRCD: data packet exchanges (see **3.6.9.2** and **3.6.9.3**) and control exchanges (see **3.6.9.4**). Data packet exchanges transfer information from a logically higher layer in an MMU/HDCU or controller/HRCD to its peer layer in an HRCD/controller or HDCU/MMU; control exchanges are used by an HDCU to grant an HRCD temporary access to the power line network to perform a data packet exchange. Control of the power line network then returns to the HDCU.

#### 3.2.3.1 *HDCU/HRCD power line communications components*

The fields transferred for both a data packet exchange and a control exchange are discussed in **3.2.3.1.1** and **3.2.3.1.2**. The actual format of the transfer over the power line including the low level data link control bytes, is detailed in **3.6**.

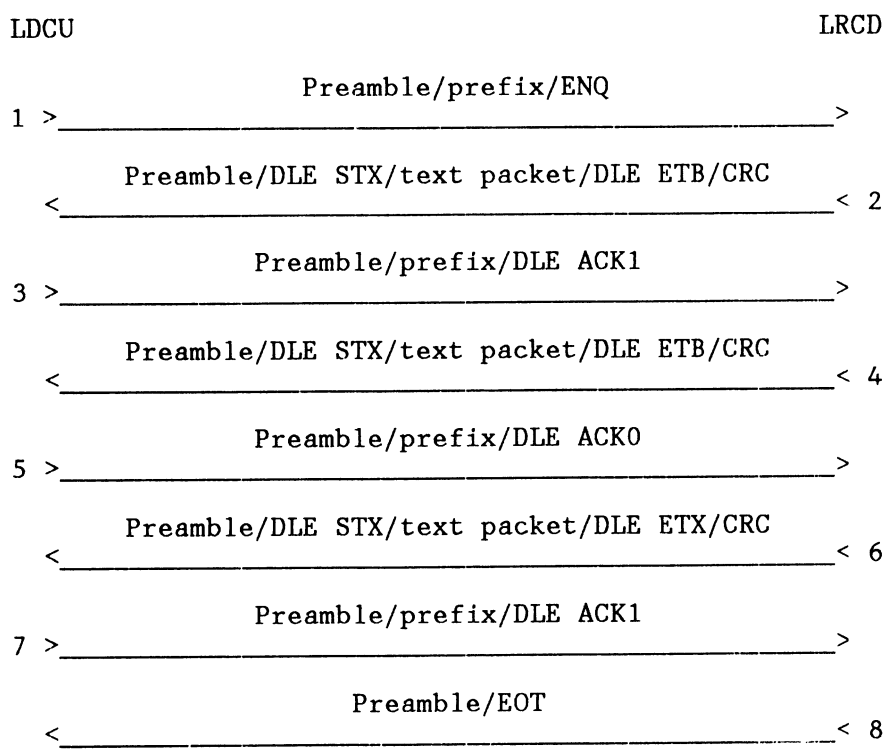



**Figure 14 — Interactive poll session transfer diagram**

**3.2.3.1.1** *Data packet exchange*

The actual data transferred during a data packet exchange contain three components: the data packet header, the data field to be transferred, and the packet check sum field:

| Header | Data | Check sum |

The protocol governing the exchange is as described in **3.6.9.2** and **3.6.9.3**.

**3.2.3.1.1.1** *Data packet header*

The data packet header contains the fields required for transfer of the data packet to the designated destination. They are as follows:

| Destination address | Source address | Packet length |

Destination address — Destination address is a 2 byte network address in the range 0000 to FFFE (hexadecimal) selecting a unique HRCD (0001 to FFFE) or the HDCU (0000). Address 0000 is reserved for the HDCU address and address FFFF (hexadecimal) is reserved for the broadcast address. Addresses are assigned to the HRCDs by the HDCU during the log in procedure. The HDCU maintains an internal lookup table associating each 11 byte ASCII RCD address (Container ID) on the power line network with a unique destination address.

Source address — Source address is the unique 2 byte network address in the range 0000 to FFFE (hexadecimal) specifying the transmitting HRCD (0001 to FFFE) or HDCU (0000).

Packet length — A 2 byte unsigned integer which gives the length of the data field.

**3.2.3.1.1.2** *Data field*

The data field contains the information to be transferred to the destination HRCD or HDCU. The format of the data field is as follows:

| Message type | Routing byte | Network address | Data |

Message type — A 1 byte command directing what type of transfer is being requested by the MMU and corresponding to the MMU/MDCU xmit type command listed in Table 2.

Routing byte — ASCII routing byte defined as shown in Figure 15.

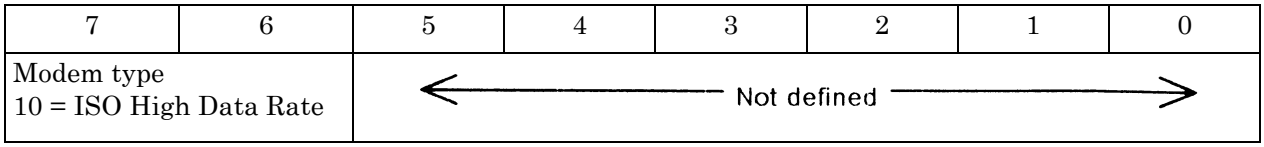



**Figure 15 — Routing byte definition**

Network address — 11 byte ASCII characters defining the address of (an) RCD(s) on the power line network, i.e. the container ISO number as defined in **2.2.2.4.1**.

Data — Variable length data field for which the contents are dependent on the type of message to be processed. The contents are defined in **3.2.1.3** and **3.2.4** for each appropriate command group.

**3.2.3.1.1.3** *Packet check sum field*

The packet check sum field contains an 8 bit exclusive-or check sum of all data packet header and data fields used in addition to the (32,8) ECC and EDC encoding (see Table 31) used on each byte of all the data packet header and data fields.

**3.2.3.1.2** *Control exchange*

The HDCU uses a control exchange to grant an HRCD temporary access to the power line network to perform a data packet exchange. Control of the power line network then returns to the HDCU. The header field consists of the destination address only. The data field is empty.

| Header | Data | ⇒ | Destination address |

The protocol governing the exchange is as described in **3.6.9.4**.

**3.2.3.2** ***HDCU/HRCD communications***

Communications between the HDCU and HRCDs are always initiated by the HDCU. The HDCU uses the message type field to determine the type of session. The 14 message types available are grouped into five different categories as given in Table 14.

**Table 14 — Message types**

| Message types | Transactions |
|---|---|
| RCD/device poll | The HDCU uses a data packet exchange to request basic information. It follows with a control exchange to retrieve the requested information. |
| RCD/device xmit1/2 | The HDCU uses a data packet exchange to transmit data to an RCD/controller. |
| RCD/device interactive1/2 | RCD/device xmit1/2 then, after a delay, the HDCU uses a control exchange to retrieve the RCD/controller response. |
| RCD/device map | The HDCU uses a broadcast data packet exchange, a data packet exchange and a control exchange to log in new RCDs on the power line network. |
| RCD/device interactive poll | The HDCU uses a packet exchange to transmit this request to the HRCD. It follows with a control exchange to retrieve information from an HRCD from a previous RCD/device xmit1/2. |

**3.2.3.2.1** *RCD/device poll*

The RCD/device poll message session allows the HDCU to request basic information from a selected HRCD. The HDCU uses a data packet exchange to transmit this request to the HRCD. The HDCU then retrieves the HRCD's response to this request using a control exchange. An example showing the flow of information is presented in Figure 16.

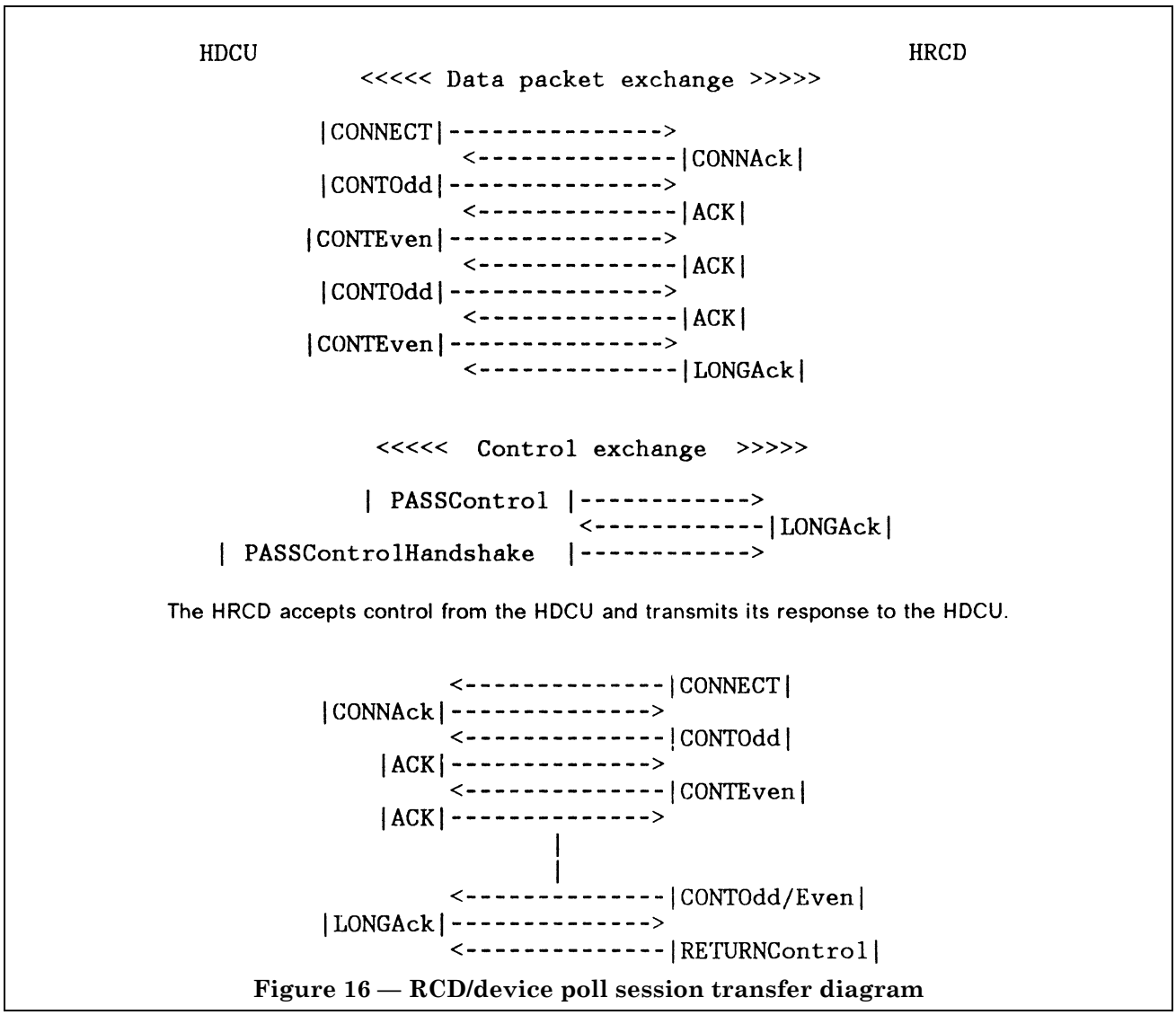



**Figure 16 — RCD/device poll session transfer diagram**

**3.2.3.2.2** *RCD/device xmit1/2*

The RCD/device xmit1/2 message session transfers data from the HDCU to a selected HRCD using a data packet exchange. The data can be a message requiring an HRCD response, in which case the HRCD buffers it in the HRCD response buffer. An interactive poll (see **3.2.3.2.5**) is then used to retrieve this buffered response. An example showing the flow of information is presented in Figure 17.

**3.2.3.2.3** *RCD/device interactive1/2*

The RCD/device interactive1/2 message session initiates an xmit1/2 message transfer session (see **3.2.3.2.2**) to a selected HRCD. The HDCU then initiates a control exchange (see **3.2.3.2.5**) to the same HRCD after a time delay as specified in the response delay field of the interactive1/2 command (see **3.2.1.1.2.3**) to retrieve the information requested and buffered from the xmit1/2 session. The HDCU is free to communicate with other HRCDs during the delay period between the xmit1/2 session and the control exchange to this HRCD. An example of the flow of information is presented in Figure 18.

**3.2.3.2.4** *RCD/device map*

The RCD/device map message session broadcasts an MMU-originated map command from the HDCU, using a broadcast data packet exchange as defined in **3.6.9.3**, requesting, with varying probabilities, an HRCD to respond. The purpose of this command is to log in, or map, any newly acquired or unpolled HRCDs on the power line network. Since it is a broadcast type of message, it cannot simply select a particular HRCD by using a unique individual network address. Instead, the 11 byte ASCII network address field described in **3.2.3.1.1.2** is broken into ASCII subfields supplying all HRCDs on the power line network with a mapping strategy (as defined in **3.2.1.1.2.4**). An "unused" 2 byte network address is included for use by a (the) responding HRCD(s). A (The) responding HRCD(s) will use this "unused" 2 byte network address for 20 s. The HDCU then attempts to send a command, e.g. a device poll or an RCD poll, using a data packet exchange addressed to this 2 byte network address, with the container ID set to the universal address. This type of broadcast technique, however, suggests the possibility of more than one HRCD attempting to receive this command. If the HDCU does not successfully complete this data packet exchange, the HDCU replies to the MDCU with the appropriate reply type (described in **3.2.1.2**). A successful data packet exchange indicates that a single HRCD has mapped onto the system and thus the HDCU grants temporary access to the power line network to this HRCD via a control exchange. The HRCD accepts temporary control and transmits its 11 byte ASCII network address to the HDCU via a data packet exchange and refreshes its 20 s timer. The log in sequence is completed when the HRCD receives an individually addressed command with its actual container ID within 20 s. An example showing the flow of information for a successful map is shown in Figure 19.

The assignment of a new 2 byte RCD network address to a specific RCD with a known 11 byte address can be used to minimize mapping response collisions. This case can address RCDs which have dropped off-line owing to power loss or lack of interrogation within the last hour.

**3.2.3.2.5** *RCD/device interactive poll*

The RCD/device interactive poll message session is used by the HDCU to retrieve information from an HRCD after the HRCD has been sent an xmit1/2 message session (see **3.2.3.2.2**) which required the HRCD to buffer response information. This process is executed in the same manner as the poll message session of **3.2.3.2.1**. The HDCU uses a data packet exchange to transmit this request to the HRCD. The HDCU then retrieves the HRCD's buffered response using a control exchange. For an example of the flow of information, see Figure 16.

