GEC ALSTHOM

T&D

Courier User Guide

R6512 E

GEC ALSTHOM T&D PROTECTION & CONTROL LIMITED, St Leonards Works, Stafford ST17 4LX, England
Tel: 01785 223251 Telex: 36240 Fax: 01785 212232

Preface

This document details the communication language used on GEC ALSTHOM T&D Protection & Control relays called "Courier" in sufficient detail to enable third parties to interface these relays to other SCADA type systems and to develop other slave devices (relays) which utilise this language. It describes how messages are constructed and how transactions take place using the Courier protocol.

This guide should be used with the appropriate user guide for the particular communication network system used to transfer the Courier language between devices and the Courier Protocol document.

The information and illustrations found in this book are not binding. GEC ALSTHOM T&D Protection & Control Limited. reserve the right to modify products in line with a policy of continuous product improvement. Information in this document is subject to change without notice.

Table of Contents **Page**

Figures **Page**

Tables **Page**

1. Courier Overview

1.1 INTRODUCTION

The Courier Communication Language has been developed to provide remote control, monitoring, data extraction and setting changes on protective relays within the substation environment. The term 'protective relay' however has been replaced by the term 'slave device' in the remainder of this document to emphasise Courier's generic nature; it may in future be used on other devices such as remote I/O devices or Remote Terminal Units.

A new language was considered necessary to provide independence between a master control unit and a slave device; most other communication systems rely on both communicating devices having inherent knowledge of locations for all items of data and the format that the data will take when it is transmitted. Courier has been specifically designed to eliminate this interdependence and enables a generic master control unit to be developed. This is able to communicate with any and all slave devices which support the Courier language, without having to reprogram it for each new slave device that is added to the system at a later date.

In designing a new communication system for relays in the electrical distribution environment, several design goals were borne in mind:

i. Ease of use was considered a very important aspect: a system should be "up and running" with minimal effort. Communication should be established between a slave device and a master control unit as soon as they are connected, enabling the user to view monitoring information and change settings straightaway without any programming being required. This should still be true when a new slave device is added to the system, or when an existing relay is replaced with a later version. The need for reference manuals, programming languages and master control unit software upgrades should be avoided wherever possible.

ii. The master control unit should be able to cope with different slave devices performing different functions and therefore having different data monitoring and remote control capabilities. It should be able to determine data formats and types from the slave device without prior knowledge or expectation.

iii. The communication system should be secure to prevent maloperation of remote commands and to ensure data integrity. The communication should not hinder the normal operation of the slave device as communication is considered a secondary function. For

this reason it is not used for protection purposes, but simply as a remote monitoring and control channel. Protection signalling being provided by separate means when required.

iv. The communication system should be suitable for the harsh environment in which it will be found. Ideally it should have isolation and noise immunity to protection standards.

1.2 COURIER COMPONENTS

The Courier language satisfies the above goals to provide a language capable of communicating with a slave device in a device-independent manner. This is achieved using the following components:

i. A protocol and set of transactions defining how a master control unit and a slave device communicate.

ii. A message structure consisting of typed data packets which inherently describe the format of data that is transmitted.

iii. A common database structure in each slave device to hold all information that can be accessed by Courier.

iv. A generic command set consisting of comparatively low-level commands which operate on a slave device's database structure, rather than on physical quantities.

The Courier communication language can be considered part of the application layer (layer 7) of the ISO-OSI 7 layer communication system model using model 2, the Enhanced Performance Architecture (EPA) model, although it effectively sits on top of the Courier Protocol. It must therefore be used with an appropriate communication system such as K-Bus or IEC870. GEC ALSTHOM T&D Protection & Control's implementation of these are described elsewhere. Reference should be made to the Courier Protocol document which describes the remainder of the application layer.

1.2.1 COURIER TRANSACTIONS

Courier is designed to operate using a 'master-slave' or 'polled' protocol in which only one device (a master control unit) can initiate a transaction (see chapter 3) by sending a request message. The other devices (slave devices) either return a response message to the master control unit or take the action requested.

The master control unit can send a request message to an individual slave device or broadcast a global request message to all slave devices by using a unique global address. Slave devices will return a response to all correctly received request messages that are addressed to them individually. Responses are not returned to global request messages.

A slave device will not send any unsolicited data. A master control unit must regularly interrogate the slave devices to extract any information it requires and to determine if the slave devices have any data they wish to send that has not been previously requested.

A Courier transaction therefore consists of a request message from a master control unit and its associated response message from a slave device. There are several types of Courier transaction each determined by the type of response that is returned. A master control unit can identify a global transaction from a request message alone. The other types of transaction can

only be determined by the response from the slave device. In the following descriptions, mention is made of 'packets' of data which are described in the following section.

Simple Transaction
A simple transaction occurs when a single request message containing a single request command results in a response containing a single packet of information which fits into the response frame.

Grouped Transaction
A grouped transaction occurs when a single request message containing a single request command results in a response containing multiple packets of information which fit into the response frame. The multiple packets associated with the request are grouped together into a larger packet so that they can be identified as a single response entity.

Multiple Transaction
A multiple transaction occurs if a request message contains multiple request commands which would therefore result in multiple responses. A multiple request message can only be sent provided:

- Each individual request would result in a simple or grouped transaction.
- All responses will fit as a whole into a single response message.

The reason for grouping multiple response packets so that they can be associated to a single request is now clear as it enables a master control unit to separate the multiple responses and match them with the corresponding request in a multiple transaction.

Blocked Transaction
A blocked transaction occurs when a single request message containing a single request command results in a response containing multiple packets of information which cannot fit into a single message. The packets are grouped into blocks such that each block fits into a response frame. The nature of the response is indicated in the first response message from the slave device. The master control unit will then request each subsequent block in the transaction until all blocks have been transferred. Each block will contain multiple packets, most probably grouped into larger packets. Blocked transactions cannot be nested, although they can be interspersed with any other type of transaction. Any request that results in a blocked transaction cannot be included in a block transaction.

1.2.2 COURIER PACKETS

Most communication languages are developed for specific master control units and slave devices, often manufactured by the same company. This allows each device to have inherent knowledge of the internal workings of the other and hence only the minimum amount of information is ever transmitted over the communication system to gain advantages in communication efficiency and data transfer rates. The master control unit is pre-programmed to know the format of data returned from a slave device in response to a particular request. If a third party manufactures a different slave device which is to work on the same communication system, a reference manual is often required to indicate the differences between itself and the other slave devices, to describe the data formats used, the locations of particular data and to enable a master control unit to be reprogrammed to access any additional information it may contain.

Although Courier has also been developed with specific master control units and slave devices in mind, care has been taken to remove any dependencies between them and hence make it suitable for other devices in the future.

The first contribution to breaking this interdependence is made by encapsulating all data into packets. A packet consists of a fundamental item of data and details of its type and length. Thus, a master control unit does not need to know the format of any data it receives in a response from a slave device since this information will be included in the response itself. By using fundamental data types, such as unsigned integers, signed integers, text strings and so on, the range that must be catered for is limited. The length of the data is separated from the data type to similarly reduce the number of data types required and theoretically allows any data type to have a length from 1 to 255 bytes.

The use of these data packets in Courier messages provides a number of advantages over the sole disadvantage of costing one or two extra bytes per information unit in a message. These are:

1. Information format is not dependent upon context; it is not implicitly assumed but explicitly stated.
2. Dialog security is improved by incorporating the context of the information which may be checked against what is expected.
3. Flexibility is increased by relaxing constraints on what may be transmitted, since the slave device is not coerced by the master control unit.
4. The data type codes are in a common format and are designed to be extensible.
5. Since the data length is specified in a consistent manner, unrecognised data type codes are easily ignored without corrupting an entire message.
6. As all Courier messages consist of these typed data packets, a passive network monitor can be easily implemented.

1.2.3 COURIER DATABASE

All data in a slave device is referenced by a 16-bit reference number. Courier views this reference number as the position of the data in a tabular database with X and Y co-ordinates where X specifies the column number in the range 0-255 and Y specifies the row number in the range 0-255 in the table. The intersection of a row and column number is referred to as a 'menu cell'.

> *The actual storage of data in a slave device is device dependent, the idea of a tabular database is conceptual only.*

Each menu cell in this database can have several attributes as follows:
- a string of text to describe the data
- a data value
- data value type and length information
- an indexed array of strings to describe individual setting values
- a set of limits indicating how to set the value

The number of attributes a menu cell has is dependent on the type of the data it contains and the operations which can be performed on it.

For ease of location of information, the table is arranged into a menu-type structure. The first row of the table contains menu cells which only have a text attribute to describe the types of menu cells stored in the column beneath it. Menu cells are therefore grouped into columns of related information.

This concept of structuring the menu can be visualised by using the analogy of a filing cabinet, although the actual structure of the data contained within a device may be quite different.

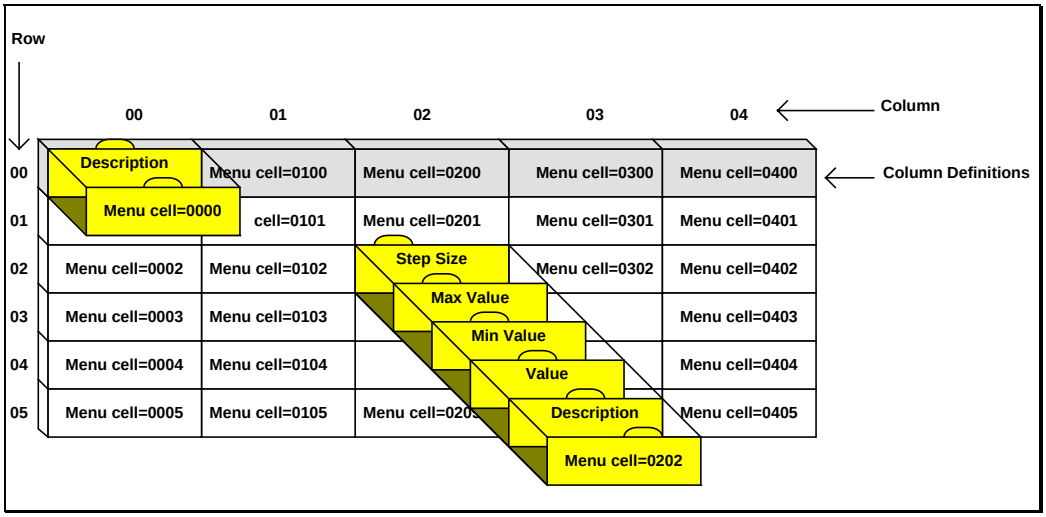


Figure 1-1 Menu Database

By storing slave device data in this way, rather than simply using address and length parameters as in other communication languages, greater flexibility is introduced:

- The master control unit only specifies the menu cell reference number and does not need to specify the type or length of the data as this is taken care of by the data packet returned in response to any request.

- Only the number of data items is restricted (to 65536 menu cells) and not their individual sizes. In schemes using address and length parameters, the range of the address is limited, thus restricting the total amount of data.

- The menu cell can contain additional information relating to the data item such as descriptive text rather than just a numeric value, which can be accessed using different commands.

- A mechanism for browsing the database can be implemented, thus removing the necessity for a reference manual for each slave device.

1.2.4 COURIER COMMAND SET

The Courier command set is based on commands that access or operate upon the various menu cell attributes in a slave device's database and are therefore independent of the type of data each cell contains. The command set is thus reduced to a set of database query and operate commands which can then be used on any type of slave device that uses Courier as a communication language. For example, there are commands to: *Get Text* of a menu cell, *Get Value* of a menu cell and *Get Limits* of a menu cell. Additional commands are provided to perform extraction of events, block transfers, setting changes and so on, but these are of a similar generic nature.

The command language has been designed to accomplish fairly low-level tasks which are common to all slave devices. Additional features specific to a particular slave device can be implemented using combinations of these low-level commands.

The remainder of this document describes the individual components of the Courier communication language in further detail. Chapter 2 describes the structure of Courier messages and the packets that information is encapsulated into. Chapter 3 discusses the various transactions that Courier uses. Chapter 4 describes the structure of the database stored in each slave device. Chapter 5 is a reference guide to the different types of Courier data packets. Chapter 6 is a command reference detailing all the available commands, their purpose and use and includes examples on each one. The remaining chapters detail specific uses and applications of the Courier language.

2. Courier Message Structure

A Courier request message consists of 3 fields: an address field, a length field and a user data field. Reply messages contain an additional reply control field (see Courier Protocol Document R6511) which is filtered out by the Courier Protocol layers to perform automatic data retrieval and annunciation.

Courier Request Message Format; sent to a slave

Address Field	Length Field	User Data Field

Courier Reply Message Format; sent by a slave

Address Field	Length Field	Reply Control Field	User Data Field

↑ *This field is not seen by the application*

> *The transmission of Courier messages from one device to another is determined by a lower level communication network system such as K-Bus or IEC870. During transmission the address field may be separated from the rest of the message and the length field adjusted to account for additional network information according to the particular protocol of the underlying network.*

The User Data field consists of Courier Data Packets, whereas the Address and Length fields do not, these latter fields having a fixed format. The format of the individual data packet types are described fully in a later chapter.

2.1 ADDRESS FIELD

In request messages, the address field identifies which slave device the message is being sent to. In reply messages, the address field identifies which slave device the message is being sent from. The address field consists of between 1 and 6 address bytes and a terminating zero byte to delimit the address field from other fields in the message.

To describe the format and use of the address field, two addressing modes are defined: 'single level addressing' and 'multi level addressing'. The mode that is used depends on the topology of the network of Courier devices. In the following sections, the term 'level' is used as a conceptual model only to aid the descriptions.

2.1.1 SINGLE LEVEL ADDRESSING.

The simplest Courier system consists of a master control unit directly connected to a maximum number of 254 lower level slave devices. Single level addressing is used for this system in which the address field contains a single address byte followed by a terminating zero byte. The address byte may have any value from 0 to 255. Address 255 is used for global messages broadcast to all slave devices. Address 0 is used for automatic address allocation and can only be used in this single level addressing mode. In this scheme, the slave devices can be described as level 0 devices and the master control unit a level 1 device.

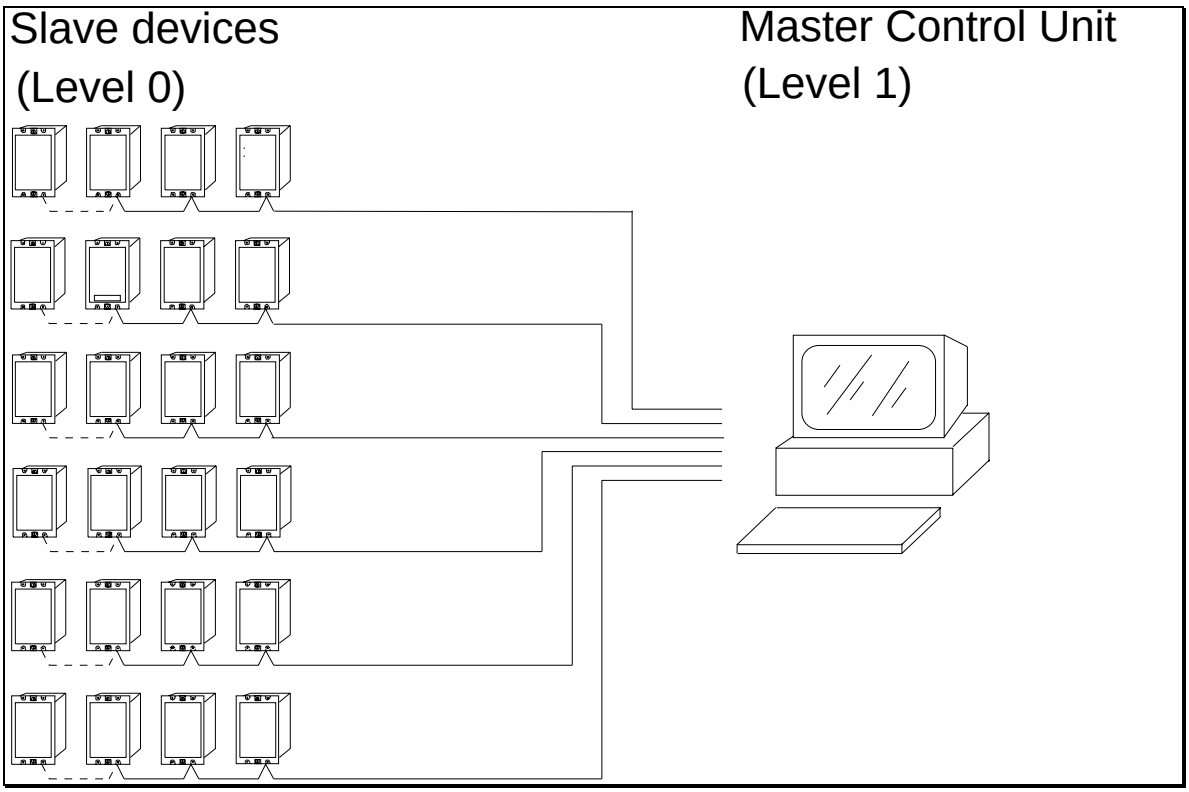


Figure 2-1 Single level addressing topology

2.1.2 MULTI-LEVEL ADDRESSING.

Courier is designed to use a hierarchical addressing scheme in which a master control unit may act as a slave device on a higher level network. In this situation the master control unit also requires an address so that a Level 2 master control unit can communicate with it. This scheme can be extended with each master control unit acting as a slave device to the next level in the hierarchy. The highest level in the hierarchy will be a level 5 master control unit. The same single level addressing will be used for direct communication between adjacent levels in the hierarchy, but an extended scheme is required to enable a higher level master control unit to communicate with a slave device several layers lower.

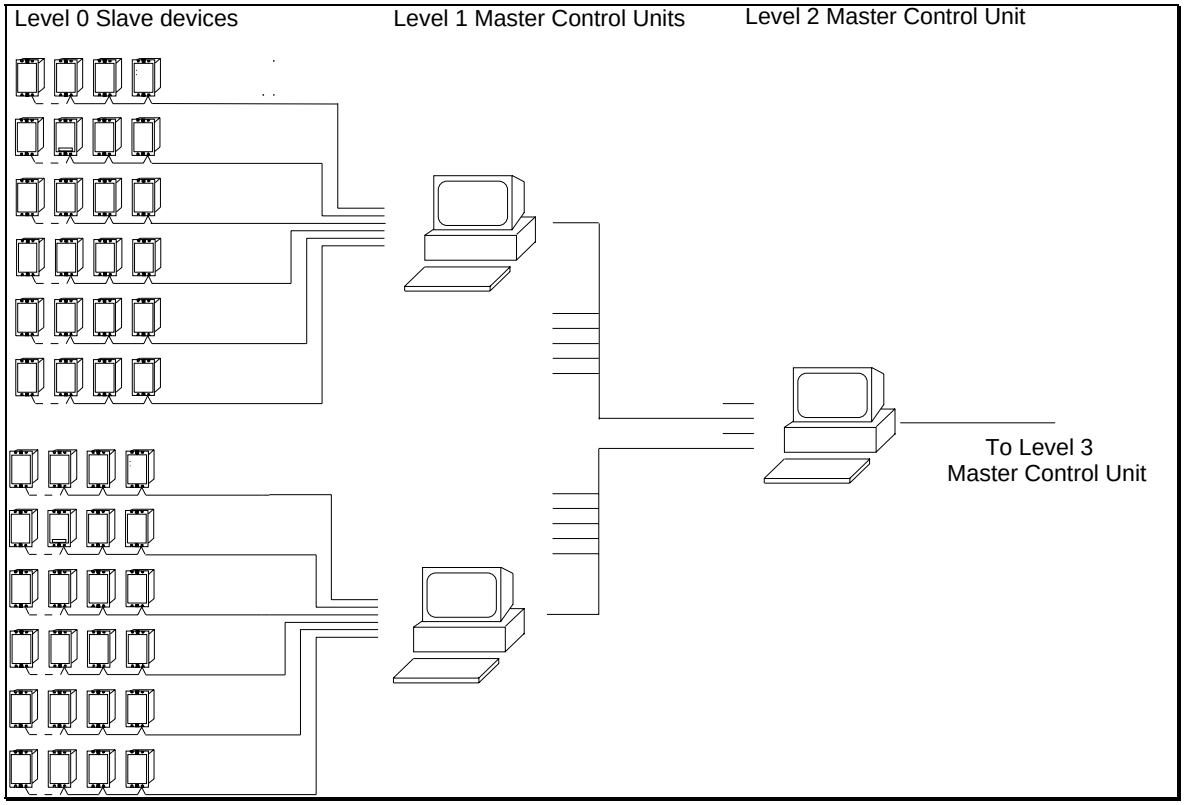


Figure 2-2 Multi level addressing topology.

Multi level addressing is used to enable non-adjacent master control units and slave devices to communicate, for example, when a level 2 master control unit wishes to communicate with a level 0 slave device.

Multi Level Request frames

A level 'H' master control unit can send a request frame to a level 'L' slave device by specifying an address field which contains the addresses of all the intermediate master control units of levels H-1, H-2, H-3,... and the address of the level L slave device, as shown in the following figure:

$Addr_{(H-1)}$	$Addr_{(H-2)}$	$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

Multi level address field format.

When the message is received by each intermediate master control unit, the second byte of the address field is examined. If it is non-zero, the message is destined for a lower level slave device. The master control unit will remove the first byte of the address field (its own) and pass the message down the hierarchy. Eventually, the master control unit at level L+1 will send the message to the level L slave device, effectively using the single level addressing format.

$Addr_{(H-1)}$	$Addr_{(H-2)}$	$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

$Addr_{(H-2)}$	$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

..... ↓ Address field descending through the hierarchy

$Addr_{(L)}$	0

Multi Level Response frames

The slave device which is the eventual destination of a multi level request frame (Level L) will always reply using a single level addressing mode to the next level in the hierarchy since it cannot distinguish between the two types of request. However, the intermediate master control units between level L and H will reply using the multi level addressing mode. They must remember that the next valid response from the lower level should be sent to the next higher level. At each intermediate master control unit, the address of the master control is added to the beginning of the address field before being passed up. In this way the address field fully specifies the path taken through the network from the slave device back to original requesting master control unit.

$Addr_{(L)}$	0

..... ↓ Address field ascending through the hierarchy

$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

$Addr_{(H-2)}$	$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

$Addr_{(H-1)}$	$Addr_{(H-2)}$	$Addr_{(H-3)}$	....	$Addr_{(L)}$	0

The following table indicates the number of possible addresses for the first three levels of a Courier based system:

Per K-Bus Spur	32
Level 1	254
Level 2	64,516
Level 3	16,387,064

K-Bus addresses per level.

Any level of master control unit may communicate with a lower level device by forming the address field appropriately.

The network topology need not be symmetrical, allowing level 0 slave devices to be placed at any level in the hierarchy. A master control unit may therefore act as a level 1 master control

unit to a level 0 slave device and a level 2 master control unit to a level 1 master control unit at the same time.

The level number of a master control unit is therefore purely conceptual and is determined by the number of intermediate master control units a message must pass though before reaching its destination. The level number of a master control unit may therefore change for each message sent or received.

2.2 LENGTH FIELD

The length field is a single byte indicating the total length in bytes of the user data field. In reply messages, this also includes the length of the reply control field but this is removed by the lower protocol layers and the length is adjusted accordingly. The 8 bit length field limits the total length of the data field to a maximum of 255 bytes, although this is further restricted by the communication network used to transport Courier messages to 230 bytes (see section 2.5).

2.3 COURIER PACKET FORMAT

Courier messages, particularly the user data fields, are composed of a sequence of DATA PACKETS (also referred to as 'Courier data packets' and 'packets'). These exist in several formats to achieve a trade-off between packet length and flexibility. The data packets ensure that data can be identified, interpreted and analysed easily. It is this feature that makes Courier a generic language enabling a master control unit to communicate with many different types of slave device, including those it may not have seen before.

The packet nature of Courier messages provides a number of advantages over the sole disadvantage of costing one or two extra bytes per information unit in a message. These are:

1. Information format is not dependent upon context; it is not implicitly assumed but explicitly stated.

2. Dialog security is improved by incorporating the context of the information which may be checked against what is expected.

3. Flexibility is increased by relaxing constraints on what may be transmitted, since the slave device is not coerced by the master control unit.

4. The data type codes are in a common format and are designed to be extensible.

5. Since the data length is specified in a consistent manner, unrecognised data type codes are easily ignored without corrupting an entire message.

6. As all Courier messages consist of these typed data packets, a passive network monitor can be easily implemented.

Each data packet consists of 2 fields: a DATA TYPE & LENGTH (DTL) field and a DATA field. The DTL field contains information about the type and length of the data field so that it may be interpreted correctly without any prior knowledge.

The DTL field is usually encoded into 1 byte for compactness as shown below, where the type is coded into the 6 highest bits and the length is coded into the 2 lowest bits:

DTL field	DATA field
TTTTTT LL	DD DD DD

TTTTTT = Data field type code bits
LL = Data field length code bits
DD = Data field data bytes (1-3 bytes)

The 6 type bits allow up to 64 possible data types. Since fundamental data types are used such as unsigned integer, signed integer, binary flags etc. the number required is quite small, especially as the length of these data types is stored separately. However, provision has been made to extend this field in the future by using a type code of TTTTTT = 000000, as follows:

DTL Field	Ext. TYPE	Data Field
000000 LL	TTTTTTTT	DD DD DD

In this encoding, the DTL field occupies 2 bytes with the extended type information being found in the second byte. This extends the number of data types to 318.

The 2 length bits in both of these encodings allow 4 possible data lengths to be specified of 0,1,2 and 3 bytes in length. A data length of 0 bytes would be meaningless, so this is used to extend the length information in the same manner as the extended type information.

DTL Field	Ext. LEN	Data Field
TTTTTT 00	LLLLLLLL	DD .. DD

This allows data lengths of up to 255 bytes to be specified in theory and is most useful for data types such as ASCII Text. In practice, the overall Courier message length is also restricted to 255 bytes and may be even further restricted depending on the communication network used to transfer the Courier message. Thus, data packets of 255 bytes in length would not be possible. (See section 2.5 on maximum user data lengths.)

Where the DTL field is all zero, both an extended type and an extended length field will be present, the extended type field coming before the extended length field as follows:

DTL Field	Ext. TYPE	Ext. LEN	Data Field
000000 00	TTTTTTTT	LLLLLLLL	DD .. DD

2.4 USER DATA FIELD

The user data field consists of one or more Courier packets. Request messages from a master control unit will consist of a command packet optionally followed by other data packets containing additional explicit parameters relating to the command. Response messages will contain data packets returning the slave device's reply to the command. Courier maintains a one-to-one correspondence between each request command and its associated reply which enables multiple commands to be sent to a slave device and multiple response packets to be received. The mechanism for doing this is described fully in chapter 3 on Courier Transactions.

2.5 MAXIMUM USER DATA LENGTHS

The maximum length of the user data field of a Courier message is dependent on the transmission method used, whether it be K-Bus, IEC870 or some other method. Both K-Bus and IEC870 include a single length byte (L_k & L_i respectively) in their message frame which limits the information content of their messages to 255 bytes. This had been adopted as the

absolute maximum total length of a Courier message. Keeping messages comparatively short like this also improves the probability of messages being transferred intact.

Since the information content of a message frame includes fields in the Courier message such as the reply control field, and headers and footers introduced by the transmission method used, the maximum size of the user data field (L_{ud}) is further restricted. In order to calculate this maximum length a worst case situation needs to be analysed. This can only be calculated for K-Bus and IEC870 transmissions since no other transmission format has been defined for Courier at the time of writing.

K-Bus calculation
Reply control field is made up of:
- A time tag of 4 byte millisecond timer count + 2 DTL bytes = 6 bytes.
- A real-time IEC870 time tag of 7 bytes + 2 DTL bytes = 9 bytes.
- A Courier status packet = 2 bytes.

Total maximum length = 17 bytes.

Other fields counted by L_k:
- IEC870 Control byte + DTL = 2 bytes.

$L_k(max) = 2+17 + L_{udk}(max)$
$L_{udk}(max) = 255 - 19 =$ 236 bytes for K-Bus messages.

IEC870 calculation
Reply control field is made up of:
- A time tag of 4 byte millisecond timer count + 2 DTL bytes = 6 bytes.
- A real-time IEC870 time tag of 7 bytes + 2 DTL bytes = 9 bytes.
- A Courier status packet = 2 bytes.

Total maximum length = 17 bytes.

Other fields counted by L_i:
- IEC870 Control byte = 1 byte.
- Address field = 6 bytes + zero terminator = 7 bytes.

$L_i(max) = 1 + 7 + 17 + L_{udi}(max)$
$L_{udi}(max) = 255 - 25 =$ 230 bytes for IEC870 messages.

For a message to be transferred over either K-Bus or IEC870, the shorter of these two maximum user data lengths should be used. This value includes the overhead of any DTL's in the user data field, so the maximum length data packet that can be transferred will be 228 bytes.

3. Courier Transactions

A set of protocol rules and a set of transaction rules are defined for Courier. They can be considered as two separate layers with the protocol rules defining the low level operation of Courier and how single request and reply messages may be sent over the communication network; the transaction rules defining the high level operation of Courier and how single request and response messages are combined into transactions. The Courier Protocol is described in a separate document.

A transaction consists of a communication request from a master control unit and an associated communication reply from the addressed slave device. These follow a one-to-one correspondence. There are five different types of transaction: Simple, global, multiple, grouped and blocked transactions.

A **simple transaction** consists of one request message containing a single request packet being sent to a slave device.

A **global transaction** is a simple transaction that can be received by more than one slave device simultaneously.

A **multiple transaction** consists of one request message which contains two or more request packets being sent to a slave device. Care is required when using these transactions since all request packets must fit within a single reply message.

A **grouped transaction** arises when a single request packet results in multiple response packets.

A **blocked transaction** occurs when a single request results in multiple grouped packets, each of which must be requested individually by the master control unit. Other simple, global, multiple or grouped transactions may occur to the same device whilst a blocked transaction is in progress, provided none of these result in a second blocked transaction being initiated.

The different transaction types are described in the following sections. In each section, diagrammatic boxes are used to exemplify the message structure. Each data packet is surrounded by double lines. Each of these packets is split vertically into its separate fields using thick single lines to separate the DTL field from the user data field and thin single lines to separate different sub-fields in the user data field. Each field is then split horizontally into 3 rows separated by a thin single line and a thick single line. The top row describes the nature

of the field, the middle row explains its use in this context and the bottom row contains hexadecimal values of the bytes used in the example.

3.1 SIMPLE TRANSACTIONS

A simple transaction consists of one request message containing a single request packet being sent to a slave device. The slave device will respond with a single message containing a single reply packet appropriate to the request. An example of this is shown below where the value of cell 010C is requested.

Request message

DTL	Command	Implicit Arg.
Command (3 bytes)	Get value	menu cell 010C
07	14	0C 01

Response message

DTL	Value requested
Courier (4 bytes)	10.00 secs
2C 04	E8 03 7C 08

The maximum size of any item of data is restricted by the maximum size of the user data field.

3.2 GLOBAL TRANSACTIONS

Global transactions are used to send the same command to several slave devices at once, thus reducing the number of transactions required and allowing the command to be actioned in all slave devices simultaneously.

There is no associated reply in a global transaction, since this would cause multiple replies from several slave devices, risking contention on the communication network. Global commands are therefore repeated in separate but consecutive messages to improve the security of the transaction.