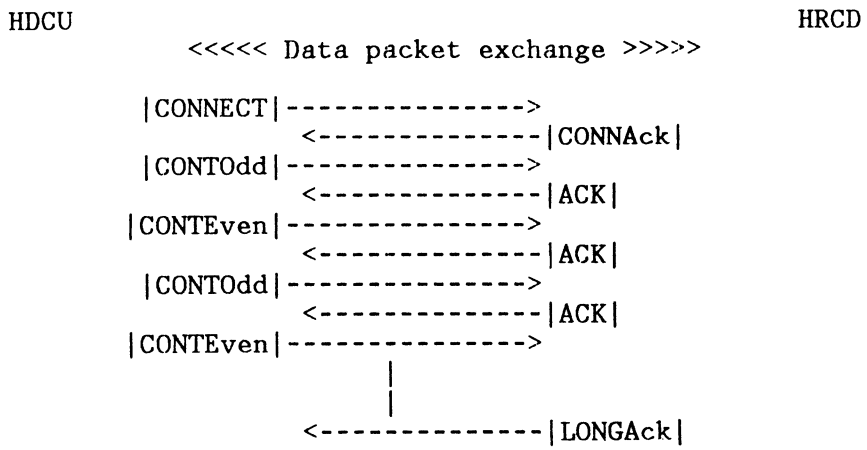



**Figure 17 — RCD/device xmit1/2 session transfer diagram**

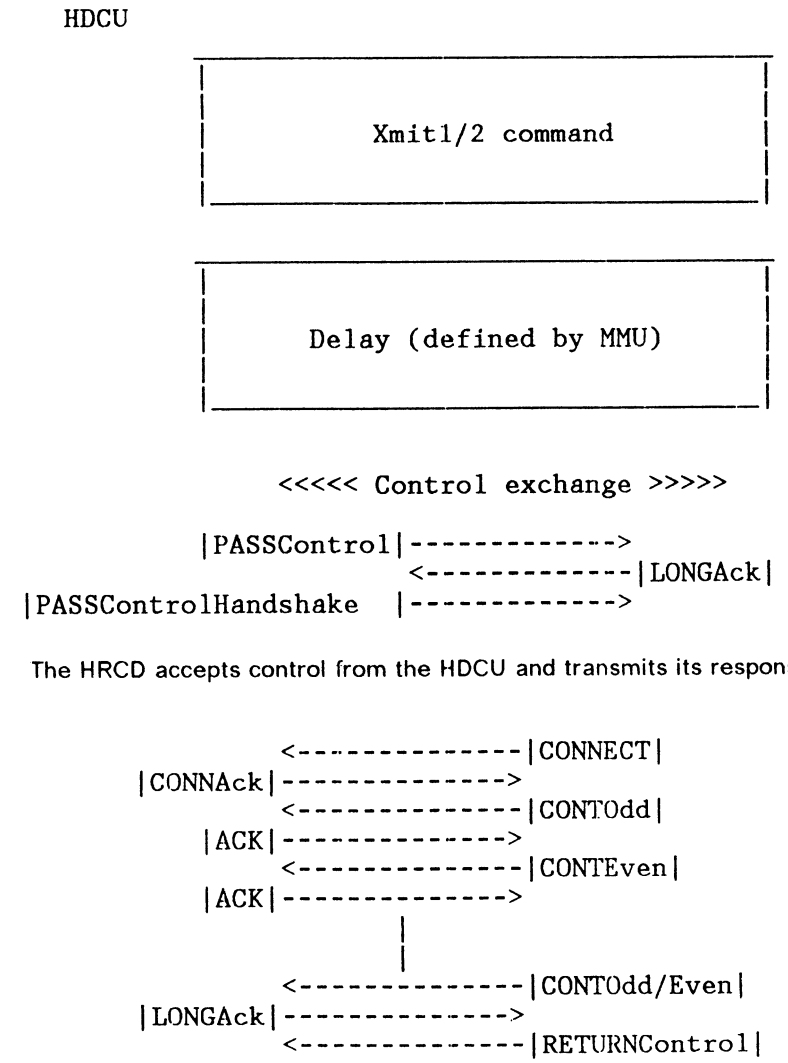
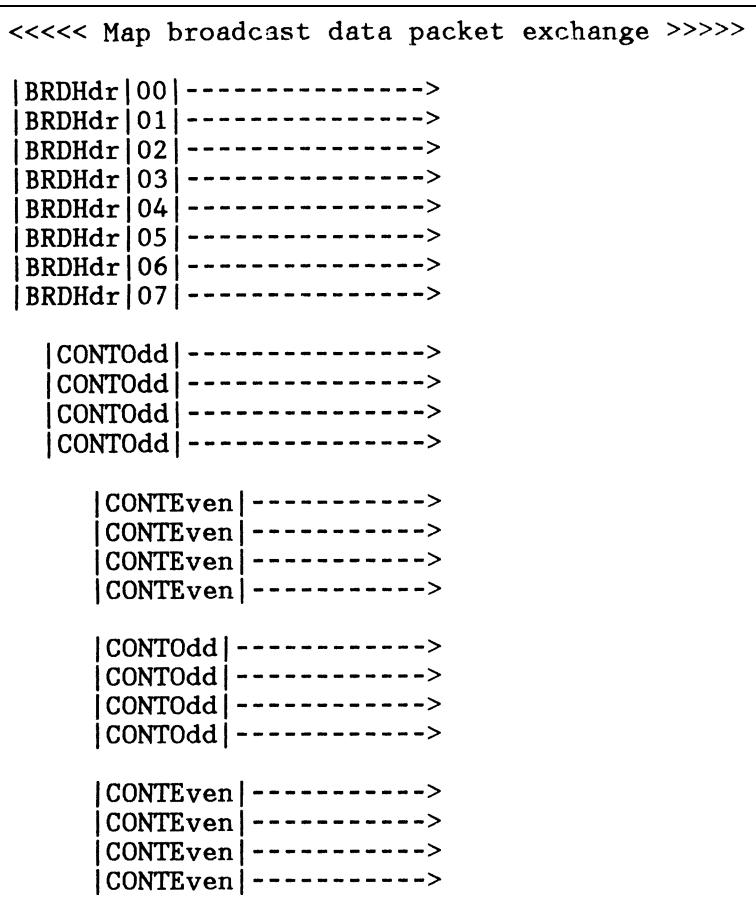


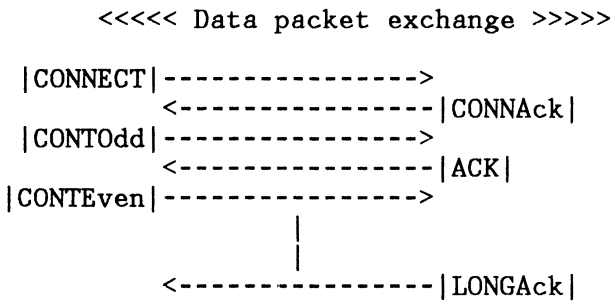

**Figure 18 — RCD/device interactive 1/2 session transfer diagram**




The HDCU broadcasts the MMU-originated map command. The 2 byte unused HRCD destination address or destination address individual included in the map command is used by an HRCD that responds to the map.

Source address = HDCU address

Data consist of 1 byte for the message type, 1 byte for the routing byte, and the 11 byte ASCII RCD network address. The optional 2 byte destination address individual is included when the 11 byte ASCII RCD network address specifies an individual container ID. The exchange is sent as specified in 3.6.9.3.




The HDCU sends a command, e.g. a device poll or an RCD poll, addressed to the 2 byte unused HRCD destination address/destination address individual address included in the previous map command. The 11 byte ASCII RCD network address used in this command is the universal container address. It is used by the HDCU to determine whether an HRCD responded to the previous map command, and if so the HRCD returns its container ID.

Destination address = Unused HRCD destination address/destination address individual from previous map broadcast data packet

Source address = HDCU address

Data is the command using the 11 byte ASCII universal address of 30H 30H 30H 30H followed by seven ASCII digits (30H to 39H).

**Figure 19 — RCD/device map session transfer diagram** *(continued)*

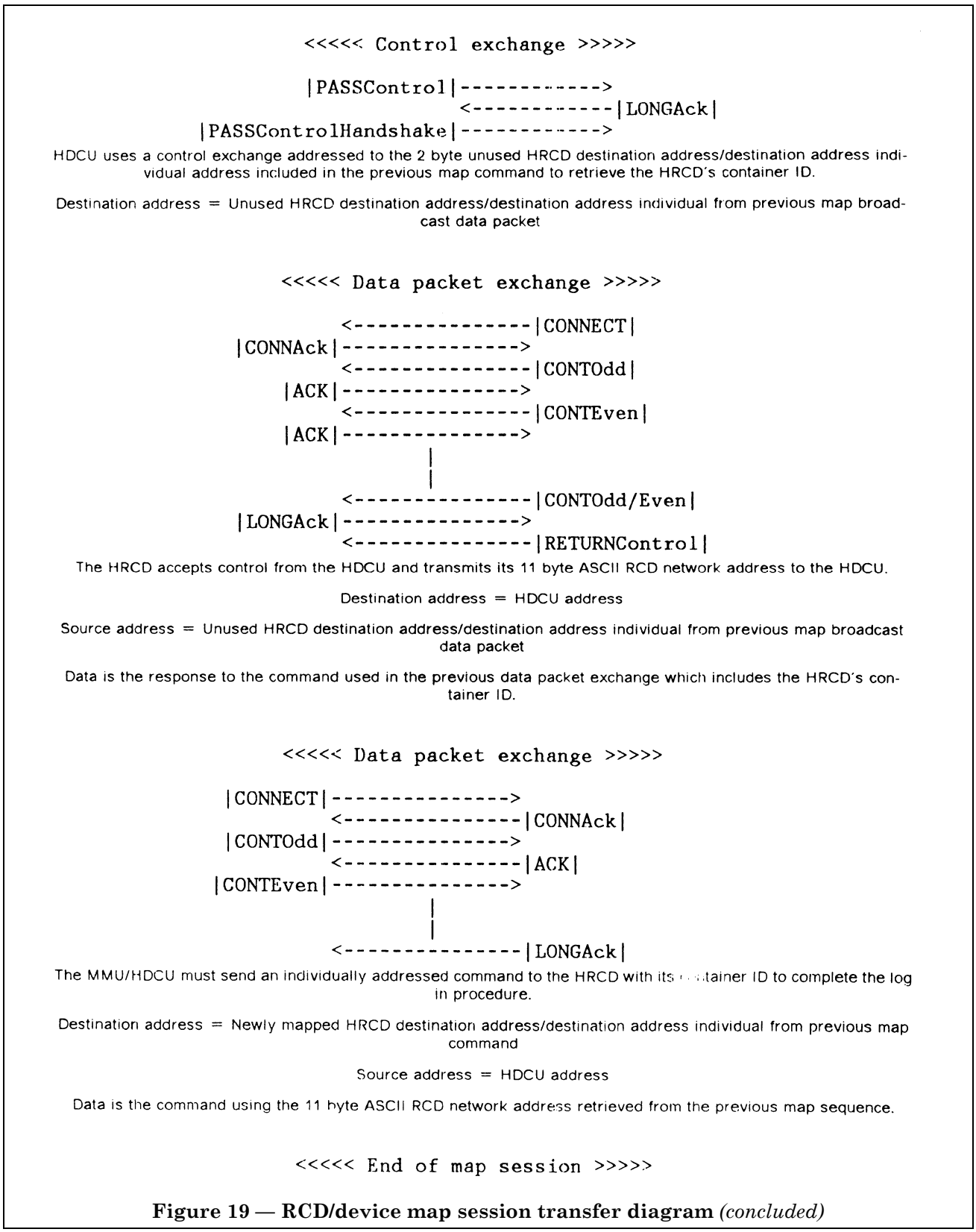



**Figure 19 — RCD/device map session transfer diagram** *(concluded)*

### 3.2.4 RCD to controller communications

The RCD/controller format is illustrated in Figure 20 and described **3.2.4.1** to **3.2.4.6**.

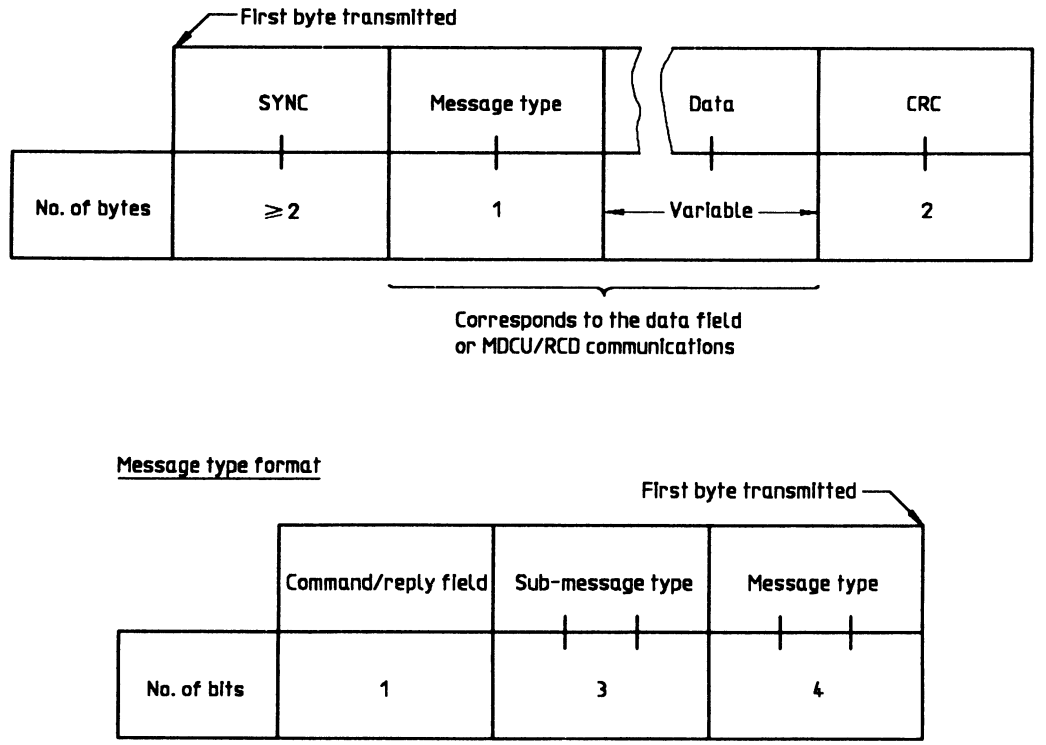



**Figure 20 — RCD/controller format**

**3.2.4.1** ***SYNC field***

The SYNC field consists of two or more characters.

It shall contain only ASCII SYNC characters (16 16).

**3.2.4.2** ***Message type field***

**3.2.4.2.1** *Purpose*

This field shall follow the SYNC field. It contains information about the message function. The length of the data field is implied by the contents of the message type field.

**3.2.4.2.2** *Length*

The message type field is 4 bits long.

**3.2.4.2.3** *Contents*

The contents of the message type field are defined in **3.4**.

**3.2.4.3** ***Sub-message type field***

**3.2.4.3.1** *Purpose*

This field shall follow the message type field. It contains additional information about the message function. Not all message types have valid sub-message type fields.

**3.2.4.3.2** *Length*

The sub-message type field is 3 bits long.

**3.2.4.3.3** *Contents*

The contents of the sub-message type field are defined in **3.4**.

**3.2.4.4** ***Command/reply field***

**3.2.4.4.1** *Purpose*

This field shall follow the sub-message type field. It defines the message as a command message from the master to the controller or as a reply message from the controller to the RCD.

**3.2.4.4.2** *Length*

The command/reply field is 1 bit long.

**3.2.4.4.3** *Contents*

Command/reply set to zero (0) denotes a command message. Command/reply set to one (1) denotes a reply message.

**3.2.4.5** ***Data field***

**3.2.4.5.1** *Purpose*

The data field contains any data that may need to be transferred between the two devices.

**3.2.4.5.2** *Length*

The data field comprises a variable, predetermined, number of characters that is dependent on the message type.

**3.2.4.5.3** *Contents*

The contents of the data field are dependent on the message type (see **3.4**).

**3.2.4.6** ***CRC (error control) field***

**3.2.4.6.1** *Purpose*

The CRC (error control) field enables the receiving computer to detect transmission errors caused by noise in the communications channel.

**3.2.4.6.2** *Length*

The error control field consists of two characters.

**3.2.4.6.3** *Contents*

The content of the error control field is a CRC check sum word. The 16 bit check sum is transmitted as 2 bytes. The first byte is the high order byte of the CRC word. The second byte is the low order byte of the CRC word.

Details of the calculation and checking of the error field check sum are given in **3.2.5**.

If the RCD detects a check sum error in a reply message packet, it can reissue the previous command message packet as a new command message.

The controller shall not reply to a command message packet that has a check sum error.

**3.2.5 State diagram**

The message transaction protocol is completely defined by the two state diagrams given in Figure 21 and Figure 22.