A slave device will only action a global command after it has correctly received twice in succession without any other intervening valid message. To gurantee this, the master sends the global command three times to cater for the situation where one of the messages is lost in the communication. To prevent two consecutive global transactions being interpreted as three commands, an innocuous Poll Status command is sent as a global command as the fourth message in the sequence, as shown below:

1. Send command message to address 255.
2. Repeat command message to address 255.
3. Repeat command message to address 255 a second time.[1]
4. Send Poll Status command to address 255.

[1] This step is omitted for the Set Real Time command. See 3.2.1 Global Command Restrictions.

3.2.1 GLOBAL COMMAND RESTRICTIONS

Global Transactions should not be sent whilst there are outstanding busy responses on any of the slave devices, since the slave devices will discard all incoming messages (except the low level Poll Status, Poll Buffer and Reset Remote Link commands) whilst in the busy state.

Global transactions are permissable on most Courier commands, although the slave devices will not respond. This would make the use of certain commands impractical in a global transaction, such as: Get Value, or Enter Setting Mode.

When control commands (such as: Load Shedding by Group), which are normally intended to be sent globally, are implemented as a simple transaction, the command is less secure than both a global transaction and a normal setting change involving reflex verification.

The Set Real Time command is only sent twice during a global transaction to improve time-synchronisation between the slave devices. Otherwise it could not be determined if the first global Set Real Time command received by the slave device was the first or second command transmitted by the master, resulting in a time synchonisation error equal to the time delay between these two command messages.

3.3 MULTIPLE TRANSACTIONS

A multiple transaction consists of one request message which contains two or more request command packets being sent to a slave device. The slave device will respond by returning one reply message containing the response packets to each request command packet, in the same order as they were requested. The following example requests the value from cell 010C as above, but also the text from cell 0204.

Request message

DTL	Command	Implicit Arg.	DTL	Command	Implicit Arg.
Command (3 bytes)	Get value	menu cell 010C	Command (3 bytes)	Get text	menu cell 0204
07	14	0C 01	07	12	04 02

Response message

DTL	Value requested	DTL	Value requested
Command (4 bytes)	10.00 secs	Text (10 bytes)	"MES1 Io %k"
2C 04	E8 03 7C 08	18 0A	4D 45 53 31 20 49 6F 20 25 6B

A slave device can service many requests in a single message and will output all responses in one reply message. However, a slave device may not check that the responses all fit into one message and hence may crash due to a buffer overrun condition, although this should be guarded against if at all possible. It is therefore the responsibility of the master control unit to ensure that the slave device can fit all the response packets into one message frame and not to issue multiple requests that will cause such a situation to occur. It is therefore important that the data packet length is consistent for each slave response to a command for a given menu location.

3.4 GROUPED TRANSACTIONS

Most requests will only result in one piece of information being returned. However, there are occasions when one request may require several pieces of information to be returned, such as a request for the setting limits of a cell. This will result in a minimum, maximum, step and actual value being returned in 4 separate data packets.

Each packet in the response must correlate directly with a request packet, so that if one response in a multiple transaction is not interpretable, it can be skipped and the next response looked at. To do this without prior knowledge of the number of packets in each response, it is necessary to group all the data packets together, so they become a single entity. This can then be skipped regardless of how many data packets it contains.

To do this, there is a 'Super data packet' type called a Group packet which precedes such a collection of related data packets to group them together. The group packet contains a *group type byte* and a *group length byte*. The group type byte specifies the format of the following data packets in the group and the group length byte specifies the size, in bytes; such that the group may be skipped if it is found not to be interpretable.

The following example illustrates the packets that will be sent and received when a request is made to get the setting limits for cell 010C.

Request message

DTL	Command	Implicit Arg.
Command (3 bytes)	Enter Setting Mode	menu cell 010C
07	05	0C 01

Response message

DTL	Group ID	Group Len	DTL	Value	DTL	Min
Group Packet	Setting Limits	24 bytes	Courier (4 bytes)	10.00 secs	Courier (4bytes)	0.100 secs
0A	21	18	2C 04	E8 03 7C 08	2C 04	64 00 76 08

DTL	Max	DTL	Step
Courier (4 bytes)	100.0 secs	Courier (4 bytes)	0.100 secs
2C 04	E8 03 7D 08	2C 04	64 00 76 08

3.5 BLOCKED TRANSACTIONS

Blocked transactions are required where several grouped data packets are to be transferred, either as the response to a single request, or as a block of data to be downloaded with the Set Value command. They serve to combine the grouped data packets into a single entity, allowing the data to span more than one message frame.

The protocol only permits one blocked transaction to be in progress on each slave device in each direction. Should an outstanding blocked transaction be interrupted by a second blocked transaction in the same direction, the first blocked transaction will be abandoned. Attempting

to continue the first blocked transaction after the second has completed will result in undefined behaviour.

The messages involved in a blocked transactions may be interspersed with other simple or multiple transactions, such as polling for data values or the extraction of event records.

Commands that initiate blocked transactions (e.g. Get Column Headings, Get Column Text etc.) can only be sent as a simple transaction and cannot be included in a multiple transaction. The Get Value command can result in a simple transaction or a blocked transaction depending on the menu cell it is used on. It should therefore not be included in a multiple transaction unless it has already been identified that the response from the particular menu cell results in a simple transaction.

The following commands may result in blocked transactions:

14h Get Value
16h Get Column Headings
17h Get Column Text
18h Get Column Values
19h Get Strings
1Ch Set Value
1Dh Get Column Setting Limits

A blocked transaction can be divided into 3 parts:
- a block header,
- multiple block transfers,
- a block footer.

3.5.1 BLOCK HEADERS.

A blocked transaction is initiated by the slave device returning a block header reply in response to a simple transaction request, or by sending a block header as the argument to the Set Value command. A block header is a type of data packet which specifies the number of blocks that are about to be sent.

Request message

DTL	Command
Command (1 byte)	Send Column Headings
05	16

Response message

DTL	Number of blocks
Block Header (1-2 bytes)	00
0D	00

The *number of blocks* field may be set to zero. In this case the master control unit should accept all further blocks until a block footer occurs.

3.5.2 BLOCK TRANSFERS

When a block header has been received, the master control unit will request the first block of data to be transmitted: Block 0. The Slave device will respond with the block of data preceded by a block identifier indicating the number of this block. The block identifier packet will be followed by one or more group packets, each grouping several data packets into a single entity.

Request message

DTL	Command	Implicit Arg.
Command (3 bytes)	Send block	Block No. XX
06	21	XX

Response message

DTL	Block No.	DTL	Group ID	Group Len	DTL	Data	DTL	Data
Block ID	XX	Group Packet	Col Headings	GL1	Menu	0000	Text	"SYSTEM DATA"
15	XX	0A	11	10	46	00 00	18 0B	53 59 53 54 45 4D 20 44 41 54 41

DTL	Group ID	Group Len	DTL	Data	DTL	Data
Group Packet	Col Headings	GL2	Menu	0100	Text	"FAULT RECORDS"
0A	11	12	46	00 01	18 0D	46 41 55 4C 54 20 52 45 43 4F 52 44 53

A block of data can be requested any number of times by using the same block number. When it has been received correctly, the master control unit will request the next block, by incrementing the block number field in the next request. This is a single byte, so it rolls over to block 0 after block 255.
The slave device can only send the last block of data again, or the next block of data. Any request for an out of sequence block of data will cause the slave device to repeat the last block of data it sent. This allows the master control unit to resynchronise its block requests.

3.5.3 BLOCK FOOTERS

When all block transfers have taken place, the slave device will send a block footer packet in response to the next block request.

Request message

DTL	Command	Implicit Arg.
Command (3 bytes)	Send block	Block No. XX
06	21	XX

Response message

DTL	Total number of blocks sent
Block Footer (2 bytes)	XXYY
12	YY XX

The block footer returns a count of the number of blocks sent. If the number of blocks indicated in the block header is not zero, it can be compared with this value to ensure the correct number of blocks have been sent. If the block footer reports N blocks sent, then the last block identifier number is (N-1) mod 256.

4. Courier Slave Device Database

The Courier communication language is designed around a standard database structure stored in each individual slave device. The slave device uses this database to store all data and settings which are accessible over the communications link. The structure of the database is very similar to that of a spreadsheet, consisting of individual cells organised into rows and columns. A single menu cell is the smallest unit of access within the database and can therefore contain only one item of data (although it may have other attributes). Menu cells are referenced by their row and column numbers, collectively referred to as a menu cell reference. Menu cell references are expressed in 4 digit hexadecimal notation with the column number appearing as the highest 2 digits and the row number as the lowest 2 digits. The reference for a menu cell in column 10 row 3 would therefore be known as 0A03[2]. This limits the maximum size of the database to a table of 256 columns by 256 rows.

Two of the columns within the database (column 00h and 0BFh) are predefined to access data common to all devices. These are described later in this chapter.

The Courier language is only able to access data within the slave device databases. This differs from previous systems in that the overall system database is now distributed amongst the individual slave devices and not concentrated in the master control unit. The advantage of this is that slave device specific information is embedded within the slave device and not in some specialised master control unit software. Moreover the communications language can be reduced to a set of generic commands which perform specific operations on the distributed database.

[2]Note that when viewing communication messages, the lowest byte of a menu cell is always sent first, so this would be seen in the communication message as two separate bytes: 03 0A

4.1 MENU CELL TYPES

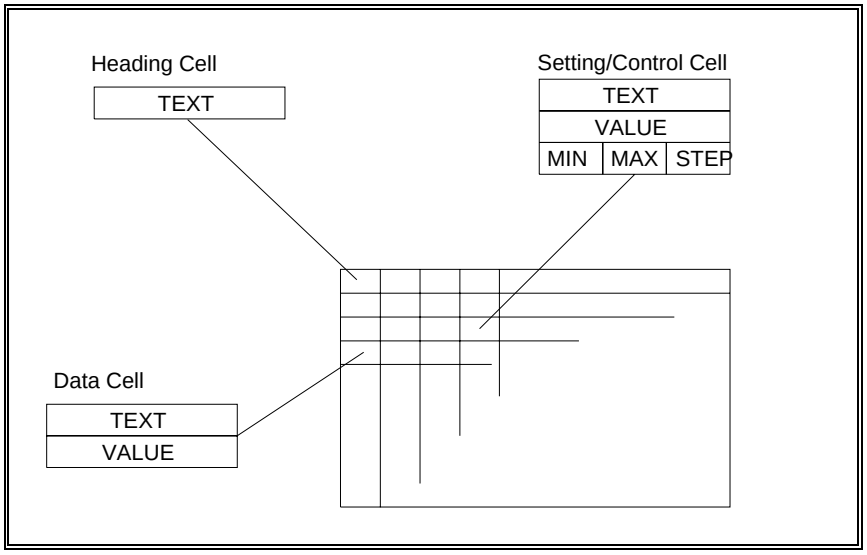


Figure 4-1 Database Layout & Cell Types

The database consists of three different types of cell, each one being a superset of the previous, as in the figure above. The three cell types are heading cells, data cells and setting/control cells.

4.1.1 HEADING CELLS

Heading cells simply contain a piece of descriptive text. These are used throughout the database as place-markers to split the database into different areas. The most common heading cells are the database column heading cells which are used to describe the contents of a column.

4.1.2 DATA CELLS

Data cells contain a piece of text to describe their contents and a value which may be read. Intrinsic in this value is a data type which instructs the master control unit how the data is to be processed. The descriptive text for these cells will also contain some formatting control codes which locate and format how the data will be presented. It is the responsibility of the remote master control unit to extract the text and data value separately and combine them into a displayable form. However, there is an additional command which requests the slave device to combine them itself and to simply return the resultant text string. Typical data cells are measurement values such as phase currents, device information such as model number, wave form records, etc.

4.1.3 SETTING AND CONTROL CELLS

Setting cells are data cells which have text and a data value, but they also allow the data value to be changed. To do this generically, the menu cell contains the setting limits for the data value which specify the setting range and step size. These can be extracted from the slave device using a communication command. There is also a command to send the new value back to the slave device after it has been altered remotely. Typical setting cells are slave device protection settings such as current thresholds.

Control cells are the same as setting cells except the action of setting particular values causes the slave device to perform control functions. Typical control cells are system control cells such as circuit breaker control, etc.

4.2 DATABASE LAYOUT

Individual cells are grouped together into columns of related information such as particular settings, measurements, fault records etc. The cell in the first row of each column is a heading cell which describes the contents of the column. Thus, the contents of any slave device can be read in the same way. First the column headings are extracted and presented to the user as a menu. From this menu the user selects a particular column. The text and values for each cell in the selected column are then extracted and again presented to the user as a menu. Individual cells may then be selected for further operation. Typically this could be change of setting, assignment to a measurement value on a mimic display, log to disc or real time graphing.

4.3 PREDEFINED MENU CELL REFERENCES

In practice it is found that all slave devices contain a certain amount of common information. This includes the device type, model number and serial number, its location, communications address, etc. This information is generally required by the master control unit when the slave device is first detected.

This common information is is stored in two predefined columns of the database: column 00h - System Data Column and column 0BFh - Communication System Data Column. The format of these columns is fixed and allows common information to be extracted in the same way as all other data using the standard generic set of Courier commands, rather than providing special commands.

4.3.1 KEY TO PREDEFINED MENU CELL REFERENCES

Protected: Indicates if the cell requires a password in order to change its value.

Cell Type: The usual type of the cell: Data or Setting/control cell. Data cells may be altered to setting cells to provide enhanced functionality. Setting cells may be altered to data cells to restrict the functionality for a particular application.

Value Type: The type of the data returned including its generic type code. (Refer to chapter 5.1.)

Value Size: The size of the value.

Use: A description of the use of the cell.

Comments: Any additional comments which may be useful.

4.3.2 SYSTEM DATA COLUMN

The first column in the slave database (column 00) is known as the SYSTEM DATA column. All menu cell references in this column are reserved and should not be used by application programmers for other purposes. This column contains standard predefined information

which will be common to all slave devices implementing the Courier language. If a standard item of information is not available on a particular slave device, the associated menu cell should not be used for anything else. All unused menu cell references in this column are reserved for future expansion of the Courier language.

The remainder of this section documents the currently predefined menu cells in the system data column.

0001 LANGUAGE.

Protected: No

Cell Type: Setting

Value Type: (50h) Indexed String

Value Size: 2 bytes

Use: To change the language used to display text.

Comments: For slave devices which can support more than one language, this setting cell determines which language is currently used and will be expressed as an indexed string data type. If the slave device only supports one language, this cell may be omitted.

0002 Password

Protected: No

Cell Type: Setting

Value Type: (1Ch) ASCII Password, Uppercase letters A..Z only.

Value Size: 4 characters

Use: Allows entry of a password to gain entry to a specific access level. Entering the correct password will allow all password protected cells pertaining to the corresponding access level to become settable.

Comments: See chapter 11 "Password Protection" for details on the implementation of password protection and access levels.

0003 System Function Links

Protected: Yes

Cell Type: Setting

Value Type: (20h) Binary flags

Value Size: 16 bits

Use: This setting contains up to 16 individual control flags to enable and disable certain communication and system wide features of the slave device.

Comments: The use of these flags may be varied as no automatic control is performed on this setting. However, it is preferable that controls which match the use of those given below (for the K-Series Overcurrent Relay) are implemented in the same positions for conformity.

Link 0 RemChgSetg	1=	Enable remote setting changes
Link 1 LoadShedT	1=	Enable Load shedding by group
Link 2 Rem CB Ctrl	1=	Enable remote control of circuit breaker
Link 3 Rem Chg Grp	1=	Enable Remote change of setting group
Link 4 Enable Grp2	1=	Enable Group 2 settings
Link 5 AutoReset	1=	Enable auto reset of trip flags and LED
Link 6 Auto Rec	1=	Enable auto reset of disturbance recorder

0004 Description

Protected: Yes

Cell Type: Setting

Value Type: (18h) Text

Value Size: 16 characters

Use: Describes the relay type or scheme configuration, for example "3 Ph Overcurrent". It can be changed by the user to a name which more aptly describes the scheme configuration if the relay is changed from the factory configuration.

Comments: A master control unit will display this value along with the Plant Reference as an aid to identification of a particular slave device, rather than just relying on its address.

0005 Plant Reference

Protected: Yes

Cell Type: Setting

Value Type: (18h) Text

Value Size: 16 characters

Use: Describes the slave device's location on the system, for example: the circuit breaker it may control or the feeder or busbar it may protect.

Comments: A master control unit will display this value along with the Description as an aid to identification of a particular slave device, rather than just relying on its address.

0006 Model Number

Protected: No (Yes, if a setting cell)

Cell Type: Data / Setting Cell

Value Type: (18h) Text

Value Size: 16 characters

Use: Stores the slave device's full model number which encodes the mechanical assembly, ratings and configuration of the slave device. It is the same as the model number printed on the front of the name plate and should be quoted in any correspondence concerning the product.

Comments: If the model number is required to be entered as part of the configuration of the slave device rather than by the user, it should be set via a separate cell in the protected configuration area of the menu.

0007 Firmware Number

Protected: No

Cell Type: Data Cell

Value Type: (18h) Text

Value Size: 16 characters

Use: The firmware number is a unique identification for all the software and memory components used in the slave device.

Comments: The individual software references are also stored in this column (see below).

0008 Serial Number

Protected: No (Yes if a setting cell)

Cell Type: Data/ Setting Cell

Value Type: (18h) Text

Value Size: 7 characters

Use: This menu cell stores the serial number of the slave device as a 7 character string. The first 6 characters form a sequential number and the 7th character encodes the year of manufacture as an uppercase letter in the range 'A' to 'Z'.

Comments: If the model number is required to be entered as part of the configuration of the slave device rather than by the user, it should be set via a separate cell in the protected configuration area of the menu.

0009 Frequency

Protected: No

Cell Type: Data Cell or Setting if programmable

Value Type: Variable, but typically integer

Value Size: Variable, but typically 2 bytes

Use: Identifies the slave device's default operating frequency: normally 50 or 60 Hz.

Comments: None

000A Communication Level

Protected: No

Cell Type: Data Cell

Value Type: (24h) Unsigned Integer

Value Size: 2 bytes

Use: Indicates the current version level of Courier that is used by the slave device to communicate.

Comments: This is for future enhancement of the Courier language and will initially be set to 1. Bit 7 of the value denotes the device is an intermediary device such as a central control unit and therefore further address fields may be added to access lower level devices in the communication hierarchy.

000B Address

Protected: No

Cell Type: Setting

Value Type: (24h) Unsigned integer

Value Size: 1 or 2 bytes

Use: Specifies the address of a Courier slave device on the communication network.

Comments: This is a single byte which ranges in value from 0 to 255.

Address 0 is used for automatic address allocation and address 255 is the global address. Slave devices are supplied with a default address of 255 and will not communicate until this has been changed to another address. For standardisation purposes, this setting may be set as a 2 byte unsigned integer.

000C Plant status word

Protected: No

Cell Type: Data Cell

Value Type: (20h) Binary flags

Value Size: Variable, but typically 2 bytes

Use: This binary value has pre-defined meanings (see section 4.3.4) for each bit-pair which indicate the state of various items of plant connected to the slave device.

Comments: Changing the plant status word should set the plant status flag in the communication status byte. The flag is reset whenever the plant status word is read via Courier.

000D Control status word

Protected: No

Cell Type: Data Cell

Value Type: (20h) Binary flags

Value Size: Variable, but typically 2 bytes

Use: The purpose of this word is to convey application specific flag changes to the master control unit and is defined in section 4.3.5

Comments: Changing the control status word should set the plant control flag in the communication status byte. The flag is reset whenever the control status word is read via Courier.

000E Setting Group

Protected: No

Cell Type: Data Cell

Value Type: (24h) Unsigned integer (1-255)

Value Size: 2 bytes

Use: Indicates the current setting group that is in effect.

Comments: This value is 1 more than the value used to change setting groups with the Select Setting Group command. To select the first setting group, the command uses a value of 0, but this cell will return the value of 1. This cell must not exist if setting groups are not supported.

000F Load Shed Stage

Protected: No

Cell Type: Data Cell

Value Type: (50h) Indexed string

Value Size: 2 bytes

Use: Indicates the current level of load shedding & type that is in effect

Comments: The text for each value is as follows:

0	"None"	No load shedding active
1	"Vreduct 1"	Load Shed to Voltage Reduction Level 1
2	"Vreduct 2"	Load Shed to Voltage Reduction Level 2
3	"Vreduct 3"	Load Shed to Voltage Reduction Level 3
4	"LS TRIP"	Load Shed due to group level
5	"Restoring"	Restoration timer is operating

This cell must not exist if load shedding is not supported.

0010 Circuit Breaker Control

Protected: No

Cell Type: Setting

Value Type: (50h) Indexed string

Value Size: 2 bytes

Use: Allows control of the circuit breakers fitted to the slave device.

Comments: The strings for this value are:

0 "No Operation"
1 "TRIP"
2 "CLOSE"

Where a slave device controls more than one circuit breaker, values 9&10 will trip and close circuit breaker 2 respectively; values 17&18 will trip and close circuit breaker 3, and so on. (Intermediate values are used to control isolators: 3&4 trip & close isolator 1 etc.). The text may be changed to indicate individual phases if required. Note that these circuit breakers correspond to the circuit breakers given in the plant status word (see above and section 4.3.4)

0011-001F Software Reference

Protected: No

Cell Type: Data Cell

Value Type: Text

Value Size: 16 characters

Use: These cells indicate the reference numbers of the individual software components contained in the slave device.

0020 Logic Input Status

Protected: No

Cell Type: Data Cell

Value Type: (20h) Binary flag

Value Size: Variable, but typically 2 bytes.

Use: Indicates the state of all the logic inputs fed into the slave device; a set bit indicating that the associated logic input is being asserted.

Comments: This menu cell reference is used by the master control unit, on extracting events, to determine that the event was due to a logic input changing state.

0021 Relay Output Status

Protected: No

Cell Type: Data Cell

Value Type: (20h) Binary flags

Value Size: Variable, but typically 1 byte.

Use: Indicates the state of all the relay outputs controlled by the slave device; a set bit indicating that the associated relay output is being asserted.

Comments: This menu cell reference is used by the master control unit, on extracting events, to determine that the event was due to a relay output changing state.

0022 Relay Alarm Status

Protected: No

Cell Type: Data Cell

Value Type: (20h) Binary flags

Value Size: Variable, but typically 2 bytes.

Use: Indicates an alarm condition in the relay. Each bit indicates a cause of the alarm. This cell would normally generate an event whenever a bit becomes set, which may also operate a watchdog alarm contact.

Comments: This menu cell reference is used by the master control unit, on extracting events, to determine that the event was due to the relay entering an alarm state.

May also be used as a setting cell to force alarm conditions for testing purposes.

0023 Protection Status

Protected: No

Cell Type: Data Cell

Value Type: (20h) Binary flags

Value Size: Variable, but typically 2 bytes.

Use: Indicates an alarm condition in the relay. Each bit indicates a cause of the alarm. This cell would normally generate an event whenever a bit becomes set, which may also operate a watchdog alarm contact.

Comments: This menu cell reference is used by the master control unit, on extracting events, to determine that the event was due to the operation of a relay protection element that did not generate a fault record.

00D0 Active Access Level

Protected: No

Cell Type: Data Cell

Value Type: (24h) Unsigned integer

Value Size: Variable, but typically 2 bytes.

Use: Indicates the active access level for the current user interface.

Comments: See chapter 11 "Password Protection" for details on the implementation of password protection and access levels.

00D1 Password Control

Protected: Yes, highest access level

Cell Type: Setting Cell

Value Type: (24h) Unsigned integer

Value Size: Variable, but typically 2 bytes.

Use: Indicates the lowest access level that is accessible without a password.

Comments: See chapter 11 "Password Protection" for details on the implementation of password protection and access levels.

00D2 Level 1 Password

Protected: Yes, access level 1

Cell Type: Setting Cell

Value Type: (1Ch) Password

Value Size: 4 bytes.

Use: Allows the password for access level 1 to be changed.

Comments: See chapter 11 "Password Protection" for details on the implementation of password protection and access levels.

00D3 Level 2 Password

Protected: Yes, access level 2

Cell Type: Setting Cell

Value Type: (1Ch) Password

Value Size: 4 bytes.

Use: Allows the password for access level 2 to be changed.

Comments: See chapter 11 "Password Protection" for details on the implementation of password protection and access levels.

00D4-00D8 Reserved

Reserved for future access levels. To be defined the same as cells 00D2 & 00D3.

4.3.3 COMMUNICATION SYSTEM DATA COLUMN

The communication system data column is a reserved column in the third quarter of the slave device's database table located at column BFh.

BF01 Disturbance Record Control Reference

Protected: No

Cell Type: Data Cell

Value Type: (44h) Menu cell reference

Value Size: 2 bytes

Use: Indicates the menu column containing the control setting to start and trigger the recorder at row 01.

Comments: None

BF02 Disturbance Record Extraction Reference

Protected: No

Cell Type: Data Cell

Value Type: (44h) Menu cell reference

Value Size: 2 bytes

Use: Indicates the column of data containing the menu cells used to extract a disturbance record from the slave device.

Comments: None

BF03 Setting Transfer

Protected: No

Cell Type: Setting Cell

Value Type: (24h) Unsigned Integer, or
(50h) String Index (same numeric values)

Value Size: 1 or 2 bytes (only the least significant byte used).

Use: This cell is used to configure the menu in preparation for setting transfer to or from a remote device. The remote device should set this cell in the slave device to 1 prior to performing the setting transfer and set the cell to 0 after the transfer is complete. This action may cause the slave device to alter the visibility of menu cells or even re-arrange them if necessary.

Comments: This cell may not exist if no preparation is required to perform setting transfers, and the remote device should proceed with the setting transfer as normal. If the cell does exist, then the setting must be successful in order to perform the setting transfer. The cell may return ERR_OK or ERR_OKCHANGE in response to changing the setting successfully. Setting extraction is performed by traversing the menu database from beginning to end, sequentially accessing each cell in each column (eg. 0000, 0001, 0002, ... 0100, 0101, 0102, ...).

It is necessary for the slave device to monitor the setting transfer process and reset this cell to 0 after an appropriate timeout, if it determines the transfer process has failed or been interrupted due to a communication breakdown.

BF04 Reset Demand Timers

Protected: No

Cell Type: Data Cell

Value Type: None

Value Size: N/A

Use: Resetting this cell causes the timers controlling the maximum demand measurement period to be reset. It is therefore only implemented in devices providing maximum demand indications.

Comments: As the Reset Menu Cell command can be sent globally, it is a means of synchronising all the demand timers in various devices on a network.

BF05 Reset Event Records

Protected: No

Cell Type: Data Cell

Value Type: None

Value Size: N/A

Use: Resetting this cell makes the events previously sent by a device available for transmission again. As a consequence, the event flag in the status word will be set, indicating that there are events to be extracted.

Comments: Not all devices will support this feature. Events may be discarded as soon as they have been accepted by a remote master, in which case resetting this cell will have no effect.

BF06 Block Transfer Reference

Protected: No

Cell Type: Data Cell

Value Type: (44h) Menu cell reference

Value Size: 2 Bytes

Use: Indicates the column of data containing the menu cells used to perform general block transfers of application data to or from a slave device.

Comments: None

4.3.4 PLANT STATUS WORD

The plant status word is a pre-defined binary word containing information about the state of various pre-defined items of plant which may be controlled by the slave device. It can be found at a reserved location in the system data column of the slave device's database. Each item of plant uses 2 bits in this word, the meaning of which is shown below:

High bit	Low bit	Meaning
0	0	Not fitted
0	1	Open
1	0	Closed
1	1	Undefined

On average there are three controllable isolators per circuit breaker, so the bit pairs are arranged as 1 circuit breaker and 3 isolators per byte as follows:

```
bit 31                                  bit 0
---- ---- ---- ---- ---- ---- ---- --XX  Circuit Breaker 1
---- ---- ---- ---- ---- ---- ---- XX--  Isolator 1
---- ---- ---- ---- ---- ---- --XX ----  Isolator 2
---- ---- ---- ---- ---- ---- XX-- ----  Isolator 3
---- ---- ---- ---- ---- --XX ---- ----  Circuit Breaker 2
---- ---- ---- ---- ---- XX-- ---- ----  Isolator 4
---- ---- ---- ---- --XX ---- ---- ----  Isolator 5
---- ---- ---- ---- XX-- ---- ---- ----  Isolator 6
---- ---- ---- --XX ---- ---- ---- ----  Circuit Breaker 3
---- ---- ---- XX-- ---- ---- ---- ----  Isolator 7
---- ---- --XX ---- ---- ---- ---- ----  Isolator 8
---- ---- XX-- ---- ---- ---- ---- ----  Isolator 9
---- --XX ---- ---- ---- ---- ---- ----  Circuit Breaker 4
---- XX-- ---- ---- ---- ---- ---- ----  Isolator 10
--XX ---- ---- ---- ---- ---- ---- ----  Isolator 11
XX-- ---- ---- ---- ---- ---- ---- ----  Isolator 12
```

The length of this word is variable depending on the number of circuit breakers and is identified when the data is requested. However, 2 bytes will usually suffice.

4.3.5 CONTROL STATUS WORD

The control status word is a binary word used to interchange control information with the master control unit and can be found at a reserved location in the system data column of the slave device's database. There are no specific definitions for the uses of the bits within this word and each slave device is free to define them for its own uses. The master control unit would need to be programmable to perform different actions depending on the state of these bits for each slave device or type of slave device.

5. Courier Data Packet Type Reference

5.1 COURIER DATA TYPES

The data type codes used in Courier are nearly all fundamental data types such as: unsigned integer, signed integer, ASCII text, binary flags, etc. A few specific data types are used to clearly identify particular packets in the message, thus making them unique, but this has been kept to an absolute minimum. The number of type codes is reduced by encoding the length of the data separately. Thus all unsigned integers have the same data type regardless of their length. This makes it easier to write setting and display routines for the different types of data, if they are written to cope with all possible data length values.

Since the data length is separately encoded from the data type of a packet, it is theoretically possible for all data types to have valid lengths 1 to 255 bytes. Although this provides for flexibility in the future, data lengths have been restricted for various data types for ease of implementation.

The data type codes are listed in the following table. The table presents all the DTL values for encoded packet lengths of 1 to 3 bytes. The next byte column sets the length bits to zero and the length of the packet is contained in the *next byte* of the packet; this is the *base* value of the DTL. The inclusion of all length encodings for each type code does not indicate that all lengths are valid. The valid length column indicates the minimum data lengths a master control unit should support. The shaded DTL's are therefore currently invalid and unsupported. The individual descriptions of these data types, which follows, indicate the base value of the DTL.

DTL Code				Data type	Abbrev.	Valid lengths (bytes)
Length						
next byte	1 byte	2 bytes	3 bytes			
00	01	02	03	Extended data type code	DTL_XTYP	Arbitrary
04	05	06	07	Command	DTL_CMD	1,2 or 3
08	09	0A	0B	Group identifier	DTL_GRP	2
0C	0D	0E	0F	Block header	DTL_BLKH	1 to 2
10	11	12	13	Block footer	DTL_BLKF	1 to 2
14	15	16	17	Block identifier	DTL_BLKI	1
18	19	1A	1B	ASCII text	DTL_TEXT	Arbitrary
1C	1D	1E	1F	ASCII password	DTL_PASS	typically 4
20	21	22	23	Binary flags	DTL_BINF	1 to 4
24	25	26	27	Unsigned integer	DTL_UNS	1,2 or 4
28	29	2A	2B	Signed integer	DTL_INT	1,2 or 4
2C	2D	2E	2F	Courier number	DTL_NUM	4
30	31	32	33	Extended Courier number	DTL_XNUM	6
34	35	36	37	IEEE floating point number	DTL_IEEE	4
38	39	3A	3B	Millisecond timer count	DTL_MSTM	4
3C	3D	3E	3F	IEC870 time & date	DTL_IECD	7
40	41	42	43	reserved		
44	45	46	47	Menu location	DTL_MENU	2
48	49	4A	4B	Reply Codes	DTL_REPY	1
4C	4D	4E	4F	reserved		
50	51	52	53	String index	DTL_ISTR	1 to 2
54	55	56	57	reserved		
58	59	5A	5B	Block transfer cell	DTL_BTFR	1
5C	5D	5E	5F	Status Byte	DTL_STAT	1
60	61	62	63	IEC870 control byte	DTL_CTRL	1
64	65	66	67	Foreign Data	DTL_FRGN	Arbitrary
68	69	6A	6B	Modem Control Strings	DTL_MODM	Arbitrary
6C	.....	.....	FF	Reserved		

Table 5-1 Courier Data Types

5.1.1 EXTENDED DATA TYPE CODE (00H)

At present there are no extended data type codes defined. These are for future expansion of the type codes if required. Extended type code 0 is reserved for further extension.

5.1.2 COMMAND CODES (04H)

Command codes are used to instruct a slave device to perform some action (such as returning the value of a menu cell) and control the data that flows over the communication network. Each command data packet consists of a single byte command code and up to 2 additional bytes to specify an implicit argument. Implicit arguments are dependent on the command code and are listed in the command reference (chapter 6) for each command.

Implicit arguments are not data typed, but should be treated as 2 byte unsigned integers. Where only 1 byte of the implicit argument is specified, this should be zero-extended to a 2 byte unsigned integer.

Additional arguments must be explicit, i.e. they are encapsulated in their own data packets which follow the command code data packet. Explicit arguments are data typed and must occur in the correct sequence for the given command. The command packet serves the same purpose as the group identifier packet does in replies from a slave device, as it marks the beginning of a command group, although the total length of the data packets in the command group is not included. No further grouping is required since the next command group will begin with a command code data packet.

Chapter 6 details each command in the Courier language including the code sequence, expected reply etc. and an example of how it should be used.

5.1.3 GROUP IDENTIFIER (08H)

In a Courier transaction[3] there is a one-to-one correspondence between a request and a reply. This enables several requests to be packed into one message and their appropriate responses to be packed into the reply message and still have the ability to match the correct responses to the correct requests. Where a single request packet results in several response packets, the response packets need to be associated together so that they can be included in a multiple response transaction and still match the correct responses to the correct requests. (see Group Transactions).

A group identifier is a special data packet which precedes a group of several data packets, which are the response to a single request, in order to group them together into a single entity. A group identifier packet has a data field of 2 bytes. The first byte is a group type which identifies the sequence and representation of the data packets to be grouped together. The second byte of the data field is the total length of the following data packets in the group.

[3] Courier transactions are described in detail in Chapter 3.

The following example of a fictitious group packet consists of 2 integer data packets with a total group length of 6 bytes.

DTL			DTL#1		DTL#2	
Group Identifier	Group Type	Group Length	Integer Length 2	Value	Integer Length 2	Value
0A	GT	06	2A	00 00	2A	00 00

The group type is used to identify the organisation of the data packets within the group and is used to provide additional integrity checking, allowing devices to reject unexpected groups. It is more concerned with the number and semantics of the data packets rather than specifying their data types, although these will be fixed for some group types.

The group length binds all the data packets together and enables a receiver of a message to skip these reply packets if they are in error, without having to ignore the remainder of the message as well. The maximum group length is 255, but since an entire group must fit within one message, which has a maximum user data field length of about 230 bytes, this is not a limitation. Larger data transfers are sent using several group packets within a blocked transaction.

There are several group types which define the data packet order in a Courier message. These are described in the next section and are summarised in the following table:

GROUP TYPE	GROUP TYPE
00h	Standard event record
01h	Short event record
02h	Long event record
03h	Complex event record
11h	Column Heading Group
12h	Column Text Group
13h	Column Value Group
20h	Indexed string Group
21h	Setting Limits Group
22h	Setting Limits with Multiplier Group
23h	Column Setting Limits Group
24h	ColumnSetting Limits with Multiplier Group
30h	reserved
40h	Repeated Data Packet

Table 5-2 Courier Group Types

5.1.4 BLOCK HEADER (0CH)

A block header appears as the only data packet in a reply message user data field which initiates a blocked transaction[4]. The data field is an unsigned integer which informs the master control unit how many blocks are going to be sent in the transaction. If the number of blocks is set to zero, the number of blocks that will be sent is unknown.

[4] Courier transactions are described in detail in Chapter 3.

5.1.5 BLOCK FOOTER (10H)

A block footer packet appears as the only data packet in a reply message at the end of a blocked transaction. The data field contains an unsigned integer indicating the total number of blocks transferred in the transaction which should be compared with the actual number of blocks received.

5.1.6 BLOCK IDENTIFIER (14H)

A block identifier packet appears at the beginning of each blocked transaction message that appears between a block footer and a block header message. The data field is a single byte unsigned integer which indicates the sequence number of the block. These numbers start at 0 and wrap around to zero after the value 255. The blocks must be sent in sequence.

5.1.7 TEXT (18H)

Text strings shall be sent as a number of characters from the Courier Character Set with the potential to include formatting codes to control the positioning and display of numeric or binary information. The string is not null terminated since its length is included in the DTL of the data packet.

String formatting shall be achieved by utilising a similar mechanism to that employed by the "C" language, i.e. the use of embedded formatting strings beginning with the character "%". Should it be necessary to send a real "%" character, then "%%" should be sent. Escape sequences as used by the 'C' language are not supported. Chapter 14 specifies the format, control codes and character set that can be used.

5.1.8 PASSWORD (1CH)

This data type is used to transmit passwords or to access password protected cells. The data field has the same format as the Text data type. It is not encrypted. Passwords are normally 4 characters in length. The range of values for each character comprises the uppercase letters in the range ‘A’ ... ‘Z’ and the asterisk ‘*’. The asterisk is only valid when reading a password, when all characters will be replaced with it to conceal the actual password. It is not valid when setting the password.

5.1.9 BINARY FLAGS (20H)

Flag values (on or off) can be sent as one bit per flag with 0 meaning off, false or disabled and 1 meaning on, true or enabled. Flags may consist of 1 to 4 data bytes (normally 2 bytes). They will always be sent with the least significant byte first. Unused bit positions will be set to zero. Where the transmitted flags contain less bytes than expected, the higher bits of the flags are padded with zeros. If the flags contain more bits than expected, the higher bits are truncated and lost. A binary flag setting may have an array of strings to describe the purpose of each flag which can be requested using the Get Strings command. The strings are associated with the bit positions of the flags, starting at bit position 0.

5.1.10 UNSIGNED INTEGER (24H)

Unsigned integer numbers are always least significant byte first. They will usually be of 2 bytes in length, but see the discussion on valid data lengths in section 5.1

5.1.11 SIGNED INTEGER (28H)

Signed integer numbers are always least significant byte first. Negative numbers are stored in two's complement form with the sign bit in the highest bit position. They will usually be of 2 bytes in length, but see the discussion on valid data lengths in section .

5.1.12 COURIER NUMBER TYPE (2CH)

The Courier Number type was created to allow a large dynamic range of numbers to be displayed with four significant digits, a variable decimal point location and an inherent unit type prefixed with a scalar multiplier, without the overhead of using floating point arithmetic. Accuracy was not considered vital for these numbers.

A Courier number consists of three fields: a mantissa, an exponent and a units field. In its standard implementation, these fields are arranged into 4 bytes as shown in the following diagram, with the least significant byte shown first (transmission order):

7	6	5	4	3	2	1	0	= bit position
M^7	M^6	M^5	M^4	M^3	M^2	M^1	M^0	Mantissa M = 0..9999
S	M^{14}	M^{13}	M^{12}	M^{11}	M^{10}	M^9	M^8	S = sign bit: 0=positive, 1=negative.
E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0	E=Exponent
U^7	U^6	U^5	U^4	U^3	U^2	U^1	U^0	U=units

Mantissa
The mantissa is stored in the two least significant bytes as a 15-bit binary unsigned integer with a separate sign bit in the highest bit. Numbers are usually stored in a normalised format such that the mantissa is in the range 1000 to 9999. For mantissas less than 1000, this is achieved by successively multiplying the mantissa by 10, whilst at the same time, decrementing the exponent by 1 (implying a division by a factor of 10).

Units
The units field is stored in the most significant byte to indicate the physical quantity of the value represented.

The following base units have been defined:

Byte Value(hex)	Physical Quantity	Base Unit	Display
00h	Current	Amps	A
01h	Voltage	Volts	V
02h	Angle	Degrees	Deg
03h	Impedance	Ohms	Ohms
04h	Power	Watts	W
05h	VA	VA	VA
06h	VAr	VAr	VAr
07h	Length	metres	m
08h	Time(interval)	Seconds	s
09h	Ratio	xxxx:1	xxxx:1
0Ah	Temperature	°C	°C
0Bh	Frequency (Speed)	Hertz (revs/sec)	Hz
0Ch	Percentage	%	%
0Dh	Per Unit Value	Per Units	PU
0Eh	Square Amps	Square Amps	A^2
0Fh	Reserved	decimal, no units	decimal, no units
10h	Energy	Watt hours	Wh
11h	Energy	VAh	VAh
12h	Energy	VArh	VArh
13h	Time (interval)	Minutes	mins
14h	Inverse ohms	mho	mho
15h	Volts per Hertz	Volts per Hertz	V/Hz
16h	Rate of change of frequency	Hertz per second	Hz/s

Table 5-3 Courier Number Units

Courier numbers with units of code F have no units and therefore no scalar quantity. This also applies to unit codes 9, C & D. The value of these numbers is therefore restricted between 0.001 and 9999. (see Chapter 14 Text Formatting Characters, regarding the handling of out of range numbers).

Exponent

The exponent is stored in the third byte and indicates the power of 10 that the mantissa should be raised to. This is an unsigned byte with an inherent offset of 126 and can therefore express decimal exponents from 10^{-126} to 10^{+129}. However, this is usually restricted to the range 10^{-18} to 10^{+18} since there are no defined scalar multipliers in SI units to express numbers outside of this range.

The 16-bit mantissa is multiplied by a power of ten as indicated by the value of the exponent byte. This byte is coded as indicated by the examples in the following table:

Exponent Value (decimal)	Normalised Multiplier
105	10^{-21}
108	10^{-18}
111	10^{-15}
114	10^{-12}
117	10^{-9}
120	10^{-6}
123	10^{-3}
125	10^{-1}
126	10^{+0}
127	10^{+1}
129	10^{+3}
132	10^{+6}
135	10^{+9}
138	10^{+12}
141	10^{+15}
144	10^{+18}

Table 5-4 Courier Number Exponents

Display of Courier Numbers

The value of the Courier type can be expressed in the following form:

$$\text{Value} = (1\text{-}2.S)(M) \times 10^{E\text{-}126} \text{ Units.}$$

and value is restricted to the range $1x10^{-18}$ to $9.999 \times 10^{+21}$.

Courier numbers are not displayed in exponential format. Instead, the appropriate scalar multiplier is calculated from the exponent and precedes the units. For example, 1000×10^{0}V is displayed as 1.000kV, and 1000×10^{3}A is displayed as 1.000MA etc.

The position of the decimal point is adjusted depending on the value of the exponent. The following are examples of the positioning:

Value	Display
1000×10^{-4}A	100.0mA
1000×10^{-3}A	1.000 A
1000×10^{-2}A	10.00 A
1000×10^{-1}A	100.0 A
1000×10^{0}A	1.000kA

The mantissa is normally in the range 1000 to 9999, thus providing 4 significant digits. Less digits are displayed for mantissas less than 1000. For example, the following values are equivalent, but are displayed differently:

Value	Display
1000 x 10^{-3}A	1.000 A
10 x 10^{-1}A	1.0 A

(Further information regarding the display of Courier Numbers is given in Chapter 14 Text Formatting Characters.)

5.1.13 EXTENDED COURIER TYPE (30H)

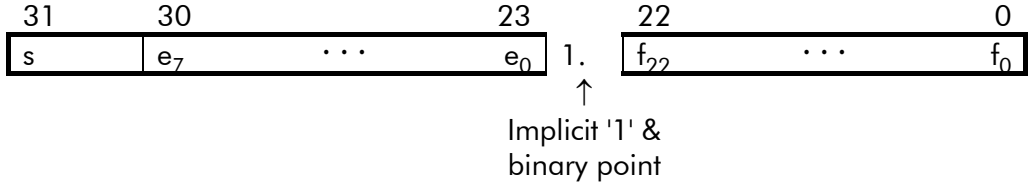
The Extended Courier type is similar to the Courier Type 26h, but has an extended mantissa to provide for 9 significant digits. This is achieved by extending the mantissa field to 4 bytes whilst maintaining the sign bit in the highest bit position of this field. The maximum value of the mantissa will therefore be 999,999,999. The length of the Courier Number Type data packet will therefore be extended from 4 bytes to 6 bytes.

7	6	5	4	3	2	1	0	= bit position
M^7	M^6	M^5	M^4	M^3	M^2	M^1	M^0	Mantissa M = 0..999,999,999
M15	M^{14}	M^{13}	M^{12}	M^{11}	M^{10}	M^9	M^8	Mantissa M = 0..999,999,999
M^{23}	M^{22}	M^{21}	M^{20}	M^{19}	M^{18}	M^{17}	M^{16}	Mantissa M = 0..999,999,999
S	M^{30}	M^{29}	M^{28}	M^{27}	M^{26}	M^{25}	M^{24}	S = sign bit: 0=positive, 1=negative.
E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0	E=Exponent
U^7	U^6	U^5	U^4	U^3	U^2	U^1	U^0	U=units

5.1.14 IEEE FLOATING POINT NUMBER (34H)

The format of the floating point number data type is based on the IEEE 754 (1985) standard for single precision floating point numbers. This has also been adopted by the IEC 870 telecontrol standard. The four bytes of the number are sent least significant first.

The IEEE Standard specifies a 32 bit format, shown below, which consists of a sign bit *s*, a 24 bit significand, and an 8 bit unsigned magnitude exponent *e*. For normalized numbers, the significand consists of a 23 bit fraction *f* and an implicit 1 that is presumed to precede f_{22}. The binary point is presumed to lie between this implicit 1 and f_{22} and effectively increases the precision of the floating point significand to 24 bits from the 23 actually stored in the data format. It also ensures that the significand of any number in the IEEE normalized-number format is always greater than or equal to 1 and less than 2.



The unsigned exponent *e* can range between $1 \leq e \leq 254$ for normal numbers in the single precision format. This exponent is *biased* by +127. To calculate the true *unbiased* exponent, 127 must be subtracted from e. The sign bit *s* holds the sign of the real number, a '1' indicates a negative number and '0' a positive one. Thus, for an exponent value of e, a fraction value of *f* and a sign bit *s*, the number being represented is: $(1-2s)((1+f) \times 2^{e-127})$.

For example, consider the following 32 bit single precision floating point number:

```
1 10000001 01000000000000000000000
```

The sign bit is '1'; the floating point number is negative. The exponent field is 129, making the exponent 129 - 127 = 2. The fraction part is $.01_2 = .25$, making the significand 1.25. Thus, this bit pattern represents the number $-1.25 \times 2^2 = -5$.

The IEEE Standard also provides for several special data types in the single precision format:

- An exponent value of 255 (all ones) with a non-zero fraction is Not-a-Number (NaN). NaN's are usually used for uninitialized values and for the results of invalid operations such as $0 \cdot \infty$.
- Infinity is represented as an exponent of 255 and a zero fraction. Both positive and negative infinity can be represented.
- Zero is represented by a zero exponent and a zero fraction. As with infinity, both positive zero and negative zero can be represented.
- A non-zero fraction and a zero exponent is a denormalized number. Denormalized numbers are used when a value smaller than the smallest normalized number is required, for the single precision format this is values below 1.0×2^{-126}. This provides a facility for gradual underflow.

5.1.15 TIMER COUNTS (38H)

Timer counts are stored as unsigned integer values of 4 bytes in length, sent least significant byte first. A count of 1 is equivalent to 1 millisecond and the maximum count is 49 days 17 hours 2 mins 47.295 secs.

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

5.1.16 IEC 870 TIME AND DATE CODES (3CH)

This data type is specified in IEC870-5-4 First Issue 1993-08 as information element "binary Time 2a" which consists of 7 bytes listed here in least significant byte order (transmission order).

7	6	5	4	3	2	1	0	= bit position
m^7	m^6	m^5	m^4	m^3	m^2	m^1	m^0	m = 0..59,999ms
m^{15}	m^{14}	m^{13}	m^{12}	m^{11}	m^{10}	m^9	m^8	
IV	R	I^5	I^4	I^3	I^2	I^1	I^0	I=0..59 Mins
SU	R	R	H^4	H^3	H^2	H^1	H^0	H=0..23 Hours
W^2	W^1	W^0	D^4	D^3	D^2	D^1	D^0	W=1..7 Day of week D=1..31 Day of Month
R	R	R	R	M^3	M^2	M^1	M^0	M=1..12 Month of year
R	Y^6	Y^5	Y^4	Y^3	Y^2	Y^1	Y^0	Y=0..99 Years

R = Reserved bit = 0
SU = summertime: 0=standard time, 1 = summer time
IV = invalid value: 0=valid, 1=invalid
range = 0ms .. 99 years

The differences between this format and "binary time 2a" are
1. www may be zero
2. All seven bytes must always be sent.

Alternative formats are not implemented.

- Since the range of this time and date is only 100 years, the century is calculated as the one which would produce the nearest time value to the point of reference. For example: 30-12-99 is 30-12-1999 when received in 1999 & 2000, but is 30-12-2099 when received in 2050.
- The IEC 870 standard does not indicate what the day of week or month of year numeric representations are. However, with reference to BS7151:1989 (ISO8601:1988), day 1 to 7 is Monday to Sunday and month 1 to 12 is January to December. If the day of week information is not known, a value of zero can be used for this field.
- Since the real time record format provides no indication of time zone (universal co-ordinated time) the interpretation of real time information is user defined, but will typically be local time.
- The invalid flag is to be set (to 1) under the following conditions:
  1. The time-date information is not known at all. In which case the remaining fields of the time-date packet should be *fixed-up* with legal values, but for a time which will be obviously wrong to an end user. In this respect it is suggested that the time fields are set to 00:00:00.000 and the date fields are set to 1-January with the year of century set to the year (or previous year) of the products development.
  2. The time-date information is valid, but its absolute accuracy is unknown. This will be the case for:
     i) products with software generated RTC's which start from some predefined datum on power-up, but will not be accurate until they are synchronised to the outside world.
     ii) products which have missed an expected time synchronisation event (this can be qualified by worst case crystal drift, etc.).

5.1.17 MENU LOCATION REFERENCES (44H)

The menu is of the same format for all relays. This takes the form of a table with each cell being addressed by its column and row number. Each menu cell is identified by a two byte word, with the highest byte identifying the column and the lowest byte the row. Thus allowing a maximum table size of 256 columns by 256 rows, i.e. 65536 cells.

	00h	01h	02h	03h	---	Cols	---	FFh
00h	0000	0100	0200	0300				
01h	0001	0101	0201	0301				
02h	0002	0102	0202	0302				
03h	0003	0103	0203	0303				
|								
Rows								
|								
FFh								FFFF

The menu table may be considered to be a form of spreadsheet where the cells can be filled with text, values, and functions. The first row (00h) of each column shall contain a text heading for the type of data in that column. This text will be used in the formats that will be developed to the display data.

The column and row number identifying a cell, in future referred to as a menu cell reference, shall also be used within a communication message to reference the contents or attributes of that cell.

5.1.18 REPLY CODES (48H)

Reply codes are returned as acknowledgements or to indicate an error in a request. These errors do not include communication errors which result in a complete message being ignored. The following table indicates the symbolic name by which the reply codes are referred to in the remainder of this document and their general interpretation.

ERR_OK	00h	Positive acknowledgement.
ERR_NOCODE	01h	Given menu location does not exist.
ERR_NODATA	02h	Menu cell has no data.
ERR_NOACCESS	03h	Cell cannot be accessed at the moment.
ERR_NOVERIFY	04h	Verify error on setting change.
ERR_NOSETTINGS	05h	This is not a settable cell.
ERR_NOPASSWORD	06h	Password is required to change setting.
ERR_LOCAL	07h	Setting is currently being changed by another user interface.
ERR_OKCHANGE	08h	Same as ERR_OK, but column and column headings should be subsequently re-read.
ERR_INVALIDCMD	09h	The command is not known or is not valid at this time.
ERR_GENERAL	FFh	Other non-specific error.

5.1.19 STRING INDEX (50H)

A menu cell which has a string index type has an unsigned integer type setting which has a minimum value of 0 and a step increment of 1. Each value in the setting's range is represented by a textual string. Null strings should be represented by a single space character.

When the value is requested using the Get Value command, the slave device will either return this data type, or more usually, will convert the data type to TEXT and return the correct textual string for the current value of the menu cell's setting.

When Get Value returns a type of string index, the master must request the indexed strings separately in order to display the value correctly.

When the value is converted to the TEXT data type, the strings must all be of the same length so that the data returned by the Get Value command does not vary. This is for application in Master Device polling applications that allocate just sufficient memory to store the original length of the data on the first Get Value command.

The cell value data type is always returned as a String Index within the setting limits group; the remaining limits will be of type unsigned integer. The master control unit will request the strings for this setting using the Get Strings command so that it can display the correct textual string for whatever value the setting may be set to.

5.1.20 RESERVED (54H)

This data type is an internal data type used in some slave devices.

5.1.21 BLOCK TRANSFER CELL (58H)

A menu cell which results in a block transfer when its value is requested cannot initiate a block transfer during the Get Column Values command since blocked transactions cannot be nested. Therefore, during a Get Column Values command, the menu cell will return a data type of Block Transfer Cell. The Data field is meaningless and should contain 0. The length is normally set to 1 byte. During a Get Value command the usual block header will be returned.

5.1.22 STATUS BYTE (5CH)

The status byte is a single byte binary flag type value. This data type is used to signify the end of the communication header and the start of the user data field. It is present in all slave device responses except the reply to a Reset Remote Link command which simply returns the IEC870 control byte.

The status byte consists of 8 flags to indicate various items of status information in the slave device.

7	6	5	4	3	2	1	0
TRIP	ALARM	EVENT	OOS	BUSY	CONTROL	PLANT	DIST

TRIP flag
The trip flag is used to indicate the state of the trip LED on the front of the slave device. It is used for annunciation purposes on mimic diagrams to indicate trip states and possibly for

mimicking a slave device's display. The clearance of this flag is application dependent, although the Reset Trip Indication command would normally reset this flag provided the trip condition does not still exist.

ALARM flag
The alarm flag is used to indicate the state of the alarm LED on the front of the slave device. It is used for annunciation purposes on mimic diagrams to indicate alarm states and for mimicking a slave device's display. The clearance of this flag is application dependent.

EVENT flag
The event flag is set whenever a slave device contains at least one event record. The master control unit should extract all events from a relay as a high priority whenever it sees this flag set.

OOS flag
The Out Of Service flag is set whenever the slave device is out of service due to a detected error, an appropriate control command, test condition or if the slave device has been put into calibration or configuration mode. This flag indicates that the protection is not running.

BUSY flag
The busy flag is set when the relay has not had sufficient time to form the reply to the previous request within the time-out period. The master control unit will poll the relay with the POLL_STATUS or POLL_BUFFER when this flag is set, until the flag is reset, which indicates that the reply is now available. The reply can then be extracted using the POLL_BUFFER command. When this flag is set, all other status flags and timer count values in the message should be ignored since the busy response message may have been returned from an intermediate device in an hierarchical system.

CONTROL flag
The control flag indicates that a binary word in the slave device called the 'Control Status Word' has changed its value. The master control unit should then read the value of this word as a normal Get Value request from the appropriate cell in the System Data column of the slave device's menu, the action of which will reset this flag. See section 4.3.2.

PLANT flag
The plant flag indicates that a binary word in the slave device called the 'Plant Status Word' has changed its value. The master control unit should then read the value of this word as a normal Get Value request from the appropriate cell in the System Data column of the slave device's menu, the action of which will reset this flag. See section 4.3.2.

DIST flag
Slave devices which contain a disturbance or wave form recorder will set this flag to indicate that they have a disturbance record ready to be extracted. Once extracted, the clearing of this flag is implementation dependent. It is usually performed by starting the recorder capturing data again.

5.1.23 IEC870 CONTROL BYTE (60H)

The IEC870 control byte is a single byte control field in IEC870 frames and is part of the Courier protocol. This packet enables the value to be transferred over K-Bus and other mediums where it is not part of the frame structure.

7	6	5	4	3	2	1	0	
RES	PRM=1 =0	FCB ACD	FCV DFC	Function				Master---->Slave Slave---->Master

RES: Reserved.
This is always 0

PRM: Primary message.
Indicates the direction of the message, 1 for master to slave messages, 0 for slave to master messages.

FCB: Frame Count Bit.
An alternating bit 0/1 for successive SEND/CONFIRM or REQUEST/RESPOND services per station. The Frame Count bit is used for suppressing the duplications of information transfers: The primary station alternates the FCB bit for each new SEND/CONFIRM or REQUEST/RESPOND transmission service directed to the same secondary station. Thus the primary station keeps a copy of the Frame Count Bit per secondary station. If an expected reply is timed-out (missing) or garbled, then the same SEND/CONFIRM or REQUEST/RESPOND service is repeated with the same FCB state.

The FCB bit is always zero for reset commands. After a reset command, the secondary station will expect the next message received from the primary station with the FCV bit set to have the FCB bit set also.

FCV: Frame Count Valid.
0 = alternating function of FCB bit is invalid.
1 = alternating function of FCB bit is valid.
SEND/NO REPLY service, broadcast messages and other transmission services that ignore the suppression of duplication or loss of information output do not alternate the FCB bit and indicate this by a cleared FCV bit; the state of the FCB bit is then undefined.

DFC: Data Flow Control.
Not used; Courier is a balanced transmission protocol. This bit should be set to 0.

ACD: Access Demand.
Not used. In the Courier environment, the function of this bit is replaced by the EVENT bit in the status byte. This bit is redundant and should be set to 0.

Functions.
The functions shown below are those used by the Courier protocol.

Function Codes of control field of messages sent from a master control unit

Function	Frame Type	Service Function	Use	FCV
0	Send-Confirm	Reset Remote Link	Resetting and identifying slave devices.	0
4	Send-No reply	User Data	Global Messages	0
11	Request-Respond	Request User Data Class 2	Normal Courier Request	1

Function Codes of control field of messages sent from a Slave device.

Function	Frame Type	Service Function	Use
0	Confirm	ACK: Positive acknowledgement of Reset Remote Link	Indicates slave device is present.
8	Respond	User Data	Normal Courier Response

5.1.24 FOREIGN DATA (64H)

This data type is typically used to interface other communication languages into a Courier communication system. A foreign message from another communication system can be packaged into a data packet of this type and transmitted across the Courier system, effectively using Courier as a transport mechanism. It may be used in both request and response messages since Courier does not make any assumptions about the data. No display standard is defined for this data, but typically it would be displayed as either raw bytes or as ASCII text.

5.1.25 MODEM CONTROL STRINGS (68H)

These are strings which are used to carry out modem initialisation, command the modem to dial a particular number or to disconnect. As individual modems vary in their abilities, the string cannot be standardised and must be entered by the user. The modem control string described here is the string stored in the slave and presented to the user; this is more than just the sequence of characters sent to the modem. The modem control string requires additional information to indicate when the transmission to the modem should be paused, when responses are expected from the modem and other actions.

Modem Control String Definition
Each string consists of eight bit characters, the length to be dependent on the individual modem but a minimum of 64 characters is to be provided. Characters with the MSB set are to be interpreted as control characters indicating special actions that may be required during the set-up or dialling procedure, these actions are taken by the slave (DTE) controlling the modem (DCE). Other characters are assumed to be ASCII characters (in the range 0 to 7Fh) to be sent to the modem. The length of the string determines the maximum number of character locations. An End of Control String character (0ffh, {ST}) shall be used to signify the end of the defined characters. All characters beyond the End of Control String character shall be space characters (20h) and shall be ignored when using the string.

Control Characters
The valid modem control characters comprise the Modem Character Set, detailed in section 14.3.3. It is suggested during the editing of these strings that the control characters be replaced by a sequence of ASCII characters which can be identified as a control character. Alternatively, as the cursor is moved over each of these control characters, the description of the control character could appear in a secondary edit window which could be selected from a known list.

5.2 COURIER GROUP TYPES

5.2.1 GROUP 00H - STANDARD EVENT RECORD

This is the standard group for event records returned in response to the Send Event command. There are four data packets in the group specifying the menu location, text description, new

data value and time tag of the event. This is the minimum amount of information required for any event. (see Chapter 7 - Event Records, for a fuller description of events).

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value

Events transmitted using this record include local setting changes, logic input changes and relay output changes.

5.2.2 GROUP 01H - SHORT EVENT RECORD

A short event record is used where information is required to be sent in addition to that defined in the standard event record. This information is appended to the standard event record structure as a format text data packet followed by a sequence of argument data packets (Arg 1.. Arg N) with varying data types. The format text data packet contains formatting control codes for each of the following arguments in order to display the additional information in a presentable format. The additional information must fit within a single message frame of the event record, which restricts the size of the format text packet.

The event type is normally used for conveying simple fault records. (see Chapter 7 - Event Records, for a fuller description of event and fault records).

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value
5	18h+LL	Format text
6	xx	Arg 1
7	xx	Arg 2
...	...	...
n	xx	Arg n

5.2.3 GROUP 02H - LONG EVENT RECORD

For larger fault records, it may be necessary to store a longer string of format text. Type 2 event records allow this by replacing the format text packet with a data packet containing a menu cell reference from which the actual format text can be extracted using the Get Text Courier command. This allows more room in the event record for the fault data packets. The menu cell from which this text is extracted should not normally be accessible via the menu system. It may therefore be located in a column which has no column heading.

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value
5	46h	Format text menu cell
6	xx	Fault data 1
7	xx	Fault data 2
...	...	...
n	xx	Fault data m

5.2.4 GROUP 03H - COMPLEX EVENT RECORD

Event types 1 and 2 are ideal for simple fault records, but are limited in the total length of the data packets that the fault record can be composed of. Other slave devices may store several fault records and each one may be very large, looking more like a report.

Event type 3 is used to specify an entire column for a complex fault record which can be extracted automatically at a later time and printed like a report. It includes the usual event information of menu cell, time tag, text and value which will be printed out as a normal event record - the text and value will usually contain the fault flag or type of trip indication. This is followed by a menu cell reference which indicates the column which contains the fault record. The Get Column Text and Get Column Values commands can then be used to extract the fault record. This type of event record also contains an event record number so that multiple events can be stored in the same column. Before performing the extraction, the value of the setting cell in row 1 of the fault record column should be set to the record number specified in the event record. The remaining cells in the column will change to reflect the record selected and then the record can be extracted.

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value
5	46h	Fault Record column
6	26h	Record number

If the fault record column is also accessible to the user, steps must be taken to ensure the fault record number cannot be set manually whilst a fault record extraction is in progress, otherwise two fault records will become mixed together. To prevent this from happening, two fault record columns could be provided: one is accessible to the user to allow viewing of all fault records manually, whilst the other is only directly accessible by the communications since the column heading is hidden, and therefore cannot be changed whilst an extraction is in progress.