**3.2.5.1** ***Master state diagram***

The state diagram that defines how the RCD sends command packets, receives reply packets, and handles error recovery is given in Figure 21. For the sake of clarity, the conditions that cause a transition are not labelled on Figure 21 but are listed in Table 15 and are cross-referenced to the figure by a letter code.

**3.2.5.1.1** *General description*

At reset, the RCD enters the idle state. In the idle state, the RCD and the controller should be synchronized. The RCD can send a command message packet at this time. Once a command message packet has been sent, the RCD sets a SYNC timer of 4 000 ms and waits for the SYNC field characters from the controller. If the SYNC timer times out before the SYNC characters are received, the RCD returns to the idle state, making the assumption that the command message packet was lost or had a transmission error.

If the SYNC characters are received, the RCD sets a receive character timer to 500 ms and waits for the next character. When the next character is received, the counter is updated and the receive character timer is set to 500 ms. If the receive character timer times out, the RCD must assume that a communication error has occurred. It may execute an error recovery routine and then must return to the idle state.

When the expected number of characters has been received, the message packet has, by definition, been received.

Once a reply message packet has been received, the RCD checks it for transmission errors and a correct message type. If an error or mismatch of message types occurs, an optional error recovery routine can be executed. Otherwise, the reply packet is processed.

The RCD shall ignore any unexpected characters that are received while it is in the idle state.

**3.2.5.1.2** *Power-up state*

The power-up state is entered after the RCD is reset or executes a software restart. The transfer to the idle state is made when the RCD is initialized and is ready to begin communications with the controller.

**3.2.5.1.3** *Idle state*

The RCD and controller should be synchronized and should both be in the idle state. No command message packets are outstanding, and the RCD should not receive any characters from the controller.

**3.2.5.1.4** *Transmit command state*

The RCD has decided to transmit a command packet to the controller. This state is entered while the command message packet is being transmitted to the controller.

Otherwise, an error has occurred and an error recovery routine should be executed.

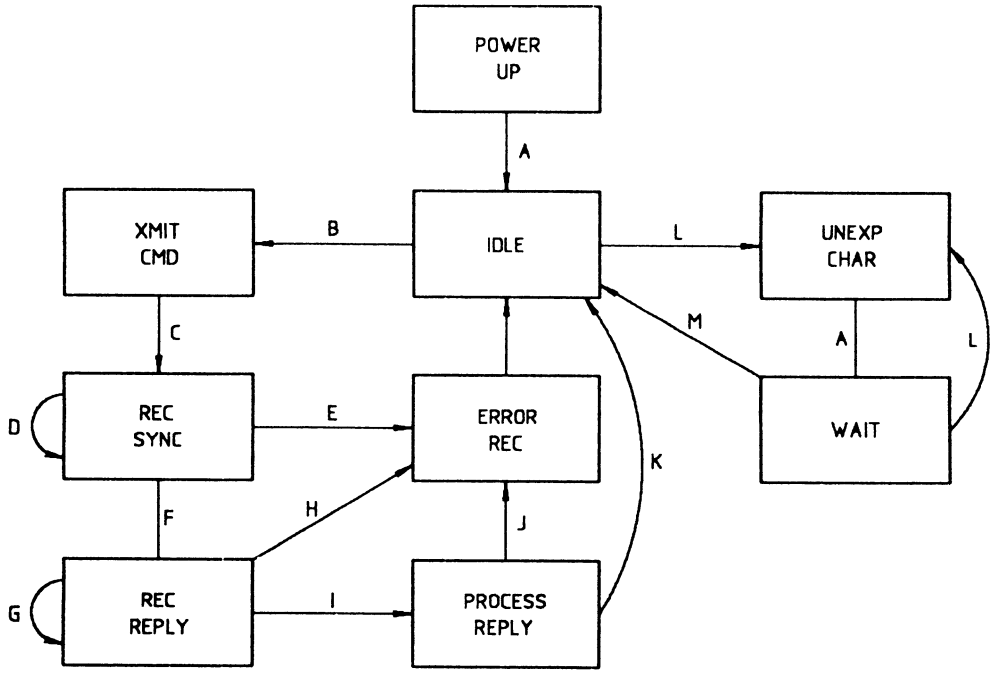



**Figure 21 — Master message transaction state diagram**

**Table 15 — Master (RCD) state diagram transition conditions**

| Letter code | Condition |
|---|---|
| A | always make this transition |
| B | RCD starts transmission of command message |
| C | message transmitted, set SYNC timer to 4 000 ms |
| D | SYNC field not received and SYNC timer has not timed out |
| E | SYNC field not received and SYNC timer has timed out |
| F | SYNC field received, set receive character timer to 500 ms, character counter |
| G | received character and character counter has not reached expected value, set receive character timer to 500 ms and update character counter |
| H | receive character timer has timed out |
| I | received character and character counter has reached expected value |
| J | transmission error or wrong reply message type |
| K | no transmission error and correct reply message type |
| L | receive unexpected character, set unexpected character timer to 2 000 ms |
| M | unexpected character timer times out |

**3.2.5.1.5** *Error recovery*

The error recovery state is defined such that the RCD may execute any error routines that might help in informing the operator about the status of the controller. For example, the RCD might retry the operation if a transmission error occurred.

**3.2.5.1.6** *Received unexpected character state*

The controller is out of synchronization with the RCD and has sent a character when the RCD was not expecting one. The RCD should set an unexpected character timer to 2 000 ms (2 s) and start the timer. The received character is ignored.

**3.2.5.1.7** *Wait for unexpected character timeout*

The RCD waits for the unexpected character timer to timeout. If another character is received, the RCD shall go to the received unexpected character state.

**3.2.5.2** ***Slave state diagram***

The state diagram that defines how the controller receives command packets, sends reply packets, and handles error recovery is given in Figure 22. For the sake of clarity, the conditions that cause a transition are not labelled on Figure 22 but are listed in Table 16 and are cross-referenced to the figure by a letter code.

**3.2.5.2.1** *General description*

After a reset or software restart, the controller is in the power-up state. After internal initialization, the controller enters the idle state.

While in the idle state, the controller is waiting for a command message packet. When a character is received, the controller looks for the SYNC characters. Note that the original character received may be the first of the two SYNC characters. Once the SYNC characters are received, the controller looks for the message packet. When the message packet is received, the check sum is checked. If the check sum is wrong, a transmission error has occurred and the controller ignores the message and returns to the idle state.

If there were not any transmission errors, the slave then processes the command and sends the reply information to the RCD in a reply message packet. The controller returns to the idle state after the reply packet has been transmitted.

**3.2.5.2.2** *Power-up state*

The power-up state is entered after the controller has been reset either by hardware or by software. When the controller is ready to communicate with an RCD, the idle state will be entered.

**3.2.5.2.3** *Idle state*

The RCD and controller should be synchronized and should both be in the idle state. The controller is waiting for a command message packet to arrive from the RCD.

The controller may not leave the idle state without first receiving a character from the master.

**3.2.5.2.4** *Receive SYNC state*

The controller has received a character. The controller sets a SYNC timer to 500 ms and waits for the next character. When two consecutive SYNC characters have been received, the controller goes to the receive packet state. Note that the first character may be a valid SYNC character. If the SYNC timer times out before the two SYNC characters have been received, the controller returns to the idle state.

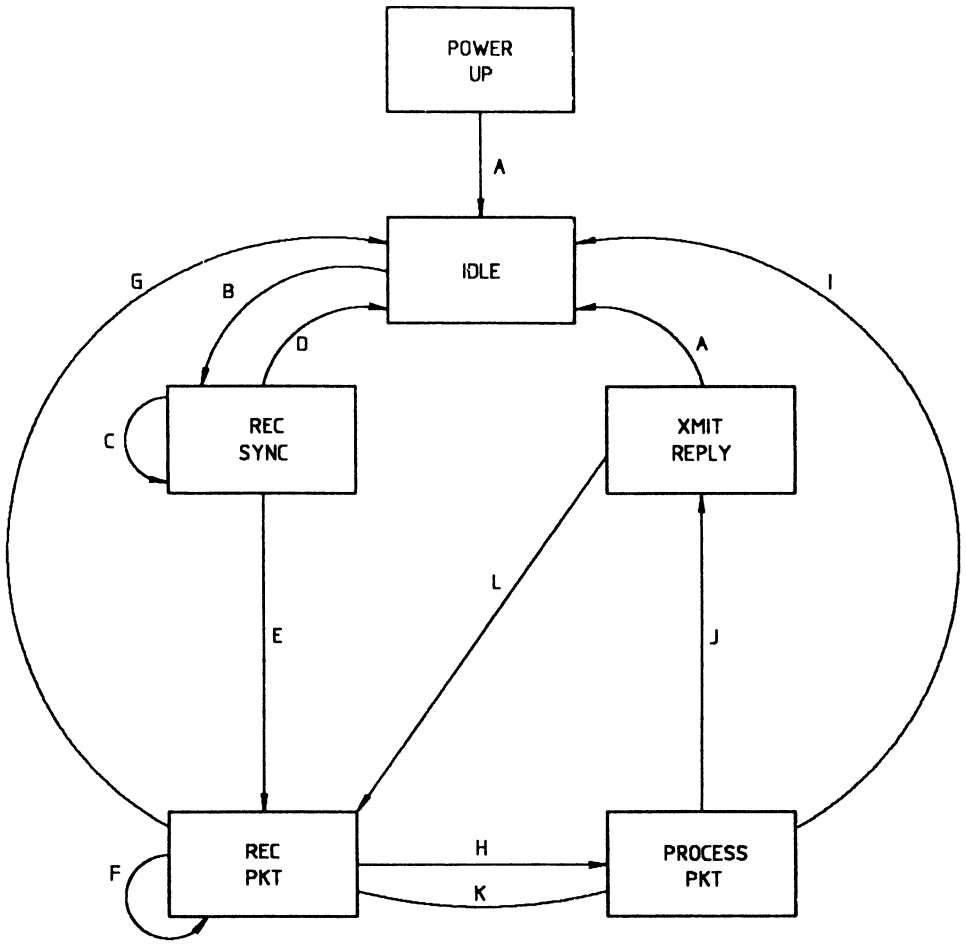



**Figure 22 — Slave (controller) message transaction state diagram**

**Table 16 — Slave (controller) state diagram transition conditions**

| Letter code | Condition |
|---|---|
| A | always make this transition |
| B | controller has received a character from the RCD, set SYNC timer to 500 ms |
| C | controller has not received two consecutive SYNC characters and SYNC timer has not timed out |
| D | controller has not received two consecutive SYNC characters and SYNC timer has timed out |
| E | controller has received two consecutive SYNC characters, set receive character timer to 500 ms |
| F | received packet not complete and receive timer has not timed out |
| G | received packet not complete and receive timer has timed out |
| H | received packet is complete |
| I | transmission error or no transmission error but message type is reserved or undefined |
| J | no transmission error, and message type is valid, and no SYNC field, and command processing finished |
| K | SYNC field received and command processing is not finished |
| L | SYNC field received and reply transmission is not finished |

**3.2.5.2.5** *Receive packet state*

The controller has received a valid SYNC field and is now receiving characters from the RCD. The next byte received contains the message and sub-message type fields. This byte will imply the number of bytes in the packet data field. If the message type is undefined or reserved, the controller will accept characters until the RCD stops transmitting.

When this state is entered, a receive character timer is set to 500 ms.

When the number of incoming bytes is known, the receive character counter is loaded with a character count. The character count is dependent on the command message type.

When a character is received by the RCD, the receive character counter is updated and the receive character timer is set to 500 ms. If the receive character timer times out, the controller throws away the message packet and returns to the idle state.

If the controller receives the required number of characters, it enters the process message state.

**3.2.5.2.6** *Process message state*

The controller has received a message packet. The controller must first check the packet for a transmission error. If no error exists, the message type field is checked. If the message type is reserved or undefined, the message is ignored and the controller returns to the idle state. No reply message is sent.

For valid message types, the requested action is taken and the controller goes to the transmit reply state.

If a SYNC field is received by the controller while processing a message, the message processing is aborted and the controller goes to the receive packet state.

**3.2.5.2.7** *Transmit reply state*

The controller is transmitting the reply message packet. The contents of the packet are dependent on the specific command message type (see **3.4**). After the reply message packet has been transmitted, the controller returns to the idle state.

If the controller receives a SYNC field from the RCD before it has finished transmitting the reply message packet, the controller will abort the transmission and go to the receive packet state.

**3.2.6 Error detection algorithm**

The error detection algorithm used is a 16 bit cyclic redundancy check (CRC) check sum. The generating polynomial for the CRC check sum is the CRC-16 polynomial given in the following equation:

$$G(x) = x^{16} + x^{15} + x^2 + 1$$

The CRC calculation shall conform to **3.2.2.1.4** and **3.2.4.6** with the exception of the bit transmission order of the check sum. The CRC word shall be transmitted high order byte first, followed by the low order byte. Within the bytes, the least significant bit shall be transmitted first. This conforms to the bit transmission characteristics of the SYNC, message, sub-message, and data fields of the controller protocol.

The CRC calculation will not include the SYNC characters. It will include the message, sub-message, command/reply, and data fields. The CRC calculation shall start after the reception of the SYNC field. The calculation of the CRC field, for transmitting messages, shall be done using an initial value of zero. Transmission errors are detected by receiving a message packet which produces a non-zero remainder after the CRC calculation.

## 3.3 Data-logging formats

This clause describes the formats and retrieval techniques for trip record data. Data-logging information shall be recorded in a reusable ring buffer. The buffer length shall be an integer multiple of 4 bytes. The buffer is extracted from the recording device in numbered blocks using the proper read message defined in **3.4.5.3**. All blocks are 128 bytes long. Block 0 is defined to contain general information concerning the recording device and trip record (see **3.3.4**). The formats for individual readings in this buffer are defined in **3.3.3**.

These formats provide for a variable number of sensors (two or four) and variable resolution depending on deviation from set-point and the application of the auxiliary sensors. Dynamic changes in the recording interval, number of sensors and recording resolution, and the occurrence of special events, are signaled by embedded control words.

A control word and its associated extended report define all parameters necessary to decode subsequent readings. Control words are described in **3.3.1**. They are entered as part of trip record headers, which are described in **3.3.2**, or spontaneously by the recording device in response to external events, as defined in **3.3.5**.

### 3.3.1 Control words

Control words are used to tell the software how to interpret subsequently recorded data or to indicate the occurrence of an external event.

**3.3.1.1** Control words are always preceded by a preamble 3 bytes of all ones (FF) and are 1 byte or 5 bytes long. The locations of the various fields contained in the first byte of all control words are as shown in Table 17.

**Table 17 — Locations of fields in the first byte of control words**

| Contents | Location of bits | No. of bits |
|---|---|---|
| Recording interval | Bits 0-1 | 2 |
| Undefined | Bits 2-3 | 2 |
| Record length | Bits 4-5 | 2 |
| Resolution | Bits 6-7 | 2 |

**3.3.1.2** The message contents are interpreted as shown in Table 18.

**Table 18 — Interpretation of message contents**

| Contents | Definition |
|---|---|
| Recording interval | 00 = 1/2 h<br>01 = 1 h<br>10 = 2 h<br>11 = 4 h |
| Record length indicator | 00 = 0 (start of trip mark)<br>01 = 128 bytes (standard header)<br>10 = 6 bytes (event mark)<br>11 = reserved |
| Resolution type | 00 = Two sensor<br>01 = Four sensor, high resolution<br>10 = Four sensor, low resolution<br>11 = reserved |

### 3.3.2 Headers

**3.3.2.1** Headers are entered using the data log write message described in **3.4.6.3**. The data field consists of a 1 byte control word and preamble followed by text fields. Each field is a string of variable length using standard ASCII characters separated with AA in hex and padded with AA in next to a combined length of 128 bytes. The recording device shall interpret the control word and place the header, as transmitted, into the ring buffer beginning at the current recording location.