5.2.5 GROUP 11H - COLUMN HEADING GROUP

A column heading group is returned in response to a Get Column Headings command. The replies will be sent as a blocked transaction with each block containing one or more of the following grouped packets:

Packet#	DTL	Use
1	46h	Menu Cell Reference
2	18h+LL	ASCII Text description

5.2.6 GROUP 12H - COLUMN TEXT GROUP

A column text group is returned in response to a Get Column Text command. The replies will be sent as a blocked transaction with each block containing one or more of the following grouped packets:

Packet#	DTL	Use
1	46h	Menu Cell Reference
2	18h+LL	ASCII Text description

5.2.7 GROUP 13H - COLUMN VALUE GROUP

A column value group is returned in response to a Get Column Values command. The replies will be sent as a blocked transaction with each block containing one or more of the following grouped packets:

Packet#	DTL	Use
1	46h	Menu Cell Reference
2	xx	Menu Cell Value

5.2.8 GROUP 20H - INDEXED STRING GROUP

The indexed string group is returned in response to the Get Strings command. The number of packets in the group is variable and needs to be determined by the receiver. Each data packet is of type ASCII text and contains the text for each value for an indexed string value, or the name of each bit for a binary flag value.

If the total length of the strings is too long to fit into one message, a blocked transaction may be used in which each block transfer contains one indexed string group of an arbitrary number of strings.

Packet#	DTL	Use
1	18h+LL	ASCII string 1
2	18h+LL	ASCII string 2
..	18h+LL	ASCII string ..
N	18h+LL	ASCII string N

Note that for an indexed string setting type, the strings should all be the same length. This condition does not apply to binary flag settings.

5.2.9 GROUP 21H - SETTING LIMITS GROUP

There are two possible setting limits groups that could be returned in response to the Enter Setting Mode command, this being the first and most common. It contains four data packets in the order shown below. The DTL of the menu cell value may be different to the other three data packets which must be of the same type.

Packet#	DTL	Use
1	yy	Menu cell value
2	xx	Minimum value
3	xx	Maximum value
4	xx	Step/increment value

The allowable steps are based on the minimum value, e.g. with a minimum value of 7 and a step value of 5 the allowable steps will be 7, 12, 17, etc. If the stated maximum does not fall on an allowable step, the true maximum will be the highest allowable step below the stated maximum, e.g. in the above example a stated maximum of 20 will result in a true maximum of 17.

5.2.10 GROUP 22H - SETTING LIMITS WITH MULTIPLIER GROUP

This is the second type of setting limits group that may be returned in response to the Enter Setting Mode command. It is used for Courier Numbers values which require more accuracy than the standard Courier numbers, or those which lose resolution because of the actual step size value, but are still within the dynamic range of the setting. An example would be a current setting with a minimum value of 0A, a maximum value of 8A and a step size of 2.5 mA. Although the dynamic range (8A-0A/2.5 mA = 3200) is small enough for numeric values (<10000), the resolution of the step size is too small to be resolved with values above 1A, i.e. 1.000A + 2.5 mA = 1.002A, which is rounded down.

The setting is treated as a standard integer value, except that all values are multiplied by the multiplier value before being displayed. This prevents consecutive rounding errors from accumulating. Although the display will be rounded down, the actual value used will still be correct.

Packet#	DTL	Use
1	2Ah	Menu cell value
2	2Ah	Minimum value
3	2Ah	Maximum value
4	2Ah	Step/increment value
5	2Ch	Multiplier value

5.2.11 GROUP 23H - COLUMN SETTING LIMITS GROUP

There are two possible column setting limits groups that could be returned in response to the Get Column Setting Limits command, this being the first and most common. It contains five data packets in the order shown below. The DTL of the menu cell value may be different to the other three data packets which must be of the same type.

Packet#	DTL	Use
1	46h	Menu cell
2	yy	Menu cell value
3	xx	Minimum value
4	xx	Maximum value
5	xx	Step/increment value

5.2.12 GROUP 24H - COLUMN SETTING LIMITS WITH MULTIPLIER GROUP

This is the second type of column setting limits group that may be returned in response to the Get Column Setting Limits command. It is used for Courier number values which require more accuracy than the standard Courier numbers, or those which lose resolution because of the actual step size value, but are still within the dynamic range of the setting. An example would be a current setting with a minimum value of 0A, a maximum value of 8A and a step size of 2.5 mA. Although the dynamic range (8A-0A/2.5 mA = 3200) is small enough for numeric values (<10000), the resolution of the step size is too small to be resolved with values above 1A, i.e. 1.000A + 2.5 mA = 1.002A, which is rounded down.

The setting is treated as a standard integer value, except that all values are multiplied by the multiplier value before being displayed. This prevents consecutive rounding errors from accumulating. Although the display will be rounded down, the actual value used will still be correct.

Packet#	DTL	Use
1	46h	Menu cell
2	2Ah	Menu cell value
3	2Ah	Minimum value
4	2Ah	Maximum value
5	2Ah	Step/increment value
6	2Ch	Multiplier value

5.2.13 GROUP 30H - RESERVED

5.2.14 GROUP 40H - REPEATED DATA PACKET

A repeated data packet is used in blocked transactions where a large number of data packets of the same data type and fixed length are to be transferred, a typical example being the extraction of disturbance records.

The first data packet in the group has a normal DTL and Data field. Subsequent data packets have only the Data field present and are assumed to have the same DTL as the first data packet. This is a departure from the normal data packet format in order to increase communication efficiency.

Packet#	DTL	Use
1	xx	first data packet
2	<none>	2nd data field
..	<none>	subsequent data fields
N	<none>	Nth data field

6. Courier Command Reference

This section details all the Courier commands, their meanings, purpose, expected replies and examples for each one. The commands are listed in alphabetical order and are presented below grouped by application.

Protocol Layer Commands
Poll Buffer
Poll Status
Reset Remote Link

Menu Browsing
Get Column Headings
Get Column Text
Get Column Values
Get Display
Get Strings
Get Text
Get Value
Get Column Setting Limits*

Setting Changes
Enter Setting Mode
Preload Setting
Abort Setting
Execute Setting
Reset Menu Cell†
Reset Trip Indication†*
Set Value*

Low Level Commands
Send Block
Send Event*
Accept Event*
Store Block Identifier*
Store Block Footer*

Control Commands
Load Shed By Group†*
Load Shed To Level†*
Select Setting Group†*
Change Device Address†

Configuration Commands
Enter Configuration Mode*
Exit Configuration Mode*
Enter Calibration Mode*
Exit Calibration Mode*
Set Real Time†*

*All commands should be implemented in a slave device, except those marked with an asterisk which are optional and dependent upon a slave device's capabilities.

† These commands can nominally be sent globally

6.1 KEY TO COMMAND REFERENCE

This chapter contains an alphabetical list of Courier commands. The documentation for each command contains the following information:

Description A brief description to explain what the command does.

Code Sequence The sequence of bytes to make up the Courier data packets to perform the command, expressed in hexadecimal.

Arguments A list of the arguments required for the command, whether implicit or explicit.

Reply The normal response from the slave device.

Error Returns Any reply code could potentially be returned as the result to any command that should return a response. The error returns listed are the minimum that are to be expected along with a more precise description of their meaning when related to that specific command.

Use More detailed information on how the command is used, and in conjunction with which other commands.

See Also Any relevant or similar commands which are referenced or are useful.

Notes Any additional information, side effects or tips for using the command.

Example A typical example of the use of the command. The examples assume a K-Bus transmission medium, rather than an IEC870 medium and include all fields of the message from the address field to the end of the user data field inclusive. All bytes are expressed in hexadecimal and each Courier data packet is surrounded by square brackets to delimit it and aid in identification. Each message begins with the slave's address and the zero byte address terminator [05 00] the message length [XX] and the IEC870 control byte [61 XX]. In slave replies, this is followed by a four byte millisecond timer count value [38 04 t1 t2 t3 t4] and the Courier status byte [5D XX]. The position of the Length byte and the IEC870 control byte should be considered as this will differ for IEC870 transmissions. In all examples, the part of the message corresponding to the user data field is underlined for clarity.

Every effort has been made to ensure the correctness of the information in this chapter, but note that some errors may have been introduced into the examples in their transcription. The actual examples have been modified in places (or in others, completely fabricated) for presentation and the same results may not be achievable from actual relays.

6.2 INDEX OF COMMANDS

6.3 COMMAND LIST BY VALUES

Command	Description
10h	Poll Buffer
11h	Poll Status
12h	Get Text
13h	Get Display
14h	Get Value
15h	Enter Setting Mode
16h	Get Column Headings
17h	Get Column Text
18h	Get Column Values
19h	Get Strings
1Ah	Reset Menu Cell
1Bh	Reset Trip Indication
1Ch	Set Value
1Dh	Get Column Setting Limits
21h	Send Block
23h	Send Event
24h	Accept Event
25h	Store Block Identifier
26h	Store Block Footer
30h	reserved
40h	Preload Setting
41h	Select Setting Group
43h	Change Device Address
44h	Load Shed By Group
45h	Set Real Time
46h	Load Shed To Level
4Ah	Abort Setting
4Eh	Execute Setting
50h	Enter Configuration Mode
51h	Exit Configuration Mode
60h	Enter Calibration Mode
61h	Exit Calibration Mode

Table 6-1 Command List by Value

ABORT SETTING

Description Requests the slave device to abandon a setting change operation.

Code Sequence 05 4A

Arguments None.

Reply 0x00 ERR_OK OK.

Error Returns 0x09 ERR_INVALIDCMD No setting operation in progress.

Use A slave device enters setting mode when it returns a setting's limits in response to the Enter Setting Mode. Once in this mode, all other User interfaces are prevented from making any setting changes since it is a mutually exclusive operation. The setting operation must therefore be completed by sending a new value to the slave device and executing the new value, or by abandoning the setting operation altogether which can be done using this command.

Abort Setting can also be used to terminate a block transfer to a slave device initiated by the Set Value command.

See Also Execute Setting, Enter Setting Mode, Set Value, Store Block Identifier.

Notes The slave device will abandon setting mode itself when its internal setting mode timer expires (after a minimum of 2 minutes). To prevent this time-out, the Enter Setting Mode command can be sent to the same menu cell reference at regular intervals to reset this timer.

Example
Enter Setting Mode for 0909
Tx: [05 00] [06] [61 7B] [07 15 09 09]
Limits group: type 21, length 0C, Value=2, Min = 0, Max 7, Step=1
Rx: [05 00] [19] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0A 21 0C] [2A 02 00] [2A 00 00] [2A 07 00] [2A 01 00]
Preload value at 0909 to 01(signed integer)
Tx: [05 00] [09] [61 7B] [07 40 09 09] [2A 01 00]
Slave device echoes new setting request
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [2A 01 00]
Abort Setting
Tx: [05 00] [04] [61 7B] [05 4A]
Reply code 0 - Ok
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

ACCEPT EVENT*

Description Requests the slave device to discard the last event that was extracted as it has been accepted as valid by the master control unit.

Code Sequence 05 24

Arguments None

Reply 0x00 ERR_OK OK.

Error Returns None

Use This command is issued when the master control unit has accepted the previous event record sent by the slave device in response to the Send Event Command. The slave device should then be prepared to send the next event when requested. If there are no more events, the EVENT bit in the status byte should be reset.

See Also Send Event.

Notes Under a communication retry situation this command may be issued several times. Only the first command should have any effect. An intervening Send Event command should be received before this command has affect again, although it should still return the ERR_OK reply code.

This command need not be implemented if a slave device does not generate event records. The EVENT bit of the status word should then never be set.

This command is normally implemented by the Courier Protocol layers.

Some devices may allow retrieval of discarded events. See section 7.3 Event Record Restoration

Example

Send Event
Tx: [05 00] [04] [61 7B] [05 23]
Group: type Event, Length 2A, menu cell 0021, time t8t7t6t5, Text 'LOG. Relay Stat %08.08b', Binary 0
Rx: [05 00] [39] [61 08] [38 04 t1 t2 t3 t4] [5D 40] [0A 00 2A] [46 21 00] [38 04 t5 t6 t7 t8] [18 1F 4C 4F 47 2E 20 52 65 6C 61 79 20 53 74 61 74 19 20 20 20 20 20 20 20 20 25 30 38 2E 38 62 1D] [21 00]
Accept Event
Tx: [05 00] [04] [61 7B] [05 24]
Reply 0 - ok
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

CHANGE DEVICE ADDRESS †

Description Sets the address of the fully qualified slave device to a new address, which may be zero ready for auto-address detection.

Code Sequence 05 43 18 07 <6 digit, 1 alpha serial number> 25 <old address> 25 <new address>

Arguments Explicit 7 character serial number.
Explicit 1 byte unsigned integer indicating old address of slave device.
Explicit 1 byte unsigned integer indicating new address of slave device.

Reply None. This command shall only be sent as a global transaction as complications in determining the source address of the response occur.

Error Returns None

Use A slave device is normally supplied with an address of 255. With this address it will never respond to any communication message; its address must be changed before it can be used (done as part of the commissioning procedure); usually performed via a front panel user interface. This command achieves the same purpose via the communications. The address of the slave device with the matching old address and the matching serial number is changed to the new address. If this is zero, the slave device can then be auto-detected by the master control unit and then automatically changed to another available address.

See Also None.

Notes Since address 255 is reserved for global or broadcast messages, it needs to be further qualified by the relay's serial number to target the command at only one slave device. The serial number may not be unique enough if other manufacturers relays are also included in Courier.

The relay will accept the command and change it's address to the new address if:

- the serial number matches it's own.
- the old address matches it's own, or the old address is 255.

Global commands must be received twice in succession to be accepted.

Example *Change slave device with address 03 and serial number '000001A' to a new address of 01*
Tx: [FF 00] [11] [61 44] [05 43] [18 07 30 30 30 30 30 31 41] [25 03] [25 01]

Change slave device with address 03 and serial number '000001A' to a new address of 01
Tx: [FF 00] [11] [61 44] [05 43] [18 07 30 30 30 30 30 31 41] [25 03] [25 01]

ENTER CALIBRATION MODE*

Description This command puts the slave device into calibration mode so that it may be calibrated.

Code Sequence 05 60 1C 06 <6 character password>

Arguments Explicit 6 character password, each character being in the range A..Z.

Reply 0x00 ERR_OK OK.

Error Returns

0x04	ERR_NOVERIFY	Password is incorrect.
0x09	ERR_INVALIDCMD	Calibration mode is already active.

Use In calibration mode, the protection routines will stop, putting the slave device out of service. An extra area of the menu's database becomes available via the communication interface to facilitate the calibration procedure.

See Also Exit Calibration Mode

Notes Whilst in calibration mode, all writing to E2PROM is disabled. Therefore no setting changes are valid until the Exit Calibration Mode command is issued. There is no Courier mechanism for exiting calibration mode without making the calibration settings permanent, save for resetting the slave device manually

This command need not be implemented if a slave device does not contain protected calibration data.

Example *Enter calibration Mode*
Tx: [05 00] [0C] [61 7B] [05 60] [1C 06 xx xx xx xx xx xx]
Reply 0 - ok
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

ENTER CONFIGURATION MODE*

Description This command puts the slave device into configuration mode so that it may be configured.

Code Sequence 05 50 1C 06 <6 character password>

Arguments Explicit 6 character password, each character being in the range A..Z.

Reply 0x00 ERR_OK OK.

Error Returns

0x04	ERR_NOVERIFY	Password is incorrect.
0x09	ERR_INVALIDCMD	Configuration mode is already active.

Use In configuration mode, the protection routines will stop, putting the slave device out of service. An extra area of the menu's database becomes available via the communication interface to facilitate the configuration procedure.

See Also Exit Configuration Mode

Notes Whilst in configuration mode, all writing to E2PROM is disabled. Therefore no setting changes are valid until the Exit Configuration Mode command is issued. There is no Courier mechanism for exiting configuration mode without making the configuration settings permanent, save for resetting the slave device manually

This command need not be implemented if a slave device does not contain protected configuration data.

Example *Enter Configuration Mode*
Tx: [05 00] [0C] [61 7B] [05 50] [1C 06 xx xx xx xx xx xx]
Reply 0 - ok
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

ENTER SETTING MODE

Description Attempts to put the specified menu cell into setting mode and returns the limits and current data value if successful so that the master control unit may change the setting.

Code Sequence 07 15 <row> <col>

Arguments Implicit menu cell reference

Reply The reply will be a grouped transaction. The group type will be either of:

1. Setting limits group type 21h
e.g. 0A 21 <group length> <value DTL> <value> <minimum DTL> <minimum> <maximum DTL> <maximum> <step DTL> <step size>

2. Multiplier setting limits group type 22h
e.g. 0A 21 <group length> <value DTL> <value> <minimum DTL> <minimum> <maximum DTL> <maximum> <step DTL> <step size> <multiplier DTL> <multiplier>

Error Returns

0x01 ERR_NOCODE	Given menu location does not exist.
0x02 ERR_NODATA	Menu cell has no data.
0x03 ERR_NOACCESS	Menu cell is not accessible for setting mode e.g. another cell is already in setting mode.
0x05 ERR_NOSETTINGS	This is not a settable cell.
0x06 ERR_NOPASSWORD	Password is required to change setting.
0x07 ERR_LOCAL	Local operator is changing a setting.

Use This command is used to change settings remotely in a slave device. The action of requesting the limits will also set a semaphore flag in the slave device preventing any setting changes being made at the front panel interface. It is therefore important that once a setting sequence operation has been initiated using this command, the sequence is completed by subsequently either sending commands to abort the setting change sequence or to download and store the new setting value. If neither of these actions are taken the slave device will time-out after a minimum of two minutes, thus allowing the front panel interface to again make setting changes. To prevent the slave device from timing out and thus prolonging the setting period, the same Enter Setting Mode command can be issued to the same menu cell location which will reset the time-out period. Issuing the command to another menu cell will result in an error, but will not abort the original setting operation.

See Also Preload Setting, Abort Setting, Execute Setting, Password Cell Timeout (section 4.3 Predefined Menu Cell References)

Notes Only one linear setting range is catered for.

Group type 22h is used for Courier values where the multiplier value is used to scale the value and limits for display purposes to prevent rounding errors from occurring.

Example

Enter Setting Mode for 0909
Tx: [05 00] [06] [61 7B] [07 15 09 09]
Limits group: type 21, length 0C, Value=2, Min = 0, Max 7, Step=1
Rx: [05 00] [19] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0A 21 0C] [2A 02 00] [2A 00 00] [2A 07 00] [2A 01 00]
Preload value at 0909 to 01(signed integer)
Tx: [05 00] [09] [61 7B] [07 40 09 09] [2A 01 00]
Slave device echoes new setting request
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [2A 01 00]
Execute Setting
Tx: [05 00] [04] [61 7B] [05 4E]
Reply code 0 - Ok
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

EXECUTE SETTING

Description Requests the slave device to accept the last setting sent using the Preload Setting command and put it into operation.

Code Sequence 05 4E

Arguments None

Reply

0x00	ERR_OK	OK.
0x08	ERR_OKCHANGE	OK. Menu re-read necessary.

Error Returns

0x03	ERR_NOACCESS	Not in setting mode.
0x04	ERR_NOVERIFY	Verify error on setting change.
0x06	ERR_NOPASSWORD	Password is required to change setting
0x09	ERR_INVALIDCMD	Not in setting mode or setting value not preloaded

Use Used to complete a setting change operation initiated by a Enter Setting Mode command. The new setting must have been previously downloaded using the Preload Setting command. The new setting is not implemented until this command has been received. The new setting value must comply with the cell's limits, otherwise a verify error will result and the setting operation will be abandoned.

See Also Abort Setting, Enter Setting Mode, Preload Setting.

Notes The Execute Setting command must be sent within the slave device's setting time-out period since the last Enter Setting Mode command was sent.

Example

Enter Setting Mode for 0909
Tx: [05 00] [06] [61 7B] [07 15 09 09]
Limits group: type 21, length 0C, Value=2, Min = 0, Max 7, Step=1
Rx: [05 00] [19] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0A 21 0C] [2A 02 00] [2A 00 00] [2A 07 00] [2A 01 00]
Preload value at 0909 to 01(signed integer)
Tx: [05 00] [09] [61 7B] [07 40 09 09] [2A 01 00]
Slave device echoes new setting request
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [2A 01 00]
Execute Setting
Tx: [05 00] [04] [61 7B] [07 4E]
Reply code 0 - OK
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

EXIT CALIBRATION MODE*

Description This command takes the slave device out of calibration mode.

Code Sequence 05 61 1C 06 <6 character password> 18 10 <16 character date string>

Arguments Explicit 6 character password, each character being in the range A..Z & a 16 character date string in the form "DD/MM/YYHH:MM:SS"

Reply None since the slave device resets.

Error Returns

0x04	ERR_NOVERIFY	Password is incorrect.
0x09	ERR_INVALIDCMD	Calibration mode is not active.

Use This command is Used after a slave device has been calibrated to store all calibration settings in E2PROM, exit the calibration mode and reset the slave device, thus enabling the protection routines again.

See Also Enter Calibration Mode

Notes There is no Courier mechanism for exiting calibration mode without making the calibration settings permanent, save for resetting the slave device manually.

This command need not be implemented if a slave device does not contain protected calibration data.

Example *Exit Calibration Mode*
Tx: [05 00] [1E] [61 7B] [05 61] [1C 06 xx xx xx xx xx xx] [18 10 xx xx xx xx xx xx xx xx xx xx xx xx xx xx]

EXIT CONFIGURATION MODE*

Description This command takes the slave device out of configuration mode.

Code Sequence 05 51 1C 06 <6 character password> 18 10 <16 character date string>

Arguments Explicit 6 character password, each character being in the range A..Z & a 16 character date string in the form "DD/MM/YYHH:MM:SS".

Reply None since the slave device resets.

Error Returns

0x04	ERR_NOVERIFY	Password is incorrect.
0x09	ERR_INVALIDCMD	Configuration mode is not active.

Use This command is used after a slave device has been configured to store all configuration settings in E2PROM, exit the configuration mode and reset the slave device, thus enabling the protection routines again.

See Also Enter Configuration Mode.

Notes There is no Courier mechanism for exiting configuration mode without making the configuration settings permanent, save for resetting the slave device manually.

This command need not be implemented if a slave device does not contain protected configuration data.

Example *Exit Configuration Mode*
Tx: [05 00] [1E] [61 7B] [05 51] [1C 06 xx xx xx xx xx xx] [18 10 xx xx xx xx xx xx xx xx xx xx xx xx xx xx]

GET COLUMN HEADINGS

Description Requests the slave device to send a list of all the column heading cells in its menu along with their text.

Code Sequence 05 16

Arguments None

Reply A blocked transaction.
Each block consists of one or more grouped data packets of type 11h - the column heading group, each consisting of two data packets. The first data packet is the menu cell reference and the second data packet is the text for that cell.
0A 11 <group length> 46 00 <col> 18 <length> <column heading text> ....

Error Returns 0x02 ERR_NODATA There are no column headings.

Use When a user wishes to interrogate or browse a slave device manually, this command is sent to extract a list of the column headings available in the slave device so that the user may select an individual column of information. Once a column has been selected, the Get Column Text and Get Column Values commands are used to extract the information for the specified column. The column headings are therefore used as a type of index to the slave device's features and settings.

See Also Get Column Text, Get Column Values, Send Block.

Notes The number of column headings may vary depending on the current state of the relay's database and whether various passwords or other settings have been entered. They should not be assumed to remain constant and should be re-read after any setting change which results in a reply code of 8 being returned.

Column numbers will monotonically increase but may not necessarily be consecutive in the replies.

Example *Get Column Headings*
Tx: [05 00] [04] [61 7B] [05 16]
Response is a block header: transaction has 0 blocks
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0D 00]
Send block 0
Tx: [05 00] [05] [61 7B] [06 21 00]
First block is sent: block 0, Group = 11-column headings, length 19; menu cell 0000, text 'SYSTEM DATA'.
Rx: [05 00] [22] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [15 00] [0A 11 13] [46 00 00] [18 0E 53 59 53 54 45 4D 20 44 41 54 41 1D 19 1D]
Send block 1

Tx: [05 00] [05] [61 7B] [06 21 01]
block footer 1 block sent.
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [12 01 00]

GET COLUMN SETTING LIMITS*

Description Requests the slave device send a grouped packet of limits for all cells in the specified column

Code Sequence 07 1D 00 <col>

Arguments implicit menu cell reference

Reply A blocked transaction.
Each block consists of one or more grouped data packets of type 23h or 24h, depending on the cell type.

Error Returns 0x02 ERR_NODATAThere are no setting cells in this column

Use Used in conjunction with the Get Column Headings and Get Column Text commands to extract a setting file from the slave device.

See Also Get Column Headings, Get Column Text, Get Column Values, Send Block.

Notes The number of menu cells in a column may vary depending on the current state of the slave device's database and whether various passwords or other settings have been entered. They should not be assumed to remain constant and should be re-read after any setting change which results in a reply code 8 (ERR_OKCHANGE) being returned.

The menu cell references will all have the same column number and their row numbers will monotonically increase, but they may not necessarily be consecutive. The number of groups returned may not equal the number of groups returned from the Get Column Text command for the same column since not all cells may be settable, but it will never exceed it.

Example *Get Column Setting Limts for column 02*
Tx: [05 00] [06] [61 7B] [07 1D 00 02]
Response is a block header: transaction has 0 blocks
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0D 00]
Send block 0
Tx: [05 00] [05] [61 7B] [06 21 00]
First block is sent: block 0,Grp=23-column limits - len 15, menu cell 0201, Value=2, Min = 0, Max 7, Step=1
Rx: [05 00] [1E] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [15 00] [0A 23 0F] [46 01 02]] [2A 02 00] [2A 00 00] [2A 07 00] [2A 01 00]
Send block 1
Tx: [05 00] [05] [61 7B] [06 21 01]
block footer 1 block sent.
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [12 01 00]

GET COLUMN TEXT

Description Requests the slave device send a list of the text for all the cells in the specified column.

Code Sequence 07 17 00 <col>

Arguments Implicit menu cell reference

Reply A blocked transaction.
Each block consists of one or more grouped data packets of type 12h - the column text group, each consisting of two data packets. The first data packet is the menu cell reference and the second data packet is the text for that cell.
0A 12 <group length> 46 <row> <col> 18 <length> <menu cell text> ....

Error Returns 0x02 ERR_NODATA There are no cells in this column.

Use Used in conjunction with the Get Column Headings and Get Column Values commands to allow a user to interrogate or browse the database in a slave device. After this and the Get Column Values commands have been issued, the two lists of menu cell references have to be correlated to produce the complete display of the selected column of information.

See Also Get Column Headings, Get Column Values, Send Block

Notes The number of menu cells in a column may vary depending on the current state of the relay's database and whether various passwords or other settings have been entered. They should not be assumed to remain constant and should be re-read after any setting change which results in a reply code of 8 being returned.

The menu cell references will all have the same column number, and their row numbers will monotonically increase, but they may not necessarily be consecutive in the replies.

Example *Get Column Text*
Tx: [05 00] [06] [61 7B] [07 17 00 00]
Response is a block header: transaction has 0 blocks
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0D 00]
Send block 0
Tx: [05 00] [05] [61 7B] [06 21 00]
First block is sent: block 0, Group = 12-column text, length 19; menu cell 0000, text 'SYSTEM DATA'.
Rx: [05 00] [22] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [15 00] [0A 12 13] [46 00 00] [18 0E 53 59 53 54 45 4D 20 44 41 54 41 1D 19 1D]
Send block 1

Tx: [05 00] [05] [61 7B] [06 21 01]
block footer 1 block sent.
Rx: [05 00] [0B] [61 08] [3B t1 t2 t3] [5D 00] [12 01 00]

GET COLUMN VALUES

Description Requests the slave device send a list of the values for all the cells in the specified column.

Code Sequence 07 18 00 <col>

Arguments Implicit menu cell reference

Reply A blocked transaction.
Each block consists of one or more grouped data packets of type 13h - the column values group, each consisting of two data packets. The first data packet is the menu cell reference and the second data packet is the data value for that cell.
0A 13 <group length> 46 <row> <col> <Data value DTL> <Data value> ....

Error Returns 0x02 ERR_NODATA There are no cells with data values.

Use Used in conjunction with the Get Column Headings and Get Column Text commands to allow a user to interrogate or browse the database in a slave device. After this and the Get Column Text commands have been issued, the two lists of menu cell references have to be correlated to produce the complete display of the selected column of information.

See Also Get Column Headings, Get Column Text, Send Block.

Notes The number of menu cells in a column may vary depending on the current state of the relay's database and whether various passwords or other settings have been entered. They should not be assumed to remain constant and should be re-read after any setting change which results in a reply code of 8 being returned.

The menu cell references will all have the same column number, and their row numbers will monotonically increase, but they may not necessarily be consecutive in the replies. The number of groups returned may not equal the number returned by the Get Column Text command for the same column, since not all menu cells have values, but it will never exceed it.

(see section 5.1.21 re:Block Transfer Cell (58h))

Example *Get Column Values*
Tx: [05 00] [06] [61 7B] [07 18 00 02]
Response is a block header: transaction has 0 blocks
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0D 00]
Send block 0
Tx: [05 00] [05] [61 7B] [06 21 00]

First block is sent: block 0, Grp=13-column values, len=7; menu cell=0201, Value=0A, Grp=13-column values, len= 7; menu cell=020C, Value=50Hz,
Rx: [05 00] [20] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [15 00] [0A 13 07] [46 01 02] [2F 00 00 06] [0A 13 07] [46 0C 02] [2F 88 13 B4]
Send block 1
Tx: [05 00] [05] [61 7B] [06 21 01]
block footer 1 block sent.
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [12 01 00]

GET DISPLAY*

Description Requests the display for a particular menu cell. Any formatting characters are replaced by the menu cell's actual data value.

Code Sequence 07 13 <row> <col>

Arguments Implicit menu cell reference

Reply 18 LL <sequence of ASCII text characters of length LL>

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x02	ERR_NODATA	Menu cell has no data.
0x03	ERR_NOACCESS	Menu cell is not accessible.

Use This command requests the slave device to internally combine the text and data value for the specified menu cell and return the resultant text string. It is not normally used, but may be useful for simple master control units which cannot combine text formatting controls and data values.

See Also Get Text, Get Menu Cell Data, LDU Get Display.

Notes The LDU Get Display command performs the same function. The difference is that in this command the menu cell is specified, whereas the LDU command relies on the current position of the menu navigation system, as determined by the previous key presses that have been sent.

This command need not be implemented unless the slave device is to be used with a simple master control unit which cannot form the display itself from the replies to the Get Text and Get Value commands. However, its implementation is recommended for compatibility with future master control units.

Example *Get menu display from cell 0201*
Tx: [05 00] [06] [61 7B] [07 13 01 02]
Text ="MES1 Ia↔↓ 0A↔", Length=11h
Rx: [05 00] [1B] [61 08] [38 04 t1 t2 t3 t4] [18 11 4D 45 53 31 20 49 61 1D 19 20 20 20 20 20 30 41 1D]

GET STRINGS

Description Requests the slave device to send a list of strings for a menu cell's data value which are used to make the setting more comprehensible.

Code Sequence 07 19 <row> <col>

Arguments Implicit menu cell reference

Reply A grouped transaction or a blocked transaction.
If all indexed strings can fit within one message length, a grouped transaction will result. This consists of a grouped data packet of type 20h. The group will contain several data packets of type TEXT 18h, the actual number must be determined by the group length.
In a blocked transaction, each block transfer contains a grouped data packet as above.
0A 20 <group length> <18> <L1> <string1> <18> <L2> <string2> ...

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x02	ERR_NODATA	Menu cell has no data.