**3.3.2.2** A header contains the following fields in the order given.

1 Container
2 Vessel voyage
3 Origin
4 Shipper
5 Temperature set-point
6 Air exchange setting
7 Humidity set-point
8 Operator
9 Date
10 Time
11 Product
12 Intermediate destination
13 Final destination
14 Booking
15 Comments

**3.3.2.3** Each field in the header shall be less than 13 characters.

**3.3.2.4** The total number of characters in a header, not including the control word and preamble, shall equal 124. The last byte shall be hex AA.

### 3.3.3 Readings

Readings are entered in the ring buffer periodically. The format of individual readings is controlled by the deviation of the controlling sensor from set-point. Large deviations are recorded in coarse increments of 1 °C. Small deviations are recorded in fine increments of 0,25 °C. The coarse/fine indicator determines the increments of the supply and return air temperature readings. Since these values can be very dynamic, each reading contains its own coarse/fine indicator. Reports will change from fine to coarse increments when the control temperature is greater than 4,5 degrees above set-point or lower than 3,0 degrees below set-point. Reports will change from coarse to fine increments when the control temperature is less than 3,0 degrees above set-point or higher than 2,0 degrees below set-point.

Readings from the auxiliary sensors are optional. They can be recorded in high or low resolution. Since the number and application of auxiliary sensors will not change often, their resolution indicator and a flag controlling their recording are contained in the appropriate control word. A two sensor system will have report formats that are identical to the first 2 bytes (byte 0 and 1) of a four sensor report.

A complete reading includes the temperature recordings and an optional extension. Extended reports are normally issued when the condition of individual alarms changes from off to on, when alarms are manually cleared, or when a change in set-point occurs. A special extended report shall be issued whenever a control word is entered and at power up. Note that a binary, as the most significant bit of an odd-numbered byte (1, 3, ...), always indicates the end of a reading.

#### 3.3.3.1 *Coarse temperature report — High resolution*

This example contains information for both a two and a four sensor system. Bytes 0 and 1 are identical for both and four sensor systems. Bytes 2 and 3 are not recorded in a two sensor system.

**3.3.3.1.1** The locations of the various fields contained in each recording are shown in Table 19.

**Table 19 — Location of fields**

| Contents | Location of bits | No. of bits |
|---|---|---|
| Coarse/fine indicator | Byte 0: Bit 0 | 1 |
| Control sensor | Byte 0: Bits 1-6 | 6 |
| Control sensor less Non-control sensor | Byte 0: Bit 7 to Byte 1: Bits 0-2 | 4 |
| Recent defrost indicator | Byte 1: Bit 3 | 1 |
| Mode indication | Byte 1: Bits 4-6 | 3 |
| Extended flag indicator1 | Byte 1: Bit 7 | 1 |
| Auxiliary sensor A | Byte 2: Bits 0-6 | 7 |
| Reserved | Byte 2: Bit 7 | 1 |
| Auxiliary sensor B | Byte 3: Bits 0-6 | 7 |
| Extended flag indicator2 | Byte 3: Bit 7 | 1 |

**3.3.3.1.2** The field contents are defined in Table 20.

**Table 20 — Definition of field contents**

| Contents | Definition |
|---|---|
| Coarse/fine indicator | 0 = Coarse |
| Control sensor | 1 °C resolution per bit<br>Relative to set-point<br>Range: 62 °C; – 32 °C to 30 °C<br>Offset: – 32 °C |
| Control sensor less Non-control sensor | 1 °C resolution per bit<br>Relative to control sensor<br>Range: 15 °C; – 8 °C to 7 °C<br>Offset: – 8 °C |
| Recent defrost indicator | 1 = Occurred in last interval |
| Mode indication | 000 = Full capacity<br>001 = Reduced capacity<br>010 = Capacity modulation<br>011 = Null<br>100 = Heat<br>101 = Defrost<br>110 = Loss of primary power<br>111 = Alarm/Shutdown |
| Extended flag indicator1 | This bit is set to 1 if it is a four sensor recording or if it is a two sensor recording and there are more data bytes to follow in this record |
| Auxiliary sensor A | 0,05 °C resolution per bit<br>Absolute temperature<br>Range: 6,35 °C; – 1,4 °C to 4,95 °C<br>Offset: – 1,4 °C |
| Reserved | Set to 1 |
| Auxiliary sensor B | 0,05 °C resolution per bit<br>Absolute temperature<br>Range: 6,35 °C; – 1,4 °C to 4,95 °C<br>Offset: – 1,4 °C |
| Extended flag indicator2 | Set to 1 for extended report |

**3.3.3.1.3** If an extended report is called for, an additional 4 bytes of information are sent after the previous 2 bytes for a two sensor system or 4 bytes for a four sensor system. The additional information is located as shown in Table 21.

**Table 21 — Location of additional fields**

| Contents | Location of bits | No. of bits |
|---|---|---|
| Set-point | Byte 4: Bits 0-7 to Byte 5: Bits 0-3 | 12 |
| Reserved | Byte 5: Bit 7 | 1 |
| Alarms | Byte 5: Bits 4-6<br>Byte 6: Bits 0-7<br>Byte 7: Bits 0-6 | 18 |
| Reserved set to 0 | Byte 7: Bit 7 | 1 |

**3.3.3.1.4** The field contents of these message bytes are defined in Table 22.

**Table 22 — Definition of field contents**

| Contents | Definition |
|---|---|
| Set-point | 0,5 °C resolution per bit<br>Range: 204,75 °C; – 154,75 °C to 50,0 °C<br>(set-point is restricted to 0 to 1 000 decimal)<br>Offset: – 154,75 °C |
| Alarms — Required | |
| High refrigerant pressure | Byte 5: Bit 4 = 1 |
| Temperature out of range | Byte 5: Bit 5 = 1 |
| High evaporator temperature | Byte 5: Bit 6 = 1 |
| Control sensor failure | Byte 6: Bit 0 = 1 |
| Non-control sensor failure | Byte 6: Bit 1 = 1 |
| Auxiliary sensor A failure | Byte 6: Bit 2 = 1 |
| Auxiliary sensor B failure | Byte 6: Bit 3 = 1 |
| Controller failure | Byte 6: Bit 4 = 1 |
| Defrost terminated on time limit | Byte 6: Bit 5 = 1 |
| Controller out of calibration | Byte 6: Bit 6 = 1 |
| Alarms — Optional | |
| Low compressor oil pressure | Byte 6: Bit 7 = 1 |
| Pumpdown terminated on time limit | Byte 7: Bit 0 = 1 |
| Microprocessor faulty | Byte 7: Bit 1 = 1 |
| Alarm reset | Byte 7: Bit 2 = 1 |
| Four miscellaneous alarms to be defined | |
| Reserved (signals end of extended report) | Byte 7: Bit 7 = 0 |

**3.3.3.2** ***Coarse temperature report — Low resolution***

The first 2 bytes of this report are the same as described in **3.3.3.1**. The field locations are also the same (see Table 19). However, the contents of the auxiliary sensor fields are as defined in Table 23.

**Table 23 — Definition of additional sensor fields — four sensor system, low resolution**

| Contents | Definition |
|---|---|
| Auxiliary sensor A | 0,5 °C resolution per bit<br>Absolute temperature<br>Range: 62 °C; – 32 °C to 30 °C<br>Offset: – 32 °C |
| Reserved | Set to 1 |
| Auxiliary sensor B | 0,5 °C resolution per bit<br>Absolute temperature<br>Range: 62 °C; – 32 °C to 30 °C<br>Offset: – 32 °C |
| Extended flag indicator2 | Set to 1 for extended report |

**3.3.3.3** ***Fine temperature report***

This example contains information for both a two and a four sensor system and normally takes up either 2 bytes or 4 bytes. Bytes 0 and 1 are identical for both two and four sensor systems. Bytes 2 and 3 are not recorded in a two sensor system.

**3.3.3.3.1** The locations of the various fields contained in each message are given in Table 24.

**Table 24 — Location of fields**

| Contents | Location of bits | No. of bits |
|---|---|---|
| Coarse/fine indicator | Byte 0: Bit 0 | 1 |
| Control sensor | Byte 0: Bits 1-5 | 5 |
| Control sensor less Non-control sensor | Byte 0: Bits 6-7 to Byte 1: Bits 0-2 | 5 |
| Recent defrost indicator | Byte 1: Bit 3 | 1 |
| Mode indication | Byte 1: Bits 4-6 | 3 |
| Extended flag indicator1 | Byte 1: Bit 7 | 1 |

**3.3.3.3.2** The field contents are defined in Table 25.

**Table 25 — Definition of field contents**

| Contents | Definition |
|---|---|
| Coarse/fine indicator | 1 = Fine |
| Control sensor | 0,25 °C resolution per bit<br>Relative to set-point<br>Range: 7,50 °C; – 3 °C to 4,50 °C<br>Offset: – 3 °C |
| Control sensor less Non-control sensor | 0,5 °C resolution per bit<br>Relative to control sensor<br>Range: 15 °C; – 8 °C to 7 °C<br>Offset: – 8 °C |
| Recent defrost indicator | 1 = Occurred in last interval |
| Mode indication | 000 = Full capacity<br>001 = Reduced capacity<br>010 = Capacity modulation<br>011 = Null<br>100 = Heat<br>101 = Defrost<br>110 = Lost of primary power<br>111 = Alarm/Shutdown |
| Extended flag indicator1 | Set to 1 for extended report |

### 3.3.4 Block 0 definitions

Block 0 is, by convention, a 128 byte buffer which contains parameters necessary to download and decode the actual trip record data. The trip record is fetched in 128 byte blocks starting with block 1 through to the end of the record. See Table 26.

**Table 26 — Definition of block 0 of the data logger**

| Address range (HEX) | Title | Use |
|---|---|---|
| 00/03 | Equipment type | 4 Character<br>Alpha field<br>Assigned by ISO |
| 04/07 | Buffer length | Length of ring<br>Bottom stored high/low<br>This number must be a multiple of 4 |
| 08/0F | Version number | Printable ASCII<br>Manufacturer specific |
| 10/1F | Serial number | Printable ASCII<br>Manufacturer specific |
| 20/23 | Current voyage pointer | Points to control word<br>Printable ASCII defining beginning of present voyage |
| 24/27 | Current reading pointer | Points to location of next reading to be recorded |
| 28/29 | Current recording interval | High byte/low byte in 1 min increments |
| 2A/54 | Reserved for future ISO use | |
| 55/67 | Manufacturer specific use | |
| 68/6F<br>66/77<br>78/7F | Extraction date<br>Extraction time<br>Extraction location | Optional printable ASCII fields written by the tracking device (MMU) to stump the record |

### 3.3.5 Event marks

Event marks are 6 bytes long not including the 4 byte preamble. The bytes are defined as follows.

Bytes 0, 1, 2: The time in minutes since the last set time [see header fields (**3.3.2**)]. Stored as an unsigned integer.

Byte 3: Event indicator.

Bytes 4, 5: Event extension.

## 3.4 Message definitions

### 3.4.1 Message packet exchange

A message packet exchange is defined to be a transmission of a command message from the MMU to the RCD, followed by a reply message packet from the RCD to the MMU.

The command message must have the command/reply field set to a value of zero (0). The reply message must have the command/reply field set to a value of one (1).

The message and sub-message fields define a specific operation to be performed by the RCD. The basic functions are as follows:

a) echo the command message;

b) read from controller function;

c) write to controller function;

d) controller enters an untimed process;

e) controller enters a timed process.

The operation and the contents of the message, sub-message and data fields for both the command and the reply messages are defined in **3.4.2** to **3.4.11**.

Table 27 contains a list of each function with the associated message and sub-message fields.

**Table 27 — Function cross-reference table**

| Function | Message field | Sub-message field |
|---|---|---|
| Echo | 0000 | XXX |
| Hardware read | 0001 | 000 |
| Software read | 0001 | 001 |
| Datalog read | 0001 | 010 |
| EPROM read | 0001 | 100 |
| RAM read | 0001 | 101 |
| Version read | 0001 | 110 |
| Non-volatile RAM read | 0001 | 111 |
| Hardware write | 0010 | 000 |
| Software write | 0010 | 001 |
| Datalog write | 0010 | 010 |
| Non-volatile RAM write | 0010 | 011 |
| Short pretrip test | 0011 | 111 |
| Long pretrip test | 0011 | 111 |
| All internal tests | 0011 | 111 |
| EPROM test | 0011 | 111 |
| EEPROM test | 0011 | 111 |
| RAM test | 0011 | 111 |
| Warm restart | 0011 | 111 |
| Calibrate sensors | 0011 | 111 |
| Change baud rate | 0100 | 111 |
| Disable controller | 0100 | 111 |
| Manufacturer specific | 0101 | XXX |
| Reserved for ISO | 1010<br>1111 | XXX |
| Illegal — SYNC character | 0110 | 001 |

### 3.4.2 Extended commands

An extended command mode is defined in which the data field of the message contains additional commands. This mode is defined by setting the sub-message field to $111_2$.

### 3.4.3 Notation

Each of the message fields is defined below. For the message type, sub-message type, and command/reply fields, a binary value is given. The least significant bit is the right-most bit in the field. This is the bit of the field that will be transmitted first over the serial communications link.

The letter X is used to indicate an undefined bit location. For command messages, the contents of this field bit location will be ignored by the RCD. For reply messages, the contents of this field bit location will be binary 0.

### 3.4.4 Echo

The echo function permits the MMU to test the integrity of the communications channel and to establish synchronization between the MMU and the RCD after changing baud rates.

At present, no extended command messages (i.e. sub-message field set to $111_2$) are defined for the echo function.

**3.4.4.1** ***Command message format***

Message field: $0000_2$
Sub-message field: $XXX_2$
Command/reply field: $0_2$
Data field length: $64_{10}$ $40_{16}$
Data field contents: The MMU may set the data field bytes to any desired value. This field is ignored by the RCD.

**3.4.4.2** ***Reply message format***

Message field: $0000_2$
Sub-message field: $XXX_2$
Command/reply field: $1_2$
Data field length: $64_{10}$ $40_{16}$
Data field contents: The RCD will echo the contents of the data field that was in the command message.

### 3.4.5 Controller read

The controller read message type contains six sub-message commands. These sub-message commands enable the RCD to read information about the controller hardware status, software status, data-logging data, EPROM contents and RAM contents.

At present, no extended command messages (i.e. sub-message field set to $111_2$) are defined for the controller read function.

**3.4.5.1** ***Hardware sub-message type***

The hardware read sub-message transfers the current state of the controller digital and analog inputs and outputs to the RCD.

**3.4.5.1.1** *Command message format*

Message field: $0001_2$
Sub-message field: $000_2$
Command/reply field: $0_2$
Data field length: $4_{10}$ $4_{16}$
Data field contents: Undefined. Ignored by the controller.

**3.4.5.1.2** *Reply message format*

Message field: $0001_2$
Sub-message field: $000_2$
Command/reply field: $1_2$
Data field length: $34_{10}$ $22_{16}$
Data field contents: See Figure 23.

**3.4.5.2** ***Software sub-message type***

The software read sub-message transfers various parameters of the controller software system to the RCD.

**3.4.5.2.1** *Command message format*

Message field: $0001_2$
Sub-message field: $001_2$
Command/reply field: $0_2$
Data field length: $4_{10}$ $4_{16}$
Data field contents: Undefined. Ignored by the controller.