Use This command is used specifically when changing the setting of a binary flag setting or an indexed string setting. A binary flag setting is a bit mask where each bit is used for a specific purpose. The indexed strings give a textual description for each bit in the bit mask starting at bit 0.

An indexed string setting is an integer setting starting at 0 increasing in steps of 1. For each value in the range the appropriate indexed string is displayed rather than the numeric value.

See Also Enter Setting Mode.

Notes A menu cell of type indexed string will return a data type of text (18h) when its value is obtained using the Get Value command. Since all values polled using the Get Value command that do not result in blocked transactions should be the same length and type each time the command is issued, this implies that the indexed strings for a particular menu cell should all be the same length.
When requesting the menu cell's limits in order to change the setting, its data type will change to Indexed string, causing the master control unit to issue this command to identify the range and values of its possible settings.

This command is not required if a slave device has no binary flag settings or indexed string settings.

Example *Get strings from 0403*
Tx: [05 00] [06] [61 7B] [07 19 03 04]

RXD: [Group-indexed strings, length 50h] [TXT:'STI30XDT'] [TXT:'LTI30XDT'] [TXT:'DT '] [TXT:'SI30XDT '] [TXT:'I30XDT '] [TXT:'VI30XDT '] [TXT:'EI20XDT '] [TXT:'EI10XDT ']
Rx: [05 00] [5F] [61 08] [38 04 t1 t2 t3 t4] [5D 80] [0A 20 50] [18 08 53 54 49 33 30 58 44 54] [18 08 4C 54 49 33 30 58 44 54] [18 08 44 54 20 20 20 20 20 20] [18 08 53 49 33 30 58 44 54 20] [18 08 49 33 30 58 44 54 20 20] [18 08 56 49 33 30 58 44 54 20] [18 08 45 49 32 30 58 44 54 20] [18 08 45 49 31 30 58 44 54 20]

GET TEXT

Description Requests the text for a particular menu cell. This includes any formatting characters to position and format any data value associated with the menu cell.

Code Sequence 07 12 <row> <col>

Arguments Implicit menu cell reference

Reply 18 LL <sequence of ASCII text characters of length LL>

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x02	ERR_NODATA	Menu cell has no data.
0x03	ERR_NOACCESS	Menu cell is not accessible.

Use This text describes the purpose of the menu cell and can be used in a 'C' style PRINTF() statement along with the menu cell's data value to form the display for the menu cell. In a polling environment, the text need only be requested once, leaving just the data value to be polled repeatedly.

See Also Get Display.

Notes The formatting characters are generally the same as used in the standard 'C' PRINTF() family of functions and are documented later (see chapter 14).

Example *Get menu text from cell 0201*
Tx: [05 00] [06] [61 7B] [07 12 01 02]
Text ="MES1 Ia↔↓%k↔", Length=12
Rx: [05 00] [18] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [18 0C 4D 45 53 31 20 49 61 1D 19 25 6B 1D]

GET VALUE

Description Requests the slave device to send the current value of the data value for the specified menu cell.

Code Sequence 07 14 <row> <col>

Arguments Implicit menu cell reference

Reply The Reply to this request may be:

1. a single data packet
2. a grouped data packet
3. a blocked transfer

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x02	ERR_NODATA	Menu cell has no data.
0x03	ERR_NOACCESS	Menu cell is not accessible.

Use A very commonly used command. Used to poll for data values on a regular basis for updating mimic diagrams or for load profiling etc. The data type of the value is returned in the Reply and should not be assumed. This allows any menu cell to return data of any possible data type.

See Also Get Strings.

Notes Data types of binary flag and indexed string may need an additional Get Strings command to be issued when the value is to be changed to obtain the associated textual information for the data value.

Example *Get menu value from cell 0201*
Tx: [05 00] [06] [61 7B] [07 14 01 02]
Unsigned value = 03E8h
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [26 E8 03]

LOAD SHED BY GROUP †*

Description A global command requesting all slave devices with a load shedding group number equal to or less than the group specified to trip their associated circuit breaker, thus reducing the system load.

Code Sequence 06 44 <group 0-7>

Arguments Implicit group number

Reply None if this is a global command, otherwise:
0x00 ERR_OK OK.

Error Returns None if this is a global command, otherwise:
0x04 ERR_NOVERIFY group load shedding is disabled.

Use This command implements load shedding by tripping non-essential loads. Level 1 loads are tripped first; level 7 loads have the highest security of supply.

Load restoration is achieved by shedding to a lower level number than a particular slave device's programmed load shed level. The slave device will only restore the load IF it was tripped by a previous load shed command and it has not lost auxiliary power in the meantime. Load restoration may have an associated time delay in each slave device to stagger the load on restoration to prevent large sudden power surges.

See Also Load Shed to Level.

Notes Global commands must be received twice in succession to be accepted.

This command need not be implemented unless the slave device is capable of performing load shedding.

Example *Load shed group 01*
Tx: [FF 00] [05] [61 44] [06 44 01]
Load shed group 01
Tx: [FF 00] [05] [61 44] [06 44 01]

LOAD SHED TO LEVEL †*

Description A global command requesting a voltage reduction on the system to reduce load.

Code Sequence 06 46 <level 0-3>

Arguments Implicit load shed level.

Reply

0x00 ERR_OK OK.

Error Returns none

Use This command implements load shedding by reducing the system voltage, normally by altering taps on transformers to reduce the voltage in steps of 3%, 6% or 9%. Three separate output contacts are normally provided on the slave device for this function if required. One contact only will be closed for each level of load shed.

See Also Load shed by group.

Notes Normally, only voltage regulating relays would respond to this command by altering the taps on a tap-changing transformer.

Global commands must be received twice in succession to be accepted.

This command need not be implemented unless the slave device is capable of performing load shedding.

Example

Load shed to level 2
Tx: [FF 00] [05] [61 44] [06 46 02]
Load shed to level 2
Tx: [FF 00] [05] [61 44] [06 46 02]

POLL BUFFER

Description Returns the slave device's reply to the previous request.

Code Sequence 05 10

Arguments None

Reply The slave device's previous reply, or if still not available, an empty user data field and the busy bit set in the status byte.

Error Returns none

Use If a slave device cannot reply to a request within its allotted time-out period, it will respond with an empty user data field and the busy bit set in the status byte: a BUSY response, thus freeing the communication bus for the master control unit to communicate with other slave devices. The master control unit should then send this command some time later to determine if the slave device has formed its previous response yet. This will be repeated until the slave device responds with its busy bit reset, in which case the user data field will contain the reply to the previous request.

See Also Poll Status, Reset Remote Link.

Notes The use of the busy bit allows the master control unit to communicate with other relays whilst the first relay is still forming its response to a reply.

This command is only valid as a simple transaction and cannot be used with any additional command.

This command is normally implemented by the Courier Protocol layers.

Example
Get menu value from cell 0201
Tx: [05 00] [06] [61 7B] [07 14 01 02]
Busy response
Rx: [05 00] [0A] [61 08] [38 04 t1 t2 t3 t4] [5D 08]
Poll buffer
Tx: [05 00] [04] [61 7B] [05 10]
Proper reply, Unsigned value = 03E8h
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [26 E8 03]

POLL STATUS

Description Returns the current status of the relay.

Code Sequence 05 11

Arguments None

Reply The reply's user data field is empty. The relay's status is returned as normal in the reply header as:
5D XX
where XX is the 8-bit status byte.

Error Returns none

Use This command is used to poll a slave device on a regular basis if no other information is required from the relay. The slave device will respond with its timer count value, IEC870 control byte and the status byte as in all other replies. The user data field, however, will be empty. From the information in the status byte, the master control unit can determine that the relay is still active, whether it has any events or disturbance records to extract and if any plant or control information has changed state.

See Also Poll Buffer, Reset Remote Link.

Notes This command is used to test if a slave device is communicating once it has been detected.

If the slave device stops communicating, communication must be re-established using the Reset Remote Link IEC870 command. Once the link has been re-established, polling using this command can be resumed.

This request should not ordinarily result in a busy response unless the device is already busy as a result of a previous operation. Doing so compromises the effectiveness of the communication system as a whole.

This command can be used in place of the Poll Buffer command to determine when a slave device is no longer busy, reverting to the Poll Buffer command to extract the data when it is finally available.

This command is only valid as a simple transaction and cannot be used with any additional command.
This command is normally implemented by the Courier Protocol layers.

Example *Poll relay 5 for status*
Tx: [05 00] [04] [61 7B] [05 11]
Reply status byte=0

Rx: [05 00] [0A] [61 08] [38 04 t1 t2 t3 t4] [5D 00]

PRELOAD SETTING

Description Sends a new setting to the slave device's menu cell in preparation for it being actioned.

Code Sequence 07 40 <row> <col> <setting data packet>

Arguments Implicit menu cell reference.
Explicit data packet containing the new setting.

Reply The slave device will echo the setting data packet.
<setting data packet>

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x03	ERR_NOACCESS	Not in setting mode
0x04	ERR_NOVERIFY	Verify error on setting change.
0x05	ERR_NOSETTINGS	This is not a settable cell.
0x06	ERR_NOPASSWORD	Password is required to change setting
0x09	ERR_INVALIDCMD	Setting mode has not been entered or has timed out, or menu cell does not match the current menu cell in setting mode. (This error will usually be returned in preference to any of the above)

Use The slave device must first be put into setting mode by issuing the Enter Setting Mode command. The master control unit will then change the setting according to the given limits and send it to the slave device using this command. The slave device will echo the setting back as a response for verification. If this echoed data packet matches the setting sent, the master control unit will issue the Execute Setting Command. The Abort Setting command can be used at any time to abandon the setting change operation and take the slave device out of setting mode.

See Also Execute Setting, Abort Setting, Enter Setting Mode.

Notes None.

Example
Enter Setting Mode for 0909
Tx: [05 00] [06] [61 7B] [07 15 09 09]
Limits group: type 21, length 0C, Value=2, Min = 0, Max 7, Step=1
Rx: [05 00] [19] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0A 21 0C] [2A 02 00] [2A 00 00] [2A 07 00] [2A 01 00]
Preload value at 0909 to 01(signed integer)
Tx: [05 00] [09] [61 7B] [07 40 09 09] [2A 01 00]
Slave device echoes new setting request
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [2A 01 00]
Execute Setting
Tx: [05 00] [04] [61 7B] [05 4E]
Reply code 0 - Ok

Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

RESET MENU CELL †

Description Requests the slave device to reset the contents of the specified cell. Not to be confused with the Abort Setting command.

Code Sequence 07 1A <row> <col>

Arguments Implicit menu cell reference.

Reply

0x00	ERR_OK	OK.
0x08	ERR_OKCHANGE	OK. Menu re-read necessary.

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x03	ERR_NOACCESS	Cell cannot be reset at this time.
0x05	ERR_NOSETTING	This cell is not resettable
0x06	ERR_NOPASSWORD	Password is required to reset setting
0x07	ERR_LOCAL	Local setting is in progress

Use The purpose of this command is to emulate the action of pressing the RESET key on the front of the slave device whilst a particular menu cell is visible. The result of this action is therefore cell specific. Examples are the resetting of counters and fault records. This command allows a menu cell's value to be changed to a single specific value without entering the setting mode and confirming with the EXECUTE SETTING command.

See Also None.

Notes This command should NOT be confused with the ABORT SETTING command and does not abort a setting change after an ENTER SETTING MODE command. Any menu cell may perform an action or function on receipt of this command. The menu cell does not have to be a setting cell.

Example

Reset Menu Cell 0003
Tx: [05 00] [06] [61 7B] [07 1A 03 00]
Reply code 08- OK, but re-read database
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 08]

RESET REMOTE LINK

Description Resets the communications in a slave device. (This is not a Courier command as it is accomplished using the IEC870 control field. However, it is included here for completeness).

Code Sequence Performed through IEC870 control byte function 0. No data packets.

Arguments None

Reply Response code returned in IEC870 control byte only.
No other data packets are returned.

Error Returns none

Use This command must be sent to IEC870 based slave devices before they will respond. It is therefore used as a means of detecting whether a slave device exists at the specified address. If a response is given the normal Poll Status command is issued.
This command re-synchronises the FCB bit in the IEC870 control byte.

See Also Poll Status.

Notes If a slave device stops communicating, this command is sent to re-initialise its communications and to re-establish communication with it.

This command is normally implemented by the Courier Protocol layers.

Example *Reset Remote Link*
Tx: [05 00] [02] [61 40]
Acknowledge of reset remote link command
Rx: [05 00] [02] [61 00]

RESET TRIP INDICATION*

Description Resets the trip indication on the slave device.

Code Sequence 05 1B

Arguments None

Reply 0x00 ERR_OK OK.

Error Returns None

Use The trip indication is usually a red LED on the slave device which may need to be reset manually. This command provides a common method of resetting this indication across all slave devices.

See Also None.

Notes It is possible to reset the trip indication apparently successfully, but still to find the trip status exists due to the trip condition still existing.

This command need not be implemented.

Example *Reset Trip Indication*
Tx: [05 00] [04] [61 7B] [05 1B]
Reply code 00- OK
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

SELECT SETTING GROUP †*

Description Requests the slave device to use a particular setting group.

Code Sequence 06 41 <setting group>

Arguments Implicit setting group number.

Reply None if this is a global command, otherwise:

0x00	ERR_OK	OK.

Error Returns None if this is a global command, otherwise:

0x02	ERR_NODATA	There are no setting groups.
0x03	ERR_NOACCESS	Setting group cannot be changed.
0x04	ERR_NOVERIFY	If group out of range.

Use Where slave devices contain alternative groups of settings, this command can be used to change the setting group used in all the slave devices simultaneously. This is useful for changing system configuration under abnormal operating conditions.

See Also None.

Notes Global commands must be received twice in succession to be accepted.

This command need not be implemented if the slave device has only one setting group.

The group number transmitted is 1 less than the value selected and 1 less than the value reported from cell 000E. That is:

to select setting group 1, send a value of 0,
to select setting group 2, send a value of 1 etc...

Example *Select setting group 1*
Tx: [FF 00] [05] [61 44] [06 41 00]
Select setting group 1
Tx: [FF 00] [05] [61 44] [06 41 00]

SEND BLOCK

Description Requests the slave device to send the next block of data (or a repeat of the previous block) in a blocked transaction.

Code Sequence 06 21 <sequence number>

Arguments Implicit single byte sequence number.

Reply A blocked transaction reply consisting of a block identifier and one or more grouped data packets.

Error Returns None.

Use A blocked transaction is initiated by the slave device returning a block header in response to a request for the master control unit. This command is used to request each separate block of data of the transaction. The sequence number is an 8-bit index number to identify the block sequence. Block 0 must be requested as the first sequence number. If a block is not received correctly, it can be requested again any number of times. If received correctly, the next sequence number can be requested; any other sequence number will result in the previous block being re-sent. (The master control unit can use this block to resynchronise if necessary). Sequence numbers wrap around after sequence number 255 to 0. A block footer response indicates the end of the blocked transaction. The master control unit should compare the number of blocks received with the number of blocks indicated in the block footer and optionally in the block header.

See Also None.

Notes None.

Example

Get Column Headings
Tx: [05 00] [04] [61 7B] [05 16]
Response is a block header: transaction has 0 blocks
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [0D 00]
Send block 0
Tx: [05 00] [05] [61 7B] [06 21 00]
First block is sent: block 0, Group = 11-column headings, length 19; menu cell 0000, text 'SYSTEM DATA'.
Rx: [05 00] [22] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [15 00] [0A 11 13] [46 00 00] [18 0E 53 59 53 54 45 4D 20 44 41 54 41 1D 19 1D]
Send block 1
Tx: [05 00] [05] [61 7B] [06 21 01]
block footer 1 block sent.
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [12 01 00]

SEND EVENT*

Description Requests the slave device to send the next (i.e. the oldest) event stored in its internal buffer.

Code Sequence 05 23

Arguments None

Reply A grouped data packet of several possible event types. Blocked transactions are not permissible.

Error Returns

0x02	ERR_NODATA	There are no more events, or the next event is not currently accessible.

Use Courier is a 'polled' protocol which prevents slave devices from sending information to the master control unit as soon as something happens (i.e. an event occurs). Therefore, the slave device sets the EVENT flag in its status byte which is sent to the master control unit in every Reply message. On detection of this flag being set, the master control unit will issue the Send Event command immediately to extract the event as a high priority task. The EVENT flag will remain set until all events have been read from the slave device.

See Also Accept Event.

Notes K-Series Relays recognise changes to opto input states, changes to relay output states, trips and local setting changes as events. All events must contain a time-tag, a menu cell reference, a piece of descriptive text and a data value. The different event group types indicate the nature of any additional data packets in the event record group.

Repeated use of this command will result in the same event being transmitted. An Accept Event command must be issued after each successfully received event record to allow the slave device to discard the event.

This command need not be implemented if a slave device does not generate event records. The EVENT bit of the status word should never then be set.

This command is normally implemented by the Courier Protocol layers.

Example *Send Event*
Tx: [05 00] [04] [61 7B] [05 23]
Group: type Event, Length 2A, menu cell 0021, time t8t7t6t5, Text 'LOG. Relay Stat %08.08b', Binary 0

Rx: [05 00] [39] [61 08] [38 04 t1 t2 t3 t4] [5D 40] [0A 00 2C] [46 21 00] [38 04 t5 t6 t7 t8] [18 1F 4C 4F 47 2E 20 52 65 6C 61 79 20 53 74 61 74 19 20 20 20 20 20 20 20 20 25 30 38 2E 38 62 1D] [21 00]
Accept Event
Tx: [05 00] [04] [61 7B] [05 24]
Reply 0 - OK
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

SET REAL TIME †*

Description Sends the current system time to a slave device for the setting of real time system clocks.

Code Sequence 05 45

Arguments Explicit data packet containing the current time and date in the 7-byte IEC870 format.

Reply None if this is a global command, otherwise:
0x00 ERR_OK OK.

Error Returns None if this is a global command, otherwise:
0x03 ERR_NOACCESS Time could not be changed.
0x04 ERR_NOVERIFY Time is invalid.

Use The main use of this command is to set the real time clock in a slave device or protocol converter to the current system time of the master control unit. Accurate synchronisation is not guaranteed due to variable propogation delay times in the transmission to several devices and variable time delays in each slave device before the command is actioned.

When sent as a global transaction, only two messages should be sent rather than three. As a consequence there is more likelihood of this message not getting through, therefore it should be sent more often. The time synchronisation should be based on the time of reception of the first of the two messages.

See Also None

Notes This command need not be implemented in a slave device if it uses the relative millisecond time format for time tagging. However, it may be useful for slave devices which contain their own real time clock.

Example *Set Real Time to xx xx xx xx xx xx xx*
Tx: [05 00] [0D] [61 7B] [05 45] [3C 07 xx xx xx xx xx xx xx]

SET VALUE*

Description Requests a slave device to immediately change a menu cell value to the one supplied.

Code Sequence 07 1C <row> <col> <setting data packet>

Arguments Implicit menu cell reference.
Explicit data packet containing the new data.

Reply

0x00	ERR_OK	OK.
0x08	ERR_OKCHANGE	Same as ERR_OK, but column and column headings should be subsequently re-read.

Error Returns

0x01	ERR_NOCODE	Given menu location does not exist.
0x02	ERR_NODATA	Menu cell has no data.
0x03	ERR_NOACCESS	Cell cannot be accessed at the moment.
0x04	ERR_NOVERIFY	Verify error on setting change.
0x05	ERR_NOSETTINGS	This is not a settable cell.
0x06	ERR_NOPASSWORD	Password is required to change setting.
0x07	ERR_LOCAL	Settings are being changed on another user interface.
0xFF	ERR_GENERAL	Other non-specific error.

Use The Set Value command is used to directly transfer a data packet to a specific menu cell of a slave device and for it to be executed immediately.

See Also Preload Setting, Execute Setting, Abort Setting, Store Block Identifier

Notes This command does not have the overhead associated with the normal setting transfer protocol of requesting the limits and reflexing the data value. As a consequence, it can operate a lot faster, but does not have the same level of security, relying on the frame validation checks (checksums/CRC) for integrity. Not recommended for manual setting changes.

This command can be sent using a simple, global or multiple transaction and can be used to initiate block transfers to a slave device. It cannot be nested within another setting change.

The main application for this command is during setting file downloads and to initiate block transfers to a slave device, where the master assumes the data is valid.

Example

Set Value of 0309 to unsigned integer of value 1.
Tx: [05 00] [09] [61 7B] [07 1C 09 03] [2A 01 00]
Reply 0 - ok
Rx: [05 00] [0C] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [49 00]

STORE BLOCK FOOTER*

Description Identifies the last block of a block transfer to a slave device.

Code Sequence 07 26 <num blocks>

Arguments Implicit number of blocks sent

Reply

0x00	ERR_OK	OK.
0x08	ERR_OKCHANGE	Same as ERR_OK, but column and column headings should be subsequently re-read.

Error Returns

0x04	ERR_NOVERIFY	Verify error on setting change.
0xFF	ERR_GENERAL	Other non-specific error.

Use The Store Block Footer command is used within the Set Value command to transfer a large amount of data to a specific menu cell.

See Also Store Block Identifier, Set Value

Notes The implicit Number of Blocks argument is an unsigned integer (usually two bytes) which indicates the total number blocks transferred. This number should match the number of blocks specified in the block header packet of the original Set Value command (unless this value was zero).

Example

Set Value of cell 0309. Block header of 1 block
Tx: [05 00] [09] [61 7B] [07 1C 09 03] [0E 01 00]
Send block 0
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [06 21 00]
Block ID 0, repeated group of 10 unsigned integers (1,2.....9,10)
Tx: [05 00] [1E] [61 5B] [06 25 00] [0A 40 15] [26 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 00 09 00 0A 00]
Send block 1
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [06 21 01]
Block Footer, 1 block sent
Tx: [05 00] [06] [61 7B] [07 26 01 00]

STORE BLOCK IDENTIFIER*

Description Identifies a sequenced block within a block transfer to a slave device.

Code Sequence 06 25 <id>

Arguments Implicit block identifier.
One or more explicit group packets.

Reply Request for next sequential block. [06 21 nn]

Error Returns

0x04	ERR_NOVERIFY	Verify error on setting change.
0xFF	ERR_GENERAL	Other non-specific error.

Use The Store Block Identifier command is used within the Set Value command to transfer a large amount of data to a specific menu cell.

See Also Store Block Footer, Set Value

Notes The implicit block identifier argument is a single byte unsigned integer which indicates the sequence number of the block. This number starts at 0 and wraps back to 0 after 255. The blocks must be sent in sequence order.

Example

Set Value of cell 0309. Block header of 1 block
Tx: [05 00] [09] [61 7B] [07 1C 09 03] [0E 01 00]
Send block 0
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [06 21 00]
Block ID 0, repeated group of 10 unsigned integers (1,2.....9,10)
Tx: [05 00] [1E] [61 5B] [06 25 00] [0A 40 15] [26 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 00 09 00 0A 00]
Send block 1
Rx: [05 00] [0D] [61 08] [38 04 t1 t2 t3 t4] [5D 00] [06 21 01]
Block Footer, 1 block sent
Tx: [05 00] [06] [61 7B] [07 26 01 00]

7. Event Records

7.1 OVERVIEW

Event records are the mechanism through which Courier slave devices indicate to a master that something has occurred on the device, which the master would not necessarily know about otherwise. They allow the master to be informed of many different types of event without having to poll for each one individually. They also provide a means of storing these events locally in the slave device until the master has time to extract them. Courier provides support for two distinct types of records: events and disturbance waveforms. Event records are discussed in this chapter and disturbance records in the following chapter.

7.2 EVENT RECORD EXTRACTION

The event record mechanism makes use of the status byte, which is present in every message from the slave, to inform the master control unit of new events. Whenever the slave device has at least one event ready to be extracted, it sets the event flag in the status byte as an indication to the master control unit, which can then extract the event records one at a time until the event flag is clear.

The extraction of an event record is considered a high priority task by a master control unit and will be performed as soon as possible after detecting the event flag being set. To facilitate the quick extraction of events, the event record is restricted to a single message containing a grouped transaction. It is not permissible to send event records in a blocked transaction as this would delay the extraction of an event if a blocked transaction was already in progress with that device (blocked transactions cannot be nested). Being a grouped transaction allows the event to be extracted at any time that the slave device is not in a busy state.

Once the event bit is seen as being set, the master control unit issues a Send Event Command. The slave device may respond with a busy message, an error code or an event record. An error code 2 (ERR_NODATA) may occur if there are events waiting to be extracted but the oldest event is not currently accessible. The Send Event command should be re-issued until a valid event record is extracted. The Poll Buffer command is issued if a busy response is returned.

Once the master has received the event record correctly, it acknowledges receipt by issuing the Accept Event command. On receipt of this command, the slave device will remove the event from its pending event queue and respond with a reply code of 0 (ERR_OK).

The Send Event and Accept Event commands operate in a handshake manner to ensure that no events are lost during extraction. Repeated Send Event commands will therefore extract the same event until the slave device is informed that the master control unit has received the last event record successfully through the receipt of the Accept Event command. Repeated Accept Event commands will only advance one event record.

It should be noted that the master control unit has no control over the order of extraction of events - it simply extracts records until there are no more left.

7.3 EVENT RECORD RESTORATION

Event records are designed to be extracted once only from a device. Once the event has been extracted and accepted, it is no longer available. However, some slave device implementations may opt to keep these extracted events and make them available again at a later time. The procedure for restoring these previously extracted events is to send a Reset Menu Cell command to menu cell BF05.

7.4 EVENT RECORD DESCRIPTION

An event record is designed to be a short, concise description of an occurrence in a slave device. Several types of event record exist, but for consistency, every event record must contain the following minimum information:

- menu cell reference - Used to identify the source and categorisation of the event.
- text description - Used as a readable description of the event.
- data value - Most events include a value that has changed in some way.
- time tag of the event - indicating when the event occurred.

The menu cell reference does not necessarily imply that a menu cell exists or is readable at that reference.

7.4.1 TYPE 0: STANDARD EVENT RECORD

Where the above contains sufficient information to describe the occurrence, the event record can be constructed from a standard event record of group type 0 (see section 5.2.1). This group contains 4 data packets to hold the event record information and forms the basis for all other event record types. The format of this group packet is repeated here:

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	Text description (<49 chars)
4	xx	data value

Where the data value is relevent to the event record, packet 3 would normally contain formatting characters to allow the data value in packet 4 to be displayed within the text description using the 'C' language printf() type functions (see chapter 14), but does not have to.

If packet 3 does not contain formatting characters, then packet 4 may contain a valid value, or it may contain an arbitary value of no specific meaning. Where the data value packet is not relevent for an event record, it is recommended it is set to a one byte integer of value 0.

This type of event record is typically used to convey the following types of occurrence:
- logic input changes
- relay output changes
- local setting changes

7.5 ADDITIONAL EVENT INFORMATION

It is intended that all events be displayed nominally with the information that is common to all of them i.e. that which is contained only in the type 0 Standard Event Record.

Where the occurrence requires additional information to be supplied to the master control unit than can be accommodated by the standard event record, three further event record types can be used. The type used depends on the type and quantity of this information.

The additional information contained in these event records should be available by expanding the display of the nominal event record.

7.5.1 TYPE 1: SHORT EVENT RECORD

A type 1 event record extends the format of the Standard Event Record by the addition of a Format Text packet and additional argument packets. The format text packet is a string of text containing a format control code for each of the remaining argument data packets. The additional event information is displayed by replacing each format control code in the format text with the appropriate representation of the corresponding data packet. The entire event record must fit within the single message frame of the event record, which restricts the size of the format text packet.

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value
5	18h+LL	Format text
6	xx	Argument 1
7	xx	Argument 2
...	...	...
n	xx	Argument m

A typical use of this event record is for fault information where the values of certain system parameters at the time of the fault are recorded.

7.5.2 TYPE 2: LONG EVENT RECORD

Where the number of arguments to be sent in a Type 1: Short Event Record is large, there may not be sufficient space left in the message frame to store the Format Text packet, or conversely, the format text packet may be so long that there is not enough space left to store the arguments.

The Long Event Record format overcomes this limitation by replacing the format text packet with a data packet containing a menu cell reference from which the actual format text can be extracted using the Get Text Courier command.

It is important that this text is constant, since it may not be requested immediately. The menu cell would normally be located in a column which has no column heading cell so that it is not normally visible.

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value
5	46h	Format text menu cell
6	xx	Argument 1
7	xx	Argument 2
...	...	...
n	xx	Argument m

A typical use of this event record is for fault information where the values of many system parameters at the time of the fault are recorded.

7.5.3 TYPE 3: COMPLEX EVENT RECORD

The complex event record allows the additional event information to be stored in a complete column of the slave device's database. Each menu cell in this column is used to store an argument's value and formatting control codes. Cells may also be used to store textual information that should be printed without any associated argument.

The additional event information is extracted from the column using the Get Column Text and Get Column Values commands. Since these result in blocked transactions, it follows that these commands may not be issued immediately, but may have to wait until any outstanding blocked transaction is complete. By this time, there is a possibility that several more complex event records may have been generated by the same device.

The extraction of complex event records is designed such that one column may be used as a template for the additional event information of many complex event records. This is achieved by including the menu cell reference of the extraction column along with an event number in the event record. The cell at row 1 of the extraction column is designated as a setting cell to hold the next event number to be extracted. Before performing the extraction, the value of the setting cell in row 1 is set to the event number specified in the event record. (The reply code on setting this cell should be checked to ensure the slave has performed the requested setting change). The remaining cells in the column will change to reflect the event number selected and then the additional event information can be extracted.

Packet#	DTL	Use
1	46h	Menu cell reference
2	38h+4 / 3Ch+7	Time tag (ms / IEC)
3	18h+LL	ASCII Text description
4	xx	Menu cell value
5	46h	Event extraction column
6	26h	Event number

A typical use of this event record is for a fault report where the values of many system parameters at the time of the fault are recorded, requiring detailed explanation or clear layout control.

7.5.3.1 Considerations For Complex Event Record Extraction

Column Display
It should not be possible to change the event number whilst the additional event information is being extracted, otherwise two fault records will become mixed together. It is therefore advisable to omit the column heading cell of the event extraction column such that it is hidden from any user performing normal menu database browsing.

Binary Flag Enumeration
Binary flag values present in the fault record should be enumerated by requesting the setting strings for the menu cell reference. Each setting string will enumerate one of the binary flags, starting at bit 0. Where a binary flag has no associated text, a setting string comprising a single space character should still be provided as a place marker. In this case the binary flag is to be ignored and no enumeration is to take place.

Event Number
The event number will use a cyclic numbering scheme to ensure records have unique identifiers.

7.6 EVENT CATEGORISATION

A master control unit may use the menu cell reference to categorise events for filtering, later retrieval or manual searching.

The following list of menu cell references should be used to indicate the following specific events:

Menu Cell Reference	Event Description
0020h	Logic input changed state
0021h	Relay output changed state
0022h	Alarm event
0023h	Protection Operation
01xxh	Fault record

Events from other menu cell references can be generated as a result of local setting changes, or by measurements changing outside pre-determined limits. The nature of these events is determined by the context and application of the particular menu cell within the slave device.

It should not be assumed that any readable data exists at the menu cell reference given in the event record, it may be used simply for event identification and categorisation.

8. Disturbance Records

8.1 OVERVIEW

Courier provides a standard mechanism for the extraction of disturbance records so that they may be extracted in a generic manner from all types of slave device. This mechanism allows for many different sizes of disturbance records to be accommodated with attributes designed to make conversion to the IEEE (C37.111) COMTRADE standard a straightforward process.

This chapter explains disturbance record extraction using a defined structure within the relay's database. Programmers should follow this model if they include disturbance records which are to be extracted using Courier.

8.2 DISTURBANCE RECORDER DESCRIPTION

A disturbance recorder stores samples of analog or digital data simultaneously on a number of channels. Each channel has a finite buffer size, so when the end of the buffer is reached, the oldest data is overwritten with new data.

In some recorders, the time interval between successive samples will be constant and determinable. In others it will vary, for example where the sampling frequency tracks the system frequency. In this instance, the time interval between each successive sample is recorded in a separate channel so that the waveform can be accurately reconstructed.

This recording process is continuous until a trigger condition occurs. Recording may stop at this point, or continue for a pre-programmed number of post-trigger samples and then stop. The time of the trigger point is recorded. The master control unit is notified of the presence of a disturbance record in a slave device by the Disturbance Record flag being set in the status byte from that device, which is returned in every response message.

8.3 COMPRESSED RECORDS

Compressed disturbance records are identified by the presence of the compression method cell at YY0A. When this cell exists it holds an unsigned integer to indicate the compression format of the available record. If its value is zero, or the cell does not exist, the record is uncompressed and can be extracted in the normal manner. For other compression formats, the compressed record can be extracted in its entirety using the Get Value command and the Block Transfer protocol on cell YY0B. It is permissible for a record to be available in either compressed format only, normal format only, or both formats simultaneously.

8.4 MENU LAYOUT FOR DISTURBANCE RECORDERS

The disturbance recorder requires two columns in the slave's menu and two entries in the communication system data column. The first column is the recorder control column containing settings and control cells for the recorder. The first item in this column is a joint status/control cell indicating whether the recorder is running, stopped or triggered and allows this state to be altered. The second item states the type of data being captured: samples, magnitudes or phases. Menu cells in rows 03-1Fh are reserved for future use but the remaining cells can be arranged as required for the particular implementation of the recorder.

The second column is used for extracting the record and is probably not available in the user menu, since the data is not normally displayable. The format of this column consists of some fixed cells at the top of the column pertaining to the structure of the recorder and information on the individual channels, followed by a variable number of cells depending on the number of channels used.

The Courier language has no specific commands for accessing the disturbance records, so all access is performed using standard menu request commands. The location of the slave's recorder columns is required by the master control unit, though. Rather than fixing these locations, the Communication System Data Column indicates their location.

8.5 SYSTEM DATA COLUMN STRUCTURE

This column is standard across all relays and is located at column BF00h.

Menu	*Cell Description*	*Data Type*
BF00	COMMUNICATION SYSTEM COLUMN	
BF01	Recorder control column menu location (XX00)	[DTL_MENU+2]
BF02	Recorder extraction column menu location (YY00)	[DTL_MENU+2]

8.6 RECORDER CONTROL COLUMN STRUCTURE

Menu	*Cell Description*	*Data Type*
XX00	RECORDER CONTROL COLUMN	
XX01	Start/Trigger recorder	[DTL_ISTR]
XX02	Recorder source: Samples, Magnitudes, Phases	[DTL_ISTR]
-------	XX03-XX1F reserved for future use	

XX01 Start/Trigger recorder
This is an indexed string cell with 3 possible values from 0 to 2.

0 = Stopped
1 = Triggered
2 = Running

The limits of this cell are from 1 to 2 since it is not possible to stop the recorder without first triggering it. Therefore, when in the stopped state, it is only possible to trigger and start the recorder. Some implementations may prevent manual restarting of the recorder until one or more previous disturbance records have been cleared or extracted.

XX02 Recorder source:
This indexed string setting cell determines the source of the analog data and has three defined possible values:

0=samples
1=magnitudes
2=phases
3=magnitudes & phases

Other values may be added since they will be displayed as text only, despite being an indexed string.

8.7 RECORDER EXTRACTION COLUMN STRUCTURE

Menu	*Cell Description*	*Data Type*
YY00	RECORDER EXTRACTION COLUMN *(NB. normally invisible to the user)*	
YY01	Select Record No.	[DTL_INT or DTL_UNS]
YY02	Trigger time	[DTL_MSTM] or [DTL_IECD]
YY03	Available Channel bit mask. 0= not fitted, 1 = available Requesting the indexed strings for this cell returns the channel names.	[DTL_BINF]
YY04	Channel types (bit mask). 0 = digital, 1 = analog	[DTL_BINF]
YY05	Channel offsets	[Repeated group of DTL_NUM]
YY06	Scaling factors	[Repeated group of DTL_NUM]
YY07	Skew Values	[Repeated group of DTL_INT]
YY08	Minimum Values	[Repeated group of DTL_INT]
YY09	Maximum Values	[Repeated group of DTL_INT]
YY0A	Compression format	[DTL_UNS]
YY0B	Upload Compressed record	[Repeated group of DTL_UNS]
-------	YY0C-YY0F reserved	
YY10	Record length	[DTL_INT or DTL_UNS]
YY11	Trigger position	[DTL_INT]
YY12	Time base (x by YY14 samples for real time intervals)	[DTL_NUM]
YY13	Δt for constant sample rate	[DTL_UNS]
YY14	Sample Timer	[Repeated group of DTL_UNS]
-------	YY15-YY1F reserved	
YY20	Upload channel 0 record	[Repeated group of DTL_INT/DTL_UNS/DTL_BINF]
YY21	Upload channel 1 record	["]
YY22	Upload channel 2 record	["]
YY23	Upload channel 3 record	["]
YY--	etc.	

YY01 Select Record Number

The Record Number selection cell is used to select a particular record for extraction for slave devices that support multiple disturbance records. Record number 0 (zero) always refers to the next available unextracted record to be read. Automatic disturbance record extraction should therefore set this cell to 0 before extraction takes place to ensure the correct record is extracted.

Once a record has been extracted, this cell should be reset by using the Reset Cell command to advance the read pointer to the next available unextracted record, which is now referred to as record 0. The record just extracted will now be referred to as record number 1. This may also have the effect of restarting the disturbance recorder if it has stopped. The disturbance record status bit may be reset by the action of this Reset Cell command if there are no more records to extract. Resetting this cell at any other time may have the undesirable effect of discarding a disturbance record.

Previously extracted records may be extracted manually by setting this cell to 1 for the last record extracted, 2 for the record before that, and so on. Negative values may be used to select as yet unextracted records out of sequence.

The limits of this cell will change to reflect the number of records that are stored in the relay, whether previously extracted or not. It is not an error for this cell to be non-settable or non-readable: such a condition simply indicates that the slave device supports only one disturbance record and the extraction procedure should continue as normal.

Changing the value of this setting will cause the remaining cells in this column to change their values to reflect the currently selected record.

YY02 Trigger time.
This cell logs the time that the trigger occurred to stop the recorder for the currently selected record. It may be expressed as a millisecond timer value or as the IEC formatted real time value.

YY03 Channels available.
The number of channels will vary between each relay model as each will monitor differing system quantities depending on their function. If the channels form a standard layout, some models may have some channels missing where some functions are not included. To cope with this, the channels in use are identified using a bit mask so that the channels need not be contiguous. A bit will be set to one for each channel used, starting at bit 0 for channel 0. The number of channels is limited by the size of the bit mask to a maximum of 32 channels. Channels may also be referred to by name rather than by number. Requesting the indexed strings for this cell will result in a group of text strings being returned where each string is the name of the corresponding channel. All channels (fitted or not) must have a name, even if this is a single space, since the indexed strings relate to the bit positions and not the fitted channels directly.

YY04 Channel types
Channels may be analog or digital, as described by this bit mask. For every channel identified as being fitted by the bit mask in YY03, the corresponding bit in this mask will be 0 if the channel is digital and 1 if the channel is analog. For COMTRADE compatibility, all analog channels should appear first, followed by any digital channels.

YY05 Channel Offsets
All recorder channels are 16-bits wide. On analog channels, this value needs to be converted to a real quantity. The COMTRADE format allows for a conversion of Y=aX+b. Requesting the value of this cell results in a repeated group of channel offsets (the 'b' values) for the analog channels only. Digital channels do not need offsets. These offsets are in Courier number or floating point format.

YY06 Scaling factors
Requesting the value of this cell results in a repeated group of channel scaling factors (the 'a' values) for the analog channels only. Digital channels do not need scaling factors. These scaling factors are in Courier number of floating point format. In this case of floating point, requesting the strings for this cell will yield the units applicable, i.e. 'A', 'V','Hz' etc.

YY07 Skew Values
Requesting the value of this cell results in a repeated group of channel skew values for the analog channels only. Digital channels do not need skew values. These skew values are in 16 bit integer format representing the skew in microseconds from the start of the sample period

YY08 Minimum Values
Requesting the value of this cell results in a repeated group of channel minimum values for the analog channels only. Digital channels do not need minimum values. These values are in 16-bit integer format representing the lower limit of the sample range for samples of each analog channel.

YY09 Maximum Values
Requesting the value of this cell results in a repeated group of channel maximum values for the analog channels only. Digital channels do not need maximum values. These values are in 16-bit integer format representing the upper limit of the sample range for samples of each analog channel.

YY0A Compression format
Compressed disturbance records are identified by the presence of the compression method cell at YY0A. When this cell exists it holds an unsigned integer to indicate the compression format of the available record. If its value is zero, or the cell does not exist, the record is uncompressed and can be extracted in the normal manner. For other compression formats, the compressed record can be extracted in its entirety using the Get Value command and the Block Transfer protocol on cell YY0B.

YY0A cell value	Format
0	Uncompressed
1	MV Platform compression
2 ->	reserved

YY0B Upload Compressed record
If the selected disturbance record is compressed (indicated by the value of cell YY0A having a value of 1 or more) the entire record can be extracted from this cell using the Get Value command. The response is a blocked transaction of repeated groups of type DTL_UNS of length 1 byte. The format of this compressed record is dependent on the compression format.

YY10 Record length
All recorder channels will be the same length, which is recorded here as a two byte integer, making the maximum record length 32767 samples, or 65535 if unsigned integers are used.

YY11 Trigger position
The trigger position is the sample number when the trigger occured and can be anywhere within the record length. The trigger position is a two byte integer.

YY12 Time base (multiply cells YY13 & YY14 by this value for real time intervals)
The recorder logs time intervals between adjacent samples, rather than absolute time, to save space. These time intervals are stored as 16-bit integer values. To convert these values to a real time in seconds, they must be multiplied by the value in this cell which is a scaling factor or time base for the relay. This value is in Courier number format.

YY13 Δt for constant sample rate
If the recorder uses a fixed sampling rate, the constant interval between each sample is stored here as a 16-bit integer. It must be multiplied by the time base value in cell YY12 to produce a real time value. This cell is not used by a variable time base recorder.

YY14 Sample Timer
This cell is used when the interval between samples varies from sample to sample, such as in frequency tracked applications. In these cases, each sample interval will be recorded in a separate timer channel accessed through this cell. Requesting values from this cell will result in a block transfer of repeated groups of unsigned 16-bit integers. The first sample time will be meaningless and should preferably be set to zero. The second sample time will be the time interval between the first and second samples. All these values should be scaled by the time base in cell YY12.

YY20-> Upload channel n record
This and remaining cells are used to store the actual recorder channel data. Channel N will be stored in cell YY20+N. Channels not fitted may or may not have a visible cell location, but will not be uploaded anyway. Requesting values from this cell will result in a block transfer of repeated groups of unsigned or signed 16-bit integers. If this channel is a digital channel, requesting the strings for this menu location will result in the name of each digital line within the channel being sent. Unused lines within a digital channel are specified by setting the name of the channel to a single space and the COMTRADE file generation will ignore the line. If the strings are not available, the master control unit will assume all lines are available and will create a name for each line by appending the line number to the name of the channel provided by cell YY02.

8.8 DISTURBANCE RECORD DIRECTORY

For slave devices that support multiple disturbance records, it is possible to present the user with a directory of all available records by using the following procedure.

1. Obtain the limits of cell YY01. The minimum value will indicate the number of unextracted records (negative value) in addition to record number 0. The maximum value will indicate the number of previously extracted records (positive value).
2. For each value in the range of the setting limits, do steps 3-5.
3. Set cell YY01 to the new value to select that record number.
4. Read the time of the record trigger from cell YY02.
5. Record the record number and trigger time for later use.
6. Next value
7. Display the list of record numbers and associated trigger times as a directory of the available disturbance records.

8.9 DISTURBANCE RECORD EXTRACTION PROCEDURE SUMMARY

The following procedure outlines the basic operation for automatically extracting a disturbance record (see also NOTES below):

1. Wait for the disturbance record flag to be set in the relay status byte.
2. Read cell BF02 to discover the disturbance record extraction column (YY00).
3. Set cell YY01 to 0 to select the next record. Ignore ERR_NOCODE (01) and ERR_NOSETTINGS (05) reply codes reported.
4. Read cell YY0A to determine the compression format.
5. If the compression format is zero, or unreadable, continue at step 8.

6. Read the entire compressed record from cell YY0B.
7. Goto step 30
8. Read cell YY02 to discover the trigger time of record.
9. Read cell YY03 to determine the number of channels.
10. Read cell YY04 to determine the type of each channel.
11. Read cell YY10 to determine the number of samples.
12. Allocate sufficient memory to store the disturbance record (based on the number of channels and samples).
13. Read cell 0009 to determine the system frequency.
14. Read cell YY12 to obtain the time base for the recorder.
15. Read cell YY13 to determine if a fixed sample rate was used. If successful, the value of cell YY12 multiplied by the value of cell YY13 indicates the time difference between two adjacent samples. Create a timer channel from this information.
16. If cell YY13 does not exist, a variable sample rate was used. Read cell YY14 to obtain a block transfer of the timer channel. Each value requires multiplication by the time base read from cell YY12.
17. Read cell YY11 to obtain the sample number of the trigger position.
18. Get the strings from cell YY03 to obtain the names of each channel.
19. If any analog channels are present do steps 20-24.
20. Read channel offsets from cell YY05.
21. Read scaling factors from cell YY06.
22. Read skew values from cell YY07.
23. Read minimum channel values from cell YY08.
24. Read maximum channel values from cell YY09.
25. End If
26. For each channel present do steps 27-28.
27. Read value of cell YY20+channel number to obtain channel samples.
28. If channel is digital, try to Get Strings for the same cell to obtain the names of each digital line.
29. Next channel.
30. Store the disturbance record.
31. Reset cell YY01 to acknowledge correct reception of the record.

NOTES

1. If the disturbance record flag in the relay status byte is ever reset during the extraction process, the current record should be discarded as this indicates that the disturbance recorder has restarted recording in the extraction buffer.
2. Care should be taken when designing slave devices to avoid overwriting a disturbance record buffer with new data whilst it is being extracted. If this condition is possible, the extraction process should be aborted by either:
   a) resetting the disturbance record status flag
   b) returning an ERR_NODATA (02) reply code in response to any extraction command.

9. Changing Settings

The operation of changing settings remotely has been designed to be generic: all information about how to change the setting is provided by the slave device itself. A method of supporting mutual exclusion is provided for slave devices which have more than one user interface.

The setting operation is divided into several distinct steps:

1. Obtain setting information from slave device
2. Change setting
3. Download new setting to slave device
4. Verify correct reception of setting
5. Execute setting change.
6. Read back setting.

The preload, verify, execute sequence for changing settings, indicated by steps 3-5 above, must not be interrupted by any other message to that device including global messages, however non-global messages to other device address are permissible.

9.1 OBTAIN SETTING INFORMATION FROM SLAVE DEVICE

Step 1 is accomplished by the master control unit issuing an Enter Setting Mode command and receiving a valid setting group in response. The slave device uses this command to set a semaphore to prevent other user interfaces from changing settings, until this setting change is complete. Likewise, if its internal semaphore is already set when this command is received, the slave device will respond with an error code indicating another setting change is already in progress.

When a valid setting group is returned, the setting operation has started and must be seen through to completion. If at any time the setting operation is to be abandoned, the Abort Setting command should be given to the slave device so that it may reset its semaphore flag to allow other setting changes.

To prevent lock-ups due to communication failures, the slave device starts a timer when it sends the limits for a setting. If the timer times-out before the setting operation is complete, the slave device will abort the setting operation itself, restoring the original setting and allowing further setting changes. This timer is usually set to a minimum of two minutes, but it may be greater than this, e.g. 15 or 30 minutes. If the master control unit requires more time to

change the setting, it can issue further Enter Setting Mode commands to the same menu cell location, within the time out period (which, for this purpose, can be assumed to be 2 minutes), which will reset the slave device's timer. Issuing an Enter Setting Mode command to any other menu cell location will result in an error, but will not abort the original setting operation.

9.2 CHANGE SETTING

No matter what type of setting is to be changed, the setting group will contain the current value of the setting, its minimum and maximum values and the step size of valid increments of the setting between these values. (Some setting limits also contain an additional multiplier value). This allows any setting to be changed within a linear range.

Non-linear setting ranges are not catered for, but this is not usually important. Non-linear setting ranges are normally used to cater for a very wide setting range where the resolution is not so important at the top end. The non-linear range allows incremental settings to step from the low end to the top end of the range in a shorter time. However, using a master control unit any setting can be directly typed in without the need to step through all intermediate values, therefore the step size is not so important.

A step size of zero is permitted to allow any setting value between the minimum and maximum values to be set. This is particularly relevent when using floating point values where rounding errors may cause otherwise acceptable values to be rejected. Where the step size is zero, the slave device may adjust the received setting to the closest internally acceptable value. The master control unit must therefore re-read the setting value after execution to determine the actual value applied to the setting cell.

When Settings Limits are received from a device they should be checked for validity as follows:

1. If step size is negative, it should be set to the absolute of its received value.
2. If the maximum is not on a step increment then it should be set to the step increment immediately below its received value.
3. If maximum is less than the minimum, set the maximum equal to the minimum.
4. If the cell value received is less than the minimum, set it to the minimum.
   If the cell value received is greater than the maximum, set it to the maximum.
   If cell value is a character string, both should be done for each character.
5. If the cell value is not on a step increment it should be put on a step increment.

If the step size is zero it indicates that any value between the maximum and minimum limits is valid.

If, as a result of checks 1 to 3, any changes are made to maximum, minimum, or step size, the user should be informed that there was an error and value(s) have been modified, before allowing the setting change to proceed.

It is valid to receive a value to be edited that is outside the maximum or minimum limits, or not on a step size. This can be the case, for example, with combined control/status cells where only a limited range of values are user settable.

Note that step values increment from the minimum value. A minimum of 5 and a step size of 7, will give valid values of 5, 12, 19, etc.

9.3 DOWNLOAD NEW SETTING TO SLAVE DEVICE

When the user has entered a new value it is sent to the slave device with the Preload Setting command. The slave device does not action the setting change, but merely stores the value, temporarily, and reflexes it back to the master control unit for verification.

9.4 VERIFY CORRECT RECEPTION OF SETTING

The master control unit must verify that the reflexed setting is the same as the one it sent initially. If it is not, the master control unit should send the Abort Setting command and indicate the error.

9.5 EXECUTE SETTING CHANGE.

If the reflexed setting verifies correctly, the master control unit should send the Execute Setting command. On receipt of this command, the slave device will check that the pre-loaded value is valid. If any discrepancy is found, the setting operation is abandoned and an appropriate error code is returned. If the verification is successful, the slave device will action the new setting and exit from setting mode, thus resetting the semaphore flag. A reply code is returned to indicate the successful setting change; either reply code 0 (ERR_OK) or code 8 (ERR_OKCHANGE). The difference between these replies is that ERR_OKCHANGE also indicates that the action of changing the setting has also affected the slave's menu elsewhere. In this event, the current column text and values should be re-read from the slave device so that anything that has changed is reflected in the master control unit. The column headings will also need re-reading the next time they are displayed.

9.6 READ BACK SETTING

It is not correct to assume that if a slave device returns ERR_OK to a setting change, that the new setting entered will be the new value of the cell the next time it is read. This is especially so in the case of control cells where the cell value is a combined status and control cell. Also, where floating point settings are involved, the slave device may accept a new setting but round it up or down slightly so that it can be represented by an internal value. It is therefore necessary for the Master Control Unit to re-read the menu cell's value before displaying it.

9.7 UNUSUAL SETTING TYPES

Most settings consist of a linear setting range where the limits are of the same type as the setting. However, some settings operate quite differently, despite having the same set of limits.

9.7.1 TEXT STRINGS

Text strings have a DTL of type TEXT, but the limits have type UNSIGNED INTEGER of 1 byte in length. The limits correspond to the (linear) range of valid characters that each character in the string can be assigned. The setting range refers to characters from the Courier character set. The number of characters in a string is fixed and is determined by the length of the string sent by the slave; trailing spaces should be added to the string to achieve this.

9.7.2 INDEXED STRINGS

An indexed string setting is treated in the same way as an UNSIGNED INTEGER setting: it is a value which can be incremented or decremented by the step size between the minimum and maximum limits. For each value in the range, the master control unit should not display a

numeric value, but the text from the Nth position in the list of text strings extracted from the slave device using the GET STRINGS command, where N is the current value of the setting.

9.7.3 INDEXED COURIER NUMBERS (GROUP TYPE 22H: SETTING LIMITS WITH MULTIPLIER)

Menu cells having a displayable value of type Courier number may have normal setting limits of Courier number, indicated by a setting limits group of type 21h. Alternatively, a group type 22h: setting limits with multiplier group, may be returned, indicating the normal value, minimum, maximum, and step size values of the setting limits are all INTEGER settings, but an additional multiplier of type Courier Number will be appended. In this instance, the setting should be treated as a normal integer setting where the value can be incremented or decremented by the step size between the minimum and maximum values. However, the values should be converted to Courier number form by multiplying them by the multiplier part of the limits before display. The integer value is sent back to the slave device; the numeric value is for display purposes only. This technique is used for settings which would otherwise result in rounding errors due to the limited resolution (4 significant digits) of the Courier number type.

9.7.4 BINARY FLAGS

Binary flag settings are actually several settings, each being a flag having only two states, combined into a single setting. The flags are stored as separate binary bits packed into an unsigned integer starting with the lowest bit (bit 0). Since the smallest integer unit is a byte, binary flag settings are of 1,2 or 4 bytes in length. The highest bit number used +1 is stored in the maximum limit value. This therefore indicates the maximum number of flags that are settable by assuming that all lower flags are settable. The minimum limit value holds a binary mask which determines which flags are settable by having the mask bit set to one if the flag is settable or zero otherwise. Non-settable flags are set to zero by default. The step size limit value is not relevant, but is set to 1. For example:

Example 1 : 3 flag settings are settable, in bit positions 0,1 & 2.

7.....210

flag arrangement in setting = 00000XXX
Value DTL = DTL_BINF, length 1
Maximum value = 3
Minimum value (mask) = 00000111 = 7

Example 2 : 3 flag settings are settable, in bit positions 0,2 & 4.

76543210

flag arrangement in setting = 000X0X0X
Value DTL = DTL_BINF, length 1
Maximum value = 5
Minimum value (mask) = 00010101 = 15h = 21 (decimal)

Each flag can have text associated with it to describe its purpose. When a binary flag setting is being set, the master control unit will issue the Get Strings command for that menu cell. If there are textual descriptions for the flags, the slave device will respond with a indexed strings group packet where each string is the text for each bit position, starting at bit position 0. Unused bit positions or un-settable flags must still have a text string, even if this is just a single space character. If there are no textual descriptions, the slave device will return an error code 2.

9.7.5 FLOATING POINT NUMBERS

Floating point numbers present an accuracy problem when trying to use certain step sizes since the step size cannot always be represented accurately. This will cause the validation routine to fail for certain valid settings. To accommodate these types of number a step size of 0 can be used to indicate that any value between the minimum and maximum value will be accepted by the slave device. However, the slave device may adjust the new setting to a value that is more appropriate (or corresponds to an internal step size value) and therefore the master control unit should re-read the setting after it has been accepted (as it should be doing anyway).

9.7.6 IEC TIME & DATE

When changing IEC time & date settings, the minimum and maximum limit information is ignored. The step size represents the resolution of the setting, i.e. the smallest increment that can be set. This is interpreted by examining the individual fields from the smallest resolution upwards until a non-zero value is found.

Setting Resolution	Example step value
1ms	1 Jan '00 00:00:00.001
20ms	1 Jan '00 00:00:00.020
1s	1 Jan '00 00:00:01.000
1min	1 Jan '00 00:01:00.000

9.8 FAMILY SETTINGS

In some slave devices, it is considered necessary to action some settings simultaneously as a family since actioning the settings individually could cause an unstable setting configuration. This procedure is not considered to be part of Courier, but rather a higher level implementation of the Courier commands determined by the slave device.

The following two methods are recommended for implementing family settings using the existing Courier communication level 1 commands and protocol.

Family Settings Method 1
The first method involves having two completely separate groups of settings in the slave device where each group contains a family of settings and is itself contained within a separate column of the slave device's menu database. Only one group is active at a particular instance in time which is controlled by an additional setting menu cell. The settings in the inactive setting group may be changed on an individual basis in the usual manner. When all settings have been downloaded, the active setting group is switched which will apply all the new settings in this second group in one go, thus minimising the amount of time that the slave device is suspended from normal operation.

Family Settings Method 2
The second method also stores a family of settings within a column of the menu, but has an additional menu cell: a status cell, indicating the status of the settings within the family. The settings are changed on an individual basis, but when they are actioned, they are merely stored in a temporary setting buffer rather than being used by the slave device immediately. Whenever any setting within that family is changed, the text of the status cell is changed to indicate that some settings within the family have been altered. When all appropriate settings

within the family have been changed, it is then necessary to update all the settings from the temporary setting buffer and apply them simultaneously. The status cell is used to do this as it is also a setting cell, probably implemented as an indexed string setting. By setting this cell to an appropriate value, all settings in the family will be updated from the temporary buffer. A second setting option will allow all changed settings within the family to be restored to their previous value. Once the family has been updated the status cell will reflect the new state of the setting group.

If an indexed string type setting is to be used for the status cell, the following values are suggested:

0 "Settings OK"
1 "Settings changed"
2 "Accept new settings"
3 "Restore old settings"

Although 4 possible values are indicated, only values 2 and 3 can be set by the user which perform the appropriate command. Values 0 and 1 are generated by the action of altering settings and applying the 2 actionable commands.

10. Setting Transfers

10.1 OVERVIEW

The settings of a slave device may be extracted for storage on an external storage device in a standard setting file format (see company standard 9106.8002). The settings within such a settings file may also be downloaded to a slave device.

10.2 CONSIDERATIONS FOR SETTING TRANSFERS

10.2.1 SETTING TRANSFER ORDER

In order to allow setting extraction and download between a slave device and a master station in a consistent generic manner, the order of setting transfer is defined to be in menu cell reference order using a newspaper-column-like traversal of the menu table. Settings are transferred column by column starting at column 00h and advancing to column FFh. Each column is transferred row by row starting at row 00h and advancing to row FFh.

Setting information is stored in the setting file in the order of extraction, i.e. with increasing menu cell references for the default setting extraction procedure above. Setting cells are downloaded to the slave device in the order they are encountered in the setting file. The menu database should therefore be designed such that a default setting file can be downloaded to a slave device allowing all setting cells to be set correcty and in the correct order.

10.2.2 FAMILY SETTINGS

Courier assumes that once a setting has been executed, no more actions are required in order for the device to use that setting. Certain menus, however, may require that once a family of settings has been changed, a further setting is required to action them as a whole (although this use is discouraged). In order to guarantee these actions are carried out when downloading settings to a device, a menu cell must be provided *after* all of these related settings to perform the collective action. On extraction, this cell must supply a value which will action the group when downloaded.

10.2.3 DEPENDENT SETTINGS

Certain settings may be dependent on other settings elsewhere in the menu. For example, in a protective overcurrent relay, all current settings may be scaled by a current transformer ratio, so that they are displayed in primary units. To faciliate correct downloading, the setting

containing the scaling factor or other dependency must appear in the menu traversal *before* the settings which depend on it.

10.2.4 SETTING ACCESSIBILITY

Two methods exist to extract settings from a slave device, as detailed below. The preferred method is to use the Get Column Setting Limits command which returns the limits of all settings within a column. This method is independent of password levels. However, if this command is not supported by the slave device, the master determines which cells are settable by using the Enter Setting Mode command to extract the limits from each cell within a column in turn. This implies that the correct level of accessibility must be set in order to extract the limits of interest by previously entering the correct password. Any password protected cells, for example, will not be extracted or downloaded unless the password has been entered.

10.2.5 SETTING TRANSFER CELL

Some or all of the above points may conflict with the normal requirements of the menu layout concerning useability. To solve this problem, a standard cell at location BF03h is used to indicate to the device whether a setting download or extraction is in progress. When set to 1 it indicates to the slave device that it may rearrange or reconfigure its menu database to be compatible with the extraction or download process. This may cause menu cells to be re-located or alter their visibility according the needs of the application.

10.3 SETTING TRANSFER PROCEDURES

The following details the procedures for performing setting extraction from, and download to, a slave device.

10.3.1 SETTING EXTRACTION

1. Request user to enter the password to unprotect all settings.
2. Set cell BF03 to the integer value of 1 to indicate that setting transfer is about to take place.
3. Use Get Column Headings to obtain a list of all valid columns.
4. For each column in turn:
5. Use Get Column Text to obtain a list of all valid rows within the column
6. Attempt to use Get Column Setting Limits to obtain a list of all setting limits for all setting cells within the column.
7. If step 6 fails with ERR_INVALIDCMD Then
8. Use Get Column Values to obtain a list of all data cells within the column
9. For each data row:
10. If Enter Setting Mode succeeds
11. Store Limits
12. Abort Setting
13. End If
14. Next Row
15. End If
16. If Step 6 succeeds then Get Column Values can be used to obtain all the data values, if required.
17. Next Column
18. Set cell BF03 to the integer value of 0 to indicate setting transfer is complete
19. Reset Password

The slave device response to the Get Column Setting Limits command is a block transaction of column setting limit groups. These limit groups are identical to group 21h (setting limits group) and group 22h (setting limits with multiplier group) with the exception that a new data packet is inserted at packet#1 position of the limit group indicating the menu cell reference. These limit groups will have group types 23h and 24h respectively.

10.3.2 SETTING DOWNLOAD

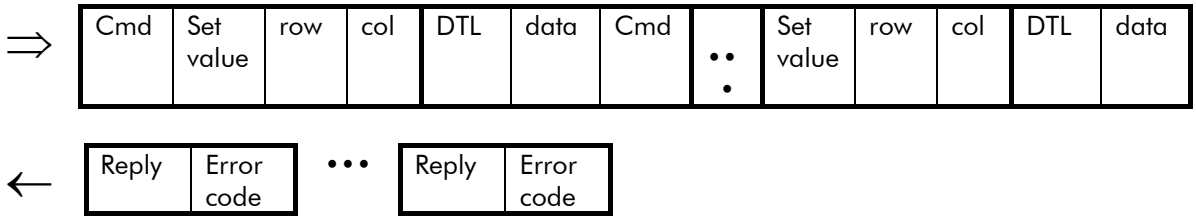
1. Request user to enter the password to unprotect all settings.
2. Set cell BF03 to the integer value of 1 to indicate that setting transfer is about to take place.
3. Attempt to use Set Value to set the first menu cell setting in the data file.
4. If Set Value fails with ERR_INVALIDCMD, goto step9
5. For each remaining menu cell setting in data file:
6. Use Multiple Transactions of Set Value to set each remaining menu cell
7. Next Menu Cell
8. Goto step 14
9. For each menu cell setting in data file:
10. Enter Setting Mode
11. Preload new Setting
12. Execute Setting
13. Next menu cell
14. Set cell BF03 to the integer value of 0 to indicate setting transfer is complete
15. Reset Password

An example of downloading several data packets in a single message is as follows:

Key

⇒ Command message from a master to a slave

← Response message from a slave to a master



11. Password Protection

11.1 OVERVIEW

This section describes how password protection is applied to Courier based slave devices. Password protection provides security for configuration data, settings and control functions etc. to prevent operation by unauthorised personnel. The application functions that require password protection and the level of access required is defined by each application. However, the mechanism for implementing passwords should conform to the details of this section.