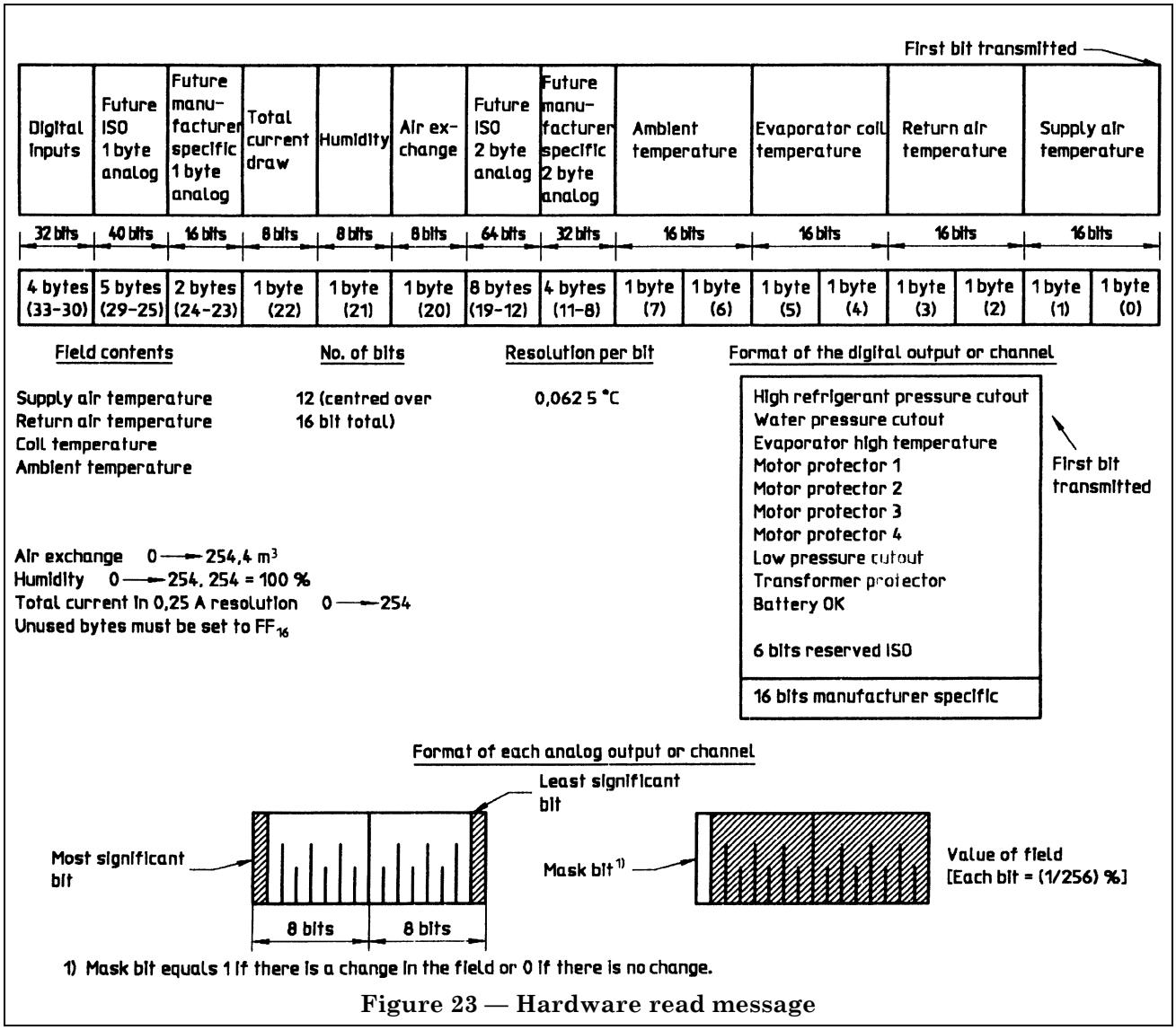



**Figure 23 — Hardware read message**

**3.4.5.2.2** *Reply message format*

Message field: $0001_2$

Sub-message field: $001_2$

Command/reply field: $1_2$

Data field length: $64_{10}$ $40_{16}$

Data field contents: Defined below.

Bytes 0 and 1 are the controller software version numbers. Byte 0 is the release number. Byte 1 is the revision number.

Bytes 2 and 3 are the set-point. This is a word in the same format as the analog channel inputs (see **3.4.6.1.2**). Byte 2 is the most significant byte.

Bytes 4 to 25 are the calibration offsets. The calibration offset is an 8 bit, 2's complement signed number with a resolution signal related to the channel. The range of temperature offset is – 4,0 °C to 4,0 °C. For example, a value of $20_{16}$ translates into an offset of 2,000 °C.

Other resolutions are 0,1 A for current measurements and 0,1 $m^3 h^{-1}$ for air exchange.

The order of the channels, in bytes 4 to 23, is listed in Table 28.

Bytes 26 to 41 are the active fault codes. Byte 26 is a fault code for the first fault that occurs. Any unused codes, when there are not 16 active faults, will be set to $00_{16}$. The codes are defined in Figure 7.

Byte 42 contains the current operating mode as defined in Figure 7.

Bytes 43 to 52 are reserved for future ISO use.

Bytes 53 to 63 are used for manufacturer specific information.

**3.4.5.3** ***Data-logging data sub-message type***

The data-logging data sub-message is used to read the data that the controller has logged.

**3.4.5.3.1** *Command message format*

Message field: $0001_2$
Sub-message field: $010_2$
Command/reply field: $0_2$
Data field length: $4_{10}$ $4_{16}$
Data field contents: Defined below.

The command message data field contains the block number of the desired data. A block is defined to be $128_{10}$ $80_{16}$ bytes. Block 0, by convention, will be the pointer information instead of data. The format of block 0 is defined in **3.3.4** and Table 26.

Byte 0 of the data field is the high byte of the block number. Byte 1 is the low byte of the block count. Bytes 2 and 3 are undefined and will be ignored by the controller.

The contents of non-zero block number blocks are defined by the data-logging specification given in **3.3**.

**3.4.5.3.2** *Reply message format*

Message field: $0001_2$
Sub-message field: $010_2$
Command/reply field: $1_2$
Data field length: $128_{10}$ $80_{16}$
Data field contents: Defined below.

The contents of the data field will be the block of $128_{10}$ $80_{16}$ bytes from the data-logging data stored in the ring buffer.

**3.4.5.4** ***EPROM sub-message type***

This sub-message type transfers a $128_{10}$ $80_{16}$ byte block of controller EPROM to the RCD.

**3.4.5.4.1** *Command message format*

Message field: $0001_2$
Sub-message field: $100_2$
Command/reply field: $0_2$
Data field length: $4_{10}$ $4_{16}$
Data field contents: Defined below.

The command message data field contains the block number of the desired data. A block is defined to be $128_{10}$ $80_{16}$ bytes. Byte 0 of the data field is the high byte of the block number. Byte 1 is the low byte of the block count. Bytes 2 and 3 are undefined and will be ignored by the controller.

**3.4.5.4.2** *Reply message format*

Message field: $0001_2$
Sub-message field: $100_2$
Command/reply field: $1_2$
Data field length: $128_{10}$ $80_{16}$
Data field contents: Defined below.

The contents of the data field will be a block of $128_{10}$ $80_{16}$ bytes from the controller EPROM memory. Note that this is the program store for the controller. Currently, only $32768_{10}$ $8000_{16}$ bytes of EPROM exist in the system. This means that only block numbers 0 to $255_{10}$ (0 to $FF_{16}$) are valid. The controller will return a data field filled with $00_{16}$ for any block number that is out of the currently implemented range.

**3.4.5.5** ***RAM sub-message type***

The RAM sub-message type transfers a $128_{10}$ $80_{16}$ byte block of controller RAM to the RCD.

**3.4.5.5.1** *Command message format*

Message field: $0001_2$

Sub-message field: $101_2$

Command/reply field: $0_2$

Data field length: $4_{10}$ $4_{16}$

Data field contents: Defined below.

This command message data field contains the block number of the desired data. A block is defined to be $128_{10}$ $80_{16}$ bytes. Byte 0 of the data field is the high byte of the block number. Byte 1 is the low byte of the block count. Bytes 2 and 3 are undefined and will be ignored by the controller.

**3.4.5.5.2** *Reply message format*

Message field: $0001_2$

Sub-message field: $101_2$

Command/reply field: $0_2$

Data field length: $128_{10}$ $80_{16}$

Data field contents: Defined below.

The contents of the data field will be a block of $128_{10}$ $80_{16}$ bytes from the controller RAM memory. Currently, only $2048_{10}$ $800_{16}$ bytes of RAM exist in the system. This means that only block numbers 0 to $15_{10}$ (0 to $F_{16}$) are valid. The controller will return a data field filled with $00_{16}$ for any block number that is out of the currently implemented range.

The contents of the RAM memory, when the controller is operating as a controller, are not defined here. This information can be found in the controller software listings and documentation.

**3.4.6 Controller write**

The controller write message type contains three sub-message commands. These sub-message commands enable the RCD to write information to the controller hardware ports, software status, and data-logging header data.

At present, no extended command messages (i.e. sub-message field set to $111_2$) are defined for the controller write function.

**3.4.6.1** ***Hardware sub-message type***

The hardware write sub-message transfers a desired state of the digital and analog outputs from the master to the slave.

Depending on the operating state of the controller, the outputs may or may not be latched. When the controller is in the control mode, it is constantly updating the outputs, i.e. setting outputs either on or off. If a hardware write message is sent, the controller will set the outputs according to the information in the hardware write message, but will then change the outputs at the next update of the outputs.

If the controller is in the disabled state, the hardware state will be set by the hardware write message. This state will continue until either the disabled state times out or the RCD transmits a new hardware write message to the controller.

**3.4.6.1.1** *Command message format*

Message field: $0010_2$

Sub-message field: $000_2$

Command/reply field: $0_2$

Data field length: $18_{10}$ $24_{16}$

Data field contents: See Figure 24.

The output write data field has two sub-fields. The first sub-field is the desired state of the digital and analog outputs. The second sub-field is a mask field which enables or disables the output states in the first sub-field. A binary 1 in the mask field enables the changing of the corresponding output port bit. A binary 0 in the mask field disables the changing of the corresponding output port bit.

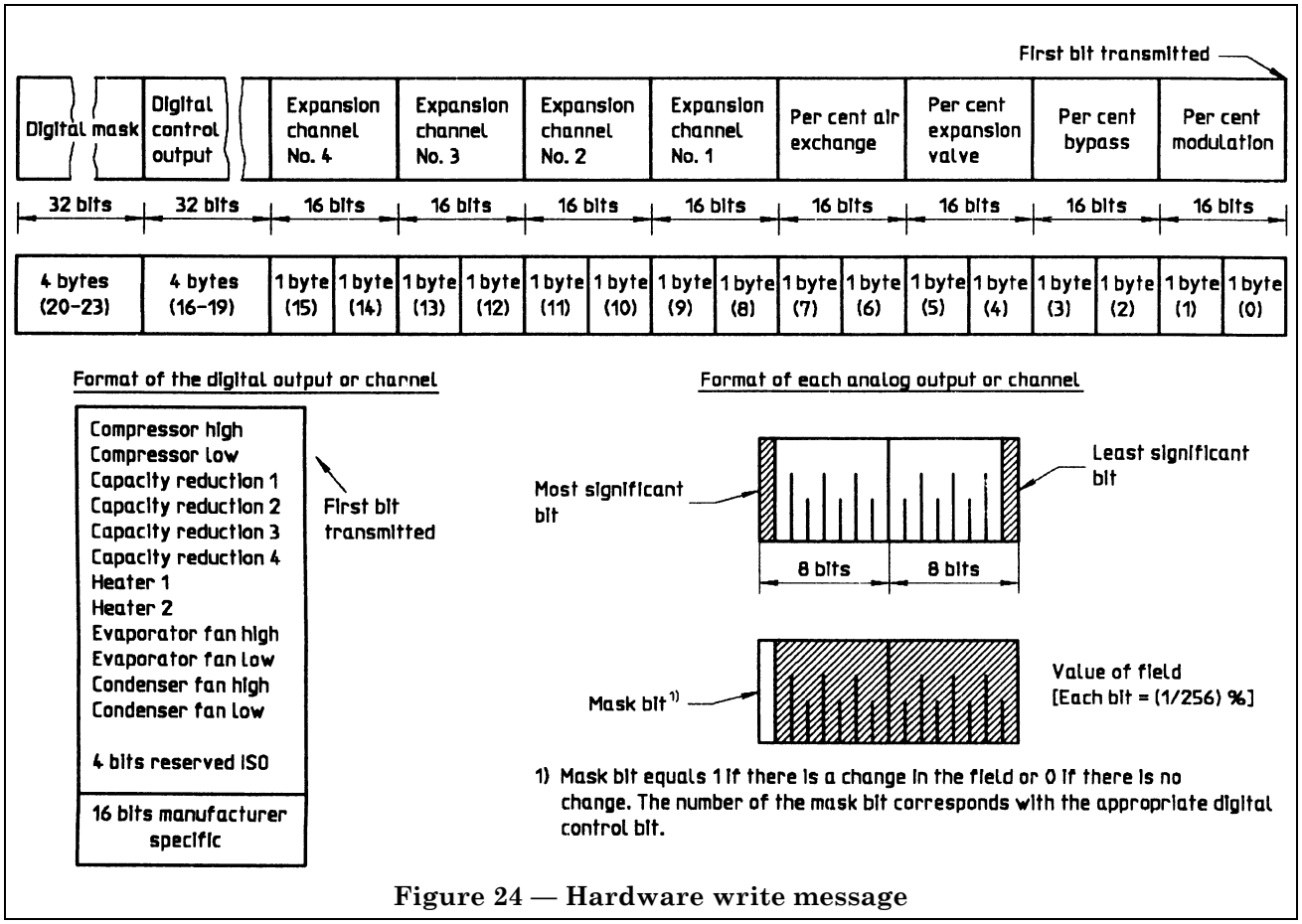



**Figure 24 — Hardware write message**

**3.4.6.1.2** *Reply message format*

Message field: $0010_2$

Sub-message field: $000_2$

Command/reply field: $1_2$

Data field length: $18_{16}$ $24_{10}$

Data field contents: Echo of the data field contents of the command message.

**3.4.6.2** ***Software sub-message type***

The software write sub-message type enables the master to change the slave set-point, defrost time interval, defrost timer, and data-logging interval.

**3.4.6.2.1** *Command message format*

Message field: $0010_2$

Sub-message field: $001_2$

Command/reply field: $0_2$

Data field length: $17_{10}$ $11_{16}$

Data field contents: Defined below.

Bytes 0 and 1 are the desired set-point, in the format defined in **3.4.5.1.2**.

Byte 2, bits 0 to 6, is the defrost timer interval. The currently valid contents for this byte are 2, 4, 6 or 8 for 2 h, 4 h, 6 h or 8 h respectively. All other values will be ignored by the slave.

Byte 2, bit 7, is the most significant bit of the defrost timer, which is a 9 bit timer.

Byte 3 is the least significant 8 bits of the defrost timer. Bit 0 is the least significant bit of the timer. The defrost timer has a resolution of 1 min.

Byte 4 is the data-logging time interval. At present, this byte is ignored by the controller.

Bytes 5 to 14 are undefined and are ignored by the controller.