Password protection should be applied independently on each user interface. Selecting an access level on one interface should not affect the access level on any other interface.

11.2 MULTI-LEVEL PASSWORDS

This implementation provides for multiple passwords to be used on a single device, with each password enabling a specific access level. All user passwords shall contain 4 uppercase characters in the range 'A' to 'Z'. These shall initially have a default setting of "AAAA".

Access levels are numbered with increasing functionality:
- Level 0 - Maximum protection
- Level 1 - Limited protection
- Level 2 - Unrestricted access

Level 0 is the default access level applied when no password has been provided.

Access levels are hierarchial such that access level N includes all of the functionality of access level N-1.

This implementation does not restrict further access levels from being defined where finer distinctions are required between the access levels.

11.3 ENTERING PASSWORDS

Entry to the various access levels of the slave device is achieved by entering the corresponding password for the required access level into the password menu cell at 0002 in the System Data Column. On acceptance of a valid password, the highest access level that matches the given password shall be applied.

Entering another valid password whilst an access level is already applied will result in the access level functionality appropriate to the new password being applied.

Entering an incorrect password at any time will result in the lowest level functionality being applied (i.e. the functionality of the device when no password is required, defined by the password control cell).

It is not possible to read the value of the password. The password cell shall return its value as a string of four asterisks ('****') at all times and for all types of request.

The return of an ERR_OKCHANGE reply code after entering a password should be avoided for efficiency reasons, since a password may need to be entered here as a result of an ERR_NOPASSWORD being returned when changing another setting. ERR_OKCHANGE would require the master to re-read the menu at an inopportune time.

11.4 ACTIVE PASSWORD DISPLAY

The currently active password level shall be displayed in the System Data Column as an integer.

Menu cell	Data type	Description
00D0	Unsigned integer	Active Access level

Value	Meaning
0	Active access level = Level 0
1	Active access level = Level 1
2	Active access level = Level 2

11.5 RESETTING PASSWORDS

The current access level may be reset by selecting the reset function of menu cell 0002 (either by the front panel, or by sending the Courier command RESET CELL (0002)).

Entering an incorrect password at any time will result in the password level being reset.

Passwords shall be reset to the default access level after a period of inactivity on the current user interface. Any password timeout scheme may be used, and thus the password protection re-instated, provided it is not indefinite and it is a minimum of 2 minutes after the last communication. This latter condition does not include normal polling commands such as Poll Status, Poll Buffer, Get Value, Send Event or Accept Event, so actions such as normal browsing of the database or setting changes will be required to maintain the current access level. The password scheme used for Courier communications should be independent from any other user interface.

11.6 CHANGING PASSWORDS

Each access level shall have its own password setting cell. These shall each be password protected cells requiring the equivalent or higher access level to be active.

Menu cell	Access Level Required	Description
00D2	1	Password for level 1
00D3	2	Password for level 2
00D4-00D8	>2	Reserved

11.7 PASSWORD CONTROL

A password control menu cell shall be provided in the System Data column to enable the functionality of the access levels. This shall be implemented as an integer setting indicating the lowest (default) level of access that is available without entering a password.

Menu cell	Access Level Required	Data type	Description
00D1	Highest available	Unsigned integer	Password control

Value	Default Access Level	Meaning
0	Level 0	Menu is fully protected
1	Level 1	Limited accessibility
2	Level 2	Full menu accessible without password

Setting a default access level makes all of the features protected by that level and all lower levels available to the user without entering a password.

11.8 BACKUP PASSWORDS

The implementation of a backup password is application specific and is not part of the Courier Standard. However, this standard defines a sequence number cell which can be used to implement a backup password scheme.

This sequence number is incremented each time a backup password is entered and alters the backup password required for the next

11.9 PREVIOUS IMPLEMENTATIONS

Earlier implementations of password protection in Courier based devices did not support multiple passwords. In these devices menu cells 00D0-00D8 will not exist. The password may be changed by entering the new password in menu cell 0002 after a valid password has been entered. This implementation is to be discouraged in new devices.

12. General Block Transfers to a Slave Device

12.1 BLOCK TRANSACTION

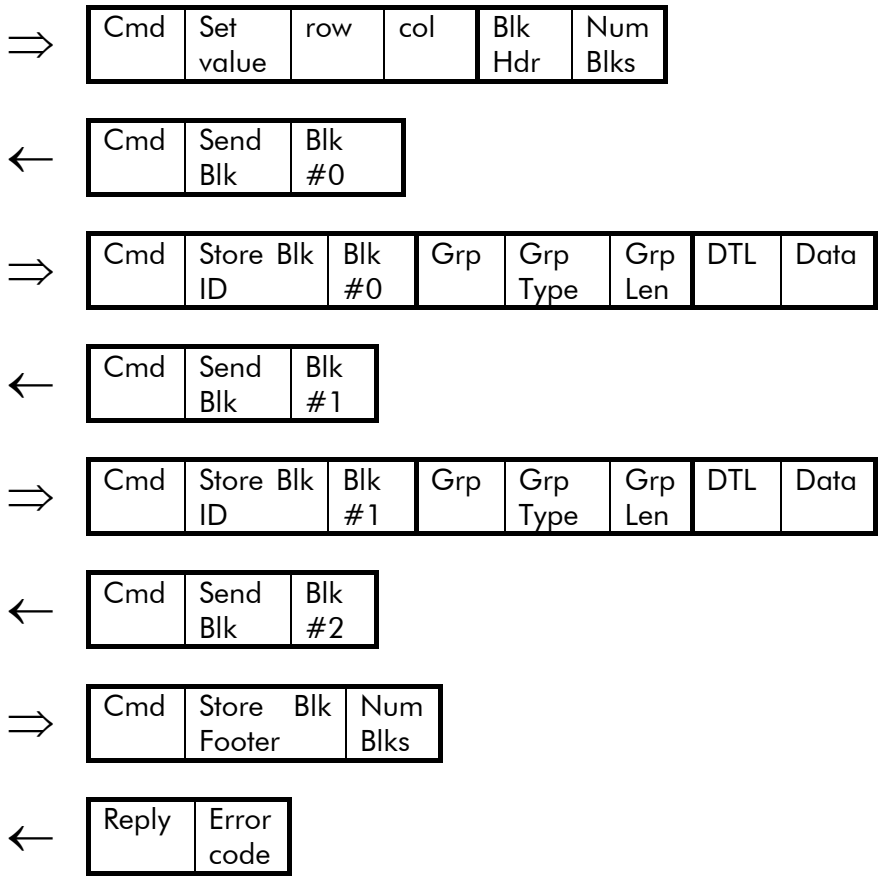
Block transfers to a slave device can be achieved using the Set Value command and specifying a Block Header as the data packet to be set. This will initiate a block transaction to the device which essentially works in the reverse sense to extractions using block transactions. However, two new commands are required to replace the block identifier and block footer data types when sending block transfers to a slave device, namely the Store Block Identifier and Store Block Footer commands, although their meanings are identical to the Block Identifier and Block Footer data types.

This block transfer mechanism is intended to be used to transfer data stored in the Courier Binary Block Transfer file Structure detailed in Company Standard 9106.8006.

Key

⇒ Command message from a master to a slave

← Response message from a slave to a master



12.2 APPLICATION LEVEL IMPLEMENTATION DETAILS

Block transfers to random menu cells are not supported. For compatibility and integrity of data, a mechanism is provided in the slave device database to allow the transferral of large blocks of data in a common manner. This can be used for many purposes such as: program changes, scheme logic, settings etc. and is application dependent.

A column of the menu is required to provide a specific layout of additional control cells required for the transfer procedure. Cell BF06 is used as a pointer to indicate which column of the menu is used for these control cells.

The transfer column requires the following control cells:

Menu	*Cell Description*	*Data Type*
UU00	DATA TRANSFER	
UU04	Domain	[DTL_ISTR]
UU08	Sub-Domain	[DTL_ISTR]
UU0C	Version	[DTL_UNS]
UU10	Start	[DTL_UNS]
UU14	Length	[DTL_UNS]
UU18	Reference	[DTL_TEXT]
UU1C	Transfer Mode	[DTL_UNS]
UU20	Data Transfer	[Repeated groups of DTL_UNS]

UU04 Domain

The memory area of the slave device may be sub-divided into areas for differing purposes, such as Programs, Scheme Logic, Settings etc. The domain name will select a specific area of the device which is to be transferred. The domain names are specific to a slave device application.

UU08 Sub-Domain

A slave device domain may be further divided into sub-domains. For example, in a multiprocessor implementation, a Program domain may be divided into separate sub-domains for each processor; or conversely, a processor domain may be divided into sub-domains for Programs, Scheme Logic, Settings etc. The sub-domain name will select a specific area of the domain which is to be transferred. The sub-domain names are specific to a domain in a slave device application.

UU0C Version

The data transferred to the slave device using this method is transparent to the Courier Protocol and its format is application and domain specific. The version number provides a means of verifying that the format of the data is compatible with the slave device. This consists of a two byte unsigned integer where the most significant byte (MSB) is the major version number and the least signicant byte (LSB) is the minor version number.

UU10 Start

This cell specifies the start address within the selected domain of the data to be transferred.

UU14 Length

This cell specifies the length of the data to be transferred in bytes.

UU18 Reference

Where data is being downloaded to the slave device, this cell may be programmed with a reference string to indicate the origin of the stored data. A minimum of 32 characters will be supported.

UU1C Transfer Mode

Transferral of large data files may require complex pre- or post-processing of the data before or after its transmission. This combined control / status cell is used to control the procedure of transferring data files to and from the slave device. It is set to various settings to perform actions within the slave device and it's value is polled to determine when the action is complete.

It can take on the following values:

Use	Value	Description
Control	0	Prepare to receive data file
Control	1	Complete reception process
Control	2	Prepare to transmit data file
Control	3	Complete transmission process
Status	4	Ready to receive data file
Status	5	Ready to transmit data file
Status	6	OK
Status	7+	ERROR

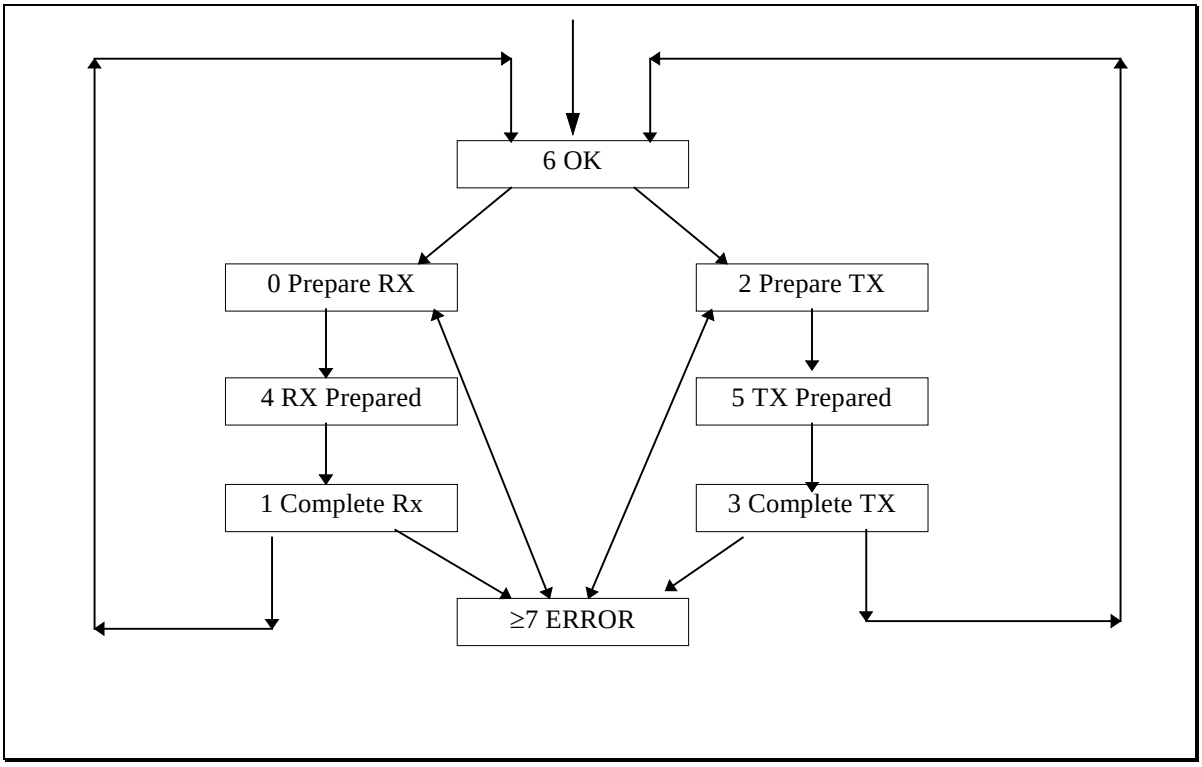


Figure 12-1 State Transition Diagram for Transfer Mode Cell

An error condition is indicated by the Transfer mode cell being set to a value of 7 or higher. The nature of the error condition can be described by assigning indexed strings to this cell on an application basis.

UU20 Data Transfer

This control cell is used to perform the actual data transfer by issuing the Get Value command for upload operations and the Set Value command for download operations. Due to the efficiency required and transparency of information afforded by this mechanism, it is recommended that the data file be sent within a repeated group of unsigned integers of length 1 byte.

12.2.1 BLOCK TRANSFERS TO A SLAVE DEVICE

This mechanism is intended to be used with data files conforming to Company standard 9106.8006. This data file type contains header records containing information to be downloaded to the slave device or verified prior to transferring the data. The following procedure should be adopted to transfer files to a slave device.

1. Confirm the model number of the slave device matches the MODEL record of the data file.
2. Write the mandatory DOMAIN record to cell UU04.
3. If a SUBDOMAIN record is present, write this to cell UU08.
4. Write the mandatory VERSION record to cell UU0C.
5. If a START record is present, write this to cell UU10.
6. If a LENGTH record is present, write this to cell UU14.
7. If a REFERENCE record is present, write this to cell UU18.

If all of the above values can be written without error, the transfer procedure can commence:

8. Write 0 to cell UU1C to allow the slave device to prepare for the transfer.

9. Wait for the slave device to set cell UU1C to 4, allowing the transfer to begin.
10. Transfer the data to cell UU20 using the Set Value command.
11. Write 1 to cell UU1C to allow the slave device to perform any necessary post-transfer processing.
12. Wait for the slave device to set cell UU1C to 6, indicating the transfer was successful.

The slave device will set cell UU1C to ≥7 if an error occurred.

12.2.2 BLOCK TRANSFERS FROM A SLAVE DEVICE.

This mechanism is intended to be used to store data files conforming to Company standard 9106.8006. This data file type contains header records that are used to identify the format and content of the data file. The following procedure should be adopted to transfer files from a slave device:

1. Select the Domain by writing to cell UU04.
1* Select the Sub-Domain by writing to cell UU08.
2* Select the Version by writing to cell UU0C.
3* Select the Start address by writing to cell UU10.
4* Select the Length by writing to cell UU14.

Steps marked by an asterisk (*) are optional depending on the application. It is anticipated that writing to a cell will cause the slave device to set the subsequent cells to default values ready for extraction.

6. Read the model number of the slave device from cell 0006 and store in the MODEL record.
7. Read the serial model number of the slave device from cell 0008 and store in the SERIAL record.
8. Read the Software reference number of the slave device from cell 0020 and store in the SOFTWARE record.
9. Read cell UU04 and store in the DOMAIN record.
10. Read cell UU08 and store in the SUBDOMAIN record.
11. Read cell UU0C and store in the VERSION record.
12. Read cell UU10 and store in the START record.
13. Read cell UU14 and store in the LENGTH record.
14. Read cell UU18 and store in the REFERENCE record.

Now that the information describing the data file have been read, the data itself can be transferred:

15. Write 2 to cell UU1C to allow the slave device to prepare for the transfer.
16. Wait for the slave device to set cell UU1C to 5, allowing the transfer to begin.
17. Transfer the data from cell UU20 using the Get Value command.
18. Write 3 to cell UU1C to allow the slave device to perform any necessary post-transfer processing.
19. Wait for the slave device to set cell UU1C to 6, indicating the transfer was successful.
20. Store the number of bytes transferred in the SIZE record.

The slave device will set cell UU1C to ≥7 if an error occurred.

12.2.3 APPLICATION ISSUES

Courier provides the basic mechanism to transfer blocks of data to a slave device. It also provides a standard menu layout for the semantics and control of the transfer.

It is the responsibility of the slave device application to determine whether downloaded data can be stored and used immediately, or whether it should be buffered and validated before being utilised after the block transfer is complete. The latter would obviously require an amount of free memory equal to the size of the data transfer to temporarily store the data before validation. Consideration should be given to the degraded cases outlined in case of communication failure.

12.2.4 SECURITY

Only the standard level of security offered by the physical and data link layers of the communication system and the Courier Protocol are provided. If additional validity checking of the data is required, this should be done by the application level of the devices, perhaps by encoding the data or by including additional CRC or checksum information. The structure of the data to be transferred is largely transparent to the transfer process.

The security of the Transfer Mode and Data Transfer cells may need to be increased depending on the application. One easy way to do this is to make one or both of these cells password protected. The password level required would be dependent on the Domain (and sub-domain) selected. It may be possible to alter the visibility of certain domains and associated control cells depending on the current level of password access.

12.2.5 ERROR CONDITION CONSIDERATIONS

Lost Messages
Any lost message will result in the Master Device re-sending the last message according to the standard retry procedure. A Slave Device will request blocks in sequence order, wrapping from block 255 to block 0. It will ignore any out of sequence blocks or retries of the same block and request the next sequential block number it is expecting. The Master Device shall send the same block until the slave acknowledges its reception by requesting the next block number in sequence, or until an error code is received,
The Slave Device must accept (and ignore) a possible retransmission of the original Set Value command whilst expecting the first block of data from the Master. At any other time during the block transfer, this would be considered an error and the slave should respond with ERR_NOACCESS.

Attempt to change settings during block transaction download.
Executing a block transfer with the Set Value command is equivalent to changing a setting and, therefore, mutual exclusion may be required to prevent other settings being changed either locally or remotely; due consideration should be given to this in the application. The Enter Setting Mode command should be spurned with the ERR_NOACCESS reply code. Similarly, this command should not be accepted whilst a setting change is already in progress.

Abort download from Master
The Master may abort the download operation by sending an Abort Setting command instead of one of the blocks or the block footer.

Abort download from Slave

The slave device may wish to abort the download if, for example, there is a communication failure or insufficient memory to store the data. In such cases it need not (nor cannot) notify the Master Device of the abortion, until the Master Device sends the next block of data. The Slave Device can then notify the Master of the download abortion by responding with the ERR_NOACCESS or ERR_NOVERIFY reply code.

Communication Failure during download

In the event of a communication failure, the Master Device will abandon the transfer procedure. It cannot be assumed that the Slave Device will receive an Abort Setting command from the Master. The Slave Device should therefore time the interval between successive blocks and internally abort the setting if a valid block transfer is not received within a time out period. (1-5 minutes programmable is suggested).

13. Courier Standard Procedures

13.1 IDENTIFYING SLAVE DEVICES

A master control unit must establish that a slave device is present at a particular address before it can communicate with it properly. The procedure for this is to use the IEC870 Reset Remote Link procedure which also synchronises the FCB bit between slave and master devices. Once this has been established, the master control unit can periodically issue the Poll Status command to confirm that the slave device is still functioning and to determine if any events have occurred.

If a communication error exists and a slave device stops communicating, the Reset Remote Link command should be issued to that address in order to re-establish communications.

On first identification of a slave device, the master control unit should interrogate the slave to determine:

1. plant reference
2. system description
3. communication level
4. plant status word
5. control status word

If event records are to be used, the initial state of menu cells that will be reporting changes to themselves should be read, including:

6. output contacts
7. opto-isolated inputs

13.2 AUTOMATIC ADDRESS ALLOCATION

By polling all the available addresses and issuing the Reset Remote Link command to unused addresses, the master control unit can identify when new slave devices are added to the system.

Valid addresses are 1-254. No response will be given by a slave device with its default address of 255 (also the global address). No slave device should have the permanent address of 0 which is used for automatic address allocation.

If a response is detected from address 0, the master control unit should automatically change the address of that device to an address which is currently unused. If a slave device stops communicating, its address should still be marked as used until it is manually marked as unused by an operator. This prevents the automatic addressing from allocating an address to a new slave device which is used by another slave device which is currently not communicating. For automatic address allocation to work properly, the master control unit must maintain a non-volatile list of all used addresses on the next level of the system; this is especially important on systems with self-powered slave devices.

Manually setting the address of a slave device to 0 therefore has the effect of forcing the master control unit to automatically assign a free address to the device, thus removing the chore of having to maintain a list of free addresses during the commissioning process.

Automatic address allocation of a slave device can only be done using single level addressing from the next immediate master control level in the address hierarchy.

13.3 CHANGING A SLAVE DEVICE'S ADDRESS

This can be performed using the standard setting change procedure on the address setting cell in the System Data Column of the slave device. However, the reply after executing the command may come from the old address or the new address. The preferred method is to use the global Change Device Address command. This changes the address of a uniquely specified slave address to a new value. Being a global command eliminates the problem of response messages coming from 2 possible addresses. The master control unit should however verify the slave has changed its address.

13.4 INSTALLING SLAVE DEVICES

Slave devices will have a default address of 255 when installed, to avoid conflict with any other address on the system. This should be manually changed to a free address (or to 0 to force the master control unit to do this automatically, see above) during the commissioning process.

Some slave devices may not have a local user interface preventing the address from being changed manually. In these cases it will be necessary to install the slave device via the communication system.

The Courier command: Change Device Address, is used for this purpose. The command is directed to all slave devices, since the message is normally sent globally to address 255. To prevent all slaves from actioning the command, the message also contains the serial number and old address of the slave device to uniquely specify it. Only this slave device will alter its address to the supplied new address. If this is zero, the master control unit can subsequently automatically change its address as above.

13.5 BROWSING A SLAVE DEVICE'S DATABASE

One of the benefits of a Courier based system is its distributed database approach. Since each menu cell contains a piece of descriptive text, it is like having a reference manual built-in to each device. It is therefore possible to simply connect the communication system to a slave device and start communicating with it by simply browsing its database. This method of browsing provides an easy and intuitive way to select menu cells from a slave's database without having to resort to a programming language.

The procedure for browsing a slave's menu is to first request its column headings with the Get Column Headings command. This results in a list of menu cell references for the available column headings and their associated text. This can be used like a table of contents to see what the slave device holds. From this list, a particular column heading can be selected.

The Get Column Text command is then issued with the selected column reference used as the argument. This will result in a second list of menu cell references, for the visible menu cells in the specified column, along with their associated text. Some, if not all, of these cells will have an associated value, in which case the text will contain formatting codes to control how the value is to be displayed.

The values are then requested using the Get Column Values command. This results in a third list of menu cell references, for the given column number, along with their associated values. This list may not be as long as the second list, since some cells in the column may not have values. It will therefore be necessary to match the correct text with the correct value by associating the menu cell references in both lists. The list of menu cells in the specified column can then be shown complete with text and values as a list for the user to select.

When a menu cell is chosen, the full menu cell reference is fully specified for that item of data, and can be used as the argument for further Courier commands. It can be used with the Enter Setting Mode command to initiate the remote setting change operation, or it can be passed to a data polling routine which uses the Get Value command.

14. Text Formatting Characters

This section details the character sets, control codes and value formatting controls which Courier provides. It focuses on the effects text processing will have on a master control unit and hence details what facilities a slave device may make use of when presenting itself over the Courier communications link. This section does not discuss: the design of (remote) menus, nor the functionality required by a slave for its own (local) user interface.

No consideration is given in this section for the translation of mnemonics and other items of text which are contained within an implementation of Courier text formatting.

14.1 OVERVIEW OF TEXT FORMATTING IN COURIER

Courier provides three textual packet types: Courier text, password text and modem (control) text. These packet types can be used, by a slave device, for menu cell values. The Courier text packet is additionally used for menu cell (descriptive) text, indexed strings and the descriptive fields of event records.

Password text uses a subset of the Courier text character set, whilst the modem and Courier character sets differ significantly.

The Courier character set consists of the printable ASCII characters, plus accented characters to cover the requirements of the German, French and Spanish languages. It also contains a number of control characters (codes) which affect character positioning when the string is displayed.

The modem character set provides the complete set of ASCII characters (printable and non-printable) plus characters for controlling the serial communications link to the modem.

Master control units and slave devices must be capable of distinguishing between these character sets and managing the display of non-printable control codes by using either multiple character mnemonics or special, single character, icons.

When Courier text is used for a menu cell text it may also contain formatting characters which position and format the cells' value (should it have one). The combining of menu cell values with their menu cell text is controlled by a value format specifier embedded in the menu cell text. Courier borrows, from the C programming language, the *printf* argument format specifiers which begin with a '%'. However, not all the C format specifiers are implemented

and some new ones have been introduced; notably '%b' for binary numbers, '%k' for Courier numbers and '%t' for the date and time type.

In summary, the formatting of menu cell text with its menu cell value for display or printing requires character set mapping, control code processing and value formatting.

14.2 LINE LENGTHS

Menu cells will be displayed, by a master control unit, on a single line. Slave devices should be designed to use no more than 50 characters for the formatted display of each menu cell (on a single line). Masters will be designed to display at least 50 characters per slave menu cell. Whether a master truncates or otherwise a line longer than 50 characters is implementation dependent. However, for event record reports, the master will perform word wrapping at the limit of its output line length.

14.3 CHARACTER SETS

The three textual packet types provided by Courier of: Courier text, password text and modem control text, make use of individual character sets. Courier text uses the Courier character set. Password text uses a subset of the Courier character set and modem text uses the modem character set.

The Courier character set (Courier text) is used for menu cell text, index strings and event record text. A menu cell value may be Courier text, password text or modem text.

14.3.1 COURIER CHARACTER SET

The Courier character set is shown in Table 14-1 and consists of:

- Seven control codes in the character code range 1 to 31. (The 24 unused characters are reserved.) These codes do not have display characters, but affect the way the text is displayed; see Section 0 for their descriptions. They are shown in Table 14-1 as mnemonics, between square brackets '[...]'.
- The printable ASCII character set for all characters in the range 32-126.
- Extra characters in the range 127 to 163 which encompass the extra characters required for the French, German and Spanish languages.
- Character codes 164 to 255 are reserved for future enhancements.

Character zero is reserved for use as a string terminator, should a device require this. The string terminator should not appear in the Courier text packet, since the length is coded into the packet.

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
0		32		64	@	96	`	128	Á	160	Í	192	224
1		33	!	65	A	97	a	129	Ä	161	Ó	193	225
2		34	"	66	B	98	b	130	Ç	162	í	194	226
3		35	#	67	C	99	c	131	È	163	ó	195	227
4		36	$	68	D	100	d	132	É	164		196	228
5		37	%	69	E	101	e	133	Ê	165		197	229
6		38	&	70	F	102	f	134	Ë	166		198	230
7		39	'	71	G	103	g	135	Î	167		199	231
8		40	(	72	H	104	h	136	Ï	168		200	232
9	[T8]	41	)	73	I	105	i	137	Ñ	169		201	233
10	[LF]	42	*	74	J	106	j	138	Ô	170		202	234
11		43	+	75	K	107	k	139	Ö	171		203	235
12		44	,	76	L	108	l	140	Ù	172		204	236
13	[CR]	45	-	77	M	109	m	141	Û	173		205	237
14		46	.	78	N	110	n	142	Ü	174		206	238
15		47	/	79	O	111	o	143	ß	175		207	239
16	[Tn]	48	0	80	P	112	p	144	à	176		208	240
17		49	1	81	Q	113	q	145	á	177		209	241
18		50	2	82	R	114	r	146	ä	178		210	242
19		51	3	83	S	115	s	147	ç	179		211	243
20	[SNL]	52	4	84	T	116	t	148	è	180		212	244
21		53	5	85	U	117	u	149	é	181		213	245
22		54	6	86	V	118	v	150	ê	182		214	246
23		55	7	87	W	119	w	151	ë	183		215	247
24		56	8	88	X	120	x	152	î	184		216	248
25	[CRLF]	57	9	89	Y	121	y	153	ï	185		217	249
26		58	:	90	Z	122	z	154	ñ	186		218	250
27		59	;	91	[	123	{	155	ô	187		219	251
28		60	<	92	\	124	|	156	ö	188		220	252
29	[T16]	61	=	93	]	125	}	157	ù	189		221	253
30		62	>	94	^	126	~	158	û	190		222	254
31		63	?	95	_	127	À	159	ü	191		223	255

Table 14-1 Courier Character Set, see Table 14-4 for a description of the mnemonics in the range 1-31. (Blank characters above code 127 and below code 32 are reserved and currently unused.)

14.3.2 PASSWORD CHARACTER SET

The password character set is a sub-set of the Courier character set. The password character set contains characters 65 to 90 ('A' - 'Z').

Password text will have each character replaced by an asterix, '*', when it is sent by a slave device.

14.3.3 MODEM CHARACTER SET

The modem character set is shown in Table 14-3 and consists of:

- Characters 0 to 127 from the ASCII character set.
- Special control codes characters in the range 128 to 254.

Character codes 0 to 31 and 128 to 254 do not have a single display character and are therefore shown as mnemonics between curly braces '{...}'. These are summarised in Table 14-3.

The Courier modem character set is based on the Company's Asynchronous Modem Specification, document 9030 0002, although not all the control codes have been implemented, as they are inappropriate for a slave device.

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
0	{NUL}	32		64	@	96	`	128	{w1}	160	{d10}	192	{#}	224	
1	{SOH}	33	!	65	A	97	a	129	{w2}	161	{d20}	193		225	
2	{STX}	34	"	66	B	98	b	130	{w5}	162	{d50}	194		226	
3	{ETX}	35	#	67	C	99	c	131	{w10}	163	{d100}	195		227	
4	{EOT}	36	$	68	D	100	d	132		164		196		228	
5	{ENQ}	37	%	69	E	101	e	133		165		197		229	
6	{ACK}	38	&	70	F	102	f	134		166		198		230	
7	{BEL}	39	'	71	G	103	g	135		167		199		231	
8	{BS}	40	(	72	H	104	h	136		168		200		232	
9	{HT}	41	)	73	I	105	i	137		169		201		233	
10	{LF}	42	*	74	J	106	j	138		170		202		234	
11	{VT}	43	+	75	K	107	k	139		171		203		235	
12	{FF}	44	,	76	L	108	l	140		172		204		236	
13	{CR}	45	-	77	M	109	m	141		173		205		237	
14	{SO}	46	.	78	N	110	n	142		174		206		238	
15	{SI}	47	/	79	O	111	o	143		175	{d0}	207		239	
16	{DLE}	48	0	80	P	112	p	144	{wOK}	176	{-DTR}	208		240	
17	{DC1}	49	1	81	Q	113	q	145	{wVAL}	177	{+DTR}	209		241	
18	{DC2}	50	2	82	R	114	r	146	{wCON}	178	{-RTS}	210		242	
19	{DC3}	51	3	83	S	115	s	147	{wCNX}	179	{+RTS}	211		243	
20	{DC4}	52	4	84	T	116	t	148		180		212		244	
21	{NAK}	53	5	85	U	117	u	149		181		213		245	
22	{SYN}	54	6	86	V	118	v	150		182		214		246	
23	{ETB}	55	7	87	W	119	w	151		183		215		247	
24	{CAN}	56	8	88	X	120	x	152		184		216		248	
25	{EM}	57	9	89	Y	121	y	153		185		217		249	
26	{SUB}	58	:	90	Z	122	z	154		186		218		250	
27	{ESC}	59	;	91	[	123	{{}	155		187		219		251	
28	{FS}	60	<	92	\	124	|	156		188		220		252	
29	{GS}	61	=	93	]	125	{}}	157		189		221		253	
30	{RS}	62	>	94	^	126	~	158		190		222		254	
31	{US}	63	?	95	_	127	{DEL}	159		191		223		255	{ST}

Table 14-2 Modem control character set, see Table 14-3 for description of mnemonics. (Blank characters above code 127 are reserved and currently unused.)