Bytes 15 and 16 are the mask bytes that apply to the software write. These bytes are defined as follows.

| | | |
|---|---|---|
| Byte 15 | bit 0 | set-point mask |
| | bit 1 | defrost timer interval mask |
| | bit 2 | defrost timer mask |
| | bit 3 | data-logging timer mask |
| | bit 4 | undefined, ignored by controller |
| | bit 5 | undefined, ignored by controller |
| | bit 6 | undefined, ignored by controller |
| | bit 7 | undefined, ignored by controller |
| Byte 16 | bit 0 | undefined, ignored by controller |
| | bit 1 | undefined, ignored by controller |
| | bit 2 | undefined, ignored by controller |
| | bit 3 | undefined, ignored by controller |
| | bit 4 | undefined, ignored by controller |
| | bit 5 | undefined, ignored by controller |
| | bit 6 | undefined, ignored by controller |
| | bit 7 | undefined, ignored by controller |

**3.4.6.2.2** *Reply message format*

| | |
|---|---|
| Message field: | $0010_2$ |
| Sub-message field: | $001_2$ |
| Command/reply field: | $0_2$ |
| Data field length: | $17_{10}$ $10_{16}$ |
| Data field contents: | Echo of the data field contents of the command message. |

**3.4.6.3** ***Data-logging header sub-message type***

The data-logging header sub-message is used to transfer a new header from the RCD to the controller.

**3.4.6.3.1** *Command message format*

| | |
|---|---|
| Message field: | $0010_2$ |
| Sub-message field: | $010_2$ |
| Command/reply field: | $0_2$ |
| Data field length: | $128_{10}$ $80_{16}$ |
| Data field contents: | The controller will write the data field into the ring buffer as a data-logging header. The embedded control word will be used to determine subsequent reading formats. |

**3.4.6.3.2** *Reply message format*

| | |
|---|---|
| Message field: | $0010_2$ |
| Sub-message field: | $010_2$ |
| Command/reply field: | $1_2$ |

Data field length: $128_{10}$ $80_{16}$

Data field contents: Echo of the data field contents of the command message.

**3.4.7 Enter untimed process**

A process is a controller operating state which is not the normal system control state. An example of a process is pretrip. There are two types of processes, i.e. timed (see **3.4.8**) and untimed.

An untimed process is one where the controller ends the process at the end of a predetermined sequence of operations and returns to the controlling state.

At present, all untimed process messages use the extended commands mode to write commands.

**3.4.7.1** ***Command message format***

Message field: $0011_2$

Sub-message field: $111_2$

Command/reply field: $0_2$

Data field length: $4_{10}$ $4_{16}$

Data field contents: Defined in **3.4.7.1.1** to **3.4.7.1.8**.

**3.4.7.1.1** *Execute diagnostics — Short pretrip test*

data field byte 0: $00_{10}$ $00_{16}$

data field byte 1: $00_{10}$ $00_{16}$

data field byte 2: $XX_{10}$ $XX_{16}$

data field byte 3: $XX_{10}$ $XX_{16}$

This is the pretrip test that can be initiated from the keyboard.

**3.4.7.1.2** *Execute diagnostics — Long pretrip test*

data field byte 0: $00_{10}$ $00_{16}$

data field byte 1: $01_{10}$ $01_{16}$

data field byte 2: $XX_{10}$ $XX_{16}$

data field byte 3: $XX_{10}$ $XX_{16}$

**3.4.7.1.3** *Execute diagnostics — All internal tests*

data field byte 0: $00_{10}$ $00_{16}$

data field byte 1: $02_{10}$ $02_{16}$

data field byte 2: $XX_{10}$ $XX_{16}$

data field byte 3: $XX_{10}$ $XX_{16}$

This command causes the controller to execute all the built-in diagnostic tests. Since the last test is the RAM test, which will erase the RAM data, the controller will execute a warm restart at the end of this test.

**3.4.7.1.4** *Execute diagnostics — Internal EPROM test*

data field byte 0: $00_{10}$ $00_{16}$

data field byte 1: $03_{10}$ $03_{16}$

data field byte 2: $XX_{10}$ $XX_{16}$

data field byte 3: $XX_{10}$ $XX_{16}$

This test checks that the EPROM check sum is correct.

**3.4.7.1.5** *Execute diagnostics — Internal NOVRAM test*

data field byte 0: $00_{10}$ $00_{16}$

data field byte 1: $04_{10}$ $04_{16}$

data field byte 2: $XX_{10}$ $XX_{16}$

data field byte 3: $XX_{10}$ $XX_{16}$

**3.4.7.1.6** *Execute diagnostics — Internal RAM test*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $05_{10}$ $05_{16}$
data field byte 2: $XX_{10}$ $XX_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This test over-writes the controller RAM and will cause the controller to execute a warm restart.

**3.4.7.1.7** *Execute a warm restart*

data field byte 0: $01_{10}$ $01_{16}$
data field byte 1: $XX_{10}$ $XX_{16}$
data field byte 2: $XX_{10}$ $XX_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This command will cause the controller to execute a warm restart.

**3.4.7.1.8** *Calibrate sensors*

data field byte 0: $02_{10}$ $02_{16}$
data field byte 1: $S_{10}$ $S_{16}$
data field byte 2: $HH_{10}$ $HH_{16}$
data field byte 3: $LL_{10}$ $LL_{16}$

This command is used to calibrate a sensor. The value that the sensor should be reading is $HHLL_{16}$, where HH represents the high order byte and LL represents the low order byte for the analog channel format obtained in **3.4.5.1.2**.

The value S is the sensor number. Table 28 shows the analog channels and their channel numbers. All other numbers will be ignored by the controller.

**Table 28 — Controller sensor channel identification numbers**

| S | Channel |
|---|---|
| 0 | Supply air |
| 1 | Return air |
| 2 | Evaporator coil |
| 3 | Ambient |
| 4-7 | ISO spare 2 byte channels |
| 8-11 | Manufacturer specific spare 2 byte channels |
| 12 | Air exchange |
| 13 | Humidity |
| 14 | Total current |
| 15-19 | ISO spare 1 byte channels |
| 20-21 | Manufacturer specific spare 1 byte channels |

**3.4.7.2** ***Reply message format***

For all diagnostic replies, the format contains an RR. This is the location where the controller is returning the results for the diagnostic tests. If RR is $01_{16}$, the test passed. If RR is $02_{16}$, the test failed.

For diagnostic tests in which the communications link is going to be broken, the results are related to the diagnostics last executed, i.e. at the last restart.

Message field: $0011_2$
Sub-message field: $111_2$
Command/reply field: $1_2$
Data field length: $4_{10}$ $4_{16}$
Data field contents: Defined in **3.4.7.2.1** to **3.4.7.2.8**.

**3.4.7.2.1** *Execute diagnostics — Short pretrip test*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $00_{10}$ $00_{16}$
data field byte 2: $RR_{10}$ $RR_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This reply message will be sent before the controller enters the short pretrip test.

**3.4.7.2.2** *Execute diagnostics — Long pretrip test*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $01_{10}$ $01_{16}$
data field byte 2: $RR_{10}$ $RR_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This reply message will be sent before the controller enters the long pretrip test.

**3.4.7.2.3** *Execute diagnostics — All internal tests*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $02_{10}$ $02_{16}$
data field byte 2: $RR_{10}$ $RR_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This reply relates to the results of the last set of tests executed.

This reply message will be sent before the controller enters the diagnostic tests.

**3.4.7.2.4** *Execute diagnostics — Internal EPROM test*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $03_{10}$ $03_{16}$
data field byte 2: $RR_{10}$ $RR_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This test checks that the EPROM check sum is correct.

This reply message will be sent after the controller performs the EPROM test.

**3.4.7.2.5** *Execute diagnostics — Internal EEPROM test*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $04_{10}$ $04_{16}$
data field byte 2: $RR_{10}$ $RR_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This test checks the validity of the EEPROM control area data. If does not lest any of the data-logging EEPROM data area.

This reply message will be sent after the controller performs the EEPROM test.

**3.4.7.2.6** *Execute diagnostics — Internal RAM test*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $05_{10}$ $05_{16}$
data field byte 2: $RR_{10}$ $RR_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This reply relates to the results of the last RAM test executed.

This reply message will be sent before the controller enters the RAM diagnostic test.

**3.4.7.2.7** *Execute a warm restart*

data field byte 0: $01_{10}$ $01_{16}$
data field byte 1: $XX_{10}$ $XX_{16}$
data field byte 2: $XX_{10}$ $XX_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This reply is sent before the warm restart is executed.

**3.4.7.2.8** *Calibrate sensors*

data field byte 0: $02_{10}$ $02_{16}$
data field byte 1: $S_{10}$ $S_{16}$
data field byte 2: $YY_{10}$ $YY_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

This informs the RCD what sensor offset was actually loaded. The offset, YY, is in the format defined in **3.4.5.2.2**.

This command is used to calibrate a sensor. The value that the sensor should be reading is $HHLL_{16}$, where HH represents the high order byte and LL represents the low order byte for the analog channel format obtained in **3.4.5.1.2**.

The value S is the sensor number (see Table 28).

**3.4.8 Enter timed process**

A process is a controller operating state which is not the normal system control state. An example of a process is pretrip. There are two types of processes, i.e. timed and untimed (see **3.4.7**).

A timed process is one whose end is determined by a timer (which was set when the controller entered the process) times out.

At present, all timed process messages use the extended commands mode to write commands.

The controller will abort the process after 15 min.

**3.4.8.1** ***Command message format***

Message field: $0100_2$
Sub-message field: $111_2$
Command/reply field: $0_2$
Data field length: $4_{10}$ $4_{16}$
Data field contents: Defined in **3.4.8.1.1** and **3.4.8.1.2**.

**3.4.8.1.1** *Change baud rate*

data field byte 0: $00_{10}$ $00_{16}$
data field byte 1: $R_{10}$ $RT_{16}$
data field byte 2: $XX_{10}$ $XX_{16}$
data field byte 3: $XX_{10}$ $XX_{16}$

RT is a byte which indicates which baud rate the controller is to use. The valid baud rate values for RT are specified in Table 29. All other values will be ignored by the controller.

**Table 29 — Baud rate definitions**

| RT | New baud rate |
|---|---|
| 00 | Default rate |
| 01 | 75 baud |
| 02 | 110 baud |
| 03 | 300 baud |
| 04 | 600 baud |
| 05 | 1 200 baud |
| 06 | 2 400 baud |
| 07 | 4 800 baud |
| 08 | 9 600 baud |
| 09 | 19,2 kilobaud |
| 10 | 38,4 kilobaud |
| 11 | 67,2 kilobaud |
| 12 | 100 kilobaud |

The default baud rate is the baud rate that the controller uses after a controller reset. The controller will remain in the default baud rate after the process has timed out.

This reply is sent before the slave changes baud rates.

**3.4.8.1.2** *Disable controller*

data field byte 0: $01_{10}\ 01_{16}$
data field byte 1: $XX_{10}\ XX_{16}$
data field byte 2: $XX_{10}\ XX_{16}$
data field byte 3: $XX_{10}\ XX_{16}$

This command will cause the controller to enter a disabled state. In this state, the controller does not update the digital or analog outputs, nor does it check the inputs for fault conditions. It does perform digital and analog input scanning.

The RCD may change the digital and analog outputs by using the hardware write message.

**3.4.8.2** ***Reply message format***

Message field: $0100_2$
Sub-message field: $111_2$
Command/reply field: $1_2$
Data field length: $4_{10}\ 4_{16}$
Data field contents: The reply message will echo the contents of the command message data field.

**3.4.9 Undefined**

**3.4.9.1** ***Command message format***

Message field: $0101_2$ to $1100_2$
Sub-message field: $XXX_2$
Command/reply field: $0_2$
Data field length: Undefined.
Data field contents: Undefined.

**3.4.9.2** ***Reply message format***

Message field: $0101_2$ to $1100_2$
Sub-message field: $XXX_2$
Command/reply field: $1_2$
Data field length: Undefined.
Data field contents: Undefined.

#### 3.4.10 Reserved

The reserved message types are reserved for ISO for possible present or future uses.

##### 3.4.10.1 *Command message format*

Message field: $1101_2$ to $1111_2$
Sub-message field: $XXX_2$
Command/reply field: $0_2$
Data field length: Undefined.
Data field contents: Undefined.

##### 3.4.10.2 *Reply message format*

Message field: $1101_2$ to $1111_2$
Sub-message field: $XXX_2$
Command/reply field: $1_2$
Data field length: Undefined.
Data field contents: Undefined.

#### 3.4.11 Illegal

The illegal message type prevents a message field from being identified incorrectly as part of the SYNC field.

##### 3.4.11.1 *Command message format*

Message field: $0001_2$
Sub-message field: $011_2$
Command/reply field: $0_2$
Data field length: Undefined.
Data field contents: Undefined.

##### 3.4.11.2 *Reply message format*

Message field: $1101_2$ to $1111_2$
Sub-message field: $XXX_2$
Command/reply field: $1_2$
Data field length: Undefined.
Data field contents: Undefined.

### 3.5 Low data rate physical requirements — LDCU to LRCD

#### 3.5.1 Frequency

The power line signals will be sent at 55 kHz with a variation of ± 2 % (i.e. in the range 53,9 kHz to 56,1 kHz).

#### 3.5.2 Modulation method

The messages will be frequency shift keyed (FSK). Each character transferred requires one start bit (low logic level), eight data bits and one stop bit (high logic level). A packet transfer includes a start preamble of three transitions followed by ten logic one bits. This preamble restricts the deadtime between any two bytes in a packet to be less than one-half bit length.

#### 3.5.3 Baud rate

The communications will be sent at 1 200 baud or 1 200 bits $s^{-1}$.

#### 3.5.4 Transmission mode

The signals will be sent half duplex. The signals will be transmitted at 6 V r.m.s. maximum into a line impedance of (15 + j15) Ω.

**3.5.5 Injection mode**

The injection mode of the signal is "voltage injection", the difference being signal levels between the three phases and the ground.

**3.5.6 Receiver sensitivity**

The receiver sensitivity is 1 mV r.m.s. at 55 kHz ± 10 kHz.

**3.5.7 Non-transmission impedance**

The signal shall be transmitted over lines with impedance levels of less than 3 kΩ at 55 kHz.

**3.5.8 Bit synchronization**

The messages sent are asynchronous.

**3.5.9 Carrier setup time**

The signal setup time is 10 ms.

**3.5.10 Out-of-band filtering requirements for "high data rate compatibility"**

The out-of-band power spectrum shall not exceed 2 mV r.m.s. from 130 kHz to 400 kHz and shall not exceed the linear plot shown in Figure 25 of decibel volts against frequency from 180 mV r.m.s. at 70 kHz to 2 mV r.m.s. at 130 kHz as measured into an 18 Ω resistive load in any 10 kHz bandwidth.

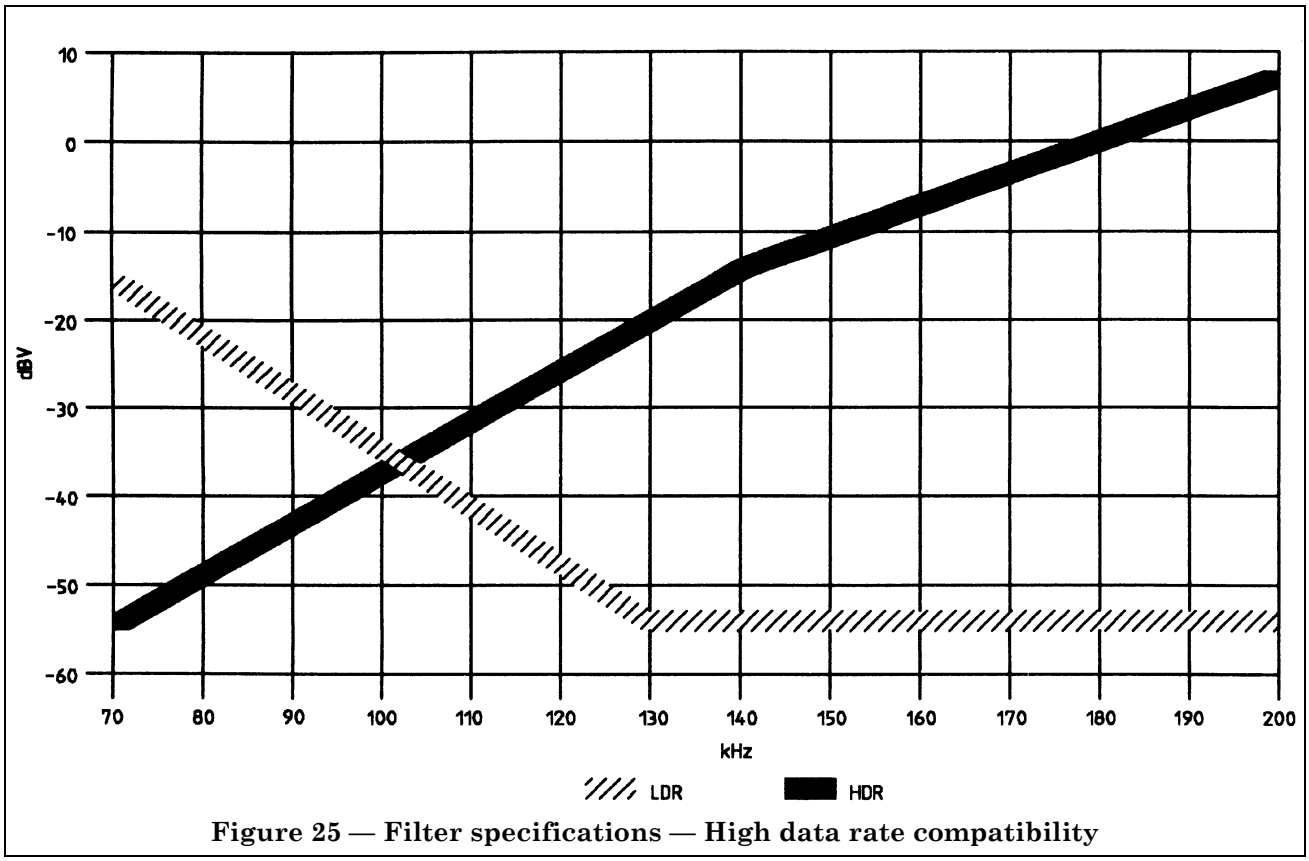



**Figure 25 — Filter specifications — High data rate compatibility**

## 3.6 High data rate physical requirements — HDCU to HRCD

**3.6.1 Modulation method — Broad band**

The transmitter shall generate a broad band signal to conform to the signal defined by **3.6.1.1** and **3.6.1.2** in the signal bandwidth of 140 kHz to 400 kHz.

**3.6.1.1** ***Spreading method***

An anti-symmetric waveform is digitally synthesized using 16 bits (chips) of a 4,300 8 MHz clock. Two periods of the anti-symmetric waveform (corresponding to 32 chips of the clock) define one raw data bit interval. The levels of the 16 chips may be chosen in any manner such that the resulting waveform is anti-symmetric and the power spectrum of one raw data bit matches that specified in **3.6.5**.

**3.6.1.2** ***Raw data bit modulation***

Each raw data bit interval is binary phase shift keyed (BPSK) using a non-return-to-zero level (NRZ-L) format, resulting in a raw data bit rate of 134,4 kbit $s^{-1}$.

**3.6.2 Transmission mode**

There is a maximum output power of 100 mW per 10 kHz bandwidth over a bandwidth of 140 kHz to 400 kHz for a line impedance of 18 Ω.

**3.6.3 Injection mode**

There is line-impedance-controlled voltage injection between three phases and ground.

**3.6.4 Output/input impedance**

The output impedance shall not exceed 18 Ω phase-to-phase and 21 Ω phase-to-ground. The input impedance (non-transmit) shall be greater than 200 Ω.

**3.6.5 Power density function**

The null-to-null bandwidth of one raw data bit shall be from a maximum of 140 kHz to a minimum of 400 kHz. The null-to-null bandwidth of transmitted data shall be from 140 kHz to 400 kHz. The power density function of transmitted data shall be as shown in Figure 26.

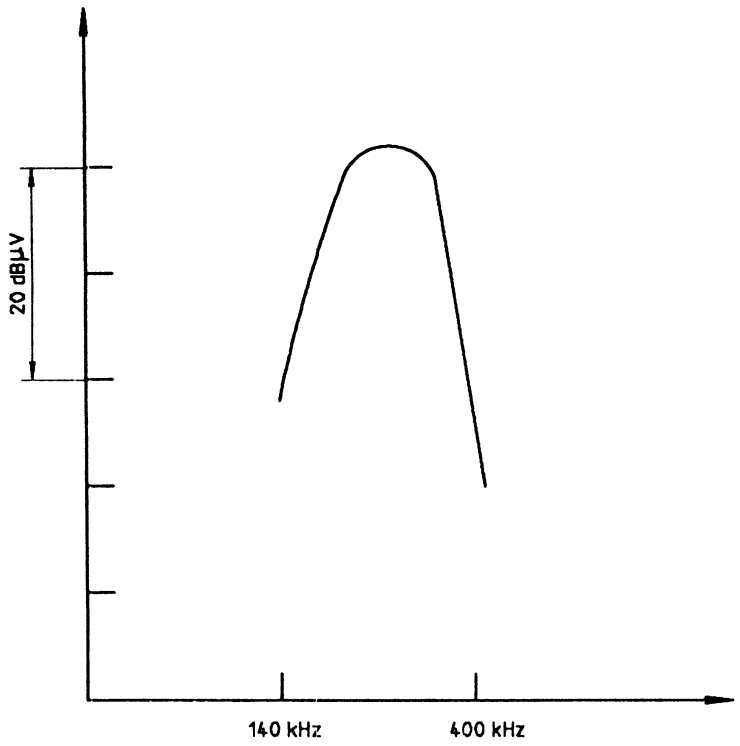



**Figure 26 — Power density function**

**3.6.6 Synchronization method**

The receiver shall synchronize to the synchronization preamble defined in **3.6.9.1.1**. The preamble synchronization time shall not exceed 0,3 ms from the start of the preamble.

**3.6.7 Demodulation method**

The receiver shall demodulate raw data as modulated in **3.6.1**. The receiver may use any method of equalization that can correct for an attenuation of at least 15 dB over 60 kHz.

**3.6.8 Receiver sensitivity**

The receiver shall operate with less than a 2 mV received signal amplitude.

**3.6.9 Data link protocol**

Codes for data link fields are given in Table 30. (32,8) ECC and EDC codes are given in Table 31.

**3.6.9.1** ***Frame formats***

**3.6.9.1.1** *Preamble*

The preamble delimits the start of a valid message. It contains the necessary sequences to synchronize communications between devices on the power line.

|SYNC|SYNC|SYNC|SYNC|SOF|-- total of 5 bytes

where

SYNC is the synchronization sequence;

SOF is the start of frame field which may be SOFLong or SOFShort.

Each field is 1 byte in length.

Preamble(L) denotes a Preamble with a SOFLong field.

Preamble(S) denotes a Preamble with a SOFShort field.

**3.6.9.1.2** *Data frames*

**3.6.9.1.2.1** *Unicast data frames*

CONNECT and FIRSTFrameReTx signal the beginning of a packet. The first transmission of a packet uses a CONNECT. Subsequent retries of the same packet use FIRSTFrameReTx.

|Preamble(L)|HDRLong|DSTHigh|DSTLow|SRCHigh|SRCLow|-- total of 25 bytes

where

HDRLong is the long (encoded) header field (= CONNECT or FIRSTFrameReTx);

DSTHigh is the destination address field high-order byte;

DSTLow is the destination address field low-order byte;

SRCHigh is the source address field high-order byte;

SRCLow is the source address field low-order byte.

Each field is encoded by (32,8) ECC and EDC.

CONTOdd is the frame succeeding a CONNECT or FIRSTFrameReTx. It contains the remaining packet header fields and the first three bytes of the data field.

|Preamble(S)|HDRShort|NDBHigh|NDBLow|DATA1|DATA2|DATA3|
-- total of 26 bytes

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= CONTOdd);

NDBHigh is the number of data bytes field high-order byte;

NDBLow is the number of data bytes field low-order byte;

DATA1, ..., DATA3 are data fields.

Each field (non-header) is encoded by (32,8) ECC and EDC.

**3.6.9.1.2.2** *Broadcast data frames*

BRDHdr signals the beginning of a new broadcast packet. The BRDHdr frame is repeated eight times to allow all receiving units an ample opportunity to initiate a broadcast packet reception since receiver requests for retransmission are not possible. The sequence field is initialized to 00 and is incremented for each subsequent transmission.

|Preamble(L)|HDRLong|Sequence|-- total of 13 bytes

where

HDRLong is the long (encoded) header field (= BRDHdr);

Sequence is the repetition count of consecutive BRDHdr transmissions and ranges in value from 00 to 07.

Each field is encoded by (32,8) ECC and EDC.

CONTOdd is the frame succeeding the BRDHdr. It contains the remaining packet header fields and the first byte of the data field.

|Preamble(S)|HDRShort|SRCHigh|SRCLow|NDBHigh|NDBLow|DATA1|
-- total of 26 bytes

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= CONTOdd);

SRCHigh is the source address field high-order byte;

SRCLow is the source address field low-order byte;

NDBHigh is the number of data bytes field high-order byte;

NDBLow is the number of data bytes field low-order byte;

DATA1 is a data field.

Each field (non-header) is encoded by (32,8) ECC and EDC.

**3.6.9.1.2.3** *Unicast/broadcast data frames*

Unicast/broadcast data frames alternate between CONTEven and CONTOdd for consecutive frames. A frame can contain 5 bytes of the data field for intermediate frames, and 0 to 4 bytes of the data field for the last frame of the packet. The last byte of the last frame shall contain the XOR check sum field.

|Preamble(S)|HDRShort|DATA1|. . .|DATA5|-- total of 26 bytes (intermediate frames)

|Preamble(S)|HDRShort|DATA1|. . .|DATA4|X0R|
-- total of 10 bytes to 26 bytes (last frame)

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= CONTEven or CONTOdd);

DATA1, ..., DATA5 are data fields;

XOR is the XOR check sum field exclusive-or of DST, SRC, NDB, and all DATA fields.

Each field (non-header) is encoded by (32,8) ECC and EDC.

**3.6.9.1.3** *Control transfer frames*

PASSControl represents the attempt of the master to poll a slave unit.

|Preamble(L)|HDRLong|DSTHigh|DSTLow|-- total of 17 bytes

where

HDRLong is the long (encoded) header field (= PASSControl);

DSTHigh is the destination address field high-order byte;

DSTLow is the destination address field low-order byte.

Each field is encoded by (32,8) ECC and EDC.

**3.6.9.1.4** *Response frames*

**3.6.9.1.4.1** *General responses*

LONGAck is the response to one of the following:

a) the last frame of a data packet, signalling the end of the current packet;

b) a FIRSTFrameReTx data frame, indicating the detection of the packet as a duplicate by the receiver;

c) a PASSControl message from the master, signalling the need of the slave to acquire control to send data to the master.

| Preamble(S) | HDRShort | HDRLong | SRCHigh | SRCLow | -- total of 18 bytes

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= NOTAck);

HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC(= LONGAck);

SRCHigh is the source address field high-order byte:

SRCLow is the source address field low-order byte.

Each SRC field is encoded by (32,8) ECC and EDC.

NAK is the response of the receiving unit indicating that it has detected errors in a received data frame or the response of a slave to a PASSControl message from the master when the slave does not have data to send.

| Preamble(S) | HDRShort | HDRLong | -- total of 10 bytes

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= NOTAck);

HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC (= NAK).

**3.6.9.1.4.2** *Data frame only responses*

CONNAck is the response to a CONNECT indicating that it has successfully received the first frame of the packet.

| Preamble(S) | HDRShort | HDRLong | -- total of 10 bytes

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= NOTAck);

HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC (= CONNAck).

ACK is the response of the receiving unit indicating that it has successfully received a data frame.

| Preamble(S) | HDRShort | -- total of 6 bytes

where HDRShort is the short (non-encoded) header field of 1 byte in length (= ACK).

BUSYNak is the response to a CONNECT or the CONTOdd following a CONNECT or FIRSTFrameReTx indicating a temporary lack of buffer resources at the receiving unit.

| Preamble(S) | HDRShort | HDRLong | -- total of 10 bytes

where

HDRShort is the short (non-encoded) header field of 1 byte in length (= NOTAck);

HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC (BUSYNak).

**3.6.9.1.4.3** *Control transfer responses*

PASSControlHandshake succeeds the acknowledgement of a PASSControl message by the receiving unit (a slave). The PASSControlHandshake signals the slave to start transmitting its data.

| Preamble(L) | HDRLong | -- total of 9 bytes

where HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC (= PASSControlHandshake).

RETURNControl follows the transmission of a data packet by a slave which has been temporarily transferred control by the master. RETURNControl signals the master that it can resume control of the network.

| Preamble(L) | HDRLong | -- total of 9 bytes

where HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC (= RETURNControl).

**3.6.9.1.5** *Control frames*

ACTIVITYFrame is transmitted by the master to let all other attached units (slaves) know that there is an operating master on the power line network. The master transmits this control frame while in the idle state, i.e. while not engaged in a data packet exchange or control exchange. The master will wait 1 ms between successive ACTIVITYFrames.

| Preamble(L) | HDRLong | -- total of 9 bytes

where HDRLong is the long (encoded) header field encoded by (32,8) ECC and EDC (= ACTIVITYFrame).

**3.6.9.2** ***Unicast data packet exchange***

**3.6.9.2.1** *First frame exchange*

First frame exchange describes the initial exchange of frames for a new packet. When the receiving unit acknowledges the first frame, the exchange continues; otherwise, an attempt is made at retransmitting the first frame.

**3.6.9.2.1.1** *Without retransmission*

```
|CONNECT|--------->
         <---------|CONNAck|
```

**3.6.9.2.1.2** *With retransmission*

```
|CONNECT|--------->
         <---------|NAK|
|CONNECT|--------->
         <---------|CONNAck|
```

**3.6.9.2.1.3** *With duplicate detection*

```
|FIRSTFrameReTx|--------->
                <---------|LONGAck|

     <<<<< end of data packet exchange >>>>>
```

The transmitter only uses a FIRSTFrameReTx if, during a previous attempt at the packet transmission, it transmitted the last frame. If the last frame was not transmitted, the receiver could not have successfully received the entire data packet, and a CONNECT should be used for the packet retransmission.

**3.6.9.2.1.4** *With lack of receiver buffer resources*

```
|CONNECT|--------->
         <---------|CONNAck|
|CONTOdd|--------->
         <---------|BUSYNak|

    <<<<< end of data packet exchange >>>>>
```

Alter receiving the NDB fields in the CONTOdd following the CONNECT, the receiver determines that it has insufficient buffer resources to accommodate the data packet. It indicates this to the transmitter by responding with a BUSYNak response frame.

```
|CONNECT|--------->
         <---------|BUSYNak|

    <<<<< end of data packet exchange >>>>>
```

After receiving the CONNECT, the receiver determines that it has insufficient buffer resources to accommodate even a single byte data packet. It indicates this to the transmitter by responding with a BUSYNak response frame.

**3.6.9.2.2** *Intermediate frame exchange*

**3.6.9.2.2.1** *Without retransmission*

```
 |CONTOdd|--------->
          <---------|ACK|
|CONTEven|--------->
          <---------|ACK|
```

Final frame:

```
|CONTOdd/Even|--------->
              <---------|LONGAck|
```

This is the most common form of data exchange between units on the power line. The frame headers alternate between CONTOdd and CONTEven for consecutive frames. Note that the response to the last frame is a LONGAck, signalling to the transmitting unit that the entire packet was received successfully.