Mnemonic	Description	Mnemonic	Description
{NUL}	Null character	{FS}	File Separator
{SOH}	Start of Header	{GS}	Group Separator
{STX}	Start of Text	{RS}	Record Separator
{ETX}	End of Text	{US}	Unit Separator
{EOT}	End of Transmission	{DEL}	Character 127; ASCII Delete
{ENQ}	Enquiry	{w1}	Wait for 1 second
{ACK}	Acknowledge	{w2}	Wait for 2 seconds
{BEL}	Bell	{w5}	Wait for 5 seconds
{BS}	Backspace	{w10}	Wait for 10 seconds
{HT}	Horizontal Tab	{wOK}	Wait for "OK" response
{LF}	Line Feed		(Hayes "AT" response)
{VT}	Vertical Tab	{wVAL}	Wait for "VAL" response
{FF}	Form Feed		(CCITT V25bis response)
{CR}	Carriage Return	{wCON}	Wait for "Connect ####" response
{SO}	Shift Out		(Hayes "AT" response)
{SI}	Shift In	{wCNX}	Wait for "CNX" response
{DLE}	Data Link Escape		(CCITT V25bis)
{DC1}	Device Control 1	{d0}	No delay between characters
{DC2}	Device Control 2, XON		(default)
{DC3}	Device Control 3	{d10}	10ms delay between characters
{DC4}	Device Control 4, XOFF	{d20}	20ms delay between characters
{NAK}	Negative Acknowledge	{d50}	50ms delay between characters
{SYN}	Synchronous Idle	{d100}	100ms delay between characters
{ETB}	End of Transmission Block	{-DTR}	Deactivate DTR
{CAN}	Cancel	{+DTR}	Activate DTR
{EM}	End of Medium	{-RTS}	Deactivate RTS
{SUB}	Substitute	{+RTS}	Activate RTS
{ESC}	Escape	{#}	Insert telephone number (place-
{{}	Replaces { from keyboard		holder)
{}}	Replaces } from keyboard	{ST}	String terminator

Table 14-3 Description of modem control and ASCII mnemonics.

Notes - As the baud rate used is not fixed, the baud rate reported in the connect message will be variable and is shown in the above table as "####". It will not necessarily be four characters in size. Control switches for DTR and RTS are included to allow the lines to be toggled during the setup or dialling procedures. It should be noted that many modems have restricted command buffers (40 characters is common) and a long command string should be broken (by using the modem control characters) to give the modem time to process the data.

14.4 SPECIAL CHARACTERS

There are three classes of special characters which affect text processing: Value formatting, Courier text positioning, and modem control codes.

14.4.1 ORDER OF EVALUATION

For a layered approach to text processing, text should be evaluated in the following order:

1. **Substitution:** Modem control codes will be expanded to their mnemonic forms. Substitutions must occur first, since this is dependent on the type of the text.

2. **Value formatting:** Menu cell text will be combined (formatted) with the menu cell's value. This involves processing any value format specifications (beginning with '%') and performing the specified insertion of the argument; see Section 14.5.
3. **Control codes:** Character positioning control codes will be processed.

Items 1 and 2 must be processed before item 3, because these items can affect the character positions, which are required in order to calculate the number of spaces a tab control character is to be replaced by.

A single layered approach to text processing may evaluate the character codes as they are encountered.

Care should be taken with the [Tn] code from the Courier character set, since this is a two character code where the second character is interpreted as the tab position - this character should not have its value altered by the substitution and value formatting processes.

14.4.2 CHARACTER POSITIONING CONTROL CODES

Table 14-4 summarises the control codes in the Courier character set, which are discussed more fully in the following subsections.

Char.	Mnemonic	Name	Description
9	[T8]	Tab 8	Tabulate to the next 8th character position.
10	[LF]	Line feed	Ignored, except in extended event information where it moves the *output cursor* down one line.
13	[CR]	Carriage return	Ignored, except in extended event information where it moves the *output cursor* to the beginning of the current line.
16	[Tn]	Tab n	Tabulate to character position n.
20	[SNL]	Auto new line suppression	Ignored, except in extended event information where it suppresses the automatic new line at the end of a menu cell display.
25	[CRLF]	Carriage return & line feed combined	Ignored, except in extended event information where it moves the *output cursor* to the start of the next line.
29	[T16]	Tab 16	Tabulate to the next 16th character position.

Table 14-4 Character positioning control codes in the Courier character set.

14.4.2.1 Tab 8, Tab 16 and Tab n

The *tab 8* form of tabulation causes space characters to be output until the character position is wholly divisible by 8. The *tab 16* form of tabulation causes space characters to be output until the character position is wholly divisible by 16.

The *tab n* form tabulates to an exact character position which is specified by the next character.[5] Character positions begin at zero. Thus "x<*tab 10*>y" results in "x<*9 spaces*>y"; the 'y' being in character position 10 and is the 11th character in the output. A *tab n* code

[5] The most significant bit (bit 7) of this tab position character should be ignored. A slave device should set this bit to avoid position codes, such as tab 38, being miss-interpreted as a value format specification; code 38 being a '%' character. ('%' format specifiers being processed before tabulation codes in the layered text processing model.)

encountered after character position *n* has been exceeded is ignored (i.e. back tabulation is not supported).

14.4.2.2 Line Feed, Carriage Return and Carriage Return & Line Feed Combined

These three control codes are only valid in type 3 extended event record information and will be ignored in the normal presentation of menu cells. These control codes will affect the presentation of the extended event information, when it is output as a report.

When the *line feed*, *carriage return* and *carriage return & line feed combined* codes are encountered, in a type 3 extended event report, they will cause the *output cursor* to be repositioned, as described in Table 14-4.

14.4.2.3 Auto new line suppression

This control code is only valid in type 3 extended event record information and will be ignored in the normal presentation of menu cells.

When encountered, in a type 3 extended event report, this control code suppresses the automatic new line at the end of a menu cell display. The intention being to achieve multiple values on a single line. For example:

Without the use of the *new line suppression* code, a type 3 extended event record would have to be formatted with a single value per line in the report:

```
Tue 1994 Aug 23   16:45:14.599
Fault Record:
Protection: 87G Generator Differential ABC, 51V Overcurrent A
Relay Output Status: Trip CB, Overcurrent Trip, Gen Diff Trip
Logic Input Status:  No operation
Scheme Output: Enabled
Active Setting Group: 1
Ia:                 1.000 A
Ib:                 1.000 A
Ic:                 1.000 A
Ia-Diff:              998mA
Ib-Diff:            1.000 A
Ic-Diff:              997mA
Ia-Mean Bias:         500mA
Ib-Mean Bias:         500mA
Ic-Mean Bias:         500mA
I2:                     0 A
I-Residual:         1.003 A
Ie:                 1.002 A
Vab:                110.1 V
Vbc:                    0 V
Vca:                110.3 V
Ve:                  0.43 V
Active Power   Aph: 55.29 W
Reactive Power Aph:-31.60 VAr
Phase Angle    Aph:  29.9 deg
Frequency:          44.49 Hz
```

Figure 14-1 Example of an extended event record with one value per line.

The application of the *new line suppression* code allows the slave device to use a different layout which reduces the (vertical) length of the report:

```
Tue 1994 Aug 23   16:45:14.599
Fault Record:
Protection: 87G Generator Differential ABC, 51V Overcurrent A
Relay Output Status: Trip CB, Overcurrent Trip, Gen Diff Trip
Logic Input Status:  No operation
Scheme Output: Enabled
Active Setting Group: 1
               Ia      Ib      Ic
Phase:     1.000 A 1.000 A 1.000 A
Diff:        998mA 1.000 A   997mA
Mean Bias:   500mA   500mA   500mA
I2 =     0 A I-Residual = 1.003 A Ie = 1.002 A
     Vab      Vbc      Vca       Ve
 110.1 V      0 V  110.3 V   0.43 V
     Active Power Reactive Power Phase Angle
Aph:      55.29 W     -31.60 VAr    29.9 deg
Frequency:          44.49 Hz
```

Figure 14-2 Example of an extended event record employing the suppress new line control code to achieve multiple values per line.

Slave devices using this control code will design their formatted output to fit maximum line lengths of 50 characters, as detailed in Section 14.2. Master control units will work with line lengths appropriate to the output device, and will place text on the next line, rather than truncate it.

Note that the descriptive text in the event record packet 3 may not contain the *new line suppression* code.

14.4.3 RESERVED CHARACTERS

A character which is reserved, in a character set, will be displayed as a period, '.'.

14.5 VALUE FORMATTING

Menu cell text may or may not contain a formatting specification for the cell's value. Cells which do not supply a value argument will not contain value formatting in their text, however, text processing will be *graceful* if this situation occurs. A cell may supply a value and have no formatting in its text for it, in which case the value is ignored for display purposes. Only menu cell text may contain value formatting.

Format specifications begin with a '%' place holder and are read left to right. Characters following the '%' will be interpreted to find the type of the value and the parameters controlling its formatting. These characters are then replaced by the formatted text representation of the value. If the initial '%' is followed by a character which has no meaning as a format field, the character is used as the replacement text. An example is the percent-sign character which can only be obtained with the character sequence '%%'.

The following sub-sections detail the various value types and format controls supported by Courier.

14.5.1 LOCAL CONVENTIONS

On occasion a formatting action will be specified as being defined by *local conventions*. Local conventions may be fixed or settable by a user, for each master control unit and slave device, and, as such, represent *local* preferences for particular aspects of display formatting.

Table 14-5 details the local conventions or preferences which can affect the formatting of values for display purposes. When a device does not support user control over these conventions, a default value is suggested.

Subject	Definable Action	Default
Date[6]	Order of day, month and year	year month day
	Leading zeros on day of month (1 vs 01)	No
	Indicate century of year (96 vs 1996)	Yes
	Separator characters[7]	' ' (blank)
	Abbreviated month (Jan vs January)	Yes
	Abbreviated day of week (Mon vs Monday	Yes
Time	12 or 24 hour notation	24 hour
	Morning & afternoon indicators (12hr clock only)	'am' & 'pm'
	Time Zone	(none)
	Leading zeros (1 vs 01)	Yes
	Separator - between hours, minutes & seconds	':'
	Separator - between seconds and milliseconds[8]	'.'
Number format	Decimal separator	'.'
	Thousands separator	(none)
	Decimal places	2
	Leading zero before decimal place (.1 vs 0.1)	Yes

Table 14-5 Local conventions which can affect the formatting of values

14.5.2 FORMAT SPECIFICATION FIELDS

A format specification, which consists of optional[9] and required fields, has the following form:

```
%[flags][width][.precision][l]type
```

Each field of the format specification is a single character or a number signifying a particular format option. The simplest format specification contains only the percent-sign and a *type* character (for example, %s). The optional fields, which appear before the type character, control other aspects of the formatting. The fields are described in the following table:

[6] The MS Windows OS provides short and long date locales. The preferred format is the long date localisation.

[7] On some systems it may be possible to specify different separators for each part of the date.

[8] This can be assumed to be the same as the decimal separator used by the number format.

[9] Optional fields are show in square brackets; [...].

Field	Description
type	Required character which determines the basis for the formatting of the value argument. Format type characters are summarised in Section 14.5.3 and discussed more fully in Section 14.5.7.
flags	Optional character or characters which control justification and the printing of prefixes. More than one flag can appear in a format specification. See Section 14.5.4.
width	Optional number which specifies the minimum number of characters output. Ignored for the Courier number and date & time formats (*k* & *t*).
precision	Optional number which specifies the maximum number of characters printed for all or part of the output field, or minimum number of digits printed for integer values. Ignored for the Courier number and date & time formats (*k* & *t*).
l (ell)	Optional long prefix: Used with the integer types (*b, d, u* & *x*) to indicate 32 bit numbers rather than 16 bit numbers. Used with the date & time type (*t*) to indicate a preference for a longer output format, see Section 14.5.7.5. Used with Courier numbers (*k*) to indicate a 6 byte rather than 4 byte number is to be formatted. Ignored for other format types (*c, f* & *s*).

Table 14-6 Format specification fields.

Format specifiers which are not defined or have no meaning for a given type will be ignored.

14.5.3 SUMMARY OF TYPE FIELD CHARACTERS

The *type* character is the only required format field; it appears after any optional format fields. The *type* character determines the basis for interpreting the value argument and the effect of the optional format fields. Courier supports five basic types: integer, floating point, Courier number, text and data & time. Some of these basic types have more than one format *type* character. The type characters are summarised below and discussed more fully in Section 14.5.7.

Value formatting assumes the data type of the value is viable for the format type character - see Table 14-8. The effect of mismatching format type characters with the type of the supplied data is undefined.[10,11]

[10] This also applies to the 'l' (ell) type modifier: for example, a 32bit integer format must use the ell type prefix, e.g. '%lb'.

[11] Value formatting will typically process the supplied data as though it is viable for the format type character. For example, the integer type format will always take the first two bytes of the supplied value argument, irrespective of the underlieing type of the data. If the data is shorter than the number of bytes required by the format type, a buffer overrun will occur and bytes of arbitrary value will be read after the end of the data.

Character	Value interpretation for display
b	Unsigned Binary integer.
c	Single character.
d	Signed decimal integer.
f	Signed floating point value. (This is a single precision IEEE754 floating point number.)
k	Signed Courier number.
s	Character string of known length.[12]
t	Courier date & time value.
u	Unsigned decimal integer.
x	Unsigned hexadecimal integer, using 'ABCDEF'.

Table 14-7 Format value type summary.

Courier packet		Viable value format	
Type code	**Data type**	**Type classification**	**Type character**
18h	Courier text	String	s, (c)
1Ch	Password text	String	s, (c)
20h	Binary flags (8/16 bits)	Integer (unsigned)	b, (u, x)
20h	Binary flags (32 bits)	Integer (unsigned long)	lb, (lu, lx)
24h	Unsigned integer (8/16 bits)	Integer (unsigned)	u, x, (b)
24h	Unsigned integer (32 bits)	Integer (unsigned long)	lu, lx, (lb)
28h	Signed integer (8/16 bits)	Integer (signed)	d, (b, u, x)
28h	Signed integer (32 bits)	Integer (signed long)	ld, (lb, lu, lx)
2Ch	Courier number	Courier number	k
2Ch	Extended Courier number	Courier number (long)	lk
34h	IEEE floating point number	Float	f
38h	Millisecond timer count	Integer (unsigned long)	lu, (lb, lx)
3Ch	Date & time	Date & time	t, lt
44h	Menu location	Integer (unsigned)	u, x (b)
50h	String index	String	s, (c)
68h	Modem Control Strings	String	s, (c)

Table 14-8 Viable value formatting types for (displayable) Courier packet types. (Type characters in brackets are viable but *generally* not appropriate.)

14.5.4 SUMMARY OF FLAG FIELD CHARACTERS

The first optional field of the format specification is *flag*, which consists of one or more flag characters. A flag character (or directive) affects justification and the output padding and legends. The affect of each flag character is summarised below and discussed more fully in Section 14.5.7.

[12] The transport packet for text strings, in Courier, provides the length information.

Flag	Meaning	Default if omitted
-	Left justify the result within the given field width. Applicable format types: *b, c, d, f, s, u, x*.	Right justify.
0	If *width* is prefixed with a 0, zeros are added until the minimum width is reached. If 0 and - appear, the 0 is ignored. Applicable format types: *b, d, f, u, x*.	No padding.
#	Used with the date & time type to indicate a preference for the date field to be prefixed with 'Date:' and the time field with 'Time:', see Section 14.5.7.5. Applicable format type: *t*.	The date and time fields are output with no prefixes.

Table 14-9 Format flag summary.

14.5.5 WIDTH SPECIFICATION

The second option field of the format specification is the width specification. The *width* argument is a non-negative decimal integer controlling the minimum number of characters printed. If the number of characters in the output value is less than the specified width, blanks are added until the minimum width is reached. Blanks are added to either the left or the right of the value, depending on whether the '-' flag (for left justification) is specified. If *width* is prefixed with 0 and the number (integer or float) is right justified (default condition), zeros are added until the minimum width is reached.

The width specification never causes a value to be truncated. If the number of characters in the output value is greater than the specified width, all the characters are output (subject to the precision specification).

If *width* is not specified it defaults to a value of zero.

The width specification is ignored for the Courier number and date & time formats (*k* & *t*).

14.5.6 PRECISION SPECIFICATION

The third optional field of the format specification is the precision specification. The *precision* argument is a non-negative decimal integer, preceded by a period ('.'), which specifies the number of characters to be printed or the number of digits after the decimal place. Unlike the width specification, the precision specification can cause truncation of the output value, or rounding in the case of the float format.

The presence of the precision period ('.') with no precision value results in the default action for no precision field specification.

The table below summarises the affect of *precision* on each *type* format.

Format Type	Effect of precision	Default if precision omitted
Integer: b, d, u, x	The precision specifies the minimum number of digits to be printed. If the number of digits in the argument is less than precision, the output is padded on the left with zeros. The value is not truncated when the number of digits exceeds precision. A zero value for precision is ignored and precision is used as though it was set to 1.	Default precision is 1.
Float: f	The precision value specifies the number of digits to be output after the decimal place. The value is rounded to the appropriate number of digits. If a decimal place appears and the number is less than unity, local display conventions determine whether a zero prefix is used. If precision is zero then no decimal place appears.	Default precision is determined by the local conventions.
String: s	The precision specifies the maximum number of characters to be output. Characters in excess of precision are truncated. If there are less characters than precision, *precision* has no effect. A precision of zero causes all characters to be output.	Default precision is 0; all characters are output.

Table 14-10 Effect of the format precision on the value types.

The precision specification is ignored for the character (*c*), Courier number (*k*) and date & time formats (*t*).

14.5.7 FORMATS BY TYPE

14.5.7.1 Integer

There are four integer formats: signed and unsigned decimal (*d* & *u*), unsigned binary (*b*) and unsigned hexadecimal (*x*). All integer formats assume the value to be 16 bits.[13] If the *type* character is prefixed by 'l' (ell), then the value is assumed to be 32 bits.[14]

The effect of the *flags*, *width* and *precision* format specification fields are described in Sections 14.5.4, 14.5.5, and 14.5.6, and is independent of the base (2, 10 or 16) of the number format.

Examples:	Value argument	Result[15]
%u	35	35
%b	35	100011
%x	78	4E
%.4x	78	004E
%.8b	35	00100011
%8b	35	ΔΔ100011
%-8b	35	100011ΔΔ
%16.14b	2189	ΔΔ00100010001101

Table 14-11 Examples of integer formatting.

[13] 8 bit values should be zero or sign extended to 16 bits, as appropriate.

[14] Three and more than 4 byte integers are not supported.

[15] 'Δ' has have been used to show were blanks should be output.

14.5.7.2 Character And String

There are three character sets in Courier: the Courier character set, the password character set and the modem character set (see Section 14.3). Character and string formatting must be capable of combining menu cell text with a menu cell text value from a different character set. Control codes, as defined by Section 14.4.2, in text (value) arguments are not allowed - they will be treated as reserved characters; see Section 14.4.3.

The effect of the *flags*, *width* and *precision* format specification fields are described in Sections 14.5.4, 14.5.5, and 14.5.6.

14.5.7.3 Floating Point

The floating point format causes the supplied value to be displayed as a real number; that is as a value with a decimal place.

The effect of the *flags*, *width* and *precision* format specification fields are described in Sections 14.5.4, 14.5.5, and 14.5.6.

The display of floating point numbers is also influenced by the local display conventions, as noted in Section 14.5.1. These affect: the decimal place and *thousands* separator characters, whether a number beginning with a decimal place should be zero prefixed (e.g. 0.1 or just .1) and the default number of decimal places.

14.5.7.4 Courier Number

14.5.7.4.1 Data type summary

A Courier number packet consists of four fields: a *mantissa*, a *sign*, an *exponent* and a *units* field. The *mantissa* is an unsigned integer which is either 15 bits or 31 bits long. The *sign* is 1 bit long and the *exponent* and *units* fields are each 8 bits long. A Courier number with a 31 bit mantissa is called an *Extended Courier Number*.

A Courier number packet is 4 bytes long and an Extended Courier number packet is 6 bytes long.

The 15 bit *mantissa* has a valid range of 0 to 9999 and the 31 bit *mantissa* has a valid range of 0 to 999,999,999.

The *sign* bit is 0 for positive mantissas and 1 for negative mantissas.

The *exponent* is an unsigned integer with a bias of 126 and represents powers of 10 which the *mantissa* should be raised to. (A power of 10^0 is represented with an exponent value of 126.)

The *units* of a number are coded as per Table 14-13.

The value of the Courier Number type can therefore be expressed in the following form:

$$\text{Value} = (1-2\times Sign)\times(Mantissa)\times 10^{E-126}\ Units.$$

Courier numbers will use the multiplier symbols defined in Table 14-12, when they are valid for the particular unit type. Thus the valid range for such a Courier number is: (±) 1×10^{-18} to $9.999\times10^{+18}$ and for an extended Courier number: (±) 1×10^{-18} to $9.99999999\times10^{+18}$.

For Courier numbers with units which do not allow the use of a multiplier symbol, the valid range is reduced to: (±) 0.001×10^{0} to 9999×10^{0} and for an extended Courier number: (±) 0.00000001×10^{0} to 999999999×10^{0}.

14.5.7.4.2 Text formatting

The standard display format for Courier numbers is:

bsdddddmuuu

Where:

b is a blank padding field - see *ddddd* description.

s is the sign field and is either blank, for positive numbers or '-' for negative numbers.

ddddd is the digits field and consists of up to 4 decimal digits and a possible decimal place. If the result is less than 5 characters wide blanks are inserted in to the *b* field, to the left of the *s* field, until the *b* + *ddddd* fields are 5 characters wide.

m is the multiplier symbol, as per Table 14-12.

uuu is the units symbol, which is of arbitrary length; although 1 to 3 characters is preferred. The *uuu* field is, by default, the only field of variable length in the Courier number display format. Units currently supported are listed in Table 14-13.

If the 'l' (ell) format flag appears the supplied value is taken to be a 6 byte extended Courier number, rather than the standard 4 byte number. In this case, the *ddddd* field is extended to 10 characters, consisting of 9 digits and a possible decimal place, and blank padding in the *b* field is adjusted to maintain this.

There are no other format options for Courier numbers and any supplied will be ignored.

The number of significant digits supplied in a Courier number is to be respected and maintained, where possible. The *mantissa* component of a Courier number is designed to hold between 1 and 4 significant digits (or nine in the case of the extended type).

The decimal place character is defined by the local display conventions, see Section 14.5.1. A *thousands* separator is not used and local display conventions for the default number of decimal places are also ignored. Local conventions are used to determine whether a number beginning with a decimal place should be zero prefixed (e.g. 0.1 or just .1).

The value of the *exponent* is used to determine the position of a possible decimal place, such that the exponent value is adjusted to be a whole multiple of 3. This facilitates the use of (SI) multiplier symbols (Table 14-12). Exponent values are adjusted to avoid a decimal place after the last digit, whilst attempting to maintain the number of significant digits in the mantissa, before the next (more positive) power is selected. e.g. $12e^{2}=1.2k$ and $12e^{1}=0.12k$, but $1200e^{0}=1.200k$. Similarly, $12e^{-5}=0.12m$ and $1200e^{-6}=1.200m$. The displayed number will

demonstrate a predilection for a more positive multiplier value; hence 1.200k rather than 1200 and 1.200m rather than 1200µ.

The use of multiplier symbols in the *m* field is only valid for certain units. Table 14-13 indicates which units may have a multiplier prefix. When the *m* field is invalid for the unit it is removed and the mantissa will be displayed as a real number (in the range 0.001 to 9999).[16] Since the real number is limited to the size of the *ddddd* field, slave devices will use this as the practical range of values possible with these particular units.

The units currently supported by Courier numbers are listed in Table 14-13.[17] Should a unit code be encountered, which is not a defined unit, the blank unit (code 15) will be used.[18]

Multiplier Value	Name	Symbol
10^{-18}	Atto	a
10^{-15}	Femto	f
10^{-12}	Pico	p
10^{-9}	Nano	n
10^{-6}	Micro	µ or u
10^{-3}	Milli	m
10^{0}	-	' ' (blank)
10^{3}	Kilo	k
10^{6}	Mega	M
10^{9}	Giga	G
10^{12}	Tera	T
10^{15}	Peta	P
10^{18}	Exa	E

Table 14-12 Courier number (SI) multiplier symbols.

[16] If local conventions specify no leading zero then the display range can be increased to .0001 to 9999, but a slave should not assume this; i.e. the specified range is 0.001 to 9999.

[17] Provision should be made to easily extend this list, as new units are added.

[18] The intention is to offer graceful degradation when the intended units are not known. It is therefore important that the full dynamic range is supported; the scientific exponent form (discussed at the end of this subsection) is one method of providing this. - The use of an SI multiplier symbol with no (SI) unit is to be discouraged.

Unit code value	Quantity	Unit	Display symbol	Multiplier field used
0	Current	Amps	A	✓
1	Voltage	Volts	V	✓
2	Angle	Degrees	deg	✓
3	Impedance	Ohms	Ohm or Ω	✓
4	Power	Watts	W	✓
5	Apparent Power	VA	VA	✓
6	Reactive Power	VAr	VAr	✓
7	Length	Metres	m	✓
8	Time (interval)	Seconds	s	✓
9	Ratio	None	:1	×
10	Temperature	Degrees Celsius	°C or Cel	✓
11	Frequency	Hertz (cycles/s)	Hz	✓
12	Percentage	None	%	×
13	Per Unit Value	Per Units	PU	×
14	Current Squared	Amps Squared	A^2	✓
15	None	None		×
16	Energy	Watt Hours	Wh	✓
17	Apparent Energy	VAh	VAh	✓
18	Reactive Energy	VArh	VArh	✓
19	Time (interval)	Minutes	min	✓
20	Inverse Ohms	Mho	mho	✓

Table 14-13 Courier number units; their display symbol and which may have a multiplier prefix symbol (see Table 14-12).

Examples:	Value argument	Result[19]
%k	$10e^2$A	ΔΔΔ1.0kA
%k	$-34e^{-5}$ W	Δ-0.34mW
%k	$1e^{-2}$:1	ΔΔ0.01:1
%k	$7687e^3$ W	Δ7.687MW
%k	$7687e^4$ W	Δ76.87MW
%k	$1e^0$ %	ΔΔΔΔΔ1%

Table 14-14 Examples of Courier number formatting.

14.5.7.4.3 Formatting numbers outside the displayable range

Courier does not define the action to be taken by a master control unit when a Courier number is encountered which exceeds the range of the available multiplier symbols, or the display range for unit types with no multiplier symbol field. Since slave devices will be designed to work with the range of the unit type selected, this situation should never occur. However, this situation may occur due to:

- The use of a new unit type which the master control unit is not aware of.
- The slave is sending illegal values (presumably due to a software error).

[19] 'Δ' has been used to show were blanks should be output. The local display conventions for a period decimal separator and a leading zero before the decimal place are assumed in these example outputs.

Master control units will therefore implement an appropriate scheme to deal with this problem. One *suggestion* is to use *scientific* notation and introduce an exponent 'e'.

Suggestion: An exponent field can be introduced as a replacement for the *m* field, in the display of a Courier number, when the multiplier value is outside the range of the symbols in Table 14-12, or the unit type precludes the use of such symbols and the number cannot be displayed without loss of significant digits. Exponent notation takes the form '*esd*' where: e is literal and indicates the exponent follows, *s* is a possible minus sign, '-', and *d* is one or more digits representing the exponent value. e.g. the ratio $145e^{-5}$ cannot be displayed as 0.00145:1 without loss of digits in the *ddddd* field. However it may be displayed as 1.45e-3:1.

14.5.7.5 Date & Time

The format for the display of date and time (Date & Time) information is largely governed by the local Date & Time display conventions provided by the host operating system (e.g. Windows 3.1), although the date is always output before the time.[20]

Two format controls are provided by the hash '#' and ell 'l' flags. The hash flag requests a legend for the Date & Time and the ell requests a display format which includes (possibly) the day of week and (definitely) the milliseconds. These flags may be ignored in which case the milliseconds will always be output.

The internal format of the Date & Time data type is such that a number of the fields can have illegal values. The values of all fields will be checked before conversion to the display string is under taken. Note that the day of week field can be set to zero when the device, supplying the data, has not calculated the actual day of week. In this case the day of week will have to be calculated, when it is required for display. The data type also includes two flag bits for *invalid time* and *summer time*.

If the invalid bit is set the date and time data will be shown between round brackets after the word 'Invalid'[21] - see examples below.

If the summer time flag is set, the suffix 'st' is appended to the time output. (The 'st' flag will follow any time suffices defined by the operating system local conventions, e.g. am/pm, GMT, etc.)

[20] Devices unable to determine what the local display conventions are should use the international form of: *[day-of-week] year month day hour:minute:seconds.milliseconds*. Where the 'day-of-week' is optionally shown and is normally used when the longer display format is requested.

[21] The invalid time bit can have a number of interpretations: there is no Date & Time data, the Date & Time data is not accurate, etc. Since the former interpretation hints at the Date & Time data not being displayed, whilst the later interpretation still requires its display, the Date & Time data will always be output.

Examples:	**Value argument**	**Result[22]**
%t	'96 Jul 1 14:59:37.345	1 July, 1996 14:59:37
%lt	'96 Jul 1 14:59:37.345	Monday July 1, 1996 14:59:37.345
%#t	'96 Jul 1 14:59:37.345	Date: 1 July, 1996 Time: 14:59:37
%#lt	'96 Jul 1 14:59:37.345	Date: Monday July 1, 1996 Time: 14:59:37.345
%t	invalid flag set, '96 Jul 1 14:59:37.345	Invalid (1 July, 1996 14:59:37.345)
%t	summer time flag set, '96 Jul 1 14:59:37.345	1 July, 1996 14:59:37 st
%t	one or more fields out of range	Illegal time value[23]

Table 14-15 Examples of date and time formatting.

[22] UK time and date conventions are used in the example display output.

[23] It may be useful to output all the field values between brackets for diagnostic purposes.

AMENDMENTS

Issue	Date	Changes
A	06/05/93	First Issue.
B	20/05/93	p44 control status word -> plant status word.
C	17/03/94	Corrections & clarifications.
	03/06/94	Tables used for formatting.
	25/08/94	Command indices added.
	12/04/95	Added time formatting.
D	20/02/97	Revised according to CUG annotations.
		Added Set Value, Get column limits & block transfers to a device.
		Added multiple password implementation.
E	02/02/98	Clarified IEC870 Date & Time, use of IV bit.
		Clarified selective disturbance record extraction.
		Clarified binary block transfers.
		Added Hz/s and V/Hz Courier units.