**3.6.9.2.2.2** *With retransmission*

```
|CONTOdd|--------->
         <---------|NAK|
|CONTOdd|--------->
         <---------|ACK|
```

The receiving unit uses a NAK control frame to request a retransmission of a data frame received with errors.

**3.6.9.3** ***Broadcast data packet exchange***

```
|BRDHdr|00|--------------->
|BRDHdr|01|--------------->
|BRDHdr|02|--------------->
|BRDHdr|03|--------------->
|BRDHdr|04|--------------->
|BRDHdr|05|--------------->
|BRDHdr|06|--------------->
|BRDHdr|07|--------------->
|CONTOdd|--------------->
|CONTOdd|--------------->
|CONTOdd|--------------->
|CONTOdd|--------------->
|CONTEven|--------------->
|CONTEven|--------------->
|CONTEven|--------------->
|CONTEven|--------------->
     .
     .
     .
```

This describes the transmission of a broadcast data packet where the receiving units cannot confirm the receipt of a transmission or request a retransmission. The BRDHdr frame is repeated eight times to allow all receiving units an ample opportunity to initiate a broadcast packet reception. All subsequent data frames, both intermediate and last, are repeated four times, again to allow all receiving units to receive at least one error-free frame from each repeated data frame.

There shall be a minimum of 1 byte time (59,5 μs) and a maximum of 2 byte times (119 μs) between consecutive transmitted frames.

**3.6.9.4** ***Control exchange — Master polling a slave***

**3.6.9.4.1** *Slave has data to transmit*

```
     |PASSControl|--------->
                 <---------|LONGAck|
|  PASSControlHandshake|--------->
                        <---------| CONNECT|
                            |
                            |
                        <---------| RETURNControl|
```

The master polls the slave by sending a PASSControl. If the slave has any data to send, it responds with a LONGAck and the master, in turn, sends a PASSControlHandshake. This allows the slave to start transmitting its data packet. On completion of its transmissions, the slave returns control to the master with a RETURNControl.

**3.6.9.4.2** *Slave does not have data to transmit*

```
|PASSControl|--------->
            <---------|NAK|
```

The master polls the slave by sending a PASSControl. If the slave does not have any data to send, it responds with a NAK.

**3.6.9.5** ***Timeouts***

The transmitter will wait for at most 1 ms to receive a response from the destination. If none is received within this time period, it will retransmit the frame (data or control).

**3.6.9.6** ***Data rate***

An error-corrected data rate of 19,2 kbits $s^{-1}$ with an error rate of less than $10^{-9}$ is achieved, and is determined by the specifications above.

**3.6.10 Out-of-band filtering requirements for "low data rate" compatibility**

The out-of-band power spectrum shall not exceed 2 mV r.m.s. from 40 kHz to 70 kHz and shall not exceed the linear plot shown in Figure 27 of decibel volts against frequency from 2 mV r.m.s. at 70 kHz to 180 mV r.m.s. at 140 kHz and the linear plot shown in Figure 27 of decibel volts against frequency from 180 mV r.m.s. at 140 kHz to 2 V r.m.s. at 200 kHz as measured into an 18 Ω load in any 10 kHz bandwidth.

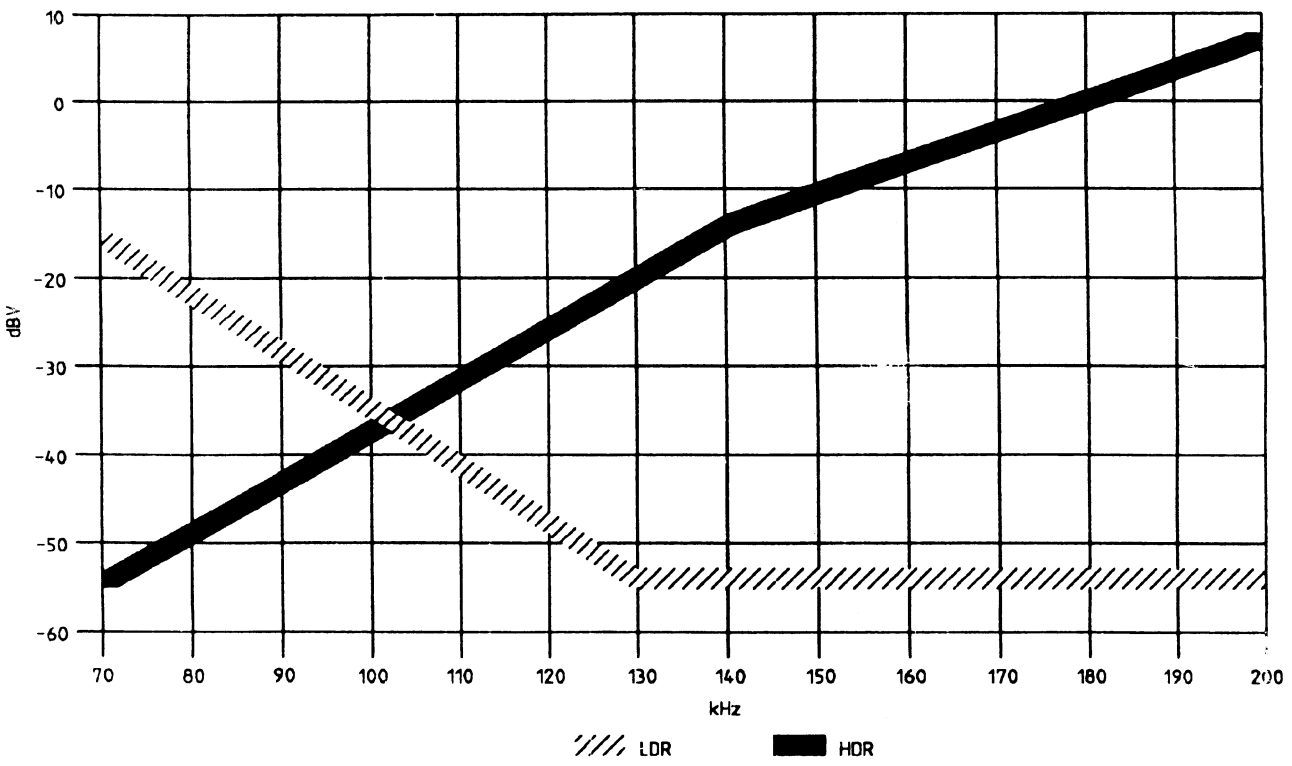



**Figure 27 — Filter specifications — Low data rate compatibility**

**Table 30 — Codes for data link protocol**

| Frame and field(s) | Hexadecimal value of field |
|---|---|
| Preamble (see **3.6.9.1.1**) | |
| SYNC | 01 |
| SOFLong | B6 |
| SOFShort | DA |
| Data frames (see **3.6.9.1.2**) | |
| CONNECT | 50 |
| FIRSTFrameReTx | A5 |
| BRDHdr | FB |
| CONTOdd | C6 |
| CONTEven | 39 |
| Control transfer frames (see **3.6.9.1.3**) | |
| PASSControl | 16 |
| Response frames (see **3.6.9.1.4**) | |
| NOTAck | 4B |
| LONGAck | 6D |
| NAK | 58 |
| CONNAck | EC |
| ACK | B4 |
| BUSYNak | 10 |
| PASSControlHandshake | 05 |
| RETURNControl | 24 |
| Control frames (see **3.6.9.1.5**) | |
| ACTIVITYFrame | 54 |

**Table 31 — (32,8) ECC and EDC codes**

| Data byte | Code | Data byte | Code | Data byte | Code | Data byte | Code |
|---|---|---|---|---|---|---|---|
| 00 | 5A5A5A5A | 40 | 5ABC892E | 80 | 5A31A5CE | C0 | 5AA27676 |
| 01 | 5D5AA589 | 41 | 5DBC76BB | 81 | 5D315ABC | C1 | 5DA289A5 |
| 02 | A55A5D31 | 42 | ASBC3144 | 82 | A5316943 | C2 | A5A29669 |
| 03 | 695A695D | 43 | 69BC96D1 | 83 | 69315DA2 | C3 | 69A23196 |
| 04 | BC5AD196 | 44 | BCBC44A2 | 84 | BC3143D1 | C4 | BCA2A25D |
| 05 | 435A4369 | 45 | 43BCA243 | 85 | 4331D144 | C5 | 43A24431 |
| 06 | 2E5A2EA5 | 46 | 2EBCBBBC | 86 | 2E31BCBB | C6 | 2EA2CE89 |
| 07 | D15ABC76 | 47 | D1BCCECE | 87 | D1312E2E | C7 | D1A2BB5A |
| 08 | 315A7644 | 48 | 31BCA531 | 88 | 31318969 | C8 | 31A25A43 |
| 09 | 895A89D1 | 49 | 89BC5ASD | 89 | 89317696 | C9 | 89A2A5A2 |
| 0A | 965A962E | 4A | 96BC695A | 8A | 96313176 | CA | 96A25DCE |
| 0B | 765A31BB | 4B | 76BC5D89 | 8B | 763196A5 | CB | 76A269BC |
| 0C | A25AA2BC | 4C | A2BC43A5 | 8C | A2314489 | CC | A2A2D1BB |
| 0D | CE5A44CE | 4D | CEBCD176 | 8D | CE31A25A | CD | CEA2432E |
| 0E | 445ACEA2 | 4E | 44BCBC96 | 8E | 4431BB5D | CE | 44A22ED1 |
| 0F | BB5ABB43 | 4F | BBBC2E69 | 8F | BB31CE31 | CF | BBA2BC44 |
| 10 | 5A5D69D1 | 50 | 5A43965D | 90 | 5A895D96 | D0 | 5ACE31A2 |
| 11 | 5D5D5D44 | 51 | 5D433131 | 91 | 5D896969 | D1 | 5DCE9643 |
| 12 | A55DA5BB | 52 | A5437689 | 92 | A5895AA5 | D2 | ASCE89BC |
| 13 | 695D5A2E | 53 | 6943895A | 93 | 6989A576 | D3 | 69CE76CE |
| 14 | BC5DBCCE | 54 | BC43CE76 | 94 | BC892E5A | D4 | BCCEBB2E |
| 15 | 435D2EBC | 55 | 4343BBA5 | 95 | 4389BC89 | D5 | 43CECEBB |
| 16 | 2E5D4343 | 56 | 2E43A269 | 96 | 2E89D131 | D6 | 2ECE4444 |
| 17 | D15DD1A2 | 57 | D1434496 | 97 | D189435D | D7 | D1CEA2D1 |
| 18 | 315D3189 | 58 | 31435DBB | 98 | 318996BC | D8 | 31CE69A5 |
| 19 | 895D965A | 59 | 8943692E | 99 | 898931CE | D9 | 89CE5D76 |
| 1A | 965D895D | 5A | 96435AD1 | 9A | 968976A2 | DA | 96CEA596 |
| 1B | 765D7631 | 5B | 7643A544 | 9B | 76898943 | DB | 76CE5A69 |
| 1C | A25DBB69 | 5C | A2432E43 | 9C | A289CE44 | DC | A2CEBC31 |
| 1D | CE5DCE96 | 5D | CE43BCA2 | 9D | CE89BBD1 | DD | CECE2ESD |
| 1E | 445D4476 | 5E | 4443D1CE | 9E | 4489A22E | DE | 44CE435A |
| 1F | BB5DA2A5 | 5F | BB4343BC | 9F | BB8944BB | DF | BBCED189 |
| 20 | 5AA52E44 | 60 | 5A2EBB31 | A0 | 5A96BC69 | E0 | 5A44CE43 |
| 21 | 5DA5BCD1 | 61 | 5D2ECE5D | A1 | 5D962E96 | E1 | 5D44BBA2 |
| 22 | A5A5D12E | 62 | A52E445A | A2 | A5964376 | E2 | A544A2CE |
| 23 | 69A543BB | 63 | 692EA289 | A3 | 6996D1A5 | E3 | 694444BC |
| 24 | BCA55DBC | 64 | BC2E31A5 | A4 | BC966989 | E4 | BC4496BB |
| 25 | 43A569CE | 65 | 432E9676 | A5 | 43965D5A | E5 | 4344312E |
| 26 | 2EA55AA2 | 66 | 2E2E8996 | A6 | 2E96A55D | E6 | 2E4476D1 |
| 27 | D1A5A543 | 67 | D12E7669 | A7 | D1965A31 | E7 | D1448944 |
| 28 | 31A5CE5A | 68 | 312EBC2E | A8 | 3196BBCE | E8 | 31442E76 |
| 29 | 89A5BB89 | 69 | 892E2EBB | A9 | 8996CEBC | E9 | 8944BCA5 |
| 2A | 96A5A231 | 6A | 962E4344 | AA | 96964443 | EA | 9644D169 |
| 2B | 76A5445D | 6B | 762ED1D1 | AB | 7696A2A2 | EB | 76444396 |
| 2C | A2A59696 | 6C | A22E69A2 | AC | A29631D1 | EC | A2445D5D |
| 2D | CEA53169 | 6D | CE2E5D43 | AD | CE969644 | ED | CE446931 |
| 2E | 44A576A5 | 6E | 442EA5BC | AE | 449689BB | EE | 44445A89 |
| 2F | BBA58976 | 6F | BB2E5ACE | AF | BB96762E | EF | BB44A55A |
| 30 | 5A694389 | 70 | 5AD1A2BB | B0 | 5A76D1BC | F0 | 5ABB44A5 |
| 31 | 5D69D15A | 71 | 5DD1442E | B1 | 5D7643CE | F1 | 5DBBA276 |
| 32 | A569BC5D | 72 | A5D1CED1 | B2 | A5762EA2 | F2 | A5BBBB96 |
| 33 | 69692E31 | 73 | 69D1BB44 | B3 | 6976BC43 | F3 | 69BBCE69 |
| 34 | BC69A569 | 74 | BCD17643 | B4 | BC765A44 | F4 | BCBB8931 |
| 35 | 43695A96 | 75 | 43D189A2 | B5 | 4376A5D1 | F5 | 43BB765D |
| 36 | 2E696976 | 76 | 2ED196CE | B6 | 2E765D2E | F6 | 2EBB315A |
| 37 | D1695DA5 | 77 | D1D131BC | B7 | D17669BB | F7 | D1BB9689 |
| 38 | 316944D1 | 78 | 31D1D15D | B8 | 3176A296 | F8 | 31BB43A2 |
| 39 | 8969A244 | 79 | 89D14331 | B9 | 89764469 | F9 | 89BBD143 |
| 3A | 9669BBBB | 7A | 96D12E89 | BA | 9676CEA5 | FA | 96BBBCBC |
| 3B | 7669CE2E | 7B | 76D1BC5A | BB | 7676BB76 | FB | 76BB2ECE |
| 3C | A26989CE | 7C | A2D15A76 | BC | A276765A | FC | A2BBA52E |
| 3D | CE6976BC | 7D | CED1ASA5 | BD | CE768989 | FD | CEBB5ABB |
| 3E | 44693143 | 7E | 44D15D69 | BE | 44769631 | FE | 44BB6944 |
| 3F | BB6996A2 | 7F | BBD16996 | BF | BB76315D | FF | BBBB5DD1 |

## Annex A (informative)
## Bibliography

[1] ISO 1745:1975, *Information processing — Basic mode control procedures for data communication systems.*

[2] ISO 2111:1985, *Data communication — Basic mode control procedures — Code independent information transfer.*

[3] ISO 2628:1973, *Basic mode control procedures — Complements.*

[4] ISO 2629:1973, *Basic mode control procedures — Conversational information message transfer.*

[5] ISO 3309:1991, *Information technology — Telecommunications and information exchange between systems — High-level data link control (HDLC) procedures — Frame structure.*

[6] ISO 6346:1984, *Freight containers — Coding, identification and marking.*

[7] ISO 7498:1984, *Information processing systems — Open Systems Interconnection — Basic Reference Model.*